ASSIGNMENT:- 1

AIM:
Load Data from Multiple Sources into a Target System

PROBLEM STATEMENT :
Import the legacy data from different sources such as Excel , Sql Server, Oracle etc and load in the target system

THEORY:
This assignment demonstrates the fundamentals of **Business Intelligence (BI)** and data integration using the **ETL (Extract, Transform, Load)** process. In real-world scenarios, organizations store data across multiple platforms such as Excel files, SQL Server databases, and Oracle systems. To enable effective decision-making, this distributed data must be centralized, cleaned, and made ready for analysis using BI tools like Power BI.
Let's briefly understand the components involved:

1.Microsoft Excel:
Excel is one of the most commonly used spreadsheet tools for storing structured data in tabular form. It allows users to perform basic data entry, sorting, filtering, and analysis using formulas and charts. In many organizations, Excel serves as a quick and flexible way to record and maintain business data like sales records, customer lists, or financial reports. In the ETL process, Excel is often the **source system** from which data is extracted.

2. SQL Server:
Microsoft SQL Server is a **relational database management system (RDBMS)** designed to handle large-scale, structured data efficiently. It stores data in tables with relationships and supports complex querying using Structured Query Language (SQL). SQL Server is often used in enterprise systems for transactional data such as inventory, HR, finance, and customer interactions. It also supports advanced operations like stored procedures, indexing, and data security. In BI, SQL Server acts as a robust data source for structured and relational datasets.

3. Oracle Database:
Oracle is another powerful **RDBMS**, widely used in large enterprises and known for its scalability, performance, and advanced security features. It supports complex transactions and massive datasets, and is often used in mission-critical applications such as ERP, CRM, and banking systems. Oracle databases also allow

querying using SQL and PL/SQL. Like SQL Server, Oracle is a frequent source in ETL pipelines, especially when integrating legacy or enterprise-grade data.

4.Power BI:

Power BI is a **Business Intelligence and Data Visualization tool** developed by Microsoft. It enables users to connect to a wide range of data sources, transform data using the Power Query Editor, and create interactive dashboards and reports. Power BI plays the role of the **target system** in the ETL process. Once data is loaded into Power BI, users can build visualizations such as bar charts, maps, and KPIs to gain insights and support strategic business decisions.

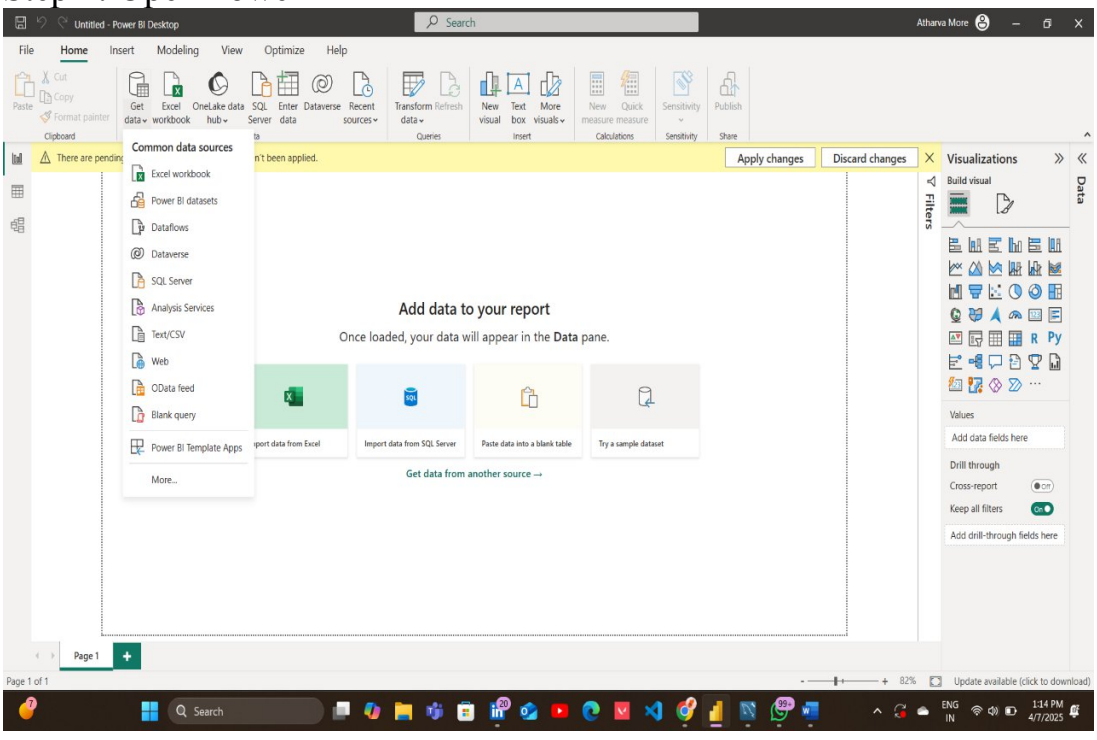
Overall Workflow in BI Context:

- **Extract** data from sources like Excel, SQL Server, and Oracle.
- **Transform** it by cleaning, reshaping, or aggregating using Power Query.
- **Load** it into Power BI for analysis and reporting.

This integrated approach ensures the data is accurate, consistent, and ready to support data-driven decision-making.

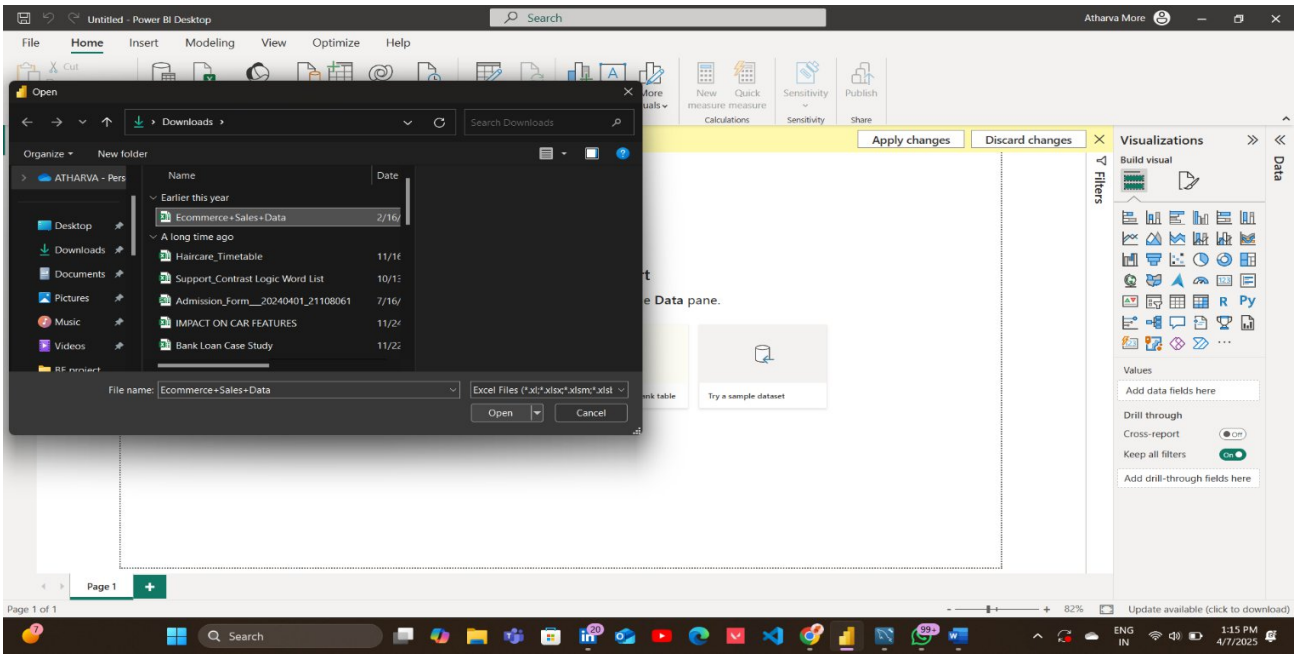
PROCEDURE :

Step 1: Open Power BI

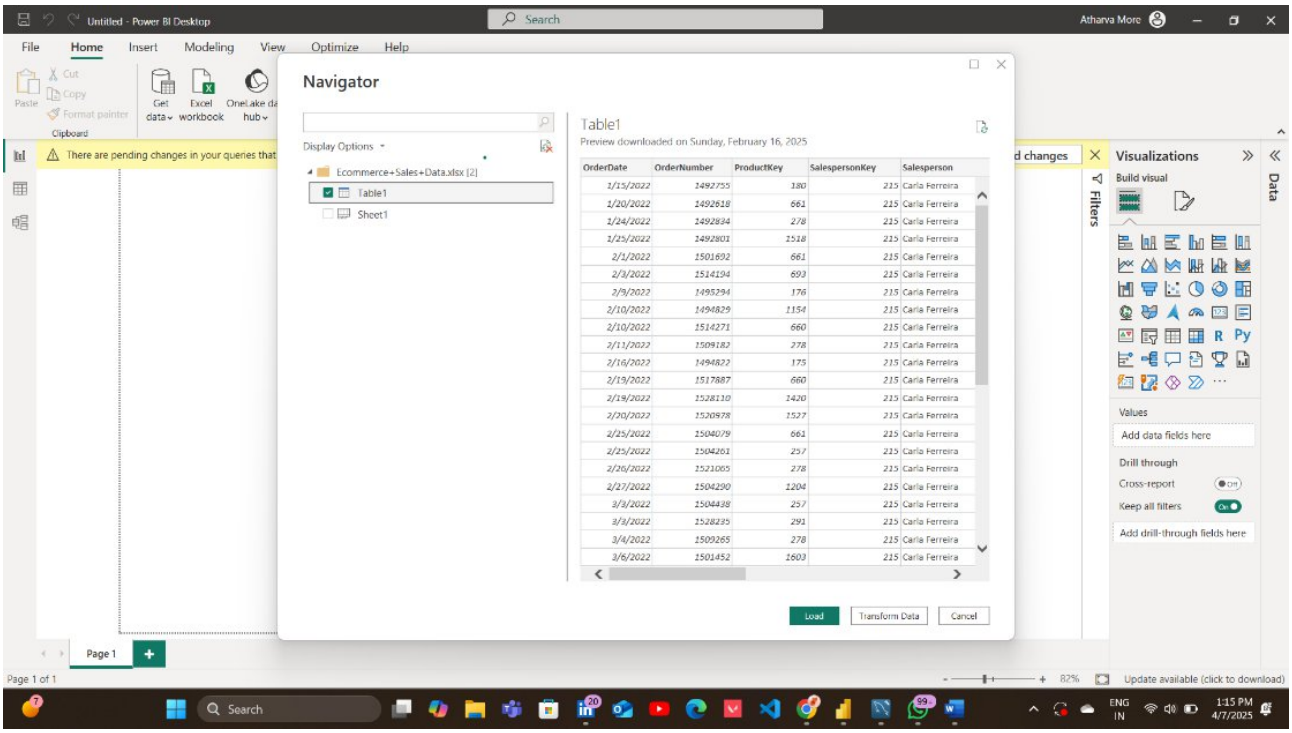


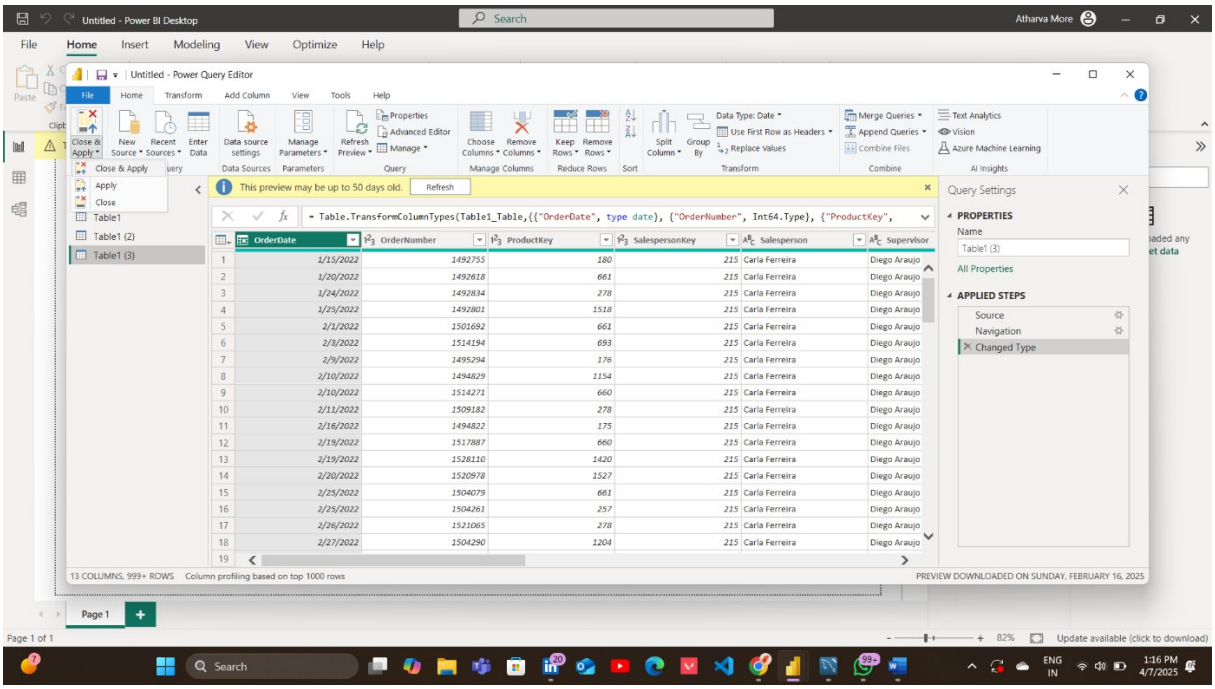
Step 2 : Click on Get data following list will be displayed → select Excel

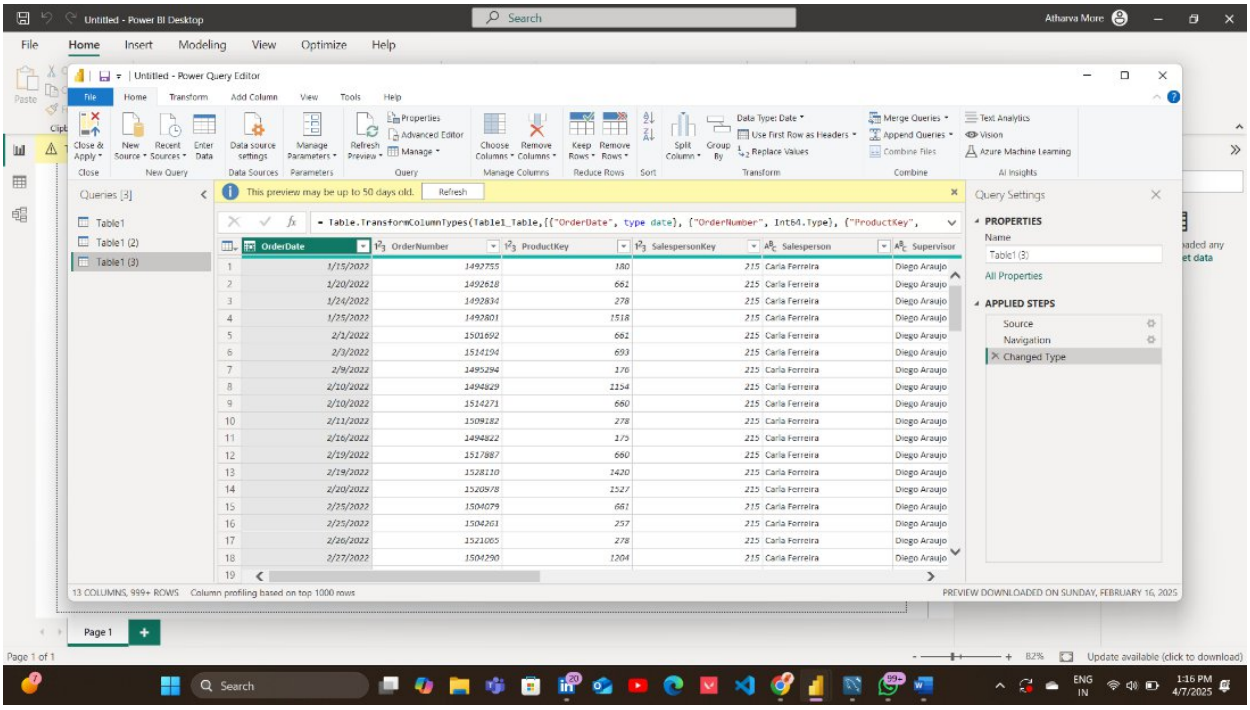
Step 3: Select required file and click on Open, Navigator screen appears



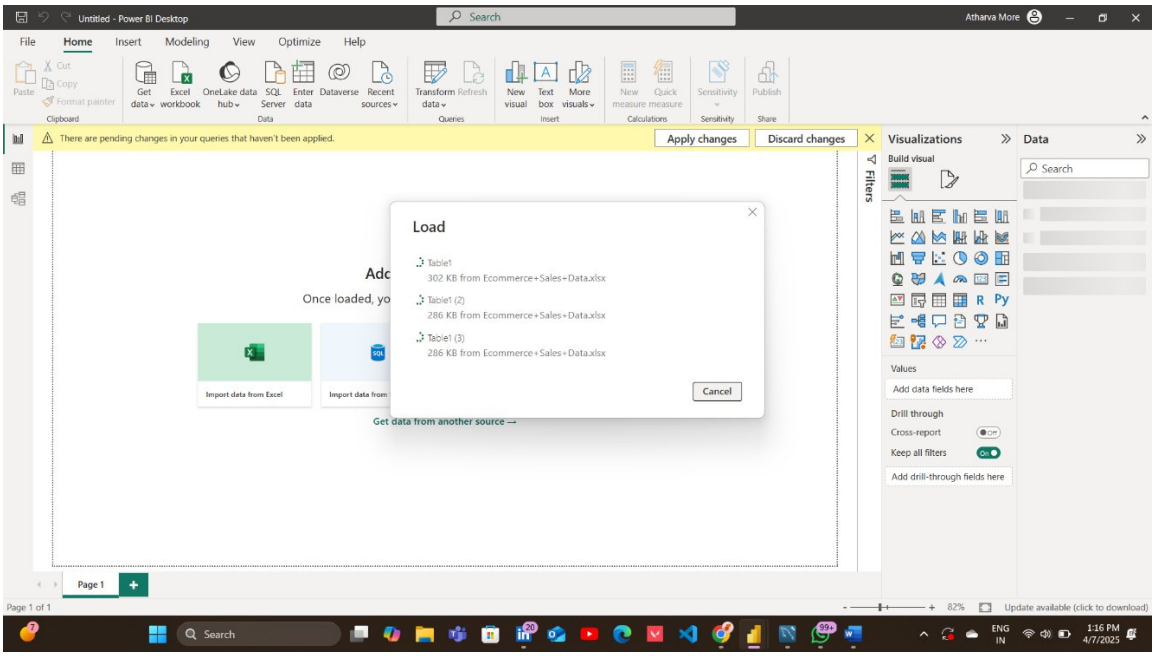
Step 4: Power query editor appears

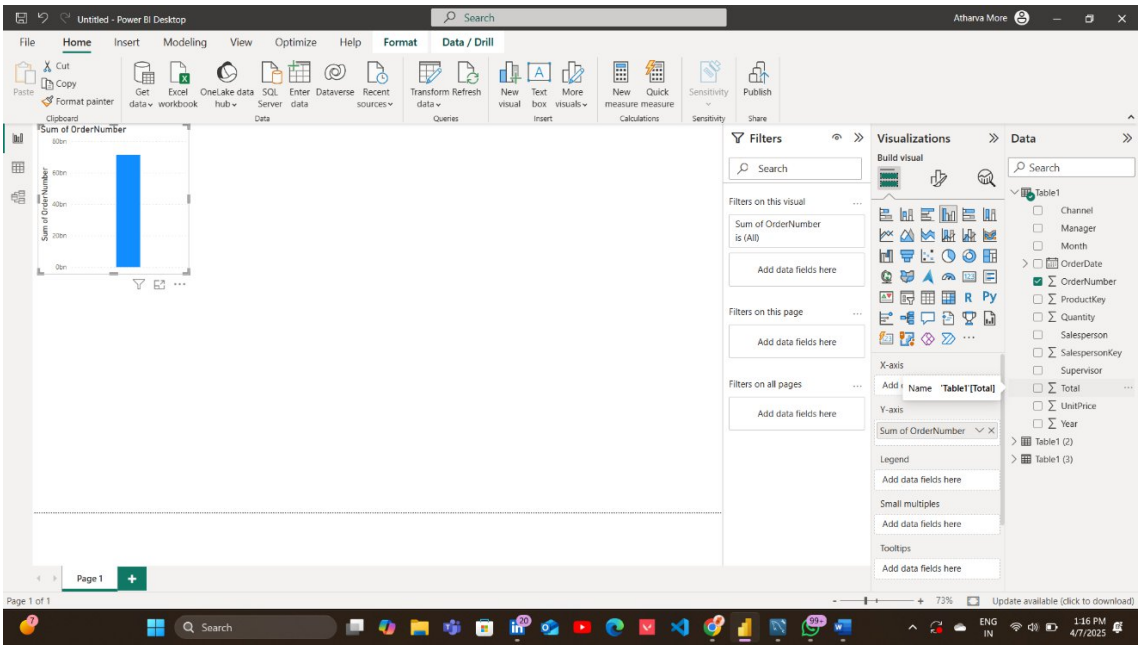






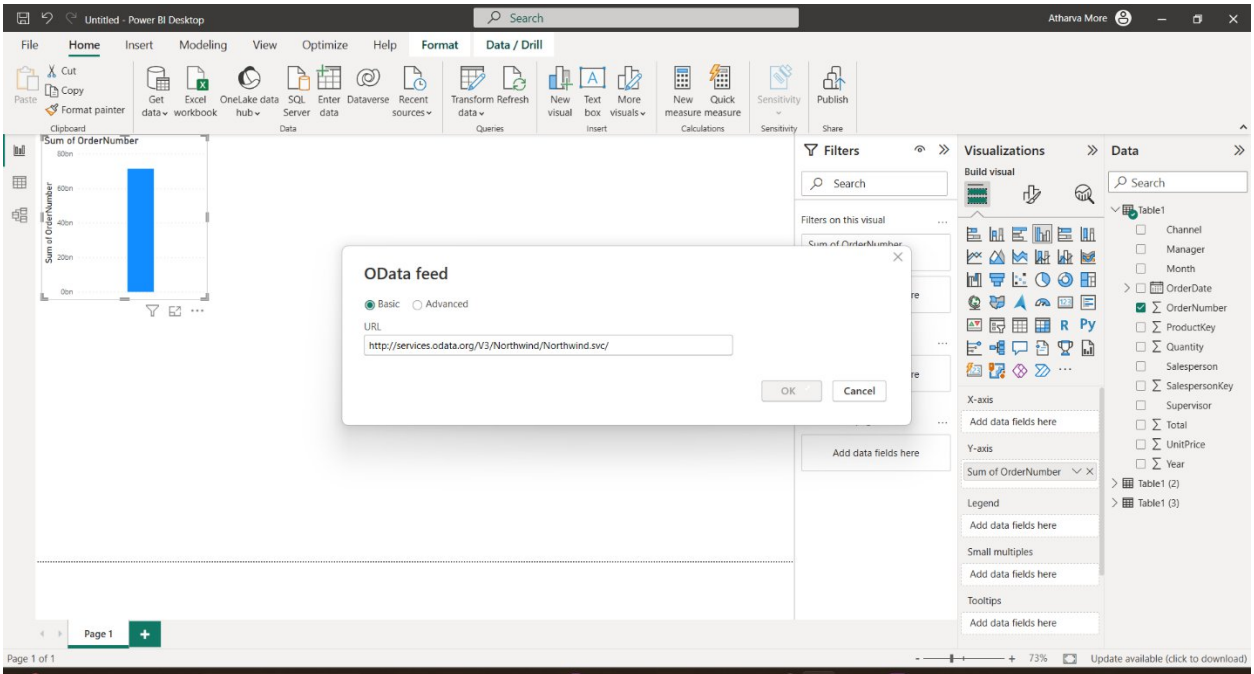
Step 5: Load data and start visualisation

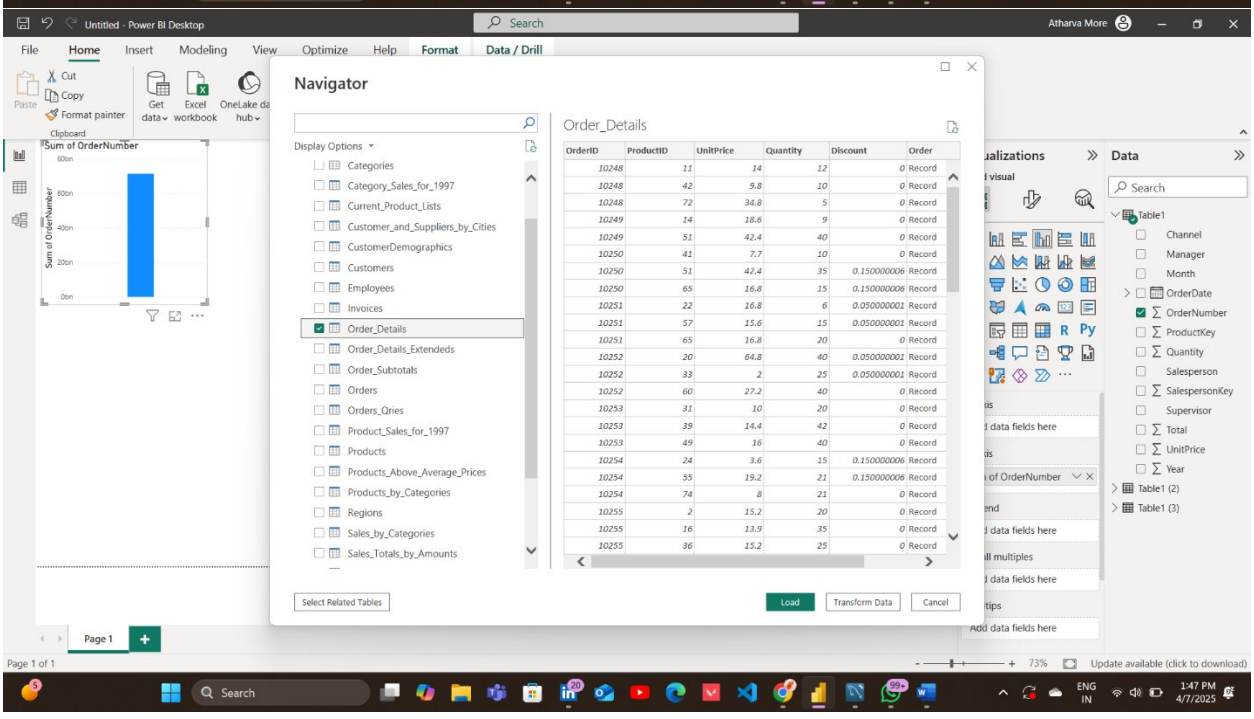

Untitled - Power BI Desktop
Search
Atharva More
File
Home
Insert
Modeling
View
Optimize
Help
There are pending changes in your queries that haven't been applied.
Apply changes
Discard changes
Load
Table1
302 KB from Ecommerce+Sales+Data.xlsx
Table1 (2)
286 KB from Ecommerce+Sales+Data.xlsx
Table1 (3)
286 KB from Ecommerce+Sales+Data.xlsx
Cancel
Get data from another source →
Visualizations
Build visual
Values
Add data fields here
Drill through
Cross-report
Keep all filters
Add drill-through fields here
Data
Search
Filters
Page 1
Page 1 of 1
82%
Update available (click to download)
1:16 PM
4/7/2025

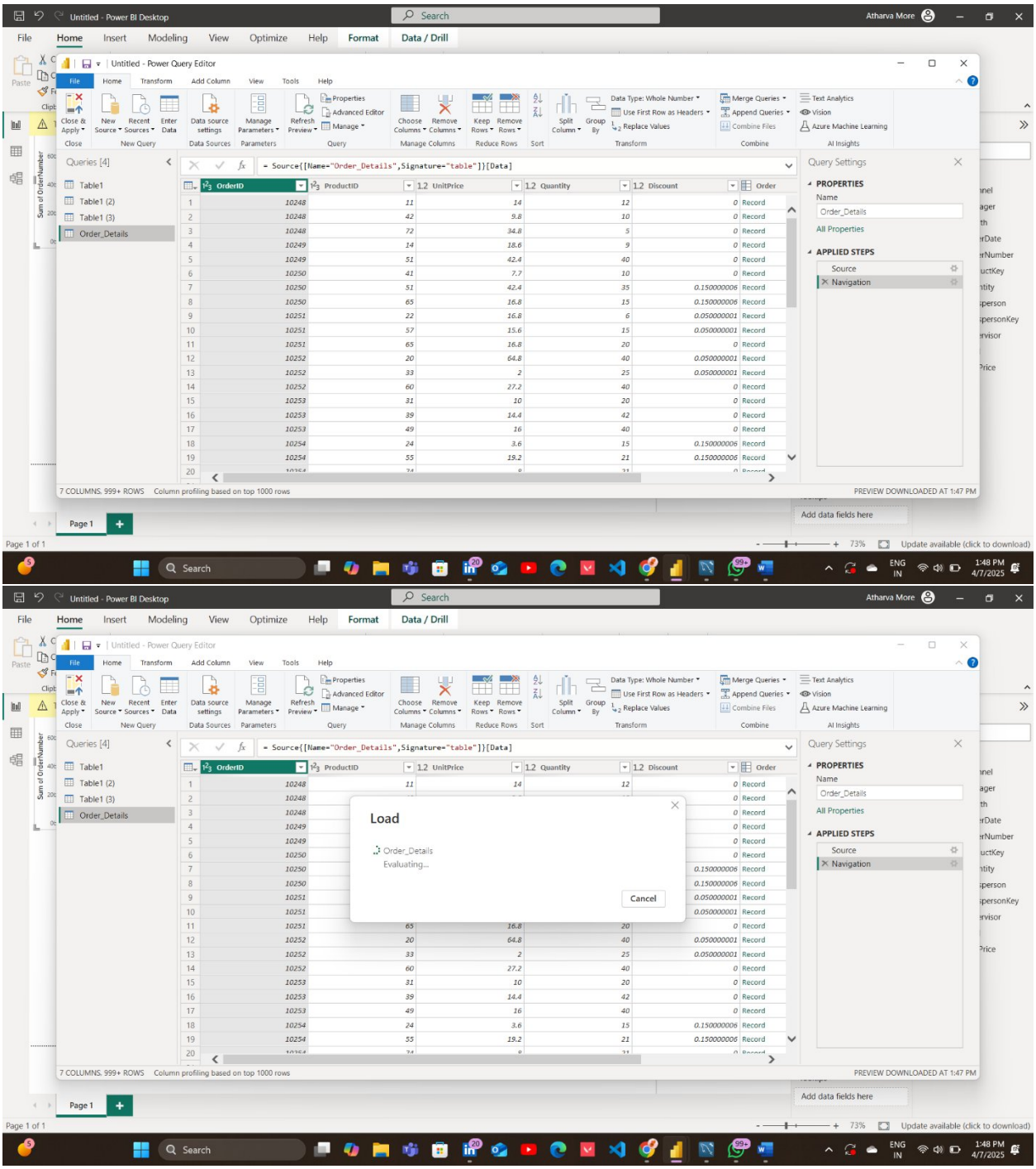

Untitled - Power BI Desktop
Search
Atharva More
File
Home
Insert
Modeling
View
Optimize
Help
Format
Data / Drill
Sum of OrderNumber
Filters
Filters on this visual
Sum of OrderNumber is (All)
Add data fields here
Filters on this page
Filters on all pages
Visualizations
Build visual
X-axis
Name 'Table1'[Total]
Y-axis
Sum of OrderNumber
Legend
Small multiples
Tooltips
Data
Table1
Channel
Manager
Month
OrderDate
OrderNumber
ProductKey
Quantity
Salesperson
SalespersonKey
Supervisor
Total
UnitPrice
Year
Table1 (2)
Table1 (3)
Page 1
Page 1 of 1
73%
Update available (click to download)
1:16 PM
4/7/2025

OD Data Feed

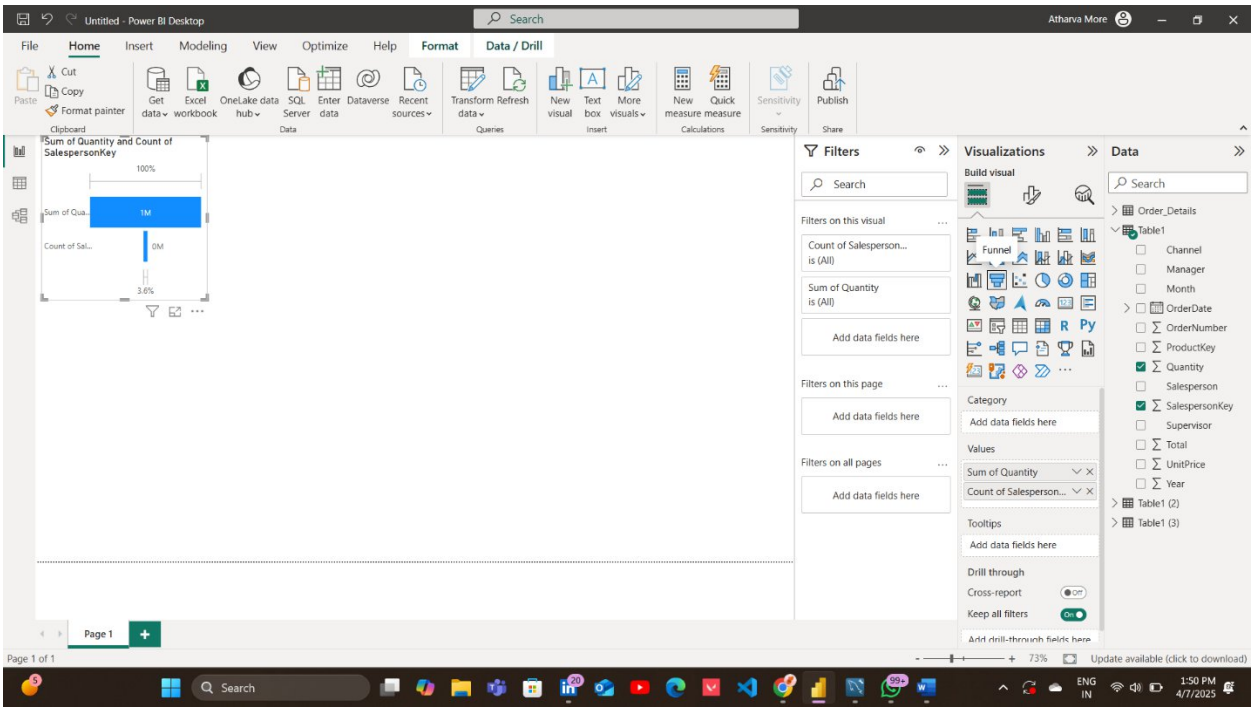
Step 6: Paste url as http://services.odata.org/V3/Northwind/Northwind.svc/
Click on ok





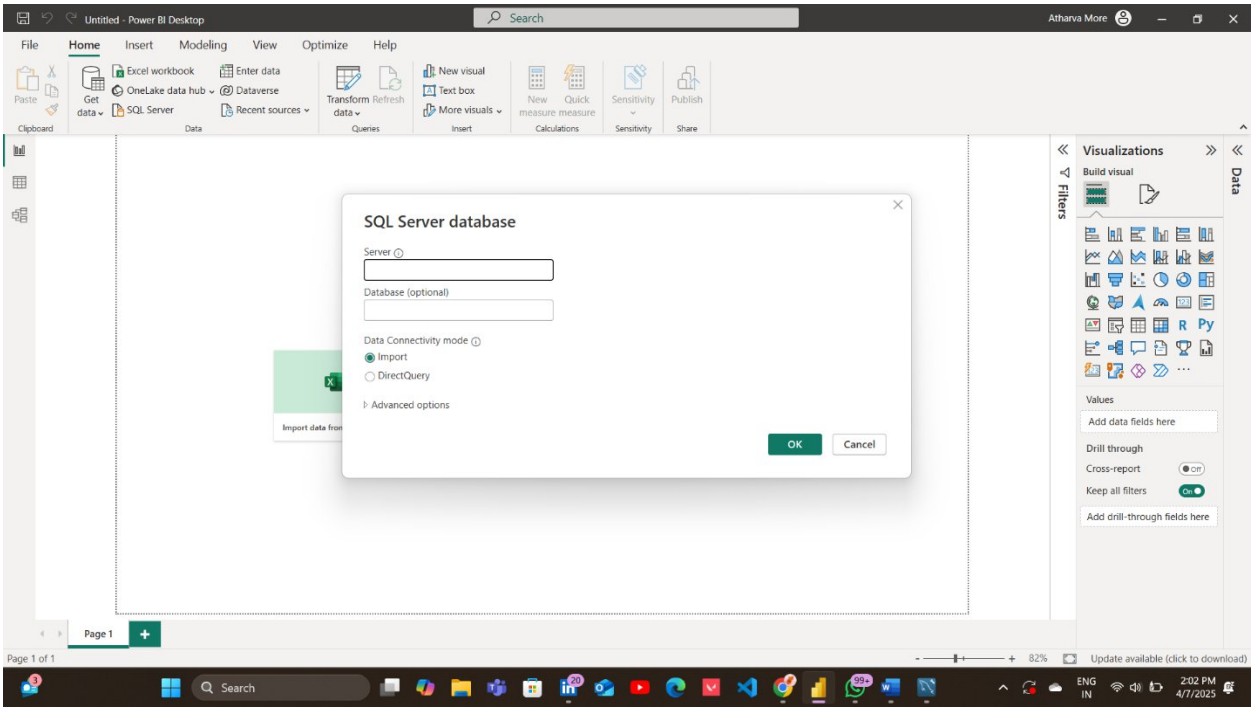

Untitled - Power BI Desktop
Search
Atharva More
File
Home
Insert
Modeling
View
Optimize
Help
Format
Data / Drill
Untitled - Power Query Editor
Transform
Add Column
Tools
Queries [4]
Table1
Table1 (2)
Table1 (3)
Order_Details
= Source{[Name="Order_Details",Signature="table"]}[Data]
OrderID
ProductID
UnitPrice
Quantity
Discount
Order
Query Settings
PROPERTIES
Name
Order_Details
All Properties
APPLIED STEPS
Source
Navigation
7 COLUMNS, 999+ ROWS
Column profiling based on top 1000 rows
PREVIEW DOWNLOADED AT 1:47 PM
Page 1
Add data fields here
Page 1 of 1
73%
Update available (click to download)
1:48 PM
4/7/2025

Load
Order_Details
Evaluating...
Cancel

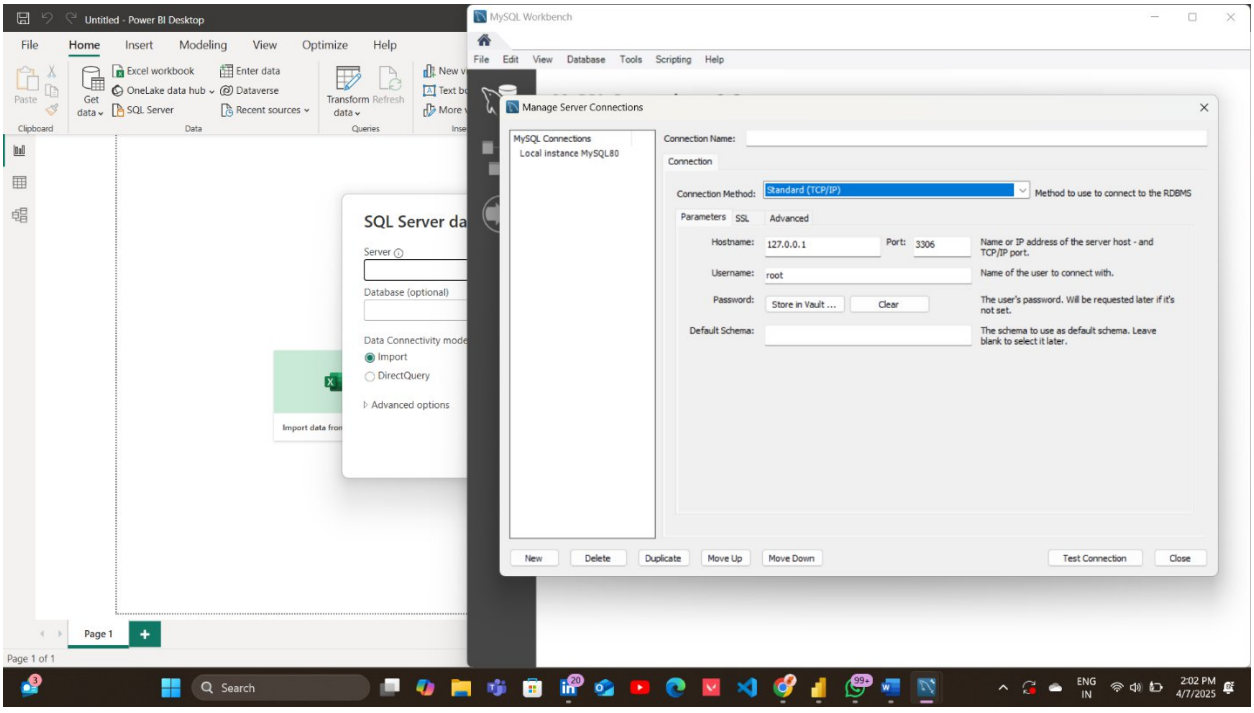

Untitled - Power BI Desktop
Search
Atharva More
File
Home
Insert
Modeling
View
Optimize
Help
Format
Data / Drill
Cut
Copy
Format painter
Paste
Clipboard
Get data
Excel workbook
OneLake data hub
SQL Server
Enter data
Dataverse
Recent sources
Data
Transform data
Refresh
Queries
New visual
Text box
More visuals
Insert
New measure
Quick measure
Calculations
Sensitivity
Publish
Share
Sum of Quantity and Count of SalespersonKey
100%
Sum of Qua...
1M
Count of Sal...
0M
3.6%
Filters
Search
Filters on this visual
Count of Salesperson... is (All)
Sum of Quantity is (All)
Add data fields here
Filters on this page
Add data fields here
Filters on all pages
Add data fields here
Visualizations
Build visual
Funnel
Category
Add data fields here
Values
Sum of Quantity
Count of Salesperson...
Tooltips
Add data fields here
Drill through
Cross-report
Off
Keep all filters
On
Add drill-through fields here
Data
Search
Order_Details
Table1
Channel
Manager
Month
OrderDate
OrderNumber
ProductKey
Quantity
Salesperson
SalespersonKey
Supervisor
Total
UnitPrice
Year
Table1 (2)
Table1 (3)
Page 1
Page 1 of 1
73%
Update available (click to download)
Search
ENG
IN
1:50 PM
4/7/2025

SQL SERVER

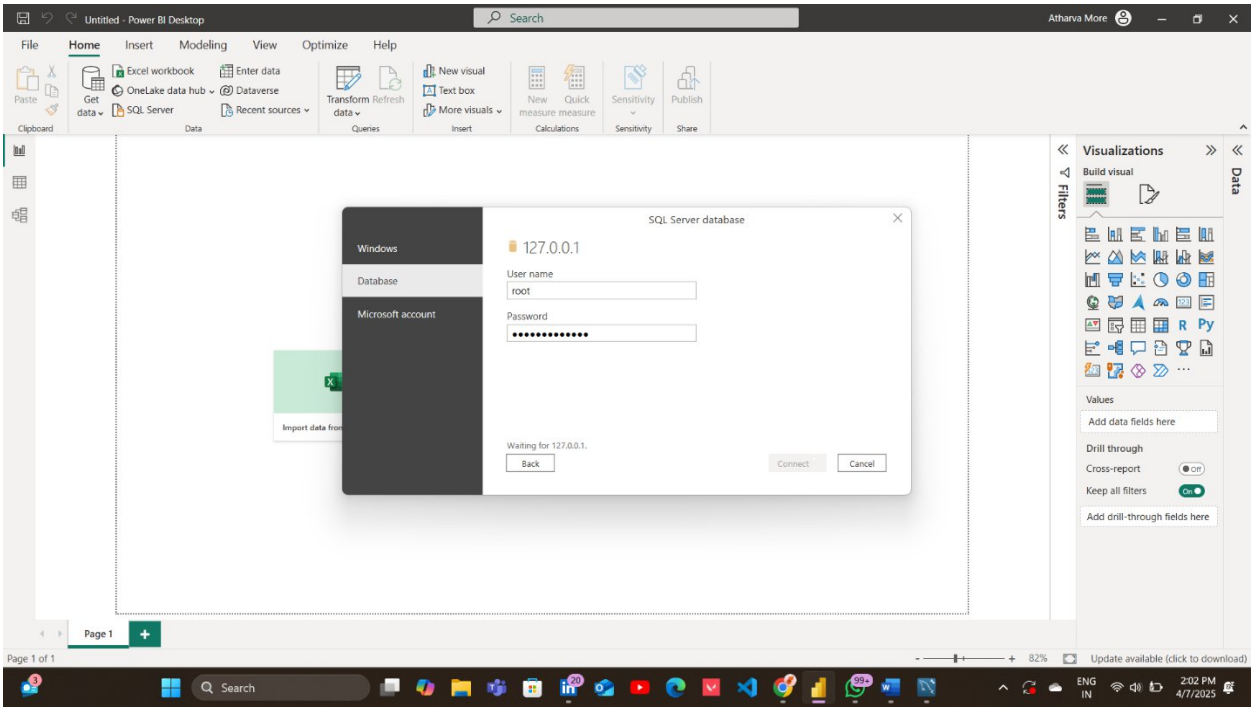
Step 1: Select SQL server from the Get Data



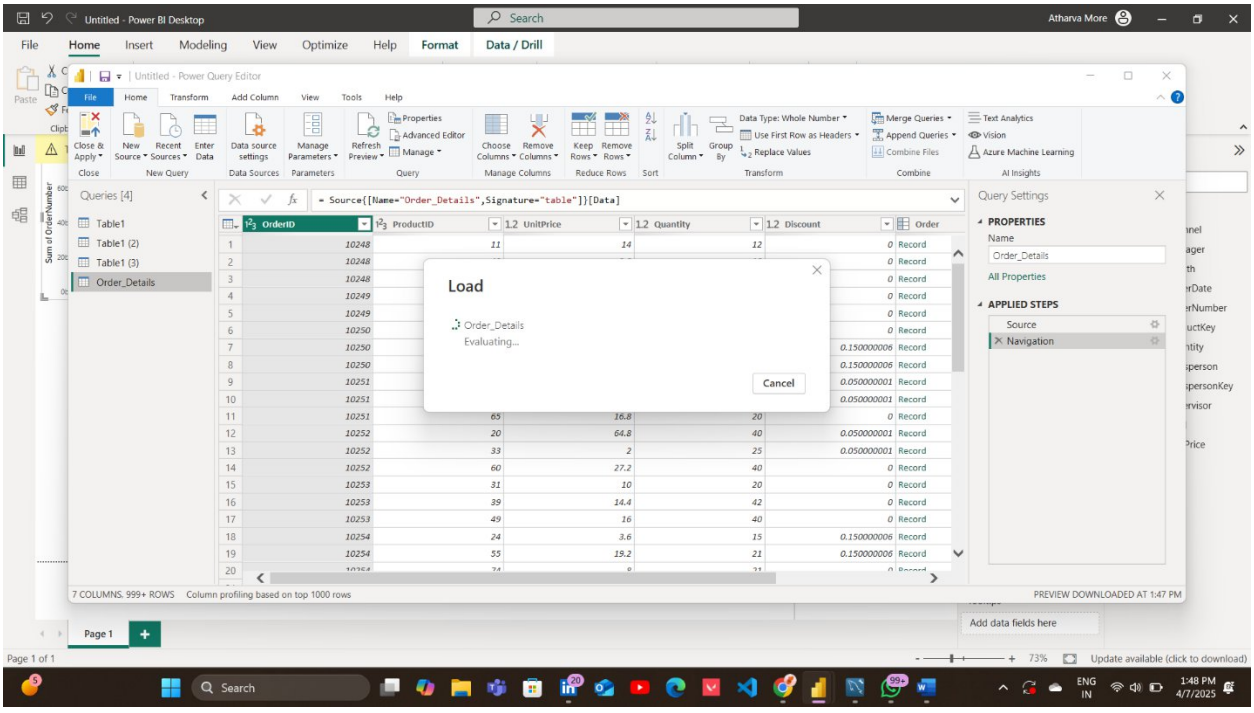
Step 2: Select the root or port number from the SQL server manage connections



Step 3: Enter User root and Password and establish the connection



Step 4: Import the Database from the SQL workbench



CONCLUSION:

Integrating data from multiple sources such as Excel, SQL Server, and Oracle plays a vital role in creating a centralized and consistent data environment. Through the ETL process, data is effectively extracted, transformed, and loaded into a target system, enabling efficient analysis and reporting. This enhances decision-making and supports better business insights.

ASSIGNMENT 2

AIM:

To extract data from raw sources, transform it into a clean and structured format, and load it into a target system for visualization and analytical insights.

PROBLEM STATEMENT:

Data Visualization from Extract, Transform, Loading process.

THEORY:

In the modern era of data-driven decision-making, Data Visualization is a powerful technique used to represent complex data in an intuitive and interactive manner. However, before data can be visualized effectively, it must go through a series of preparatory steps known as the ETL process — Extraction, Transformation, and Loading. This process is essential in ensuring that the data being visualized is clean, consistent, and meaningful.

1. **Extraction**

Extraction is the first phase of the ETL process, where raw data is collected from multiple and often heterogeneous sources. These sources can include:

- Flat files like Excel and CSV,
- Relational databases such as SQL Server or Oracle,
- Cloud platforms,
- APIs and web services.

The main goal of extraction is to gather all relevant data without affecting the performance of the source systems. This step ensures that no valuable information is missed during the data preparation process.

2. **Transformation**

Once data is extracted, it usually exists in different formats, structures, or quality levels. In the transformation phase, the data is cleaned, enriched, standardized, and restructured. Common transformation tasks include:

- Removing duplicates or null values,
- Changing data types (e.g., text to numbers),
- Normalizing data formats (e.g., dates and currencies),
- Creating calculated columns or aggregating data.

Transformation ensures that the data becomes uniform and suitable for accurate analysis and reporting.

3. **Loading**

The final step in the ETL process is loading the transformed data into a target system or data warehouse, where it becomes ready for visualization. Tools like Power BI or Tableau are then used to connect to this prepared data and create dashboards, charts, and other interactive visual reports.

4. **Data Visualization**

Once the ETL process is complete, data visualization tools allow users to explore and analyze the data graphically. Visualizations help in:

- Identifying trends and patterns,
- Comparing values across categories,
- Tracking KPIs (Key Performance Indicators),
- Making predictions and business decisions.

Tools like Power BI not only support visualizing data from local files (e.g., Excel) but also allow connections to live databases like SQL Server or cloud services, offering real-time insights.

PROCEDURE :

- **Import Required Libraries:**
  The necessary Python libraries such as pandas for data handling and matplotlib for visualization are imported to begin the ETL process.
- **Extract Data from Source:**
  An Excel file containing e-commerce sales data is used as the source. The file is opened and the relevant sheet is selected for further processing.
- **Convert Date Column:**
  The OrderDate column is converted to a proper datetime format to ensure consistency in date values, which is essential for time-based analysis.
- **Check for Missing Values (Before Cleaning):**
  The dataset is examined to identify missing values in each column. The total number of rows before transformation is also recorded to track changes.
- **Transform - Clean the Data:**
  Rows containing any missing values are removed to clean the dataset. This step ensures the quality and integrity of the data used for analysis.

- **Recheck After Cleaning:**
  The dataset is re-evaluated to verify that all missing values have been removed. The number of records after cleaning is noted for comparison.
- **Load - Save the Cleaned Data:**
  The cleaned dataset is exported and saved as a CSV file. This marks the completion of the "Load" phase in the ETL process.
- **Visualize the ETL Record Flow:**
  A bar chart is created to visually represent the number of records at each stage—extracted, after transformation, and loaded. This helps in understanding the impact of the transformation process and data cleaning on the dataset size.

CONCLUSION :

The ETL process plays a crucial role in preparing data for meaningful analysis and visualization. Through this assignment, we successfully extracted data from an Excel source, transformed it by handling missing values and ensuring consistency, and then loaded the cleaned data for further use. The final visualization of the ETL stages provided a clear understanding of how data volume and quality are affected during the transformation process. This structured approach not only improves data reliability but also enhances the effectiveness of data visualization, ultimately enabling better business insights and informed decision-making.

In [18]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

Load Data

In [19]:
```
file_path = "Ecommerce+Sales+Data.xlsx"
excel_file = pd.ExcelFile(file_path)
```

Extract

In [20]:
```
sheet_name = excel_file.sheet_names[0]
df = excel_file.parse(sheet_name)
```

Transform

In [21]:
```
df['OrderDate'] = pd.to_datetime(df['OrderDate'], errors='coerce')
```

In [22]:
```
# Check missing values before transformation
nulls_before = df.isnull().sum()
rows_before = len(df)
nulls_before
```

Out[22]:
```
OrderDate            1
OrderNumber          1
ProductKey           1
SalespersonKey       1
Salesperson          1
Supervisor           1
Manager              1
Channel              1
Quantity             1
UnitPrice            1
Total             3015
Month                1
Year                 1
dtype: int64
```

In [23]:
```
# Drop rows with any nulls (cleaning)
df_cleaned = df.dropna()
df_cleaned
```

Out[23]:

	OrderDate	OrderNumber	ProductKey	SalespersonKey	Salesperson	Supervisor	Manager
0	2022-01-15	1492755.0	180.0	215.0	Carla Ferreira	Diego Araujo	Victor Castro
1	2022-01-20	1492618.0	661.0	215.0	Carla Ferreira	Diego Araujo	Victor Castro
2	2022-01-24	1492834.0	278.0	215.0	Carla Ferreira	Diego Araujo	Victor Castro
3	2022-01-25	1492801.0	1518.0	215.0	Carla Ferreira	Diego Araujo	Victor Castro
4	2022-02-01	1501692.0	661.0	215.0	Carla Ferreira	Diego Araujo	Victor Castro
...	...	...	...	...	...	...	...
38567	2020-10-22	2075402.0	157.0	215.0	Carla Ferreira	Diego Araujo	Victor Castro
38568	2020-10-23	2148884.0	157.0	215.0	Carla Ferreira	Diego Araujo	Victor Castro
38569	2020-10-24	2118590.0	157.0	215.0	Carla Ferreira	Diego Araujo	Victor Castro
38570	2020-10-25	2104650.0	157.0	215.0	Carla Ferreira	Diego Araujo	Victor Castro
38571	2020-10-26	2152594.0	157.0	215.0	Carla Ferreira	Diego Araujo	Victor Castro

38572 rows × 13 columns

In [24]:
```
nulls_after = df_cleaned.isnull().sum()
rows_after = len(df_cleaned)
rows_after
```

Out[24]:
```
38572
```

Load

In [25]:
```
df_cleaned.to_csv("Cleaned_Ecommerce_Data.csv", index=False)
```

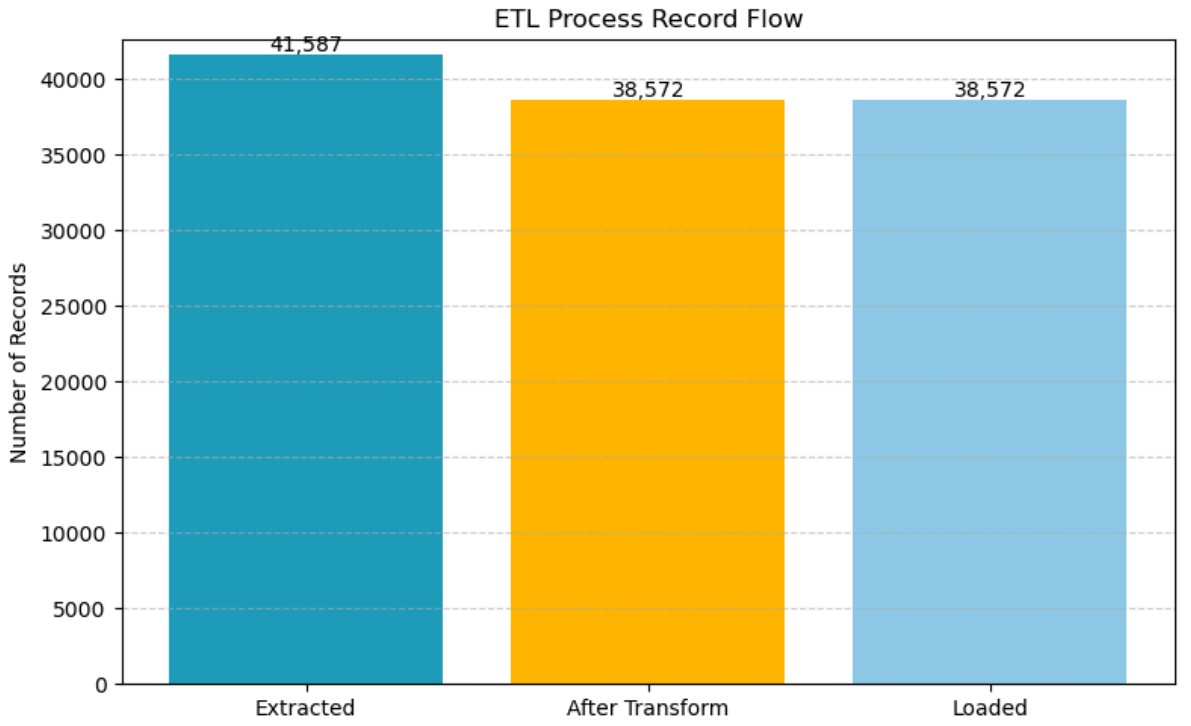
In [26]:
```
stages = ['Extracted', 'After Transform', 'Loaded']
record_counts = [rows_before, rows_after, rows_after]
colors = ['#219ebc', '#ffb703', '#8ecae6']
```

Visualization

In [27]:
```
plt.figure(figsize=(8, 5))
plt.bar(stages, record_counts, color=colors)
plt.title('ETL Process Record Flow')
plt.ylabel('Number of Records')
plt.grid(axis='y', linestyle='--', alpha=0.6)
plt.ylim(0, rows_before + 1000)

for i, count in enumerate(record_counts):
    plt.text(i, count + 200, f"{count:,}", ha='center', fontsize=10)
```

```
plt.tight_layout()
plt.show()
```



In [ ]:

ASSIGNMENT 4

AIM :

To perform data analysis and create insightful visualizations using advanced Excel features such as Pivot Tables, Charts, Conditional Formatting, thereby supporting decision-making in a business context.

PROBLEM STATEMENT :

A retail company collects monthly sales data across different regions and product categories. However, raw data alone does not provide meaningful insights. The company needs a way to analyze and visualize this data to identify sales trends, high-performing products, and regional performance. The goal is to help the management make informed decisions for inventory planning and marketing strategies.

THEORY :

Business Intelligence (BI) is a technology-driven process that helps organizations analyze data and present actionable information to aid in decision-making. One of the most accessible BI tools is **Microsoft Excel**, which, when used effectively, can transform raw data into meaningful insights.

Advanced Excel encompasses a suite of tools and features beyond basic spreadsheet functionality. These tools enable data analysts to perform complex calculations, manipulate large datasets, visualize data trends, and create interactive dashboards. Below are the key Advanced Excel concepts relevant to Business Intelligence:

1. **Pivot Tables**:
   Pivot Tables are powerful tools used to automatically sort, count, and total data stored in one table or spreadsheet and create a second table displaying the summarized data. With just a few clicks, they can reveal patterns and relationships that are not immediately obvious.
2. **Pivot Charts**:
   These are visual representations of Pivot Tables. They dynamically reflect changes in the Pivot Table and provide a graphical interface to explore patterns in the data.
3. **Conditional Formatting**:
   This feature highlights cells based on specific criteria (e.g., color scales,

data bars, or custom rules). It visually emphasizes high or low values, trends, and outliers, making it easier to interpret data at a glance.

4. **Data Validation**:
   Used to create dropdown lists or limit the type of data that can be entered in cells. This helps in standardizing data entry and preventing errors.
5. **Charts (Bar, Line, Pie, etc.)**:
   Excel supports various chart types that are useful for comparing data (bar), showing trends (line), or illustrating proportions (pie).
6. **Lookup Functions (VLOOKUP, HLOOKUP, INDEX-MATCH)**:
   These functions fetch data from different parts of a dataset or other sheets, allowing for flexible data integration and comparison.
7. **Dashboards**:
   A dashboard in Excel combines multiple charts, tables, KPIs, and filters into a single interactive sheet. It serves as a snapshot for decision-makers to assess performance at a glance.

By using these features together, users can derive insights from data and present them in a way that's easy to interpret and act upon.

PROCEDURE

Step 1: Importing the Data

- Open Microsoft Excel and import the dataset (e.g., monthly sales data).
- Ensure that the data is structured in tabular form with clearly labeled columns like Region, Product Category, Month, Units Sold, Sales, and Profit.

Step 2: Data Cleaning and Preparation

- Remove any duplicate records or empty rows.
- Standardize column formats (e.g., currency for Sales and Profit, text for Region).
- Use Excel's **Format as Table** option to apply structured formatting.

Step 3: Data Validation

- Apply dropdown menus for columns like Region and Product Category using **Data Validation > List**.
  This ensures consistency in data entry.

Step 4: Applying Conditional Formatting

- Highlight key metrics using color scales or rules:
  - Green for high profits
  - Red for very low sales
- Use **Data Bars** to show profit margins visually.

Step 5: Creating Pivot Tables

- Insert Pivot Tables to analyze:
  - Sales by Region
  - Monthly Profit Trends
  - Product Category performance
- Place the Pivot Tables on separate sheets for clarity.

Step 6: Visualizing the Data

- Create charts based on Pivot Table outputs:
  - **Bar Chart** for Regional Sales
  - **Line Chart** for Monthly Trends
  - **Pie Chart** for Category-wise Sales
- Customize with titles, data labels, and colors.

Step 7: Building the Dashboard

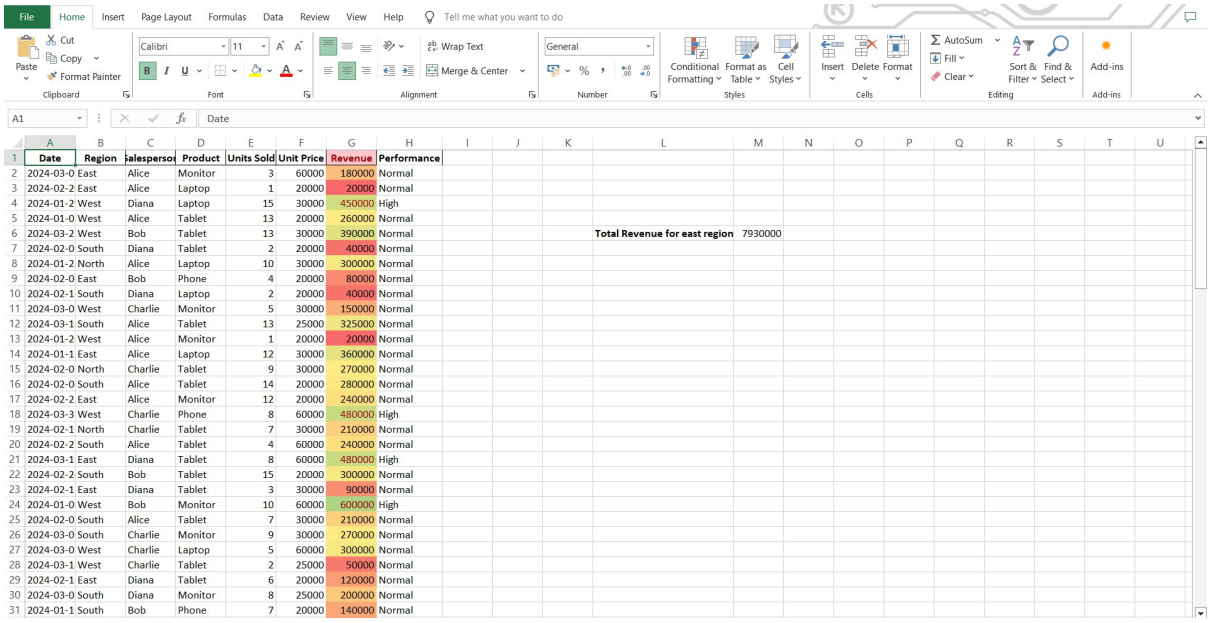
- Create a new sheet titled “Dashboard.”
- Arrange charts neatly with appropriate spacing.
- Optionally add slicers (if using Excel 2013 or later) to allow users to filter the dashboard dynamically by Region or Month.

Step 8: Drawing Insights

- Interpret the visualizations:
    - Identify which region performs best
    - Spot seasonal trends in monthly sales
    - See which product category generates the most revenue
- Note these observations for use in the final report or presentation.

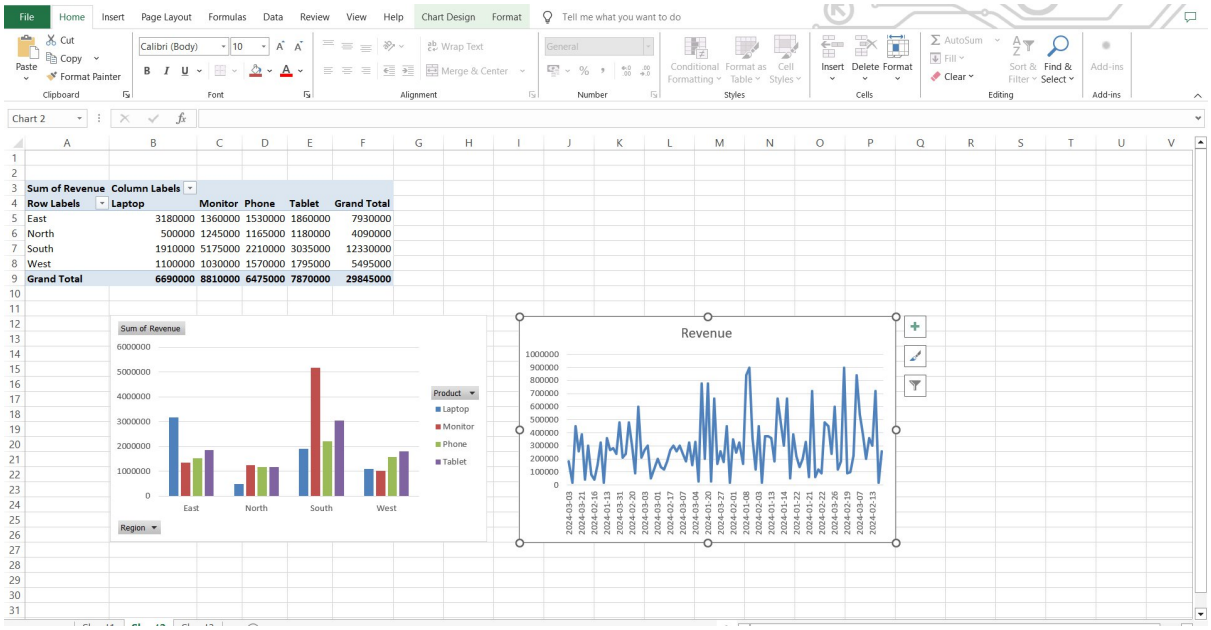
Step 9: Saving and Exporting

- Save the Excel workbook with all sheets: raw data, Pivot summaries, charts, and dashboard.
- Optionally export the dashboard as a PDF or image for sharing.

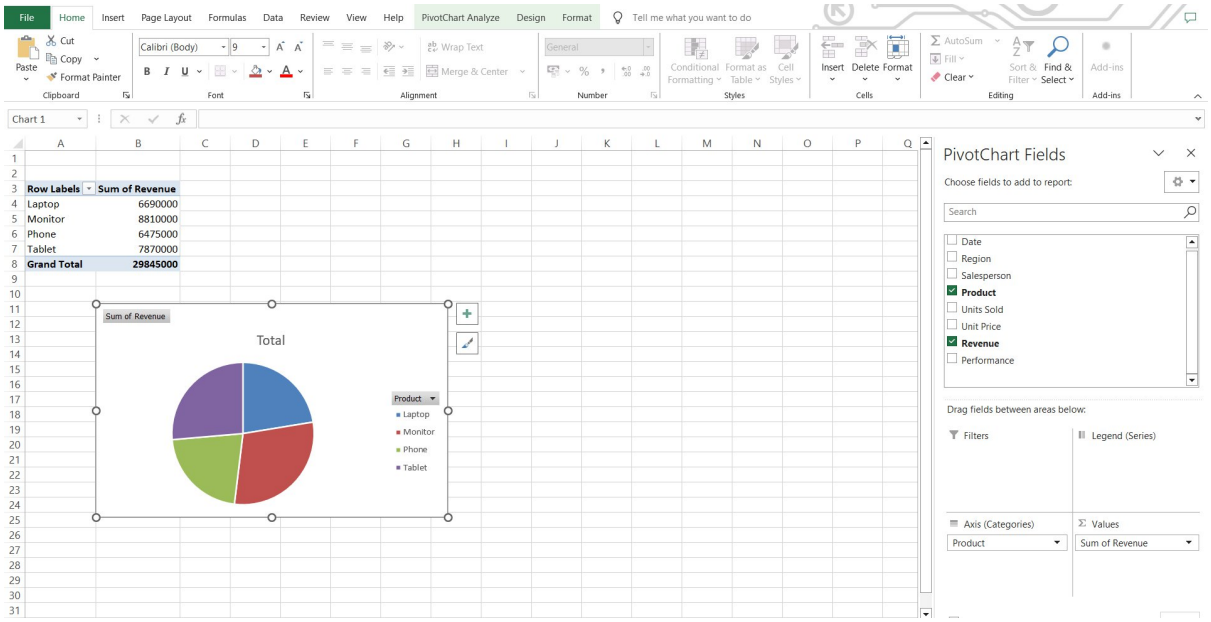


Date	Region	Salesperson	Product	Units Sold	Unit Price	Revenue	Performance
2024-03-0	East	Alice	Monitor	3	60000	180000	Normal
2024-02-2	East	Alice	Laptop	1	20000	20000	Normal
2024-01-2	West	Diana	Laptop	15	30000	450000	High
2024-01-0	West	Alice	Tablet	13	20000	260000	Normal
2024-03-2	West	Bob	Tablet	13	30000	390000	Normal
2024-02-0	South	Diana	Tablet	2	20000	40000	Normal
2024-01-2	North	Alice	Laptop	10	30000	300000	Normal
2024-02-0	East	Bob	Phone	4	20000	80000	Normal
2024-02-1	South	Diana	Laptop	2	20000	40000	Normal
2024-03-0	West	Charlie	Monitor	5	30000	150000	Normal
2024-03-1	South	Alice	Tablet	13	25000	325000	Normal
2024-01-2	West	Alice	Monitor	1	20000	20000	Normal
2024-01-1	East	Alice	Laptop	12	30000	360000	Normal
2024-02-0	North	Charlie	Tablet	9	30000	270000	Normal
2024-02-0	South	Alice	Tablet	14	20000	280000	Normal
2024-02-2	East	Alice	Monitor	12	20000	240000	Normal
2024-03-3	West	Charlie	Phone	8	60000	480000	High
2024-02-1	North	Charlie	Tablet	7	30000	210000	Normal
2024-02-2	South	Alice	Tablet	4	60000	240000	Normal
2024-03-1	East	Diana	Tablet	8	60000	480000	High
2024-02-2	South	Bob	Tablet	15	20000	300000	Normal
2024-02-1	East	Diana	Tablet	3	30000	90000	Normal
2024-01-0	West	Bob	Monitor	10	60000	600000	High
2024-02-0	South	Alice	Tablet	7	30000	210000	Normal
2024-03-0	South	Charlie	Monitor	9	30000	270000	Normal
2024-03-0	West	Charlie	Laptop	5	60000	300000	Normal
2024-03-1	West	Charlie	Tablet	2	25000	50000	Normal
2024-02-1	East	Diana	Tablet	6	20000	120000	Normal
2024-03-0	South	Diana	Monitor	8	25000	200000	Normal
2024-01-1	South	Bob	Phone	7	20000	140000	Normal

Total Revenue for east region 7930000



Sum of Revenue	Column Labels				
Row Labels	Laptop	Monitor	Phone	Tablet	Grand Total
East	3180000	1360000	1530000	1860000	7930000
North	500000	1245000	1165000	1180000	4090000
South	1910000	5175000	2210000	3035000	12330000
West	1100000	1030000	1570000	1795000	5495000
Grand Total	**6690000**	**8810000**	**6475000**	**7870000**	**29845000**


File
Home
Insert
Page Layout
Formulas
Data
Review
View
Help
PivotChart Analyze
Design
Format
Tell me what you want to do
Paste
Cut
Copy
Format Painter
Clipboard
Calibri (Body)
9
Font
Wrap Text
Merge & Center
Alignment
General
Number
Conditional Formatting
Format as Table
Cell Styles
Styles
Insert
Delete
Format
Cells
AutoSum
Fill
Clear
Sort & Filter
Find & Select
Editing
Add-ins
Add-ins
Chart 1
Row Labels
Sum of Revenue
Laptop 6690000
Monitor 8810000
Phone 6475000
Tablet 7870000
Grand Total 29845000
Sum of Revenue
Total
Product
Laptop
Monitor
Phone
Tablet
PivotChart Fields
Choose fields to add to report:
Search
Date
Region
Salesperson
Product
Units Sold
Unit Price
Revenue
Performance
Drag fields between areas below:
Filters
Legend (Series)
Axis (Categories)
Values
Product
Sum of Revenue

CONCLUSION

Using Advanced Excel for data analysis and visualization enables quick, accurate, and meaningful interpretation of business data. In this assignment, we used various Excel tools to transform raw sales data into actionable insights. The created dashboard and visualizations supported decision-making processes such as identifying top-performing regions and optimizing product strategies. Thus, Advanced Excel proves to be a powerful Business Intelligence tool in today's data-driven business environment.

ASSIGNMENT:- 5

AIM:-

Perform the data classification algorithm using any Classification algorithm.

PROBLEM STATEMENT:-

To use the Logistic Regression algorithm for supervised data classification. You have a dataset that contains demographic and travel-related information of Titanic passengers such as age, sex, fare, and passenger class. Apply the Logistic Regression algorithm to classify whether a passenger survived or not based on these features. The objective is to build a predictive model that identifies survival outcomes using labeled data and evaluates its performance using appropriate classification metrics.

THEORY:-

Classification

Classification is a **supervised machine learning** technique used to categorize data into predefined classes based on input features. In Business Intelligence (BI), classification is used to make informed decisions such as whether a customer will churn, whether a patient is at risk, or whether a transaction is fraudulent.

Unlike clustering (unsupervised), classification uses labeled training data to predict the category of new data points.

Types of Classification Algorithms

- **Logistic Regression** (linear classifier)
- **Decision Tree**
- **Random Forest**
- **Support Vector Machine (SVM)**
- **Naïve Bayes**
- **K-Nearest Neighbors (KNN)**

Each has its own strengths depending on the dataset and business objective.

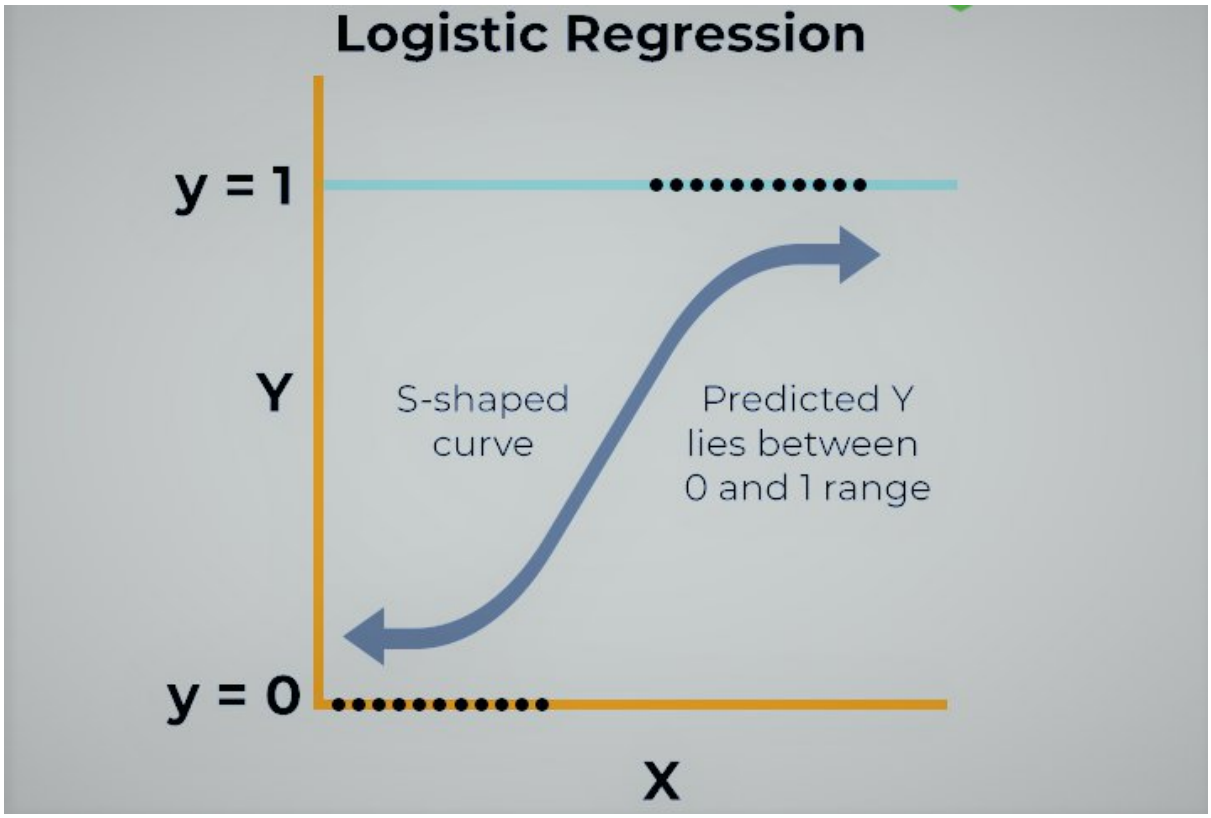
Logistic Regression Classification Algorithm

Logistic Regression is a statistical method commonly used for **binary classification**. It estimates the probability of a binary outcome (0 or 1) based on one or more predictor variables using the **sigmoid function**.

Sigmoid Function:

$$P(y=1) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n)}}$$

This output is interpreted as a probability, and a threshold (usually 0.5) is applied to convert it into class labels (0 or 1).



Working of Logistic Regression Algorithm

1. **Model Initialization:**
    - Assume coefficients (β) for each feature.
2. **Training the Model:**
    - Use a dataset with known labels.
    - Optimize the coefficients to minimize the error using **maximum likelihood estimation**.
3. **Prediction:**
    - For new data points, calculate the probability of the positive class using the sigmoid function.
4. **Classification:**
    - Assign label 1 if the predicted probability ≥ 0.5, otherwise 0.

Dataset Used: Titanic Survival Dataset

The Titanic dataset is a classic binary classification dataset used to predict **survival (1) or death (0)** of passengers aboard the Titanic.

Features:

- Age
- Sex
- Pclass (Passenger Class)
- Fare
- Family Size (engineered feature)

Target Variable:

- Survived (0 = No, 1 = Yes)

Data Preprocessing Steps

1. **Missing Value Handling:** Replace missing age values with the median.
2. **Feature Encoding:** Convert categorical features (e.g., Sex) into numerical.
3. **Feature Engineering:** Create new features such as family size from SibSp and Parch.
4. **Feature Scaling:** Standardize the numerical features.
5. **Train-Test Split:** Split data into training and testing sets for evaluation.

Evaluation Metrics for Classification

Metric	Meaning
Accuracy	Percentage of total predictions that are correct.
Precision	Out of all positive predictions, how many were correct.
Recall	Out of all actual positives, how many were correctly predicted.
F1-Score	Harmonic mean of precision and recall.
Confusion Matrix	Table showing TP, TN, FP, FN values.
ROC-AUC Score	Area under the ROC curve; higher values indicate better performance.

Visualization of Classification Results

1. **Correlation Heatmap:** Shows interdependencies among features.
2. **Survival Distribution (Bar Chart):** Highlights imbalance between survived and not.

3. **Survival by Gender (Bar Plot):** Helps understand gender influence on survival.
4. **Age vs Survival (Box Plot):** Insights into which age groups had higher survival.
5. **Fare vs Survival (Violin Plot):** Distribution of fare paid by survivors vs non-survivors.
6. **Pclass vs Survival (Stacked Bar Chart):** Shows class-wise survival percentage.

Applications of Logistic Regression in Business Intelligence

1. **Customer Retention:**
    - Predicting whether a customer is likely to churn.
2. **Credit Scoring:**
    - Determining the probability of loan default.
3. **Medical Diagnosis:**
    - Predicting the presence/absence of disease.
4. **Marketing Response:**
    - Classifying whether a user will click on an ad or not.
5. **Fraud Detection:**
    - Flagging suspicious transactions.

Advantages of Logistic Regression

- Simple and interpretable
- Fast training on large datasets
- Probabilistic output
- Good baseline for binary classification tasks

Limitations of Logistic Regression

- Assumes linear decision boundary
- Sensitive to outliers
- Not suitable for complex nonlinear relationships
- Requires feature scaling for better performance

Code:

jupyter Assignment_5_BI Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score, roc_curve, accuracy_score
from sklearn.preprocessing import StandardScaler, LabelEncoder
```

In [2]:
```
df = pd.read_csv("titanic.csv")
```

In [3]:
```
print(df.head())
```

```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

jupyter Assignment_5_BI Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [4]:
```
print(df.isnull().sum())
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [5]:
```
# Drop irrelevant features
df.drop(columns=['Name', 'PassengerId', 'Cabin', 'Ticket'], inplace=True)
```

In [6]:
```
# Fill missing Age with median, Embarked with mode
df['Age'].fillna(df['Age'].median(), inplace=True)
df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)
```

In [7]:
```
print(df.isnull().sum())
```

```
Survived    0
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
Embarked    0
dtype: int64
```

jupyter Assignment_5_BI Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [8]:
```
# Encode categorical variables
df['Sex'] = LabelEncoder().fit_transform(df['Sex'])  # male:0, female:1
df = pd.get_dummies(df, columns=['Embarked'], drop_first=True)
```

In [9]:
```
# Create family size feature
df['FamilySize'] = df['SibSp'] + df['Parch'] + 1

# Drop Parch and SibSp
df.drop(columns=['SibSp', 'Parch'], inplace=True)
```

In [10]:
```
df.head()
```

Out[10]:

	Survived	Pclass	Sex	Age	Fare	Embarked_Q	Embarked_S	FamilySize
0	0	3	1	22.0	7.2500	False	True	2
1	1	1	0	38.0	71.2833	False	False	2
2	1	3	0	26.0	7.9250	False	True	1
3	1	1	0	35.0	53.1000	False	True	2
4	0	3	1	35.0	8.0500	False	True	1

In [11]:
```
#Feature Scaling
scaler = StandardScaler()
scaled_features = scaler.fit_transform(df.drop('Survived', axis=1))
X = pd.DataFrame(scaled_features, columns=df.drop('Survived', axis=1).columns)
y = df['Survived']
```

In [12]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

jupyter Assignment_5_BI Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [13]:
```
model = LogisticRegression()
params = {
    'C': [0.01, 0.1, 1, 10],
    'solver': ['liblinear', 'lbfgs']
}
grid = GridSearchCV(model, param_grid=params, cv=5, scoring='accuracy')
grid.fit(X_train, y_train)

best_model = grid.best_estimator_
```

In [14]:
```
y_pred = best_model.predict(X_test)

print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
Confusion Matrix:
 [[136  21]
 [ 31  80]]

Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.87      0.84       157
           1       0.79      0.72      0.75       111

    accuracy                           0.81       268
   macro avg       0.80      0.79      0.80       268
weighted avg       0.81      0.81      0.80       268

Accuracy: 0.8059701492537313
```

jupyter Assignment_5_BI Last Checkpoint: an hour ago (unsaved changes)

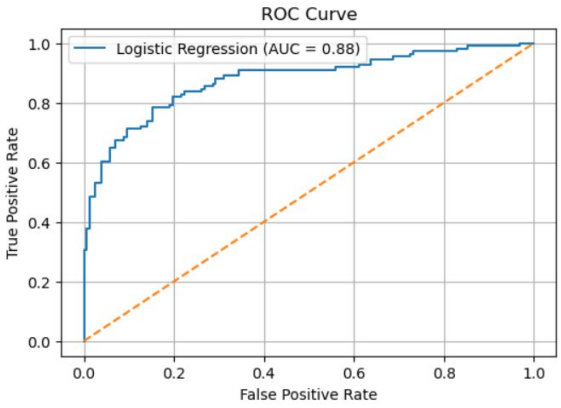
Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [21]:
```
y_probs = best_model.predict_proba(X_test)[:,1]
auc_score = roc_auc_score(y_test, y_probs)
fpr, tpr, thresholds = roc_curve(y_test, y_probs)
plt.figure(figsize=(6,4))
plt.plot(fpr, tpr, label=f"Logistic Regression (AUC = {auc_score:.2f})")
plt.plot([0, 1], [0, 1], linestyle='--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.grid()
plt.show()
```



jupyter Assignment_5_BI Last Checkpoint: an hour ago (unsaved changes)

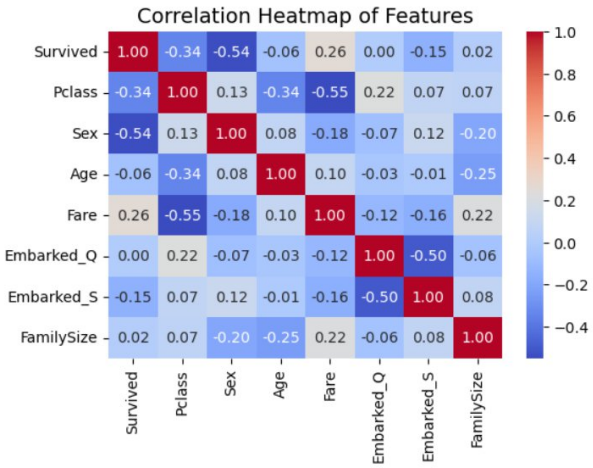
Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [24]:
```
plt.figure(figsize=(6, 4))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title("Correlation Heatmap of Features", fontsize=14)
plt.show()
```



jupyter Assignment_5_BI Last Checkpoint: an hour ago (autosaved)

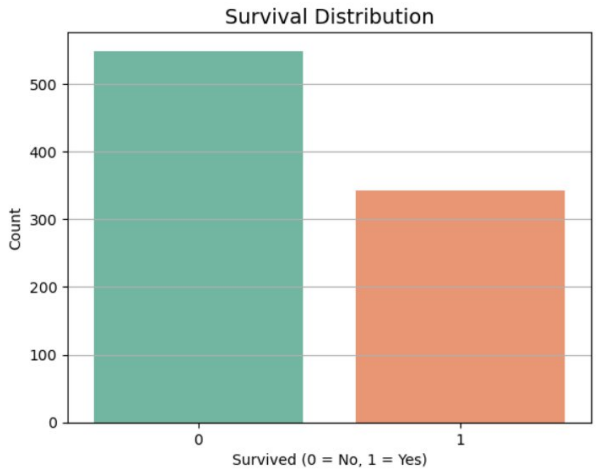
Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [17]:
```
sns.countplot(x='Survived', data=df, palette='Set2')
plt.title("Survival Distribution", fontsize=14)
plt.xlabel("Survived (0 = No, 1 = Yes)")
plt.ylabel("Count")
plt.grid(axis='y')
plt.show()
```



jupyter Assignment_5_BI Last Checkpoint: an hour ago (autosaved)

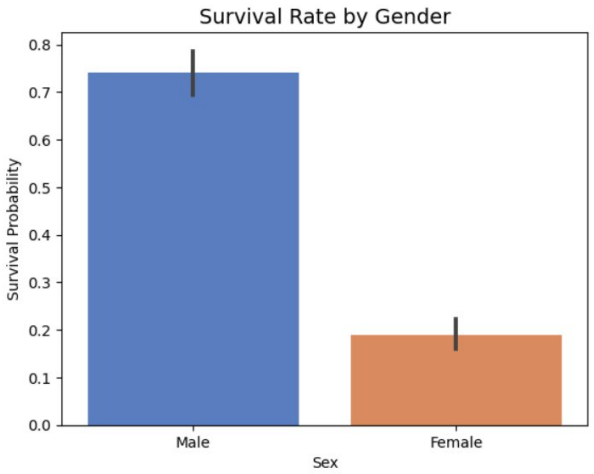
Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [18]:
```
sns.barplot(x='Sex', y='Survived', data=df, palette='muted')
plt.xticks([0, 1], ['Male', 'Female'])
plt.title("Survival Rate by Gender", fontsize=14)
plt.ylabel("Survival Probability")
plt.show()
```



jupyter Assignment_5_BI Last Checkpoint: an hour ago (unsaved changes)

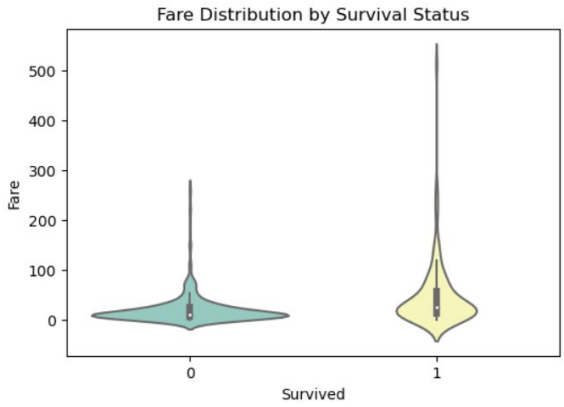
Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [25]:
```
plt.figure(figsize=(6, 4))
sns.violinplot(x='Survived', y='Fare', data=df, palette='Set3')
plt.title("Fare Distribution by Survival Status")
plt.xlabel("Survived")
plt.ylabel("Fare")
plt.show()
```



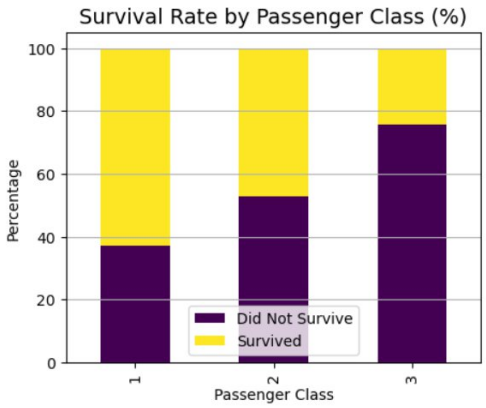
jupyter Assignment_5_BI Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

In [27]:
```
ct = pd.crosstab(df['Pclass'], df['Survived'], normalize='index') * 100
ct.plot(kind='bar', stacked=True, colormap='viridis', figsize=(5, 4))
plt.title("Survival Rate by Passenger Class (%)", fontsize=14)
plt.ylabel("Percentage")
plt.xlabel("Passenger Class")
plt.legend(["Did Not Survive", "Survived"])
plt.grid(axis='y')
plt.show()
```



CONCLUSION

Logistic Regression is a powerful binary classification algorithm, especially useful in BI applications like risk assessment, healthcare analytics, and marketing. When applied to the Titanic dataset, it accurately predicts survival based on multiple passenger attributes. With proper preprocessing, feature engineering, and evaluation, Logistic Regression provides valuable insights for data-driven business decisions.

ASSIGNMENT:- 6

AIM:- Perform the data clustering algorithm using any Clustering algorithm

PROBLEM STATEMENT:- To use the K-Means Clustering algorithm for unsupervised data segmentation. You have a dataset that contains measurements of iris flowers (sepal length, sepal width, petal length, and petal width). Apply the K-Means algorithm to group the flowers into clusters based on feature similarities, so that the natural structure of the data can be identified and different species of iris can be distinguished without using their actual labels.

THOERY:-

Clustering

Clustering is an unsupervised machine learning technique used to group similar data points together based on features or properties. The aim is to ensure that objects within a cluster are more similar to each other than to those in other clusters.
Types of Clustering Algorithms are:

- **Partition-based Clustering** (e.g., K-Means, K-Medoids)
- **Density-based Clustering** (e.g., DBSCAN)
- **Hierarchical Clustering** (e.g., Agglomerative & Divisive)
- **Model-based Clustering** (e.g., Gaussian Mixture Models)

K-Means Clustering Algorithm

K-Means is a centroid-based algorithm and one of the most popular clustering techniques. It partitions the dataset into K distinct, non-overlapping clusters based on distance from the cluster centroid.

How K-Means Clustering Algorithm Works

The steps involved in K-Means Clustering Algorithm include:

1. **Initialization**:
   - Choose the number of clusters K.
   - Randomly initialize K centroids.
2. **Assignment Step**:
   - Assign each data point to the nearest centroid using a distance metric (usually Euclidean distance).
   - This forms K clusters.
3. **Update Step**:
   - Recalculate the centroid (mean) of each cluster based on the new members.
4. **Repeat** steps 2 and 3 until:

- Centroids do not change significantly.
- A maximum number of iterations is reached.
- Convergence criteria is met.

Distance Metric Used

Euclidean Distance:
For points $x = (x_1, x_2, ..., x_n)$ and $y = (y_1, y_2, ..., y_n)$,

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

How to Choose K (No. of Clusters)

- Elbow Method: Plot the Within-Cluster-Sum of Squared Errors (WCSS) against various K values and choose the "elbow point".
- Silhouette Score: Measures how well each point fits in its cluster.

Applications of K-Means Clustering on the Iris Dataset

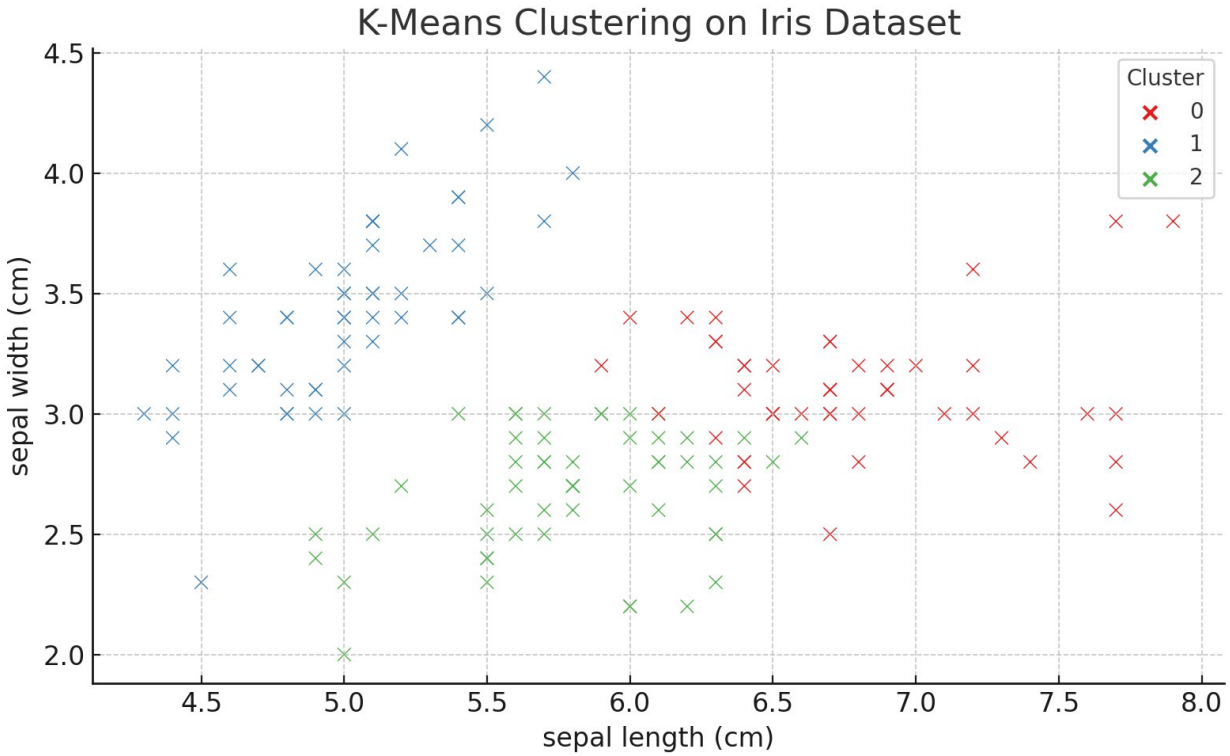
The Iris dataset contains 150 samples of iris flowers with 4 features: sepal length, sepal width, petal length, and petal width. It includes 3 species: Setosa, Versicolor, and Virginica. This dataset is commonly used for classification, clustering, and pattern recognition tasks.

1. **Unsupervised Pattern Discovery:**
   K-Means helps identify natural groupings of iris flowers based on their physical attributes (sepal & petal length/width), even without knowing the species.
2. **Dimensionality Reduction & Visualization:**
   Clustering helps visualize high-dimensional biological data in 2D using only two features.
3. **Data Compression:**
   In applications like image compression or biological image analysis, flowers can be represented using cluster centroids instead of raw data.
4. **Exploratory Data Analysis (EDA):**
   Before applying classification, K-Means helps in initial understanding of how well data points can be grouped.

Visualization of K-Means clustering

The **visualization of K-Means clustering** on the Iris dataset :

- The plot shows how the flowers are **grouped into 3 clusters** using the first two features (sepal length and sepal width).
- Each color represents a **different cluster** discovered by the algorithm.
- This helps us visually understand the **natural separation** between different types of flowers.



Advantages of K-Means Clustering

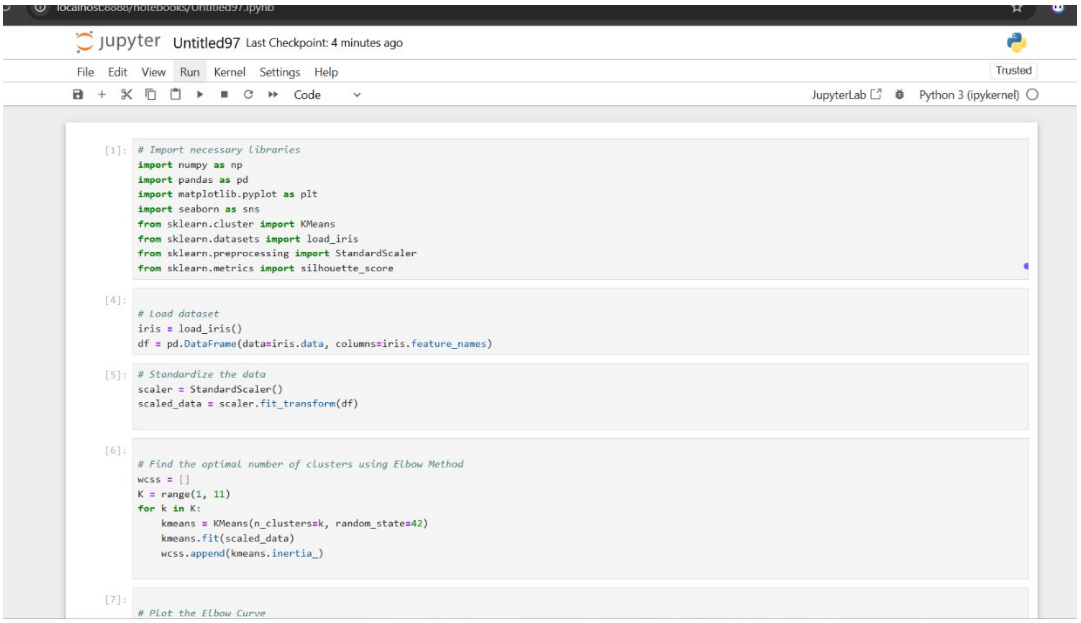
1. **Simple and Fast:** Easy to implement and computationally efficient for large datasets.
2. **Scalable:** Performs well with large datasets due to its linear time complexity.
3. **Unsupervised Learning:** Does not require labeled data, making it ideal for exploratory analysis.
4. **Easily Adaptable:** Works well with various types of numerical data and can be modified for different objectives.

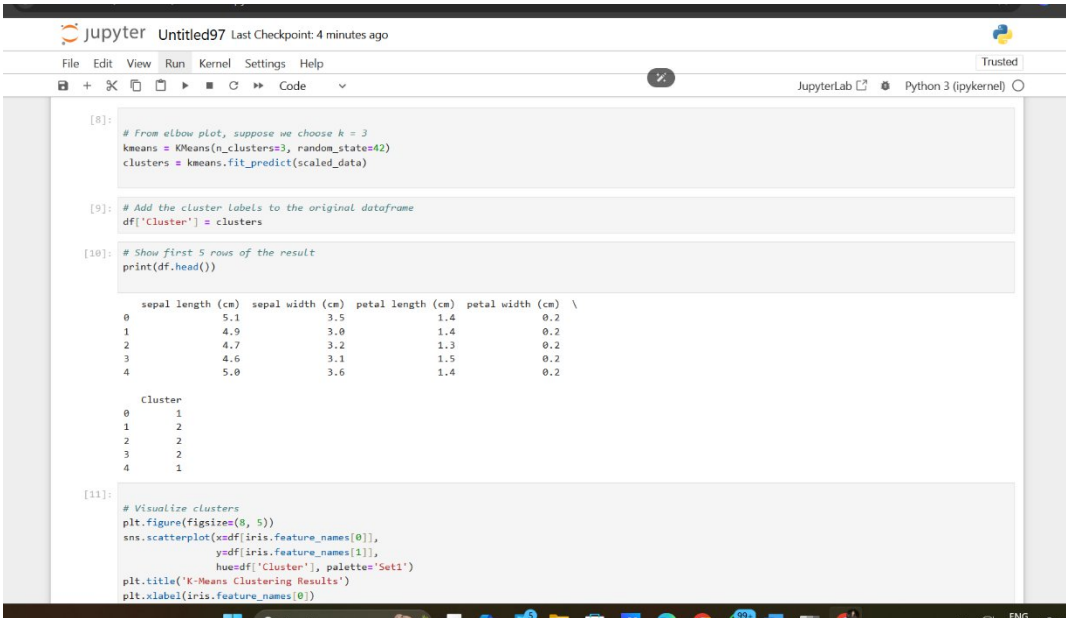
Limitations of K-Means Clustering

1. **Assumes Spherical Clusters:** Works best when clusters are roughly circular in shape and size.
2. **Sensitive to Initial Centroids**: Different initializations may lead to different results.

3. **Needs Predefined K:** You must specify the number of clusters (K) beforehand, which may not be known.
4. **Not Ideal for Outliers:** K-Means is sensitive to outliers which can distort the cluster centers.

CODE:-


jupyter Untitled97 Last Checkpoint: 4 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
[1]:
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
[4]:
# Load dataset
iris = load_iris()
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
[5]:
# Standardize the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)
[6]:
# Find the optimal number of clusters using Elbow Method
wcss = []
K = range(1, 11)
for k in K:
kmeans = KMeans(n_clusters=k, random_state=42)
kmeans.fit(scaled_data)
wcss.append(kmeans.inertia_)
[7]:
Plot the Elbow Curve


jupyter Untitled97 Last Checkpoint: 4 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
[8]:
# From elbow plot, suppose we choose k = 3
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(scaled_data)
[9]:
Add the cluster labels to the original dataframe
df['Cluster'] = clusters
[10]:
# Show first 5 rows of the result
print(df.head())
sepal length (cm) sepal width (cm) petal length (cm) petal width (cm) \
0 5.1 3.5 1.4 0.2
1 4.9 3.0 1.4 0.2
2 4.7 3.2 1.3 0.2
3 4.6 3.1 1.5 0.2
4 5.0 3.6 1.4 0.2
Cluster
0 1
1 2
2 2
3 2
4 1
[11]:
# Visualize clusters
plt.figure(figsize=(8, 5))
sns.scatterplot(x=df[iris.feature_names[0]],
y=df[iris.feature_names[1]],
hue=df['Cluster'], palette='Set1')
plt.title('K-Means Clustering Results')
plt.xlabel(iris.feature_names[0])

localhost:8888/notebooks/Untitled97.ipynb

jupyter Untitled97 Last Checkpoint: 5 minutes ago

File Edit View Run Kernel Settings Help

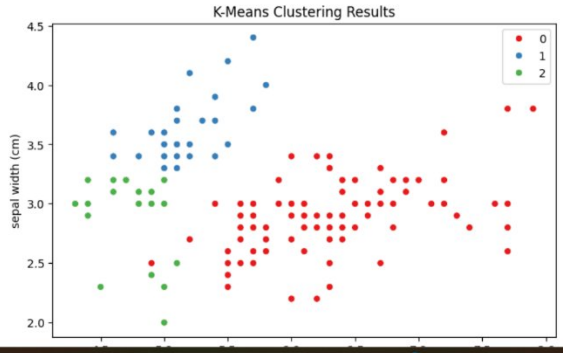
Trusted

Code

JupyterLab Python 3 (ipykernel)

[11]:

```
# Visualize clusters
plt.figure(figsize=(8, 5))
sns.scatterplot(x=df[iris.feature_names[0]],
                y=df[iris.feature_names[1]],
                hue=df['Cluster'], palette='Set1')
plt.title('K-Means Clustering Results')
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])
plt.legend()
plt.show()
```



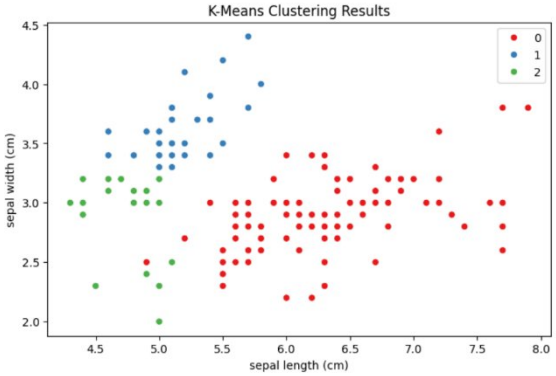
jupyter Untitled97 Last Checkpoint: 5 minutes ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)



[12]:

```
# Calculate silhouette score
score = silhouette_score(scaled_data, clusters)
print("Silhouette Score:", score)
```

```
Silhouette Score: 0.4798814508199817
```

CONCLUSION:- K-Means is a powerful and intuitive clustering algorithm used widely for grouping similar data points. On the Iris dataset, it successfully classifies flowers into distinct groups based on their features, showcasing the algorithm's practical utility in unsupervised learning tasks. However, one must carefully choose the number of clusters and preprocess the data properly to achieve optimal results.