Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Information Engineering

Bachelor Thesis

Data Integration
Techniques in Database Warehouse applications

Hamza Bacara

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

BACHELOR THESIS ASSIGNMENT

Hamza Bacara

Informatics

Thesis title

Data Integration Techniques in Database Warehouse applications

---

Objectives of thesis

The main objective of this bachelor thesis is to define the best way to use Database warehouse tool to improve Data Integration in order to generate a report from the main server and assist statistical analysis for any type of business.

The partial goals of the thesis are:
- to characterize the fundamentals of data warehouse and data integration,
- to identify possibilities of data integration tools, and
- to design data warehouse model.

Methodology

To achieve the goals will be need quantitative and qualitative data. In this thesis will be used secondary data which is already been collected from "adventure works" sample data bases. Primary data which author collected will be used.

The author of the thesis will use experimental data to create business reports from SSMS(SQL Server Management studio) and SSIS(SQL Server Integration Services) which is his main applications for this thesis. The author will continuously work with Visual studio and SSMS. The Visual Studio will be used to design the SQL statements, Loop Containers, Flat-file sources and by designing interfere the data to control and maintain the reliability. SQL Server Management Studio will be used to design database, procedure's, temporary tables and original tables.

The proposed extent of the thesis

30 – 40 pages

Keywords

Data Integration, SSIS, Database Warehouse, Visual Studio, SQL

Recommended information sources

ECKERSON, Wayne W. Performance dashboards: measuring, monitoring, and managing your business. John Wiley & Sons, 2010.

CHAUDHURI, Surajit; DAYAL, Umeshwar; NARASAYYA, Vivek. An overview of business intelligence technology. Communications of the ACM, 2011, 54.8: 88-98.

KIMBALL, Ralph; ROSS, Margy. The data warehouse toolkit: the complete guide to dimensional modeling. John Wiley & Sons, 2011.

TYRYCHTR, J. – VASILENKO, A. Business Intelligence in Agribusiness – Fundamental Concepts and Research. Brno: KONVOJ, spol. s r. o. , 2015, 100s. ISBN 978-80-7302-170-2.

Expected date of thesis defence

2021/22 SS – FEM

The Bachelor Thesis Supervisor

doc. Ing. Jan Tyrychtr, Ph.D.

Supervising department

Department of Information Engineering

Electronic approval: 1. 11. 2021

Ing. Martin Pelikán, Ph.D.

Head of department

Electronic approval: 23. 11. 2021

Ing. Martin Pelikán, Ph.D.

Dean

Prague on 29. 07. 2023

Declaration

I declare that I have worked on my bachelor thesis titled ***"Data Integration Techniques in Database Warehouse applications"*** by myself and I have used only the sources mentioned at the end of the thesis. As the author of the bachelor thesis, I declare that the thesis does not break any copyrights.

In Prague on 07.09.2023

Acknowledgement

I would like to thank my supervisor Jan Tyrychtr, for his unwavering guidance, patience, and invaluable insights. I am deeply honored to had him as a mentor.

I would also like to state my deepest thanks to my family. Their support, encouragement, and belief in me have been my source of strength. During these challenging times their sacrifices and love have been pillars of my academic pursuit.

Lastly, a special mention to my friend Deniz Tuna. His guidance and constructive criticism, and unique perspective have greatly enriched this work. Our countless discussions and brainstorming sessions greatly improved my ideas and approach. His friendship and undivided attention have meant more to me than words can explain.

Data Integration Techniques in Database Warehouse applications

Abstract

The focal point of this bachelor thesis centers on the exploration and application of advanced data integration techniques with the overarching objective of developing a robust database application tailored for warehouse management. This complex process is adeptly orchestrated through the utilization of four distinct yet synergistic software programs: **SSMS** (SQL Server Management Studio), **SSIS** (SQL Server Integration Services), and the versatile **Visual Studio**.

The main goal of this scholarly endeavor is to not only enhance the efficacy of data integration within a warehouse environment but also to establish a viable and adaptable model that holds the potential to revolutionize real-world warehousing practices. By amalgamating theoretical underpinnings with practical implementation, this thesis aspires to bridge the gap between conceptual frameworks and tangible results.

The practical facet of this bachelor thesis is described by a comprehensive demonstration of the intricate workflows executed within the aforementioned software programs. These elucidative demonstrations are further reinforced through the inclusion of meticulously captured screenshots, which meticulously detail each progressive step. These visual aids serve as a guide for readers, affording them a tangible grasp of the methodologies employed and the intricacies involved.

Keywords: Data, integration, SQL Server Management Studio, SSIS, SQL, Microsoft, server.

Techniky integrace dat v aplikacích databázového skladu

Abstrakt

Hlavním bodem této práce je zkoumání a aplikace pokročilých technik integrace dat s hlavním cílem vyvinout robustní databázovou aplikaci přizpůsobenou pro správu skladů. Tento složitý proces je obratně řízen pomocí čtyř různých, avšak synergicky působících softwarových programů: SSMS (SQL Server Management Studio), SSIS (SQL Server Integration Services) a univerzálního Visual Studia.

Prvořadým cílem tohoto vědeckého úsilí je nejen zvýšit účinnost integrace dat v prostředí datového skladu, ale také vytvořit životaschopný a přizpůsobivý model, který má potenciál revolučně změnit praxi reálného warehousingu. Spojením teoretických základů s praktickou implementací se tato práce snaží překlenout propast mezi koncepčními rámci a hmatatelnými výsledky.

Praktická stránka tohoto výzkumu se zabývá komplexní ukázkou složitých pracovních postupů prováděných v rámci výše uvedených softwarových programů. Tyto názorné ukázky jsou dále posíleny zahrnutím pečlivě zachycených snímků obrazovky, které pečlivě popisují každý postupný krok. Tyto vizuální pomůcky slouží čtenářům jako průvodce a umožňují jim hmatatelně pochopit používané metodiky a složitosti.

Klíčová slova: Data, integrace, Studio pro správu serveru SQL Server, SSIS, SQL, Microsoft, server.

Table of content

1. Introduction

In today's fiercely competitive market, businesses are constantly striving to gain a significant edge over their rivals. To achieve this, they recognize the importance of harnessing the potential of emerging data sources. However, traditional technologies fall short when it comes to analysing enormous amounts of data or processing data streams in real-time. As a result, an increasing number of businesses have made substantial investments in innovative technologies capable of effectively processing such data. It's worth noting that the utilization of these advanced data processing methods often comes with considerable costs.

Among the various approaches available, data warehousing stands out as an exceptionally successful means of transforming vast quantities of organizational information and knowledge into accurate, relevant, and actionable insights. By centralizing the storage of information and knowledge, data warehousing enables businesses to access meaningful answers to critical questions, thereby simplifying the decision-making process. Furthermore, the methods and technologies employed within a data warehouse, such as ELT (Extract, Load, and Transform), ensure that the data is appropriately prepared for analysis and immediate utilization. The comprehensive process of extracting, converting, and loading data, commonly referred to as ETL or ELT, plays a pivotal role in safeguarding the highest possible quality and accuracy of the data housed within a data warehouse.

Within the context of this thesis, the author aims to apply the knowledge acquired in relation to data transfer, dimensional table creation, and testing the data loading, transferring, and analysis. The work is divided into two distinct chapters: the theoretical part and the practical part. The theoretical section equips the author with insights into potential challenges that may arise when working with data, including data mismatch, integration problems, and cross-database joint issues, among others. By identifying these potential obstacles, the author can develop strategies to address them effectively. This proactive approach ensures the smooth and efficient execution of data handling processes.

The practical part of the thesis involves applying the theoretical knowledge to real-world scenarios. By leveraging the author's understanding of data transferring, dimensional table creation, and various testing methodologies, the practical chapter serves as a platform to validate and demonstrate the effectiveness of the proposed strategies. It allows the author to navigate potential hurdles that may arise during the implementation phase and provides

an opportunity to fine-tune the data handling processes based on practical experience and empirical findings.

By combining theoretical insights with practical application, this thesis aims to contribute to the body of knowledge in data management and analysis. The author seeks to provide valuable insights into the challenges and solutions associated with data handling, enabling businesses to make informed decisions and optimize their data-driven operations. Ultimately, the comprehensive exploration of data transferring, dimensional table creation, and testing methodologies paves the way for enhanced data quality, accuracy, and overall organizational performance in an increasingly data-centric world.

2. Objectives and Methodology

2.1 Objectives

The main objective of this bachelor thesis is to define the best way to use Database warehouse tool to improve the “Data integration in order to generate a report from the main server and assist statistical analysis for any type of business.

The partial goals of this thesis are:

- To characterize the fundamentals of data warehouse and data integration
- To identify the possibilities of data integration tools and
- To design data warehouse model

2.2 Methodology

Within this methodology section, the author describes the tools and software utilized in the practical part.

1. **SQL**

When it came to data searching and management for my thesis, SQL was an invaluable tool. It was the main language used for data extraction, manipulation, and the preparation of datasets for further analysis and display. To be more specific, SQL queries were used to create results, such as the ones shown in Figure 6, in which data was picked and aggregated to offer the required insights. These findings were then analyzed in order to draw conclusions.

2. **SQL Server Management Studio (SSMS)**

The System Center SQL Server Management Studio (SSMS) is a combined platform developed by Microsoft for the purpose of managing and administering SQL Server databases. It offers a set of tools that may be used to configure, monitor, and manage editions of SQL Server. In the context of this thesis, the SQL Server administration Studio (SSMS) was indispensable for database administration, the execution of SQL queries, and data exploration.

3. **Visual Studio with SQL Server Integration Services (SSIS)**

This study would not have been possible without the participation of Visual Studio and its SSIS capabilities. The DBCenter SSIS Package is a set of data integration tasks and flows

that are meant to operate together to retrieve, manipulate, and integrate data. This package was created using Visual Studio. The package contains many diagrams, each of which represents a particular data job, and the flow between these diagrams specifies the order in which these tasks are performed as well as any relationships between them. The arrows reflect the direction in which the data flow, and the various shapes stand for the various operations that may be performed on the data, such as extraction, transformation, or loading.

4. **Microsoft Report Builder**

Utilizing this tool to create visual representations of the processed data was very necessary. Picture 13 is an example of something that was made using Microsoft Report Builder. A rich environment for planning, creating, and publishing paginated reports is provided by the program. In this thesis, it was used to graphically portray statistical insights that were gained from the modified data. This helped to make understanding and interpretation of the data much easier.

5. **Data and Analysis**

The Adventure Works sample databases served as the major source of information for this investigation. After the data from these databases had been retrieved using SQL, the DBCenter SSIS Package in Visual Studio was used to modify it, and then it was imported into the environment of choice for analysis.

The modified data was input into Microsoft Report Builder so that it could be represented visually in images such as Picture 13. This visualization provides a statistical insight into the data by highlighting certain variables (such as patterns of sales or customer profiles). The process of developing such visualization begins with the aggregation of data according to the measure of interest, followed by the selection of a suitable chart or graph style that may best show the findings.

3. Literature Review

3.1 Key definitions

The significance of several essential concepts, when applied to the context of data integration, may regularly start to be viewed in a totally different manner (Macura, 2014). In addition, the distinct business subjects and information settings, such as the governmental sector, financial services, logistic, healthcare, and scientific disciplines, may require a unique approach to the data processing and translation processes. It has to be represented with the help of the proper instruments, which are competent to translate into knowledge and data that is helpful. The author will list down a few commonly known terms in regards of the data, integration of data and other terminologies that will be mentioned in the following chapters.

3.1.1 Data

Data, for the most part represents a structured codification of single primary entities, as well as stored and processed involving two or more primary entities (Macura, 2014). According to the Tous (2008), the term "data" refers to anything or form that is potentially accessible by a person or a figure in a processor for the reason of maintaining, analyzing, or sending over a digital channel. This could come in the shape of a picture, a number, text, or any other kind of storage. The information is presented by the processor in a form that is binary, meaning it's meaningless. The significance of the information is determined by performing process in order for the data to turn into information. As data, we regard things like bicycles, televisions, and cups to be examples of real-world objects. Data may be stored and retrieved, as well as elaborated using software processes and sent using an internet connection. In addition, it has these capabilities (Stefan and Cichy, 2019).

3.1.2 Information

Macura (2014) describes that information is the result of the collection of data and its subsequent processing. For instance, every student in the course completed the test, and the student's scores can be treated as data. On the opposite hand, the student's score, percentage terms, and place within the classroom are regarded as data Tous (2008).

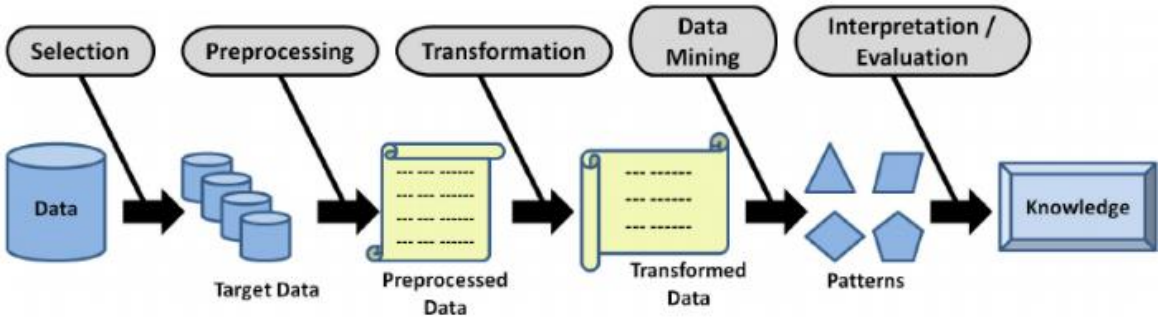
3.1.3 Knowledge

Whenever information is utilized to create decisions and the development of actions that correlate with those decisions, it is converted into knowledge (Macura, 2019).

Knowledge, that consists of information as well as collectively implements it into a specific working area, enhances via the practical understanding and competence of top management in addressing and responding to complicated challenges. Knowledge is a collection of information. The authors presented a strategy for acquiring information known as "***knowledge discovery***" which included carrying out a number of processes:

- Cleansing in data integration
- Data selection
- Data transformation
- Analytical technologies
- Information interpretation

Picture 1: Knowledge discovery database.



Source: Gullo (2015)

3.1.4 Metadata

In brief, metadata can be defined as "information regarding other data." – Tous (2008: p.45). Its purpose is to condense the information regarding the data while making it more easily identifiable. Metadata consists of organized information that exemplifies what makes it more available for collecting and processing, or controlling, a data resource. It may also be referred to as "information about information." For example, in the header portion of the HTML document, we're able to view the Metadata, that offers a description of the information about the website's content.

3.1.5 XML

Based to the description of XML, the Extensible Markup Language, XML is an extra textual syntax that might be utilized to explain formatting, actions, structure information, text semantics, attributes, and so on. In most cases, the content of XML will consist of key values label, each of them will have values included within it. The goal of the text file format known as XML, which can be interpreted by both humans and machines, is to organize and explain the data that is being stored. It is usual practice to utilize an XML file while setting up the configuration of a system's settings (Macura, 2014).

3.1.6 Semantic WEB

The Semantic Web is a collection of internet technologies led by the W3C that offer a data model to the web (Miguel, Victor, and Ivan, 2009). These innovations enable it to be feasible for both humans and machines to interpret information. In addition, the processes impart a clear meaning onto the information, making it possible for robots automatically interpret and incorporate it (Tous, 2008).

It is built on the Resource Description Framework (RDF), which unifies a number of different applications by making use of XML for the syntax and URIs for the name. It makes the process that the machine uses to find information more efficient. As a result, we are able to put this technology to use in a manner that represents information in data sources that successfully manage structures and makes the method of data unification easier. The following categories are the fundamental components of these technologies:

RDF – Resource Description Framework allows the information to be presented in the form of metadata (Tous, 2008). It defines what is referred to as triplets on the internet and includes a subject, a predication, and a component. The predication is what establishes the connection between the subject, which stands for the resource, and the object, which is identified by its URL. Connectivity among apps which exchange information that can be read by machines is made possible via RDF. The Resource Description Framework (RDF) makes it possible to provide a system for describing resources without first developing a specific application (Miguel, Victor, and Ivan, 2009).

RDF is beginning to be utilized in a wide variety of application areas, including discovering resources, cataloguing (contents connection), material evaluation systems, and the protection of intellectual property. RDF follows an arrangement for its class structure that is identical to that of object-oriented design systems. RDF contains a model, that's a set of classes arranged in a hierarchy. The structure offers extension via the modification of subclasses. The capacity to reuse metadata descriptions is enabled by the RDF standards.

OWL – Ontology Web Language, is a set of words that may be used to represent the characteristics and classes of RDF resources. The RDFS language is improved by OWL by clarifying, among other things, the connection between subclasses. OWL, OWLDL, and OWL complete are the three sub-languages that are included in OWS. OWL Lite is the most basic. The W3C has established OWL as a form of speech that may be utilized to convey the meaning in terms of vocabularies and the connection between concepts. Therefore, we decided to name it Ontology (Dong, Halevy and Yu, 2009).

OWL is described as “A formal and explicit specification of a shared conceptualization" as stated (Cichy & Stefan, 2019). This serves as an explanation of the ideas and connections that could be present across one agent or for a whole population of individuals. According to the writers, the language for ontology was used to construct the highest degree of abstractions in the information merging phase. Although it is able to express the significance of the language which is used in the strategies that were collected from various reports, OWL deserves to be proposed by us as a solution to this problem. As a result, the computer will be able to successfully complete reasoning assignments using those methods.

3.2 Data Integration

The problems of data integration consist of integrating data from different places and providing the end user with a consolidated picture of the information. The production of reliable access across a variety of sources of information is the goal of integrating data, which was developed to achieve this goal (Al-Saduiry and Vasista, 2011). Integration of data is essential in big organizations that have various sources since it contributes to improved search performance across the millions of pieces of data that are available on the World Wide Web and improves collaboration between government agencies. Data integration is a field of study which studies and addresses an issue that is encountered in applications that require to query across various independent and heterogeneous data sources. It does this by gathering information about the problem from a variety of resources.

3.2.1 Types of Data Integration

The integration of data from a variety of resources is carried out in accordance with the requirements posed by the users. The requirements are portrayed through a perspective known as the unified viewpoint, also known as the global paradigm (Dong, Halevy and Yu, 2009). Materialized View and Virtual View were the two primary strategies that were tested in an effort to find a solution to the challenge.

Materialized view approach: The Materialized View Approach is quite similar to the materialized view that is used in database systems. The process involves saving all of the information from all sources locally and then accessing it. The materialized view method is illustrated by the data warehouse, which is an established example. According to Dong et. el. (2009), part of the filtered content which has been extracted from various resources is stored in a storehouse (also known as a warehousing) in order that it may be searched by customers at a later time. One can also refer to this method of data integration as an enthusiastic method (Cichy & Stefan, 2019).

Virtual view approach: The Virtual View Approach is a method that seeks to provide the identical outcomes without needing to retain and constantly update each of the information coming from its many sources. The searches that are posted via it are handled by an auto-acting rescript at the point they're being processed and then transmitted to the

actual sources of data. According to Dong et el. (2009), whenever someone sends a request to the software, the data can be obtained from various sources on-demand. This is stated in the system's user documentation. One may also refer to this kind of integrating data as a "a lazy method".

3.2.2 Data Warehouse

Based to the description provided by the American computer specialist Bill Inmon, data warehouse is an accumulation of information that is subject-oriented, integrated, non-volatile, and time variable in order to assist in managerial decision-making processes (Macura, 2014).

The data are arranged and connected according to their subject matter. Typically, there is a distinct collection of topics that are specific to the organization, and some examples of these are customers, real estate, sale orders, and products. A data warehouse collects collected information through a variety of data sources and integrates it all together. When database is stored in a data warehouse, it is given a unified and consistent representation of the company (Phungtua-Eng, 2019). Monitoring and recording the time-variable changes that occur to the data stored in a data warehouse. data in a data warehouse that is non-volatile after it has been carried out data that cannot be overwritten or eliminated data that is stable and only accessible through reading, and data that is kept for purposes of reporting. In the event that the upgrade is carried out, a fresh snapshot will be created.

Back-room and front-room are the respective names for the two sections that make up the data warehouse. The data are processed to provide information, which is then brought through the front room. The backroom serves as a staging area and is equipped with a variety of resources for the sole purpose of documenting and recording operations. The term "extract-transformation load" (ETL) is widely used to describe the numerous different procedures that are carried out in the back office (Agrawal, 2010).

3.2.3 ETL process

The Extract, Transform, and Load (ETL) phase is a vital component of the data integration phase. During this process, data from a variety of resources are transported to data warehouse systems and given additional requirements (Azeroual, Saake and Abuosba,

2019). In order to move data from one or more source databases to a single or multiple destination data bases, ETL methods are utilized. The ETL procedures were broken down into the individual ETL operations.

The most important step in constructing a data warehouse is called ETL, which stands for "extract, transfer, and load." ETL refers to the transformation that takes place when moving data from its original sources into a data centre (Cristóbal-Salas, 2019). The method of conversion carried out by ETL to convert operation-data into decision-data for storage in a data warehouse includes a number of crucial functions and improvements regarding the data. ETL operations are liable for integrating, restructuring, and combining an enormous amount of information via multiple sources, in addition to providing data to a DW". ETL could be categorized into 5 different groups (Theodorou, 2017) such as:

Extraction of data to consolidate information from several sources into a usable format. The initial step in extracting useful information from data that is stored in a variety of formats and structures is called "extraction" in the ETL process (Dumitriu, 2020). There are two options considered before one is settled on for data extraction. Before continuing with processing, **data validation** performs a number of checks, including those for the foreign-key, properties, quality, and effective value of the data. Incomplete or inaccurate data, data that has been copied, and so on are all eliminated throughout the **data cleaning** process. **Data Conversion** encompasses a wide range of processes, including but not limited to date conversion, string-format conversion, and date operations. **Data loading** refers to the action of transferring information into the tables of a data warehouse. Common approaches include using a SQL query or a batch-loading program. In terms of authors, they divide ETL processes into three distinct categories (Dimitriu, 2020).

3.2.4 ELT process

The chapter is a part of the practical part within this work. The author bases the work on the ELT technique.

The most noticeable distinction is that the ELT process executes the Load function before the Transform function. This is the opposite order in which the ETL process does the second and third phases. ELT will either copy or export the data from its source locations; however, rather than transporting the data to a staging place where it will be converted, it

will load the raw data straight into the target data store. From there, the data may be modified as required. During transmission, ELT does not alter any of the data in any way (Vyas, 2017). When using the ELT methodology, data are imported into the warehouse or data lake in their original format without undergoing any transformations beforehand. since of this, configuring tasks is simplified since it is only necessary to provide an origin and a destination (Vyas, 2017).

The processing of enormous amounts of data, such as those needed for business intelligence (BI) and big data analytics, is where ELT shines the brightest. Because the data is copied "as is" from the source, an ELT technique works better with non-relational and unstructured data than it does with relational data. A "schema on read" technique is often used when analytics are applied to unstructured data. This is in contrast to the standard "schema on write" approach that is utilized by relational database systems (Theodorou, 2017: Vyas, 2017).

3.2.5 Data Integration Tools

The need for data integration in business has led to the proliferation of data integration software. However, data is generated swiftly and unexpectedly in the context of dynamic businesses, which may render the DI tool's processing of them wasteful and out of date. That's why it's so challenging to find a DI tool that can provide entirely computerized assistance for checking off all the boxes on the many quality requirements that must be satisfied throughout the manufacturing process. There is still a significant amount of human labour needed from the designers (Theodorou, 2017).

Theodorou (2017) used a Kettle ETL tool for the suggested research. Previously known as Kettle ETL, the freely available solution for integrating data has been rebranded as Pentaho Data Integration. Data integration software Kettle or Pentaho contains four phrases:

(1) Collection

(2) Transfer

(3) Processing

(4) storage.

3.2.5.1 Kettle method

Kettle/Pentahottle is a set of utilities for working with data from several sources. It's made up of these five parts (Theodorou, 2017):

Spoon – provides a visual user interface for designing ETL process modifications. Data cleansing, studying, confirming, converting, and writing are all carried out in this pane for each of the data sources and the ultimate destination datastore. Changes provided by Spoon may be used in conjunction with additional appliances like the Pan and the Kitchen. **Pan** is a program that may be used to put into action Kettle Spoon-created data transformations. **Chef** had to develop tasks to automate the laborious process of updating databases. To simplify starting and controlling ETL processing, **Kitchen** was designed to run tasks in a batch mode. **Carte** it is a web server that allows the entire software to be monitored and controlled remotely. Microsoft SQL Server, IBM DB2, PostgreSQL, InterSystems Caché, Informix, Sybase, dBase, Firebird SQL, MaxDB (SAP DB), Hypersonic, CA Ingress, SAP R/3 System, and other DBMS employing ODBC driver are just some of the data types **Kettle** is compatible with (El-Sappagh, 2017).

3.2.5.2 TPC – DI method

For the TPC-DI benchmark, the DIGen source codes are often the starting point. DIGen is built based top of what is known as PDGF, or Parallel Data Generation Framework. The PDGF is enhanced so that it can generate a data collection meeting the strict criteria of this standard. PDGF's highly parallel and adaptable design makes it an ideal foundation for data production on a cloud-scale. This system uses two XML files to describe and distribute configuration data, which is then used to produce dissemination of predefined sets of data. The structure is necessary for the subsequent DIGen rules, as stated by Yang and Helfert (2017):

- In the first place, you need to use DIGen to generate your inviolable sources of information.
- Both the standard and DIGen must be up-to-date if PDGF is intended to be used.
- DIGen is recommended for usage with a minimum of Java SE-7 while developing the data source.

3.2.5.3 SSIS

Data movement has never been easier than using Microsoft's SQL Server Integration Services (SSIS). Since the integration and modification of data takes place entirely in memory, the time required to complete the operation is drastically reduced. Because it was developed by Microsoft, SSIS works solely with the Microsoft database management system. Specifics of SSIS by (Cristóbal-Salas, 2019: Vyas, 2017):

- Commercially recognizable tool
- Export/Import of wizard delivers the data to its final destination
- Drag/Drop graphic interface for designing SSIS packages
- Integrated scripting language for programmers
- Simple access to error management and debugging tools

3.2.5.4 ODI

Oracle Data Integration - (ODI) is a visual interface for creating and controlling data integration. If your company is huge and requires regular migrations, this package is for you. High-volume, service-oriented architecture (SOA)-enabled data services are supported by this complete data integration platform. Specifics of ODI:

- Commercially licensed tool
- Improved usability by rethinking the model's underlying flowchart structure
- Rapid, low-effort design and upkeep; facilitates declarative design methodology for data integration and transformations
- An important function that may detect invalid information and discard it before sending it to the final destination application.
- Countless options for connecting to databases, including IBM DB2, Tera-data, Sybase, Netezza, Exadata, and more.
- Utilizes pre-existing RDBMS capabilities via integration with Oracle products for data processing and transformation.

3.2.5.5 SSMS

SQL Server Management Studio (SSMS) is an integrated environment that can handle any SQL facilities from SQL Server to Azure SQL Database. This includes both on-premises and cloud-based SQL databases. The SQL Server Management Studio (SSMS) offers users with the tools necessary to setup, monitor, and manage SQL Server instances as well as databases. Build queries and scripts, as well as deploy, monitor, and update the components of the data tier that are utilized by your applications, with the help of SSMS. A user may query, construct, and administer databases and data warehouses using SSMS, regardless of where they are located: on personal machine, in the cloud, or somewhere else. There is a feature of SQL Server Management Studio, which is called Object Explorer. It actually serves as the primary interface via which, users may navigate, select and take action inside the server (Shields, 2019).

3.3 Data Problems

Data problems might occur in a completely different manner. For example, A direct incompleteness dimension is represented by a number that is absent. During the process of data integration, a data value is considered to be missing if that column's data includes a zero or a blank value.

3.3.1 Quality problems of data

Data is the most essential and fundamental component prior to extract to information and transform them into knowledge, which is used in data analytic to drive numerous elements, such as understanding the consumer pattern, strategic-planning, policy development, and making choices, which was have discussed in the previous part (Knowledge Discovery and Background Information). The accuracy and quantity of data collected are both crucial. A little issue in the quality of data, for instance, might cause incorrect conclusions to be drawn and steer decisions in the incorrect direction (Yang, Ge, and Helfert, 2017). So, it's clear that having high-quality information is crucial to the performance of most businesses and makes a big difference for those establishments. Both governmental and commercial sectors have come to appreciate the importance of high-quality data to their operations. Poor data manipulation is the outcome of poor data quality, that in turn renders the information unusable and has far-reaching consequences for further analysis.

Quality is defined as "suitability for usage" as well as " the extent in which a collection of inherent features satisfies the criteria and the degree to which a set of characteristics of data fulfill the specifications. Based to the aforementioned definitions, low-quality data may be thought of as a collection of data properties that falls short of expectations. When data consumers can't rely on complete and accurate requirements, it might have an effect on the firm, the client, and the manufacturing process.

3.3.2 Types of Errors

3.3.2.1 Missing Values

It's a direct incompleteness dimension because of the missing value. If a single row of input data is missing or includes nulls, this is treated as an error in the data integration procedure.

Inquiry results clearly show where a missing value exists. The following examples illustrate the two most common kinds of missing value:

1) Missing Value in Fields it signifies the scenario where an obligatory value or non-null requirement value do not exist in the given field. This missing number often shows up and has the potential to significantly alter the result. For instance, if the found-date column in the database that stores the company's history lacks a value, it can have an effect on the company's grade.

2) Filling in the Blanks in an Old Record. This occurs when some rows have been updated with new information but others have been left unchanged. This situation is typical of a data warehouse, where instead of updating current values, records are designated as obsolete or inactive, and new entries are created and given the appropriate key.

3.3.2.2 Spelling errors

The common problem of incorrect spelling of scientific names is identified here as the primary data quality issue in taxonomic databases (Dalcin, 2005). This issue is connected to both the substance of the data and the underlying context of data quality. There are a number of ways in which spelling mistakes might manifest themselves; it would seem that the following three spelling mistakes are the most prominent ones (Hong, 2000; Dalcin, 2005).

The lack of familiarity with writers' names appears to be connected to the discrepancy between how the names are pronounced and how they are spelled. This can result in consistent spelling errors. Typographic issues during typing can also lead to errors. Specifically, the term "error" refers to the incorrect positioning of a key on the keyboard, commonly known as a typo. Although typos can be inconsistent, they may also exhibit some predictability. Furthermore, mistakes can occur during the transfer, conversion, and storage of data. These errors can be attributed to specific coding and transmission technologies, such as the utilization of optical character recognition (OCR) during the data input process.

In regard to various recurring types of misspellings, they might be divided into the following three categories (Irmert, et. el., 2004):

(1) Errors caused by typography

(2) Cognitive errors

(3) Mistakes in the phonetics.

Incorrect spelling, which accounts for around 80 percent of a single incidence, may include insertion, removal, replacement, or transposition of words. There are two categories of misspellings:

(1) single-error misspellings and

(2) multiple-error misspellings; misspellings that include several errors, often known as multi-error misspellings.

Figure 1: Example of missing values.

ClientID	ContactNumber	Email	TransType
19885	+415 655	Dezho9@gmail.com	New
19885	+415 655 7655	NULL	Update
19885	NULL	Dezho19@gmail.com	Update

Source: Cichy, & Stefan (2019)

3.3.3 Data problems in ETL

It is not unusual for there to be issues with the data throughout the ETL (Extract, Transform, Load) process, which may occur for a variety of different causes. The following is a list of typical data issues that could arise during the ETL process (Cichy & Stefan, 2019) along with some possible remedies to these issues:

- Quality of data issues: Data quality concerns might include missing values, wrong formats, discrepancies, or duplication. Implementing data validation criteria, using data cleaning methods, doing data profiling in order to find abnormalities, and establishing data quality monitoring systems are some of the options available to you for resolving these difficulties.
- Mistakes in data transformation: While the data is being converted, it is possible that the data will not be translated appropriately in accordance with the business rules or data mappings that were planned. In order to guarantee that the transformation logic is accurate, it is essential to test and verify it in great detail. The implementation of logging

and error handling systems is recommended in order to record and correct any transformation mistakes that may arise.

- Violations of the data integrity: There is a possibility that ETL procedures may run into data integrity problems, such as referential integrity limitations or primary key violations. In order to identify and deal with breaches of this kind, appropriate validation and error handling procedures should be in place. This will help to guarantee that the data will be accurate and consistent.
- ETL operations might run into performance challenges if they have to deal with bottlenecks in the form of large amounts of data or sophisticated transformations. Improving speed and removing bottlenecks may be accomplished by optimizing the workflow of the ETL tool, fine-tuning the database setups, implementing the proper searching, and using methods for simultaneous processing.
- Retrieval of data that is either insufficient or inconsistent: While the information is being extracted, there is a possibility that some of the information may either not be retrieved completely or will be retrieved inconsistently. The use of techniques such as change data capture (CDC) or ensuring that there is enough data synchronization between the original systems and the ETL process might assist to solve these challenges.
- Data privacy and security issues arise because ETL operations include the processing of sensitive data. As a result, it is vital to apply suitable security measures, such as encryption, access restrictions, and data masking, in order to safeguard the data along the whole ETL pipeline.

4. Practical Part

In this chapter, the author describes the steps of practical part. In order to perform all the processes, the author works in three different programs, SQL Server Management Studio, Visual Studio and Microsoft Report Builder – Power BI Report Builder.

4.1 Microsoft Report Builder – Power BI Report Server

Users are able to generate, design, and publish paginated reports with the help of Microsoft Report Builder, which is a client-side reporting tool created and distributed by Microsoft. Traditional, pixel-perfect, and printed reports with predetermined layouts are known as paginated reports. These reports are often used in reporting circumstances pertaining to both operations and transactions.

The author particularly uses the Version: 15.0.20073.0, with a file name: ReportBuilder.msi.

System requirements

- Windows 10, Windows Server 2019, Windows Server 2022, Windows 11
- 80 MB of available hard disk space
- 512 mb of RAM

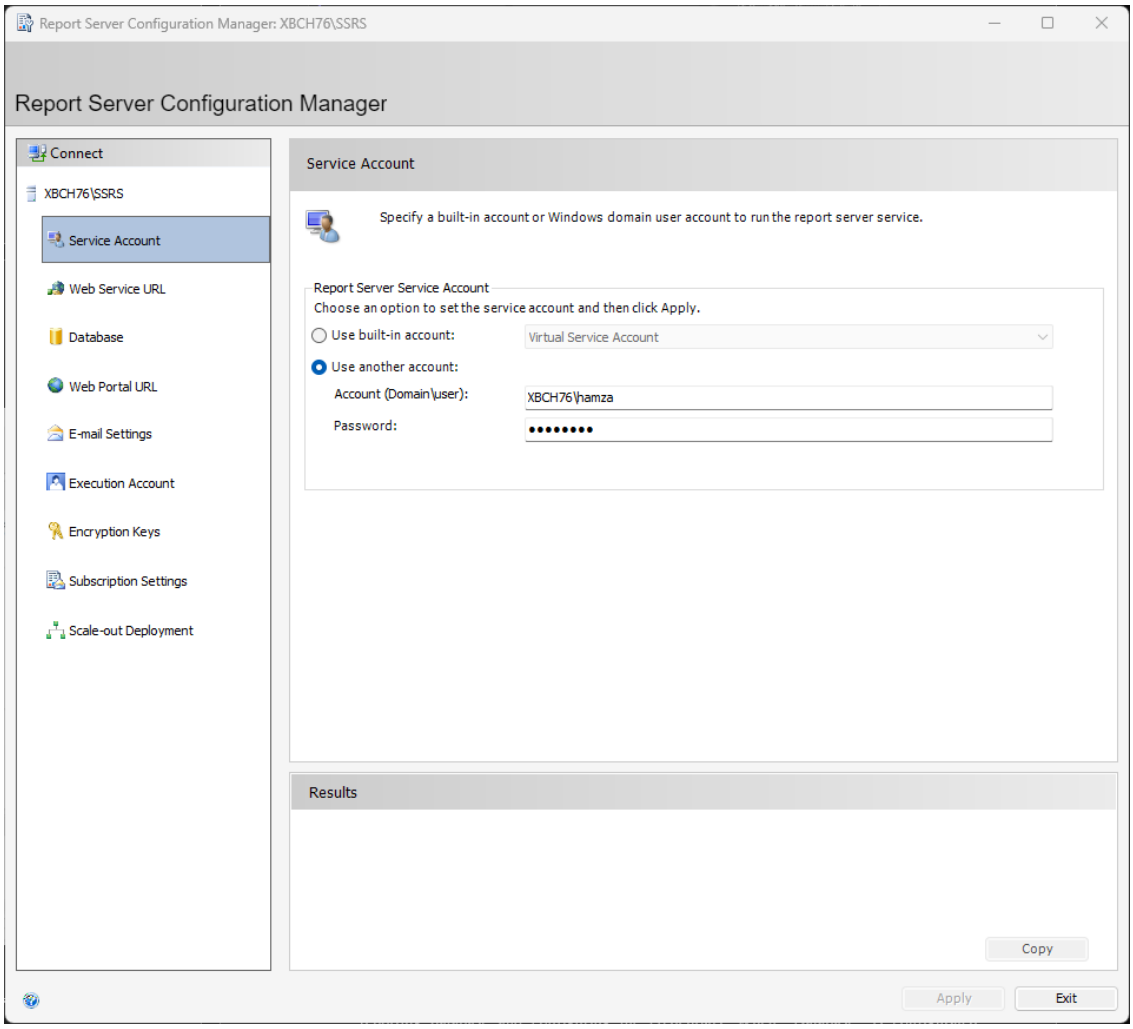
NOTE: This component also requires **Microsoft .NET Framework 4.6**

Installation steps:

- Install Microsoft .NET Framework version 4.6 (In our case we did when we downloaded Visual Studio)
- Install Report Builder from Microsoft

We start with configuring the Report Server Before we can begin running the SSRS in Visual Studio or Report Builder app, we establish the network server, reporting database, web service URL, and create a backup of the server in case of an emergency.

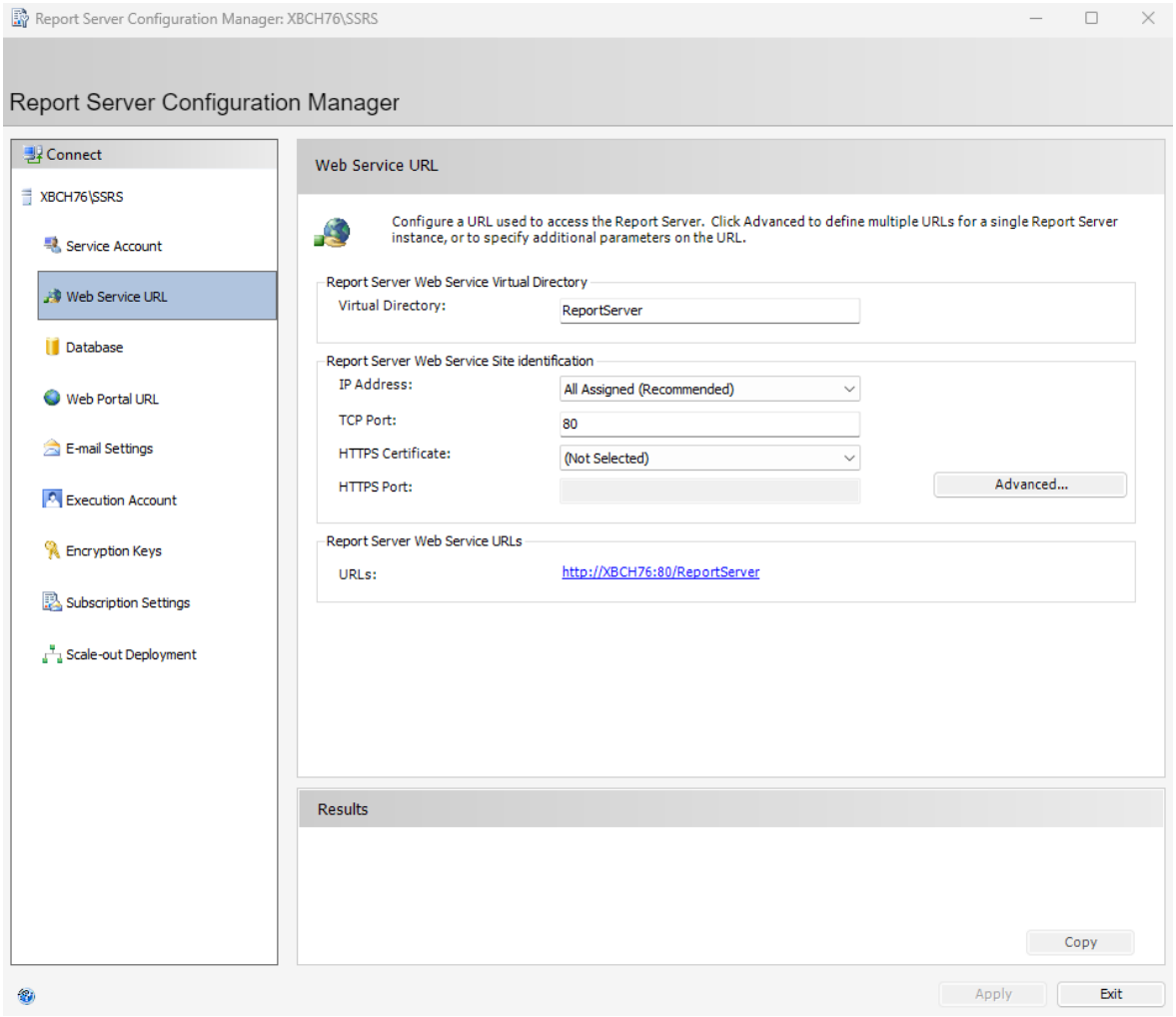
Picture 1: RSCM - Service Account Configuration.



Source: Own processing.

Above, we are configurate the Service Account Configuration, administrator user is given which is written in “domain\user” style.

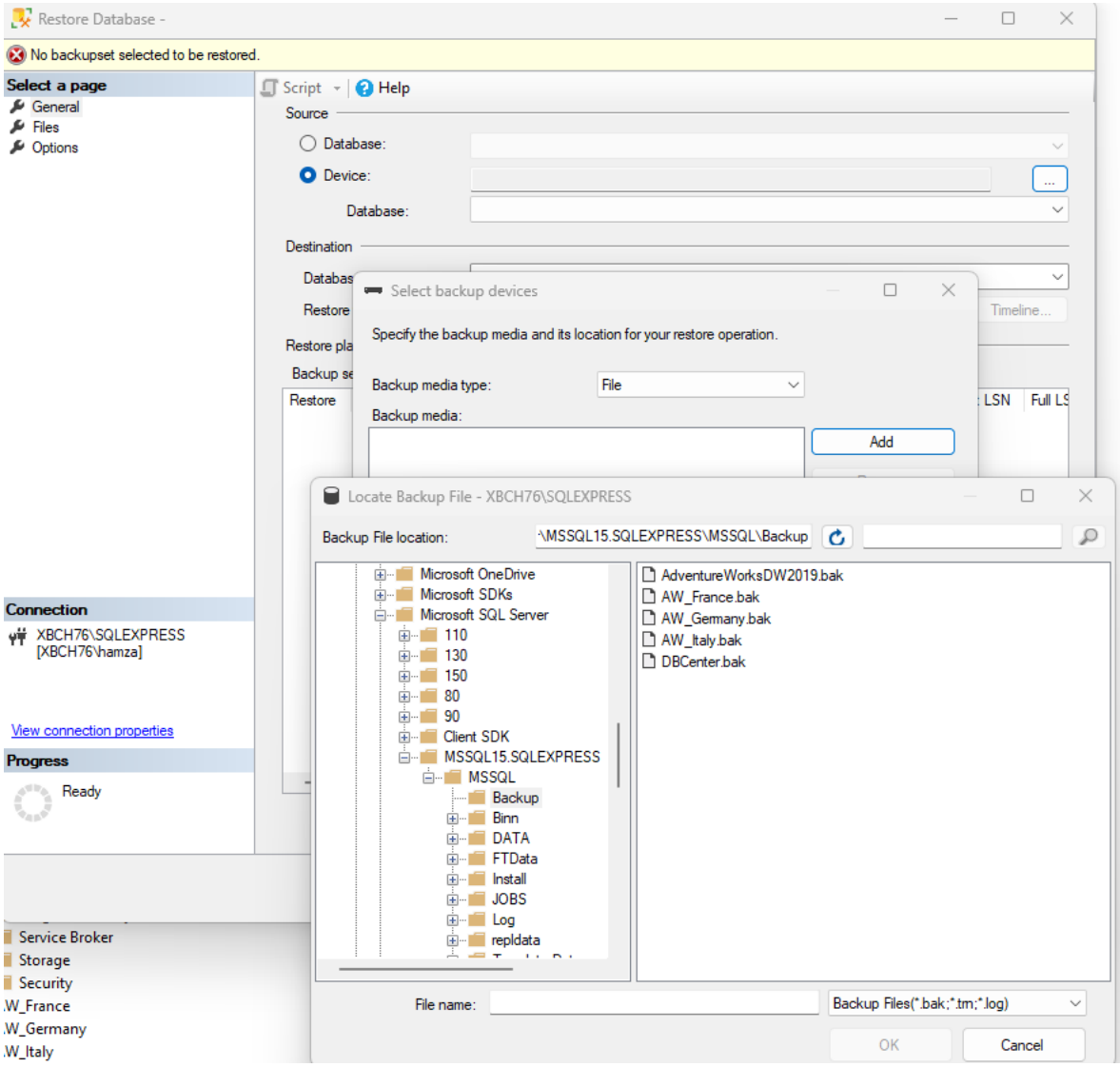
Picture 2: RSCM - Web Service URL configuration



Source: Own processing.

When the configuration of „Web services "is done, we move to the Database tab to create the reporting database and configure the credentials. When "Database" is configured, the administrator can create Paginated Reports.

Picture 3: Restoring database.



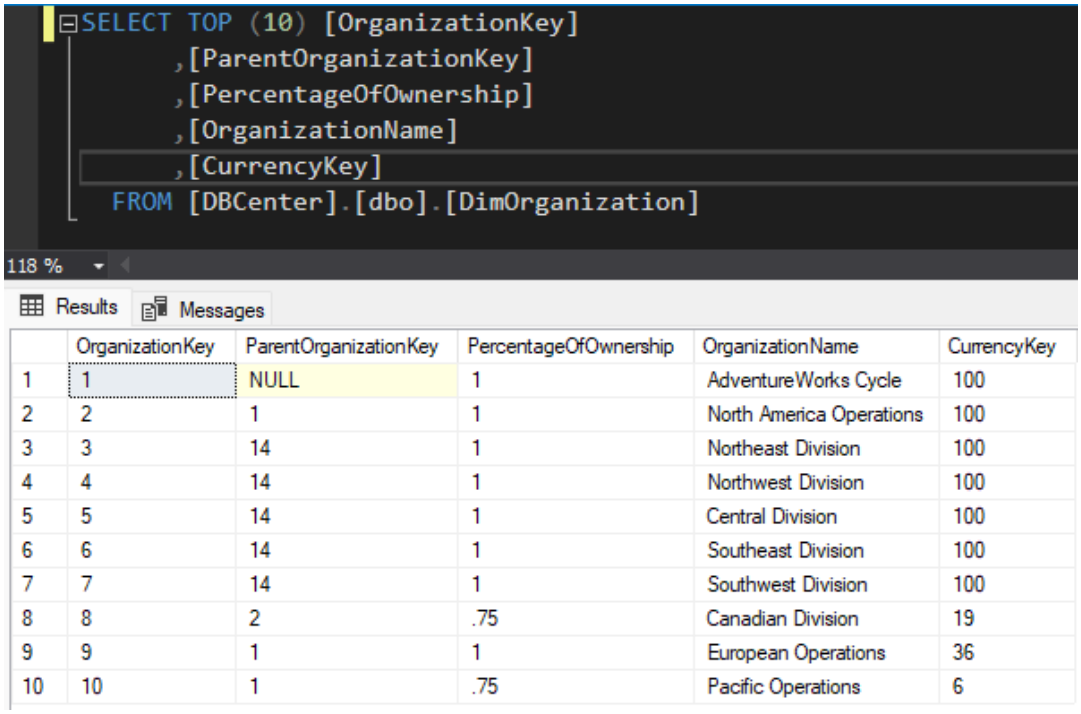
Source: Own processing.

By choosing "Device" under the Restore Database task and then pressing the "..." button choose the.bak file to restore the downloaded sample database.

When the restoration gets completed, simple inquiries are performed to check whether the data responds by writing the query: "SELECT * FROM DimOrganization." If the table has many rows, we can easily use the "TOP" clause to understand a bit of data response: "SELECT TOP 10 * FROM DimOrganization". Firstly, we check the ERD (if possible) to see the relationship among tables. After that, we can use "Join" types to populate the data between the Dims and Facts. Facts are the numeric data, and that could be used for statistical

analysis. Exemplary Picture 6 depicts the selected table and the types of data it consists of. Basically, it gives an idea of what data the author works with.

Picture 4: SQL Query



```
SELECT TOP (10) [OrganizationKey]
      ,[ParentOrganizationKey]
      ,[PercentageOfOwnership]
      ,[OrganizationName]
      ,[CurrencyKey]
  FROM [DBCenter].[dbo].[DimOrganization]
```

118 %

Results | Messages

	OrganizationKey	ParentOrganizationKey	PercentageOfOwnership	OrganizationName	CurrencyKey
1	1	NULL	1	AdventureWorks Cycle	100
2	2	1	1	North America Operations	100
3	3	14	1	Northeast Division	100
4	4	14	1	Northwest Division	100
5	5	14	1	Central Division	100
6	6	14	1	Southeast Division	100
7	7	14	1	Southwest Division	100
8	8	2	.75	Canadian Division	19
9	9	1	1	European Operations	36
10	10	1	.75	Pacific Operations	6

Source: Own processing.

Assuming that we have checked the data within the Dim table and understand the relationship of such a table, However, Picture 6 depicts the "NULL" value, meaning that the data has a missing value. Shortly, the data is not representable and reliable. In this case, the author turns to the ELT process instead of ETL.

The author extracted, loaded, and transformed the data. One of the main reasons is to check the unified data for its quality, consistency, and compatibility. Another key feature is to test the data's adaptability and see if, in that way, the data is functional in the real-world example, to conclude how the data would function and visualize it.

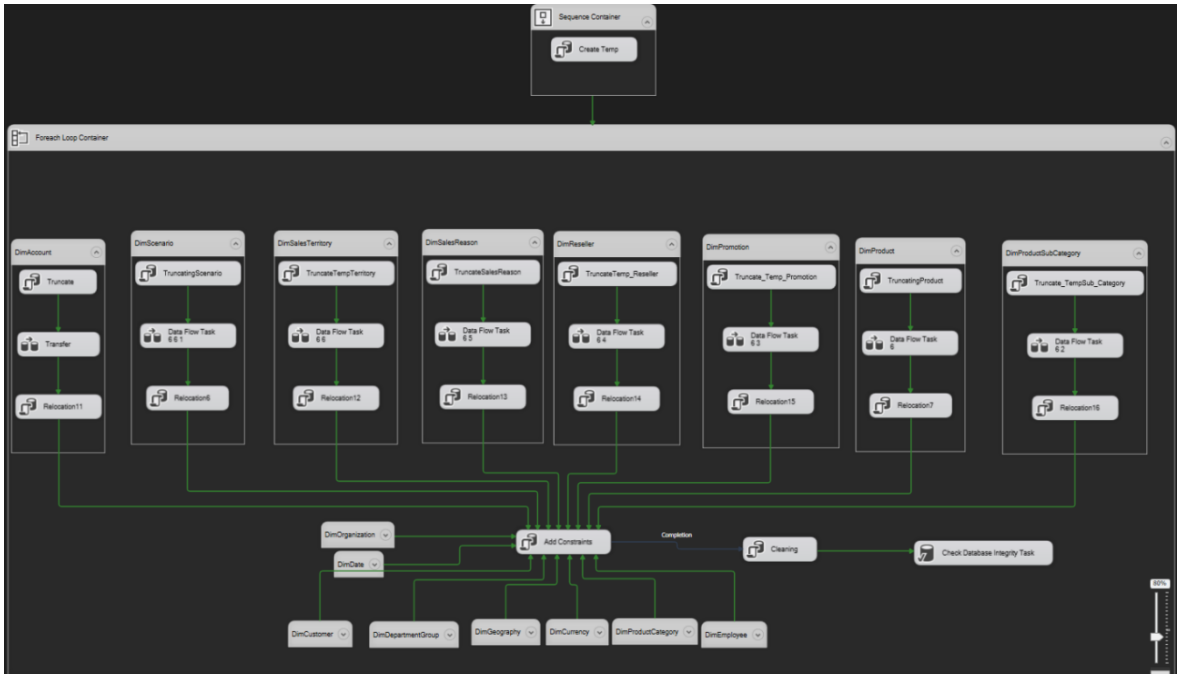
Preparation steps before the ELT process (In SSMS)

- Before we dive into the ELT process, I have started setting up custom databases in **SSMS**. These custom databases, called **"AW_Germany", "AW_France", AW_Italy,"** and **DBCenter,"** serve as the foundation for our data processing and analysis. In our design, we have divided the fact tables into fact tables and dimension tables. The division allows us to effectively manage and organize our data based on its characteristics and relationships.
- To establish a comprehensive view of our data, we have created views within the "DBCenter" database. These views serve as a way to populate the dimension tables with the corresponding fact tables from the 3 databases **"AW_Germany", "AW_France" and "AW_Italy"**. In other words, the views act as mining queries, bringing together the necessary data for analysis and reporting purposes.
- By populating the dimensions with the appropriate data from the fact tables through these views, we ensure that the reporting processes have access to a comprehensive set of data that includes both dimensions and corresponding facts.
- To facilitate reporting and provide a more user-friendly interface, we have created stored procedures within each database. These stored procedures leverage the views we created earlier, incorporating the aggregation of data and applying transformations specific to our reporting requirements after loading the data. By encapsulating the logic within the stored procedures, we have streamlined the data retrieval and manipulation process, making it more convenient for generating reports and visualizing data in tools like SSRS. Although some procedures are used specifically for SSIS package tasks, we will describe them in a further demonstration of our ELT process.
- By using this approach, we have created a structured data environment that supports data analysis, reporting, and visualization. The subsequent execution of the SSIS package, as outlined in the previous explanation, builds upon the foundation to further process the data, ensuring data integrity, cleansing, and transformation before finalizing the analysis and reporting tasks.
- In summary, some of the steps before the SSIS package execution involve setting up custom databases, creating views to integrate dimensions with facts (populating after

the ELT), using aggregations and manipulations in stored procedures, procedures getting the data from views, and views populated from different databases.

The author provides visual examples on Picture 7, with procedures that are used in the SSIS Package and in the reporting part of the work.

Picture 5: ELT Process in Visual Studio.



1. DBCenter SSIS Package

We have several connection managers to run the queries/procedures to get the source database files and loading into destinations. We have chosen SSIS package design in such way to achieve more control and efficiency. Each table group has 3 tasks inside and those tasks are differentiated by the procedures from SSMS and the connection managers.

4.1.1 Structure of the SSIS package

1- SQL Task (Create Temp Tables proc): At the beginning of the SSIS package, it start with a SQL task that creates temporary tables using the procedure named “CREATE_TEMP_TABLES” this procedures aim to create temporary tables that match the each dimensional table and its table design, note that author may manipulate the data types in order to achieve data match with reporting application to present the data efficiently. Below we can see one of the example queries for temporary table creation

proc., since we have 16-dimensional tables and it exceeds the visible screen space, we will show the main examples.

Picture 6: SQL Tasks

```
ALTER PROC [dbo].[CREATE_TEMP_TABLES]
 AS
IF NOT EXISTS (SELECT * FROM SYS.TABLES WHERE NAME=N'TEMP_DimAccount' AND type ='U')
BEGIN
 CREATE TABLE dbo.TEMP_DimAccount(
  [AccountKey] [int] NOT NULL,
     [ParentAccountKey] [int] NULL,
     [AccountCodeAlternateKey] [int] NULL,
     [ParentAccountCodeAlternateKey] [int] NULL,
     [AccountDescription] [nvarchar](50) NULL,
     [AccountType] [nvarchar](50) NULL,
     [Operator] [nvarchar](50) NULL,
     [CustomMembers] [nvarchar](300) NULL,
     [ValueType] [nvarchar](50) NULL,
     [CustomMemberOptions] [nvarchar](200) NULL,

 )

     END

IF Not EXISTS ( SELECT * FROM sys.tables WHERE NAME=N'TEMP_DimCurrency' AND type='U')
BEGIN
CREATE TABLE dbo.TEMP_DimCurrency(

     [CurrencyKey] [int]  NOT NULL,
     [CurrencyAlternateKey] [nchar](3) NOT NULL,
     [CurrencyName] [nvarchar](50) NOT NULL,


 )
 END
```

Source: Own processing.

Above picture gives us the information of used database **"DBCenter"** and the table design of **"DimAccount"** and the other 15 tables. Note that we are creating the tables IF it is not existing in the **"DBCenter"** to prevent creation of duplicate tables and data corruption.

2- The second task is to execute a Truncation of each . Inside of Each Loop container, we truncate the temps to ensure they are empty and ready to receive the data. Inside of our task, instead of using procedure (it might take unnecessary effort) the author simply runs the query **"TRUNCATE TABLE TEMP_TableName"** (We can use SQL Instances thanks to our connection with SQL Server under the connection manager tab in Visual Studio).

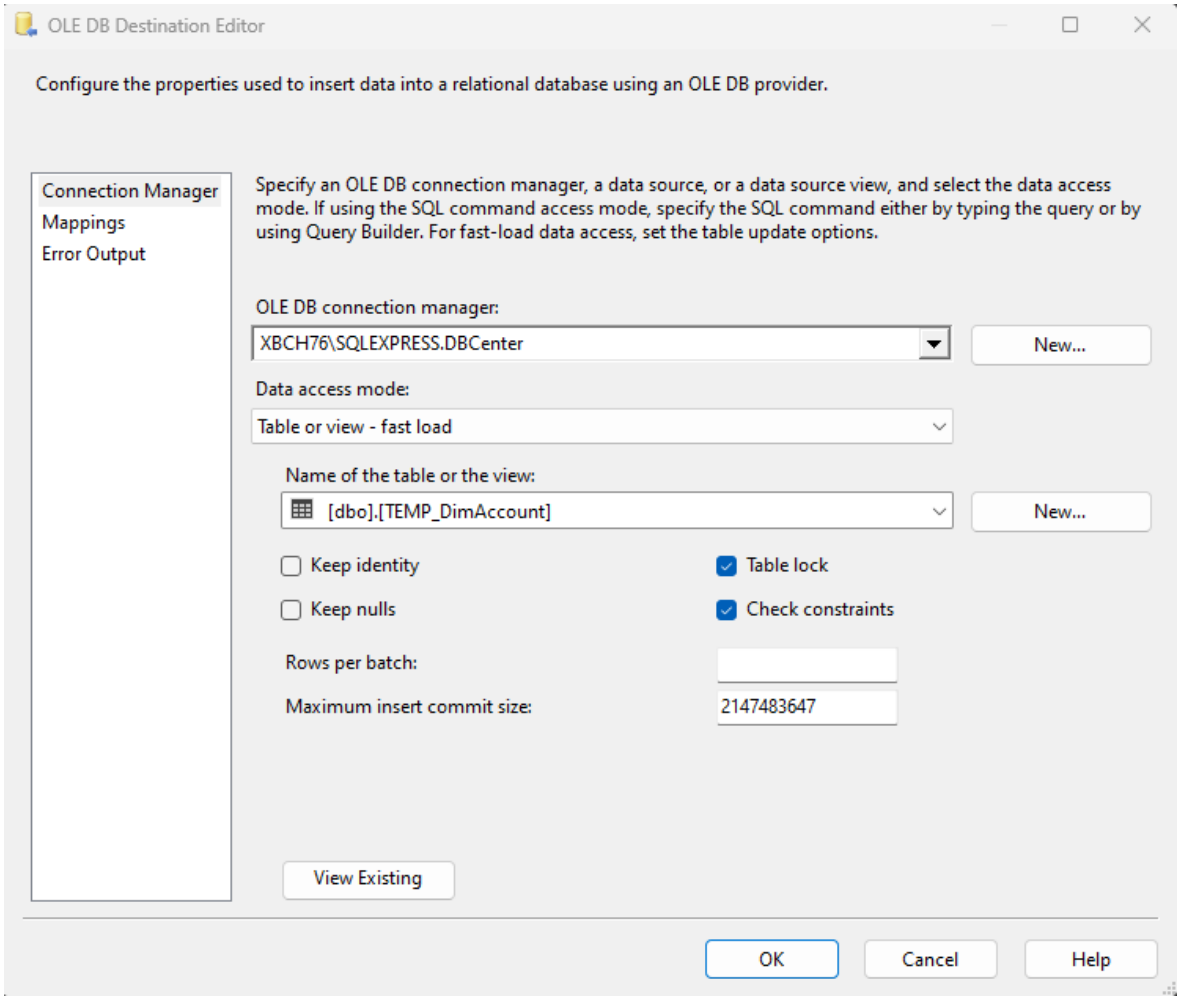
3- Data Flow Task. In this step we transfer the source database files to the temporary tables in the destination table. In the below Picture – 9 inside of our data flow task we have

OLE DB Source and destination, configuring the column mapping and correct DB connections with each task.

Picture 7: Data Flow Task

Source: Own processing.

Picture 8: OLE DB editor

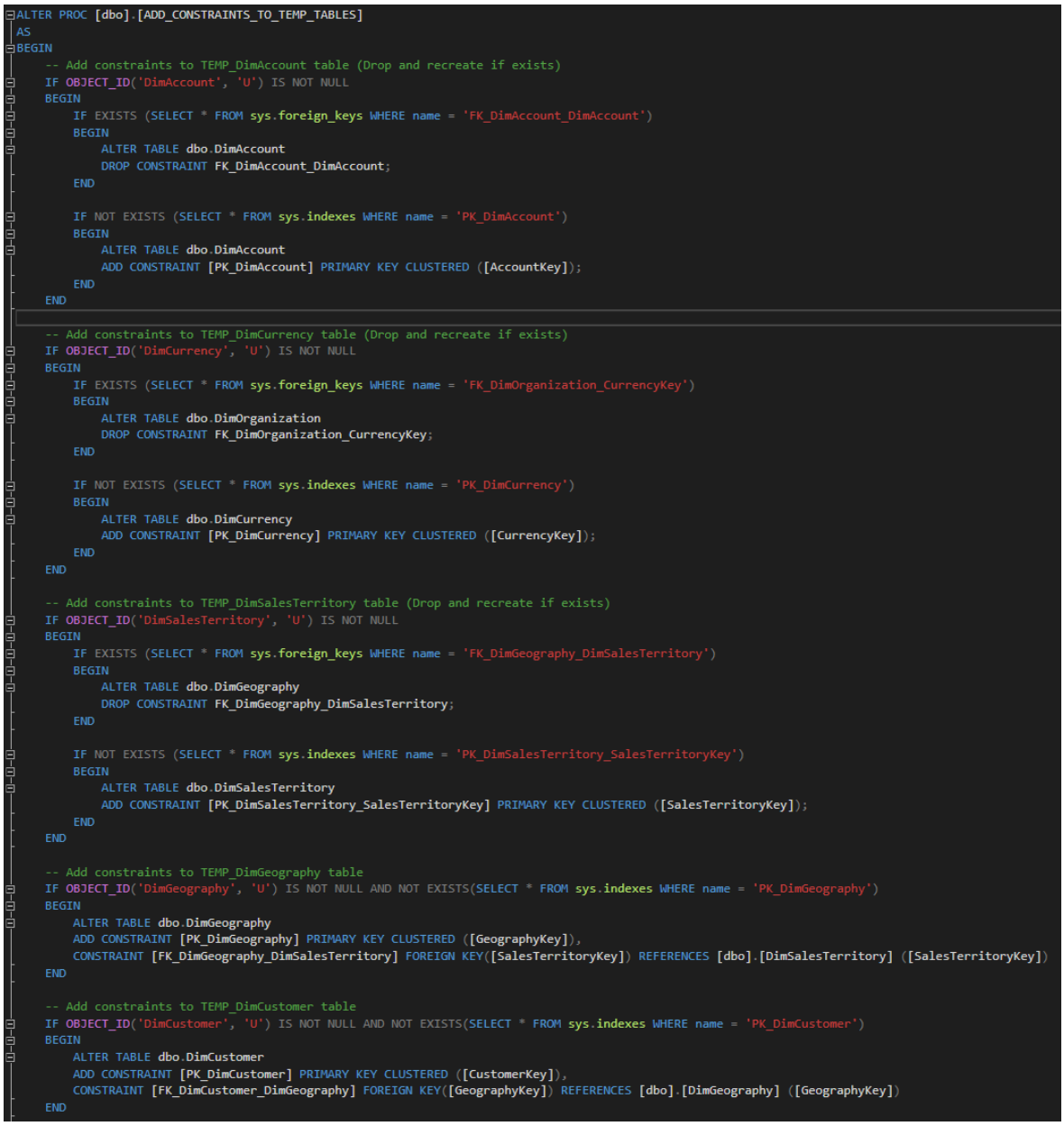


Source: Own processing.

4- Renaming the temporary table's as original source table name's; This step allows us to effectively replace the original tables with the updated information from the temps, by renaming the tables, we ensure a seamless integration of the modified data into the database.

5- Adding constraints to renamed tables; The task is to add constraints to the renamed tables, the aim of this task is to ensure the relationships between dimension tables, such as primary keys, foreign keys, and alternate keys. We ensure the data consistency and accuracy.

Picture 9: Constraints



```
ALTER PROC [dbo].[ADD_CONSTRAINTS_TO_TEMP_TABLES]
AS
BEGIN
    -- Add constraints to TEMP_DimAccount table (Drop and recreate if exists)
    IF OBJECT_ID('DimAccount', 'U') IS NOT NULL
    BEGIN
        IF EXISTS (SELECT * FROM sys.foreign_keys WHERE name = 'FK_DimAccount_DimAccount')
        BEGIN
            ALTER TABLE dbo.DimAccount
            DROP CONSTRAINT FK_DimAccount_DimAccount;
        END

        IF NOT EXISTS (SELECT * FROM sys.indexes WHERE name = 'PK_DimAccount')
        BEGIN
            ALTER TABLE dbo.DimAccount
            ADD CONSTRAINT [PK_DimAccount] PRIMARY KEY CLUSTERED ([AccountKey]);
        END
    END

    -- Add constraints to TEMP_DimCurrency table (Drop and recreate if exists)
    IF OBJECT_ID('DimCurrency', 'U') IS NOT NULL
    BEGIN
        IF EXISTS (SELECT * FROM sys.foreign_keys WHERE name = 'FK_DimOrganization_CurrencyKey')
        BEGIN
            ALTER TABLE dbo.DimOrganization
            DROP CONSTRAINT FK_DimOrganization_CurrencyKey;
        END

        IF NOT EXISTS (SELECT * FROM sys.indexes WHERE name = 'PK_DimCurrency')
        BEGIN
            ALTER TABLE dbo.DimCurrency
            ADD CONSTRAINT [PK_DimCurrency] PRIMARY KEY CLUSTERED ([CurrencyKey]);
        END
    END

    -- Add constraints to TEMP_DimSalesTerritory table (Drop and recreate if exists)
    IF OBJECT_ID('DimSalesTerritory', 'U') IS NOT NULL
    BEGIN
        IF EXISTS (SELECT * FROM sys.foreign_keys WHERE name = 'FK_DimGeography_DimSalesTerritory')
        BEGIN
            ALTER TABLE dbo.DimGeography
            DROP CONSTRAINT FK_DimGeography_DimSalesTerritory;
        END

        IF NOT EXISTS (SELECT * FROM sys.indexes WHERE name = 'PK_DimSalesTerritory_SalesTerritoryKey')
        BEGIN
            ALTER TABLE dbo.DimSalesTerritory
            ADD CONSTRAINT [PK_DimSalesTerritory_SalesTerritoryKey] PRIMARY KEY CLUSTERED ([SalesTerritoryKey]);
        END
    END

    -- Add constraints to TEMP_DimGeography table
    IF OBJECT_ID('DimGeography', 'U') IS NOT NULL AND NOT EXISTS(SELECT * FROM sys.indexes WHERE name = 'PK_DimGeography')
    BEGIN
        ALTER TABLE dbo.DimGeography
        ADD CONSTRAINT [PK_DimGeography] PRIMARY KEY CLUSTERED ([GeographyKey]),
        CONSTRAINT [FK_DimGeography_DimSalesTerritory] FOREIGN KEY([SalesTerritoryKey]) REFERENCES [dbo].[DimSalesTerritory] ([SalesTerritoryKey])
    END

    -- Add constraints to TEMP_DimCustomer table
    IF OBJECT_ID('DimCustomer', 'U') IS NOT NULL AND NOT EXISTS(SELECT * FROM sys.indexes WHERE name = 'PK_DimCustomer')
    BEGIN
        ALTER TABLE dbo.DimCustomer
        ADD CONSTRAINT [PK_DimCustomer] PRIMARY KEY CLUSTERED ([CustomerKey]),
        CONSTRAINT [FK_DimCustomer_DimGeography] FOREIGN KEY([GeographyKey]) REFERENCES [dbo].[DimGeography] ([GeographyKey])
    END
```

Source: Own processing.

The above example of our constraint's procedure starts with checking if each table exists using OBJECT_ID function and then adds the specified constraints using the ALTER TABLE Statement.

6- Cleaning Null values: Within the for each loop container, a procedure is executed to change the Null values for more representable way for the end-users in the reporting part. One of the examples of cleaning null value procedure.

Picture 10: Nulls corrections

```
USE [DBCenter]
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO




ALTER PROCEDURE [dbo].[UpdateDimAccount]
AS
BEGIN
    SET NOCOUNT ON;

    -- Temporarily disable the foreign key constraint
    ALTER TABLE DimProduct
    NOCHECK CONSTRAINT FK_DimProduct_ProductSubcategoryKey;

    -- Update DimAccount without modifying the ProductSubcategoryKey
    UPDATE DimAccount
    SET
        AccountKey = ISNULL(AccountKey, 0),
        ParentAccountKey = ISNULL(ParentAccountKey, 0),
        ParentAccountCodeAlternateKey = ISNULL(ParentAccountCodeAlternateKey, '0'),
        AccountType = ISNULL(AccountType, 'DefaultType'),
        CustomMemberOptions = ISNULL(CustomMemberOptions, 'Unknown'),
        CustomMembers = ISNULL(CustomMembers, 'Unknown')
    WHERE
        AccountKey IS NULL
        OR ParentAccountCodeAlternateKey IS NULL
        OR ParentAccountKey IS NULL
        OR AccountType IS NULL
        OR CustomMemberOptions IS NULL
        OR CustomMembers IS NULL;

    -- Re-enable the foreign key constraint
    ALTER TABLE DimProduct
    CHECK CONSTRAINT FK_DimProduct_ProductSubcategoryKey;
END;
GO
```

Source: Own processing.

In above provided procedure demonstrates a practical example of temporarily disabling and re-enabling a foreign key (Or primary key in another table) while performing updating/cleansing on the chosen table. This procedure structure used for other 9-dimension tables that needs to be cleaned.

7- Database integrity check: Finally, after the cleaning, a database integrity check is performed. The purpose is to verify that the data is correctly structured and complies

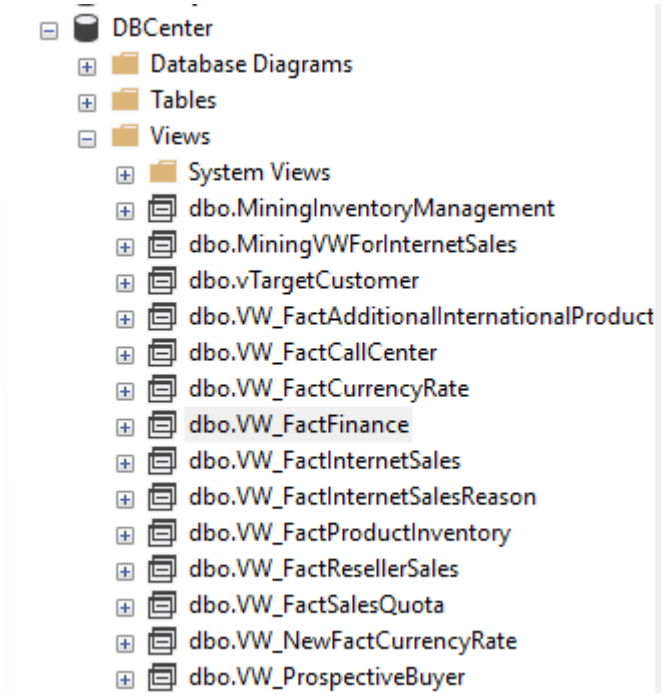
with the defined integrity constraints. This step ensures the reliability and consistency of the database, identifying any potential issues or anomalies that need to be addressed.

We are following the same structure for the fact tables, however, different connection managers and slightly similar queries used in fact table transfer.

4.1.2 Mining Queries (Views) in SSMS

Firstly, we are creating the views to populate the data from different databases and then we are creating a procedure that uses the created view. Since Procedures can be used in Report builder and can carry complex queries/sub-queries its fast and reliable solution to use for Report datasets instead of just using queries or tables in Report designs.
Let's see our Views for the Report datasets.

Picture 13: DBCenter



Source: Own processing.

We will give example of Inventory Management view in our Database. In above Picture – 13, we can see our views for the fact tables. Let's give it a look inside of "MiningInventoryManagement".

Picture 11: Minning Inventory Management

```
ALTER VIEW [dbo].[MiningInventoryManagement]
AS

SELECT
    p.ProductKey,
    s.CustomerKey as CustomerID,
    s.OrderDateKey as OrderID,
    CONVERT(DATE, s.OrderDate) AS OrderDate,
    YEAR(s.OrderDate) as OrderYear,
    p.EnglishProductName,
    pc.EnglishProductCategoryName,
    psc.EnglishProductSubcategoryName,
    SUM(s.OrderQuantity) AS TotalQuantity,
    SUM(s.SalesAmount) AS TotalSalesAmount,
    SUM(s.SalesAmount) / SUM(s.OrderQuantity) AS AveragePrice,
    p.StandardCost,
    p.StandardCost * SUM(s.OrderQuantity) AS TotalCost,
    SUM(s.SalesAmount) - (p.StandardCost * SUM(s.OrderQuantity)) AS TotalProfit
FROM (

    SELECT *, 'France' as Country  FROM AW_France.dbo.FactInternetSales
    UNION ALL

    SELECT *, 'Germany' as Country FROM AW_Germany.dbo.FactInternetSales
    UNION ALL

    SELECT *, 'Italy' as Country  FROM AW_Italy.dbo.FactInternetSales
) AS s
INNER JOIN dbo.DimProduct p
    ON s.ProductKey = p.ProductKey
INNER JOIN dbo.DimProductSubcategory psc
    ON p.ProductSubcategoryKey = psc.ProductSubcategoryKey
INNER JOIN dbo.DimProductCategory pc
    ON psc.ProductCategoryKey = pc.ProductCategoryKey
GROUP BY
    p.ProductKey,
    s.CustomerKey,
    p.EnglishProductName,
    pc.EnglishProductCategoryName,
    s.OrderDateKey,
    psc.EnglishProductSubcategoryName,
    p.StandardCost,
    CONVERT(DATE, s.OrderDate),
    YEAR(s.OrderDate);

GO
```

Source: Own processing.

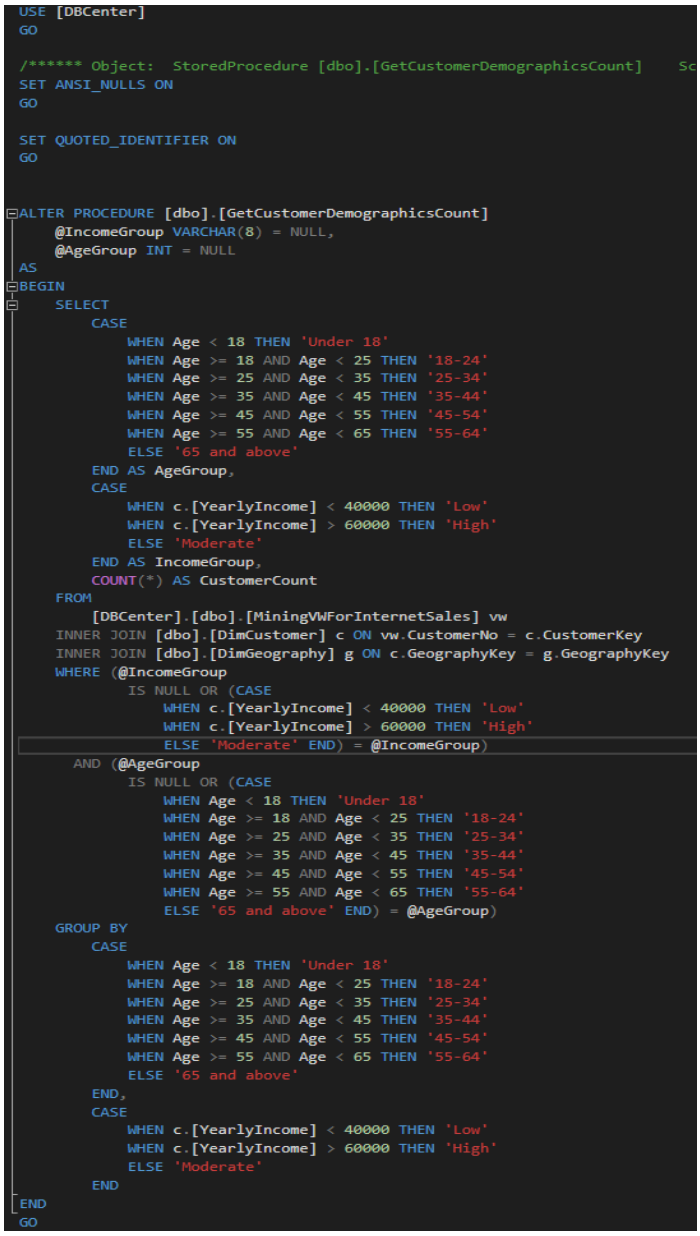
Picture 14 depicts the mining view in our DBCenter. The query combines data from multiple tables to retrieve information about product sales, category, and other related attributes. The resulting dataset is used for reporting purposes. The join clause is to populate the fact table with the corresponding dimensional tables.

After the execution of Mining Inventory Management, we are aiming to create procedures to use the view itself as a data source to create more complex mining queries to achieve different report types.

Procedures used for:

- Code encapsulation: Stored procedures provide database administrators with the ability to encapsulate complex SQL logic, making it easier to manage and maintain. The SQL logic can be written once and reused multiple times in applications if any modifications are needed afterwards.
- Improved performance: stored procedures are precompiled and can provide performance benefits by reducing network traffic and query parsing overhead. Executing the procedure requires fewer round-trips between the database and applications.
- Data Manipulation and transformation Before returning the results to the application This allows us to perform various calculations, aggregations, and data formatting; other data transformations that happened FROM the source dataset (View, table) can also be aggregated to make preferred decisions for reporting purposes.
- In summary, stored procedures are very powerful mechanisms and are usually used in enterprise applications to handle complex business logic, improve execution performance, and ensure data integrity, as we did in our SSIS package.

Picture 12: Customer Demographics Count Procedure.



```
USE [DBCenter]
GO

/****** Object:  StoredProcedure [dbo].[GetCustomerDemographicsCount]    Sc
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO


ALTER PROCEDURE [dbo].[GetCustomerDemographicsCount]
    @IncomeGroup VARCHAR(8) = NULL,
    @AgeGroup INT = NULL
AS
BEGIN
    SELECT
        CASE
            WHEN Age < 18 THEN 'Under 18'
            WHEN Age >= 18 AND Age < 25 THEN '18-24'
            WHEN Age >= 25 AND Age < 35 THEN '25-34'
            WHEN Age >= 35 AND Age < 45 THEN '35-44'
            WHEN Age >= 45 AND Age < 55 THEN '45-54'
            WHEN Age >= 55 AND Age < 65 THEN '55-64'
            ELSE '65 and above'
        END AS AgeGroup,
        CASE
            WHEN c.[YearlyIncome] < 40000 THEN 'Low'
            WHEN c.[YearlyIncome] > 60000 THEN 'High'
            ELSE 'Moderate'
        END AS IncomeGroup,
        COUNT(*) AS CustomerCount
    FROM
        [DBCenter].[dbo].[MiningVWForInternetSales] vw
    INNER JOIN [dbo].[DimCustomer] c ON vw.CustomerNo = c.CustomerKey
    INNER JOIN [dbo].[DimGeography] g ON c.GeographyKey = g.GeographyKey
    WHERE (@IncomeGroup
            IS NULL OR (CASE
                WHEN c.[YearlyIncome] < 40000 THEN 'Low'
                WHEN c.[YearlyIncome] > 60000 THEN 'High'
                ELSE 'Moderate' END) = @IncomeGroup)
      AND (@AgeGroup
            IS NULL OR (CASE
                WHEN Age < 18 THEN 'Under 18'
                WHEN Age >= 18 AND Age < 25 THEN '18-24'
                WHEN Age >= 25 AND Age < 35 THEN '25-34'
                WHEN Age >= 35 AND Age < 45 THEN '35-44'
                WHEN Age >= 45 AND Age < 55 THEN '45-54'
                WHEN Age >= 55 AND Age < 65 THEN '55-64'
                ELSE '65 and above' END) = @AgeGroup)
    GROUP BY
        CASE
            WHEN Age < 18 THEN 'Under 18'
            WHEN Age >= 18 AND Age < 25 THEN '18-24'
            WHEN Age >= 25 AND Age < 35 THEN '25-34'
            WHEN Age >= 35 AND Age < 45 THEN '35-44'
            WHEN Age >= 45 AND Age < 55 THEN '45-54'
            WHEN Age >= 55 AND Age < 65 THEN '55-64'
            ELSE '65 and above'
        END,
        CASE
            WHEN c.[YearlyIncome] < 40000 THEN 'Low'
            WHEN c.[YearlyIncome] > 60000 THEN 'High'
            ELSE 'Moderate'
        END
END
GO
```

Source: Own processing.

Description of Picture – 15:

- The stored procedure retrieves customer demographic information and counts the number of customers based on specified income groups and age groups. Likely represents a data warehouse view specifically designed for mining internet sales data. The procedure uses the "**DimCustomer**" and "**DimGeography**" tables to join customer and geographic information. Also

notice that this procedure receives the necessary data from the "MiningVWForInternetSales" view.

- The **CASE** statements are used to categorize the customers into specific age groups and income groups based on their demographic data. Notice that in the View MiningVWForInternetSales," Age has been calculated and given as **ALIAS (as).** To use this alias in our procedure as a column, we used a sub-query for the parameter to recognize the alias. For Age grouping, customers are categorized into various age ranges, as seen in Figure 5.
- The **FROM** clause specifies the tables used in the query. The "MiningVWForInternetSales" view is joined with the "**DimCustomer**" table on the **CustomerNo** and **CustomerKey** Additionally, the "**DimGeography**" table joins the **GeographyKey** column.
- The **WHERE** Clause filters the data based on the provided input parameters. If the @**IncomeGroup** parameter is not null, it compares the income group of the customer with the input value. Similarly, if the @**AgeGroup** parameter is not null, it compares the **age group** of the customer with the input value.
- The **GROUP BY** clause groups the result set by age group and income group.

By using the optional input parameters, the stored procedure allows for customization of the query results based on specific age or income groups. This stored procedure is utilized for reporting purposes. The reason we are making the parameters optional is to allow the end-user to select "All" in our report.

Picture 13: Different diagrams with the customer's age category

Source: Own processing.

- In Picture 16, users are able to see the visual result of total customers by their Age Group and Income group. Also, on the main bar chart, they are able to select a column and see more details about customers. For example, if we click a column from the **"Low"** Income group and choose the age group **"45–54,"** we are directed to a sub-report that shows more details about customers in that group (see Picture 16).
- **The first chart** provides a visual representation of the total number of customers grouped by income group. The income groups are displayed on the x-axis, while the y-axis represents the sum of customer counts. Within each income group, the chart further breaks down the data by age group using different series (e.g., different colors or patterns). This chart allows users to compare the customer distribution across income groups and observe how different age groups contribute to each income group. It helps

users identify trends and patterns in customer demographics based on both income and age.

- **The second chart** focuses solely on the distribution of customers based on income groups. It provides a visual representation of the total customer number in each income group without further breaking down the data by age group and allows users to compare the customer counts across income groups directly, providing a high-level overview of the customer distribution based on income.
- **The third chart (Age Group)** focuses on the distribution of customers based on age groups. This chart allows us to compare the customer counts across age groups directly, providing insights into the age composition of the customer base.

Potential Uses of the Report: Market Segmentation By analyzing the customer distribution across income and age groups, business users can identify distinct market segments within our customers, such as product development, customer retention strategies, and marketing campaigns.

Let’s show another procedure that uses the "MiningVWFactInternetSales" view. This example is similar to the above procedure example. The difference between the above example (See Picture 21) and the example we are about to describe is that the main report gives quick information about our data, while sub-reports are more comprehensive.

Picture 14: Customer Demographics

```
USE [DBCenter]
GO

/****** Object:  StoredProcedure [dbo].[GetCustomerDemographics]    Script Date: 18.07.2023 00:07:3
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO


ALTER PROCEDURE [dbo].[GetCustomerDemographics]
    @IncomeGroup VARCHAR(8) = NULL,
    @AgeGroup VARCHAR(MAX) = NULL
AS
BEGIN
   SELECT DISTINCT
    c.CustomerKey as CustomerNo,
    CONCAT(c.FirstName, ' ', c.LastName) AS CustomerName,
    c.EmailAddress,
    CASE
        WHEN Month(GetDate()) < Month(c.[BirthDate]) THEN DateDiff(yy,c.[BirthDate],GetDate()) - 1
        WHEN Month(GetDate()) = Month(c.[BirthDate]) AND Day(GetDate()) < Day(c.[BirthDate])
        THEN DateDiff(yy,c.[BirthDate],GetDate()) - 1
        ELSE DateDiff(yy,c.[BirthDate],GetDate())
    END AS [Age],
    CASE
        WHEN c.[YearlyIncome] < 40000 THEN 'Low'
        WHEN c.[YearlyIncome] > 60000 THEN 'High'
        ELSE 'Moderate'
    END AS [IncomeGroup],
    g.StateProvinceName as State,
    g.City,
    g.EnglishCountryRegionName as Country,
    c.Phone
FROM
    [DBCenter].[dbo].[MiningVWForInternetSales] vw
INNER JOIN [dbo].[DimCustomer] c ON vw.CustomerNo = c.CustomerKey
INNER JOIN [dbo].[DimGeography] g ON c.GeographyKey = g.GeographyKey
WHERE (@IncomeGroup IS NULL OR IncomeGroup = @IncomeGroup)
AND (@AgeGroup
    IS NULL OR (CASE
        WHEN Age < 18 THEN 'Under 18'
        WHEN Age >= 18 AND Age < 25 THEN '18-24'
        WHEN Age >= 25 AND Age < 35 THEN '25-34'
        WHEN Age >= 35 AND Age < 45 THEN '35-44'
        WHEN Age >= 45 AND Age < 55 THEN '45-54'
        WHEN Age >= 55 AND Age < 65 THEN '55-64'
        ELSE '65 and above' END) = @AgeGroup)
END;
GO
```

Source: Own processing.

The parameters are used as in Figure 17, so that there will be no mismatch of values and preventing errors between the reports. Also note that we are hiding parameters in the sub-report to show related information about the main report; we may add selectable parameters inside the sub-report to filter the preferred column. See picture **18** for the precise illustration.

Picture 15: Customer Information Sub-Report

Customer No	Customer Name	Income Group	Email Address	Phone	Country	State	Age
11335	Carla Raman	Low	carla14@adventure-works.com	1 (11) 500 555-0124	Germany	Bayern	45
11336	Shaun Raji	Low	shaun21@adventure-works.com	1 (11) 500 555-0122	France	Seine (Paris)	50
11338	Frank Navarro	Low	frank17@adventure-works.com	1 (11) 500 555-0139	Germany	Nordrhein-Westfalen	50
11342	Marshall Wang	Low	marshall0@adventure-works.com	1 (11) 500 555-0161	France	Pas de Calais	50
11343	Arthur Carlson	Low	arthur41@adventure-works.com	1 (11) 500 555-0166	United Kingdom	England	48
11347	Roy Navarro	Low	roy30@adventure-works.com	1 (11) 500 555-0137	Germany	Nordrhein-Westfalen	51
11379	Gary Vazquez	Low	gary25@adventure-works.com	1 (11) 500 555-0130	France	Hauts de Seine	47
11381	Meredith Raman	Low	meredith11@adventure-works.com	1 (11) 500 555-0162	United Kingdom	England	52
11382	Edward Patterson	Low	edward59@adventure-works.com	1 (11) 500 555-0118	France	Nord	49
11384	Tiffany Wang	Low	tiffany1@adventure-works.com	1 (11) 500 555-0192	France	Essonne	51
11385	Miguel Allen	Low	miguel24@adventure-works.com	1 (11) 500 555-0128	United Kingdom	England	51
11386	Jaclyn Cai	Low	jaclyn22@adventure-works.com	1 (11) 500 555-0164	Germany	Hessen	51
11390	Christine Raji	Low	christine16@adventure-works.com	1 (11) 500 555-0153	France	Yveline	47
11391	Lindsay Xie	Low	lindsay3@adventure-works.com	1 (11) 500 555-0115	Germany	Hessen	47
11393	Kurt Pal	Low	kurt12@adventure-works.com	1 (11) 500 555-0117	France	Loir et Cher	49
11394	George McDonald	Low	george13@adventure-works.com	1 (11) 500 555-0150	United Kingdom	England	48
11395	Beth Gutierrez	Low	beth13@adventure-works.com	1 (11) 500 555-0110	Germany	Bayern	47
11396	Ian Lopez	Low	ian27@adventure-works.com	1 (11) 500 555-0182	Germany	Saarland	48
11397	Latoya Shan	Low	latoya9@adventure-works.com	1 (11) 500 555-0122	France	Seine (Paris)	48
11398	Colin Nath	Low	colin41@adventure-works.com	1 (11) 500 555-0115	United Kingdom	England	48
11399	Brenda Mehta	Low	brenda17@adventure-works.com	1 (11) 500 555-0155	France	Seine (Paris)	48
11410	Maurice Goel	Low	maurice20@adventure-works.com	1 (11) 500 555-0179	France	Seine (Paris)	49
11482	Adrienne Torres	Low	adrienne8@adventure-works.com	1 (11) 500 555-0120	United Kingdom	England	45
11487	Morgan Jones	Low	morgan26@adventure-works.com	1 (11) 500 555-0155	France	Nord	46
11488	Jermaine Lopez	Low	jermaine15@adventure-works.com	1 (11) 500 555-0181	United Kingdom	England	46
11489	Deborah Goel	Low	deborah21@adventure-works.com	1 (11) 500 555-0168	United Kingdom	England	46

Source: Own processing.

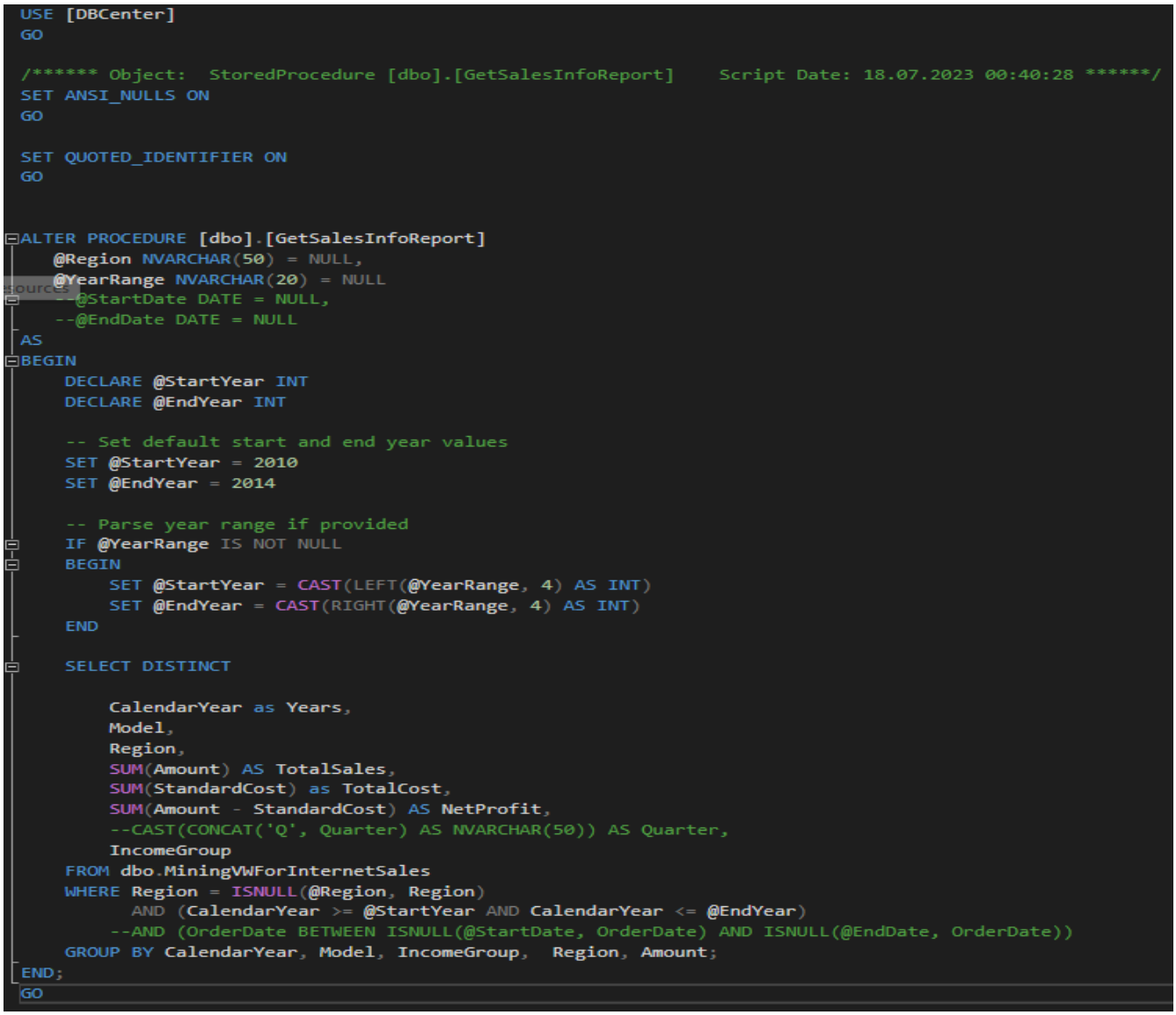
Picture 18 shows us the detailed customer demographics that were selected in the main report. The user selected the age group "45–54" under the Income Group "Low" column.

Let’s take a look at another procedure that is used for reporting purposes. The below picture 19 shows us another SQL Query logic for reporting purposes. Let’s discuss the purpose of the query and the logic inside of it.

In our procedure named **GetSalesInfoReport,"** we retrieve sales data from the MiningVWForInternetSales table, which likely contains a pre-processed and optimized view of the internet sales data. This ensures that the query operates on a subset of data relevant to internet sales, improving performance and reducing complexity. The WHERE Clause

includes a condition to filter the sales data by the specified @Region parameter. If @Region is not provided or is NULL, the query will return sales data for all regions; however, if @Region is specified, the query will include the internet sales data for that particular region. This allows users to focus on sales information as they like.

Picture 16: Sales and report steps



```
USE [DBCenter]
GO

/****** Object:  StoredProcedure [dbo].[GetSalesInfoReport]    Script Date: 18.07.2023 00:40:28 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO


ALTER PROCEDURE [dbo].[GetSalesInfoReport]
    @Region NVARCHAR(50) = NULL,
    @YearRange NVARCHAR(20) = NULL
    --@StartDate DATE = NULL,
    --@EndDate DATE = NULL
AS
BEGIN
    DECLARE @StartYear INT
    DECLARE @EndYear INT

    -- Set default start and end year values
    SET @StartYear = 2010
    SET @EndYear = 2014

    -- Parse year range if provided
    IF @YearRange IS NOT NULL
    BEGIN
        SET @StartYear = CAST(LEFT(@YearRange, 4) AS INT)
        SET @EndYear = CAST(RIGHT(@YearRange, 4) AS INT)
    END

    SELECT DISTINCT

        CalendarYear as Years,
        Model,
        Region,
        SUM(Amount) AS TotalSales,
        SUM(StandardCost) as TotalCost,
        SUM(Amount - StandardCost) AS NetProfit,
        --CAST(CONCAT('Q', Quarter) AS NVARCHAR(50)) AS Quarter,
        IncomeGroup
    FROM dbo.MiningVWForInternetSales
    WHERE Region = ISNULL(@Region, Region)
          AND (CalendarYear >= @StartYear AND CalendarYear <= @EndYear)
        --AND (OrderDate BETWEEN ISNULL(@StartDate, OrderDate) AND ISNULL(@EndDate, OrderDate))
    GROUP BY CalendarYear, Model, IncomeGroup,  Region, Amount;
END;
GO
```

Source: Own processing.

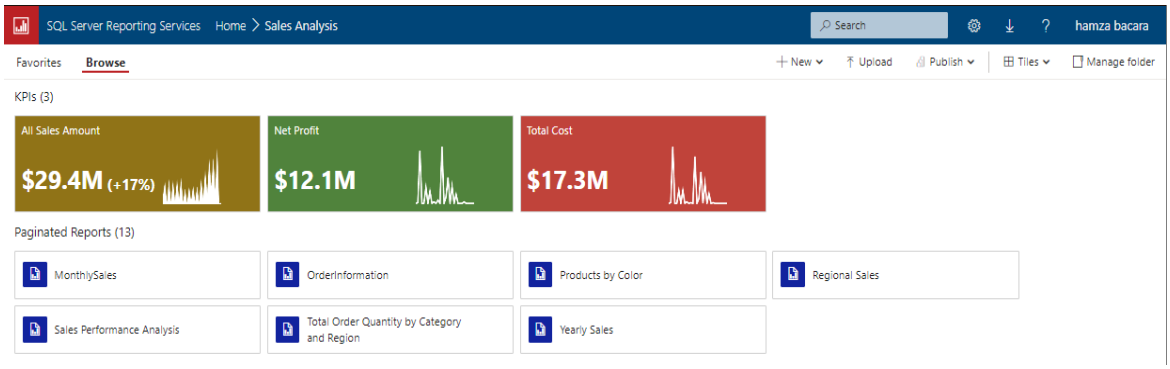
In the provided code snippet, the CAST function is used to convert a substring of the **'@YearRange'** parameter from a string type to an integer type. If a year range is provided, the code uses the **'CAST'** function to extract the first four characters (representing the end year), which are extracted and converted to an integer using

‘CAST(RIGHT(@YearRange,4) as INT). This allows the code to dynamically set the **‘@StartYear’** and **‘@EndYear’** variables based on the provided **‘@YearRange‘.**

Defining the database year range, the stored procedure allows users to define a custom year range for the sales report; by default, the @StartYear and @EndYear parameters are set to 2010 and 2014, which are our database range for the sales. However, if the @YearRange parameter is provided, the logic parses it to extract the start and end years. The query then includes a condition to filter the sales data based on the specified year range. This flexibility enables users to analyze sales trends and performance within their desired timeframe.

Aggregation of sales information: The **GROUP BY** clause groups the sales data by calendar year, Model, income group, region, and Amount. This aggregation allows us to summarize sales information and calculate various metrics. The **SELECT** statement then retrieves the aggregated data, including the **Total Sales**, **total cost, and net profit** for each combination of calendar year**, Model, income group, region, and Amount.** Overall, the query logic aims to generate a sales information report that is customizable based on the specified region and year range. By filtering and aggregating the sales data, the logic enables users to analyze sales performance, identify trends, and make informed business decisions.

Picture 17: Sales Analysis

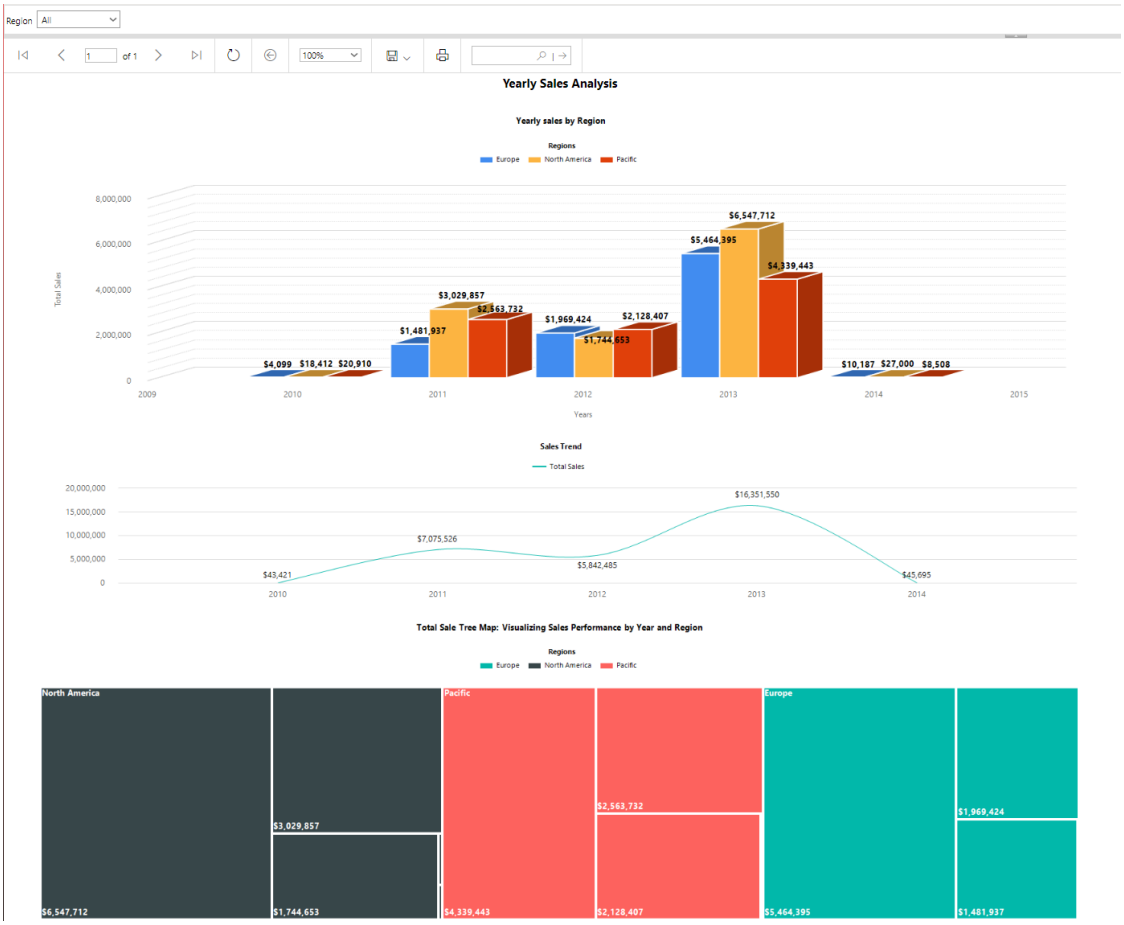


Source: Own processing.

Picture 20 presents most of the paginated reports generated from Picture 20. The Procedure "GetSalesInfo" aims to provide valuable insights. We have three Key Performance indicators (KPI) that users can click on and navigate to related sales paginated reports or visualizations. These paginated reports are informative and offer a comprehensive overview of our sales performance.

4.2 Results of reports

Picture 18: Yearly Sales Analysis



Source: Own processing.

Above picture The Yearly Sales Analysis paginated report aims to provide an overview of the total sales performance over the years 2010 to 2014 across different regions. The report includes a 3D Clustered column chart, a Smooth line chart, and a Tree Map. The visual representation of the data enhances our ability to identify trends, patterns, and disparities in sales, empowering decision-makers to make informed strategic choices and optimize sales strategies based on regional performance over the specified period. The Sales analysis reveals distinct sales patterns and trends in each region during the five-year period.

In 2010, the Pacific region had the highest sales with $20.910, followed by North America with $18.412 and Europe with $4.099. Europe in 2011 saw a significant surge, with sales reaching $1.418.937, while North America and the Pacific experienced substantial growth with sales of $3,029.857 and $2.128.407. In 2012, sales in Europe remained strong at

$1.969.424, while North America and the Pacific had sales of $1.744.653 and $2.128.407. In the Following year, the Europe region exceeded a remarkable sales amount that reached $5.464.395. Also, North America and the Pacific experienced growth with sales of $6.547.712 and $4.339.443, respectively.

Overall, the Pacific had the highest sales in 2010, while **Europe** dominated in 2011. In 2012, the Pacific surged ahead, and **Europe** rebounded in 2013. However, **North America** outperformed all regions in 2014.

If the user prefers table representation and more details, for example, 2010 Pacific, he or she can select related columns from one of the charts. Thus, we navigated to the below paginated report.

Picture 19: Extracted report, based on region.

Sales Information

Years	Region	Category	Model	Product No	Total Sales	Cost	Net Profit
2010	Pacific	Bikes	Mountain-100	344	$3,400	$1,912.15	$1,488
				346	$3,400	$1,912.15	$1,488
				351	$3,375	$1,898.09	$1,477
			Road-150	310	$3,578	$2,171.29	$1,407
				313	$3,578	$2,171.29	$1,407
				314	$3,578	$2,171.29	$1,407
Grand Total					$20,910	$12,236.29	$8,673.49

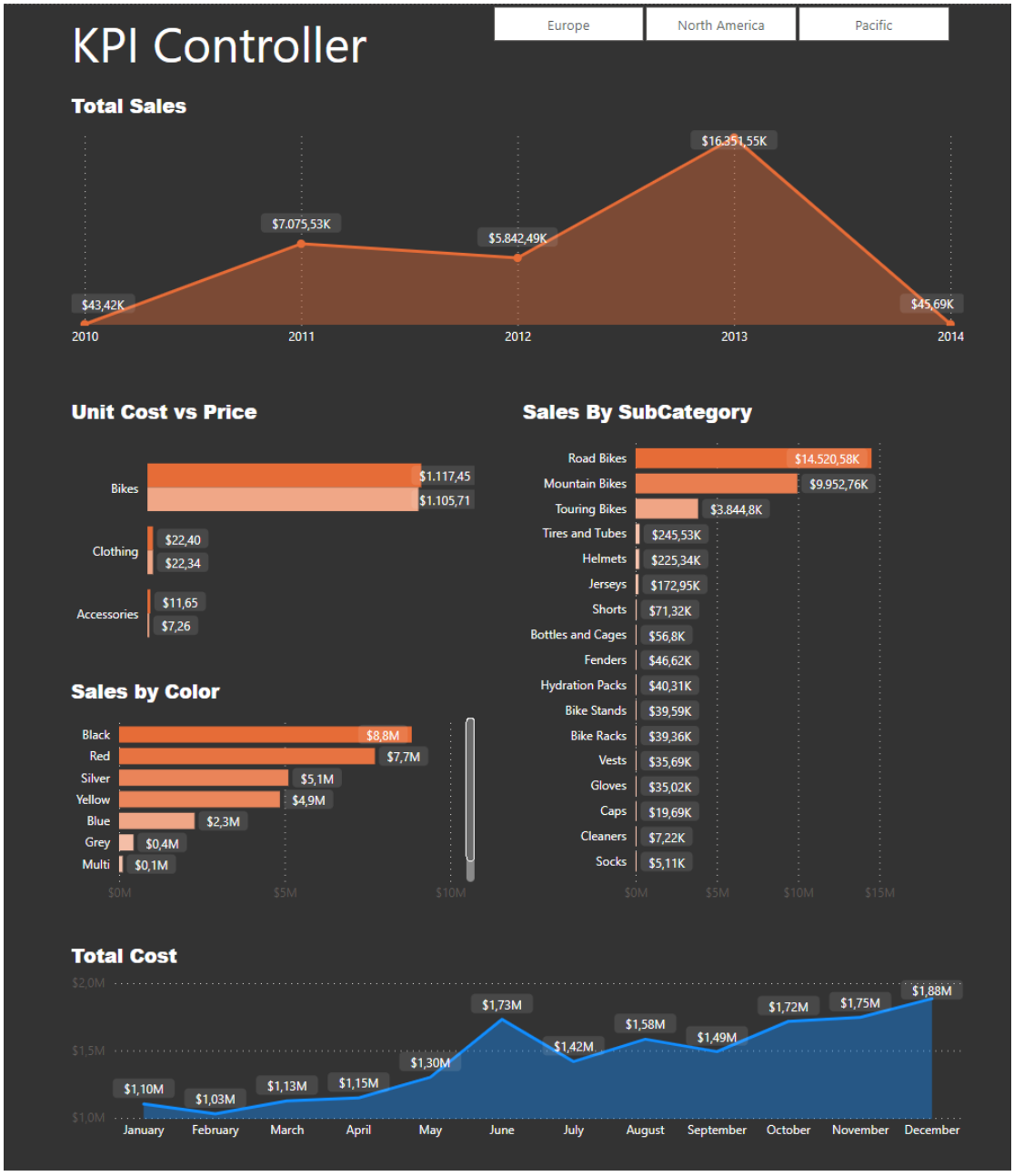
Page 1 of 1

Source: Own processing.

4.2.1 Power BI

The author created “Sales Analysis” from the Power BI Desktop application to show what could bring the integrated data sets usable in many applications with different approaches. In below Picture 23, the report can be accessed via Microsoft Power BI or Power BI Mobile application.

Picture 20: Sales Analysis



Source: Own processing.

4.3 Limitations of the research

While the research successfully culminated in the creation of integrated data, it is imperative to acknowledge a significant constraint that looms over its applicability - the absence of a real-world use case. The integrated data, though meticulously crafted by the author, has yet to undergo the litmus test of real-life implementation. This crucial step holds profound importance as it can unravel latent challenges and intricacies that are often obscured in controlled environments.

The absence of real-world application refrains the research from encountering the dynamic hurdles that manifest in practical scenarios. By subjecting the integrated dataset to the crucible of reality, a myriad of potential challenges would surface, enriching the depth and breadth of the study. Issues such as server loading and speed could potentially impede the seamless flow of data, while the intricacies of data reading and transformation may demand innovative solutions to ensure efficiency and accuracy. Furthermore, the specter of data leakage, a critical concern in contemporary data management, could cast its shadow, necessitating robust protective measures.

The merit of applying the integrated dataset in a real-life context transcends the realm of theoretical validation. It serves as a litmus test that gauges the viability, scalability, and resilience of the proposed data integration technique. The crucible of real-world application has the potential to either fortify the foundations of the research or expose areas that demand refinement and enhancement.

Incorporating the integrated dataset into an actual scenario would not only substantiate the findings but also empower the author with a wealth of experiential insights. These insights, garnered through the crucible of practical challenges, can significantly enrich the understanding of data integration dynamics. As the research charts its course towards practical relevance and tangible impact, the prospect of real-world application stands as an indispensable milestone, poised to elevate the study from a theoretical construct to a potent instrument of transformative change.

5. Conclusion

Throughout the whole of the thesis, the subject of "Data Integration Techniques in Database Warehouse Applications" is investigated and dissected in great detail. As a consequence of studying the theoretical component of the content, the author has a better understanding of the "Data Integration" setting as well as the ETL and ELT ideas that are associated with it. In addition, the author has mentioned Data Integration Tools, in addition to potential data issues and worries that may be the result of the ETL procedure.

However, the author begins by providing a brief overview of the most important terminology that pertain to data integration. These definitions include XML processes, semantic WEB, Data integration and its kinds, data warehouse, and the application of ETL processes as well as ELT processes, along with a discussion of the ways in which these processes vary from one another.

In addition, the author explores the "Kettle Method," as well as the TPC-DI technique, SSIS, and ODI.

The author presents a step-by-step overview of his or her own database in the portion of the book that is dedicated to practical application. The author focuses primarily on how the different data items are related to one another throughout this walkthrough. In addition, the author digs even further into the specifics by presenting a demonstration of the "Sales reports" and the many options for filtering that can be found inside the database. During the phase of the project in which the author is responsible for hands-on work, he discovers a few minor issues with data integration, one of which is the presence of "NULL values," but he is able to efficiently address these issues.

The results of this study have a number of important repercussions for companies that are interested in improving their data management procedures. As was proven, efficient data integration may result in more accurate reporting as well as richer statistical insights, which in turn helps drive informed decision-making.

It would be beneficial to future initiatives to go further into enhancing data integration procedures, possibly by investigating newer tools or approaches. This would be a good area to focus on. Additionally, the incorporation of artificial intelligence or machine learning methods might be a viable route, having the ability to automate and further enhance data management and analysis procedures. This would be a positive step in the right direction.

6. References

1. **Macura, M.** *Integration of Data from Heterogeneous Sources using ETL Technology. [online]. [Accessed: 30-04-2023]. Available at: Computer Science.* s.l. : Computer Science, AGH., 2014.

2. **Tous, R.** *Data integration with XML and semantic web technologies: novel approaches in the design of modern data integration systems. ISBN 978-3836471381.* s.l. : VDM Verlag Dr. Müller, 148 pages., 2008.

3. **Cichy, C. & Stefan, R.** *An Overview of Data Quality Frameworks. [online]. [Accessed: 30-05-2023]. Available at: https://www.researchgate.net/publication/331142988_An_Overview_of_Data_Quality_Frameworks.* 2019.

4. **Gullo, F.** *From Patterns in Data to Knowledge Discovery: What Data Mining Can Do. [online]. [Accessed: 30-06-2023]. Available at: https://www.researchgate.net/publication/274425359_From_Patterns_in_Data_to_Knowledge_Discovery_What_Data_Mining_Can_Do.* s.l. : Physics Procedia 62:18–22, 2015.

5. **Miguel A. Macias-Garcia, Victor J. Sosa-Sosa, and Ivan L.** *A generic approach for data integration using RDF, OWL and XML. [online]. [Accessed: 30-06-2023]. Available at: http://www.micai.org/2009/proceedings/complementary/cd/ws-mldm/215/20091105_MiguelMacias_Micai09_workshop_ML_and_DM.pdf.* 2009.

6. **Dong, X.L., Halevy, A.Y. and Yu, C.** *Data integration with uncertainty. [online]. [Accessed: 30-06-2023]. Available at: https://link.springer.com/article/10.1007/s00778-008-0119-9.* s.l. : In: VLDB J. 18.2, pp. 469–500., 2009.

7. **Al-Sudairy Dr. Mohammed T and T. G. K Vasista.** *Semantic Data Integration Approaches fro E-Governance. .* 2011.

8. **Agrawal, P.** *Foundations of Uncertain-Data Integration. [online]. [Accessed: 30-06-2023]. Available at: http://www.vldb.org/pvldb/vldb2010/pvldb%5C_vol3/R96.pdf.* s.l. : In: Proc. VLDB Endow. 3.1. pp. 1080–1090., 2010.

9. **Azeroual, O. Saake, G. and Abuosba, M.** *ETL Best Practices for Data Quality Checks in RIS Databases. [online]. [Accessed: 30-06-2023]. Available at: https://www.mdpi.com/2227-9709/6/1/10.* s.l. : Informatics 6,1., 2019.

10. **Theodorou, V.** *Data generator for evaluating ETL process quality. [online]. [Accessed: 30-06-2023]. Available at: https://doi.org/10.1016/j.is.2016.04.005.* s.l. : In: Inf. Syst. 63. pp. 80–100., 2017.

11. **Cristóbal-Salas, A.** *ETL Processing in Business Intelligence Projects for Public Transportation Systems. [online]. [Accessed: 30-06-2023]. Available at: https://doi.org/10.1007/978-3-030-38043-4%5C_4.* 2019.

12. **Yang, Q. and Helfert, M.** *Data Quality Problems in TPC-DI Based Data Integration Processes. [online]. [Accessed: 30-06-2023]. Available at: .* s.l. : In: Enterprise Information Systems - 19th International Conference., 2017.

13. **Q, Yang, M, Ge, and M, Helfert.** *Data Quality Problems in TPC-DI Based Data Integration Processes.* s.l. : Enterprise Information Systems - 19th International Conference, 2017.

14. **Dalcin, E.** *Data quality concepts and techniques applied to taxonomic databases. [online]. [Accessed: 30-06-2023]. Available at: https://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.427422.* s.l. : Southampton, 2005.

15. **Vyas, S.** *A comparative study of various ETL process and their testing techniques in data warehouse. [online]. [Accessed: 30-06-2023]. Available at: https://www.researchgate.net/publication/321101967_A_comparative_study_of_various_ETL_process_and_their_testing_tech.* s.l. : Journal of Statistics & Management Systems., 2017.

16. **Shields, W.** *SQL QuickStart Guide: The Simplified Beginner's Guide to Managing, Analyzing, and Manipulating Data With SQL. ISBN13: 978-1945051753.* s.l. : QuickStart Guides™ - Technology, 2019.

17. **T. Phungtua-Eng and S. Chittayasothorn.** *Slowly Changing Dimension Handling in Data Warehouses Using Temporal Database Features.* s.l. : Asian Conference on Intelligent Information and Database Systems. pp. 675-687, 2019. , 2019.

18. **S. H Ali El-Sappagh, A. M Ahmed Hendawi and A. Hamed El Bastawissy.** *A proposed model for data warehouse ETL processes. [online]. [Accessed at: 17-07-2023]. Available at: https://www.sciencedirect.com/science/article/pii/S131915781100019X.* 2017.

19. **D. Dumitriu and M. A.-M. Popescu.** *Enterprise architecture framework design in IT management.* s.l. : Procedia Manufacturing, vol. 46, pp. 932-940,, 2020.

20. **M. Souibgui, F. Atigui, S. Zammali, S. Cherfi and S. B. Yahia.** *Data quality in ETL process: A preliminary study.* s.l. : Procedia Computer Science, vol. 159, pp. 676-687, 2019.

21. **H. Homayouni, S. Ghosh and I. Ray.** *An approach for testing the extract-transform load process in data warehouse systems.* s.l. : Proceedings of the 22nd International Database Engineering & Applications Symposium., 2018.

22. **K. Jagadeesh.** *Business Intelligence Tools for Process Optimization in a Public Electricity Company.* 2020.

23. **H. M. Botos.** *Business Intelligence and Competitive Intelligence: The Evolution of The Terms.* s.l. : Research and Science Today, vol. 16, pp. 56-62, 2018.

24. **J. O. Chinyere.** *Data wharehouse: A tool for organizational efficiency.* s.l. : Nigerian Journal of Management Science, vol. 23., 2022.

25. **Chinyere, J.O.** *Data Warehouse: A tool for organizational efficiency.* s.l. : vol. 23., 2022.

26. **Simon. A.** *The Data Warehouse Staging Area.* New York. : Wiley Computer Publishing, 2011.

7. List of pictures, tables, graphs and abbreviations

7.1 List of pictures

7.2 List of figures