

|  |
| --- |
| Photo displaying partial image of two pie charts on a canvas-textured page |
|  |

**Data SQL&Analysis**

**Mudar Mostafa Ibraheem Alotoum**

Table of Contents

[1: SQL Server](#_Toc146972415)

1. [Part 1 Online Example](#_Toc146972415)
2. [Part 2 HR Database](#_Toc146972415)

[2: Data Analytics](#_Toc146972416)

[3: My Project](#_Toc146972417)

[Introduction](#_Toc146972418)

[My plan:](#_Toc146972419)

[My Implementation:](#_Toc146972420)

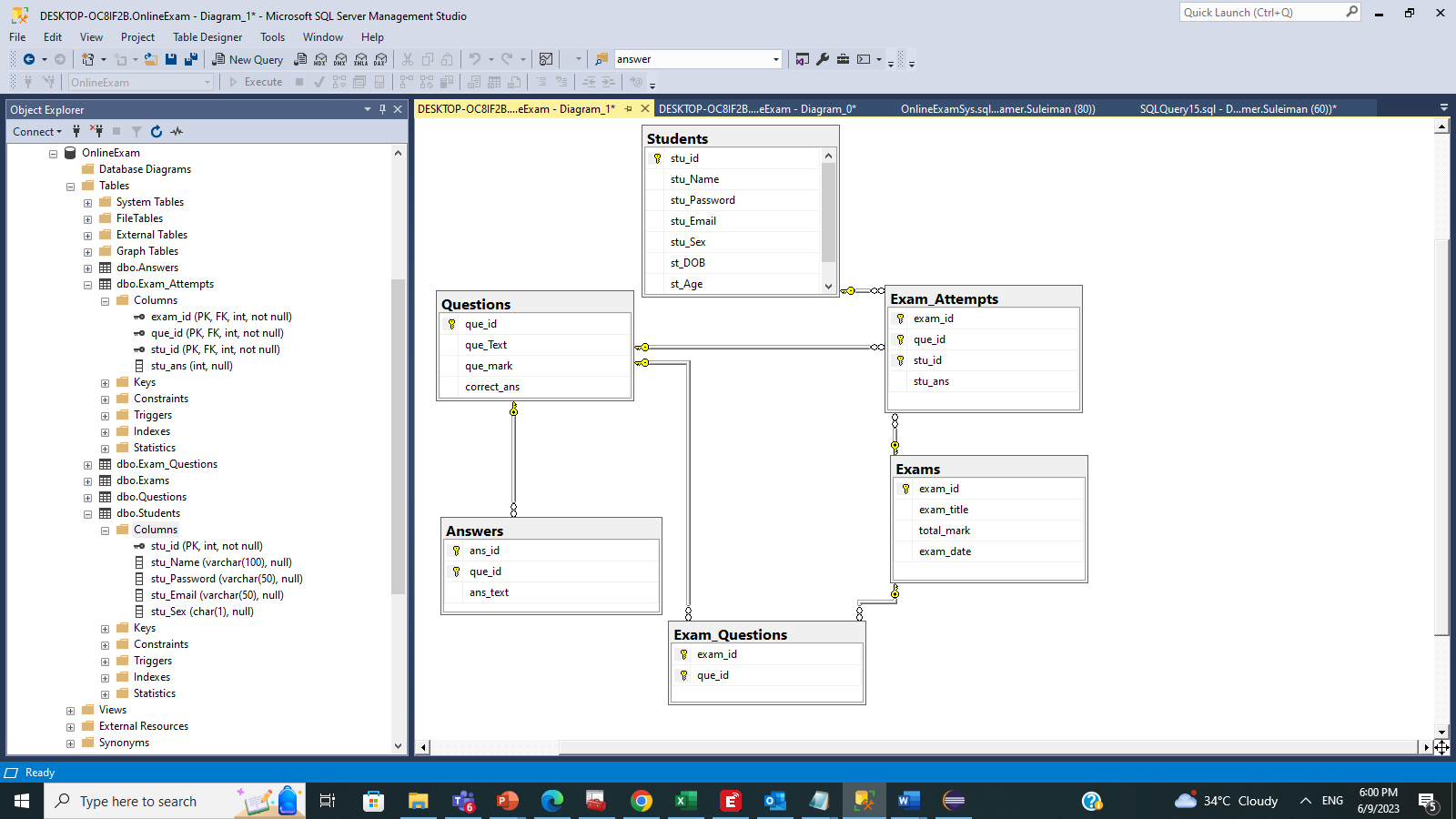
[My Data analysis results:](#_Toc146972421)

[Conclusion:](#_Toc146972422)

# **1: SQL Server**

# **Introduction**

This technical report intends to outline the design and schema of a database system for an online exam platform. This report aims to provide a comprehensive overview of the necessary tables and relationships required to support online exams. In the first step of my project, I will focus on an ERD diagram that illustrates an (Online Exam System).



Online Exams Portal. [1]

(The relationships between entities).

The diagram illustrates the connection between students, exam questions, and exam answers. The student’s database contains the students' information, such as their name, email address, and student ID. The Exam Questions database contains the questions' details, including the question text, question ID, and exam ID. The Answers database contains the answers' information, such as the answer text, answer ID, question ID, and student ID.

The tables have the following relationships:

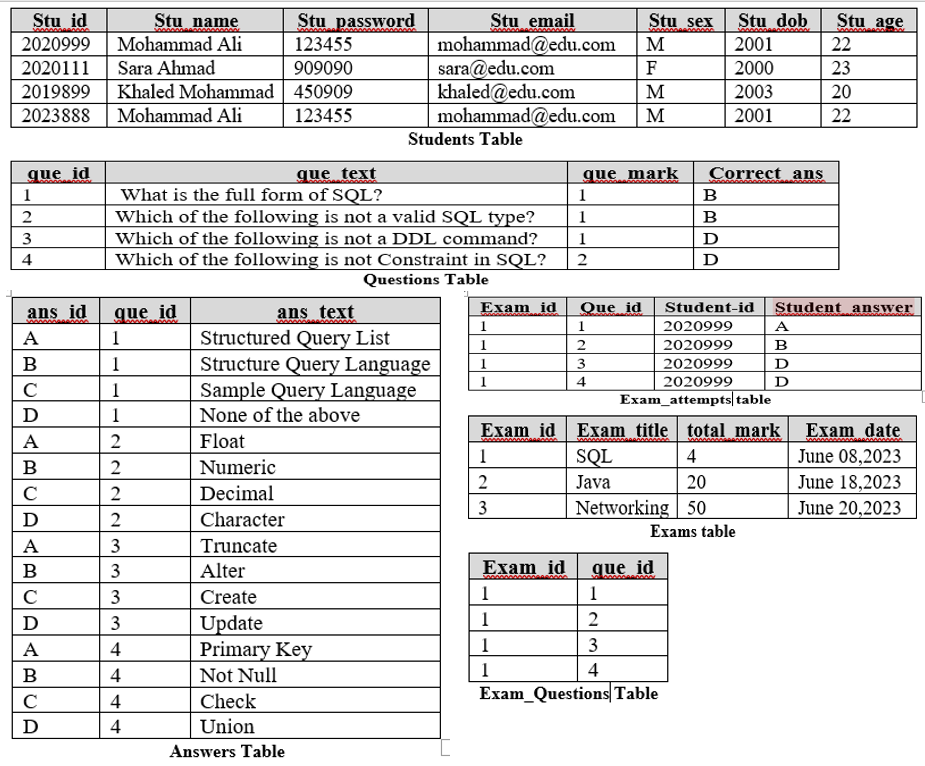
Students to Exam Questions: A student can take many exams, and each exam has many questions. This is a many-to-many relationship.

Exam Questions to Answers: An exam question can have many answers, and each answer belongs to one exam question. This is a one-to-many relationship.

Students to Answers: A student can give many answers, and each answer belongs to one student. This is a one-to-many relationship.

The ERP diagram can be utilized to monitor students' exam performance and identify areas where they need improvement. For instance, if a student consistently provides incorrect answers to a particular question type,

Here is an example of how the ERD diagram could be used:



I have inserted my data into the database. [2]

Based on the provided tables, there are a few potential improvements that could be made to enhance the design's normalization forms:

The image uses the Third Normal Form (3NF) for normalization. This is because all the tables are already in 1NF and 2NF and there are no transitive dependencies in any of the tables.

I created the database "Online\_Exam", and it was challenging work for me. As shown in the diagram, the yellow key outside the table indicates the presence of a primary key in the table. This means that three main tables contain primary keys and three others that contain foreign keys.

The SQL code you provided creates a database for an online exam system. The database includes the following tables(constraint).

• students: This table stores students' information, such as id, name, password, email, gender, date of birth, and age (one primary key).

• question: This table stores information about the questions, such as question ID, question text, question mark, and correct answer (one primary key).

• exams: This table stores exam-related information, such as exam ID, title, total mark, and exam date (one primary key).

• Exam Attempts: This table stores information about students' attempts at exams, such as exam ID, question ID, student ID, and student's answer (one composite key and three foreign keys).

• answers: This table stores different answers to each question (one composite key and one foreign key).

• Exam questions: This table stores the relationship between exams and questions (one composite key and two foreign keys).

Here I use subquery to find the correct answers for all students.

SELECT \* FROM Exam\_Attempts WHERE exam\_id in (SELECT exam\_id FROM question WHERE correct\_ans = Exam\_Attempts.stu\_ans);

Here I will use a single row function to customize output.

select CONCAT (students.STU\_name, students.STU\_password) from students

output: Khaled Mohammad450909

Here I will invoke conversion functions and conditional expressions,

to convert the upper and use case for convert ‘m,’ and’f’ to male and female.

SELECT

upper(students.STU\_name),

CASE

WHEN students.stu\_sex = 'm' THEN 'mail'

WHEN students.stu\_sex = 'f' THEN 'female'

ELSE 'nothing'

END AS uppersalary\_category

FROM students;

Here I used transact-SQL can be shared in multiple forms for management applications and users to view all information without a password because it is critical information.

CREATE VIEW vwEmployees AS

SELECT EmployeeID, FirstName, LastName, Position, Department

FROM Employees;

Now the database is ready and assessed with some queries against single and multiple tables.

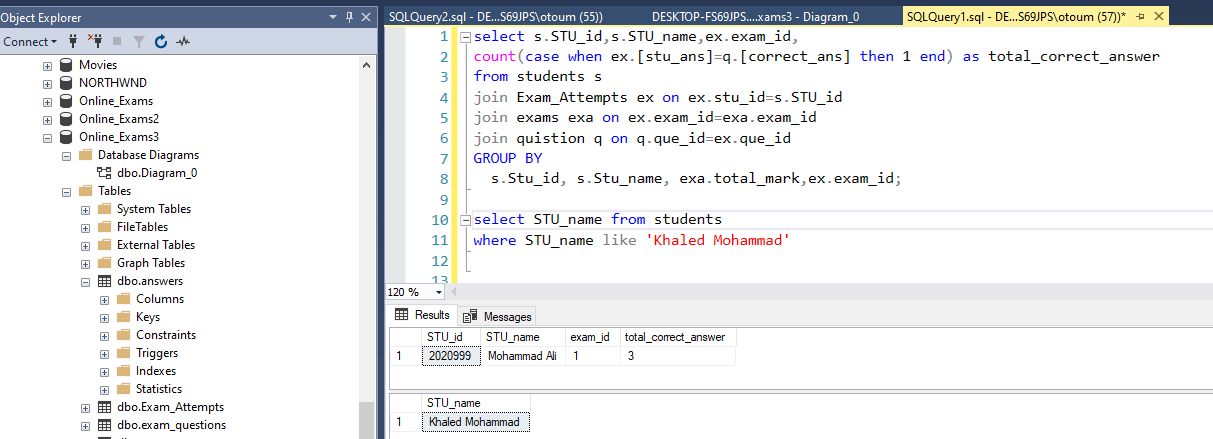
A screenshot of a computer

Description automatically generated

This query is for single tables [1].

These queries can be used in combination with conversion functions and conditional expressions to further enhance the output as below. Also, I used group functions to report aggregated data.

For the question " How can we know the student’s grade for a given exam, by query? ".



This query is for multiple tables [1].

Using a DDL command, I can easily modify an existing table's structure. This includes adding new columns, modifying column definitions, and adding or removing constraints.

like that:

ALTER TABLE Exams

ADD column\_name data\_type;

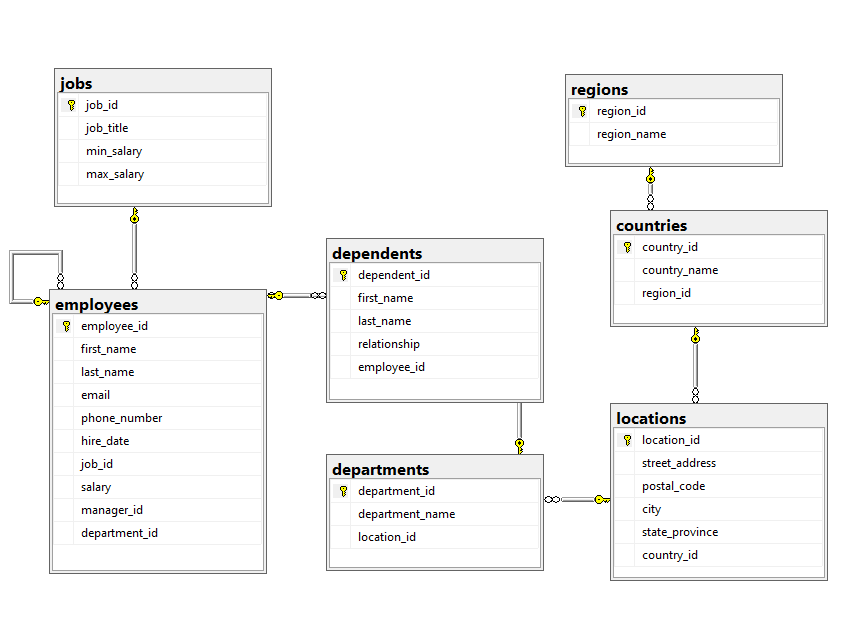
Or deleted

like that:

DROP TABLE Exams;

Part 2:

This example represents a basic HR database (ERD) diagram. It includes tables for employee information, departments, dependents, countries, locations, jobs, and regions.



[1]

Information about the tables:

|  |  |  |
| --- | --- | --- |
| **attribute** | **Description** | **Datatype** |
| 1. Regions: | | | |
| |  |  |  | | --- | --- | --- | | Region\_id | Region id for each row. | INT IDENTITY (1,1) PRIMARY KEY, | | Region\_name | Name of the region | VARCHAR (25) DEFAULT NULL | | | | |
| 1. Countries: | | | |
| |  |  |  | | --- | --- | --- | | Country\_id | Country\_id for each row. | CHAR (2) PRIMARY KEY | | Country\_name | Name of the Country | VARCHAR (40) DEFAULT NULL | | Region\_id | Region id for each row. | INT NOT NULL | | | | |
| 1. Locations: | | | |
| |  |  |  | | --- | --- | --- | | Location\_id | Location\_id for each row. | INT IDENTITY (1,1) PRIMARY KEY | | Street\_address | Name of street address | VARCHAR (40) DEFAULT NULL | | Postal\_code | Postal code for each row | VARCHAR (12) DEFAULT NULL | | city | City for each row | VARCHAR (30) NOT NULL | | State\_province | State province for each row | VARCHAR (25) DEFAULT NULL | | Country\_id | Country id for each row | CHAR (2) NOT NULL | | | | |
| 1. Jobs: | | | |
| |  |  |  | | --- | --- | --- | | Job\_id | Job id for each row | INT IDENTITY (1,1) PRIMARY KEY | | Job\_title | Job title for each row | VARCHAR (35) NOT NULL | | Min\_salary | Min salary for each row | DECIMAL (8, 2) DEFAULT NULL | | Max\_salary | Max salary for each row | DECIMAL (8, 2) DEFAULT NULL | | | | |
| 1. Departments: | | | |
| |  |  |  | | --- | --- | --- | | Department\_id | Department id for each row | NT IDENTITY (1,1) PRIMARY KEY | | Department\_name | Department name for each row | VARCHAR (30) NOT NULL | | Location\_id | Location id for each row | INT DEFAULT NULL | | | | |
| 1. Employees: | | | |
| |  |  |  | | --- | --- | --- | | Employee\_id | Employee\_id for each row | NT IDENTITY (1,1) PRIMARY KEY | | First\_name | First\_name for each row | VARCHAR (20) DEFAULT NULL | | Last\_name | Last\_name for each row | VARCHAR (25) NOT NULL | | email | Email for each row | VARCHAR (100) NOT NULL | | Phone\_number | Phone\_number for each row | VARCHAR (20) DEFAULT NULL | | Hire\_date | Hire\_date for each row | DATE NOT NULL | | Job\_id | Job\_id for each row | INT NOT NULL | | salary | Salary for each row | (8, 2) NOT NULL | | Manager\_id | Manager\_id for each row | INT DEFAULT NULL | | Department\_id | Department\_id for each row | INT DEFAULT NULL | | | | |
| 1. Dependents: | | | |
| |  |  |  | | --- | --- | --- | | Dependent\_id | Dependent\_id for each row | INT IDENTITY (1,1) PRIMARY KEY | | First\_name | First\_name for each row | VARCHAR (50) NOT NULL | | Last\_name | Last\_name for each row | VARCHAR (50) NOT NULL | | relationship | relationship for each row | VARCHAR (25) NOT NULL | | Employee\_id | Employee\_id for each row | INT NOT NULL | | | | |

**The provided database schema includes relationships between the tables. Here is an overview of the relationships:**

1-One-to-Many Relationship:

* The regions table has a one-to-many relationship with the countries table. One region can have multiple countries, but each country belongs to only one region. This is represented by the foreign key region\_id in the countries table referencing the primary key region\_id in the regions table.
* The countries table has a one-to-many relationship with the locations table. One country can have multiple locations, but each location belongs to only one country. This is represented by the foreign key country\_id in the locations table referencing the primary key country\_id in the countries table.
* The department's table has a one-to-many relationship with the employee’s table. One department can have multiple employees, but each employee belongs to only one department. This is represented by the foreign key department\_id in the employee's table referencing the primary key department\_id in the department's table.

2-Many-to-One Relationship:

* The countries table has a many-to-one relationship with the regions table. Many countries can belong to the same region, but each country belongs to only one region. This is represented by the foreign key “region\_id” in the countries table referencing the primary key “region\_id” in the regions table.
* The table locations have a many-to-one relationship with the countries table. Many locations can belong to the same country, but each location belongs to only one country. This is represented by the foreign key “country\_id” in the locations table referencing the primary key “country\_id” in the countries table.
* The employee’s table has a many-to-one relationship with itself to represent the manager-employee relationship. Many employees can have the same manager, but each employee has only one manager. This is represented by the foreign key “manager\_id” in the employee's table referencing the primary key “employee\_id” in the same table.

3-One-to-One Relationship:

* There are no explicit one-to-one relationships defined in the provided schema.

I will develop anonymous Transact-SQL blocks, and stored procedures, here, if I put a number greater than 10,000, then the executive manager’s salary will be visible through procedures.

create PROCEDURE [dbo]. [upsemployees]

@salary decimal

AS

BEGIN

declare @emo varchar(max);

set @emo = ' is manager';

SELECT first\_name+@emo, salary

FROM

employees

where salary>@salary

ORDER BY

Salary

print @emo;

END;

exec ups employees 10000;

Here I used Table-valued Functions to show all the employees have salaries of more than ten thousand.

.

CREATE FUNCTION udfPemployees (@salary INT)

RETURNS TABLE

AS

RETURN

SELECT employees.first\_name FROM employees WHERE salary > @salary;

select \* from udfPemployees(10000);

and here I will use the DML command to find all the information about employees.

select \* from employees e

full join dependents dep on dep.employee\_id=e.employee\_id

join departments d on e.department\_id=d.department\_id

join jobs j on j.job\_id=e.job\_id

join locations l on l.location\_id=d.location\_id

join countries c on c.country\_id=l.country\_id

join regions r on r.region\_id=c.region\_id

if I want to find the employee who gets the highest salary.

select top 1 first\_name,salary from employees

order by salary desc

# **2: The data analytics tools we have learned and utilized.**

# **2: The data analytics tools we have learned and utilized.**

* Power BI: Power BI is a business intelligence tool by Microsoft that enables users to connect to various data sources, create interactive visualizations, and share insights within an organization. It offers features like data modeling, advanced analytics, and real-time dashboards.
* SQL Server Integration Services (SSIS): SSIS is an Extract, Transform, Load (ETL) tool used for data integration and workflow automation. It provides a graphical development environment to design and manage ETL workflows, enabling you to extract data from various sources, transform it according to business rules, and load it into target systems or data warehouses. SSIS supports data cleansing, data quality checks, and data transformation operations.
* SQL Server Reporting Services (SSRS): SSRS is a server-based reporting platform that enables the creation, management, and delivery of formatted reports. It provides a wide range of features for designing and publishing reports, including data visualization, parameterization, and scheduling. SSRS supports various data sources, and reports can be rendered in different formats like PDF, Excel, or web-based formats.
* These three tools work together to form a comprehensive BI solution:
  + SSIS can extract data from various sources, transform and clean it, and load it into a data warehouse or data mart.
  + SSAS can then process and analyze the data in the data warehouse to create OLAP, allowing for interactive and multidimensional analysis.
  + Finally, SSRS can leverage the data from SSAS or other data sources to generate formatted reports and distribute them to end-users.
* Microsoft Excel: Excel is a widely used spreadsheet software that offers basic data analysis capabilities, including sorting, filtering, calculations, and charting. It also supports more advanced features like pivot tables and built-in functions for statistical analysis.
* SQL: Structured Query Language (SQL) is a programming language used for managing and analyzing relational databases. SQL allows you to query and manipulate data, perform aggregations, join tables, and apply filtering conditions.

# **3: Superstore Dataset**

Superstore is an online retailer that provides a variety of products in three categories: office supplies, furniture, and technology. Through my analysis, I hope to uncover trends, patterns, and insights that can be used to make informed decisions. The superstore's primary goal is to be the best shopping destination for customers, and my investment aims to contribute to that objective.

|  |  |  |
| --- | --- | --- |
| **attribute** | **Description** | **Datatype** |
| Row ID | Unique ID for each row. | Float |
| Order ID | Unique Order ID for each Customer. | Nvarchar (255) |
| Order Date | Order Date of the product. | Date Time |
| Ship Date | Shipping Date of the Product. | Date Time |
| Ship Mode | Shipping Mode specified by the Customer. | Nvarchar (255) |
| Customer ID | Unique ID to identify each Customer. | Nvarchar (255) |
| Customer Name | Name of the Customer. | Nvarchar (255) |
| Segment | The segment where the Customer belongs. | Nvarchar (255) |
| Country | Country of residence of the Customer. | Nvarchar (255) |
| City | City of residence of the Customer. | Nvarchar (255) |
| State | State of residence of the Customer. | Nvarchar (255) |
| Postal Code | Postal Code of every Customer. | Float |
| Region | The region where the Customer belongs. | Nvarchar (255) |
| Product ID | Unique ID of the Product. | Nvarchar (255) |
| Category | Category of the product ordered. | Nvarchar (255) |
| Sub-Category | Sub-Category of the product ordered. | Nvarchar (255) |
| Product Name | Name of the Product | Nvarchar (255) |
| Sales | Sales of the Product. | Float |
| Quantity | Quantity of the Product. | Float |
| Discount | Discount provided. | Float |
| Profit | Profit/Loss incurred. | Float |

In the table mentioned earlier, there are twenty columns and 9994 rows. The objective is to enhance data analysis by maximizing the depth of analysis and prediction capabilities. Additionally, the goal is to develop a user-friendly dashboard that streamlines the analysis process and reduces the time required for analysis.

**My plan:**

Constructing the dashboard necessitated a comprehensive comprehension of the dataset. Through extensive research, I identified redundant columns that needed to be removed. Additionally, I utilized Bower BI to modify certain column types, such as converting the "sales" column from a fixed decimal number to a decimal number format. This meticulous data preparation process laid the foundation for an optimized and accurate analysis.

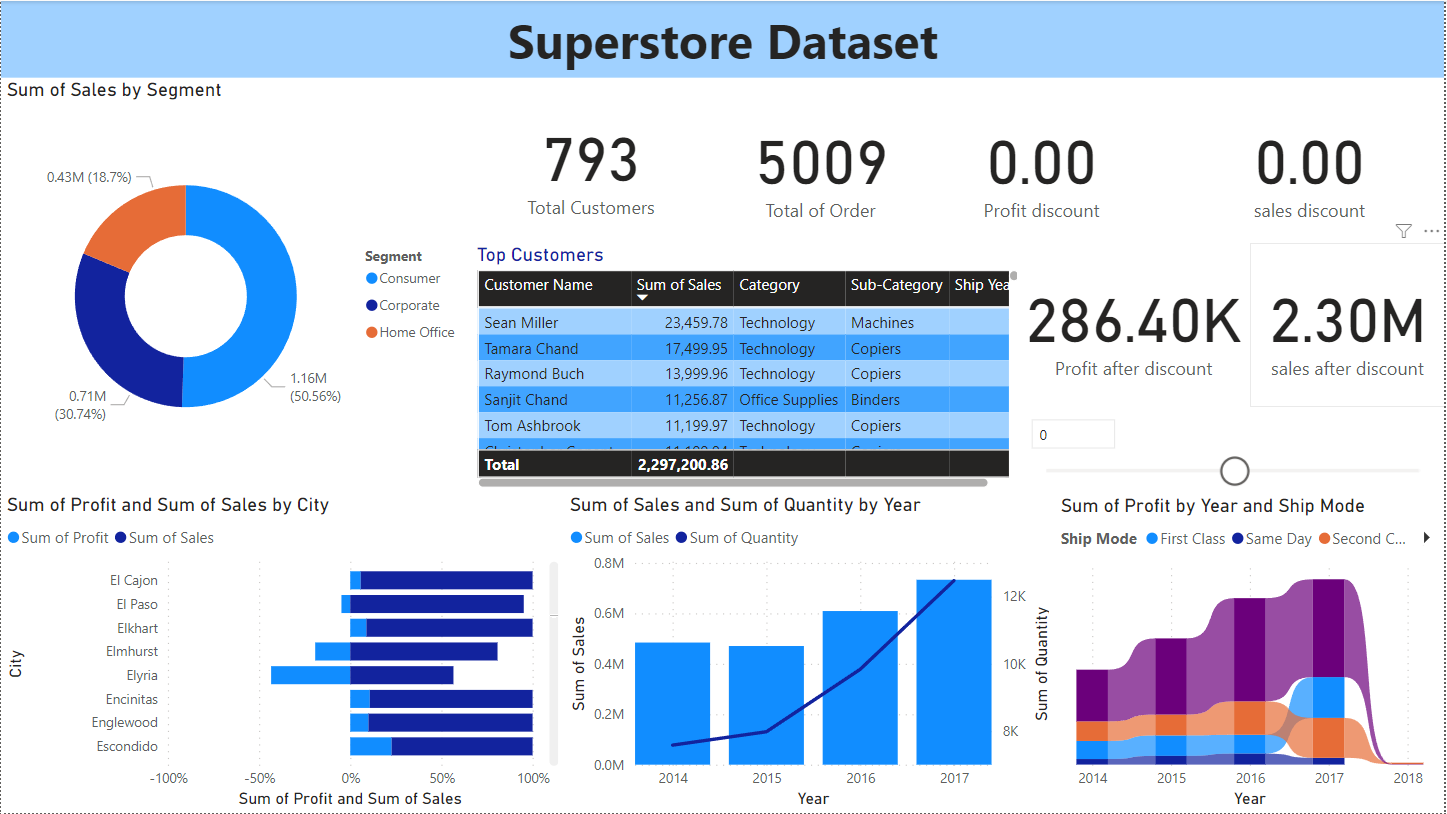
I created a new column called "Total Sales", calculated as the sum of the product of "Sales" and "Quantity,” I have a valuable metric that can be utilized to calculate the overall cost of all products. This additional column enhances the dataset by providing a comprehensive view of the total sales revenue generated. By leveraging the "Total Sales" column, further analysis, and calculations, such as determining the total cost, become more accessible and efficient.

"Total Sales": To gain insights into our most profitable customers.

" Shipping days": I want to know the average shipping time for each Ship mode.

**My Implementation:**

Now I want to present my dashboard to show a comprehensive overview of data which is useful for monitoring, measuring, and analyzing relevant data in key areas.



In the same way, I use the dashboard to analyze data to utilize the database tables to their best capacity. [3]

Firstly, I use a card from visualization to show the total profit amount and total sales amount after changing the percentage in a rectangular slicer for predictive analytics and a fixed part for the" total of orders" and "a total of customers".

Secondly, I use the donut chart to show the sum of sales by segment to describe composition static for a percentage of the segment of our customers.

Thirdly, I use the table to show the top customers and the details of their purchases.

Fourthly, I use a 100% stacked bar chart to show the sum of profit and the sum of sales by city.

The purpose of this chart is to know what the cities' profit increases or decreases.

Fifth, I used line and clustered column charts to show the sum of sales and the sum of quantity by year. The purpose of this chart is to show the number of quantities in store for every year.

Sixthly, I use a ribbon chart to show the sum of profit by year and ship mode. The purpose of this chart is to know what the kind of ship mode is through the years.

After creating the table and verifying data integrity, it was ready for database upload by ETL Tools.

**A screenshot of a computer

Description automatically generated**

Here I use (What-if analysis) to analyze different situations to decrease or increase the percentage of discount. [3]

This analysis aims to provide the institution owner with statistics to aid decision-making.

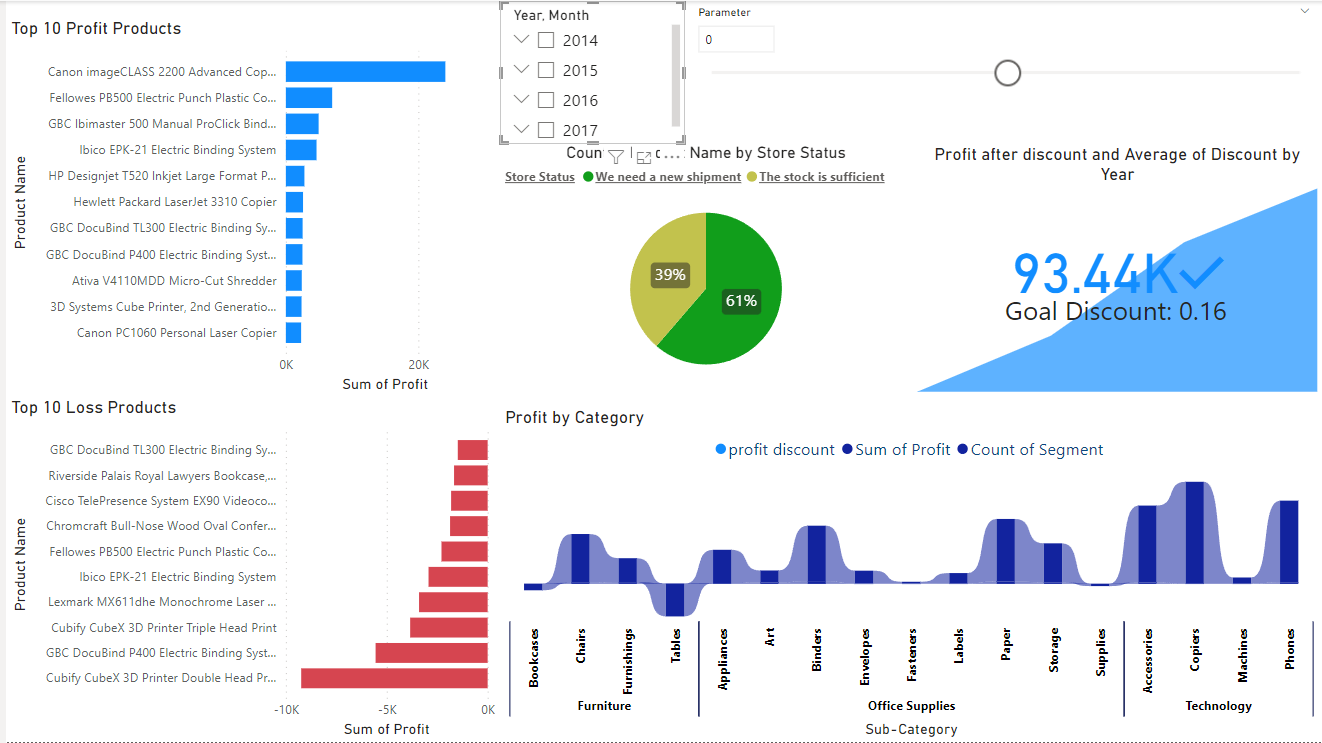
**A screenshot of a computer

Description automatically generated**

In the next step, I used measure to activate parameters in the (what-if analysis).

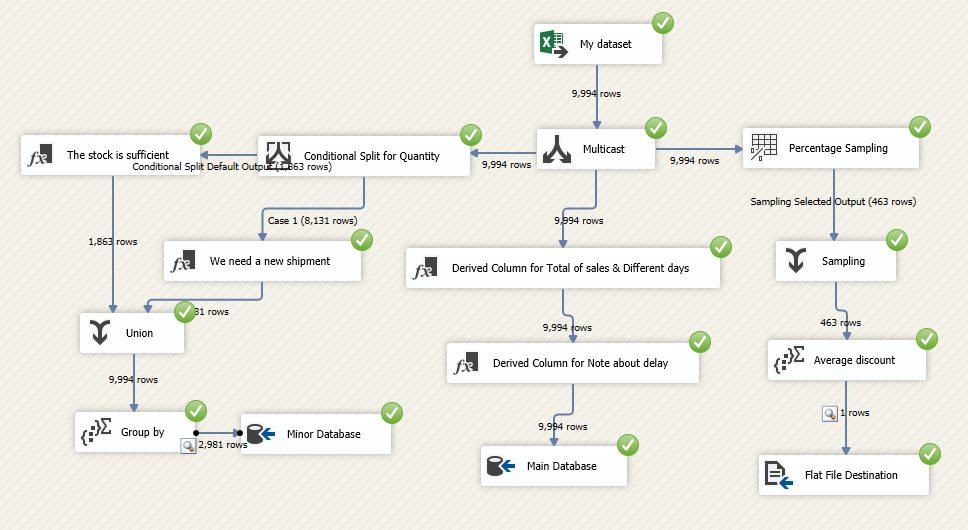
One example of a positive impact on a store is increasing the percentage of profits and sales. The benefits of increased profit can only be seen once the percentage has gone up, and this increase in profits can also lead to an increase in sales. [3]

After I built my dashboard, I used Microsoft SQL Server Integration Services for ETL (Extract, transform, and load), to load data from the Excel sheet to the SQL server and the opposite.



Here in this dashboard, the what-if analysis can be used to display results that affect all charts.

It is possible to specify a specific time during the analysis process or increase the discount percentage for all products with small volumes (need new shipment) or stock is sufficient.



As shown above we can use a raw dataset to transform (load) and manipulate our data. [4]

I utilized the multicast feature from a toolbox to perform three tasks concurrently.

The first task takes place through four stages.

1. I employ" Percentage Sampling" to randomly select a chosen percentage.
2. I use "Union All" from the "toolbox" that exists in "common" to take the selected output five percentages.
3. I use the "aggregate" function in the "toolbox" to find the average for a specific column (discount).
4. I insert the discount percentages on the text document (.txt) using the "flat file destination".

The second task consists of three stages.

1. I use the "Derived Column" to create the two columns needed.
2. I created a new column using the "Derived Column" function based on the previous step.
3. I use the "OLE DB Destination" component to transfer data to Microsoft SQL Server Management Studio.

The third task involves four stages.

1. To insert a split column with a specific condition, I utilize the "Conditional Split" function.
2. I use two “Derived Columns” to confirm or unconfirm the condition.
3. To gather data from various "Derived Column" sources, I utilize the " Union All " function.
4. I use the "aggregate" function in the "toolbox" to group by (Product Name) and (quantity on store)
5. I use the "OLE DB Destination" component to transfer data to Microsoft SQL Server Management Studio.

**My Data analysis results:**

1. Who are our customers by segment?

Consumer 50%, corporate 30% ,and home Office 20%

1. Do we have any cities that are currently making net profits or losses?

We are experiencing losses in Abilene but have been profitable in Aberdeen.

1. What about the yearly quantity?

Here are the numbers: 7581 in 2014, 7979 in 2015, and 9837 in 2016.

1. What do our customers prefer in shipping mode?

Standard Class

1. What is the total number of orders we have received?

The current total is 5009.

1. How many customers do we have in total?

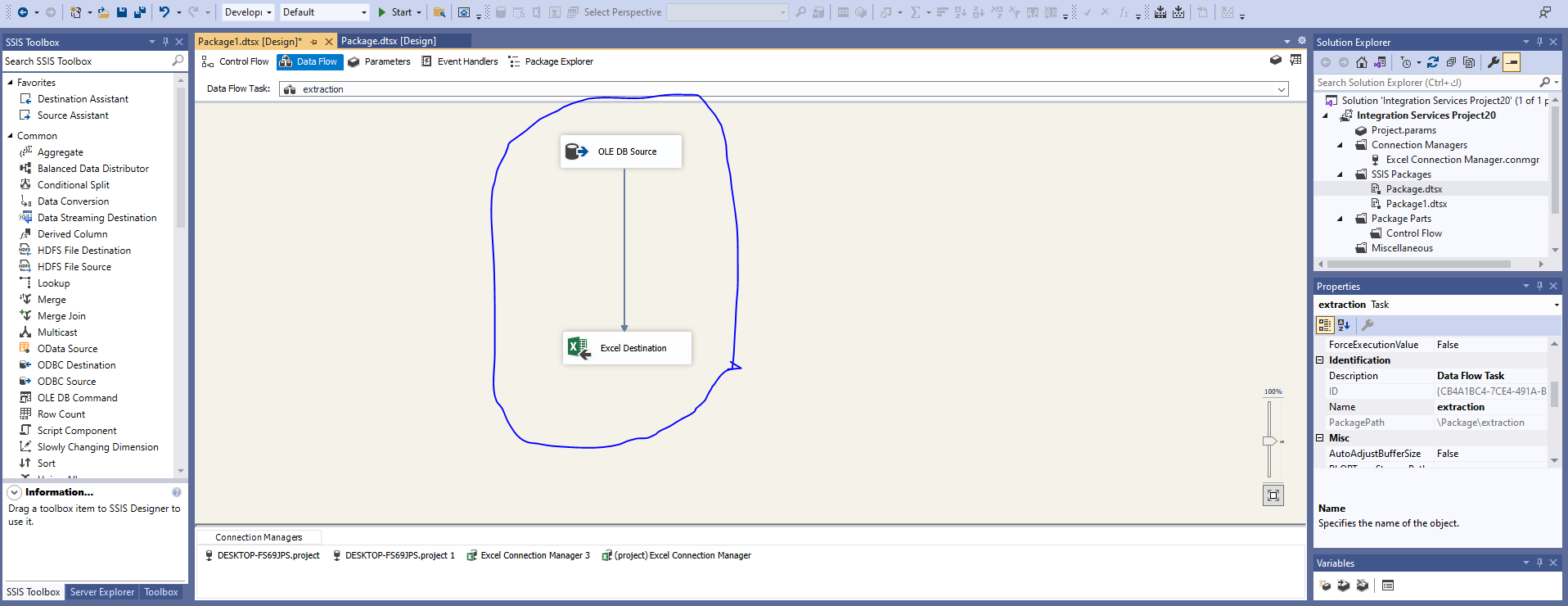
We have a total of 793 customers.

1. Who are our top customers?

Sean Miller is currently our top customer.

1. How will we benefit from a discount?

Increasing the discount rate has the potential to result in higher profits for us.



[4]

Then I use Microsoft SQL Server Integration Services for ETL (Extract, transform, and load), to extract data I need to retrieve information from an SQL server and transfer it to an Excel sheet to make sure that everything is okay, and My data is ready to extract and load.

To display the superstore report, I utilize SQL Server Reporting Service. Firstly, I employ a header and footer to display the report's name, page numbers, and the current date and time. The table I constructed comprises eight columns.

Then, I add two parameters to display dates that are later than the input date and search for

the desired category.

I use color expressions from the Quantity properties to get the colors right for printing purposes.



A screenshot of a computer

Description automatically generated

As shown above, I differentiate between the quantities using expressions- quantities exceeding five appear in black while those below five appear in purple. [5]

**Conclusion:**

For this analysis, I used standard and simple data manipulation to gain insight into a dataset.

conducting real-time data analysis and designing dashboards often involves multiple rounds of feedback between the client (the team requesting the analysis) and the data analyst. The clients typically have evolving needs and are unsure of what they want. As a result, multiple workshops and iterations are necessary.

Finally, Power BI is a great tool for diving into datasets, and it provides a robust and easy-to-use dashboard for end users.

# **References**

-------------------------------------------------------------------------------------------------------------------

|  |  |
| --- | --- |
| [1] | AUTHOR NAME VIVEK CHOWDHURY, "Superstore Dataset," kaggle, 2014. [Online]. Available: https://www.kaggle.com/datasets/vivek468/superstore-dataset-final |
| [2] | "Microsoft Power BI (Microsoft Corporation)". |
| [3] | "Microsoft Excel version 2308 ". |
| [4] | Microsoft SQL Server Integration Services, Visual Studio 2019. |
| [5] | Microsoft SQL Server Report Services, Visual Studio 2019. |