Universiteit Leiden

ICT in Business and the Public Sector

A CRITERIA DRIVEN METHOD FOR SELECTING INFORMATION EXTRACTION TOOLING FOR DATA LAKE OPTIMIZATION

Name: Ahmed Lachal
Student-no: s2368609

Date: 24/11/2020

1st supervisor: Guus Ramackers
2nd supervisor: Bas Kruiswijk

MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

ABSTRACT

The application of big data forms a challenge for many organisations, given the variety, velocity, and veracity of the data. Furthermore, a diversity of data sources often results in information silos that are hard to access.

Data lake systems have been proposed to store and manage data from different sources, e.g., unstructured, semi-structured, and structured data in one place.

However, simply storing all data in a data lake without effective metadata management and information extraction tooling only results in the creation of a “data swamp”.

Evaluation and selection of the appropriate set of data lake tooling is a complex decision-making process that involves multiple (sometimes conflicting) criteria and differing tool sets. Furthermore, this process needs to take into account that each organisation has specific requirements and varying knowledge levels. To address this issue, this research proposes a criteria driven method for selecting the appropriate information extraction tooling, which can be used to generate a solution that is tailored to an organisation’s specific needs.

This study describes the different models, and commercial and open-source tooling available for information abstraction from data lakes, and discusses each of their advantages and disadvantages. It serves to illustrate how different information extraction tools can be used to handle different situations based on user criteria. The result of this research is an automated solution that enables the user to input organisation specific criteria, and subsequently generates a priority based list of the most appropriate toolsets. The application of this method has been verified at five large-scale companies in the Netherlands.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

First and foremost, praises and thanks to God, the Almighty, for His blessings throughout the research to complete it successfully during covid-19.

I would like to express my sincere gratitude to everyone who helped and guided me during the research.

Many thanks to the first supervisor, Guus Ramackers, and the second supervisor, Bas Kruiswijk. Also, I would like to thank my managers Marleen Blok and Gijs de Witte, and my buddies Carla Parilo and Dennis van Herk for giving me advice and input during and even after the research. Also, a big thanks to all the companies that helped execute the validation and opened their doors to execute the interviews, especially Ahold Delhaize, Jumbo, Plus, Etos, and Airbus. Finally, I would like to thank my parents and my friends, who helped me stay motivated writing the thesis.

1. INTRODUCTION

1.1 Current state

At this moment in time, organizations are enthusiastic about data lake implementations. So does Koninklijke Ahold Delhaize is referred to as KAD, which generates approximately five terabytes of data each month. Gartner believes that 70 percent of mature organizations will have more data flowing from data lakes than from data warehouses in the upcoming years.

According to new research, the global data lake market will continue growing (Market Research Future, 2020).

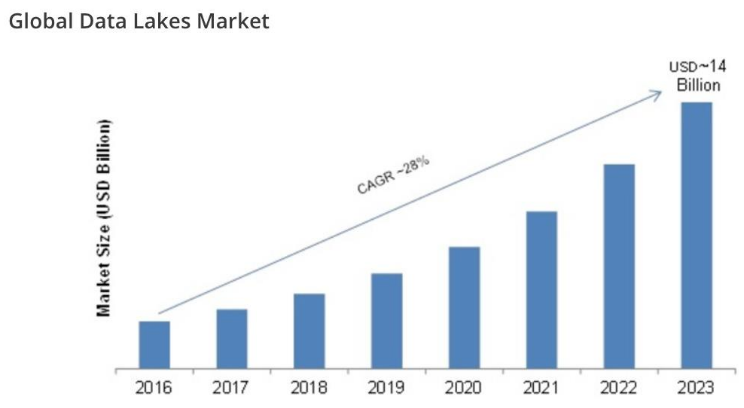


Figure 1: Data lake market size by 2023 (Market Research Future, 2020)

Data lakes support unknown data (i.e., structured, unstructured, and semi-structured data), low-cost storage, and easy landing data in the data lake without pre-processing.

Furthermore, data lakes support fast data coming from IoT devices. Data lakes often come with new tools and services that need to be understood. Investments need to be made in skills and transition from data warehouses to data lakes. Data lakes can easily land data, but data lakes use different query engines than data warehouses to query the data.

Wherein data warehouses support relational data from transactional systems and operational databases (i.e., pre-defined schema, structured data). Data warehouses are a well-established and proven solution, and SQL server solutions are widely available. Such warehouses also deliver good performance, partially due to its structure. A data warehouse has a written schema, and therefore SQL servers and other data warehouse technologies already understand its structure. However, the storage costs are more expensive than data lake storage. Although there is much value in having a model, it also takes time to have the development team support getting data into that model. Typically, the development of Extract Transform Load (ETL) processes creates latency because it has high costs. Organizations might only store specific amounts of data in the data warehouse, which will result in limited exploration.

The gap

One gap in the literature is that no information is available about the combination of tooling for Data Lake purposes in one method. It means there is a lack of knowledge of query languages, ETL tooling, and visualization tooling in one method.

Furthermore, no information is available about commonly used architectures or frameworks and criteria associated with it. So, while there has been much research on the different aspects individually, as of today no researchers have considered the combination.

This research intends to fill this gap. This gap will be filled by research on the query languages, ETL tooling, and visualization tooling, including advantages and disadvantages, followed by a method that will connect the different parts. Furthermore, will the method be validated and tested at different organizations.

1.2 Objectives

While data lake adoption is growing, the difficulties of this technology are growing as well. Data lakes are not working well for each company because of data lake adoption failures, high expectations, and insufficient preparations.
This research study aims to develop a method that helps organizations assess and choose their data lake architecture. After the literature review, a solution will be developed, which will be a selection method.

The deliverables will be as follows:

- List of criteria for the tooling
- Top 3 query languages (dynamically specified for each situation)
- Top 3 ETL tooling (dynamically specified for each situation)
- Top 3 visualization tooling (dynamically specified for each situation)
- Limitations
- Future work

It will be shown which techniques and tools can increase data accessibility and tackle common failures, as discussed in the next chapter. This research will provide insights into techniques and tools to address current issues and improve the accessibility of data lakes in general. The created method will be evaluated and verified at KAD and, more specific at Albert Heijn. Furthermore, will the solution be evaluated at Etos, Jumbo, Plus, and AirbusDS. These data lake problems are suitable for these companies and suitable for similar situations in the retail industry. Look for instance to Kruidvat, Bol.com, Lidl or COOP; these retail companies collect a massive amount of data, so these failures and solutions could also be suitable for them. The research is focused on enterprise organization. Furthermore, will this research not be limited to Microsoft tooling but rather take another tooling available in the industry into account.

1.3 Research question

1.3.1 Main question

“What method is best suitable for selecting the appropriate tools to rapidly and *efficiently extract data from the data lake?”*

1.3.2 Sub-questions

Which criteria are essential for rapid and efficient data extraction?

Which query language will enable rapid and efficient data extraction?

Which ETL tool will enable rapid and efficient data extraction?

Which visualization tool will enable rapid and efficient data extraction?

What is the process/mechanism for selecting the appropriate tooling?

These questions are essential because the data stored in data lakes are difficult to find and extract. Often not all the data is stored in a data lake. Data is still stored in separate relational database management systems, but the transition to data lake usage increases. It makes it a bottleneck in the organizational network because analysts are waiting for data, but IT cannot provide the data, and therefore data analysts try to find a workaround.

The introduction of data lakes also needs control and maturity levels, but most companies start using data lakes without real knowledge about their capabilities and preparations (Woods, 2018, pp. 1–3). The list below shows the common failures of data lakes:

- Data lakes become data swamps.
- Data never put into production.
- Failing to gain added value.
- A lack of business impact
- A lack of data governance

- A lack of data quality

The failures mentioned before shows how companies need to rethink their processes and data architecture to fully utilize data lakes (Woods, 2018, pp. 1–3).

1.4 Method

The first step in this research is to gather as much information as possible about the subject. An extensive literature review incl. focus on existing frameworks or architectures will focus on the subject. Next, the method to select the appropriate tools to extract data rapidly and efficiently from the data lake will be built. Finally, a case study through a qualitative research method using interviews and survey will be used for validation. Interviewing people who are using this process and software will provide insights into their world, opinions, and thoughts. For example:

- Behaviors: what a person has done or is doing.
- Opinions/values: what a person thinks about the topic.
- Knowledge: to get facts about the topic.
- Background/demographics: standard background questions, such as age, education.

This approach will use a standardized, open-ended interview, wherein open-ended questions are asked to all interviewees, which facilitates faster interviews that can be more easily analyzed and compared.

2. BACKGROUND AND RELATED WORK

2.1 What is a data lake?

General information

Research executed by Madera & Laurent (2016, pp. 1–3) showed that from the big data wave and the Apache Hadoop projects in 2014, a new concept appeared: the data lake. Another early work in this area showed that data lakes tended to govern data swamps. However, no formal definition was found in the literature.

Data Lake is defined as a methodology to approach the raw data, structured and non-structured within an enterprise and seen as an evolution of existing data architecture. Besides, a Data Lake is a methodology, a concept that embraces all enterprise data, moves them into one physical place to use them for future insight. The concept addresses the variety and the volume of the four significant data characteristics (Madera & Laurent, 2016, pp. 1–3). However, even if a Data Lake is defined as a methodology, it is not only a methodology. It is an actual new data architecture solution composed of hardware, software, and conceptual design, thus not limited to a methodology. The landscape is more comprehensive than a methodology and is an actual new reference data architecture and a new step in information architecture evolution (Madera & Laurent, 2016, pp. 1–3).

The volume, the variety, and the velocity of data is another essential thing. A Data lake is a low-cost storage physical environment based on Hadoop technology, populated by all data sources available in the enterprise. When the data is processed and used by power users or data scientists, the results will be saved in the data warehouse (Madera & Laurent, 2016, pp. 1–3).

The final proposed definition by Madera & Laurent (2016, pp. 1–3) is: A data lake is a logical view of all data sources or data set, in their raw format, available and accessible by a data scientist or statistician to find new insight.

They gave the next short overview of what a Data Lake is:

- A data lake is governed by a metadata sources index to guarantee the data quality.
- Rules, tools, and processes control a data lake to guarantee data governance.
- A data lake is limited to data scientist or data statistician access to guarantee data security, data privacy and compliance.
- A data lake accesses all types of data.
- A data Lake has a logical and physical design.

The establishment of this definition is the foundation to go further to be able to explore the impacts of this new evolution into the information architecture design. Figure 2 shows that before the end of 2020, more than 44 zettabytes of data will be generated, with more than 80 percent of them being unstructured (Madera & Laurent, 2016, pp. 1–3).

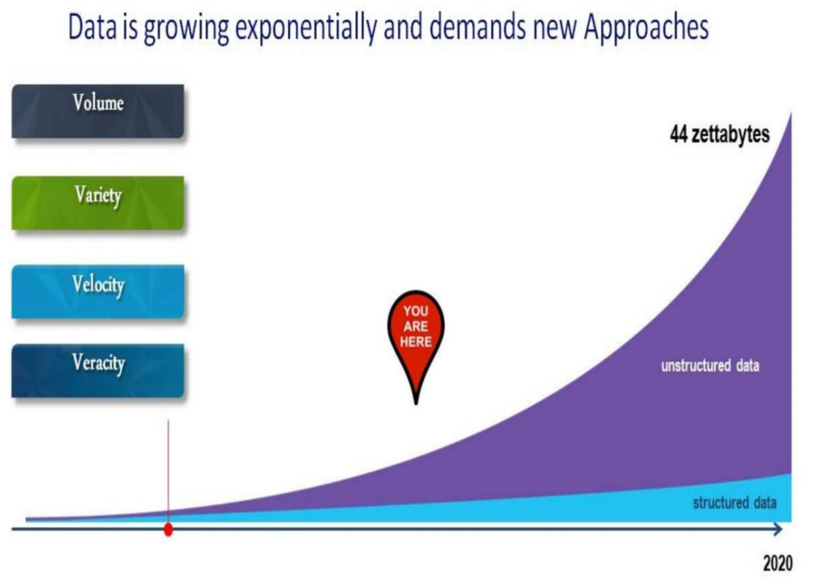


Figure 2: Data evolution (Madera & Laurent, 2016, pp. 1–3).

The term itself was introduced by James Dixon. Almost all modern enterprises get a massive amount of data about their IT infrastructure's current state. These data need to be processed promptly and correctly to identify information useful for business needs. Most of this data is unstructured (Miloslavskaya & Tolstoy, 2016, p. 303). According to another study executed by the IDC study “The Digital Universe of Opportunities:

Rich Data and the Increasing Value of the Internet of Things", the amount of unstructured data in 2020 is expected to be around 44 ZB (IDC, 2014).
Data can be structured, semi-structured, and unstructured, making it impossible to manage and process them effectively in a traditional way (Miloslavskaya, 2014). The criteria given by Miloslavskaya & Tolstoy (2016, p. 303) for determining the difference between big data IT and traditional IT are three "V's":

Volume: vast volumes of data
Velocity: very high data transfer rate
Variety: weakly structured data

Later four additional v's were added to the existing three:

Veracity: trust in the data
Variability: to what extent, and how fast, is the structure of the data changing?
Value: the meaning and value to derive business value from the data
Visibility: see what is happening

The data lake strategies can combine SQL and NoSQL database approach (Miloslavskaya & Tolstoy, 2016, p. 303).

Another research executed by Laskowski (2016) showed that a data lake refers to a massively scalable storage repository that holds a vast amount of raw data in its native format until it is needed and Shalom (2014) researched that in the current dynamic world, the enterprises data is growing too fast. As the stream of data from sensors, actuators, and machine-to-machine communication in the Internet of Things and modern networks is very large, it has become vital for enterprises to identify what data is time-sensitive and should be acted upon right away and what data can sit in a database or data lake until there is a reason to explore it.

Research executed by Khine & Wang (2018, p. 3025) described that many implementations of Data Lake are based initially on Apache Hadoop. A variety of data from heterogeneous data stores will be extracted to be stored in the Hadoop Cluster. HADOOP (Highly Available Object-Oriented Data Platform) is a widely popular big data tool especially suitable for batch processing workload of big data. Hadoop has two main components, HDFS (Hadoop Distributed File System) and MapReduce engine. HDFS File System handles the single point of failure and scalability by replicating multiple copies of data blocks in different cluster nodes. All data stored in these data blocks will be processed in the MapReduce approach. Data will be retrieved as a list of key-value pairs, i.e., the Map phase. The same data keys will be shuffled, sorted, and listed into groups to perform necessary operations, i.e., Reduce phase. All data produced by an enterprise will be dumped into the Data Lake Hadoop Cluster. This research found that concepts from distributed and parallel systems are reapplied as the foundation of big data, such as MapReduce paradigms for handling the significant V's characteristics, volume, velocity, variety, value, and value. The incumbent SQL databases with ACID (Atomic, Consistent, Isolated, and Durable) characteristics are challenged (and sometimes even replaced) by NoSQL databases with BASE (Basically Available, Soft state, Eventual consistency) characteristics. Also, all data generated by an organization, regardless of types, structures, or formats, will be stored in Hadoop clusters or other similar frameworks in their original forms. A data lake may contain raw, unstructured, or multi-structured data where most of these data may have unrecognized value for the organization. Metadata management is an essential aspect of Data Lake. As Data Lakes do not have a pre-defined schema like data warehouses, they must rely on metadata during the query time for the analysis process, added when data are stored.

The research of Khine & Wang (2018, p. 3025) continues and found that the basic idea of Data Lake is simple, all data emitted by the organization will be stored in a single data structure called Data Lake. Data will be stored in the lake in their original format. Complex preprocessing and transformation of loading data into data warehouses will be eliminated. The upfront costs of data ingestion can also be reduced. Once data are placed in the lake, it is available for analysis by everyone in the organization. A data lake uses a flat architecture to store data in its raw format. Each data entity in the lake is associated with a unique identifier and a set of extended metadata, and consumers can use purpose-built schemas to query relevant data, which will result in a smaller set of data that can be analyzed to help answer a consumer's question. Data are extracted and transformed to conform to data warehouse schema and loaded into the Data Warehouse. Data Lake is a data repository where all data in an enterprise, i.e., structured, semi-structured, unstructured data, are stored altogether regardless of types, format, or structure. Pentaho CEO Jame Dixon first initiated the idea of Data Lake.

Another approach is that all the data from these databases (Extract) will be stored (Load) into the data lake without changing their format. When data are required, data in the lake will be transformed (Transform) according to the enterprise system's parts (Khine & Wang, 2018, p. 3025). Data Lake concepts deviate from the data warehouse by processing data in the ELT order and utilizing the "Schema-on-Read" approach, then data warehouses that follow the traditional ETL process approach. First, data from operational databases are extracted (E). The data are then processed, cleaned, and transformed (T) before loading (L) them into the data warehouses or data marts. Data warehouses are specially designed to handle a read-heavy workload for analytics. Data warehouses need to define their schema in advance before data are loaded. Therefore, they are considered the "Schema-On-Write" approach (Khine & Wang, 2018, p. 3025). However, one of the biggest pitfalls of Data Lake is becoming a data swamp. No one knows what will be put into the lake. Moreover, no procedures are preventing them from entering incorrect data, repeated data, or incorrect data.

The research of Khine & Wang (2018, p. 3025) ends with introducing maturity levels. A Data Lake may need to pass through five maturity levels. They are

1. Consolidated and categorized raw data
2. Attribute-level metadata tagging and linking such as joins
3. Data set extraction and analysis
4. Business-specific tagging, synonym identification, and links

5. The convergence of meaning within context. As the data lake maturity level increases, the usage of Data Lake across the enterprise and the value of analytics will increase.

Existing design/architectures

Data lakes ingest raw data in its original format from heterogeneous data sources and allow users to query and explore them. Research executed by Hai, Geisler, & Quix (2016, pp. 1–3) mentioned that schema information, mappings, and other constraints are not defined explicitly or required initially for a Data Lake; it is crucial to extract as much metadata as possible from the data sources during the ingestion phase. Metadata management is crucial for data reasoning, query processing, and data quality management. The Data lake is hardly usable without any metadata as the data's structure and semantics are not known, which turns a Data Lake quickly into a data swamp.

Because of the problems discussed before, Hai, Geisler, & Quix (2016, pp. 1–3) propose a framework called Constance, which can be used as a basis in Data Lake projects because it provides flexibility extensible framework for data management problems within Data Lake systems. Constance manages structural and semantic metadata, provides means to enrich the metadata with schema matching and schema summarization techniques, and offers a unified interface for query processing.

Fig. 3 shows the architecture of Constance, as well as its key components. Constance can be roughly divided into three functional layers: ingestion, maintenance, and querying.

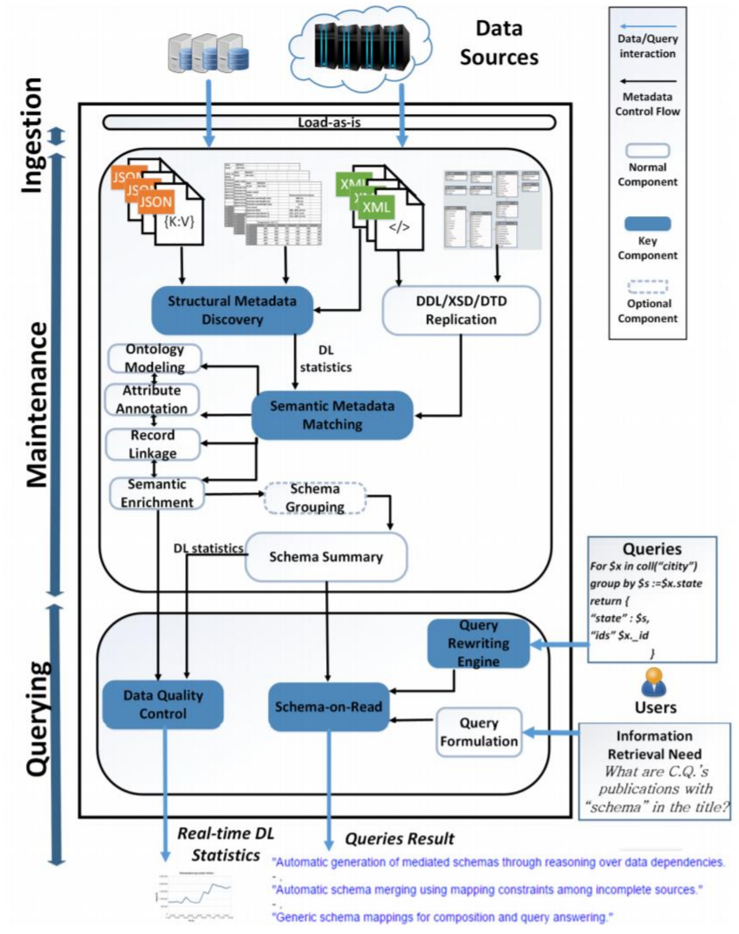


Figure 3: Constance System Overview

The ingestion layer is responsible for importing data from heterogeneous sources into the Data lake system.

The maintenance layer mainly contributes to Constance's metadata management functions.

The backend of the maintenance layer provides the necessary functions for data storage and efficient querying. Metadata is crucial for future querying.

All the above functions, eventually, serve for information retrieval, in the form of query answering. In typical cases, users either input concrete queries in a particular query language or have an information retrieval need that supports the user in formulating a query starting from some keywords (Hai, Geisler, & Quix, 2016, pp. 1–3).

Another research executed by Beheshti et al. (2017, pp. 1–3) created a Data Lake architecture. Their questions were how to store information items (from structured entities to unstructured documents)? What technology to use for persisting the data (from Relational to NoSQL databases)? How to deal with the large volume of data generated continuously (from Key-value and document to object and graph store)? How to trace and persist information about data (from descriptive to administrative)? What technology to use for indexing the data/metadata? How to query the data lake (from SQL to full-text search)?
To address the challenges mentioned above, they presented CoreDB, an open-source data lake service.

CoreDB offers researchers and developers a single REST API to organize, index, and query their data and metadata. CoreDB manages multiple database technologies (from Relational to NoSQL databases), exposes the power of Elasticsearch, and weaves them together at the application layer (Beheshti et al., 2017, pp. 1–3).

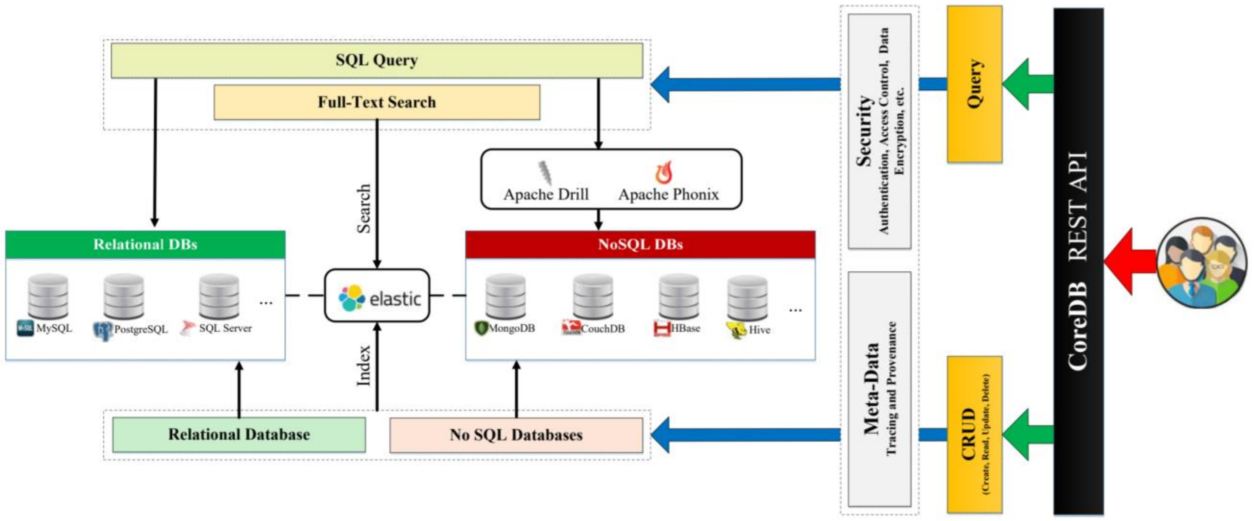


Figure 4: CoreDB Architecture

It starts with a Database Service that powers multiple relational and NoSQL (key/value, document, and graph stores) database-as-a-service for developing Web data applications, i.e., data-driven web applications. Therefore, analysts are enabled to build a data lake, create relational and NoSQL datasets within the data lake, and CRUD (Create, Read, Update and Delete) and query entities in those datasets. Next, Elasticsearch is used as a search engine based on Apache Lucene to support a robust index and full-text search.

Based on the literature review results in chapter 5, an extra dimension to consider is metadata management. Without metadata management, Data Lake will turn into a data swamp.

The metadata management is used in the Data Lake ETL layer or ingestion phase as discussed in the literature review by Constance and CoreDB, and therefore, the research will continue to look for any tooling which will enhance the metadata management in the ETL part of this study. This study is mainly focused on the ETL part and not the ELT part because ETL is the most common way of organizing the process.

2.2 Data Warehouse vs Data Lake

After the literature review, it can be concluded what main differences Data warehouses and Data Lakes have. The table below shows the differences between a Data Warehouse and a Data Lake. This table can be filled with more accuracy based on the research where the difference in query languages is added.

Characteristics	**Data Warehouse**	**Data Lake**
Storage	Extracted and transformed data from other database resources	Data is raw and unchanged.
Data	Structured data	Structured, semi-structured, and unstructured data
Data Quality	Data ready for use and serves as a single point of truth	Raw data needs transformation
Normalization	Denormalized	Not normalized
Data timeline	Current data	All kinds of past, present and future data
Costs of storage	Higher cost storage	Low-cost storage
Accessibility	Data is complicated	Can be quickly updated and changed
Compatibility	Stored data is transformed and may give problems when changes made	Data is raw and is flexible for changing
Schema	Created before implementation	Created at the time of analysis
Query Language	OQL, LINQ, T-SQL, SQL, GraphQL	Big SQL, U-SQL, Google Big Query, NoSQL

Table 1: Data Warehouse vs Data Lake

2.3 Commonly used criteria

Criteria are used to choose something suitable for a specific situation. These criteria are conditions that a product must have to be accepted by a user or customer. These conditions can apply to many things such as cars, clothing, or in this case, software.

The list with criteria is built out of reasoning and common sense. Furthermore, discussions with architects, engineers, and managers in the interviews helped compose the criteria used in this study.
The criteria used are:

1. Speed/Efficiency: measures how fast a tool is and how it reaches speed.

2. Integration: measures the number of different programming languages and external tooling that can be used.

3. Usability: measures how easy the user can use a tool.

4. Flexibility: measures if a tool can handle structured, semi-structured, and unstructured data.

5. Graphical User Interface (GUI): measures the ability to have a graphical user interface.

6. Security: describes what security measures are taken.

7. Meta Data Management (MDM): measures if a tool can handle metadata management.

8. Application Programming Interface (API): measures if a tool can have an API to talk with other sources.

9. Code: measures the ability of coding in a tool.

10. Low code: measures the ability to use no or low code in a tool.

The criteria discussed will be applied to the query languages, ETL tooling, and visualization tooling, respectively, and how all the aspects can interrelate to each other.

2.4 Commonly used query languages

A database is a collection of organized information to be accessed, managed, and updated. Alternatively, a database is simply a place where the data is stored. Databases provide means of retrieving records or parts of records and performing various calculations before displaying the results.

The interface by which such data is accessed, managed, updated, and calculated is called the query language. It is a specialized language for requesting information from a database. It refers to any computer programming language that requests and retrieves data from database and information systems by sending queries.

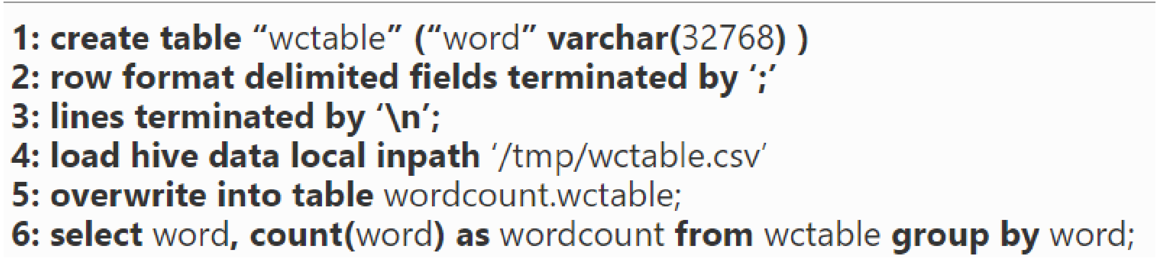
One part of this study is researching different query languages and finding strengths and weaknesses. The list is divided into two sub-chapters. The first one is the commonly used query languages with examples, and the second one is about other query languages that are not commonly used without examples.

The commonly used query languages are listed below:

MapReduce-based Big SQL:
Big SQL is MapReduce-based designed for providing native SQL for querying data managed by Hadoop (Birjali, Beni-Hssane, & Erritali, 2018, pp. 1–3).

Big SQL is fast due to its parallel processing SQL abilities and low-latency parallel execution processing. It runs on the top of Hadoop and translates all queries to native MapReduce (MR) jobs, supports queries expressed in native SQL declarative language, JDBC/ODBC driver access from Linux and Windows platforms, Java, C#, Python, C++, and R integration. Furthermore, does it use HCatalog (metastore) of Hbase for data access and the Hive storage engines to read/write data. Big SQL is flexible, and usability is added with Cloudera Navigator, the GUI of this tool. Furthermore, does it use end to end security and extra security layer is added with open source projects Knox and Ranger. Knox provides a framework for managing security and supports security implementations on

Hadoop clusters. The Ranger project is focused on developing tools and techniques to help users deploy and standardize security across Hadoop clusters. Big SQL has integration with other languages and can query unstructured data. However, it does not benefit from adding nodes. As a result, the running time performance is decreased by 43% from one to ten nodes and has no API capabilities.



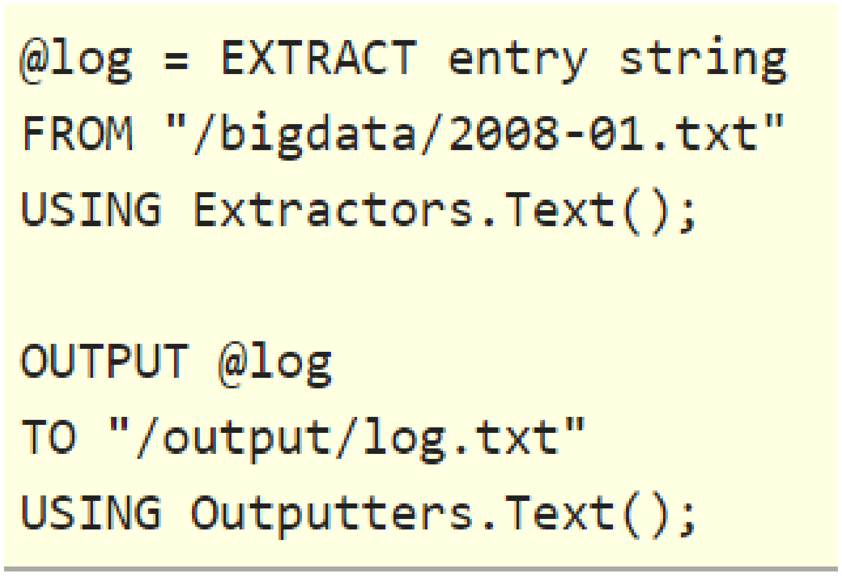
```
create table "wctable" ("word" varchar(32768) )
row format delimited fields terminated by ';'
lines terminated by '\n';
load hive data local inpath '/tmp/wctable.csv'
overwrite into table wordcount.wctable;
select word, count(word) as wordcount from wctable group by word;
```

Figure 5: Big SQL query example

U-SQL:

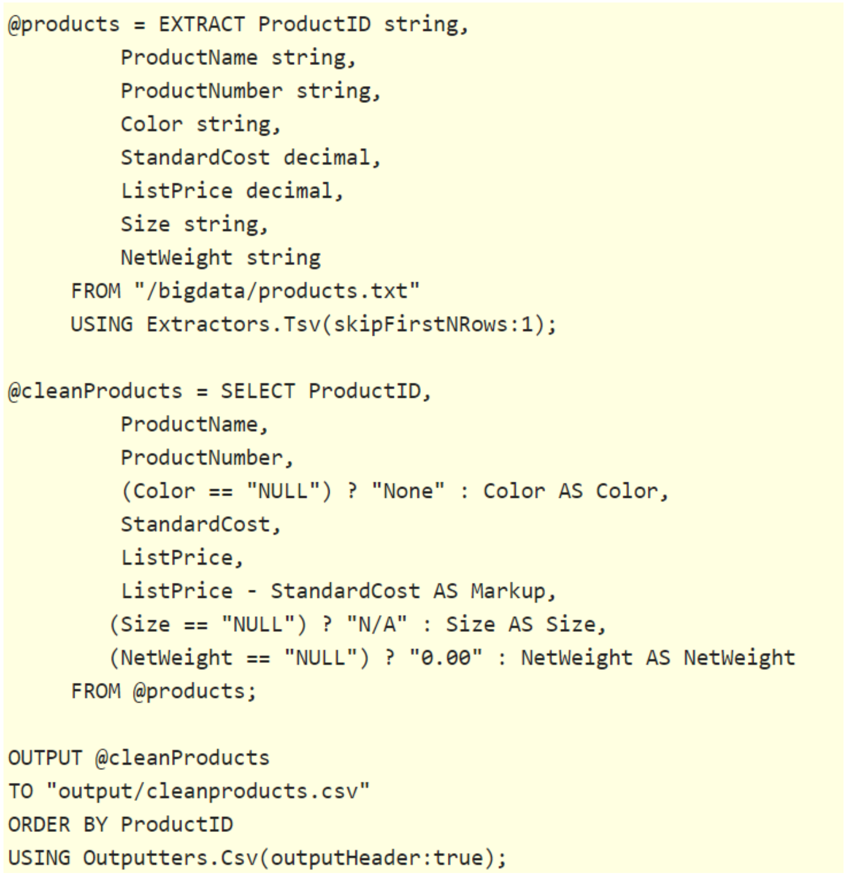
U-SQL is a language that combines declarative SQL with imperative C# to let processing data at any scale. Through the scalable, distributed-query capability of U-SQL, data can efficiently be analyzed across relational stores such as Azure SQL Database. USQL is intended to be a cross-platform query language that enables discovering various types of services in a unified manner (Tsalgatidou, Pantazoglou, & Athanasopoulos, 2006, pp. 1–3).

U-SQL is based on T-SQL while it uses C# types as default. This easily allows conceptualization of how data will be processed while writing queries and not being scared with new frameworks or concepts. With this architecture, it can process any type of data and integrates custom code seamlessly. U-SQL can efficiently scale to any size of data, is for massive data processing, and therefore can dump whatever in the data lake and run U-SQL on top of it. It is flexible and easy to develop. U-SQL can handle any type of data, is integrated with many different languages, up to 1 million GB supported with visual studio or web portal as GUI, and is secure with Azure Active Directory. However, the U-SQL language is not to substitute the existing and emerging service description protocols in the various service areas (e.g., WSDL, WSDL-S, OWL-S). Furthermore, it is not available on other than Azure platforms yet and has no API capabilities.



```
@log = EXTRACT entry string
FROM "/bigdata/2008-01.txt"
USING Extractors.Text();

OUTPUT @log
TO "/output/log.txt"
USING Outputters.Text();
```

Figure 6: Easy USQL query



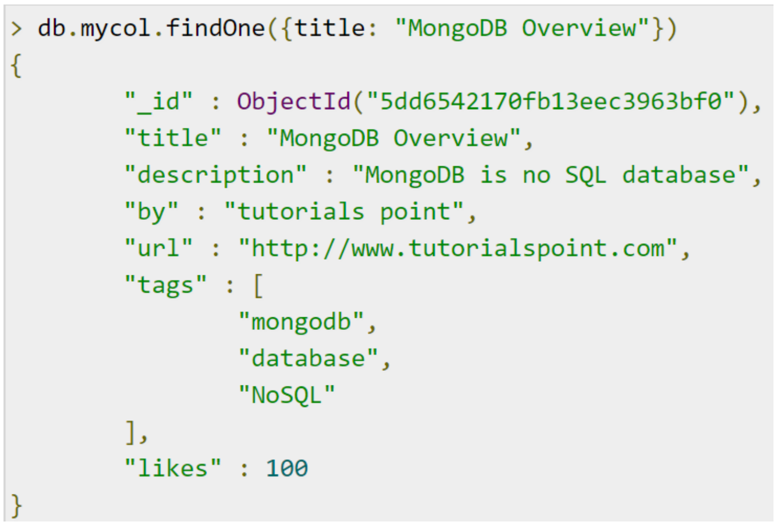
```
@products = EXTRACT ProductID string,
         ProductName string,
         ProductNumber string,
         Color string,
         StandardCost decimal,
         ListPrice decimal,
         Size string,
         NetWeight string
     FROM "/bigdata/products.txt"
     USING Extractors.Tsv(skipFirstNRows:1);

@cleanProducts = SELECT ProductID,
         ProductName,
         ProductNumber,
         (Color == "NULL") ? "None" : Color AS Color,
         StandardCost,
         ListPrice,
         ListPrice - StandardCost AS Markup,
        (Size == "NULL") ? "N/A" : Size AS Size,
        (NetWeight == "NULL") ? "0.00" : NetWeight AS NetWeight
     FROM @products;

OUTPUT @cleanProducts
TO "output/cleanproducts.csv"
ORDER BY ProductID
USING Outputters.Csv(outputHeader:true);
```

Figure 7: Difficult USQL query

NoSQL:

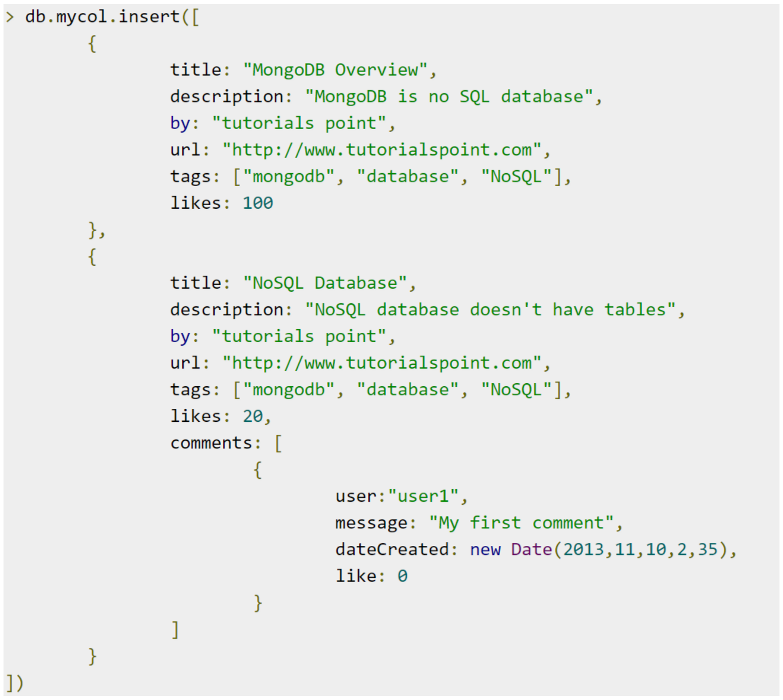
NoSQL databases are interchangeably referred to as nonrelational, NoSQL DBs to highlight the fact that they can handle vast volumes of rapidly changing, unstructured data in different ways than a relational SQL database with rows and tables (Cattell, 2011, p. 12).

NoSQL allows freedom, has more speed due to the efficient use of distributed indexes and RAM for data storage and flexibility to change both schema and queries to adapt to data requirements. It provides compelling operational advantages and savings with the ability to scale "out" horizontally or add less expensive servers without having to upgrade. No SQL is compatible with large volumes of rapidly changing structured, semi-structured, and unstructured data. Geographically distributed scale-out architecture instead of expensive, monolithic architecture. Integration with C# and .NET. API connections available and a GUI with MongoDB manager. Nevertheless, NoSQL is not mature, has less support, and the system can have only two out of three of the following properties: consistency, availability, and partition-tolerance. The NoSQL systems generally give up consistency and are less secure.



```
> db.mycol.findOne({title: "MongoDB Overview"})
{
        "_id" : ObjectId("5dd6542170fb13eec3963bf0"),
        "title" : "MongoDB Overview",
        "description" : "MongoDB is no SQL database",
        "by" : "tutorials point",
        "url" : "http://www.tutorialspoint.com",
        "tags" : [
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 100
}
```

Figure 8: Easy NoSQL query



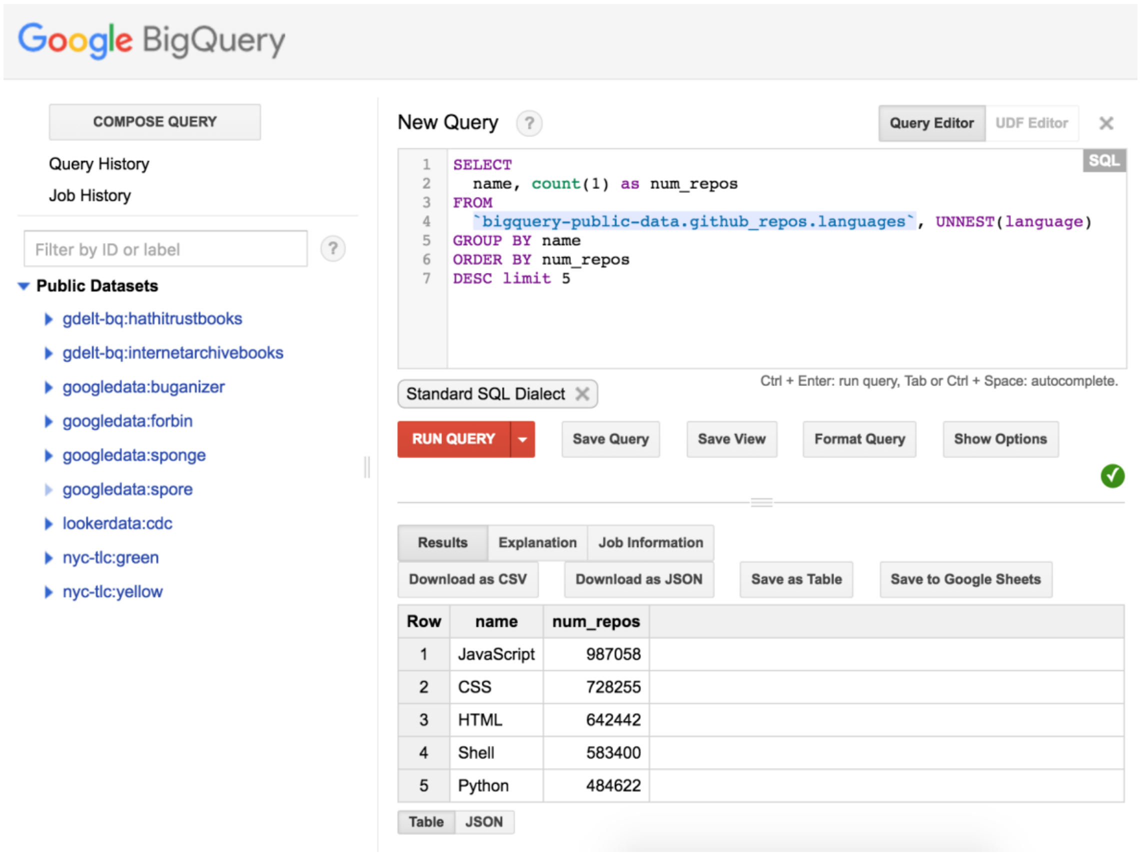
```
> db.mycol.insert([
	{
		title: "MongoDB Overview",
		description: "MongoDB is no SQL database",
		by: "tutorials point",
		url: "http://www.tutorialspoint.com",
		tags: ["mongodb", "database", "NoSQL"],
		likes: 100
	},
	{
		title: "NoSQL Database",
		description: "NoSQL database doesn't have tables",
		by: "tutorials point",
		url: "http://www.tutorialspoint.com",
		tags: ["mongodb", "database", "NoSQL"],
		likes: 20,
		comments: [
			{
				user:"user1",
				message: "My first comment",
				dateCreated: new Date(2013,11,10,2,35),
				like: 0
			}
		]
	}
])
```

Figure 9: Difficult NoSQL query

Google big query:

Google big query is a fully managed cloud service that enables storage and fast querying of large and multi-faceted datasets. (Lopez, Seaton, Ang, Tingley, & Chuang, 2017, pp. 1–3).

Google Big Query is highly scalable, cost-effective, has no technical overhead costs for maintaining infrastructure, scalability of processing research data products across a growing number of courses and users, is fast, is secure due to Identity and Access Management, and has API capabilities. However, Google Big Query is only compatible with specific extensions such as JSON, CSV, or Avro. It has file size limits; query prices are high, queries need to be optimized to be cost-effective, cannot use it outside the Google platform, is challenging to learn, cannot join different tables with unstructured data, and has only a web UI as GUI.

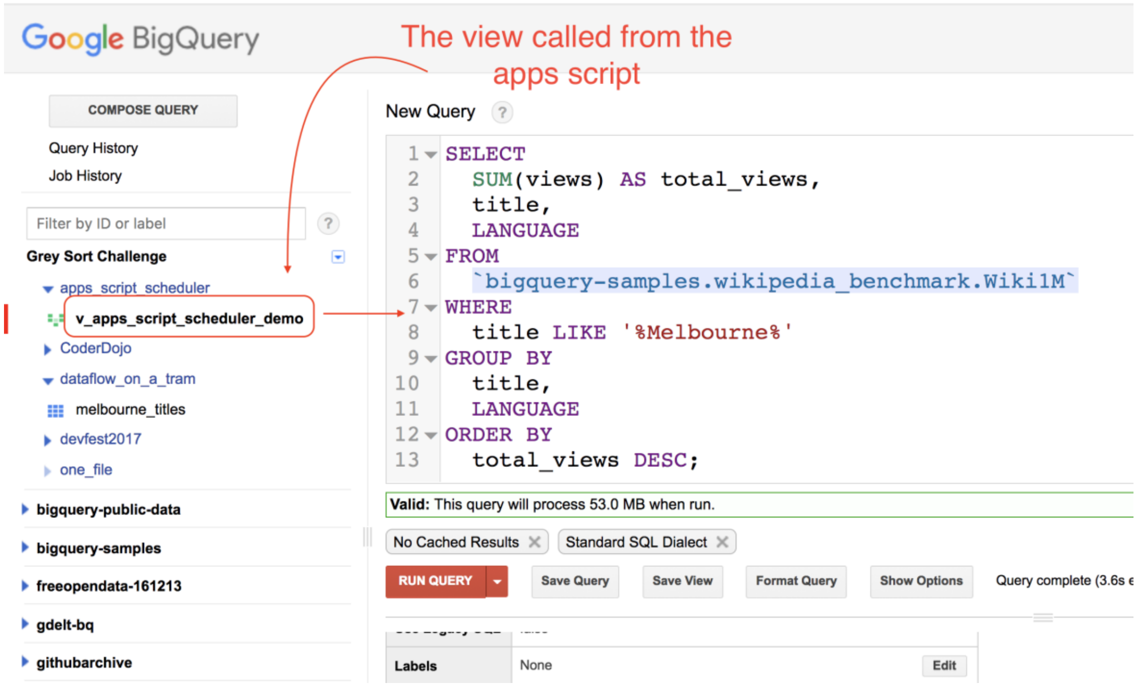


```
SELECT
  name, count(1) as num_repos
FROM
  `bigquery-public-data.github_repos.languages`, UNNEST(language)
GROUP BY name
ORDER BY num_repos
DESC limit 5
```

Ctrl + Enter: run query, Tab or Ctrl + Space: autocomplete.

Row	name	num_repos
1	JavaScript	987058
2	CSS	728255
3	HTML	642442
4	Shell	583400
5	Python	484622

Figure 10: Easy Google BigQuery query

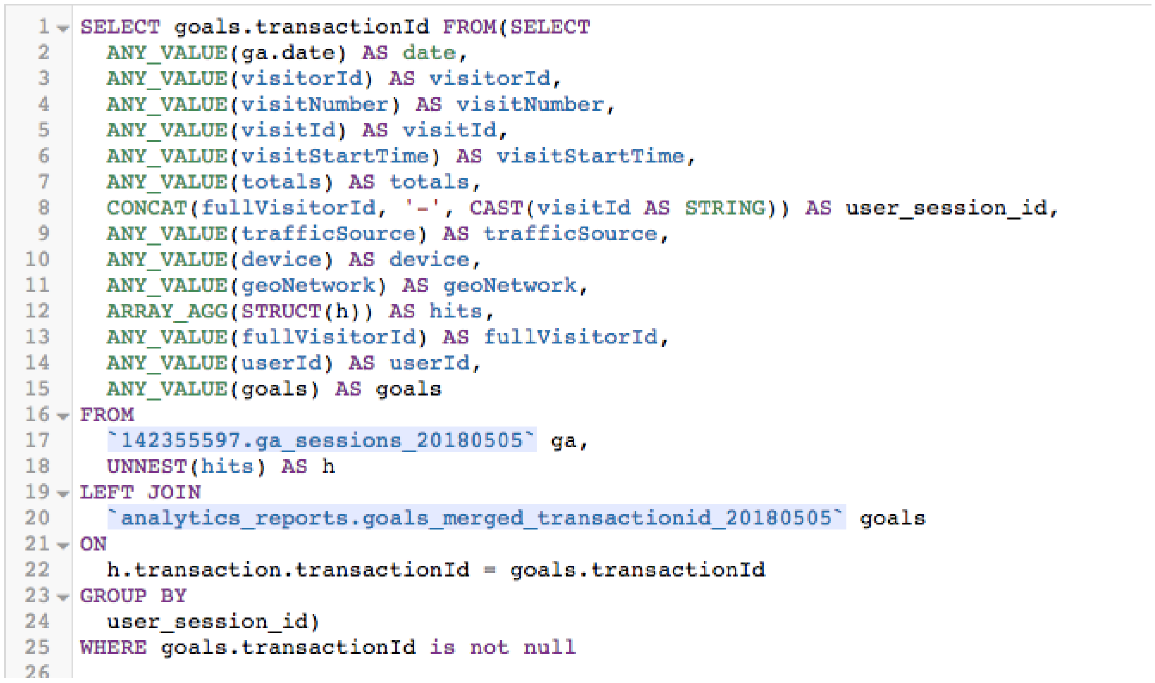


```
SELECT
  SUM(views) AS total_views,
  title,
  LANGUAGE
FROM
  `bigquery-samples.wikipedia_benchmark.Wiki1M`
WHERE
  title LIKE '%Melbourne%'
GROUP BY
  title,
  LANGUAGE
ORDER BY
  total_views DESC;
```

Valid: This query will process 53.0 MB when run.

Query complete (3.6s e

Figure 11: Difficult Google BigQuery query



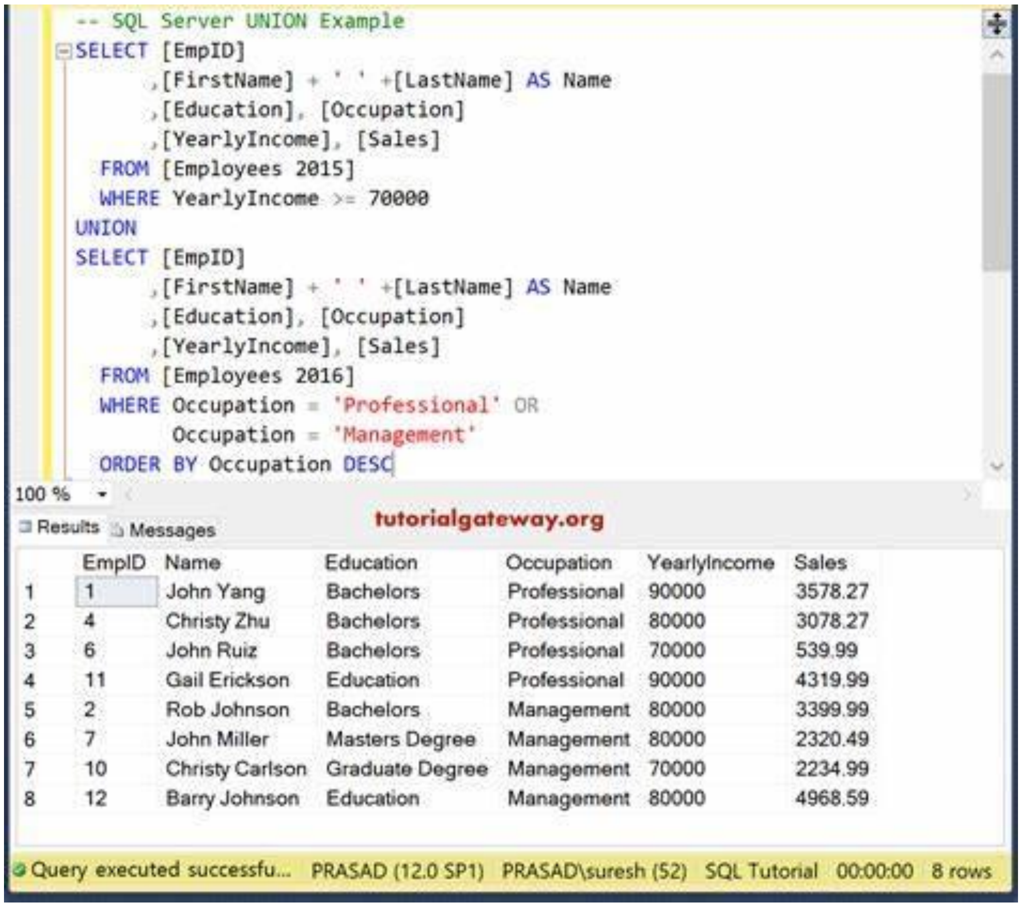
```sql
SELECT goals.transactionId FROM(SELECT
  ANY_VALUE(ga.date) AS date,
  ANY_VALUE(visitorId) AS visitorId,
  ANY_VALUE(visitNumber) AS visitNumber,
  ANY_VALUE(visitId) AS visitId,
  ANY_VALUE(visitStartTime) AS visitStartTime,
  ANY_VALUE(totals) AS totals,
  CONCAT(fullVisitorId, '-', CAST(visitId AS STRING)) AS user_session_id,
  ANY_VALUE(trafficSource) AS trafficSource,
  ANY_VALUE(device) AS device,
  ANY_VALUE(geoNetwork) AS geoNetwork,
  ARRAY_AGG(STRUCT(h)) AS hits,
  ANY_VALUE(fullVisitorId) AS fullVisitorId,
  ANY_VALUE(userId) AS userId,
  ANY_VALUE(goals) AS goals
FROM
  `142355597.ga_sessions_20180505` ga,
  UNNEST(hits) AS h
LEFT JOIN
  `analytics_reports.goals_merged_transactionid_20180505` goals
ON
  h.transaction.transactionId = goals.transactionId
GROUP BY
  user_session_id)
WHERE goals.transactionId is not null
```

Figure 12: Difficult Google BigQuery query

SQL:

SQL or Structured query language is used as a medium of communication with the relational database management systems.

SQL is fast on structured data, has data integration standards, is mature, is secure due to its RBAC capabilities, has data integration script available, and has SQL management studio as GUI. Nevertheless, SQL has a difficult interface, is expensive, has no API capabilities, and is not flexible because it has predefined schemas.

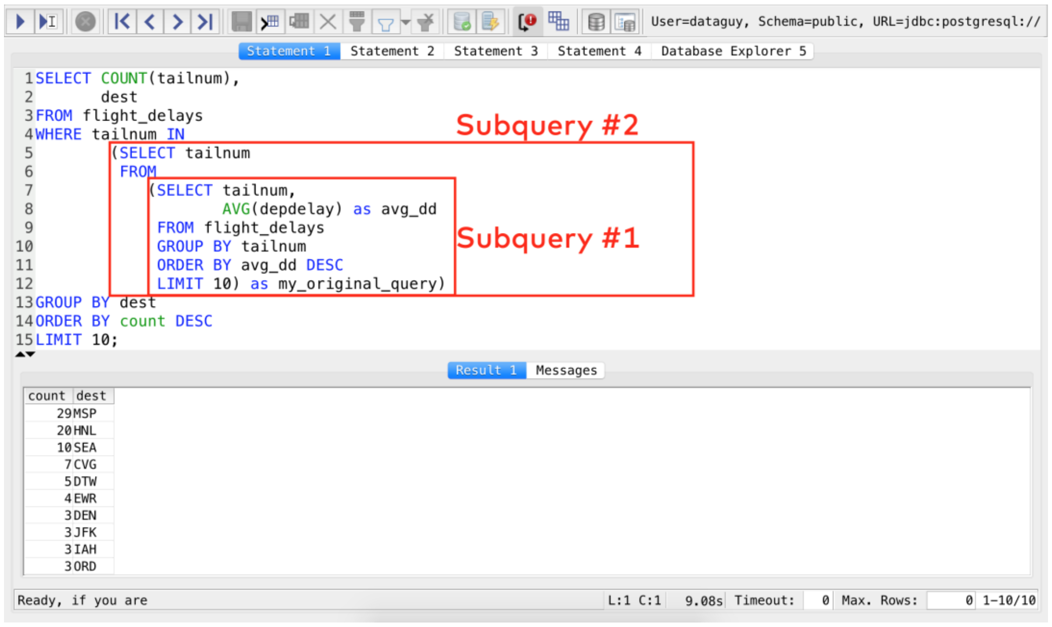


```
-- SQL Server UNION Example
SELECT [EmpID]
      ,[FirstName] + ' ' +[LastName] AS Name
      ,[Education], [Occupation]
      ,[YearlyIncome], [Sales]
  FROM [Employees 2015]
  WHERE YearlyIncome >= 70000
UNION
SELECT [EmpID]
      ,[FirstName] + ' ' +[LastName] AS Name
      ,[Education], [Occupation]
      ,[YearlyIncome], [Sales]
  FROM [Employees 2016]
  WHERE Occupation = 'Professional' OR
        Occupation = 'Management'
  ORDER BY Occupation DESC
```

	EmpID	Name	Education	Occupation	YearlyIncome	Sales
1	1	John Yang	Bachelors	Professional	90000	3578.27
2	4	Christy Zhu	Bachelors	Professional	80000	3078.27
3	6	John Ruiz	Bachelors	Professional	70000	539.99
4	11	Gail Erickson	Education	Professional	90000	4319.99
5	2	Rob Johnson	Bachelors	Management	80000	3399.99
6	7	John Miller	Masters Degree	Management	80000	2320.49
7	10	Christy Carlson	Graduate Degree	Management	70000	2234.99
8	12	Barry Johnson	Education	Management	80000	4968.59

```
Query executed successfu...   PRASAD (12.0 SP1)   PRASAD\suresh (52)   SQL Tutorial   00:00:00   8 rows
```

Figure 13: Easy SQL query



```
SELECT COUNT(tailnum),
       dest
FROM flight_delays
WHERE tailnum IN
        (SELECT tailnum
         FROM
            (SELECT tailnum,
                   AVG(depdelay) as avg_dd
             FROM flight_delays
             GROUP BY tailnum
             ORDER BY avg_dd DESC
             LIMIT 10) as my_original_query)
GROUP BY dest
ORDER BY count DESC
LIMIT 10;
```

```
count dest
   29 MSP
   20 HNL
   10 SEA
    7 CVG
    5 DTW
    4 EWR
    3 DEN
    3 JFK
    3 IAH
    3 ORD
```

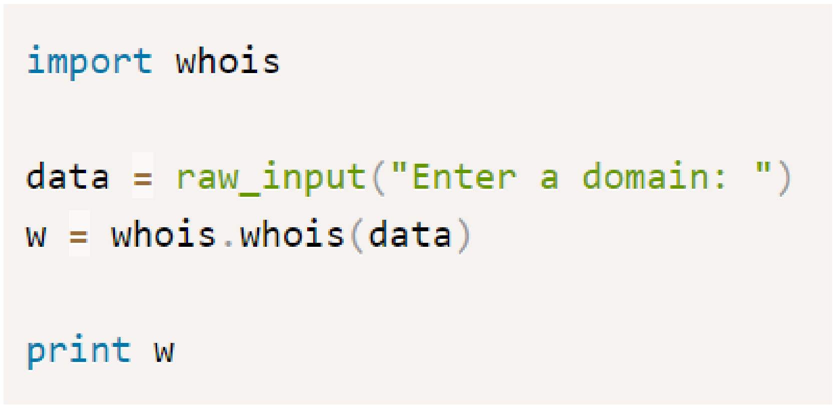
```
Ready, if you are        L:1 C:1   9.08s  Timeout:  0  Max. Rows:  0  1-10/10
```

Figure 14: Difficult SQL query

Python:

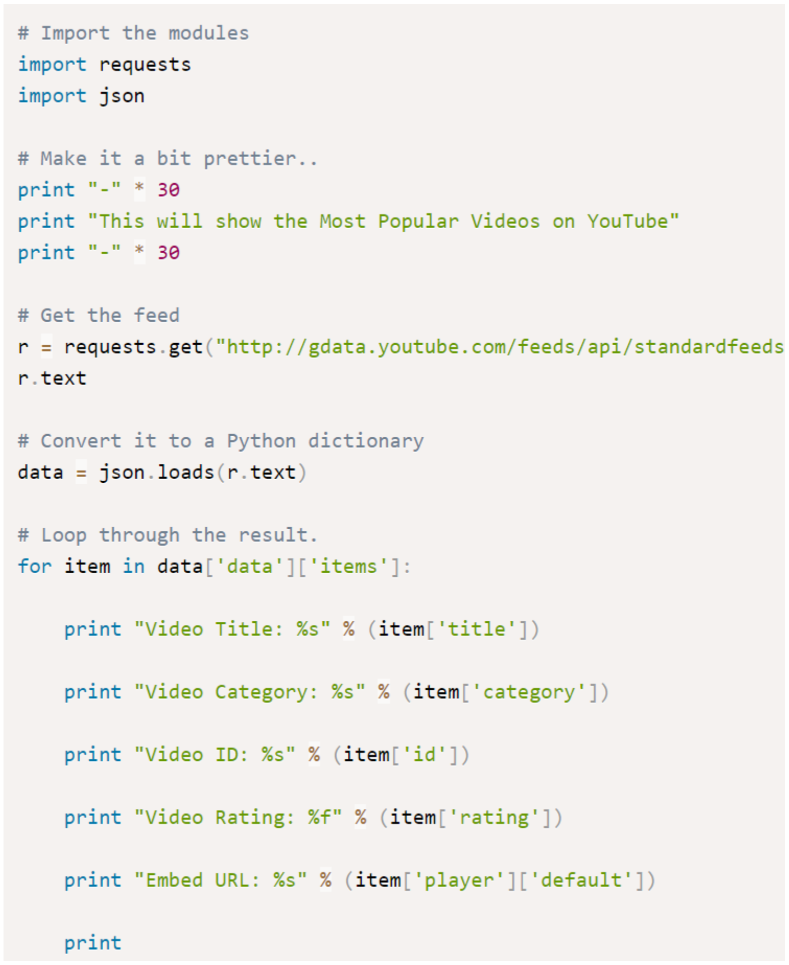
Python is an object-oriented, open-source programming language. (Lutz, Lewin, & Willison, 2001, pp. 1–3)

It is easy to work with, runs on every platform, has integration with other tools and languages. Furthermore, Python is stable, has API capabilities, is flexible, has a GUI with PyCharm but is not fast, has run-time errors, lacks multi-processor support, database access layer problems, and is not secure (broaden security with security scans available).



```
import whois

data = raw_input("Enter a domain: ")
w = whois.whois(data)

print w
```

Figure 15: Easy Python query



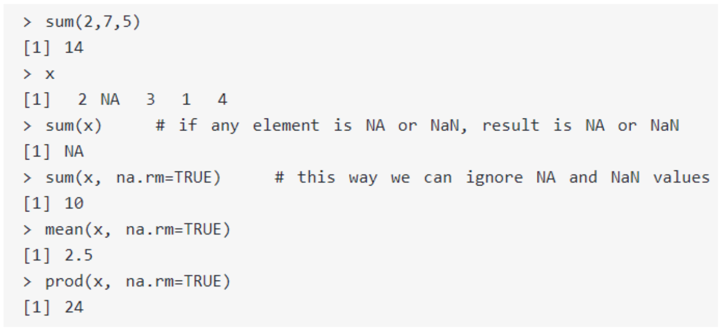
```
# Import the modules
import requests
import json

# Make it a bit prettier..
print "-" * 30
print "This will show the Most Popular Videos on YouTube"
print "-" * 30

# Get the feed
r = requests.get("http://gdata.youtube.com/feeds/api/standardfeeds
r.text

# Convert it to a Python dictionary
data = json.loads(r.text)

# Loop through the result.
for item in data['data']['items']:

    print "Video Title: %s" % (item['title'])

    print "Video Category: %s" % (item['category'])

    print "Video ID: %s" % (item['id'])

    print "Video Rating: %f" % (item['rating'])

    print "Embed URL: %s" % (item['player']['default'])

    print
```

Figure 16: Difficult Python query

R:

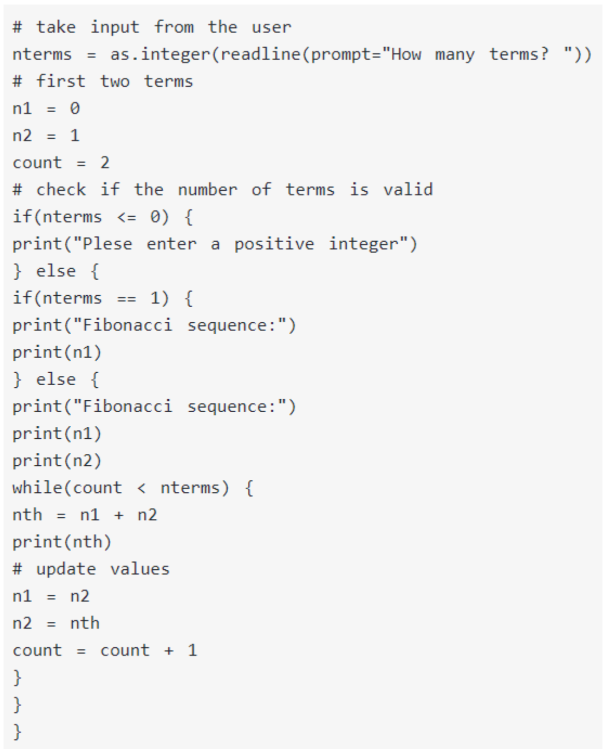
R is a scripting language for statistical data manipulation and analysis (Matloff, 2011, pp. 1–3).

R is an open-source tool, has many packages available, can visualize data, is highly compatible with other languages, has API capabilities, is compatible with many sources, is flexible, and has a GUI with R studio. However, R's security capabilities are not well, are not fast and efficient on large datasets due to memory management problems, and without programming experience, a bit hard to learn.



```
> sum(2,7,5)
[1] 14
> x
[1]  2 NA  3  1  4
> sum(x)     # if any element is NA or NaN, result is NA or NaN
[1] NA
> sum(x, na.rm=TRUE)     # this way we can ignore NA and NaN values
[1] 10
> mean(x, na.rm=TRUE)
[1] 2.5
> prod(x, na.rm=TRUE)
[1] 24
```

Figure 17: Easy R query



```
# take input from the user
nterms = as.integer(readline(prompt="How many terms? "))
# first two terms
n1 = 0
n2 = 1
count = 2
# check if the number of terms is valid
if(nterms <= 0) {
print("Plese enter a positive integer")
} else {
if(nterms == 1) {
print("Fibonacci sequence:")
print(n1)
} else {
print("Fibonacci sequence:")
print(n1)
print(n2)
while(count < nterms) {
nth = n1 + n2
print(nth)
# update values
n1 = n2
n2 = nth
count = count + 1
}
}
}
```

Figure 18: Difficult R query

The less commonly used query languages are:

Embedded SQL:

Embedded SQL combines a programming language's computing power and SQL's database manipulation capabilities. This is a method for combining SQL's data manipulation capabilities and any programming language's computing power. Then embedded statements are in line with the program source code of the host language. The code of embedded SQL is parsed by a preprocessor, which is also embedded and replaced by the host language called for the code library; it is then compiled via the host's compiler. This language was used in the past, but it is old now and therefore not considered in the mechanism.

Embedded SQL is easy to understand because of less syntax to learn and its one-step deployment. Furthermore, it integrates with other languages available and has good error handling but does not perform well on large datasets.

HTSQL:

According to Evans (2006, p. 2-4), HyperText Structured Query Language (HTSQL) is a schema-driven URI to SQL query language that takes a request over HTTP, converts it to a SQL query, executes the query against a database, and returns the results in a format best suited for the user agent (CSV, HTML). HTSQL is an extension to the HTTP/1.1 protocol that allows clients to access a standard SQL database remotely. This language was used, but it is old now and therefore not considered in the method.

HTSQL has a rapid web application development architecture, is a fast language on transactional data, has a web-friendly syntax, has integrated use of the HTTP protocol to provide authentication, and uses data caching and encryption. Furthermore, is HTSQL mainly considered as a web query language.

Object Query Language:

Object Query Language (OQL) is a query language standard for object-oriented databases modeled after SQL as researched by (Li, J. Z., Ozsu, M. T., Szafron, D., & Oria, V. (1997, September)). This language was used, but it is old now and therefore not considered in the method.

It uses entity and association names instead of actual database table names and can use predefined relations to join objects without calculating which columns should be coupled quickly. Many SQL keywords also work in OQL and deal with complex objects without changing the set construct and the select-from-where clause. Furthermore, does it integrate with different languages, and is API compatible. However, OQL queries do not take security into account out of the box.

Language Integrated Query:

Language Integrated Query (LINQ, pronounced as "link") is a Microsoft .NET Framework component that adds native data querying capabilities to .NET languages, released initially as a significant part of .NET Framework 3.5 in 2007.

LINQ allows users to write queries uniformly in the programming language itself, taking full advantage of strong typing and tool support (Torgersen, 2006, p. 736). LINQ extends the language by adding query expressions, which are takin to SQL statements, and can be used to extract conveniently and process data from arrays, enumerable classes, XML documents, relational databases, and third-party data sources. This language was used, but it is old now and therefore not considered in the method.

LINQ supports safety, is easy to deploy, is easy to learn, is compatible with .NET, and supports multiple databases. Furthermore, LINQ allows writing queries uniformly in the programming language itself, taking full advantage of strong typing and tool support. The LINQ framework comes with LINQ providers for in-memory objects, SQL data, and XML documents and is fast due to its multi-threading processing. However, it needs to process the entire query, which might have a negative performance impact. Suppose a change was made in the query; the entire query needs to be recompiled and redeployed. Without knowledge of this query language, it is easy to build inefficient code.

GraphQL:

GraphQL is an open-source data query and manipulation language for APIs, and a runtime for fulfilling queries with existing data and need to write all endpoints manually. GraphQL is a recently proposed and increasingly adopted a conceptual framework for providing a new type of data access interface on the Web. The framework includes a new graph query language whose semantics has been specified informally only. This has prevented the formal study of the main properties of the language. (Hartig & Pérez, 2018, pp. 1–3). This language is mainly used for API configuration and therefore not taken into consideration in the mechanism.

GraphQL is self-documenting, defines precisely the user wants, replaces multiple REST requests with a single call, adopted by an increasing number of users including Coursera, GitHub, Neo4J, and Pinterest, works efficiently, has integration with other API and languages, is fast and stable. Nevertheless, it does not support all entities and relationships with APIs, does not follow the HTTP spec for caching, and instead uses a single endpoint, has some cache errors, and a fundamental understanding of the language's properties is missing.

Transact-SQL:

Transact-SQL is Microsoft's and Sybase's proprietary extension to the SQL (Structured Query Language) used to interact with relational databases. T-SQL expands on the SQL standard to include procedural programming, local variables, various support functions for string processing, date processing, mathematics, and changes to the DELETE and UPDATE statements. This language was used, but it is old now and therefore not considered in the method.

T-SQL has error checking and is globally accepted. Nevertheless, it works best with Microsoft SQL server and cannot use T-SQL in SQL environments.

PL SQL:

PL SQL or Procedural Language extensions to the Structured Query Language is an Oracle query language (Feuerstein & Pribyl, 2005, pp. 1–3). This language is immature and therefore not taken into consideration in the method.

It is highly structured, readable, accessible, well-integrated with Oracle databases, has high performance, and has API capabilities. However, it is not integrated with other databases, is not secure due to its vulnerabilities, such as SQL injection.

Spark SQL:

Spark SQL is a new module in Apache Spark that integrates relational processing with Spark's functional programming API (Armbrust et al., 2015, p. 1384). This language is immature and therefore not taken into consideration in the method.

Spark SQL can call complex analytics libraries in Spark, is API friendly, has high performance, has a variety of data sources available, can process large amounts of data, API capabilities, has integrations with Java, Scala, Python, and R. But has problems with small files and has big latency problems.

Scala:

Scala stands for scalable language. Scala can be applied to various programming tasks (Odersky, Spoon, & Venners, 2008, pp. 1–3). This language is immature and therefore not taken into consideration in the method.

Scala is easy to get into, has java integration, and is mainly for building large systems. It is scalable, secure due to Authentication and Authorization access control, and has API capabilities. However, it is immature and has a risk of abandonment.

2.5 Commonly used ETL tooling

Extract transform and load or ETL were introduced in the 1970s to integrate and load data into mainframes or supercomputers. This data was used for computation and analysis. From the late 1980s through the mid-2000s, it was the primary process for creating data warehouses that support business intelligence.

ETL is the process of transferring data from the source database to the destination data warehouse. In the process, there are three different sub-processes, which are E for Extract, T for Transform, and L for Load. The data is extracted from the source database, which can be any source in the extraction process, which is then transformed into the required format, such as changing portal codes from 1234aB to 1234AB and then loaded to the destination data warehouse; below a more detailed overview of the three sub-processes.

EXTRACT:
In the data extraction step, data is copied or exported from source locations to a staging area. The data can come from any structured or semi-structured or unstructured source such as SQL or NoSQL servers, CRM databases, ERP databases, text and document files, emails, web pages.

TRANSFORM:
In the staging area, the raw data is transformed to be useful for analysis and fit the eventual target data warehouse's schema. In this stage, filtering, cleansing, de-duplicating, validating, and authenticating the data is completed.

LOAD:
In this last step, the transformed data is moved from the staging area into a target data warehouse. Typically, this involves an initial loading of all data, followed by periodic loading of incremental data changes and, less often, full refreshes to erase and replace data in the warehouse.
There is a second form of ETL, ELT extract, load, and transform, which is the reverse of the ETL process's second and third steps. It copies or exports the data from the source

locations, but instead of moving it to a staging area for transformation, it loads the raw data directly to the target data store, where it can be transformed if needed.

In ELT, the target data store can be a data warehouse, but more often, it is a data lake, a large central store designed to hold both structured semi-structured and unstructured data.

Another part of this study is researching different ETL tools and finding strengths and weaknesses. The list is divided into two sub-chapters. The first one is the commonly used ETL tools, and the second one is about other ETL tools that are not commonly used.

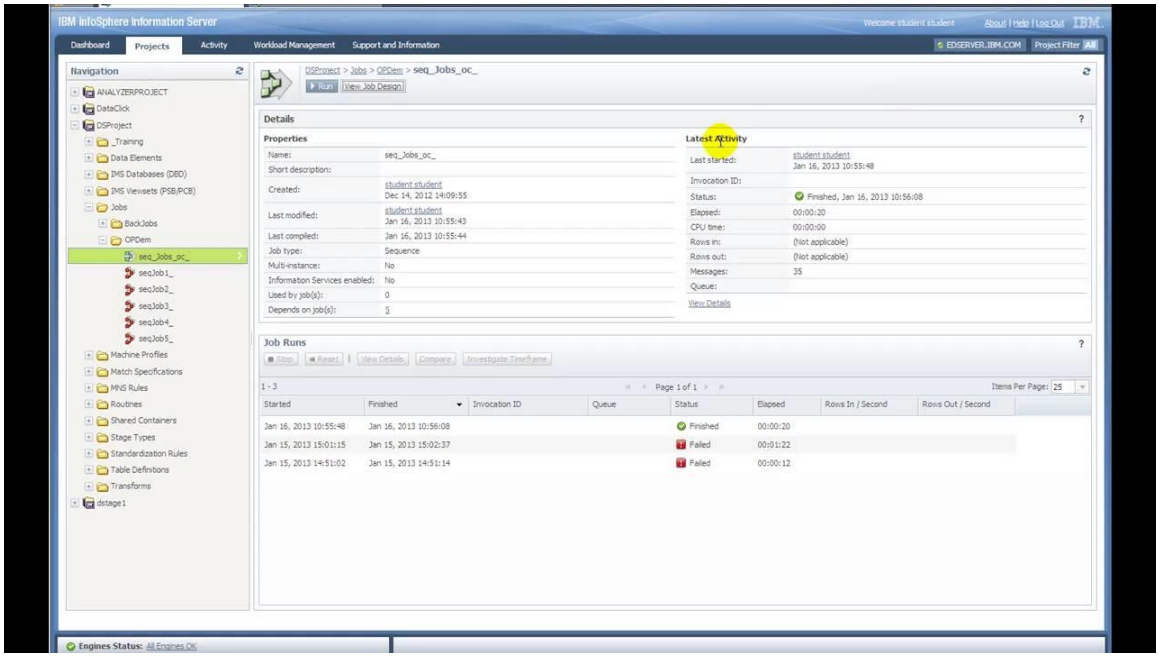
The commonly used ETL tools are listed below:

IBM Infosphere Information Server:

It is a product family that provides a unified data integration platform so that companies can understand, cleanse, transform and deliver trustworthy and context-rich information (Zhu et al., 2011, pp. 1–3).

Unique Selling Point: Increased investment in open source with new Apache Spark capabilities.

It can be integrated with Oracle, IBM DB2, and Hadoop System, it supports SAP via various plug-ins, ETL without coding, data source integration, integrated with BI, infrastructure rationalization and risk compliance, secure due to RBAC, API capabilities, flexible, fast due to symmetric multiprocessing (SMP) and massively parallel processing. Furthermore, does it have a built-in MDM system. Nevertheless, it lacks a robust web development environment, metadata propagation in jobs is somewhat complicated, slow configuration, and is expensive.



Details

Properties

Name:	seq_Jobs_oc_
Short description:	
Created:	student student Dec 14, 2012 14:09:55
Last modified:	student student Jan 16, 2013 10:55:43
Last compiled:	Jan 16, 2013 10:55:44
Job type:	Sequence
Multi-instance:	No
Information Services enabled:	No
Used by job(s):	0
Depends on job(s):	5

Latest Activity

Last started:	student student Jan 16, 2013 10:55:48
Invocation ID:	
Status:	✓ Finished, Jan 16, 2013 10:56:08
Elapsed:	00:00:20
CPU time:	00:00:00
Rows in:	(Not applicable)
Rows out:	(Not applicable)
Messages:	35
Queue:	

View Details

Job Runs

1 - 3 Page 1 of 1 Items Per Page: 25

Started	Finished	Invocation ID	Queue	Status	Elapsed	Rows In / Second	Rows Out / Second
Jan 16, 2013 10:55:48	Jan 16, 2013 10:56:08			✓ Finished	00:00:20		
Jan 15, 2013 15:01:15	Jan 15, 2013 15:02:37			Failed	00:01:22		
Jan 15, 2013 14:51:02	Jan 15, 2013 14:51:14			Failed	00:00:12		

Figure 19: Example of IBM Infosphere Information Server

Oracle Data Integrator:

Oracle Data Integrator (ODI) is a graphical environment to build and manage data integration (Lungu, 2015, p. 19).

Unique Selling Point: Best integration with Oracle databases and APIs.

ODI is mature, has high performance, data source integration, can perform complex transformations, it automatically identifies incorrect data and recycles it before moving into the target application. ODI supports databases like IBM DB2, Teradata, Sybase, Netezza, Exadata. Able to handle complex transformations, secure due to authentication, and is flexible. ODI has MDM capabilities due to a reverse engineering approach with Oracle Enterprise Metadata Management and API capabilities. However, it has a complex user interface GUI, a complex building tool, is expensive, needs in-depth technical knowledge, and can be faster.

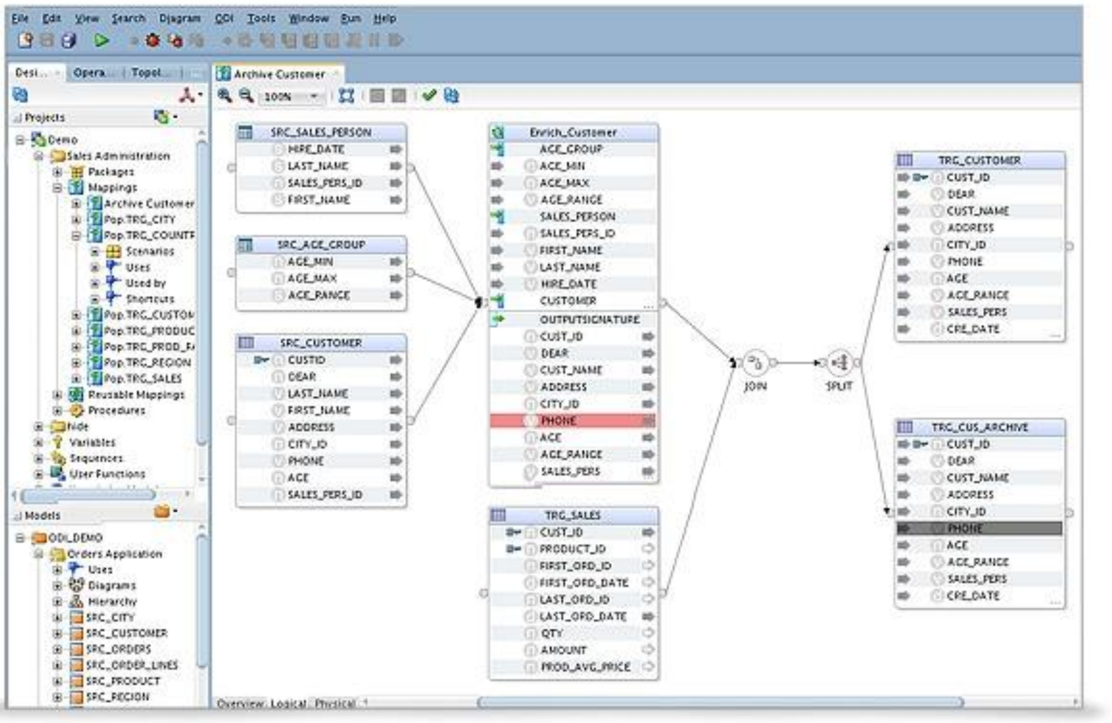


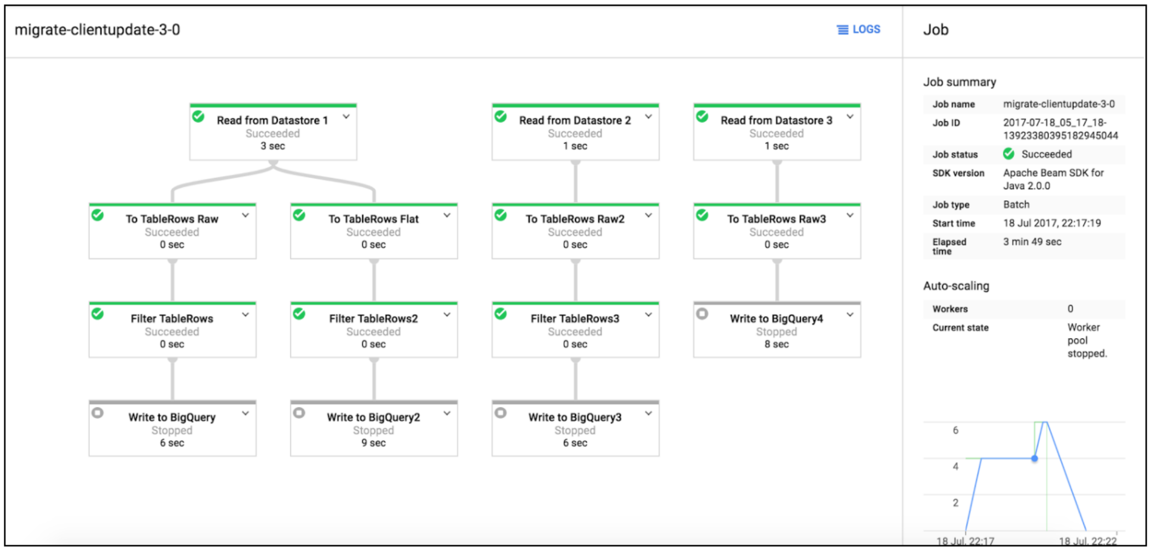
Figure 20: Example of Oracle Data Integrator

Google Dataflow:

Google Dataflow is a unified stream and batch data processing service (Palmer, Sferrazza, Just, & Najman, 2015, pp. 1–3).

Unique Selling Point: Offers the ability to create jobs based on templates.

Google Dataflow has reduced infrastructure administration, is fast due to its automatic scaling, SDK with native support for both batch and streaming modes, has API capabilities, is secure due to dataflow permissions, supports up to 60GB of data per minute for streaming, integration and has MDM with Google Data Catalog. Google Data Catalog is a metadata management service that quickly discovers, understands, and manages all data. Nevertheless, dataflow needs a new development approach, no low code available, not flexible, and need Data Proc for unstructured data.


migrate-clientupdate-3-0
LOGS
Read from Datastore 1 Succeeded 3 sec
Read from Datastore 2 Succeeded 1 sec
Read from Datastore 3 Succeeded 1 sec
To TableRows Raw Succeeded 0 sec
To TableRows Flat Succeeded 0 sec
To TableRows Raw2 Succeeded 0 sec
To TableRows Raw3 Succeeded 0 sec
Filter TableRows Succeeded 0 sec
Filter TableRows2 Succeeded 0 sec
Filter TableRows3 Succeeded 0 sec
Write to BigQuery4 Stopped 8 sec
Write to BigQuery Stopped 6 sec
Write to BigQuery2 Stopped 9 sec
Write to BigQuery3 Stopped 6 sec
6
4
2
18 Jul, 22:17
18 Jul, 22:22

Job

Job summary

Job name	migrate-clientupdate-3-0
Job ID	2017-07-18_05_17_18-13923380395182945044
Job status	✓ Succeeded
SDK version	Apache Beam SDK for Java 2.0.0
Job type	Batch
Start time	18 Jul 2017, 22:17:19
Elapsed time	3 min 49 sec

Auto-scaling

Workers	0
Current state	Worker pool stopped.

Figure 21: Example of Google Dataflow

Hadoop data lake:

Data Lake is defined as a methodology to approach the raw data, structured and non-structured within an enterprise and seen as an evolution of existing data architecture. The data is physically moved into one physical place, based on Hadoop technology; no change is made around the origin's format at the captured moment. Data Lake is more batch processing oriented based on MapReduce usage (Madera & Laurent, 2016b, pp. 1–3).

Unique Selling Point: It is open source.

Hadoop Data Lake is fast, less expensive, scalable, flexible, performs well on large amounts of data, secure due to access and privileges, has network isolation and data protection secure search. Furthermore, MDM with Cloudera Navigator helps manage and organize the data stored in the data lake (Quinto, 2018, p. 501). Cloudera Navigator is a data governance solution for Hadoop, offering capabilities such as data discovery, continuous optimization, audit, lineage, metadata management, and policy enforcement. Furthermore, good integration, low-code, and API capabilities are less mature, not real-time, and have no advanced analytics.

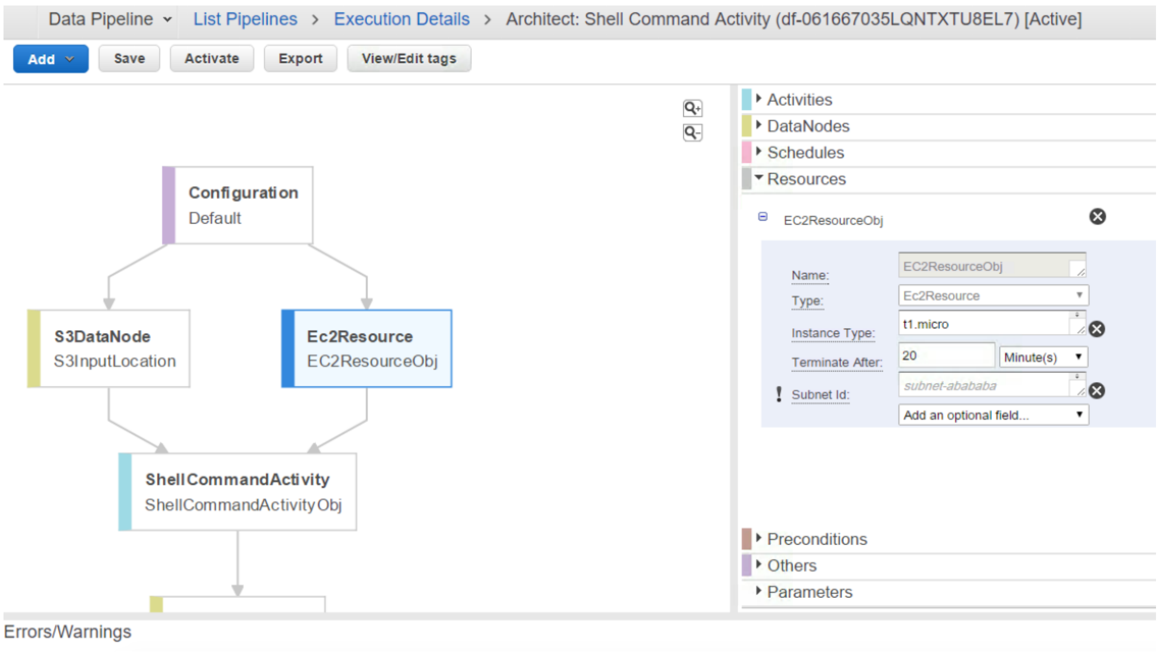
AWS Data Pipeline:

Data Pipeline as an ETL platform in the form of a web service with a control panel. This web service will help process and move data between different AWS compute and storage services and on-premises data sources.

Unique Selling Point: An infinitely scalable and cheap platform as low as 1-3cents a gig per month for hosting a Data Lake.

AWS Data Pipeline is simple to use, added security suite for used data, fault-tolerant architecture, fast, is flexible, has good error handling, is scalable, API, low-code and has MDM with EMRFS. EMRFS tracks consistency using a DynamoDB to track objects in Amazon S3 synced with or created by EMRFS. The metadata is used to track all operations. This metadata is used to validate whether the objects or metadata received from Amazon

S3 matches what is expected. However, it lacks integration with third-party data sources, difficult to use with on-premises data sources, fast but not fast on large amounts of data.



Name: EC2ResourceObj
Type: Ec2Resource
Instance Type: t1.micro
Terminate After: 20 Minute(s)
Subnet Id: subnet-abababa

Figure 22: Example of AWS Data Pipeline

Informatica PowerCenter:

PowerCenter provides an environment that allows loading data into a centralized location. Data can be extracted from multiple sources, transform the data according to business logic, built in the client application, and load the transformed data into a file and relational targets (Informatica Corporation, 2014).

Unique Selling Point: It has add-on packages.

PowerCenter is easy to use, good support, fast due to parallel processing, data quality monitoring, and data migration capabilities. PowerCenter has a single point of control, ensuring a high degree of security and MDM with Enterprise data catalog. Enterprise Data Catalog is a data catalog that provides a machine-learning-based discovery engine to scan and catalog data assets.

It is powered by the CLAIRE engine, which provides intelligence by leveraging metadata to deliver recommendations, suggestions, and automation of data management tasks. Furthermore, API capabilities, flexible, has a GUI, and low code. However, it lacks integration with other languages such as R, python, java, no reporting functionality, many settings interfaces, no AI capabilities, and connectors not working well, such as Hadoop connector.

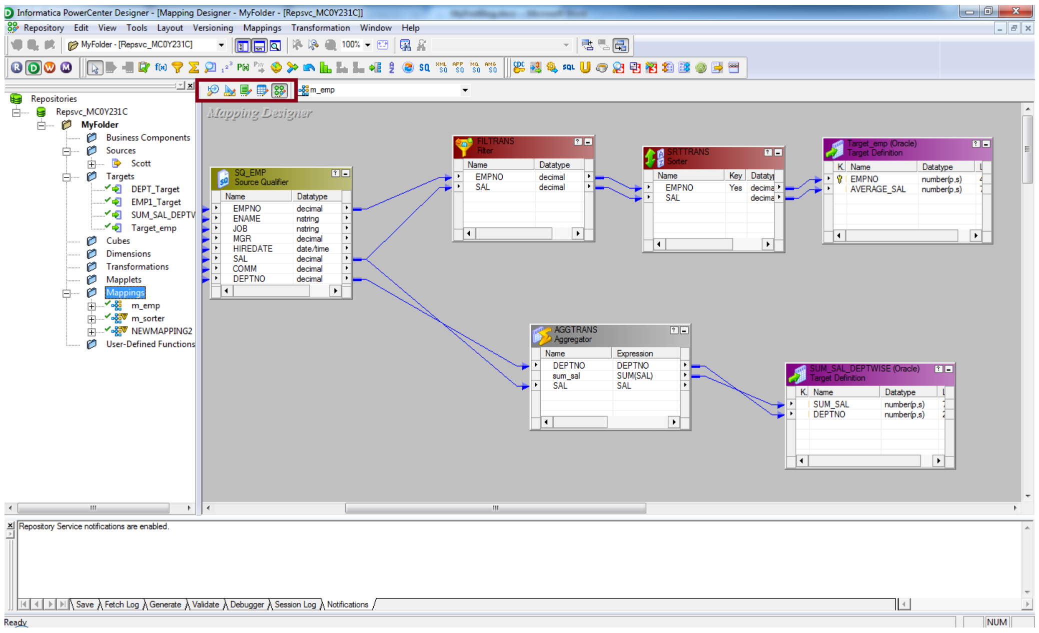

Informatica PowerCenter Designer - [Mapping Designer - MyFolder - [Repsvc_MC0Y231C]]
Repository Edit View Tools Layout Versioning Mappings Transformation Window Help
Mapping Designer
Repositories
Repsvc_MC0Y231C
MyFolder
Business Components
Sources
Scott
Targets
DEPT_Target
EMP1_Target
SUM_SAL_DEPTV
Target_emp
Cubes
Dimensions
Transformations
Mapplets
Mappings
m_emp
m_sorter
NEWMAPPING2
User-Defined Functions
SQ_EMP Source Qualifier
EMPNO ENAME JOB MGR HIREDATE SAL COMM DEPTNO
FILTRANS Filter
SRTTRANS Sorter
Target_emp (Oracle) Target Definition
EMPNO AVERAGE_SAL
AGGTRANS Aggregator
DEPTNO sum_sal SAL
SUM_SAL_DEPTWISE (Oracle) Target Definition
SUM_SAL DEPTNO
Repository Service notifications are enabled.
Save Fetch Log Generate Validate Debugger Session Log Notifications
Ready
NUM

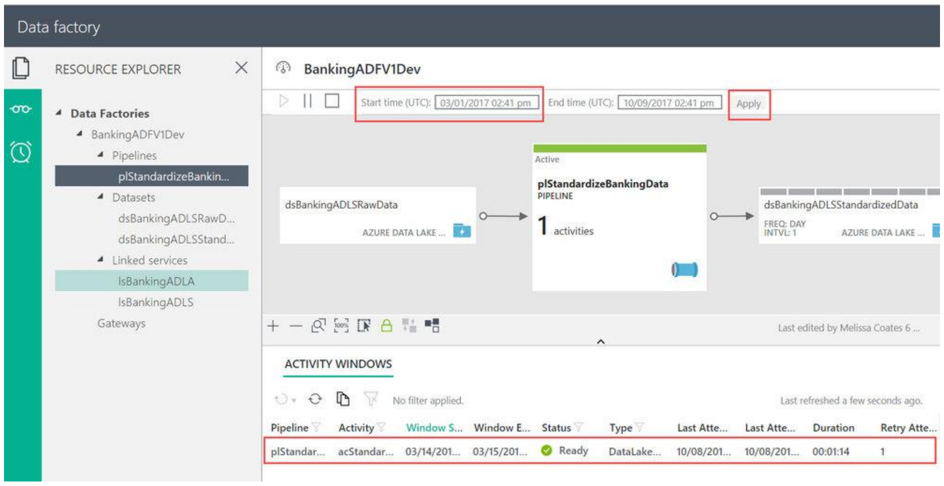
Figure 23: Example of Informatica PowerCenter

Microsoft Azure Data Factory:

Azure Data Factory is a hybrid data integration service offering a code-free experience. Azure Data Factory is a data integration service explicitly designed to collaborate with existing services for the movement, transformation, and processing of raw data from disparate systems and transform it into useful information (Klein, 2017, pp. 1–3). For Microsoft Azure Data Factory, the Azure Data Catalog is most suitable to add metadata management. Data Catalog tags data with metadata stored in the Azure Data Catalog for easy discovery (Klein, 2017b, pp. 1–3).

Unique Selling Point: It is a unique data integration service that manages and automates the movement and transformation of data across an organization with low-code capabilities.

Azure Data Factory is fast due to parallel processing, has a visual drag-and-drop UI, has SSIS migration to cloud, multiple language support, hybrid data movement, and transformation. Furthermore, does Data Factory has support, a lot of data source integration is secure due to AAD, MI, and VNET. Data Factory has MDM due to Data Catalog, is flexible, has API capabilities, can program in low-code, and is stable, but scratch configuration could be challenging. So, it needs an understanding of the tool to use it at its best.



Start time (UTC): 03/01/2017 02:41 pm
End time (UTC): 10/09/2017 02:41 pm

Last edited by Melissa Coates 6 ...

No filter applied.

Last refreshed a few seconds ago.

Pipeline	Activity	Window S...	Window E...	Status	Type	Last Atte...	Last Atte...	Duration	Retry Atte...
plStandar...	acStandar...	03/14/201...	03/15/201...	✓ Ready	DataLake...	10/08/201...	10/08/201...	00:01:14	1

Figure 24: Example of Azure Data factory

The less commonly used ETL tooling are:

SAP BW ETL:

SAP BW ETL provides a collection of objects and tools that allow users to import, export, and transform heterogeneous data between one or multiple types of data formats, such as MS Excel, text files, SAP ECC (Lomet & Chaudhuri, 1999, p. 38). This tool was used, but it is old now and not considered in the method.

SAP BW ETL runs on various third-party RDBMSs, provides open Business Application Programming Interfaces (BAPIs) for data loading, and provides a pre-configured metadata repository InfoCube catalog, report catalog, and information source catalog. Furthermore, shipped with many pre-defined InfoCubes for typical business applications, e.g., market segment analyses, profitability analyses, stock inventory analyses, and corporate indicator systems. It has a GUI, sharing functions and Microsoft Office compatibility, data visualization, and analytics applications, extending with SAP BI security for role-based security and API capabilities. However, this tool has expensive licensing and is not able to perform on high loads.

Talend:

Talend is a French software vendor specialized in open source integration. The company democratizes integration through its products and enables IT users and organizations to deploy complex architectures in more straightforward and comprehensive ways. (Azarmi, 2014, pp. 1–3). Talend is provided for on-premises deployment and the Software as a Service (SaaS) delivery model. Talend Open Studio is used for integrating operational systems as well as an ETL tool for Data Warehousing, Business Intelligence, and data migration. (Katragadda, Sremath Tirumala, & Nandigam, 2015, pp. 1–3). This tool is not mainly considered an ETL tool and, therefore, not considered in the method.

Talend is easy to use, helps business users graphically design their business processes, has a high-volume integration, parallelization feature, data source integration, free open source ETL tool, and secure and API capabilities. However, to use more inside tools like machine learning add-ons, licenses need to be bought. It is developed as a product for individual use only, and so it is not possible to have more than one user (not just one user at a time but just one user per system); the free version does not support automation of tasks like scheduling, routing data, Lack of any commercial support and not efficient.

Microsoft SQL Server Integrated Services (SSIS):

SQL Server Integration Services (SSIS) is one of the Business Intelligence tools developed by Microsoft to ease and automate the ETL process (Katragadda, Sremath Tirumala, & Nandigam, 2015, pp. 1–3). The data integration is much faster as the integration process, and data transformation is processed in the memory. This tool is not mainly used as an ETL tool anymore and, therefore, not considered in the mechanism.

SSIS automates the SQL Server Maintenance Plan by creating an SSIS package. SSIS can handle data from heterogeneous data sources at the same package. Data sources can be diverse, including custom or scripted adapters, deficient in cost compared to the famous Informatica Power Center and almost offers everything needed to build the ETL solution. It is easier to maintain with package configuration, better for complex transformations, multi-step operations, aggregating data from different data sources or types, and structured exception handling. Furthermore, data can be loaded in parallel to many varied destinations, build ETL solutions with very minimum background knowledge, very easy to install and configure, offers comprehensive documentation and support, provides best practices, debugging capabilities, easy to use, API capabilities and secure due to threat and vulnerability mitigation. However, to see the package execution report, the Management Studio is needed rather than being published to reporting services or another way. SSIS cannot support non-windows operating systems; SSIS is more suitable for enterprise and may not be cost-efficient for small businesses, no integration with other languages, need more sources, and sometimes an error is returned without knowing what error it is.

MongoDB:

MongoDB is an open-source NoSQL database developed in C++ (Abramova & Bernardino, 2013, p. 17). This tool is not an ETL tool and, therefore, not taken into consideration in the method.

MongoDB is efficient, fast, durable, secure due to authentication, access control, and encryption. Has API capabilities and integration but uses much internal memory, lacks the support of join queries, has no MDM.

Cloudera Apache Hive.

Hive is a data warehouse software that facilitates queries and manages an extensive data set in distributed storage. Hive runs on top of Hadoop (Fuad, Erwin, & Ipung, 2014, p. 298). This tool is like Hadoop Data Lake and, therefore, not taken into consideration in the method.

Apache Hive can process petabytes of data, integration, scalable, fast, MDM, API capabilities, but the configuration is a bit tricky, designed for analytical purposes not for transactional purposes, lack of security, lack of support and runtime errors.

2.6 Commonly used visualization tooling

Visualization tooling allows users to create charts, images, diagrams, and dashboards to communicate a message. It is a graphical representation of data and information. In today's world, data visualization tools and technologies are essential to analyze massive amounts of information and make data-driven decisions.

The commonly used visualization tooling are:

Power BI:

Power BI is a tool that displays interactive dashboards that can be created and updated from many different data sources (NEGRUT, 2012, pp. 1–3).

Unique Selling point: Live dashboards, trend analysis, sharing capabilities and data source integration.

Power BI is not expensive, easy to use, receives constant updates, integrates with many sources, and can handle large amounts of data, sharing and collaboration features, fast, secure due to RBAC authentication, and RLS. Furthermore, Power BI has API capabilities, has a GUI, low-code and code capabilities, has an internal MDM if needed, is flexible but is limited in the free version capabilities, table relationship is a bit lagging, not able to handle large amounts of data in the free version and the dataset has a max of one GB in the free version and ten GB in the pro version.

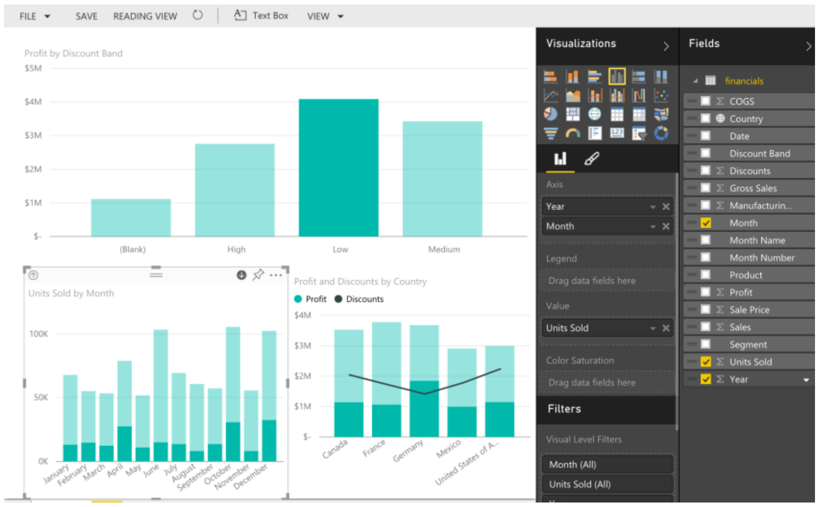


Figure 25: Example of Power BI

Tableau:

Tableau is a software that can help users explore and understand their data by creating interactive visualizations (Ko & Chang, 2017, pp. 1–3)

Unique Selling Points: Perform queries without a single line of code, sharing capabilities.

Tableau connects users with various data sources and enables them to create data visualizations by making charts, maps, dashboards, and stories through a simple drag and drop interface. Tableau has a tremendous and reliable speed due to TDE files. Furthermore, doe Tableau offers API capabilities and is flexible with the MarkLogic feature. Tableau is secure due to IT controls, which are regularly audited by independent firms. Extra security is added due to database login with authentication. Finally, does Tableau offer MDM with VizQL Model, and can be used with low-code and code. However, Tableau is still not widely used and has basic security with no RLS.

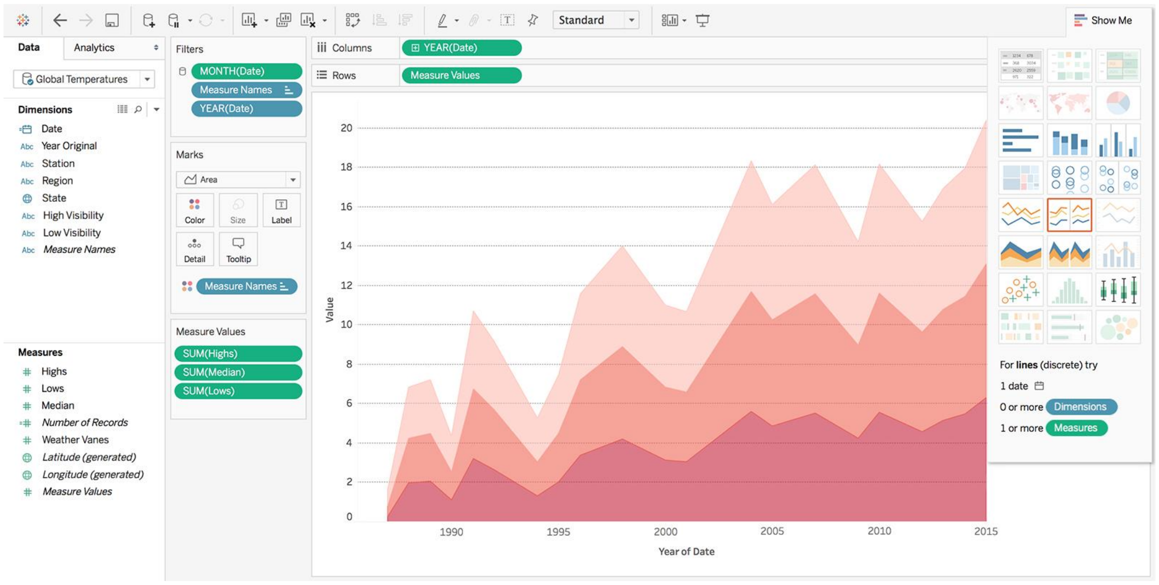


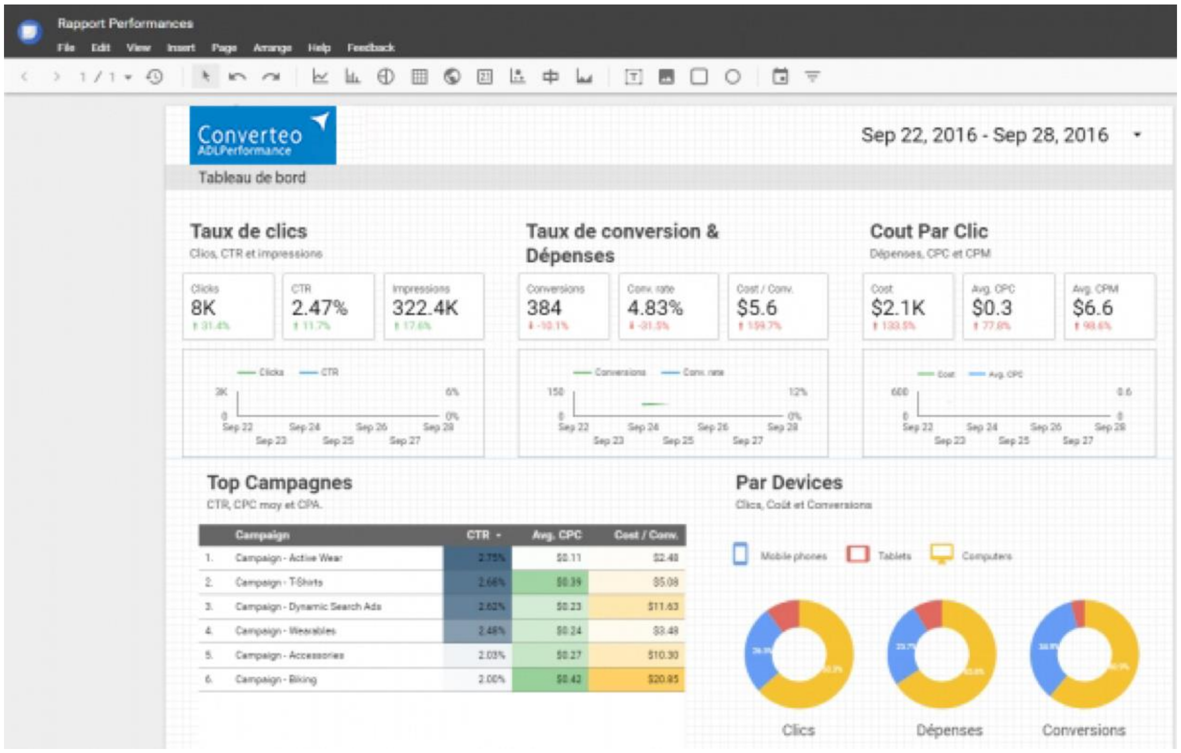
Figure 26: Example of Tableau

Google data studio:

Google Data Studio is a new data visualization program designed as a user-friendly tool for representing complex data sets attractively and clearly (Snipes, 2018, pp. 1–3).

Unique Selling Points: People who are using the Google platform are familiar and easy to use.

Google Data Studio is recognizable to anyone who works with the Google office suite, free to anyone with a Google account, has an interactive GUI, sharing and collaboration features, unlimited amount of data can be used, fast, API capabilities, secure due to physical security, encryption, incident management, identity, and access management. Furthermore, does it support code and low code. Nevertheless, it cannot comply with IRB requirements for protecting personally identifiable data. It cannot modify underlying data, offers fewer calculation and visualization options, limited data sources, 100MB file size limit per dataset, not flexible, and no metadata feature.



Top Campagnes

CTR, CPC moy et CPA.

	Campaign	CTR	Avg. CPC	Cost / Conv.
1.	Campaign - Active Wear	2.75%	$0.11	$2.48
2.	Campaign - T-Shirts	2.66%	$0.39	$5.08
3.	Campaign - Dynamic Search Ads	2.62%	$0.23	$11.63
4.	Campaign - Wearables	2.48%	$0.24	$3.48
5.	Campaign - Accessories	2.03%	$0.27	$10.30
6.	Campaign - Biking	2.00%	$0.42	$20.85

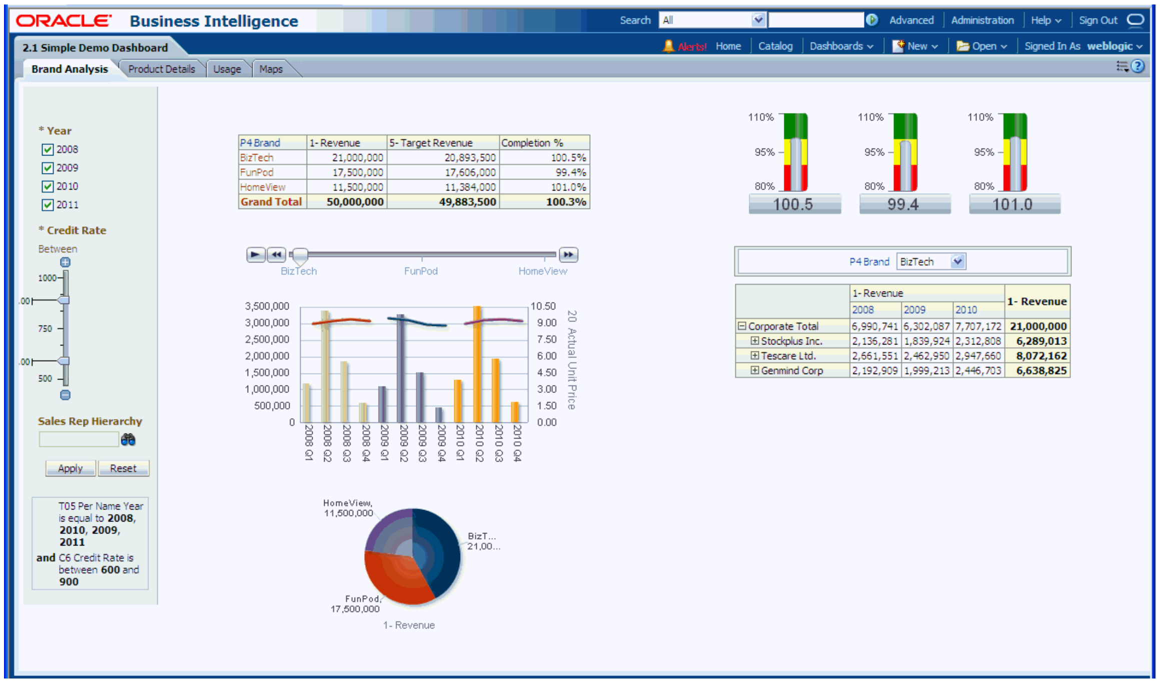
Figure 27: Example of Google Data studio

Oracle Business Intelligence:

This solution was designed to address the entire spectrum of analytical requirements facing businesses, including information access, analysis, and reporting. (Bozdoc, 2011, pp. 1–3).

Unique Selling Points: Aggregate content from various sources, including the Internet, shared file servers, and document repositories.

Oracle Business Intelligence has ad hoc analysis, enterprise reporting, Microsoft Office and other integration, API capabilities, security due to Authentication and authorization, and user groups. It has a GUI for low-code and code, but has high prices for large configurations, not fast due to the number of users, lacks visualization, needs technical knowledge to use, not flexible, and no MDM available.



T05 Per Name Year is equal to **2008**, **2010**, **2009**, **2011**
and C6 Credit Rate is between **600** and **900**

P4 Brand	1- Revenue	5- Target Revenue	Completion %
BizTech	21,000,000	20,893,500	100.5%
FunPod	17,500,000	17,606,000	99.4%
HomeView	11,500,000	11,384,000	101.0%
Grand Total	**50,000,000**	**49,883,500**	**100.3%**

P4 Brand BizTech

	1- Revenue			**1- Revenue**
	2008	2009	2010	
⊟ Corporate Total	6,990,741	6,302,087	7,707,172	**21,000,000**
⊞ Stockplus Inc.	2,136,281	1,839,924	2,312,808	**6,289,013**
⊞ Tescare Ltd.	2,661,551	2,462,950	2,947,660	**8,072,162**
⊞ Genmind Corp	2,192,909	1,999,213	2,446,703	**6,638,825**

Figure 28: Example of Oracle Business Intelligence

Amazon Quicksight:

Amazon QuickSight is a fast, cloud-powered BI service that makes it easy to build visualizations, perform ad-hoc analysis, and quickly get business insights from massive data (Mathavi, Jeyarubi, Ganesh, Tamilselvi, & Karthi, 2018, pp. 1–3).

Unique Selling Points: Pay only for what you use.

Quicksight is superfast due to parallel and in-memory processing. Furthermore, does it use a Calculation Engine (SPICE), which uses a combination of columnar storage, in-memory technologies enabled through hardware, machine code generation, and data compression to allow users to run interactive queries on large datasets and get rapid responses. It is easy to use, secure due to RLS, compliance programs, AWS WAF logs, training and awareness, integration with many sources, has a GUI which allows low code. However, still immature, sharing to non-AWS users not possible, not available on android, limited APIs and extensions, the standard edition has a limit of 25GB per dataset, and the enterprise edition has a limit of 500GB per dataset and is not flexible and no MDM available.

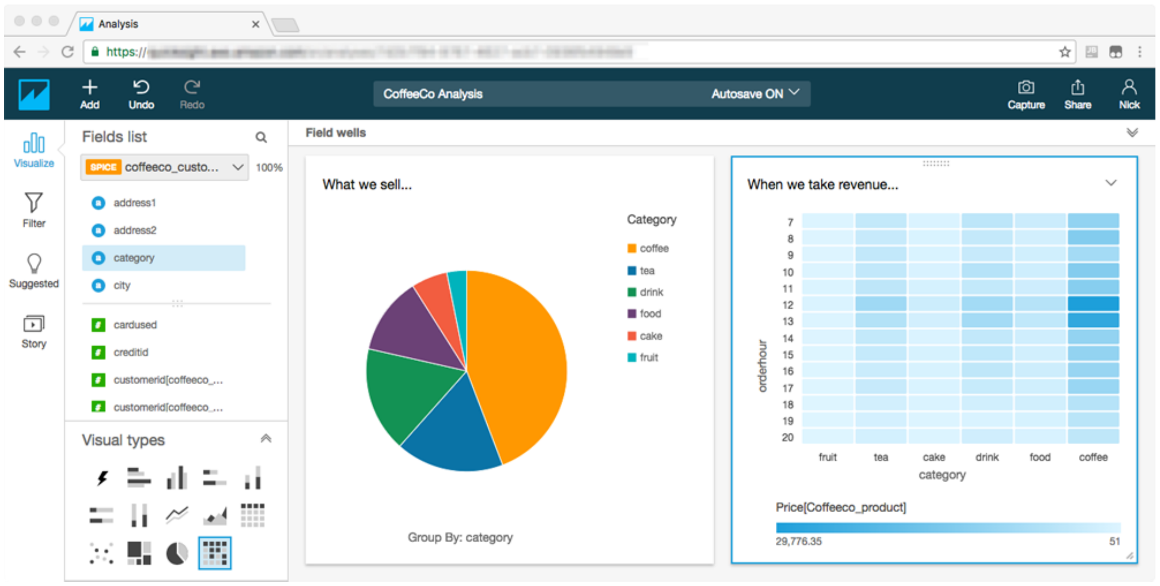


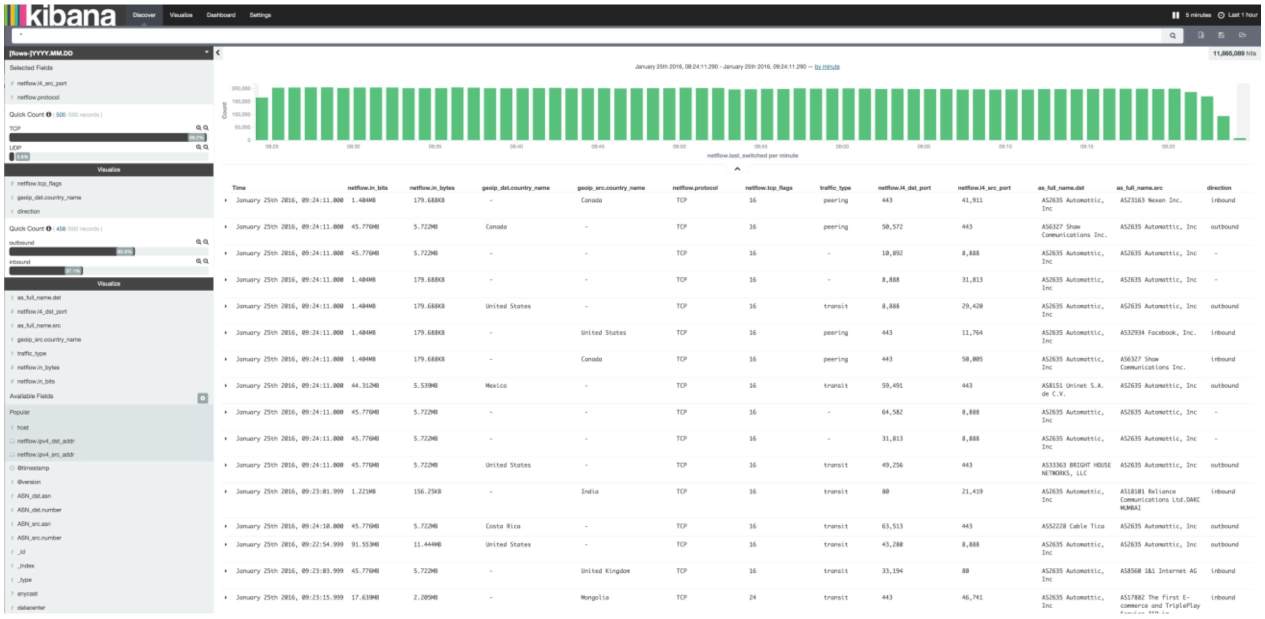
Figure 29: Example of Amazon Quicksight

Kibana:

Kibana was designed as a visualization platform for Elasticsearch. It provides a web-based interface for search, view, and analyzing data stored in the Elasticsearch cluster and is part of the Elastic stack Elasticsearch, Logstash, and Kibana (Bajer, 2017, p. 67).

Unique Selling Points: Open-source.

Kibana is fast with the Elasticsearch engine, is interactive, efficient, easy to extend to needs, able to show massive volumes of data, custom visuals, secure data sharing, secure due to X-Pack RBAC, has API capabilities, is even faster with NGINX, flexible with Elasticsearch index and is low-code. However, it has no user management available out of the box, works on top of elastic only, issues with large datasets, less integration and no MDM available.



Time	netflow.in_bits	netflow.in_bytes	geoip_dst.country_name	geoip_src.country_name	netflow.protocol	netflow.tcp_flags	traffic_type	netflow.l4_dst_port	netflow.l4_src_port	as_full_name.dst	as_full_name.src	direction
January 25th 2016, 09:24:11.000	1.404MB	179.688KB	-	Canada	TCP	16	peering	443	41,911	AS2635 Automattic, Inc	AS23163 Nexen Inc.	inbound
January 25th 2016, 09:24:11.000	45.776MB	5.722MB	Canada	-	TCP	16	peering	50,572	443	AS6327 Shaw Communications Inc.	AS2635 Automattic, Inc	outbound
January 25th 2016, 09:24:11.000	45.776MB	5.722MB	-	-	TCP	16	-	10,092	8,888	AS2635 Automattic, Inc	AS2635 Automattic, Inc	-
January 25th 2016, 09:24:11.000	1.404MB	179.688KB	-	-	TCP	16	-	8,888	31,813	AS2635 Automattic, Inc	AS2635 Automattic, Inc	-
January 25th 2016, 09:24:11.000	1.404MB	179.688KB	United States	-	TCP	16	transit	8,888	29,420	AS2635 Automattic, Inc	AS2635 Automattic, Inc	outbound
January 25th 2016, 09:24:11.000	1.404MB	179.688KB	-	United States	TCP	16	peering	443	11,764	AS2635 Automattic, Inc	AS32934 Facebook, Inc.	inbound
January 25th 2016, 09:24:11.000	1.404MB	179.688KB	-	Canada	TCP	16	peering	443	50,005	AS2635 Automattic, Inc	AS6327 Shaw Communications Inc.	inbound
January 25th 2016, 09:24:11.000	44.312MB	5.539MB	Mexico	-	TCP	16	transit	59,491	443	AS8151 Uninet S.A. de C.V.	AS2635 Automattic, Inc	outbound
January 25th 2016, 09:24:11.000	45.776MB	5.722MB	-	-	TCP	16	-	64,582	8,888	AS2635 Automattic, Inc	AS2635 Automattic, Inc	-
January 25th 2016, 09:24:11.000	45.776MB	5.722MB	-	-	TCP	16	-	31,813	8,888	AS2635 Automattic, Inc	AS2635 Automattic, Inc	-
January 25th 2016, 09:24:11.000	45.776MB	5.722MB	United States	-	TCP	16	transit	49,256	443	AS33363 BRIGHT HOUSE NETWORKS, LLC	AS2635 Automattic, Inc	outbound
January 25th 2016, 09:23:01.999	1.221MB	156.25KB	-	India	TCP	16	transit	80	21,419	AS2635 Automattic, Inc	AS18101 Reliance Communications Ltd.DAKC MUMBAI	inbound
January 25th 2016, 09:24:10.000	45.776MB	5.722MB	Costa Rica	-	TCP	16	transit	63,513	443	AS52228 Cable Tica	AS2635 Automattic, Inc	outbound
January 25th 2016, 09:22:54.999	91.553MB	11.444MB	United States	-	TCP	16	transit	43,280	8,888	AS2635 Automattic, Inc	AS2635 Automattic, Inc	outbound
January 25th 2016, 09:23:03.999	45.776MB	5.722MB	-	United Kingdom	TCP	16	transit	33,194	80	AS2635 Automattic, Inc	AS8560 1&1 Internet AG	inbound
January 25th 2016, 09:23:15.999	17.639MB	2.205MB	-	Mongolia	TCP	24	transit	443	46,741	AS2635 Automattic, Inc	AS17882 The first E-commerce and TriplePlay	inbound

Figure 30: Example of Kibana

MicroStrategy:

MicroStrategy is a vendor in BI products that supports interactive dashboards, scorecards, highly formatted reports, ad hoc queries, thresholds and alerts, and automated report distribution. (Anoshin, Rana, & Ma, 2016, pp. 1–3).

Unique Selling Points: easily integrated into the broader enterprise landscape.

MicroStrategy is easy to use, fast due to in-memory processing, is secure, integration with different data sources, two TB data size availability, has monitoring features, has a GUI, is stable, secure due to ACL, and has API capabilities. Nevertheless, with more than two TB of data, timeout errors are thrown, sharing not available, no support, need more visualizations, not flexible and no MDM available.

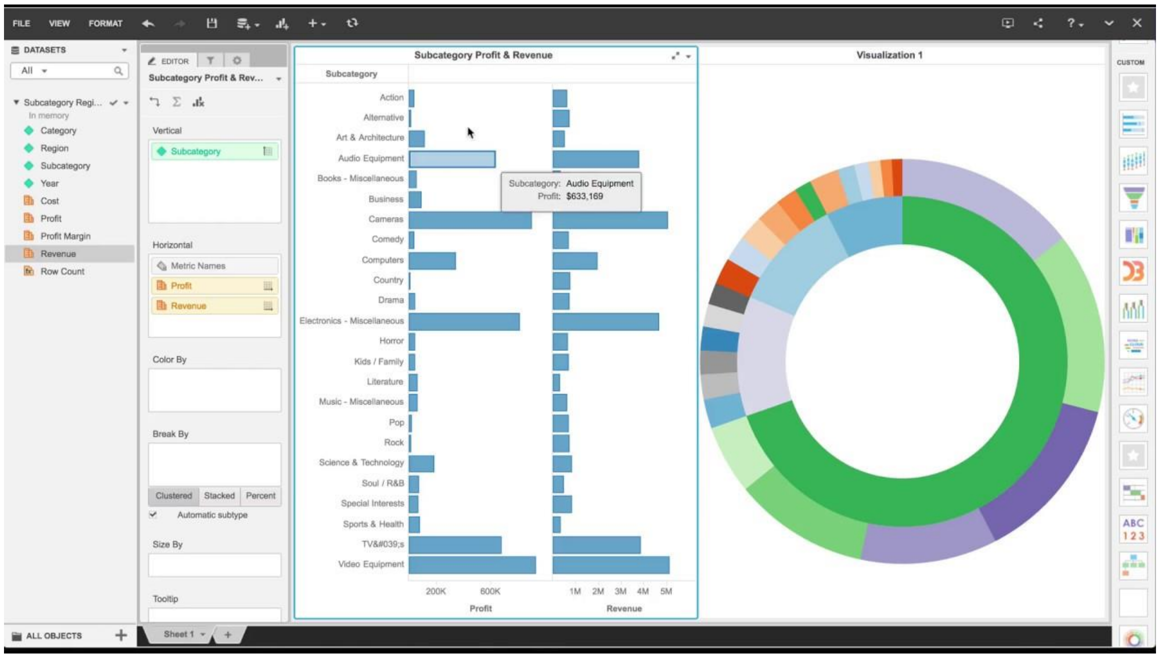


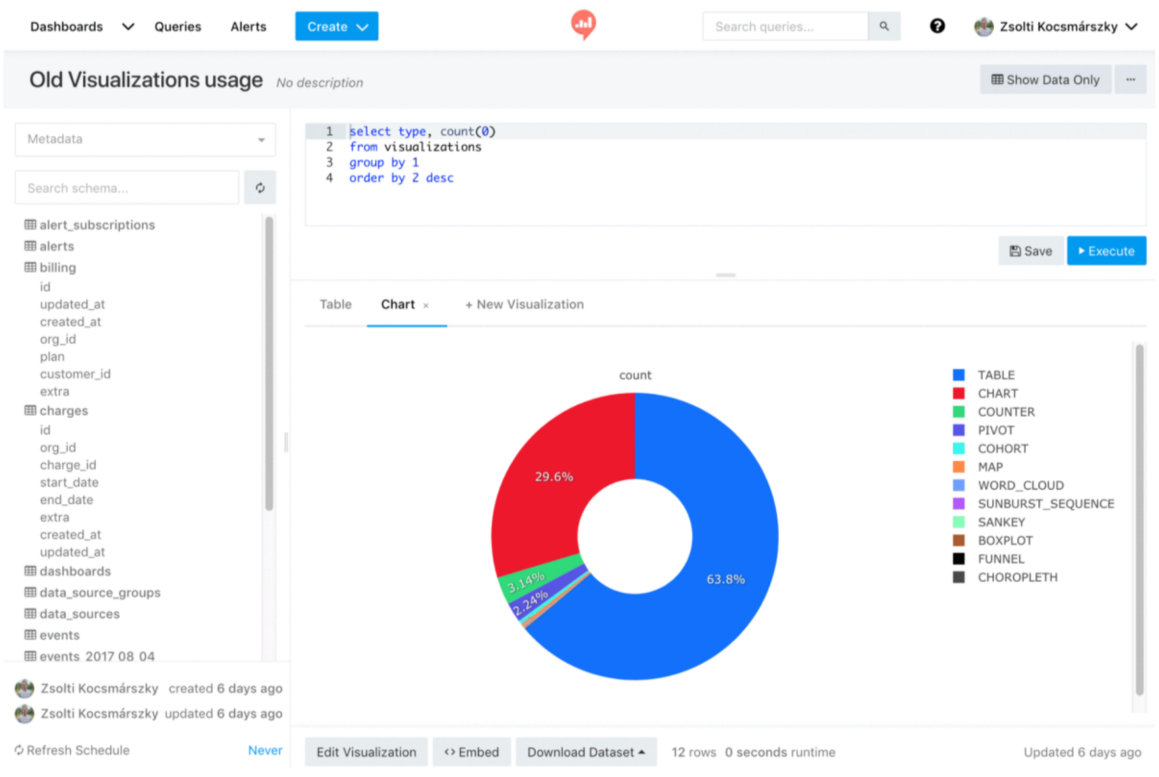
Figure 31: Example of MicroStrategy

Redash:

It is an open-source tool used to create, visualize, and share queries and dashboards (Leibzon & Leibzon, 2018, pp. 1–3).

Unique Selling Points: SQL client makes it easy to browse data in-app.

Redash is easy to use and to setup with a GUI. No installation is needed because Redash is browser-based and uses SQL templates. Furthermore, many data source integration is available; it has API capabilities; it is easy to export data to different formats and is fast due to the delta engine of Databricks. The security is developed due to complete industry standards and penetration tests. Low-code and code are available, and MDM possible with Databricks. However, Redash has scalability problems, and technical knowledge is required and is not flexible.



```
select type, count(0)
from visualizations
group by 1
order by 2 desc
```

Figure 32: Example of Redash

The less commonly used visualization tooling is:

IBM Cognos:
Cognos provides a unified workspace for business intelligence and analytics that the entire organization can use to answer critical business questions. (Browne et al., 2010, pp. 1–3). Deliver timely, accurate, and actionable performance management solutions to users across the enterprise (Oehler, Gruenes, & Ilacqua, 2012, pp. 1–3). This tool is old and, therefore, not taken into consideration in the method.

Cognos has an easy view, good collaboration, transparency, and accountability, access data everywhere (mobile devices), link dashboards to workflows, different data sources available, flexible, drag and drop feature, secure due to data protection protocol, LDAP and active directory. Furthermore, it is fast due to performance monitoring and tuning in the database, application server, web server, IBM Cognos BI, and API capabilities. Nevertheless, only a few sites and communities exist, most experience has been gained by intuition and trial, and error, the file size limit of 100MB per user, no predictive analytics, and reports themselves can become very large.

SAS:
SAS is a software pack with various visualization capabilities implemented in the product suite, an interface that allows interaction with charts (ARGHIR, DUŞA, & ONUŢĂ, 2019, p. 87). This tool is not widely used anymore and, therefore, not taken into consideration in the method.

Various visualization capabilities implemented in the product suite, interactive, allows the creation of basic queries and reports, handle large databases, debug feature, secure due to data security feature and has a GUI but has high costs.

Qlik:
Qlik provides a possibility for end-users to use integrated ETL and to construct their data schema themselves. (Grabova, Darmont, Chauchat, & Zolotaryova, 2010, p. 39). This tool lacks many features and is therefore not taken into consideration in the method.

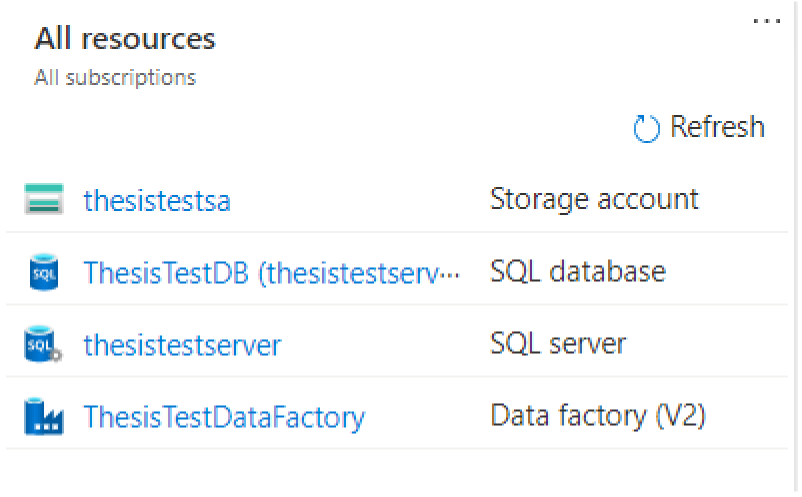
Qlik provides a clean interface to analysts, removes the need to pre-aggregate data, can change analysis axes any moment at any level of query detailing, ability to connect tables, flexible, integration, API automatically. However, it lacks a unified metadata view, lack of predicting models, lacks advanced visualization features to help them graphically wade through complex data, no customization, lack of speed, the data security model is complicated, data greater than ten GB cannot be uploaded and a glitch in user access which gives access to unauthorized users.

2.7 Working example

The chapters before explained the different tooling with advantages and disadvantages. To illustrate how such an environment could work together, we developed a working example.

It starts with extracting the data from different sources, which is the E step in the ETL process. Next, the data is processed and, in this case, joined in the T step of the ETL process. After these steps, the data is loaded into a SQL database to be visualized in Power BI in the L step of the ETL process.

Figure 33 shows the resources needed for this example. First, a storage account is created to store any data. Furthermore, a SQL server, SQL database, and a Data Factory are created. The SQL server is used to host the SQL database, the SQL database is used to store transformed data, and the Data Factory is used to create the pipeline the user wants.



thesistestsa	Storage account
ThesisTestDB (thesistestserv···	SQL database
thesistestserver	SQL server
ThesisTestDataFactory	Data factory (V2)

Figure 33: All resources

In the storage account, a container is created to store input. In this case, two files with car data are stored and named input.

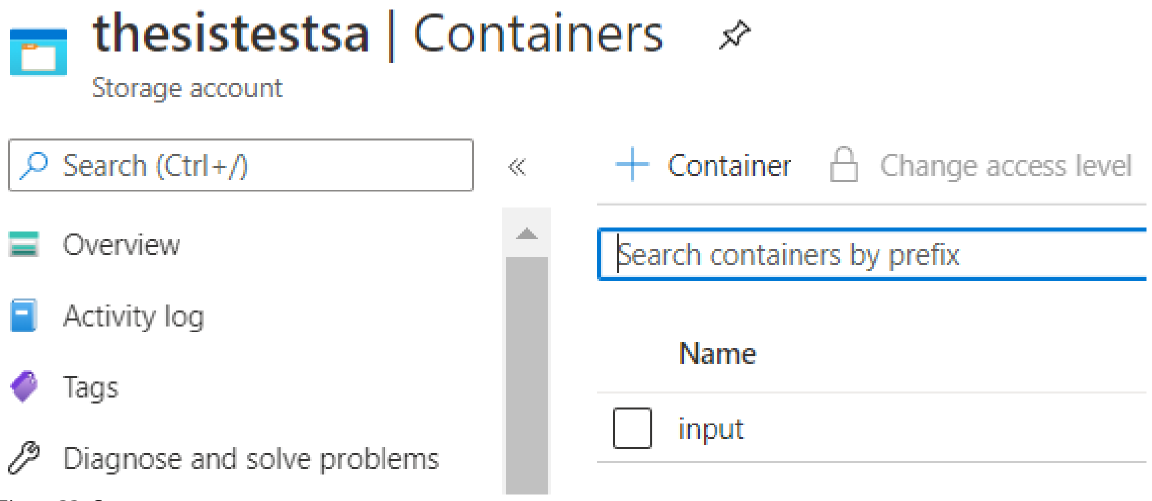
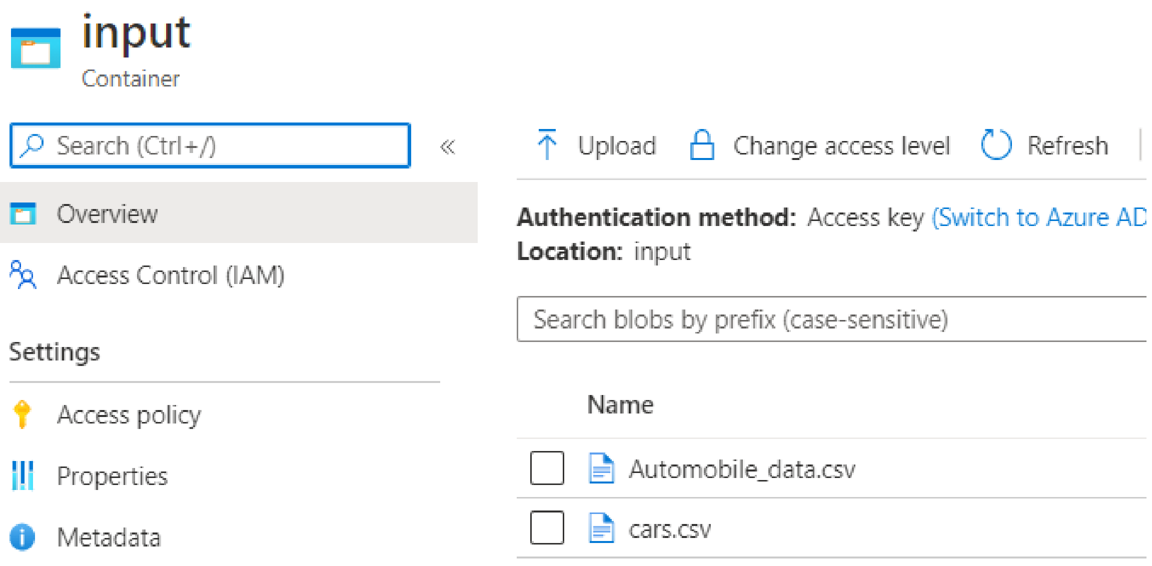


Figure 32: Storage account



Authentication method: Access key (Switch to Azure AD
Location: input

Name
Automobile_data.csv
cars.csv

Figure 34: Stored files

Figure 35 shows the dataflow created in the data factory. This dataflow imports the two car files and joins them in one. Next, the output is exported to a SQL database.

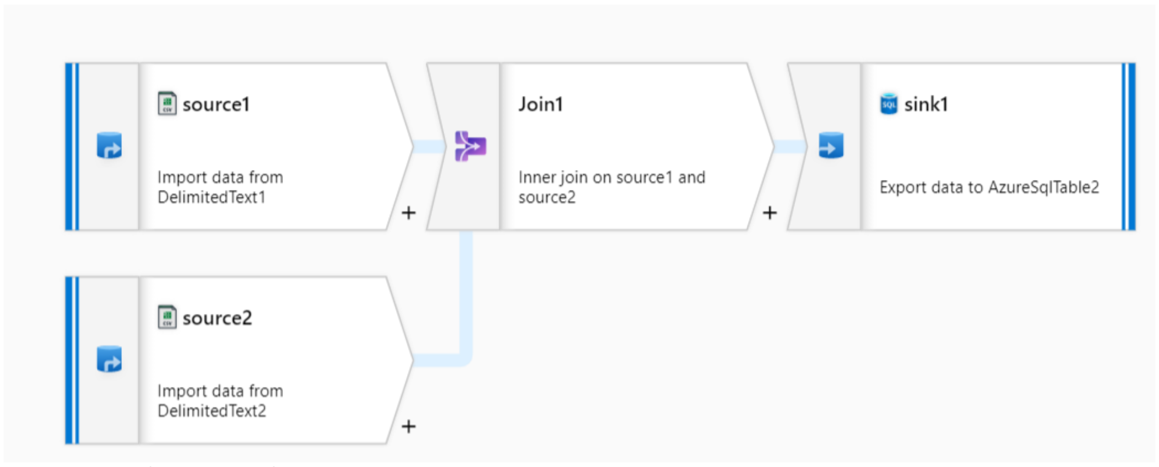

source1
Import data from DelimitedText1
Join1
Inner join on source1 and source2
sink1
Export data to AzureSqlTable2
source2
Import data from DelimitedText2

Figure 35: Dataflow in a data factory

The picture below shows the pipeline created. This pipeline is created to execute the dataflow created in the previous step.

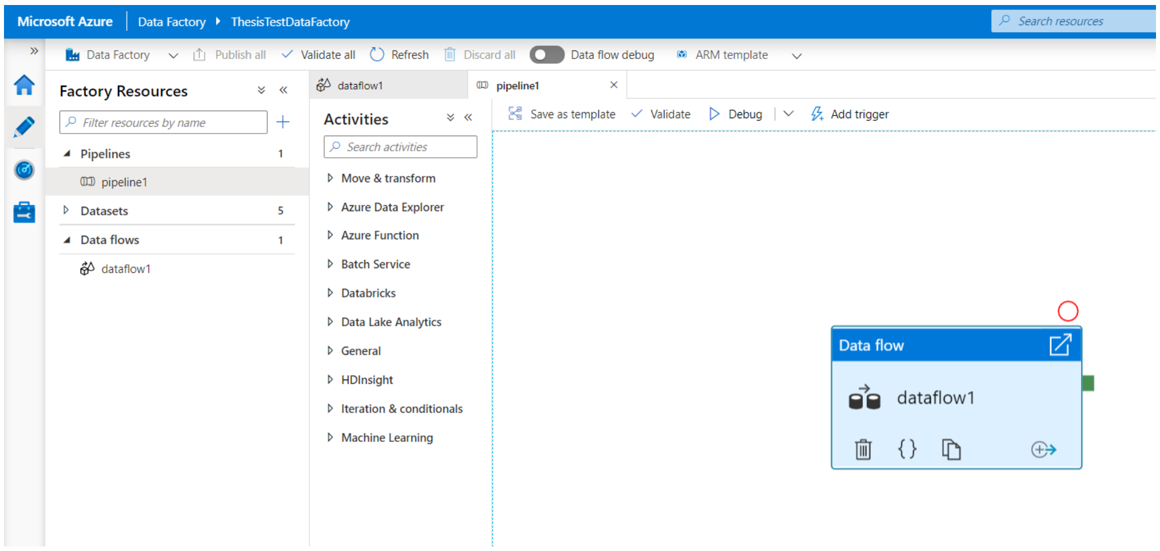
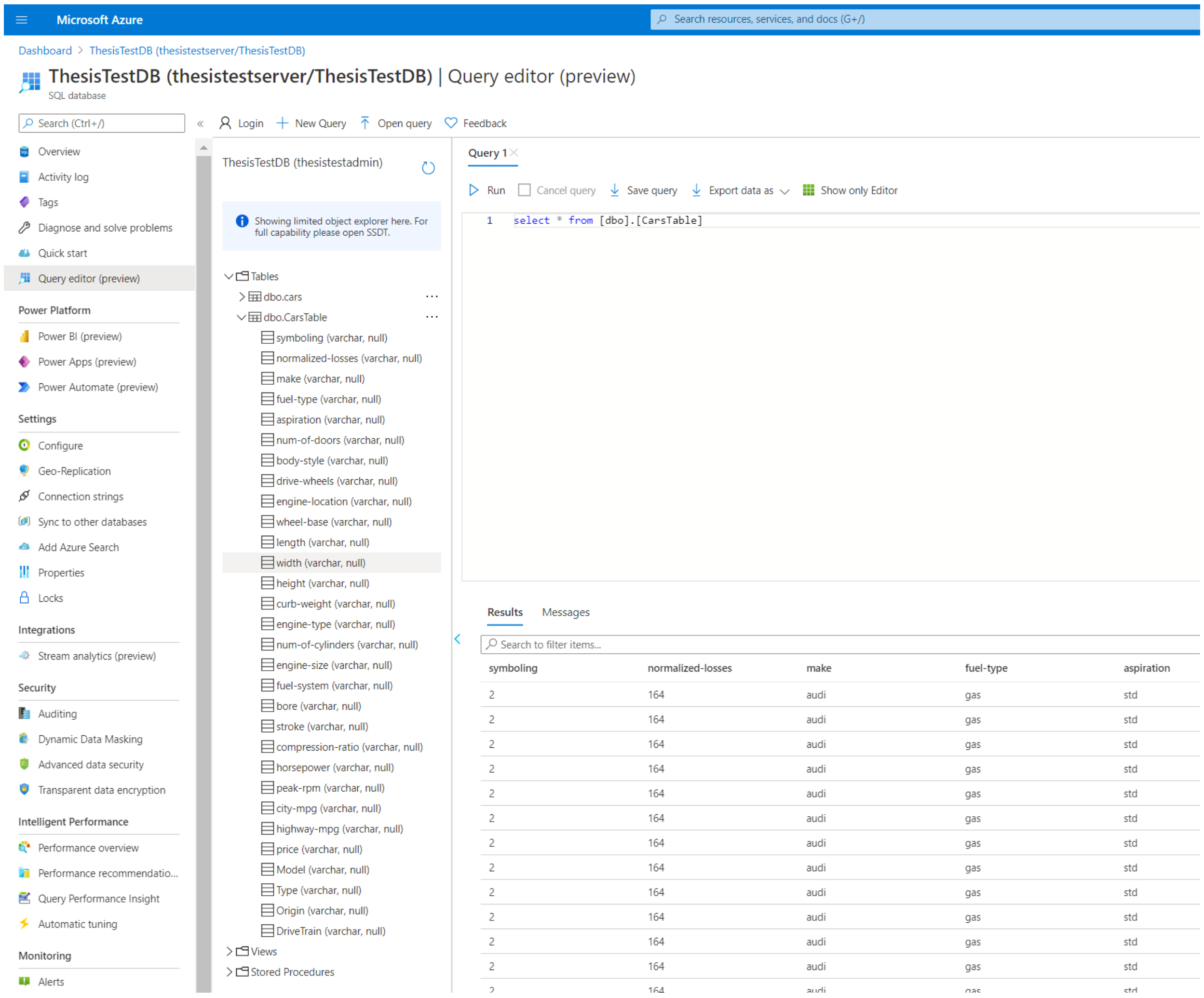


Figure 36: Pipeline in data factory

The picture below shows the resulted database after completion of the pipeline.



Microsoft Azure

Search resources, services, and docs (G+/)

Dashboard > ThesisTestDB (thesistestserver/ThesisTestDB)

ThesisTestDB (thesistestserver/ThesisTestDB) | Query editor (preview)

SQL database

Search (Ctrl+/) « Login + New Query Open query Feedback

- Overview
- Activity log
- Tags
- Diagnose and solve problems
- Quick start
- Query editor (preview)

Power Platform

- Power BI (preview)
- Power Apps (preview)
- Power Automate (preview)

Settings

- Configure
- Geo-Replication
- Connection strings
- Sync to other databases
- Add Azure Search
- Properties
- Locks

Integrations

- Stream analytics (preview)

Security

- Auditing
- Dynamic Data Masking
- Advanced data security
- Transparent data encryption

Intelligent Performance

- Performance overview
- Performance recommendatio...
- Query Performance Insight
- Automatic tuning

Monitoring

- Alerts

ThesisTestDB (thesistestadmin)

Showing limited object explorer here. For full capability please open SSDT.

- Tables
  - dbo.cars
  - dbo.CarsTable
    - symboling (varchar, null)
    - normalized-losses (varchar, null)
    - make (varchar, null)
    - fuel-type (varchar, null)
    - aspiration (varchar, null)
    - num-of-doors (varchar, null)
    - body-style (varchar, null)
    - drive-wheels (varchar, null)
    - engine-location (varchar, null)
    - wheel-base (varchar, null)
    - length (varchar, null)
    - width (varchar, null)
    - height (varchar, null)
    - curb-weight (varchar, null)
    - engine-type (varchar, null)
    - num-of-cylinders (varchar, null)
    - engine-size (varchar, null)
    - fuel-system (varchar, null)
    - bore (varchar, null)
    - stroke (varchar, null)
    - compression-ratio (varchar, null)
    - horsepower (varchar, null)
    - peak-rpm (varchar, null)
    - city-mpg (varchar, null)
    - highway-mpg (varchar, null)
    - price (varchar, null)
    - Model (varchar, null)
    - Type (varchar, null)
    - Origin (varchar, null)
    - DriveTrain (varchar, null)
- Views
- Stored Procedures

Query 1

Run Cancel query Save query Export data as Show only Editor

```
select * from [dbo].[CarsTable]
```

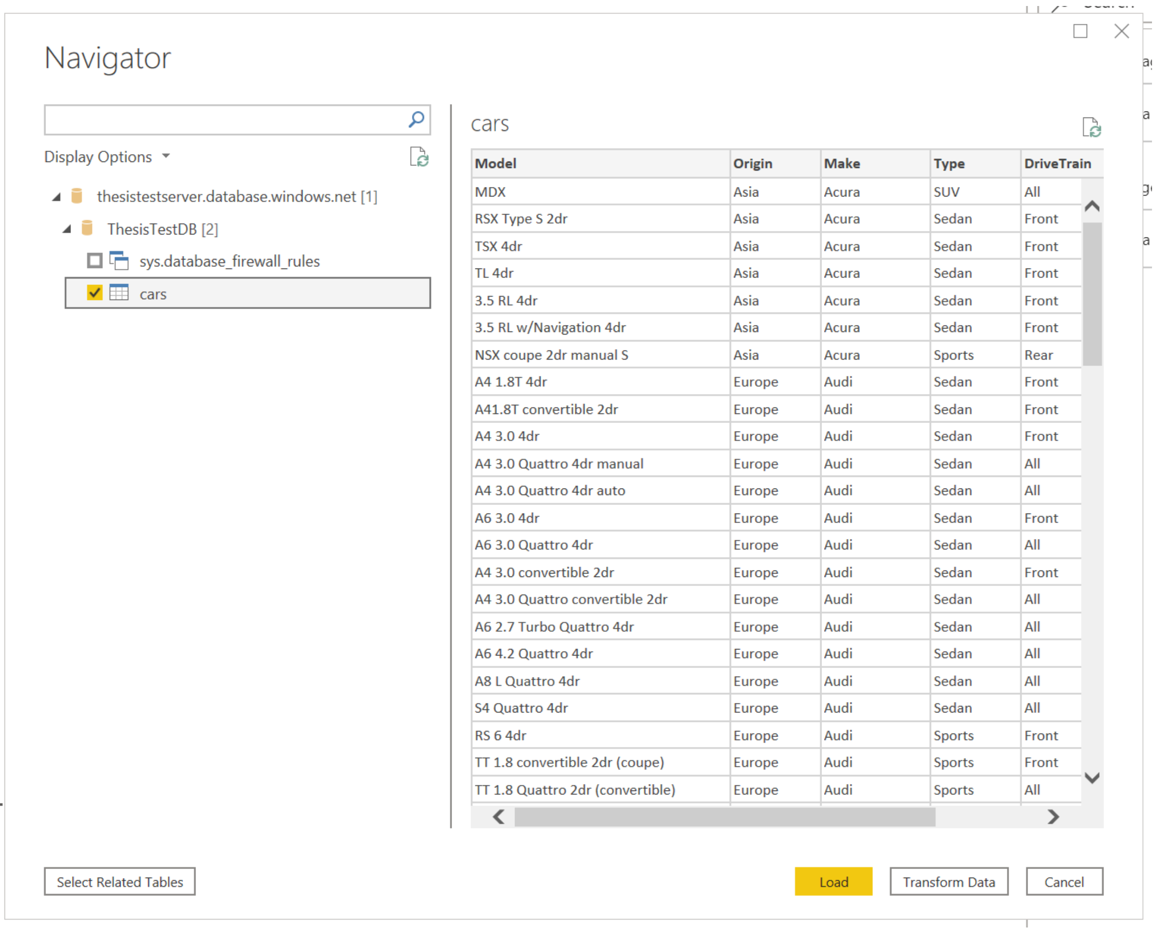
Results Messages

Search to filter items...

symboling	normalized-losses	make	fuel-type	aspiration
2	164	audi	gas	std
2	164	audi	gas	std
2	164	audi	gas	std
2	164	audi	gas	std
2	164	audi	gas	std
2	164	audi	gas	std
2	164	audi	gas	std
2	164	audi	gas	std
2	164	audi	gas	std
2	164	audi	gas	std
2	164	audi	gas	std
2	164	audi	gas	std
2	164	audi	gas	std

Figure 37: SQL server

After the data is stored in the SQL server, the next tool is the visualization tool. The picture below shows the connection made from Power BI to the SQL server.



cars

Model	Origin	Make	Type	DriveTrain
MDX	Asia	Acura	SUV	All
RSX Type S 2dr	Asia	Acura	Sedan	Front
TSX 4dr	Asia	Acura	Sedan	Front
TL 4dr	Asia	Acura	Sedan	Front
3.5 RL 4dr	Asia	Acura	Sedan	Front
3.5 RL w/Navigation 4dr	Asia	Acura	Sedan	Front
NSX coupe 2dr manual S	Asia	Acura	Sports	Rear
A4 1.8T 4dr	Europe	Audi	Sedan	Front
A41.8T convertible 2dr	Europe	Audi	Sedan	Front
A4 3.0 4dr	Europe	Audi	Sedan	Front
A4 3.0 Quattro 4dr manual	Europe	Audi	Sedan	All
A4 3.0 Quattro 4dr auto	Europe	Audi	Sedan	All
A6 3.0 4dr	Europe	Audi	Sedan	Front
A6 3.0 Quattro 4dr	Europe	Audi	Sedan	All
A4 3.0 convertible 2dr	Europe	Audi	Sedan	Front
A4 3.0 Quattro convertible 2dr	Europe	Audi	Sedan	All
A6 2.7 Turbo Quattro 4dr	Europe	Audi	Sedan	All
A6 4.2 Quattro 4dr	Europe	Audi	Sedan	All
A8 L Quattro 4dr	Europe	Audi	Sedan	All
S4 Quattro 4dr	Europe	Audi	Sedan	All
RS 6 4dr	Europe	Audi	Sports	Front
TT 1.8 convertible 2dr (coupe)	Europe	Audi	Sports	Front
TT 1.8 Quattro 2dr (convertible)	Europe	Audi	Sports	All

Figure 38: Power BI import wizard

The data can be visualized in the shape or form the user wants. Below, a count of car models by the origin and car type by make are created.

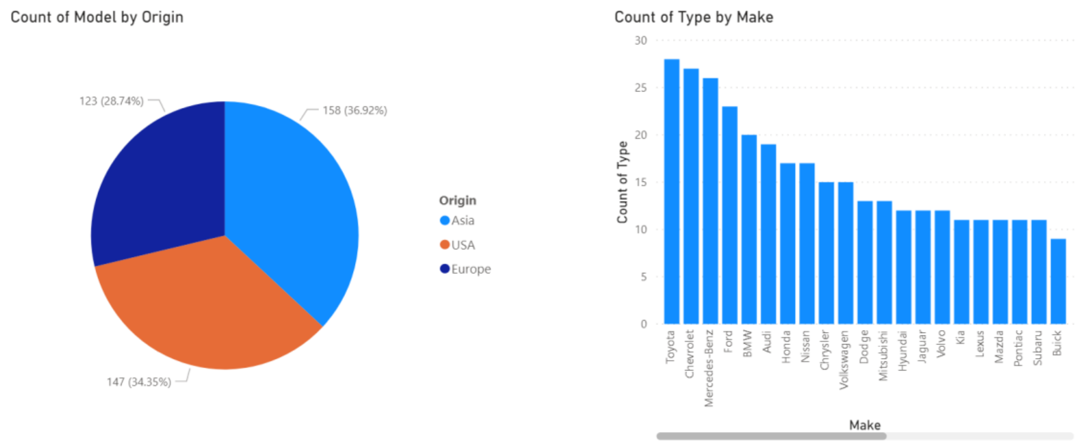
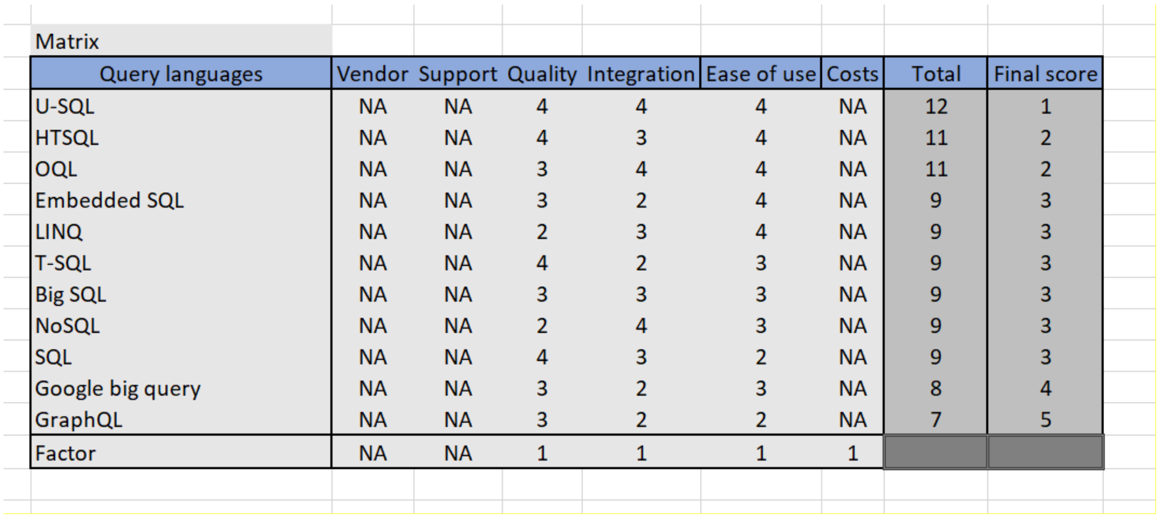


Figure 39: Power BI visual

3. RECOMMENDED SELECTION METHOD

3.1 Decision matrix analysis

A decision matrix analysis, also known as Multi-Attribute Utility Theory, is a useful technique to make decisions. It is powerful when several good alternatives are available and many different factors to consider. The matrix below is the first matrix made with the different query languages, criteria, and points. The criteria can be found in chapter 2.3.



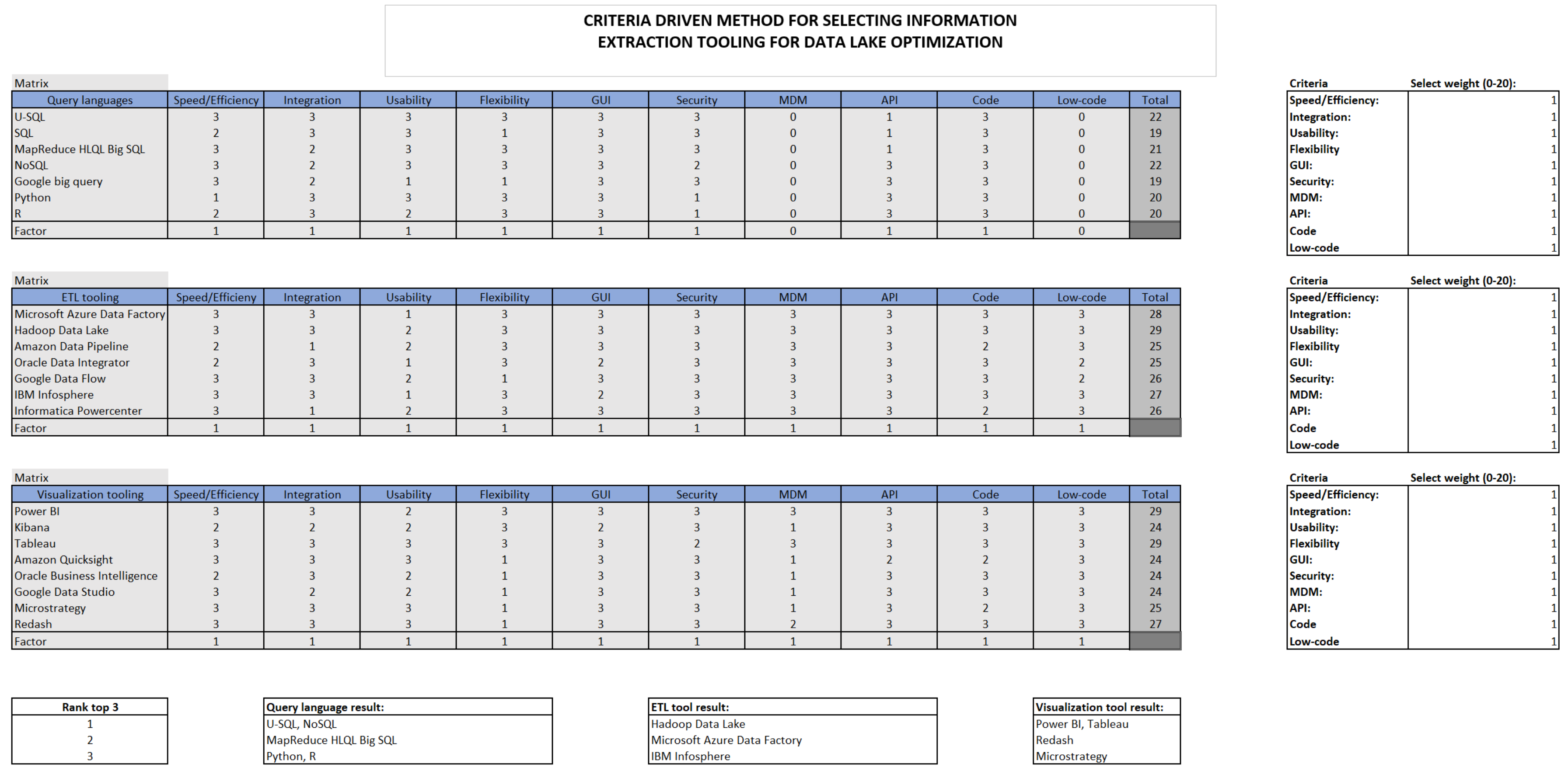
Matrix

Query languages	Vendor	Support	Quality	Integration	Ease of use	Costs	Total	Final score
U-SQL	NA	NA	4	4	4	NA	12	1
HTSQL	NA	NA	4	3	4	NA	11	2
OQL	NA	NA	3	4	4	NA	11	2
Embedded SQL	NA	NA	3	2	4	NA	9	3
LINQ	NA	NA	2	3	4	NA	9	3
T-SQL	NA	NA	4	2	3	NA	9	3
Big SQL	NA	NA	3	3	3	NA	9	3
NoSQL	NA	NA	2	4	3	NA	9	3
SQL	NA	NA	4	3	2	NA	9	3
Google big query	NA	NA	3	2	3	NA	8	4
GraphQL	NA	NA	3	2	2	NA	7	5
Factor	NA	NA	1	1	1	1		

Figure 40: Example of first decision matrix

The next step was to improve this matrix to a dynamic mechanism with reliable criteria, query language, ETL tooling, visualization tooling and advice. It will show how the different query languages, ETL tools, and visualization tools are scored based on the pros, cons, and criteria found through the research.

Then, through a user interaction pane, the user can spend a max of 20 points to distribute over the different criteria. Based on the distributed 20 points, advice is showed at the bottom of the method. The formula returns a top 3 of query languages, ETL tools, and visualization tools based on the points the user has spent. On the next page, a picture of the method is shown.



CRITERIA DRIVEN METHOD FOR SELECTING INFORMATION EXTRACTION TOOLING FOR DATA LAKE OPTIMIZATION

Matrix

Query languages	Speed/Efficiency	Integration	Usability	Flexibility	GUI	Security	MDM	API	Code	Low-code	Total
U-SQL	3	3	3	3	3	3	0	1	3	0	22
SQL	2	3	3	1	3	3	0	1	3	0	19
MapReduce HLQL Big SQL	3	2	3	3	3	3	0	1	3	0	21
NoSQL	3	2	3	3	3	2	0	3	3	0	22
Google big query	3	2	1	1	3	3	0	3	3	0	19
Python	1	3	3	3	3	1	0	3	3	0	20
R	2	3	2	3	3	1	0	3	3	0	20
Factor	1	1	1	1	1	1	0	1	1	0	

Criteria	Select weight (0-20):
Speed/Efficiency:	1
Integration:	1
Usability:	1
Flexibility	1
GUI:	1
Security:	1
MDM:	1
API:	1
Code	1
Low-code	1

Matrix

ETL tooling	Speed/Efficieny	Integration	Usability	Flexibility	GUI	Security	MDM	API	Code	Low-code	Total
Microsoft Azure Data Factory	3	3	1	3	3	3	3	3	3	3	28
Hadoop Data Lake	3	3	2	3	3	3	3	3	3	3	29
Amazon Data Pipeline	2	1	2	3	3	3	3	3	2	3	25
Oracle Data Integrator	2	3	1	3	2	3	3	3	3	2	25
Google Data Flow	3	3	2	1	3	3	3	3	3	2	26
IBM Infosphere	3	3	1	3	2	3	3	3	3	3	27
Informatica Powercenter	3	1	2	3	3	3	3	3	2	3	26
Factor	1	1	1	1	1	1	1	1	1	1	

Criteria	Select weight (0-20):
Speed/Efficiency:	1
Integration:	1
Usability:	1
Flexibility	1
GUI:	1
Security:	1
MDM:	1
API:	1
Code	1
Low-code	1

Matrix

Visualization tooling	Speed/Efficiency	Integration	Usability	Flexibility	GUI	Security	MDM	API	Code	Low-code	Total
Power BI	3	3	2	3	3	3	3	3	3	3	29
Kibana	2	2	2	3	2	3	1	3	3	3	24
Tableau	3	3	3	3	3	2	3	3	3	3	29
Amazon Quicksight	3	3	3	1	3	3	1	2	2	3	24
Oracle Business Intelligence	2	3	2	1	3	3	1	3	3	3	24
Google Data Studio	3	2	2	1	3	3	1	3	3	3	24
Microstrategy	3	3	3	1	3	3	1	3	2	3	25
Redash	3	3	3	1	3	3	2	3	3	3	27
Factor	1	1	1	1	1	1	1	1	1	1	

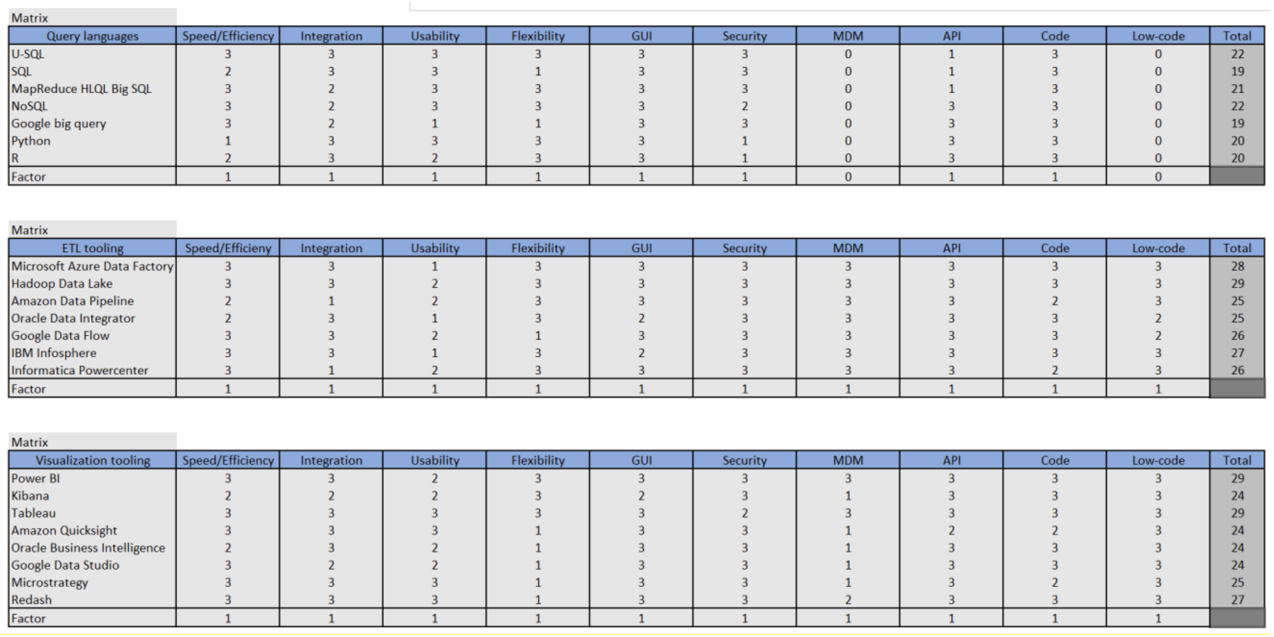
Criteria	Select weight (0-20):
Speed/Efficiency:	1
Integration:	1
Usability:	1
Flexibility	1
GUI:	1
Security:	1
MDM:	1
API:	1
Code	1
Low-code	1

Rank top 3	Query language result:	ETL tool result:	Visualization tool result:
1	U-SQL, NoSQL	Hadoop Data Lake	Power BI, Tableau
2	MapReduce HLQL Big SQL	Microsoft Azure Data Factory	Redash
3	Python, R	IBM Infosphere	Microstrategy

Figure 41: Example of method

The method is built out of several things. The criteria are shown on the horizontal axes; these criteria are explained in more detail in chapter 2.3. The different tools are explained in more detail in chapter 2.4, 2.5, and 2.6.

At the bottom of the matrix, a factor is added to identify the importance of the criteria. In this matrix, they are all set to 1, but the user can change the factor to his/her wishes as shown in chapter 4. Furthermore, a total is shown to visualize the number of points a tool has received.



Matrix

Query languages	Speed/Efficiency	Integration	Usability	Flexibility	GUI	Security	MDM	API	Code	Low-code	Total
U-SQL	3	3	3	3	3	3	0	1	3	0	22
SQL	2	3	3	1	3	3	0	1	3	0	19
MapReduce HLQL Big SQL	3	2	3	3	3	3	0	1	3	0	21
NoSQL	3	2	3	3	3	2	0	3	3	0	22
Google big query	3	2	1	1	3	3	0	3	3	0	19
Python	1	3	3	3	3	1	0	3	3	0	20
R	2	3	2	3	3	1	0	3	3	0	20
Factor	1	1	1	1	1	1	0	1	1	0	

Matrix

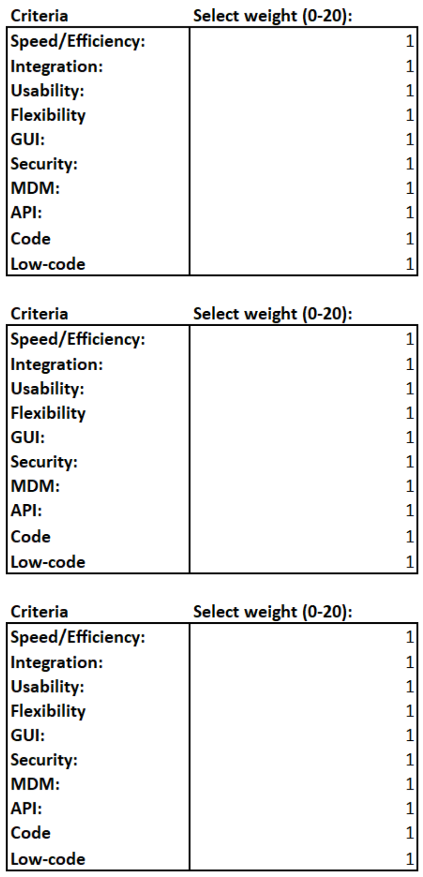
ETL tooling	Speed/Efficieny	Integration	Usability	Flexibility	GUI	Security	MDM	API	Code	Low-code	Total
Microsoft Azure Data Factory	3	3	1	3	3	3	3	3	3	3	28
Hadoop Data Lake	3	3	2	3	3	3	3	3	3	3	29
Amazon Data Pipeline	2	1	2	3	3	3	3	3	2	3	25
Oracle Data Integrator	2	3	1	3	2	3	3	3	3	2	25
Google Data Flow	3	3	2	1	3	3	3	3	3	2	26
IBM Infosphere	3	3	1	3	2	3	3	3	3	3	27
Informatica Powercenter	3	1	2	3	3	3	3	3	2	3	26
Factor	1	1	1	1	1	1	1	1	1	1	

Matrix

Visualization tooling	Speed/Efficiency	Integration	Usability	Flexibility	GUI	Security	MDM	API	Code	Low-code	Total
Power BI	3	3	2	3	3	3	3	3	3	3	29
Kibana	2	2	2	3	2	3	1	3	3	3	24
Tableau	3	3	3	3	3	2	3	3	3	3	29
Amazon Quicksight	3	3	3	1	3	3	1	2	2	3	24
Oracle Business Intelligence	2	3	2	1	3	3	1	3	3	3	24
Google Data Studio	3	2	2	1	3	3	1	3	3	3	24
Microstrategy	3	3	3	1	3	3	1	3	2	3	25
Redash	3	3	3	1	3	3	2	3	3	3	27
Factor	1	1	1	1	1	1	1	1	1	1	

Figure 42: Example of the first part of the matrix

This pane shows the different criteria available, and the user can use 20 points to distribute over the criteria, based on his/her situation and wishes.



Criteria	Select weight (0-20):
Speed/Efficiency:	1
Integration:	1
Usability:	1
Flexibility	1
GUI:	1
Security:	1
MDM:	1
API:	1
Code	1
Low-code	1

Criteria	Select weight (0-20):
Speed/Efficiency:	1
Integration:	1
Usability:	1
Flexibility	1
GUI:	1
Security:	1
MDM:	1
API:	1
Code	1
Low-code	1

Criteria	Select weight (0-20):
Speed/Efficiency:	1
Integration:	1
Usability:	1
Flexibility	1
GUI:	1
Security:	1
MDM:	1
API:	1
Code	1
Low-code	1

Figure 43: Example of the second part of the method

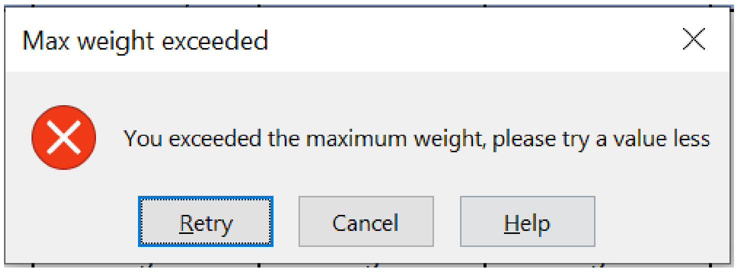
Different tooling is advised based on the weight given to a criterion by the user at the bottom of the matrix.

Rank top 3	Query language result:	ETL tool result:	Visualization tool result:
1	U-SQL, NoSQL	Hadoop Data Lake	Power BI, Tableau
2	MapReduce HLQL Big SQL	Microsoft Azure Data Factory	Redash
3	Python, R	IBM Infosphere	Microstrategy

Figure 44: Example of the third part of the method

These matrixes are a scored overview of the different tooling with a neutral factor 1. They are sorted from best scorer to worst scorer. When using this matrix for a specific situation, the factors can be changed to other values. This might result in a different outcome based on that specific situation. In chapter 4, the method is used for specific situations at different organizations. Here the outcomes might be different than used in the neutral matrixes.

Furthermore, if the user has spent more than the max number of points, an exception is thrown. This exception is shown below:


Max weight exceeded

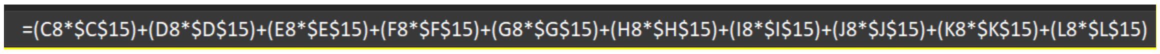
You exceeded the maximum weight, please try a value less

Figure 45: Example of the thrown exception

After this exception is thrown, the user can try again by clicking the “Retry” button.

3.2 Mechanism formula

The researcher developed a formula to create an automated and dynamic matrix that responds to the number of points a user has given to a criteria. This formula is shown in figure 46.

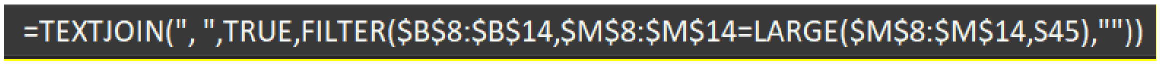


```
=(C8*$C$15)+(D8*$D$15)+(E8*$E$15)+(F8*$F$15)+(G8*$G$15)+(H8*$H$15)+(I8*$I$15)+(J8*$J$15)+(K8*$K$15)+(L8*$L$15)
```

Figure 46: Formula 1

This formula takes the amount of points one criteria has and multiplies it by the factor (weight) a criteria is given.

When a user has distributed the 20 points over all the criteria, some advice is given. The formula chosen is a dynamic formula that changes every time the user has changed the number of points.



```
=TEXTJOIN(", ",TRUE,FILTER($B$8:$B$14,$M$8:$M$14=LARGE($M$8:$M$14,S45),""))
```

Figure 47: Formula 2

Another formula is needed to make a top 3 of tooling. One of the problems faced is that tied values gave a spill error. This formula solved the problem of ranking the top 3 tools, including tied values.

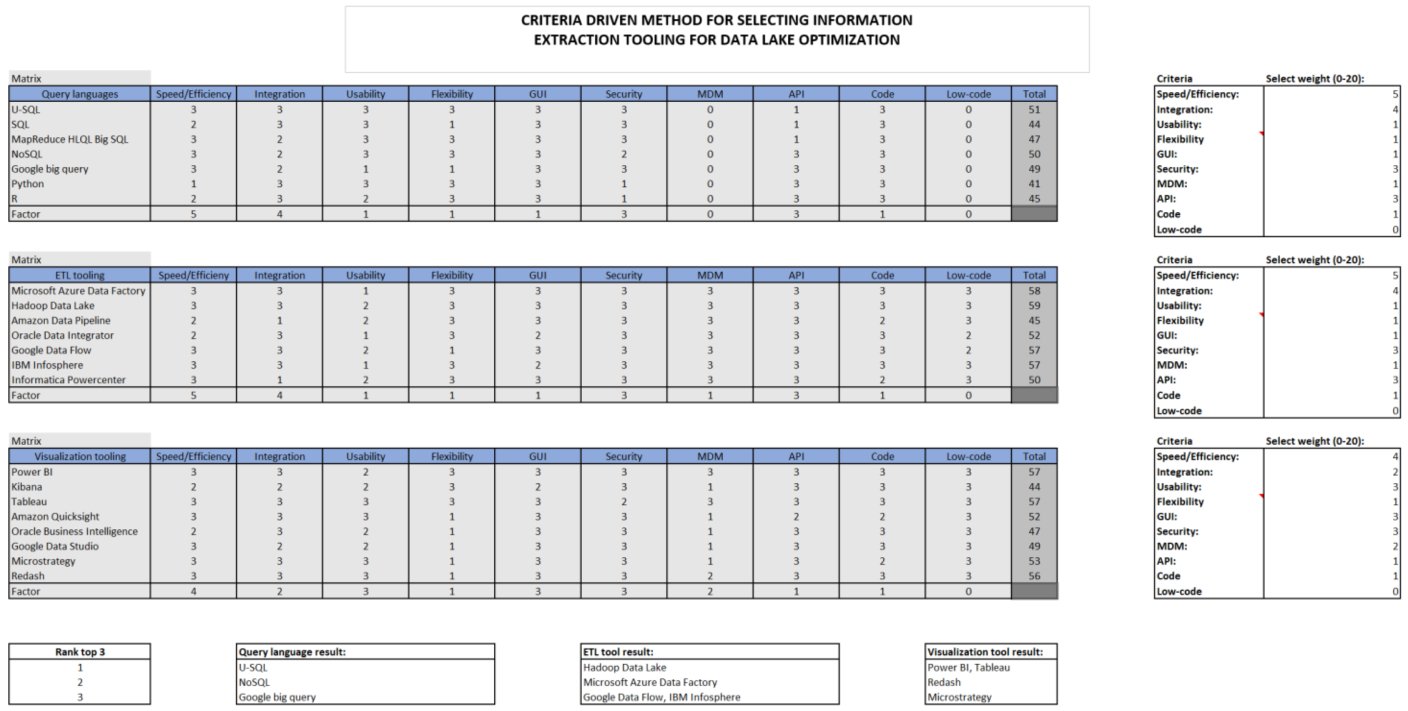
The formula connects the list of tools on the left in the matrix with the list of the total amount of points on the right in the matrix and takes a large amount of the list calculated. Next, the filter function is used to rank the top 3 of the resulted calculation.

If the formula is just left like this, Excel will return a “spill” error because this formula cannot handle tied values. This problem is solved by using the “textjoin” function.

The “textjoin” function handles the tied values by recognizing the delimiter, which is a comma in our case. The formula needs to ignore empty cells by setting the following formula to TRUE, and finally, the first formula with the filter function is combined in the “textjoin” function. This formula will show the top 3 values, and if there is one tied value, it will separate the tied values with a comma and show it in a shared place 1, 2, or 3.

4. VALIDATION RESULTS

4.1 Application of mechanism to KAD



CRITERIA DRIVEN METHOD FOR SELECTING INFORMATION EXTRACTION TOOLING FOR DATA LAKE OPTIMIZATION

Matrix

Query languages	Speed/Efficiency	Integration	Usability	Flexibility	GUI	Security	MDM	API	Code	Low-code	Total
U-SQL	3	3	3	3	3	3	0	1	3	0	51
SQL	2	3	3	1	3	3	0	1	3	0	44
MapReduce HLQL Big SQL	3	2	3	3	3	3	0	1	3	0	47
NoSQL	3	2	3	3	3	2	0	3	3	0	50
Google big query	3	2	1	1	3	3	0	3	3	0	49
Python	1	3	3	3	3	1	0	3	3	0	41
R	2	3	2	3	3	1	0	3	3	0	45
Factor	5	4	1	1	1	3	0	3	1	0	

Criteria	Select weight (0-20):
Speed/Efficiency:	5
Integration:	4
Usability:	1
Flexibility	1
GUI:	1
Security:	3
MDM:	1
API:	3
Code	1
Low-code	0

Matrix

ETL tooling	Speed/Efficieny	Integration	Usability	Flexibility	GUI	Security	MDM	API	Code	Low-code	Total
Microsoft Azure Data Factory	3	3	1	3	3	3	3	3	3	3	58
Hadoop Data Lake	3	3	2	3	3	3	3	3	3	3	59
Amazon Data Pipeline	2	1	2	3	3	3	3	3	2	3	45
Oracle Data Integrator	2	3	1	3	2	3	3	3	3	2	52
Google Data Flow	3	3	2	1	3	3	3	3	3	2	57
IBM Infosphere	3	3	1	3	2	3	3	3	3	3	57
Informatica Powercenter	3	1	2	3	3	3	3	3	2	3	50
Factor	5	4	1	1	1	3	1	3	1	0	

Criteria	Select weight (0-20):
Speed/Efficiency:	5
Integration:	4
Usability:	1
Flexibility	1
GUI:	1
Security:	3
MDM:	1
API:	3
Code	1
Low-code	0

Matrix

Visualization tooling	Speed/Efficiency	Integration	Usability	Flexibility	GUI	Security	MDM	API	Code	Low-code	Total
Power BI	3	3	2	3	3	3	3	3	3	3	57
Kibana	2	2	2	3	2	3	1	3	3	3	44
Tableau	3	3	3	3	3	2	3	3	3	3	57
Amazon Quicksight	3	3	3	1	3	3	1	2	2	3	52
Oracle Business Intelligence	2	3	2	1	3	3	1	3	3	3	47
Google Data Studio	3	2	2	1	3	3	1	3	3	3	49
Microstrategy	3	3	3	1	3	3	1	3	2	3	53
Redash	3	3	3	1	3	3	2	3	3	3	56
Factor	4	2	3	1	3	3	2	1	1	0	

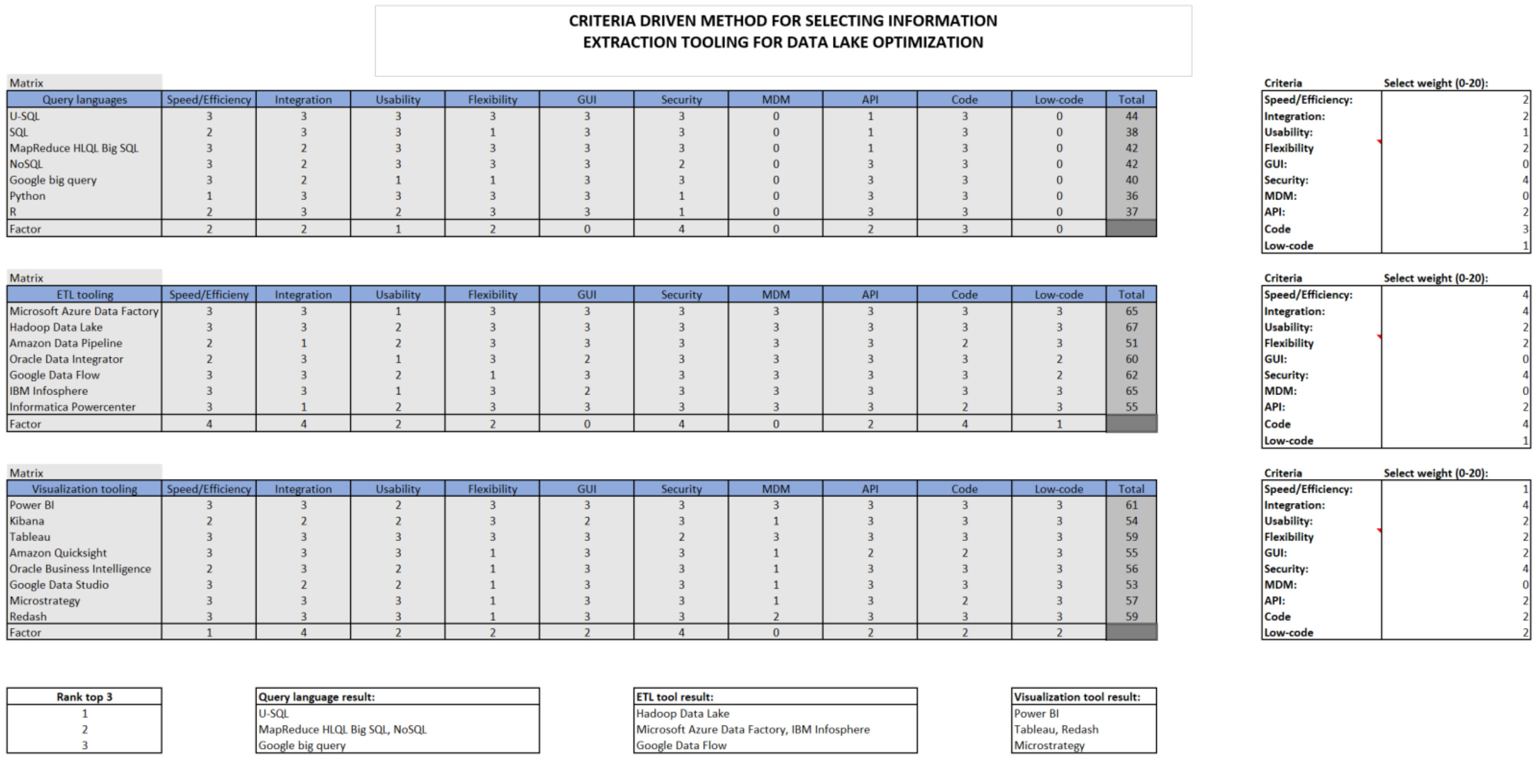
Criteria	Select weight (0-20):
Speed/Efficiency:	4
Integration:	2
Usability:	3
Flexibility	1
GUI:	3
Security:	3
MDM:	2
API:	1
Code	1
Low-code	0

Rank top 3	**Query language result:**	**ETL tool result:**	**Visualization tool result:**
1	U-SQL	Hadoop Data Lake	Power BI, Tableau
2	NoSQL	Microsoft Azure Data Factory	Redash
3	Google big query	Google Data Flow, IBM Infosphere	Microstrategy

Figure 48: Method for KAD the Netherlands

The KAD team gave some useful advice. Some criteria to add to the method are cloud provider specification, team skills, costs, and market penetration tools (how is a tool used in the market). Some tools are missing, for example, Spark, Airflow, Kafka, Apache Beam, and elastic search. The interesting thing is that according to these respondents, it is not essential to have MDM as criteria. It is more important to structure this information in a process instead of using MDM.

4.2 Application of mechanism to Jumbo



CRITERIA DRIVEN METHOD FOR SELECTING INFORMATION EXTRACTION TOOLING FOR DATA LAKE OPTIMIZATION

Matrix

Query languages	Speed/Efficiency	Integration	Usability	Flexibility	GUI	Security	MDM	API	Code	Low-code	Total
U-SQL	3	3	3	3	3	3	0	1	3	0	44
SQL	2	3	3	1	3	3	0	1	3	0	38
MapReduce HLQL Big SQL	3	2	3	3	3	3	0	1	3	0	42
NoSQL	3	2	3	3	3	2	0	3	3	0	42
Google big query	3	2	1	1	3	3	0	3	3	0	40
Python	1	3	3	3	3	1	0	3	3	0	36
R	2	3	2	3	3	1	0	3	3	0	37
Factor	2	2	1	2	0	4	0	2	3	0	

Criteria	Select weight (0-20):
Speed/Efficiency:	2
Integration:	2
Usability:	1
Flexibility	2
GUI:	0
Security:	4
MDM:	0
API:	2
Code	3
Low-code	1

Matrix

ETL tooling	Speed/Efficieny	Integration	Usability	Flexibility	GUI	Security	MDM	API	Code	Low-code	Total
Microsoft Azure Data Factory	3	3	1	3	3	3	3	3	3	3	65
Hadoop Data Lake	3	3	2	3	3	3	3	3	3	3	67
Amazon Data Pipeline	2	1	2	3	3	3	3	3	2	3	51
Oracle Data Integrator	2	3	1	3	2	3	3	3	3	2	60
Google Data Flow	3	3	2	1	3	3	3	3	3	2	62
IBM Infosphere	3	3	1	3	2	3	3	3	3	3	65
Informatica Powercenter	3	1	2	3	3	3	3	3	2	3	55
Factor	4	4	2	2	0	4	0	2	4	1	

Criteria	Select weight (0-20):
Speed/Efficiency:	4
Integration:	4
Usability:	2
Flexibility	2
GUI:	0
Security:	4
MDM:	0
API:	2
Code	4
Low-code	1

Matrix

Visualization tooling	Speed/Efficiency	Integration	Usability	Flexibility	GUI	Security	MDM	API	Code	Low-code	Total
Power BI	3	3	2	3	3	3	3	3	3	3	61
Kibana	2	2	2	3	2	3	1	3	3	3	54
Tableau	3	3	3	3	3	2	3	3	3	3	59
Amazon Quicksight	3	3	3	1	3	3	1	2	2	3	55
Oracle Business Intelligence	2	3	2	1	3	3	1	3	3	3	56
Google Data Studio	3	2	2	1	3	3	1	3	3	3	53
Microstrategy	3	3	3	1	3	3	1	3	2	3	57
Redash	3	3	3	1	3	3	2	3	3	3	59
Factor	1	4	2	2	2	4	0	2	2	2	

Criteria	Select weight (0-20):
Speed/Efficiency:	1
Integration:	4
Usability:	2
Flexibility	2
GUI:	2
Security:	4
MDM:	0
API:	2
Code	2
Low-code	2

Rank top 3	**Query language result:**	**ETL tool result:**	**Visualization tool result:**
1	U-SQL	Hadoop Data Lake	Power BI
2	MapReduce HLQL Big SQL, NoSQL	Microsoft Azure Data Factory, IBM Infosphere	Tableau, Redash
3	Google big query	Google Data Flow	Microstrategy

Figure 49: Method for Jumbo

This respondent reacted excitedly and said that this is a relevant, exciting method and can be used as an architectural picture. Some things to add are an extra layer in speed/efficiency, which counts how fast a cluster is started, Databrick, Azure Synapse, and the primary format in which the data is saved. Furthermore, would it help if there is some explanation of the criteria and the points per tool in a second tab because it is not useful to have the points in the center of the method; criteria and advice are essential. Finally, add additional criteria such as one cloud vendor vs multi-cloud vendor but criteria used now are sufficient.

4.3 Application of mechanism to Plus

There is no picture for this example because the person for this use case still did not send the filled in method after the interview. Nevertheless, during the interview, we talked about many aspects of this method.

In practice, there is a real need for a solution that can help better understand the available tools, and he found that this method is a useful tool that can help gain a better understanding of all those tools available. However, the question raised is if this method is sufficiently covering the tools available.

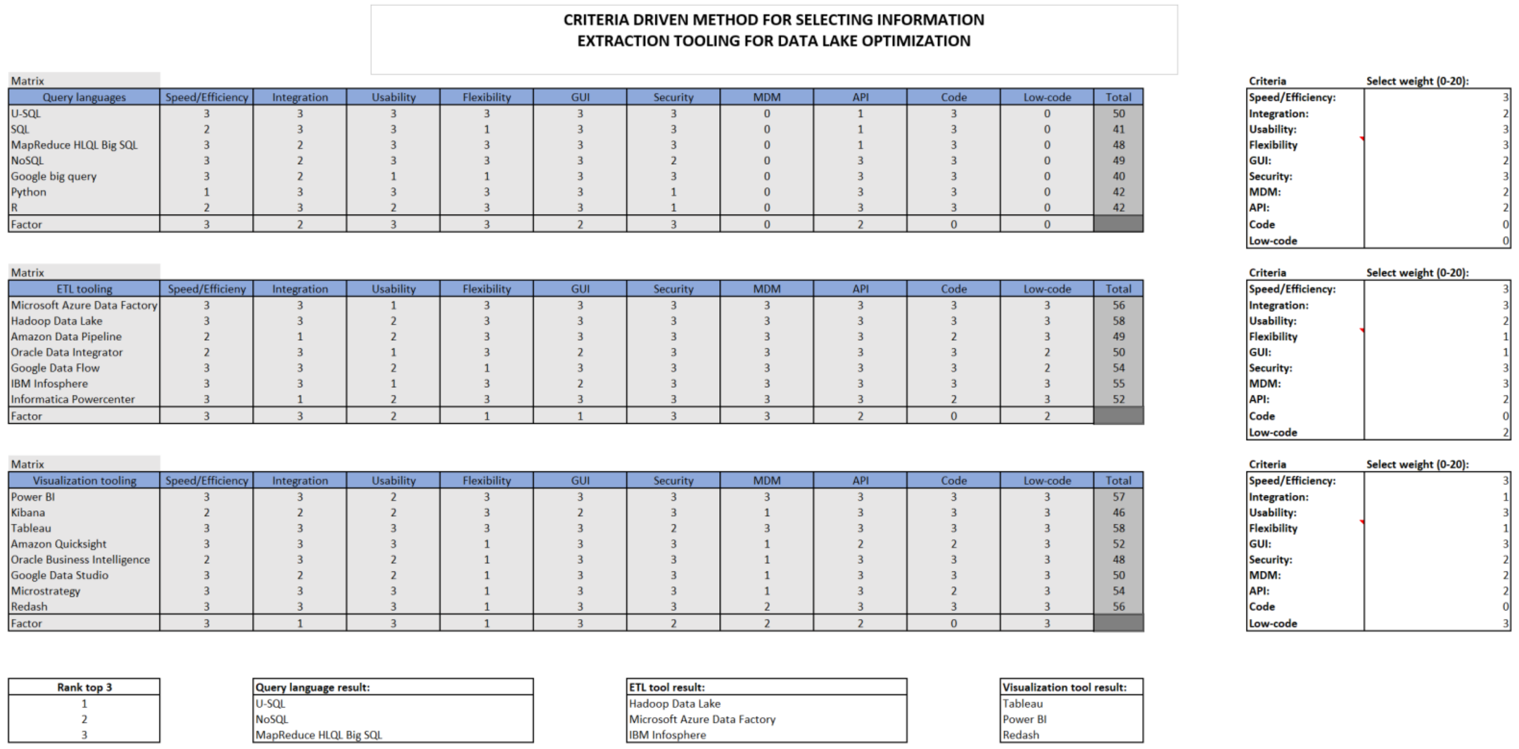
The criteria used in this method are sufficient, but there could be more criteria added such as costs, tool maturity, payment method (fixed vs pay as you go), industry, phase of data lake implementation (start, middle, finishing), and SMB vs Enterprise.

He said that a separate part could be added with more information about the criteria and the method's layout. The current method uses comments for some criteria to explain the meaning, but it could become chaotic when all the criteria have a red dot as a comment. By adding a separate part with extra information about the criteria, further interpretation of criteria from different people can be eliminated. It could help if the input parts of the points light up in a different color to highlight where the user is at that moment.

The covered tools in this method are not sufficient. For instance, PostgreSQL and Qlik are missing. Other tools need to be added to the method. A link of all the used tools to the magic quadrant could help as well. This way, the user can elaborate more on how the tool is performing in the market.

Bottom line will this method help with choosing a toolset, but it can be enhanced more.

4.4 Application of mechanism to Etos



CRITERIA DRIVEN METHOD FOR SELECTING INFORMATION EXTRACTION TOOLING FOR DATA LAKE OPTIMIZATION

Matrix

Query languages	Speed/Efficiency	Integration	Usability	Flexibility	GUI	Security	MDM	API	Code	Low-code	Total
U-SQL	3	3	3	3	3	3	0	1	3	0	50
SQL	2	3	3	1	3	3	0	1	3	0	41
MapReduce HLQL Big SQL	3	2	3	3	3	3	0	1	3	0	48
NoSQL	3	2	3	3	3	2	0	3	3	0	49
Google big query	3	2	1	1	3	3	0	3	3	0	40
Python	1	3	3	3	3	1	0	3	3	0	42
R	2	3	2	3	3	1	0	3	3	0	42
Factor	3	2	3	3	2	3	0	2	0	0	

Criteria	Select weight (0-20):
Speed/Efficiency:	3
Integration:	2
Usability:	3
Flexibility	3
GUI:	2
Security:	3
MDM:	2
API:	2
Code	0
Low-code	0

Matrix

ETL tooling	Speed/Efficieny	Integration	Usability	Flexibility	GUI	Security	MDM	API	Code	Low-code	Total
Microsoft Azure Data Factory	3	3	1	3	3	3	3	3	3	3	56
Hadoop Data Lake	3	3	2	3	3	3	3	3	3	3	58
Amazon Data Pipeline	2	1	2	3	3	3	3	3	2	3	49
Oracle Data Integrator	2	3	1	3	2	3	3	3	3	2	50
Google Data Flow	3	3	2	1	3	3	3	3	3	2	54
IBM Infosphere	3	3	1	3	2	3	3	3	3	3	55
Informatica Powercenter	3	1	2	3	3	3	3	3	2	3	52
Factor	3	3	2	1	1	3	3	2	0	2	

Criteria	Select weight (0-20):
Speed/Efficiency:	3
Integration:	3
Usability:	2
Flexibility	1
GUI:	1
Security:	3
MDM:	3
API:	2
Code	0
Low-code	2

Matrix

Visualization tooling	Speed/Efficiency	Integration	Usability	Flexibility	GUI	Security	MDM	API	Code	Low-code	Total
Power BI	3	3	2	3	3	3	3	3	3	3	57
Kibana	2	2	2	3	2	3	1	3	3	3	46
Tableau	3	3	3	3	3	2	3	3	3	3	58
Amazon Quicksight	3	3	3	1	3	3	1	2	2	3	52
Oracle Business Intelligence	2	3	2	1	3	3	1	3	3	3	48
Google Data Studio	3	2	2	1	3	3	1	3	3	3	50
Microstrategy	3	3	3	1	3	3	1	3	2	3	54
Redash	3	3	3	1	3	3	2	3	3	3	56
Factor	3	1	3	1	3	2	2	2	0	3	

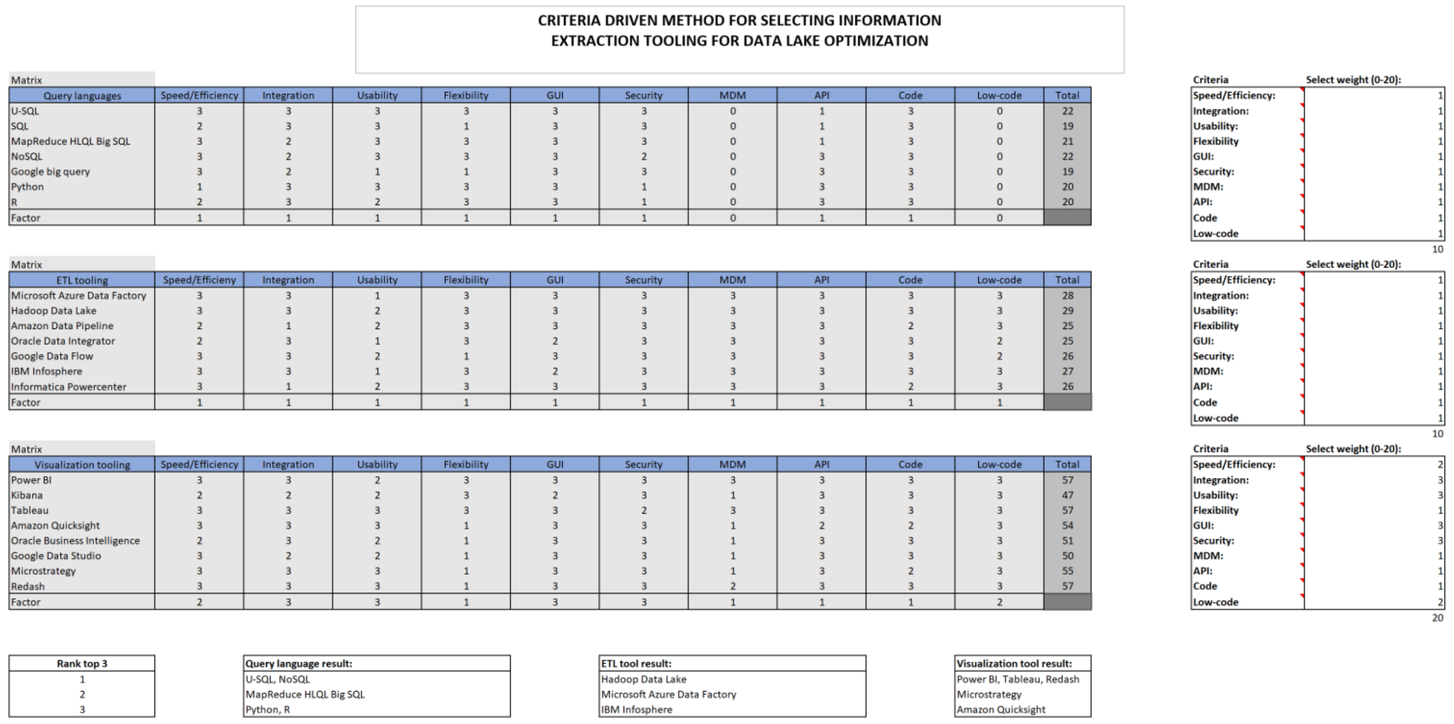
Criteria	Select weight (0-20):
Speed/Efficiency:	3
Integration:	1
Usability:	3
Flexibility	1
GUI:	3
Security:	2
MDM:	2
API:	2
Code	0
Low-code	3

Rank top 3	Query language result:	ETL tool result:	Visualization tool result:
1	U-SQL	Hadoop Data Lake	Tableau
2	NoSQL	Microsoft Azure Data Factory	Power BI
3	MapReduce HLQL Big SQL	IBM Infosphere	Redash

Figure 50: Method for Etos

This respondent said that there is a lot of tooling available, and it is difficult to understand and select these tooling. This method would help and is welcome and could be used as a referential architecture. The criteria used in this method are complete, but I could add the amount of knowledge a company has, which contracts a company already has, and the tooling costs. Furthermore, a comparison between two tooling helps better understand a specific number of tooling and a list with all the criteria meaning. Last, tools as Plotydash, Salesforce, MuleSoft and a comparison between master data management and metadata management are missing.

4.5 Application of mechanism to Airbus



CRITERIA DRIVEN METHOD FOR SELECTING INFORMATION EXTRACTION TOOLING FOR DATA LAKE OPTIMIZATION

Matrix

Query languages	Speed/Efficiency	Integration	Usability	Flexibility	GUI	Security	MDM	API	Code	Low-code	Total
U-SQL	3	3	3	3	3	3	0	1	3	0	22
SQL	2	3	3	1	3	3	0	1	3	0	19
MapReduce HLQL Big SQL	3	2	3	3	3	3	0	1	3	0	21
NoSQL	3	2	3	3	3	2	0	3	3	0	22
Google big query	3	2	1	1	3	3	0	3	3	0	19
Python	1	3	3	3	3	1	0	3	3	0	20
R	2	3	2	3	3	1	0	3	3	0	20
Factor	1	1	1	1	1	1	0	1	1	0	

Criteria	Select weight (0-20):
Speed/Efficiency:	1
Integration:	1
Usability:	1
Flexibility	1
GUI:	1
Security:	1
MDM:	1
API:	1
Code	1
Low-code	1
	10

Matrix

ETL tooling	Speed/Efficieny	Integration	Usability	Flexibility	GUI	Security	MDM	API	Code	Low-code	Total
Microsoft Azure Data Factory	3	3	1	3	3	3	3	3	3	3	28
Hadoop Data Lake	3	3	2	3	3	3	3	3	3	3	29
Amazon Data Pipeline	2	1	2	3	3	3	3	3	2	3	25
Oracle Data Integrator	2	3	1	3	2	3	3	3	3	2	25
Google Data Flow	3	3	2	1	3	3	3	3	3	2	26
IBM Infosphere	3	3	1	3	2	3	3	3	3	3	27
Informatica Powercenter	3	1	2	3	3	3	3	3	2	3	26
Factor	1	1	1	1	1	1	1	1	1	1	

Criteria	Select weight (0-20):
Speed/Efficiency:	1
Integration:	1
Usability:	1
Flexibility	1
GUI:	1
Security:	1
MDM:	1
API:	1
Code	1
Low-code	1
	10

Matrix

Visualization tooling	Speed/Efficiency	Integration	Usability	Flexibility	GUI	Security	MDM	API	Code	Low-code	Total
Power BI	3	3	2	3	3	3	3	3	3	3	57
Kibana	2	2	2	3	2	3	1	3	3	3	47
Tableau	3	3	3	3	3	2	3	3	3	3	57
Amazon Quicksight	3	3	3	1	3	3	1	2	2	3	54
Oracle Business Intelligence	2	3	2	1	3	3	1	3	3	3	51
Google Data Studio	3	2	2	1	3	3	1	3	3	3	50
Microstrategy	3	3	3	1	3	3	1	3	2	3	55
Redash	3	3	3	1	3	3	2	3	3	3	57
Factor	2	3	3	1	3	3	1	1	1	2	

Criteria	Select weight (0-20):
Speed/Efficiency:	2
Integration:	3
Usability:	3
Flexibility	1
GUI:	3
Security:	3
MDM:	1
API:	1
Code	1
Low-code	2
	20

Rank top 3	Query language result:	ETL tool result:	Visualization tool result:
1	U-SQL, NoSQL	Hadoop Data Lake	Power BI, Tableau, Redash
2	MapReduce HLQL Big SQL	Microsoft Azure Data Factory	Microstrategy
3	Python, R	IBM Infosphere	Amazon Quicksight

Figure 51: Method for Airbus

Finally, the Airbus respondents reacted enthusiastically about the method. In their own words, "this is awesome". It would help gain structural and basic knowledge about the Data lake ecosystem and help choose the tooling. However, some parts are missing with the current method, which will make the method more usable. First of all, a necessary explanation about how the method works and how to use it. Furthermore, should a list with an explanation of all the acronyms be added. Last are the criteria costs, existing vendor support contracts, the table's customizability, and cloud vs on-premise missing.

4.6 Summary of main survey findings

Survey results

First question:

"How effective is this method in helping you find alternative/replacement components for your existing toolset (if existing)?"

This question scored an average of 2.8.

Second question:

"Did the method give you a better overview of the tool components required for data lake information extraction?"

This question scored an average of 4.

Third question:

"Do you think the set of criteria identified cover the complete set of criteria applicable to data lake extraction tooling (if not, please suggest additional ones in the comment section below)?"

This question scored an average of 3.4.

Fourth question:

"How effective is this method in helping you selecting a data lake toolset for your organization?"

This question scored an average of 3.

Fifth question:

"Do you have any other comments or suggestions?"

This was an open question for comments. See chapter 5 and appendix 5 for more information.

Reflection

The respondents were excited and confirmed that it would help them gain extra knowledge of the tools available in a structured way. Furthermore, could and should the method be used as a reference architecture.

The first part on why the method scores the points above is that visualizing in Excel is right, but it could be better with a better layout. The most important thing to consider is enhancing the less technical criteria such as team knowledge; this part will be explained more in chapter 5. Furthermore, should the customizability of the method be considered. For instance, the way how to add new criteria as a column or a new tool as a row.

The data suggest that the method is not effective in finding an alternative or replacing the existing toolset. Although the method is not effective in finding an alternative in the existing toolset, it shows a better overview of the tool components needed for data lake information extraction. The method does not cover all the criteria and tools needed, but the current criteria and tools are not wrong. However, they should be enhanced more to reach a more accurate and useful method. The criteria and tools to add will be discussed in chapter 5. On average, does the method effectively help to select the data lake toolset for organizations.

Finally, does the data suggest that more research is needed on other parts of this method as well. The method's effectiveness and overview can increase by adding the missing criteria and tooling, as discussed in chapter 5.

Lessons learned

As the interviews and surveys are an essential part of the study, one should start planning the interviews on time because it takes much time to respond and plan a date. If someone does not answer the first mail, do not wait too long to send the reminder because that will take a while. For this study, the researcher admits that he waited too long before sending a reminder in some cases.

Use a tool that will help create a stable and transparent form for the survey, such as Microsoft Forms or Google Forms. Due to such a tool's abilities, the researcher can create a form and send it to anyone. This research made use of Microsoft Forms, and that worked out well.

Another lesson is that for this study, most retail organizations were interviewed and filled in the survey. For other industries, only Airbus was interviewed, but this number of respondents in the Airbus industry is too small to generalize.

5. DISCUSSION

Research-problem

At the beginning of this research, the research problem was a gap in the literature that no information is available about the combination of tooling in one place to use for Data Lake purposes. Furthermore, no information is available about commonly used architectures or frameworks and criteria associated with it. So, while there has been much research on the different aspects individually, no researchers have considered the combination. The research questions will be discussed with the help of the interview input and the literature review.

The main research question is:
"What method is best suitable for selecting the appropriate tools to rapidly and efficiently extract data from the data lake?"

The sub-questions are:

Which criteria are essential for rapid and efficient data extraction?

Which query language will enable rapid and efficient data extraction?

Which ETL tool will enable rapid and efficient data extraction?

Which visualization tool will enable rapid and efficient data extraction?

What is the process/mechanism for selecting the appropriate tooling?

The main findings

In line with the literature review, we found that in practice, the data lakes are hot. All the respondents have data lakes running or are in the data lake transition and know precisely what a data lake is. The big v's found in the literature review are used in practice, and companies consider the possibilities of different data structures. Apache Hadoop was used a lot during the literature review because research executed by Khine & Wang(2018, p. 3025) described that many implementations of Data Lake are based initially on Apache Hadoop. A variety of data from heterogeneous data stores will be extracted to be stored in the Hadoop Cluster. The results show that the claims of Khine & Wang (2018, p. 3025) are valid but are decreasing. Many implantations of data lakes are based on new technologies such as those of Microsoft and AWS.

Research by Madera & Laurent (2016, pp. 1–3) concludes that the volume, the variety, and the velocity of data is another essential thing. A Data lake is a low-cost storage physical environment based on Hadoop technology, populated by all data sources available in the enterprise. When the data is processed and used by users or data scientists, the data warehouse will save the results. However, the analysis does not support this outcome. Data lakes can be based on different technology and not only on Hadoop technology.
Furthermore, does the processed data not only be saved to data warehouses, but it could also be saved back to the data lake. It depends on how scientists or analysts will use the data. The data warehouses are logic storage if the data needs fast pre-processing or aggregated data or reporting purposes.

Furthermore, do the results indicate that the criteria used in this study are sufficient. During the interview and the survey, the respondents found that the number of criteria used is enough. However, the respondents found that some missing criteria need to be added as well. Some proposed criteria to add are cloud provider specification, team skills, costs and market penetration tools, extra layer in speed/efficiency which counts how fast a cluster is started, amount of knowledge a company has, which contracts a company already has, tool maturity, fixed price model vs pay as you go, customizability of the method and the costs of the tooling.

The different tools compared are not sufficient; tools such as Spark, Airflow, Kafka, Apache beam, DataBricks, Azure Synapse, PostgresSQL, Qlik, Plotydash, MuleSoft, and elastic search should be added to the method to create a better view of the current landscape. The tools mentioned came from the interviews, but future research could take more tools into account.

Furthermore, is there no clear answer for the best query language, ETL tooling, and visualization tooling. It depends on the user and the organization. However, the most common query language during the interviews was U-SQL, the most common ETL tool was Hadoop data lake, and the most common visualization tool was Power BI and Tableau.

This method will work like a reference architecture, but the method's tools are not sufficient. Some essential tools are missing, which needs to be added to represent a better picture. Finally, should the method explain the different criteria in a separate tab, and should the unique selling points be added to the method.

What-do-these-results-mean?

In line with the expectations, we found that there is a need for such a method. People are looking for a reference architectural design which can help them with decision making.

However, the method can be enhanced more. The survey patterns expose that the method helps identify the right tooling for the data lake ecosystem. However, there is still some work needed to enhance the number of criteria. Furthermore, should the center part of the method with all the points given by the researcher be separated with more points or better elaboration on why a point is given. The change of criteria points by the user gives no real difference when making choices. This gives the feeling that it is a subjective selection.

Selecting a tool does not only depend on the technical fit within the subject. Existing contracts, team knowledge, and company resources play a significant role in choosing such a toolset. This, again, is linked to the criteria which need to be enhanced more.

The current method does not take care of a specific combination of tooling. Does a combination of some specific tooling add extra benefit for choosing linked tools? This means that further research should take into consideration to check this as well.

Overall, does the results meet the expectations. Beforehand we have not thought that the criteria are complete or the method would help that good. Nevertheless, the interviews and the surveys show that such a method is needed in the market. Furthermore, the hypothesis at the beginning of this research is in line with the research results.

Finally, the respondents did not choose MDM as an essential part of the method. We found that MDM is a critical subject in choosing and architecting the data lake during the literature review. However, the respondents did not focus on that at all. It was more about the knowledge which a company has and how much money a tool cost. It is interesting to discover MDM's ability in more detail, but for this study, the survey concludes that MDM is not essential, countering the literature review findings. Another interesting part of MDM is the abilities of Meta Data Management vs the abilities of Master Data Management and what they can add to the method.

The results do agree and support previous research and do add extra information to it. The research provides new insights into the relationship between using a method and the practical decision-making process. It seems that companies cannot see all the tools anymore because they come with so much. Although all information needed to get to a decision is available, it is difficult to get there in a structured way while still keeping the business criteria in mind. This research added a significant amount of knowledge to the current information, mainly the created method and its findings.

6. CONCLUSION

The objective of building a data lake is to derive value from it. If done correctly, having all data stored in a single repository and quickly analyzing the raw data will provide organizations with significant new insights. Mixing different data sources and analyzing them opens new possibilities, and with a data lake, that process suddenly becomes a lot easier. Therefore, organizations that have implemented a data lake will reap the benefits from it in the future if done correctly.

The study shows that large companies often lack a standard when choosing data lake software. This research aimed to identify the use of a method for enterprise organizations using or planning to use a data lake. Based on an extensive literature review and fife case studies, it can be concluded that a method is an essential factor to consider when designing data lake architectures. The results indicate that organizations are confused with the availability of tools and need help choosing from everything available.

Referring to the main research question, it can be concluded that the created method is suitable for selecting tooling to extract data from a data lake. This approach provides new insight into such a method's usage and urgency. This research clearly illustrates that the created method can help organizations choose the right tooling, but it also raises the question of using such a method to help with everything in one method.

Bottom line, even though most of the respondents were optimistic about the research, it should not be forgotten that choosing tooling is not only about the technical aspects of tooling; instead, it is also important to consider non-technical aspects.

Based on these conclusions, practitioners should consider the ability of a method to answer all the questions.

Contributions

This research contributes in several ways; first, returning to the problem statement, the research helps solve and show the importance of a standard for choosing data lake tooling. Current approaches are lacking to focus on concrete requirements and practices. The research has focused on the applicability of a method in general. The findings here confirm and validate previous research; and, they combine the information in one place. This study extends to show how a method can be built and what factors to consider. This study assessed the formula to calculate the advice, the criteria, and the used tools. Referring to the literature review, the study addressed the gap of using a combination of tooling with the right criteria in the form of a method. Although in-depth research is needed to elaborate more on some criteria. As discussed in the first chapter, data lakes have common failures, such as data lakes become data swamps, and therefore the data will never be put into production and failing to gain added value. The connection to that is that a data lake will lack business impact, lack data governance, and lack data quality. An important note here is that a standard or method, plays a significant role in mitigating these problems. An overall, structured method of building the data lake ecosystem is key to reaping this technology's benefits.

Limitations

The context where this study was executed was within one industry and one country. Four of the five participants are from one industry, which means they are familiar with the same environments and cultures. It does not give a definite viewpoint on how the method will work in the context of complex IT environments in other industries and countries. The method has only been validated by a group of people interviewed and filled in the survey, which means that it possibly will not serve a larger group of people. A group of people gave their feedback, but there is still a small chance that the feedback is biased. The study used a group of active companies in enterprise environments while there is still room for optimizing the method for smaller companies.

Recommendations for future research

This study provides some directions for future research. One of the things future research could cover is the development of a more detailed method. Concrete, it means that more research is needed on extra tooling, additional criteria such as costs and cloud on-premise, and a better way to present all this information in one page, such as another platform and availability to add additional criteria. All the feedback and a starting points can be found in chapter 5. Furthermore, further research is required to establish whether the method is a factor in smaller companies and other industries.

To better understand the implications of these results, future studies could also address several things. First, further research on the difference between metadata management and master data management and what their connection could be to a method. Second, research on business criteria to add to the already known technical criteria. Third, research on how MDM (master data management or metadata management) can help eliminate the data swamp risk of data lakes or eliminate this risk through another approach. Furthermore, research on the relationship between tooling. Is a specific set of tooling better to use in combination rather than separate?

7. APPENDICES

Appendix 1: Query language table

	Pros	Cons
Embedded SQL: is a method of combining the computing power of a programming language and the database manipulation capabilities of SQL.	- Easy to understand	- Not performing well on large data sets
HTSQL: is a schema-driven URI-to-SQL query language that takes a request over HTTP, converts it to a SQL query, executes the query against a database, and returns the results in a format best suited for the user agent	- Rapid web development - Fast on transactional data - Web-friendly syntax - Authentication, data caching, and encryption over HTTP	- Considered mainly as a web query language
OQL: is a query language standard for object-oriented databases	- Uses entity and association names instead of database table names - Can use predefined relations - Many SQL keywords also work in OQL - Deals with complex objects - Integration - API compatible	- OQL queries do not take security into account out-of-the-box
LINQ: is a Microsoft .NET Framework component that adds native data querying capabilities to .NET languages	- Supports safety - Easy to deploy - Easy to learn, - Compatible with .NET - Supports multiple databases, - Write queries uniformly in your programming language itself - Taking full advantage of strong	- It needs to process the entire query, which might have a negative performance impact

	typing and tool support - In-memory objects - SQL data and XML documents.	
GraphQL: is an open-source data query and manipulation language for APIs	- Self-documenting, - Precisely define the data you want, - Replacing multiple REST requests with a single call - Adopted by an increasing number of users - efficient, - integration with other API and languages	- Does not support all entities and relationships with API's - Does not follow the HTTP spec for caching and instead uses a single endpoint, - Fundamental understanding of the properties of the language is missing
T-SQL is Microsoft's and Sybase's proprietary extension to the SQL (Structured Query Language) used to interact with relational databases	- Error checking - Globally accepted	- Works best with Microsoft SQL server - Not able to use tsql into SQL environments
Big SQL: is MapReduce-based designed for providing native SQL for querying data managed by Hadoop	- Parallel processing SQL, - Low-latency parallel execution processing, - Run on the top of Hadoop and to translate all queries to native MR jobs, - Supports queries expressed in native SQL declarative language, - JDBC/ODBC driver access from Linux and Windows platforms - Uses HCatalog (metastore) of Hbase for data access and the Hive storage engines to	- It does not benefit from adding nodes. As a result, the running time performance is decreased by 43% from one to ten nodes.

	read/write data	
U-SQL: is a language that combines declarative SQL with imperative C# to let you process data at any scale	- The syntax is based on T-SQL while it uses C# types as default. - Process any type of data, - Integrates custom code seamlessly, - Efficiently scale to any size of data, - For massive data processing, - Dump whatever you want in the data lake and run USQL on top of it, - Flexibility and extensibility, and ease of development - A handful of state-of-the-art tools for XML processing - any type of data, - integration, - MDM	- Not to substitute the existing and emerging service description protocols in the various service areas (e.g., WSDL, WSDL-S, OWL-S)
No-SQL: is referred to as “nonrelational” or “non-SQL” to highlight the fact that they can handle vast volumes of rapidly changing, unstructured data in different ways than a relational (SQL) database with rows and tables.	- More freedom, - Speed - Flexibility - Provide compelling operational advantages and savings with the ability to scale "out" horizontally—or add less expensive servers without having to upgrade, - Large volumes of rapidly changing structured, semi-structured, and unstructured data, - Geographically distributed scale-out	- Not mature - Less support - The system can have only two out of three of the following properties: consistency, availability, and partition-tolerance. The NoSQL systems generally give up consistency - less secure

	architecture instead of expensive, - Monolithic architecture, - Integration with c# and .NET, - The ability to horizontally scale "simple operation" throughput over many servers, - Efficient use of distributed indexes and RAM for data storage.	
Google Big Query: a fully managed cloud service that enables storage and fast querying of large and multi-faceted datasets.	- High scalable - Cost-effective - No technical overhead costs for maintaining infrastructure, - Scalability of processing research data products across a growing number of courses and users - Fast - secure	- Only compatible with specific extensions such as JSON, CSV, or Avro - File size limits
SQL: Structured query language or SQL is used as a medium of communication with the relational database management systems.	- Fast, - Data integration standards, - Mature - secure (RBAC)	- Difficult interface, - Expensive
PL SQL	- It is highly structured - Readable - Accessible - Is right integrated with Oracle databases - Has high performance - Has API capabilities.	- It is not integrated with other databases - Not secure due to its vulnerabilities, such as SQL injection.

Spark SQL	- Can call complex analytics libraries in Spark - Is API friendly - High performance - Has a variety of data sources available - Can process large amounts of data - API capabilities - Has integrations with Java, Scala, Python, and R	- But it has problems with small files and latency.
Scala	- Is easy to get into - Has java integration and is mainly for building large systems. - It is scalable - Secure - API capabilities	- Immature - Risk of abandonment
Python	- Easy to work with - Runs on every platform - Integration with other tools and languages - Stable - API capabilities - Flexible - GUI	- Not fast - Run-time errors - Lack of multi-processor support - Database access layer problems - Not secure
R	- Open-source - A lot of packages available - Can visualize data - Highly compatible with other languages - API capabilities - Is compatible with many sources - Flexible - GUI	- Not secure - Not fast - Not efficient on large datasets - Hard to learn

Appendix 2: ETL tooling table

	Pros	Cons
SAP BW ETL: provides a collection of objects and tools that allow users to import, export, and transform heterogeneous data	- Runs on a variety of third-party RDBMSs - Provides open Business Application Programming Interfaces for data loading - Provides a pre-configured metadata repository consisting of InfoCube catalog - Report catalog - Information source catalog - Shipped with many pre-defined InfoCubes for typical business applications - Profitability analyses - Stock inventory analyses - Corporate indicator systems - Graphical user interface - Sharing functions and Microsoft Office compatibility - Data visualization and analytics applications - API compatible	- Expensive licensing - Not able to perform on high loads - No MDM integration
Talend: is used for integrating operational systems as well as an ETL tool for Data Warehousing, Business Intelligence, and data migration	- Easy to use - Helps business users to design their business processes graphically	- To use more inside tools like machine learning add-ons, you need to buy licenses

	- High volume integration - Parallelization feature - Data source integration - Free open source ETL tool - Secure - API compatible	- It is developed as a product for individual use only, and so it is not possible to have more than one user - The free version does not support automation of tasks like scheduling, routing data. - Lack of any commercial support. - Need to add on Talend MDM platform for MDM
Microsoft Azure Data Factory: is a data integration service that is explicitly designed to collaborate with existing services for the movement, transformation, and processing of raw data	- Visual drag-and-drop UI - SSIS migration to the cloud - Comprehensive orchestration Multiple language support - Hybrid data movement and transformation - Support Data source integration. - Secure - API compatible	- Configuration from scratch could be a bit hard - Need understanding of the tool - Need to add on data factory MDM platform for MDM
IBM – Infosphere Information Server: Is a product family that provides a unified data integration	- It can be integrated with Oracle, IBM DB2, and Hadoop System - It supports SAP via various plug-ins - ETL without coding - Data source integration - Integrated with BI - Master data management - Infrastructure rationalization and risk compliance	- Lack of a robust web development environment - Metadata propagation in Jobs is somewhat complex - Slow configuration - Expensive - Need to add on infosphere information server MDM platform for MDM

	- Secure (R and C) - API compatible	
Oracle Data Integrator: Is a graphical environment to build and manage data integration	- Mature - High performance - Data source integration - Can perform complex transformations - It automatically identifies faulty data and recycles it before moving into the target application - Supports databases like IBM DB2, Teradata, Sybase, Netezza, Exadata. - Able to handle complex transformations - ODI security - API compatible	- Difficult user interface - Complex build tool - Expensive - Need deep technical knowledge - Need to add on to data integrator MDM platform for MDM
Google Dataflow: Google Dataflow is a unified stream and batch data processing service	- Reduced infrastructure administration - Automatic scaling - SDK with native support for both batch and streaming modes - API compatible	- Needs a new way of the development approach - Need to add on to dataflow data catalog for MDM
Microsoft – SQL Server Integrated Services (SSIS): is one of the Business Intelligence tools (BI) to ease and automate the ETL process	- Automate the SQL Server Maintenance Plan - Very low in cost - Easier to maintain and package configuration - Better for complex transformations, multi-step operations, aggregating data	- To see the package execution report, need Management Studio rather than being published to reporting services or another way - The lack of ability to support non-windows operating systems

	from different data sources or types, and structured exception handling - Data can be loaded in parallel to many varied destinations - Build ETL solutions with very minimum background knowledge - Very easy to install and configure - Offers broad documentation and support - Provides best practices	- Is more suitable for enterprise and may not be cost-efficient for small businesses - Need third party tools for API integration - No MDM
Hadoop data lake: The data is physically moved into one physical place. Based on Hadoop technology, no change is made around the origin's format at the captured moment. The Data Lake is more batch processing oriented as it is based on MapReduce usage	- Fast - Less expensive - Integration - API compatible	- Less mature - Not realtime - No advanced analytics - Data not secured by default - Need add to Hadoop, Cloudera navigator for MDM
AWS Data Pipeline: This is an ETL platform in a web service with a control panel.	- Simple to use - Secure - Fault-tolerant - API compatible	- Lacks third-party integration - Difficult to use with on-premises data sources - Need the third party for MDM
MongoDB: MongoDB is an open-source NoSQL database developed in C++	- Efficient - Fast - Durability - Flexible - API compatible	- uses internal memory - lacks the support of join queries - Need the third party for MDM
Cloudera Apache Hive	- Able to process petabytes of data - Integration - Scalable - Fast - MDM	- The configuration is a bit tricky - Designed for analytical purposes, not for transactional purposes

	- API	- Security - Lack of support - Runtime errors
Informatica PowerCenter	- Easy to use - Support - Fast - Data quality monitoring - Data migration - Single point of - MDM data catalog - API - Flexible - GUI - Low code	- Lacks integration with other languages - No reporting functionality - A lot of settings interfaces - No ai capabilities - Connectors not working well

Appendix 3: Visualization tooling table

	Pros	Cons
Power BI: is an online SaaS service offer (SaaS) from Microsoft Power BI that displays interactive dashboards	- Not expensive - Easy to use - Receive constant updates - Integration with many sources - Able to handle large amounts of data - Sharing and collaboration features	- Limited free version capabilities - A table relationship is a bit lagging - Not able to handle large amounts of data in the free version - Simple dashboards easy to make, complicated dashboard challenging to make
Tableau: is a software that can help users explore and understand their data by creating interactive visualizations	- Connects users with a variety of data sources and enables them to create data visualizations by making charts, maps, dashboards, and stories through a simple drag and drop interface - Help users explore and understand their data by creating interactive visualizations - Easy to use by dragging and dropping	- Is still not widely used
Google data studio: is a new data visualization program	- Recognizable to anyone who works with the Google office suite - Free to anyone with a Google account - Interactive GUI - Sharing and collaboration features	- Cannot comply with IRB requirements for protecting personally identifiable data - Lacks the ability to modify underlying data - Offers fewer calculation and visualization options
IBM Cognos: provides a unified workspace for business intelligence and analytics	- Easy View - Collaboration - Transparent and accountable - Access data everywhere (mobile devices)	- Only a few sites and communities exist - Most experience has been gained by intuition and trial and error

	- Link dashboards to workflows	
Oracle Business Intelligence: This solution was designed to address the entire spectrum of analytical requirements	- Ad hoc analysis - Enterprise reporting - Microsoft Office integration	- High prices for large configurations
SAS: Software pack with various visualization capabilities	- Various visualization capabilities implemented in the product suite - Interactive - Allows the creation of basic queries and reports	- High costs
Qlik: Provides a possibility for end-users to use integrated ETL and to construct their data schema themselves	- Provides a clean interface to analysts - Removes the need to pre-aggregate data - Possible to change analysis axes any moment at any level of query detailing - Ability to automatically connect tables	- Lack of a unified metadata view - Lack of predicting models - Lacks advanced visualization features to help them graphically wade through complicated data
Amazon Quicksight: is a fast, cloud-powered BI service that makes it easy to build visualizations	- Superfast - Parallel - In-memory - Calculation Engine (SPICE)	- Immature - Sharing - Not available on android
Kibana: Kibana was designed as a visualization platform for Elasticsearch	- Interactive - Efficient - easy to extend to needs, able to show massive volumes of data	- No user management available
MicroStrategy	- Easy to use - Fast - Secure - Integration with different data sources - 2TB data size - Monitoring - GUI - Stable - Secure - API	- Timeout errors with more than 2 TB of data - Sharing - Support - Need more visualizations - Not flexible - No MDM
Redash	- Easy to use and setup	- Scalability problems - Technical knowledge required - Not flexible

	- No installation needed (browser-based) - SQL templates - Many data source integration - API - Easy export to different formats - Fast - Secure - Low-code and code - MDM	

Appendix 4: Interview setup

Hi ….,

Hope you are doing well. I am a CSA intern and team leader at Albert Heijn XL in Leiden, so for me the customer to interview for my thesis can hopefully be Ahold Delhaize. Can you please support me in reaching out to possible respondents?

I am currently working on my thesis about the use of data lakes. While data lake adoption is growing, the complexity and options for technology are growing as well. My research aims to provide a methodology for architecture design and tool selection based on business requirements.

The main question of the thesis is: "What method is best suitable for selecting the appropriate tools to rapidly and efficiently extract data from the data lake?"

It would be great if I can validate or mirror my methodology with people from Ahold Delhaize as a customer organization. The ideal roles would be a technology lead, architect, and data scientist.

The initial meeting would take 45 minutes

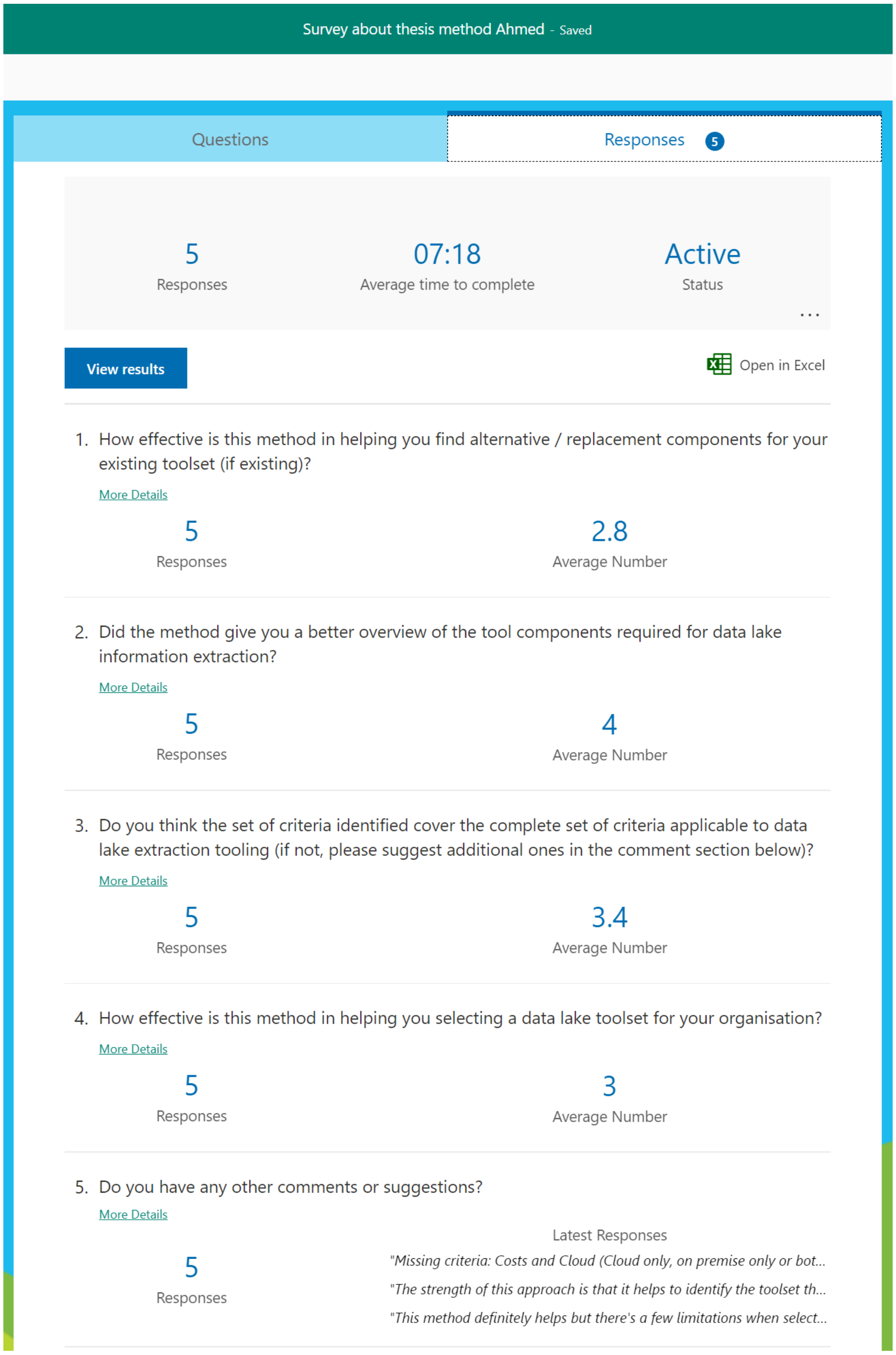
Of course, I will present my end result and share my thesis with them.

Thanks a lot in advance.

Kind Regards,

Ahmed Lachal
Cloud Solution Architect Intern

Appendix 5: Survey results



Survey about thesis method Ahmed - Saved

Questions | Responses 5

5
Responses

07:18
Average time to complete

Active
Status

View results

Open in Excel

1. How effective is this method in helping you find alternative / replacement components for your existing toolset (if existing)?

More Details

5
Responses

2.8
Average Number

2. Did the method give you a better overview of the tool components required for data lake information extraction?

More Details

5
Responses

4
Average Number

3. Do you think the set of criteria identified cover the complete set of criteria applicable to data lake extraction tooling (if not, please suggest additional ones in the comment section below)?

More Details

5
Responses

3.4
Average Number

4. How effective is this method in helping you selecting a data lake toolset for your organisation?

More Details

5
Responses

3
Average Number

5. Do you have any other comments or suggestions?

More Details

5
Responses

Latest Responses

"Missing criteria: Costs and Cloud (Cloud only, on premise only or bot...

"The strength of this approach is that it helps to identify the toolset th...

"This method definitely helps but there's a few limitations when select...

8. REFERENCES

Affetti, L., Tommasini, R., Margara, A., Cugola, G., & Della Valle, E. (2017). Defining the execution semantics of stream processing engines. Journal of Big Data, 4(1). https://doi.org/10.1186/s40537-017-0072-9

Anoshin, D., Rana, H., & Ma, N. (2016). Mastering Business Intelligence with Microstrategy. Zaltbommel, Netherlands: Van Haren Publishing.

ARGHIR, D. C., DUŞA, I. G., & ONUŢĂ, M. (2019). Organizational development through Business Intelligence and Data Mining. Database Systems Journal, 87–89. Retrieved from http://dbjournal.ro/archive/30/30_9.pdf

Armbrust, M., Ghodsi, A., Zaharia, M., Xin, R. S., Lian, C., Huai, Y., … Franklin, M. J. (2015). Spark SQL. Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data - SIGMOD '15, 1383–1394. https://doi.org/10.1145/2723372.2742797

Azarmi, B. (2014). Talend for Big Data. Birmingham, United Kingdom: Packt Publishing.

Bajer, M. (2017). Building an IoT Data Hub with Elasticsearch, Logstash and Kibana. ABB Corporate Research Center, 63–68. https://doi.org/10.1109/W-FiCloud.2017.40 Abramova, V., & Bernardino, J. (2013). NoSQL databases. Proceedings of the International C* Conference on Computer Science and Software Engineering - C3S2E '13, 14–22. https://doi.org/10.1145/2494444.2494447 Feuerstein, S., & Pribyl, B. (2005). Oracle PL/SQL Programming. Culemborg, Netherlands: Van Duuren Media.

Beheshti, A., Benatallah, B., Nouri, R., Chhieng, V. M., Xiong, H., & Zhao, X. (2017). CoreDB. Proceedings of the 2017 ACM on Conference on Information and Knowledge Management - CIKM '17. https://doi.org/10.1145/3132847.3133171

Birjali, M., Beni-Hssane, A., & Erritali, M. (2018). Evaluation of high-level query languages based on MapReduce in Big Data. Journal of Big Data, 5(1). https://doi.org/10.1186/s40537-018-0146-3
Madera, C., & Laurent, A. (2016b). The next information architecture evolution. Proceedings of the 8th International Conference on Management of Digital EcoSystems - MEDES. https://doi.org/10.1145/3012071.3012077

Bozdoc, D. (2011). Oracle BI Publisher 11g: A Practical Guide to Enterprise Reporting. Birmingham, United Kingdom: Packt Publishing.

Browne, D., Desmeijter, B., Dumont, R. F., Kamal, A., Leahy, J., Masson, S., … Keen, M. (2010). IBM Cognos Business Intelligence (1st ed.). New York, United States: IBM Redbooks.

Cattell, R. (2011). Scalable SQL and NoSQL data stores. Acm Sigmod Record, 39(4), 12-27.

Evans, C. (2006). HyperText Structured Query Language. Prometheus Research, LLC, 1–4. Retrieved from https://querycombinators.org/dist/icomp07_paper.pdf

Fuad, A., Erwin, A., & Ipung, H. P. (2014). Processing performance on Apache Pig, Apache Hive and MySQL cluster. *Proceedings of International Conference on Information, Communication Technology and System (ICTS) 2014*, 297–302. https://doi.org/10.1109/icts.2014.7010600

Godoe, P. and Johansen, T.S., 2012. Understanding adoption of new technologies: Technology readiness and technology acceptance as an integrated concept. Journal of European Psychology Students, 3(1), pp.38–52. DOI: http://doi.org/10.5334/jeps.aq

Grabova, O., Darmont, J., Chauchat, J.-H., & Zolotaryova, I. (2010). Business intelligence for small and middle-sized entreprises. ACM SIGMOD Record, 39(2), 39. https://doi.org/10.1145/1893173.1893180

Hartig, O., & Pérez, J. (2018). Semantics and Complexity of GraphQL. Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18. https://doi.org/10.1145/3178876.3186014

Informatica Corporation. (2014, June). Informatica PowerCenter. MicroQuill Software Publishing, Inc. Retrieved from https://kb.informatica.com/proddocs/Product%20Documentation/4/PC_961_GettingStarted_en.pdf

Katragadda, R., Sremath Tirumala, S. S., & Nandigam, D. (2015). ETL tools for Data Warehousing: An empirical study of Open Source Talend Studio versus Microsoft SSIS. The 2nd World Congress on Computer Applications and Information Systems. Retrieved from https://unitec.researchbank.ac.nz/bitstream/handle/10652/3366/ETL%20tools%20for%20Data%20Warehousing%20An%20empirical%20study.pdf?sequence=1&isAllowed=y

Khalifa, S., Bloor, C., Middelton, W., & Jones, C. (2000). Educational computer software, technical, criteria, and Quality. School of Computing Engineering & Technology University of Sunderland, 1–8. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.101.8180&rep=rep1&type=pdf

Khine, P. P., & Wang, Z. S. (2018). Data lake: a new ideology in big data era. ITM Web of Conferences, 17, 03025. https://doi.org/10.1051/itmconf/20181703025
Negrut, V. (2018). POWER BI: EFFECTIVE DATA AGGREGATION. Quaestus, (13), 146-152.

Klein, S. (2017). Azure Data Factory. IoT Solutions in Microsoft's Azure IoT Suite, 105–122. doi:10.1007/978-1-4842-2143-3_7

Klein, S. (2017, b). IoT Solutions in Microsoft's Azure IoT Suite. Redmond, Washington, USA: Apress. https://doi.org/10.1007/978-1-4842-2143-3

Ko I, Chang H. Interactive Visualization of Healthcare Data Using Tableau. Healthc Inform Res. 2017 Oct;23(4):349-354. https://doi.org/10.4258/hir.2017.23.4.349

Koskinen, J., Ahonen, J. J., Sivula, H., Tilus, T., Lintinen, H., & Kankaanpaa, I. (2005). Software Modernization Decision Criteria: An Empirical Study. Ninth European Conference on Software Maintenance and Reengineering. https://doi.org/10.1109/csmr.2005.50

Leibzon, Y., & Leibzon, Y. (2018). Redash V5 Quick Start Guide. Zaltbommel, Netherlands: Van Haren Publishing.

Li, J. Z., Ozsu, M. T., Szafron, D., & Oria, V. (1997, September). MOQL: A multimedia object query language. In Proceedings of the 3rd International Workshop on Multimedia Information Systems (pp. 19-28)
Torgersen, M. (2006). Language Integrated Query. RightsLink, 736. Retrieved from https://dl.acm.org/doi/abs/10.1145/1176617.1176700

Lomet, D., & Chaudhuri, S. (1999). Data Engineering. IEEE Computer Society, 22(2), 38. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.29.4396&rep=rep1&type=pdf#page=35

Lopez, G., Seaton, D. T., Ang, A., Tingley, D., & Chuang, I. (2017). Google BigQuery for Education. Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale - L@S '17. https://doi.org/10.1145/3051457.3053980

Lungu, I. (2015). Data integration approaches using ETL. Database Systems Journal, VI, 19–21. Retrieved from http://www.dbjournal.ro/archive/21/21.pdf#page=20

Lutz, M., Lewin, L., & Willison, F. (2001). Programming Python. Sebastopol, USA: O'Reilly.

M. (2017, October 11). U-SQL Language Reference - U-SQL. Retrieved from https://docs.microsoft.com/en-us/u-sql/

Madera, C., & Laurent, A. (2016). The next information architecture evolution. Proceedings of the 8th International Conference on Management of Digital EcoSystems - MEDES. https://doi.org/10.1145/3012071.3012077

Matloff, N. (2011). The Art of R Programming: A Tour of Statistical Software Design (1st ed.). San Francisco, USA: No Starch Press.

Miloslavskaya, N., & Tolstoy, A. (2016). Big Data, Fast Data and Data Lake Concepts. Procedia Computer Science, 88, 300–305. Retrieved from https://s3.amazonaws.com/academia.edu.documents/60526649/big-data-fast-data-and-data-lake-concepts20190908-41709-1pq148w.pdf?

Nikoukaran, J., Hlupic, V., & Paul, R. J. (1998). Criteria for simulation software evaluation. 1998 Winter Simulation Conference. Proceedings (Cat. No.98CH36274). https://doi.org/10.1109/wsc.1998.745014

Odersky, M., Spoon, L., & Venners, B. (2008). Programming in Scala. California, USA: Artima.

Oehler, K., Gruenes, J., & Ilacqua, C. (2012). IBM Cognos TM1 The Official Guide. New York, United States: McGraw-Hill Education.

OQL - Studio Pro 8 Guide. (2020, July 5). Retrieved from https://docs.mendix.com/refguide/oql

Palmer, N., Sferrazza, S., Just, S., & Najman, A. (2015). Market Reconstruction 2.0: A Financial Services Application of Google Cloud Bigtable and Google Cloud Dataflow. FIS. Retrieved from https://www.academia.edu/28969577/Market_Reconstruction_2.0_A_Financial_Services_Application_of_Google_Cloud_BigTable_and_Google_Cloud_Dataflow?sm=b

Quinto, B. (2018). Big Data Governance and Management. Next-Generation Big Data, 495–506. https://doi.org/10.1007/978-1-4842-3147-0_11

Snipes, G. (2018). Product Review Google Data Studio. Journal of Librarianship and Scholarly Communication, 6(General Issue). https://doi.org/10.7710/2162-3309.2214

Tsalgatidou, A., Pantazoglou, M., & Athanasopoulos, G. (2006). Specification of the Unified Service Query Language (USQL). Dept. of Informatics & Telecom. University of Athens. Retrieved from http://cgi.di.uoa.gr/~s3lab/TR/2006/usql-1.0-spec.pdf

Venkatesh, V., Morris, M.G., Davis, G.B. and Davis, F.D. (2003), "User acceptance of information technology: toward a unified view", MIS Quarterly, Vol. 27 No. 3, pp. 425-478

Williams, Michael & Rana, Nripendra & Dwivedi, Yogesh. (2015). The unified theory of acceptance and use of technology (UTAUT): A literature review. Journal of Enterprise Information Management. 28. 443-488. 10.1108/JEIM-09-2014-0088.

Zhu, W. D., Alon, T., Arkus, G., Duran, R., Haber, M., Liebke, R., … Redbooks, I. (2011). Metadata Management with IBM InfoSphere Information Server (1st ed.). New York, United States: IBM Redbooks.