AISSMS
INSTITUTE OF INFORMATION TECHNOLOGY
(IOIT)
ADDING VALUE TO ENGINEERING

Department of Artificial Intelligence & Data Science

Academic year-2023-24

Lab Manual

Computer Laboratory IV

BE
AI & DS

Semester VIII

Prepared by, Mr.
G. H. Wani.

Department of Artificial Intelligence and Data Science

Vision

To be amongst the top artificial intelligent & data science programs for catering to the changing needs of the industry and society.

Mission

- To foster an environment to provide intelligent solutions applicable for multidisciplinary needs of industry & society.
- To promote career development with ethical responsibility.

Program Education Objectives(PEOs)

PEO1: Graduates will be able to analyze, formulate and function efficiently in a multi-disciplinary context to address industrial problems.

PEO2: Graduates will be able to work collaboratively with professionalism and ethical responsibilities to provide innovative needs of societies.

PEO3: Graduates will excel in their careers by adapting to new technologies.

Program Specific Outcomes (PSOs)

PSO1 Problem Solving and Programming Skills: Graduates will be able to apply programming skills to identify, modify and test algorithms that apply intelligence to make realistic decisions in problem solving.

PSO2 Professional Skills: Graduates will be able to collect, analyze, interpret and visualize data to solve problems in agriculture, automation, finance and medical domains.

Program Outcomes (POs)

Graduates will be able to

1. Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems. **[Engineering knowledge]**
2. Identify, formulate, research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences. **[Problem analysis]**
3. Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations. **[Design/development of solutions]**
4. Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions. **[Conduct investigations of complex problems]**
5. Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations. **[Modern tool usage]**
6. Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice. **[The engineer and society]**
7. Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development. **[Environment and sustainability]**
8. Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice. **[Ethics]**
9. Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings. **[Individual and team work]**
10. Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions. **[Communication]**
11. Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments. **[Project management and finance]**
12. Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change. **[Life-long learning]**

Index

ASSIGNMENT 1

TITLE: Import Data from different Sources such as (Excel, SQL Server, Oracle etc.) and load in targeted system.

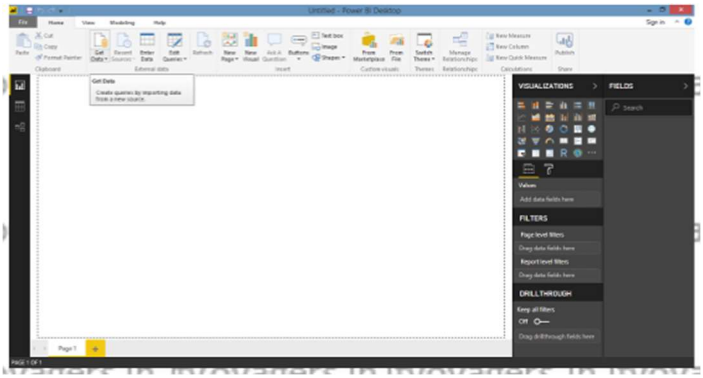
AIM: To import Data from different Sources such as (Excel, SQL Server, Oracle etc.) and load in targeted system.

OBJECTIVES: Students will be able to import data from different sources and load it in targeted systems.

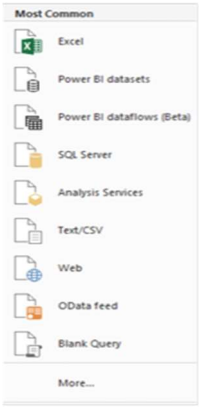
THEORY:

STEPS:

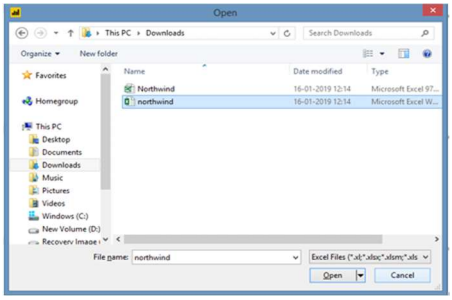
Step 1: Open Power BI



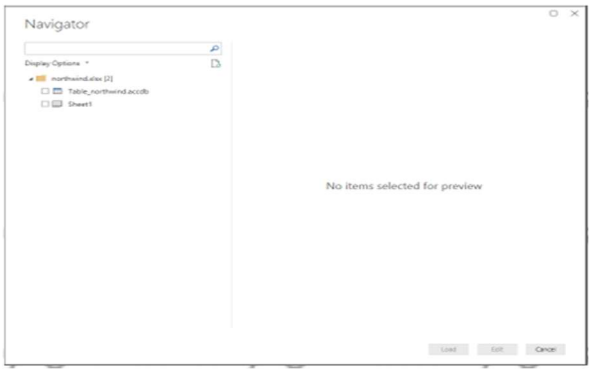
Step 2: Click on Get data following list will be displayed → select Excel

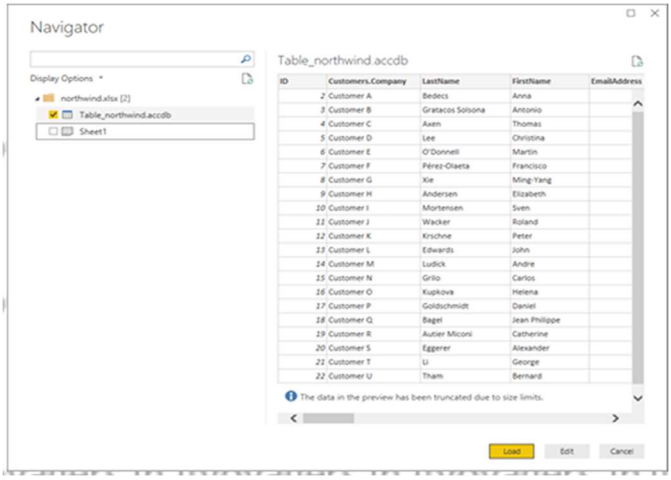


Step 3: Select required file and click on Open, Navigator screen appears

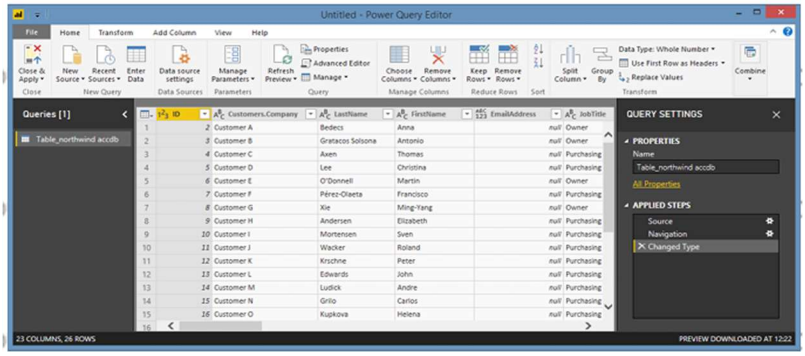


Step 4: Select file and click on edit



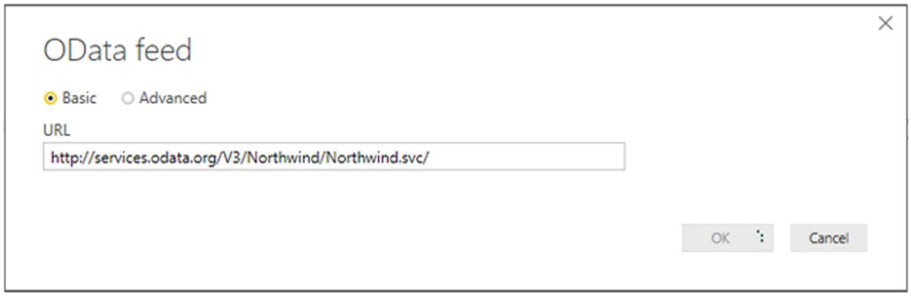


Step 5: Power query editor appears



Step 6: Again, go to Get Data and select OData feed

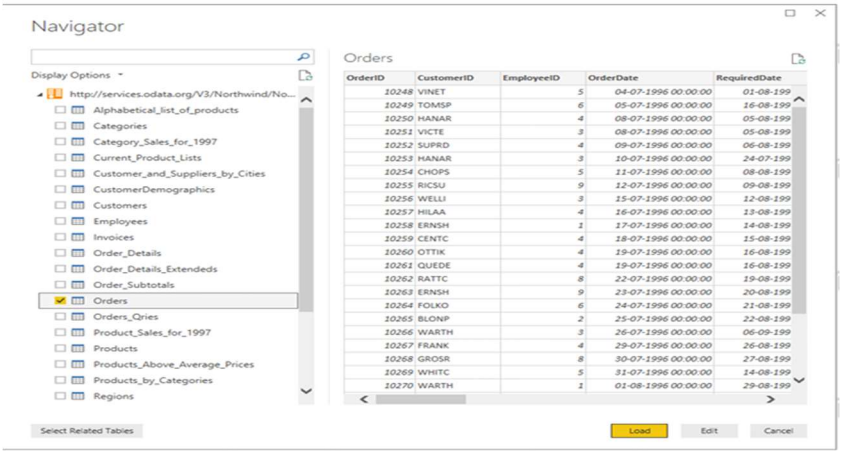
Step 7: Paste url as
http://services.odata.org/V3/Northwind/Northwind.svc/
Click on ok

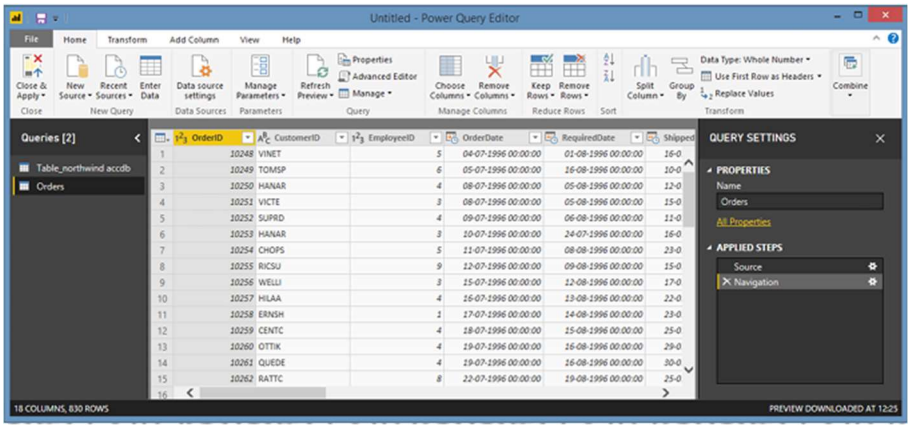


Step 8: Select orders table and click on edit

Note: If you just want to see preview you can just click on table name without clicking on checkbox

Click on edit to view table





CONCLUSION:

Hence, we understood how to import data from different sources and load it in targeted systems.

ASSIGNMENT 2

TITLE: Perform the Extraction Transformation and Loading (ETL) process to construct the database in the SQL server / Power BI.

AIM: To perform the Extraction Transformation and Loading (ETL) process to construct the database in the SQL server / Power BI.

OBJECTIVES : Students will be able to understand the Extraction Transformation and Loading (ETL) process to construct the database

THEORY:

ETL stands for Extract, Transform, Load and it is a process used in data warehousing to extract data from various sources, transform it into a format suitable for loading into a data warehouse, and then load it into the warehouse. The process of ETL can be broken down into the following three stages:

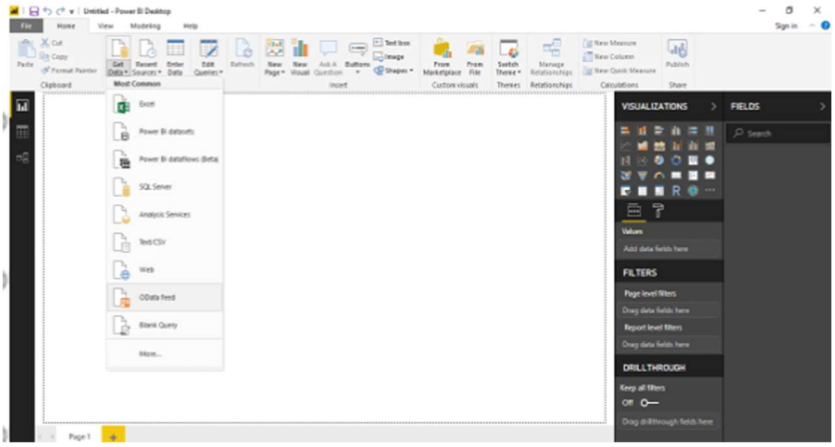
1. **Extract**: The first stage in the ETL process is to extract data from various sources such as transactional systems, spreadsheets, and flat files. This step involves reading data from the source systems and storing it in a staging area.
2. **Transform**: In this stage, the extracted data is transformed into a format that is suitable for loading into the data warehouse. This may involve cleaning and validating the data, converting data types, combining data from multiple sources, and creating new data fields.
3. **Load**: After the data is transformed, it is loaded into the data warehouse. This step involves creating the physical data structures and loading the data into the warehouse.

The ETL process is an iterative process that is repeated as new data is added to the warehouse. The process is important because it ensures that the data in the data warehouse is accurate, complete, and up-to-date. It also helps to ensure that the data is in the format required for data mining and reporting.

ETL Tools: Most commonly used ETL tools are Hevo, Sybase, Oracle Warehouse builder, CloverETL, and MarkLogic.

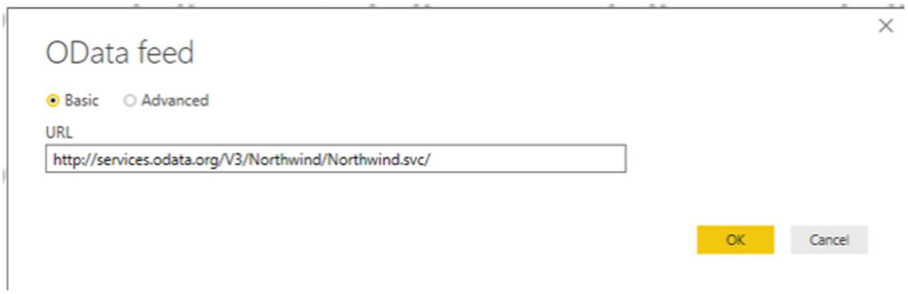
STEPS:

Step 1: Open Power BI, Click on Get Data → OData Feed

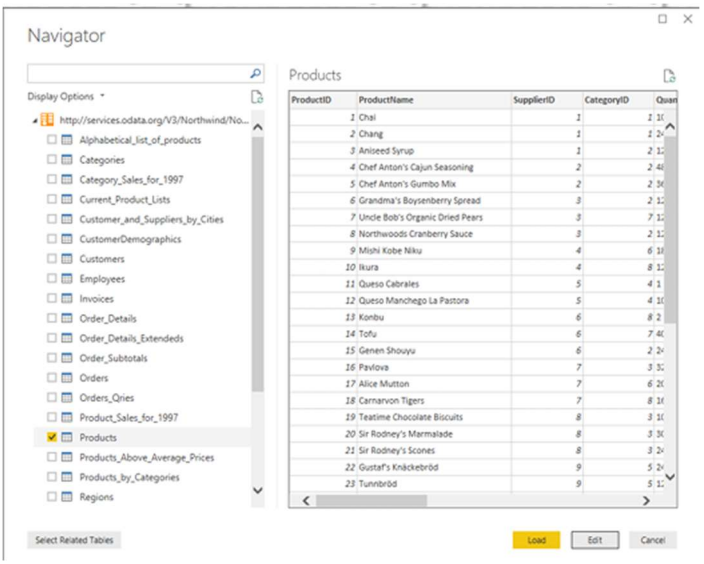


Paste Url : http://services.odata.org/V3/Northwind/Northwind.svc/

And Click OK

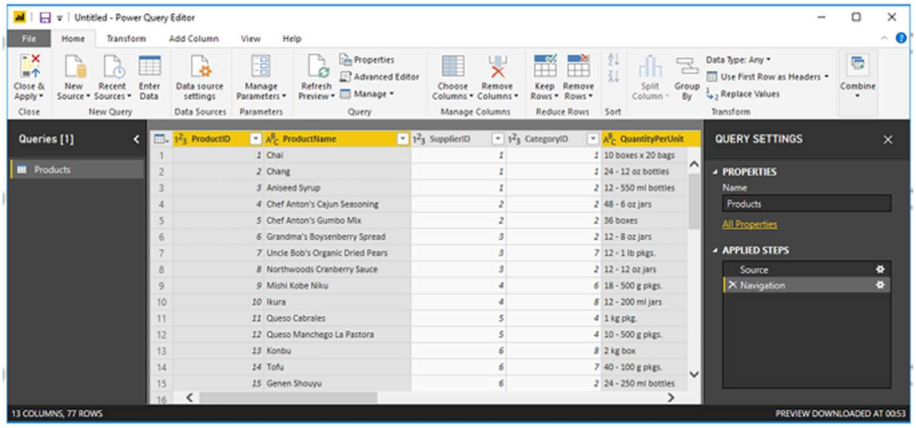


Step 2: Click on Check Box of Products table and then click on Edit

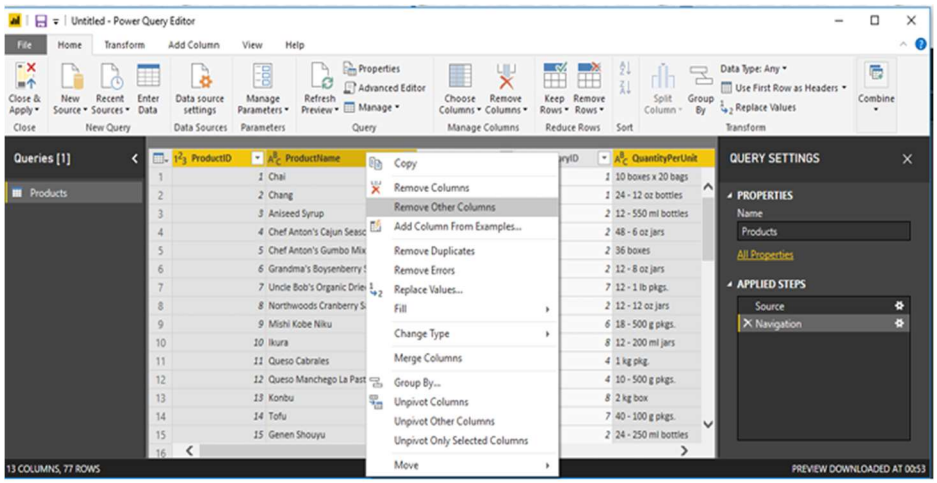


1) Remove other columns to only display columns of interest

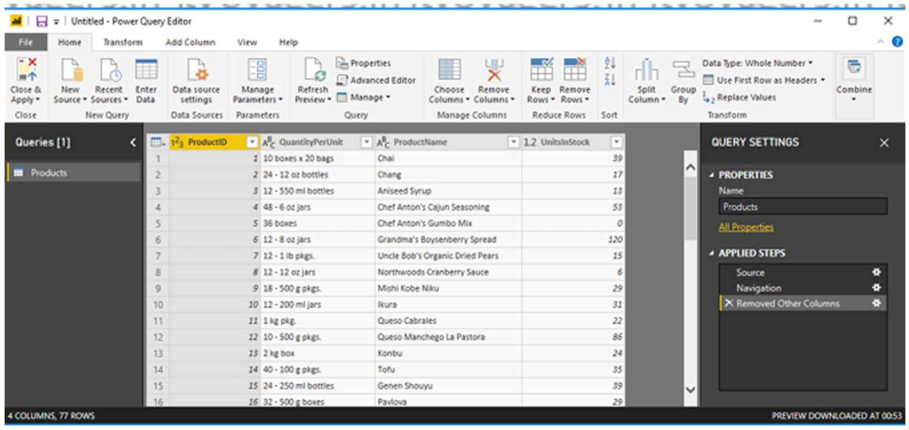
In Query Editor, select the ProductID, ProductName, QuantityPerUnit, and UnitsInStock columns (use Ctrl+Click to select more than one column, or Shift+Click to select columns that are beside each other).



Select Remove Columns > Remove Other Columns from the ribbon, or right click on a column header and click Remove Other Columns

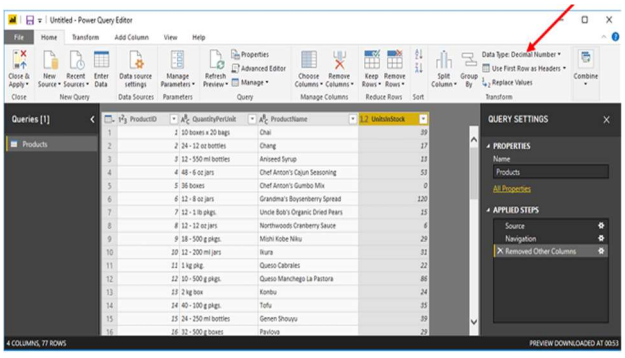


After selecting Remove Other Columns only selected four columns are displayed other columns are discarded.



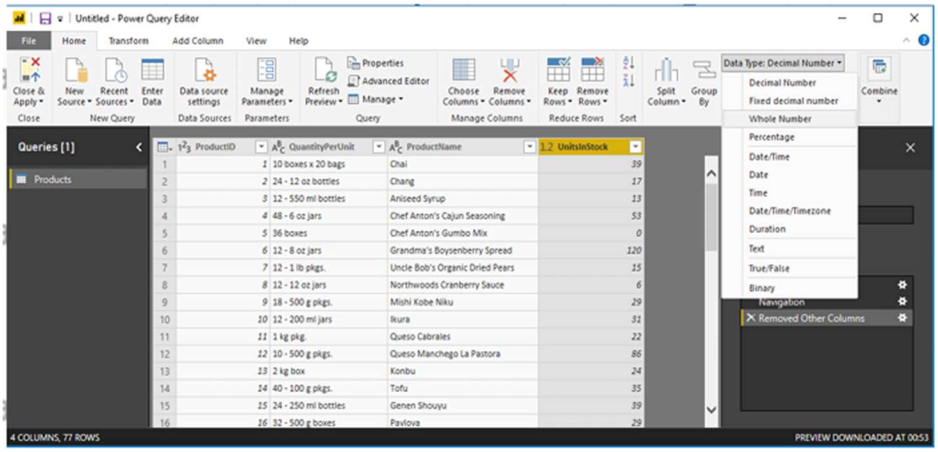
Change the data type of the UnitsInStock column

a) Select the UnitsInStock column.

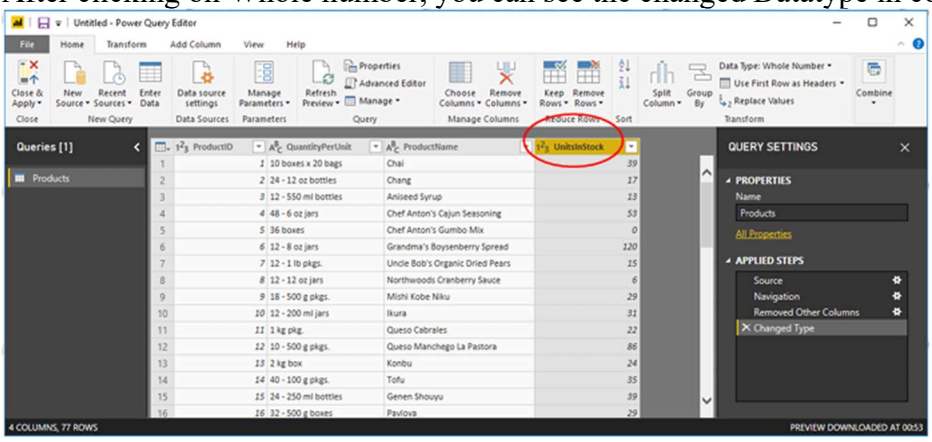


b) Select the Data Type drop-down button in the Home ribbon.

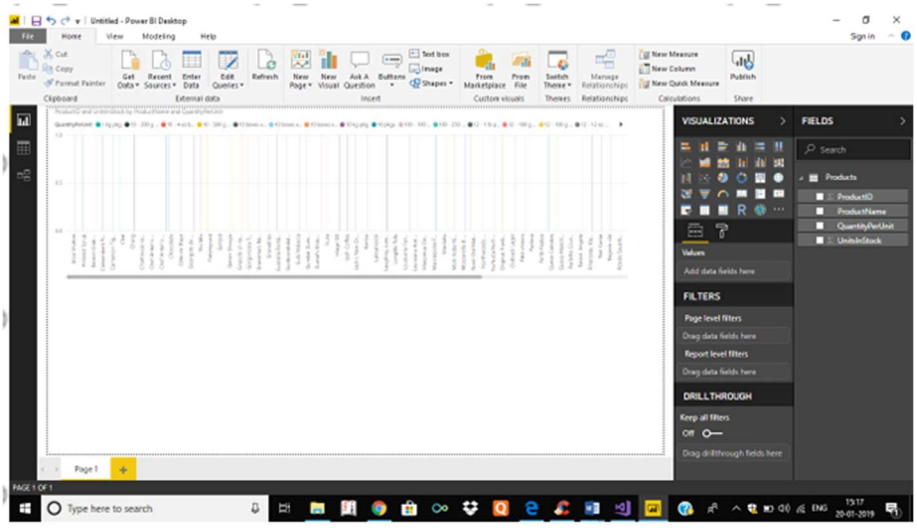
c) If not already a Whole Number, select Whole Number for data type from the drop down (the Data Type: button also displays the data type for the current selection).



After clicking on Whole number, you can see the changed Datatype in column header of UnitsInStock

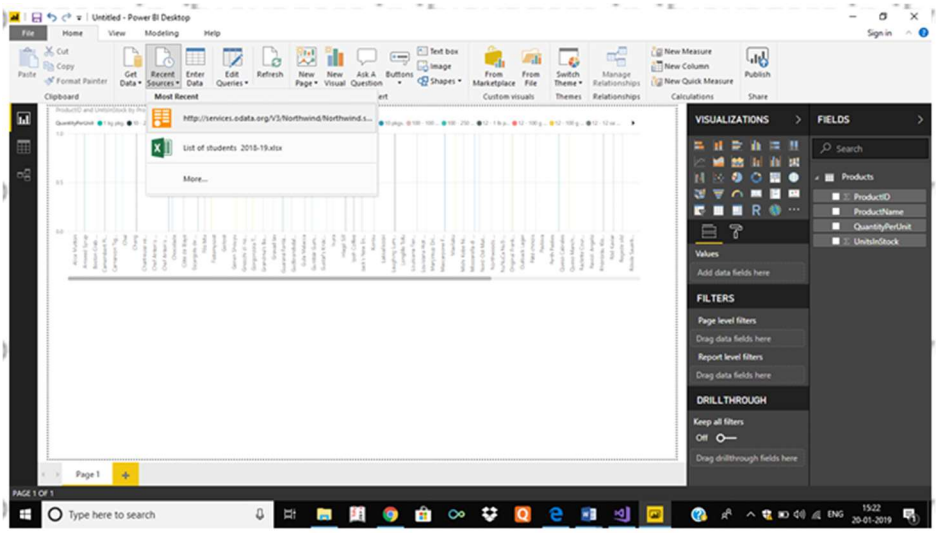


After above step, close query editor and click on Yes to save changes.
Now you can view fields of Products table on right side, check all the fields of table to get representation in charts form.

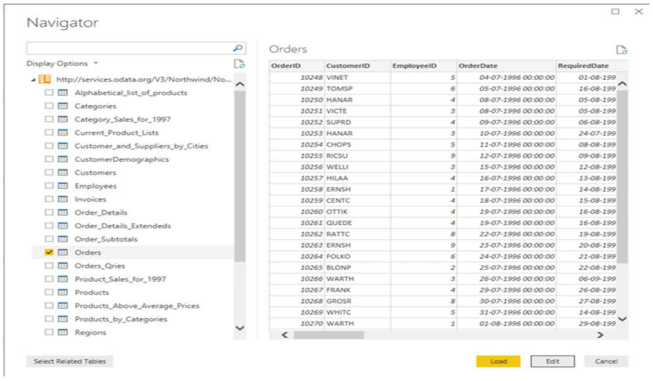


3. Expand the Orders table

Once You have loaded a data source, you can click on Recent Sources to select desired table (Orders).

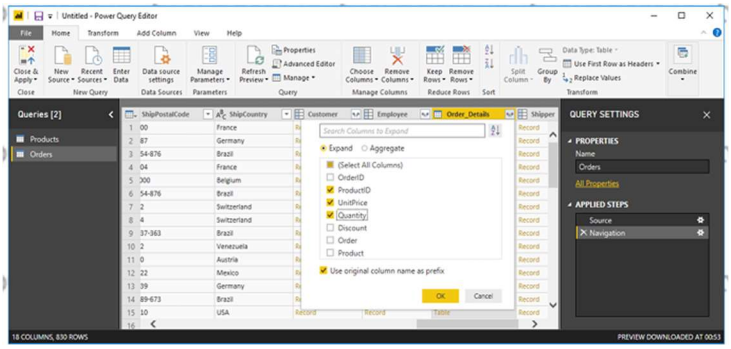


After selecting the URL, Navigator window will appear from which you can select Orders table. Click on Edit.

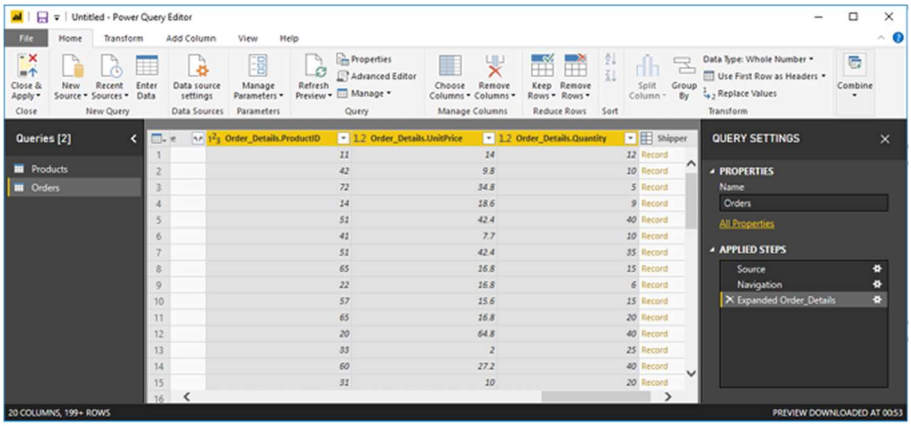


Query Editor Window will appear

1. In the Query View, scroll to the Order_Details column.
2. In the Order_Details column, select the expand icon.
3. In the Expand drop-down:
    a. Select (Select All Columns) to clear all columns.
    b. Select ProductID, UnitPrice, and Quantity.
    c. Click OK.



After clicking on OK following screen appears with combined columns

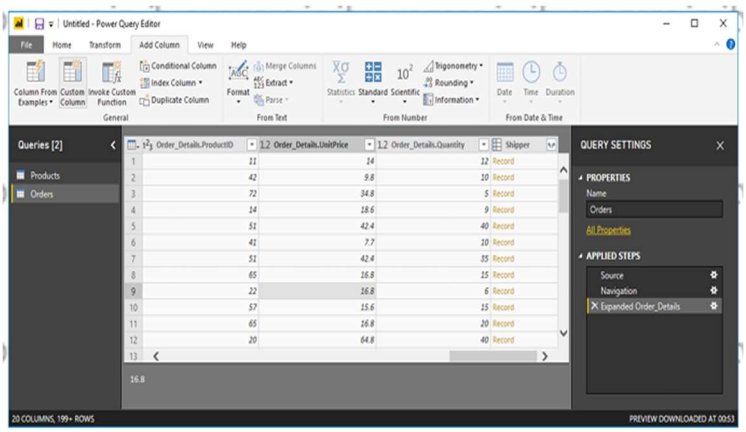


4. Calculate the line total for each Order_Details row

Power BI Desktop lets you to create calculations based on the columns you are importing, so you can enrich the data that you connect to. In this step, you create a Custom Column to calculate the line total for each Order_Details row.

Calculate the line total for each Order_Details row:

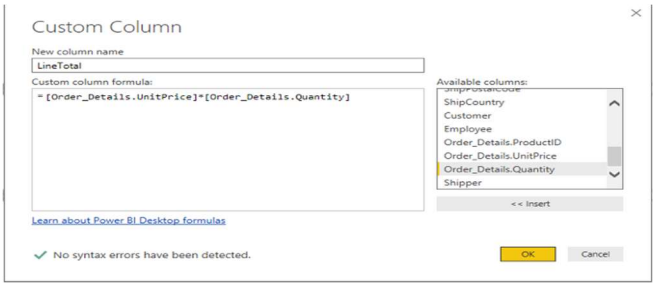
a) In the Add Column ribbon tab, click Add Custom Column.

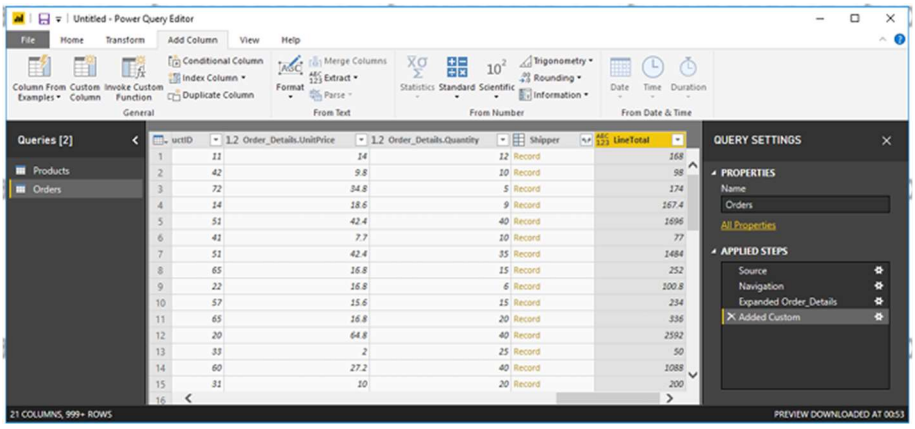


b) In the Custom Column dialog box, in the Custom Column Formula textbox, enter [Order_Details.UnitPrice] * [Order_Details.Quantity] by selecting from available columns and click on insert for each column.

c) In the New column name textbox, enter LineTotal.

d) Click OK.

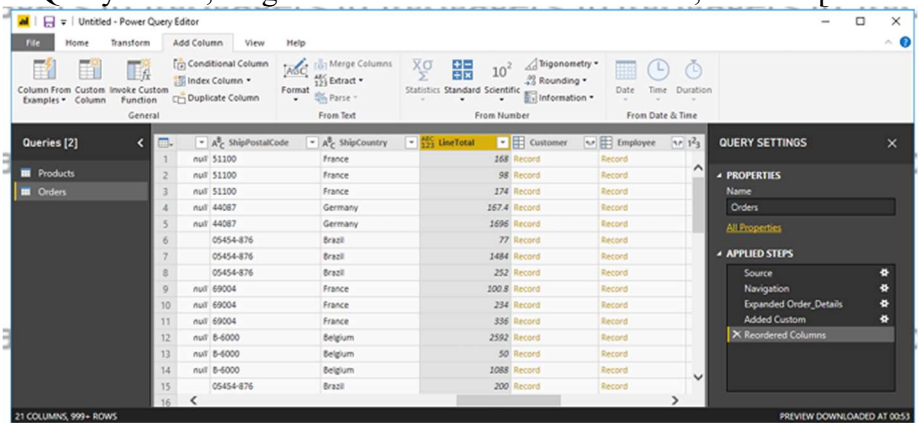




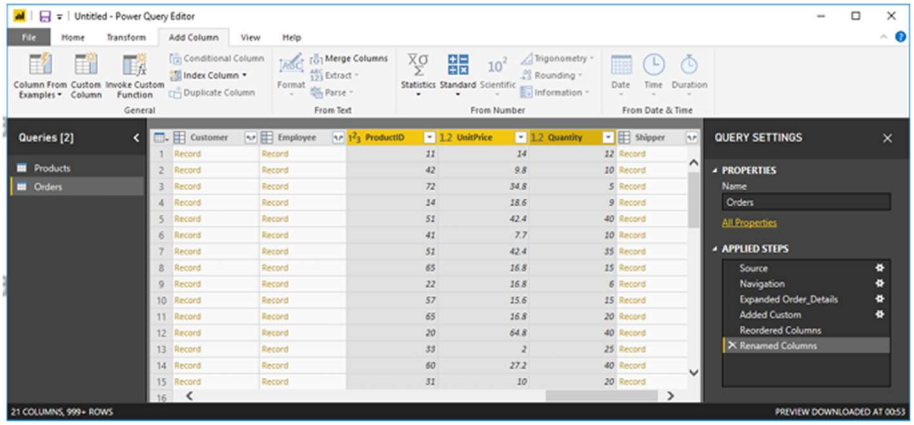
5. Rename and reorder columns in the query

In this step you finish making the model easy to work with when creating reports, by renaming the final columns and changing their order.

a) In Query Editor, drag the LineTotal column to the left, after ShipCountry.



b) Remove the Order_Details. prefix from the Order_Details.ProductID, Order_Details.UnitPrice and Order_Details.Quantity columns, by double-clicking on each column header, and then deleting that text from the column name.



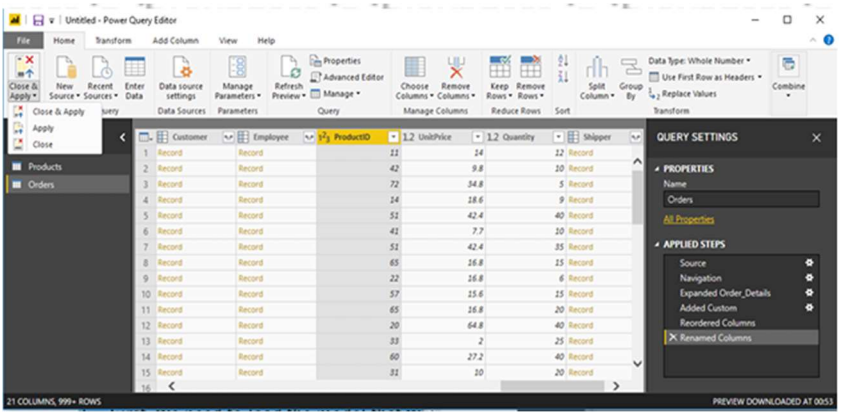
4. Combine the Products and Total Sales queries

Power BI Desktop does not require you to combine queries to report on them. Instead, you can create relationships between datasets. These relationships can be created on any column that is common to your datasets.

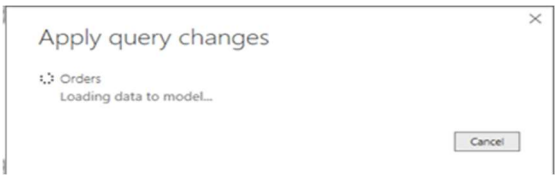
We have Orders and Products data that share a common 'ProductID' field, so we need to ensure there's a relationship between them in the model we're using with Power BI Desktop. Simply specify in Power BI Desktop that the columns from each table are related (i.e. columns that have the same values). Power BI Desktop works out the direction and cardinality of the relationship for you. In some cases, it will even detect the relationships automatically.

In this task, you confirm that a relationship is established in Power BI Desktop between the Products and Total Sales queries

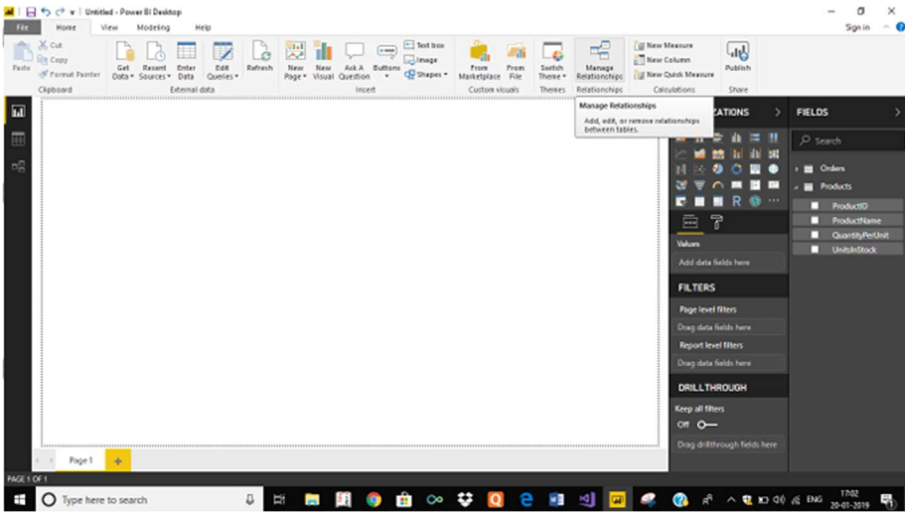
Step 1: Confirm the relationship between Products and Total Sales 1. First, we need to load the model that we created in Query Editor into Power BI Desktop. From the Home ribbon of Query Editor, select Close & Apply.



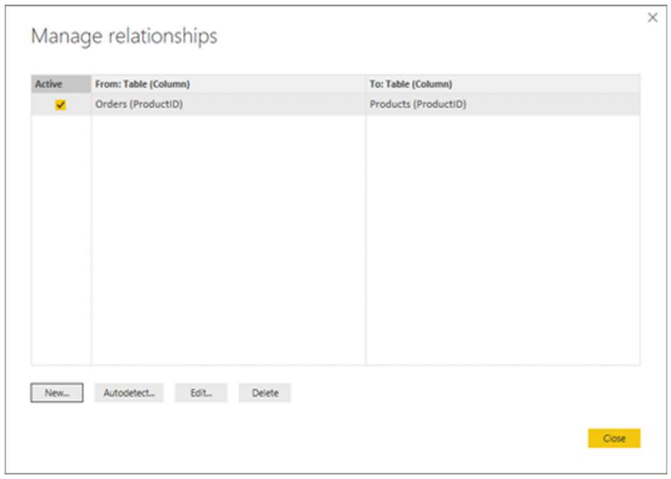
Step 2: Power BI Desktop loads the data from the two queries.



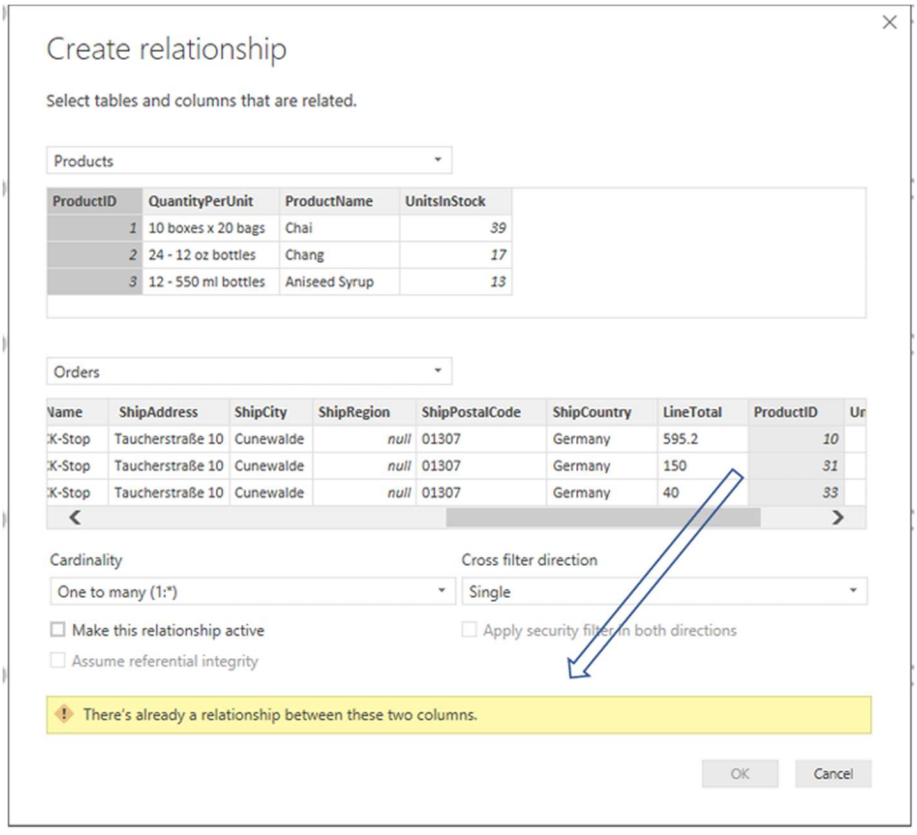
Step 3: Once the data is loaded, select the Manage Relationships button Home ribbon



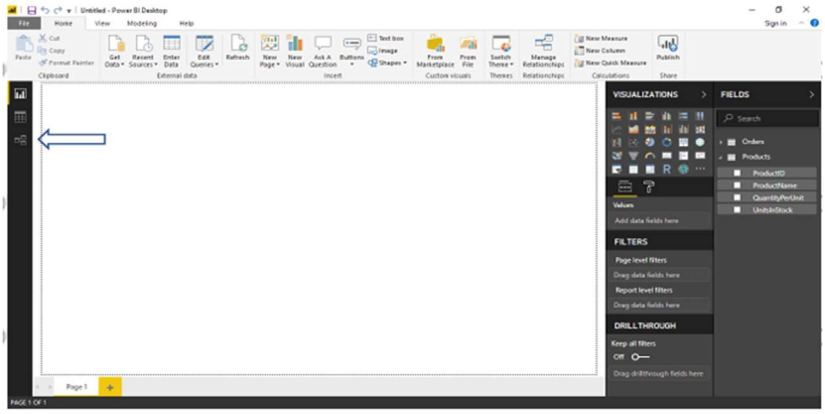
Step 4. Select the New… button



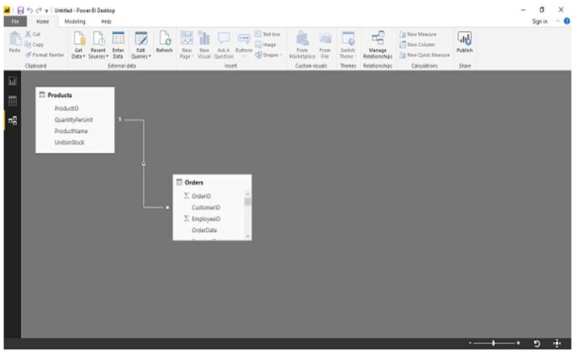
Step 5: When we attempt to create the relationship, we see that one already exists! As shown in the Create Relationship dialog (by the shaded columns), the ProductsID fields in each query already have an established relationship.



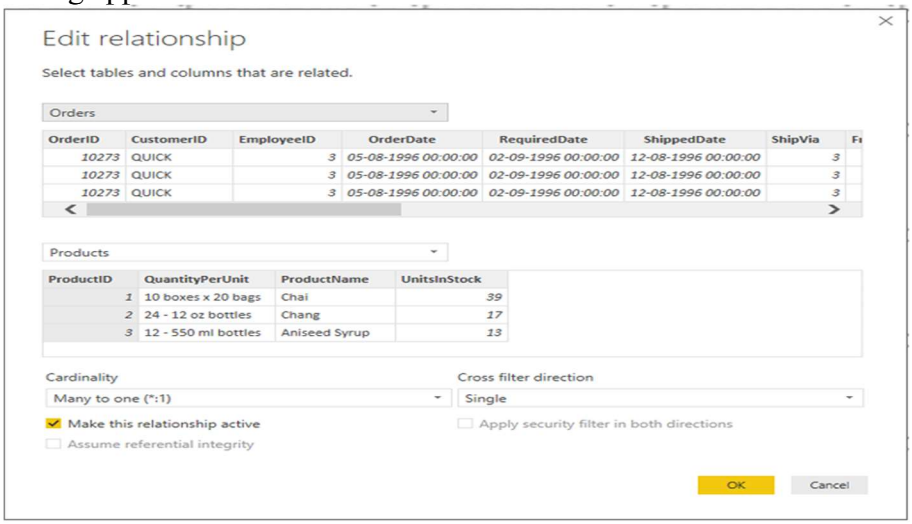
Step 6: Select Cancel, and then select Relationship view in Power BI Desktop.



Step 7: We see the following, which visualizes the relationship between the queries.



Step 8: When you double-click the arrow on the line that connects the to queries, an Edit Relationship dialog appears.



Step 9: No need to make any changes, so we'll just select Cancel to close the Edit Relationship dialog.

CONCLUSION:

Hence, we understood how to perform the Extraction Transformation and Loading (ETL) process to construct the database in the SQL server / Power BI.

ASSIGNMENT 3

TITLE: Data Analysis and Visualization using Advanced Excel

AIM: To perform data analysis and visualization using advanced excel

OBJECTIVES: Students will be able to perform data analysis and visualization using advanced excel

THEORY:

Data Analysis is a process of studying, cleaning, modeling, and transforming data with the purpose of finding useful information, suggesting conclusions, and supporting decision-making.

Data analysis is a process for getting large, unstructured data from different sources and converting it into information that is gone through the below process:

- Data Requirements Specification
- Data Collection
- Data Processing
- Data Cleaning
- Data Analysis
- Communication

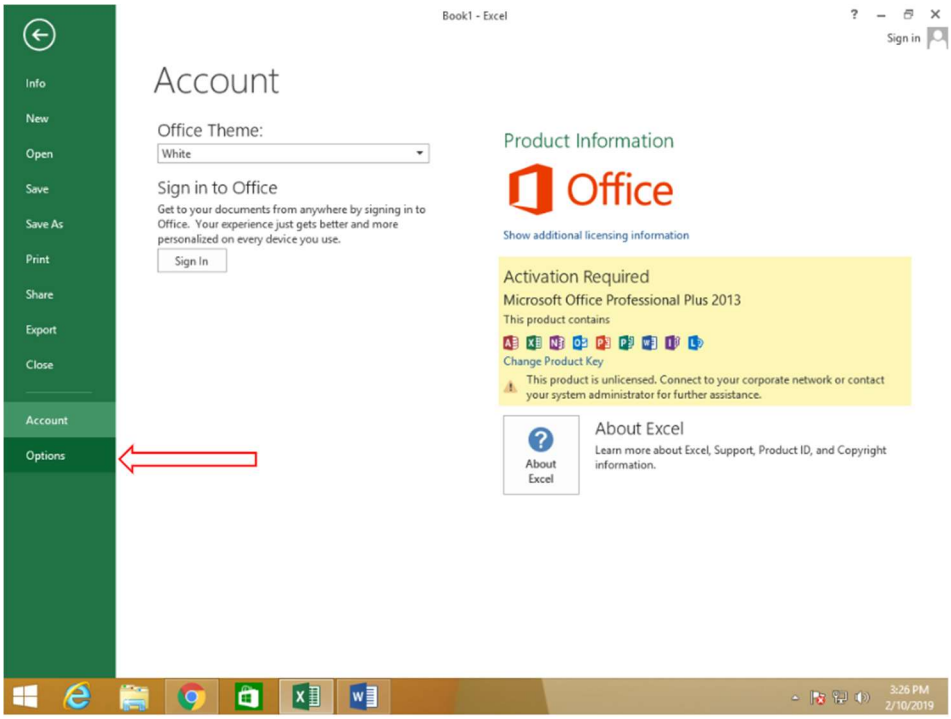
Data analytics is significant for business optimization performance. An organization can also use data analytics to make better business decisions and support analyzing customer trends and fulfillment, which can lead to unknown and better products and services.

Tools Used in Data Analysis

- Microsoft Excel
- Python
- R
- Jupyter Notebook
- Apache Spark
- Tableau
- Power BI

STEPS:

Create Power View Sheet
Make sure your Power View Add-In Is enabled in Excel 2013
Step 1: Open Excel 2013 (Professional version) → Open blank Workbook
Step 2: Go to File → Options

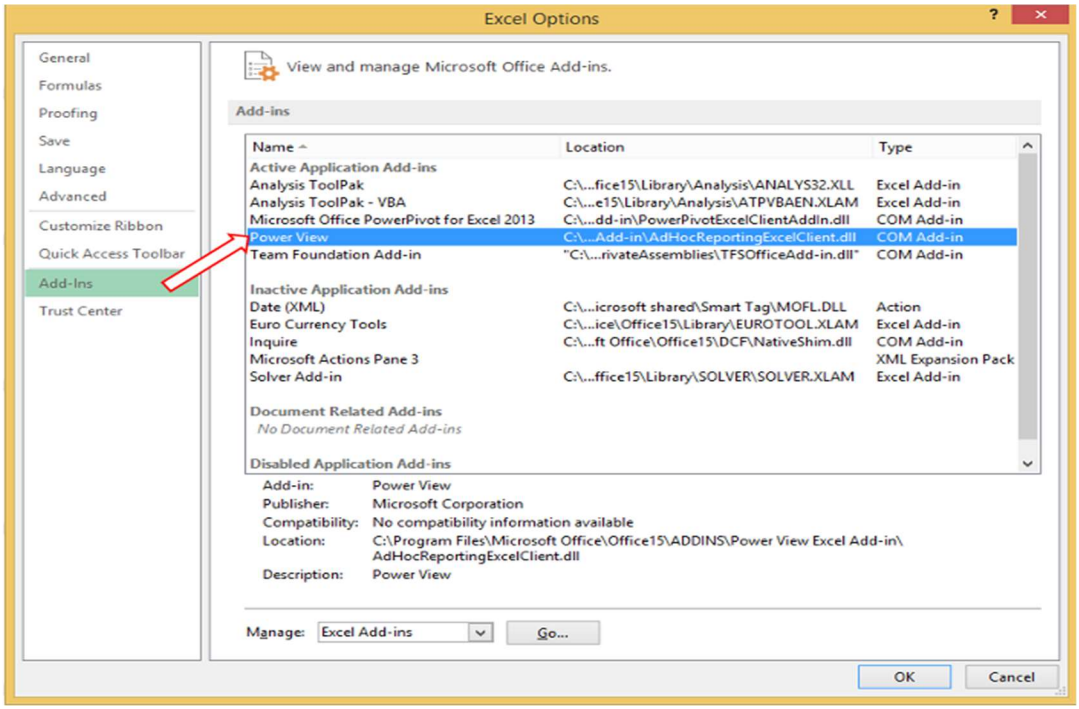


The Excel Options window appears.

Step 3 − Click on Add-Ins.

Step 4 − In the Manage box, click the drop-down arrow and select Excel Addins.

Step 5 − All the available Add-ins will be displayed. If Power View Add-in is enabled, it appears in Active Application Add-ins.



Step 6: If Power View does not appear in Active Add-Ins, follow these steps:

Step 1 − In the Excel Options Window, Click on Add-Ins.

Step 2 − In the Manage box, click the drop-down arrow and select COM Add-ins

Step 3 − Click on the Go button. A COM Add-Ins Dialog Box appears.

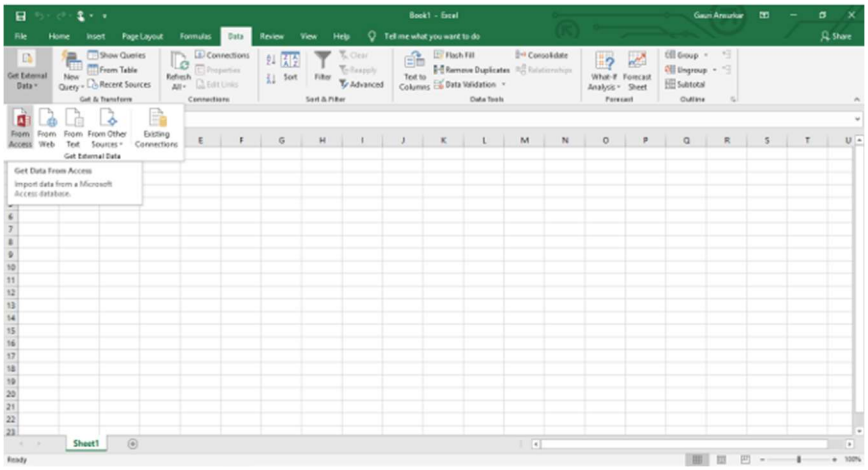
Step 4 − Check the Power View Check Box.

Step 5 − Click OK.

Now, you are ready to create the Power View sheet

To download OlympicMedal.accdb (required dataset) go to following link: https://download.microsoft.com/download/e/9/8/e981203a-d902-4d63afbf-424027b1e88c/olympicmedals.accdb

Step 6 : Go to data tab → get external data→ from access → olympicmedals.accdb



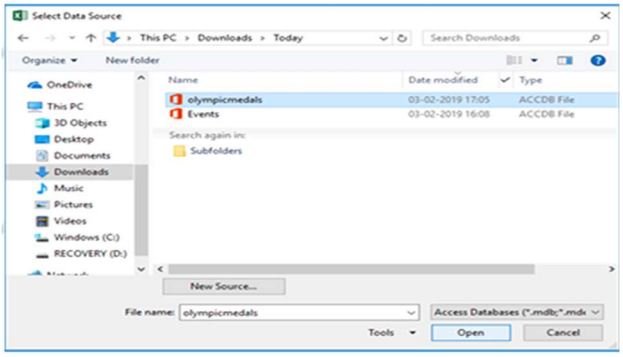
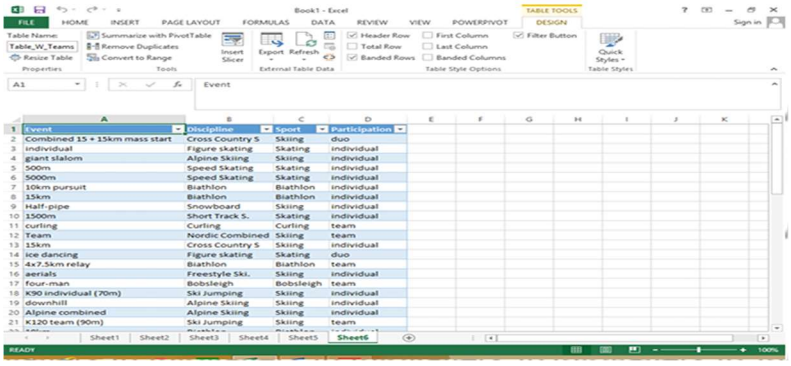
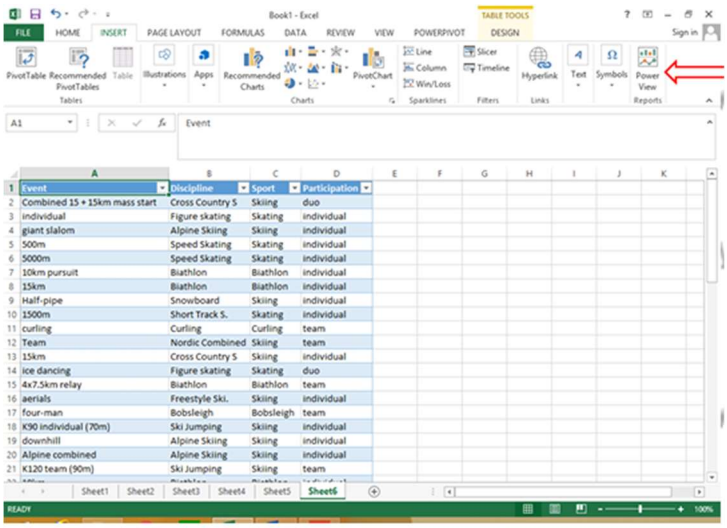


Table opens in new worksheet.

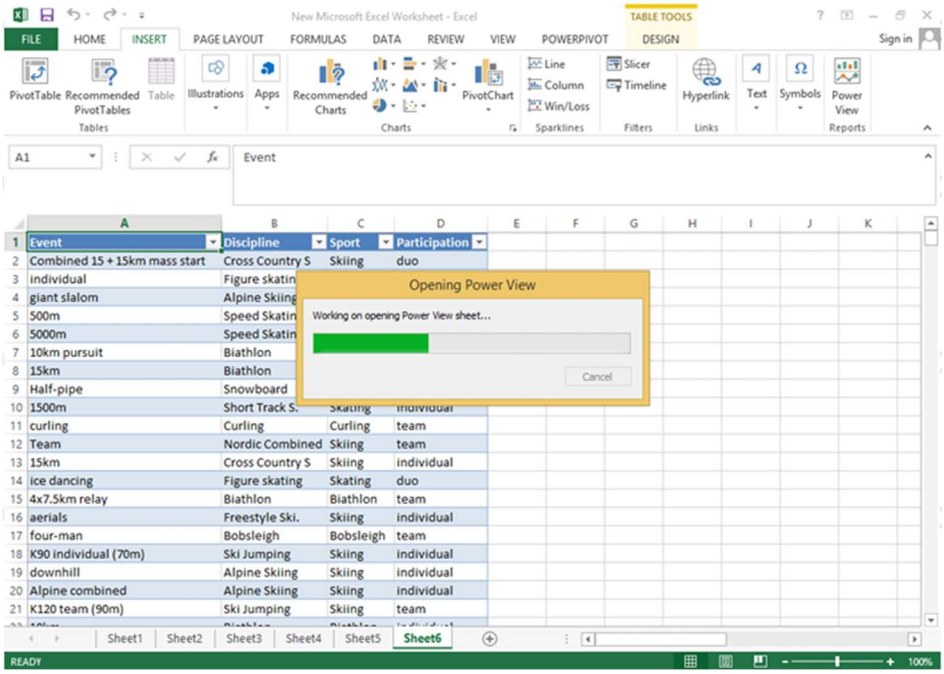


Step 7 − Click on Insert tab.

Step 8 − Click on Power View in Reports Group.

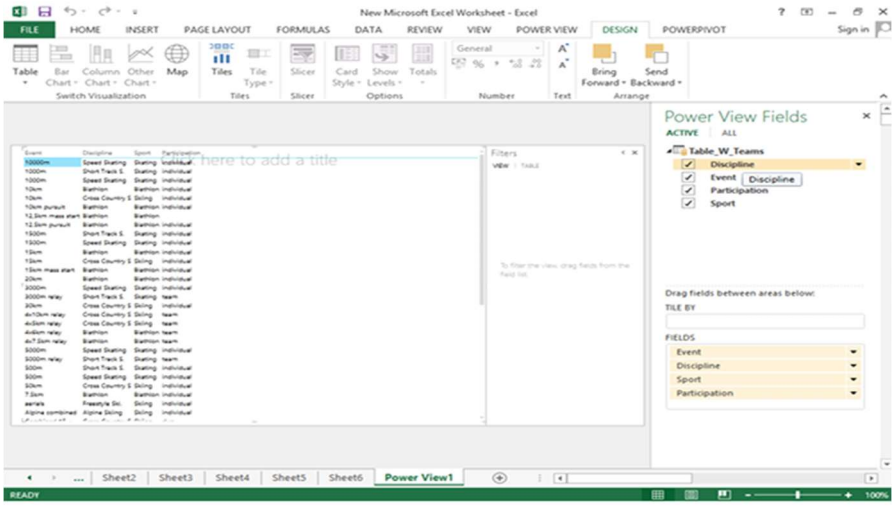


An Opening Power View window appears, showing the progress of Working on opening Power View sheet.



Power View Sheet is opened.

The Power View sheet is created for you and added to your Workbook with the Power View. On the Right-side of the Power View, you find the Power View Fields. Under the Power View Fields, you will find Areas. In the Ribbon, if you click on Design tab, you will find various Visualization options.

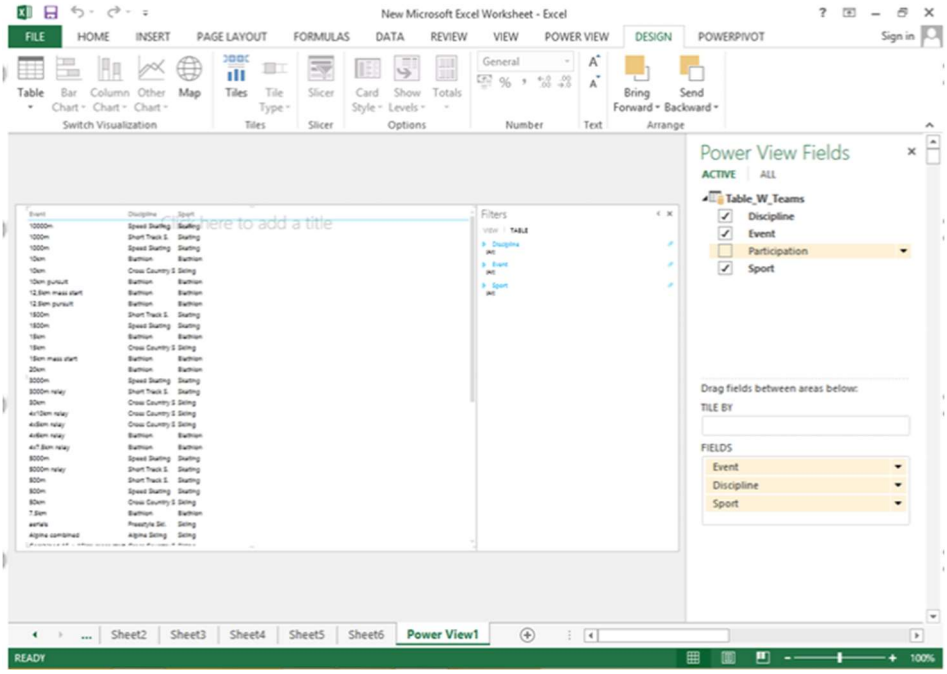


Create Charts and other Visualizations For every visualization you want to create, you start on a Power View sheet by creating a table, which you then easily convert to other visualizations, to find one that best illustrates your Data.

Step 1 − Under the Power View Fields, select the fields you want to visualize.

Step 2 − By default, the Table View will be displayed. As you move across the Table, on the top-right corner, you find two symbols – Filters and Pop out.

Step 3 − Click on the Filters symbol. The filters will be displayed on the right side. Filters has two tabs. View tab to filter all visualizations in this View and Table tab to filter the specific values in this table only.

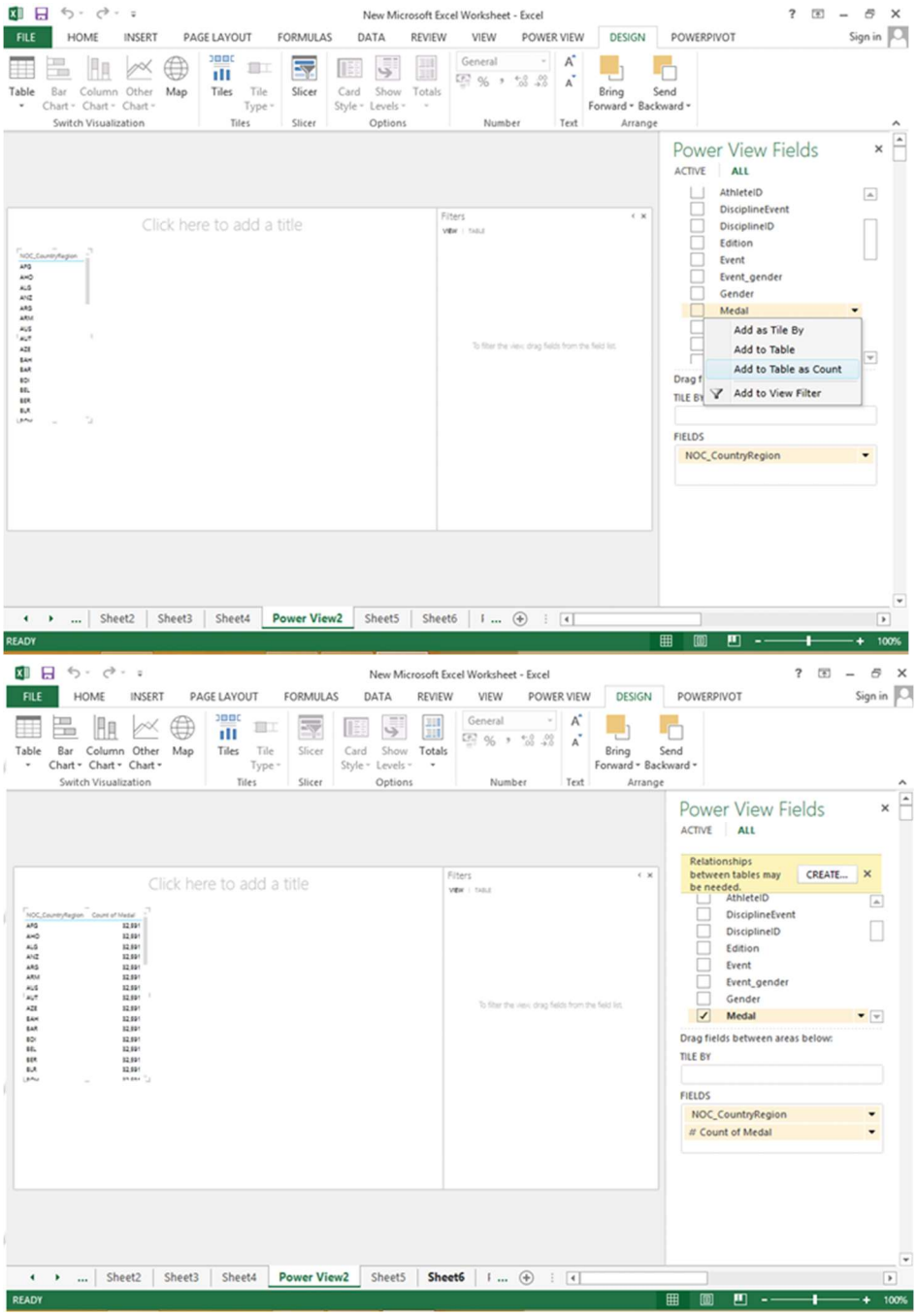


Visualization – Charts

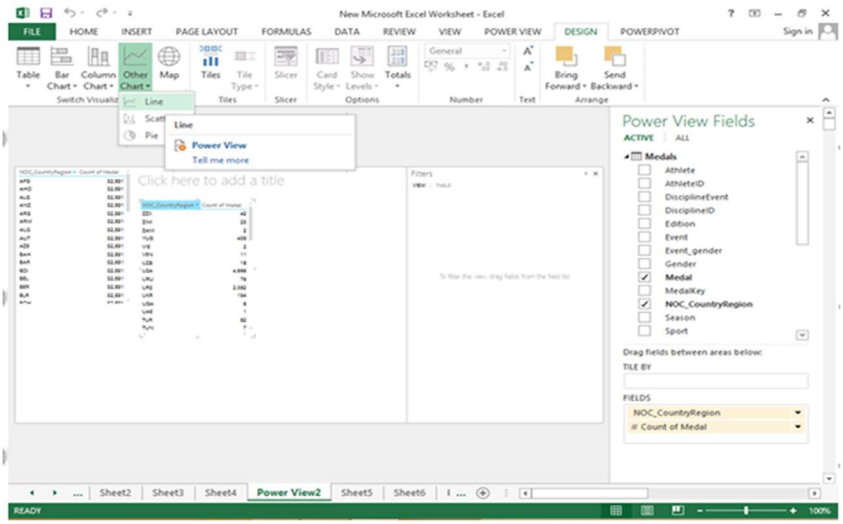
Step 1 − Create a Table Visualization from Medals data. You can use Line, Bar and Column Charts for comparing data points in one or more data series. In these Charts, the x-axis displays one field and the y-axis displays another, making it easy to see the relationship between the two values for all the items in the Chart. Line Charts distribute category data evenly along a horizontal (category) axis, and all numerical value data along a vertical (value) axis.

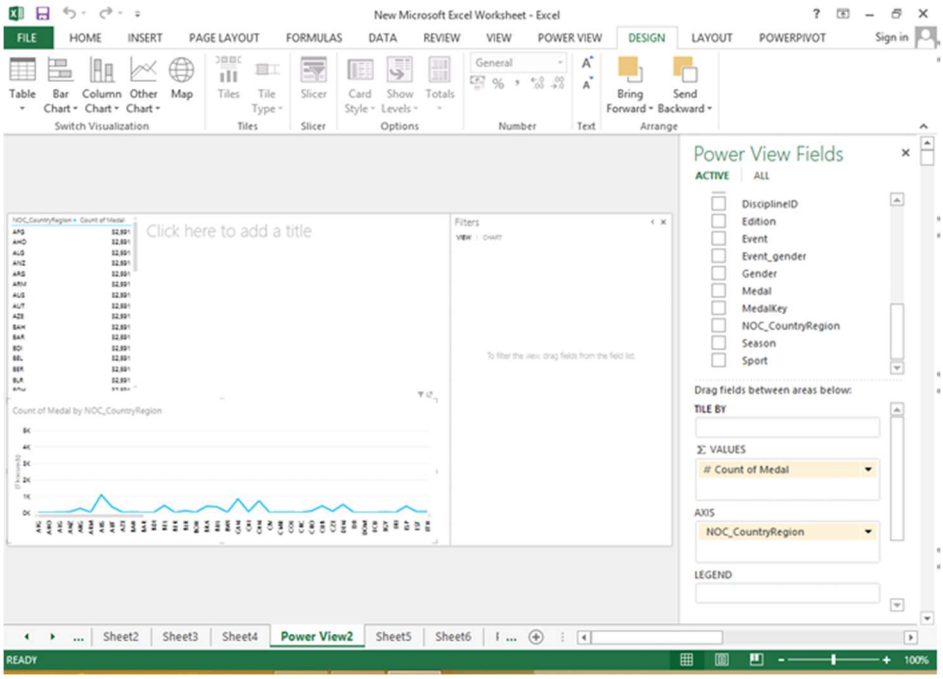
Step 2 − Create a Table Visualization for two Columns, NOC_CountryRegion and Count of Medal.

Step 3 − Create the same Table Visualization below.



Step 5 − Click on Line.



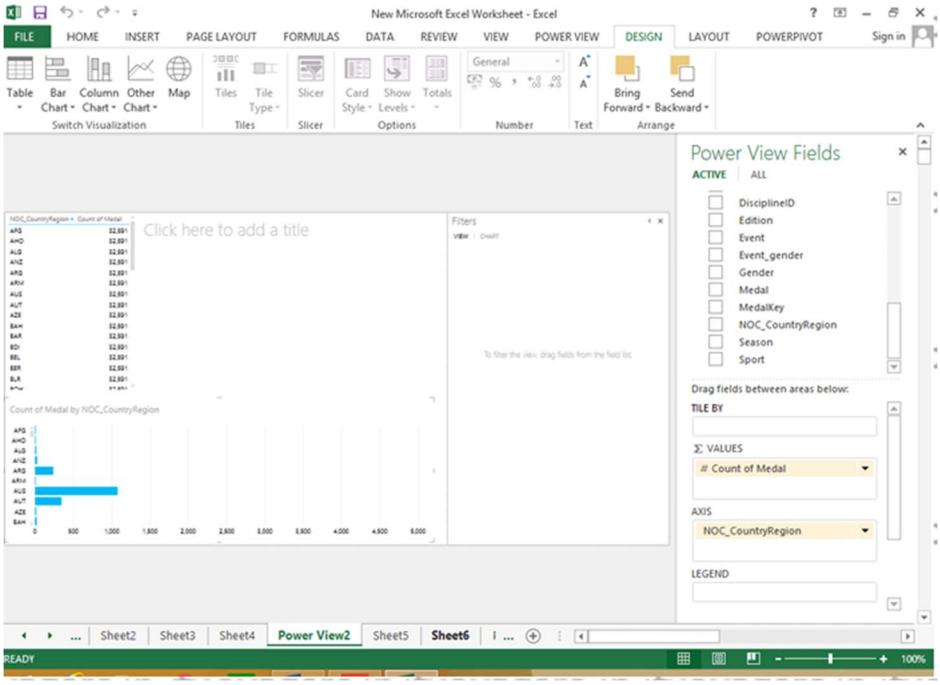


In a Bar Chart, categories are organized along the vertical axis and values along the horizontal axis. In Power View, there are three subtypes of the Bar Chart: Stacked, 100% stacked, and Clustered.

Step 7 − Click on the Line Chart Visualization.

Step 8 − Click on Bar Chart in the Switch Visualization Group.

Step 9 − Click on the Stacked Bar option.



CONCLUSION:

Hence, we performed data analysis and visualization using advanced excel.

ASSIGNMENT 4

TITLE: Perform the data classification algorithm using any Classification algorithm

AIM: To perform the data classification task using any classification algorithm.

OBJECTIVE: Students will be able to implement the classification algorithm.

THEORY:

Logistic Regression is a type of regression analysis used for predicting the probability of a binary outcome. It's particularly suited for situations where the dependent variable is categorical and the independent variables are continuous or categorical.
STEPS:
- Dataset Loading: The code loads a dataset, such as the Iris dataset, for analysis. In practice, datasets can vary widely depending on the problem at hand, ranging from simple toy datasets to large, real-world datasets.

- Data Preparation: Once the dataset is loaded, the relevant features and target variable are extracted. Features represent the independent variables, while the target variable represents the dependent variable we are trying to predict. In logistic regression, the target variable usually represents a binary outcome.

- Feature Scaling: Before fitting the logistic regression model, it's common practice to scale the features. This ensures that all features have a similar scale and prevents certain features from dominating the optimization process during model training.

- Model Training: Logistic regression is initialized and trained using the training data. During training, the model learns the relationship between the independent variables (features) and the dependent variable (target). This is typically achieved through an optimization process, such as gradient descent, where the model parameters are iteratively adjusted to minimize the error between predicted and actual outcomes.

- Prediction and Evaluation: Once the model is trained, it's used to make predictions on unseen data (testing set). The accuracy of the model is then evaluated using metrics such as accuracy score and classification report, which provide insights into the model's performance in terms of correctly predicting the classes of the target variable.

Overall, logistic regression is a simple yet powerful algorithm for binary classification tasks, and the provided code demonstrates its application on a dataset, along with necessary preprocessing steps and model evaluation techniques.

CODE:

```
from sklearn.datasets import load_iris
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# Load the iris dataset
iris = load_iris()

# Features
X = iris.data

# Target variable
y = iris.target

# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Initialize and train the logistic regression model
logreg = LogisticRegression()
logreg.fit(X_train_scaled, y_train)

# Predictions
y_pred_train = logreg.predict(X_train_scaled)
y_pred_test = logreg.predict(X_test_scaled)

# Model evaluation
train_accuracy = accuracy_score(y_train, y_pred_train)
test_accuracy = accuracy_score(y_test, y_pred_test)

print("Training Accuracy:", train_accuracy)
print("Testing Accuracy:", test_accuracy)

# Additional evaluation metrics
print("Classification Report on Test Data:")
print(classification_report(y_test, y_pred_test))
```

```
Training Accuracy: 0.9666666666666667
Testing Accuracy: 1.0
Classification Report on Test Data:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

CONCLUSION:

Hence, we have used Logistic regression classification algorithm and performed classification task using python.

ASSIGNMENT 5

TITLE: Perform the data clustering algorithm using any Clustering algorithm.

AIM: To perform the data clustering task using any Clustering algorithm.

OBJECTIVE: Students will be able to implement the Clustering algorithm.

THEORY:

K-Means is one of the most popular unsupervised machine learning algorithms used for clustering tasks. The goal of clustering is to group similar data points together based on their features, with the objective of maximizing intra-cluster similarity and minimizing inter-cluster similarity.

K-Means Working:

1. Initialization:
   - The algorithm starts by randomly initializing K cluster centroids. These centroids represent the initial cluster centers in the feature space.

2. Assignment Step:
   - Each data point is assigned to the nearest centroid based on some distance metric, typically Euclidean distance. This step partitions the dataset into K clusters.

3. Update Step:
   - After assigning data points to clusters, the centroids are updated by computing the mean of all data points assigned to each cluster. This moves the centroids to the center of their respective clusters.

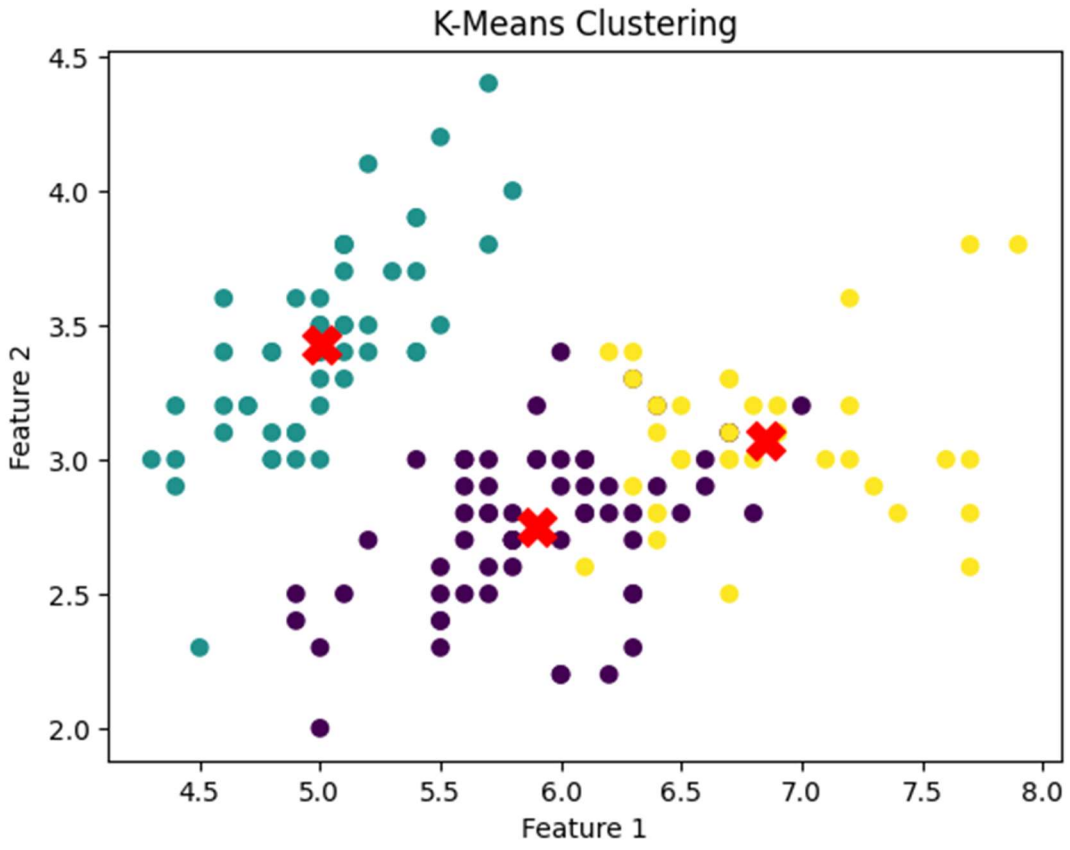
4. Iteration:
   - Steps 2 and 3 are repeated iteratively until convergence. Convergence occurs when the centroids no longer change significantly or when a predefined number of iterations is reached.

5. Final Clustering:
   - Once convergence is achieved, the final clusters are formed, and each data point belongs to the cluster whose centroid is closest to it.

CODE:

```
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Load the iris dataset
iris = load_iris()

# Features
X = iris.data
```

```
# Initialize KMeans with the number of clusters (3 for the iris dataset)
kmeans = KMeans(n_clusters=3)

# Fit KMeans to the data
kmeans.fit(X)

# Get the cluster centroids and labels
centroids = kmeans.cluster_centers_
labels = kmeans.labels_

# Visualizing the clusters (assuming 2D data for simplicity)
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')
plt.scatter(centroids[:, 0], centroids[:, 1], marker='X', s=200, c='red')
plt.title('K-Means Clustering')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()
```



CONCLUSION:

Hence, we performed the task of clustering by using the K-Means clustering algorithm using python.

ASSIGNMENT 6

Title: Exploring Word Frequency Calculation Using MapReduce Paradigm

Aim: To develop a MapReduce program for efficiently calculating the frequency of a specific word in a given file.

Objective:

- Understand the MapReduce paradigm and its application in distributed computing.
- Implement a MapReduce program to process large volumes of data and compute word frequencies in parallel.
- Develop a scalable solution to handle varying sizes of input files.

Theory:

- MapReduce Paradigm: Introduction to the MapReduce programming model for distributed data processing.
- Word Frequency Calculation: Explanation of the task involving counting the occurrences of a given word in a text document.
- Mapper Function: Description of the mapper function's role in processing input data and emitting intermediate key-value pairs.
- Reducer Function: Explanation of the reducer function's role in aggregating intermediate results and generating the final output.
- Input Splitting: Understanding how input data is divided into manageable chunks for processing across multiple nodes.
- Shuffling and Sorting: Overview of the process where intermediate key-value pairs are shuffled and sorted before being sent to reducer nodes.
- Output Generation: Description of how the reducer nodes combine intermediate results to produce the final word frequency count.

Input Data Format: Explanation of how input data is typically structured for processing in a MapReduce program, often as key-value pairs where the key represents the document identifier and the value contains the content of the document.

Combiner Function: Introduction to the combiner function, which acts as a mini-reducer and performs local aggregation of intermediate results on each mapper node, reducing the volume of data shuffled over the network.

Partitioning: Description of how partitioning divides the intermediate key-value pairs among reducer nodes based on a hashing function, ensuring that all occurrences of a particular word are processed by the same reducer.

Fault Tolerance: Discussion on the fault tolerance mechanisms inherent in MapReduce, such as automatic re-execution of failed tasks and replication of intermediate data to tolerate node failures.

Scalability: Explanation of how MapReduce programs can easily scale to handle larger datasets by adding more compute nodes to the cluster, making them suitable for processing terabytes or even petabytes of data.

Optimization Techniques: Overview of various optimization techniques, including data compression, speculative execution, and task pipelining, aimed at improving the performance and efficiency of MapReduce jobs.

MapReduce Frameworks: Introduction to popular MapReduce frameworks such as Apache Hadoop and Apache Spark, which provide high-level abstractions and APIs for writing MapReduce programs and managing distributed computing resources.

Real-World Applications: Examples of real-world applications of MapReduce beyond word frequency calculation, including web indexing, log analysis, machine learning, and data warehousing.

Challenges and Limitations: Discussion on the challenges and limitations of MapReduce, such as overhead associated with data shuffling, difficulty in expressing complex data processing workflows, and the need for specialized skills to optimize job performance.

Future Directions: Speculation on the future evolution of MapReduce and its role in the broader landscape of distributed computing, considering emerging technologies such as serverless computing and container orchestration platforms.

Code:

```
def map_reduce(file_path, target_word):
    word_count = 0
    with open(file_path, 'r') as file:
        # Map phase: Emit each word with count 1
        mapped_data = [(word.lower(), 1) for line in file for word in line.strip().split()]

    # Reduce phase: Aggregate counts for the target word
    for word, count in mapped_data:
        if word == target_word.lower():
            word_count += count

    return word_count

# Example usage
file_path = 'G.txt'  # Update to your file path
target_word = "Girish"
```

```
# Calculate the frequency of the target word in the file
frequency = map_reduce(file_path, target_word)
print(f"The word '{target_word}' appears {frequency} times.")
```

```
The word 'hello' appears 1 times.
```

Conclusion:

The creation of a MapReduce program to calculate word frequencies showcases the MapReduce paradigm's prowess in handling large-scale data. Through parallel processing, the program efficiently distributes tasks across multiple nodes, yielding quicker execution times compared to sequential methods.

ASSIGNMENT 7

Title: Visualization: Connect to data, Build Charts and Analyze Data, Create Dashboard, Create Stories using Tableau/PowerBI.

Aim: This project aims to leverage the capabilities of data visualization tools such as Tableau or PowerBI to seamlessly connect to various data sources, construct informative visualizations, perform in-depth data analysis, design interactive dashboards, and narrate compelling data stories.

Objective:

1. Familiarize with Tableau/PowerBI and their features for data visualization and analysis.
2. Explore data connectivity options to seamlessly integrate diverse data sources.
3. Develop skills in constructing meaningful visualizations to convey insights effectively.
4. Master dashboard creation to present key metrics and trends in a concise and interactive manner.
5. Understand the art of crafting data narratives to communicate findings and drive actionable insights.

Theory:

Data Visualization Tools: Introduction to Tableau/PowerBI and their significance in transforming raw data into visually appealing and insightful representations.

Data Connection: Explanation of methods to connect Tableau/PowerBI to various data repositories, including databases, spreadsheets, and cloud services, enabling seamless data integration.

Visualization Construction: Overview of techniques for designing effective visualizations, including selecting appropriate chart types, customizing aesthetics, and employing best practices for data representation.

Data Analysis: Discussion on analytical capabilities within Tableau/PowerBI for exploring data trends, performing calculations, and deriving actionable insights through interactive exploration.

Dashboard Creation: Explanation of the process of building dynamic dashboards in Tableau/PowerBI to consolidate key metrics, facilitate data exploration, and enable decision-making through interactive filtering and drill-down functionalities.

Storytelling: Introduction to the concept of data storytelling and its importance in contextualizing insights, engaging stakeholders, and driving action through compelling narratives woven around data visualizations.

Data Connectivity: Overview of Tableau/PowerBI's ability to connect to various data sources including databases, spreadsheets, cloud storage, and web services, facilitating real-time data analysis and visualization.
Visualization Techniques: Description of advanced visualization techniques such as heatmaps, treemaps, and geographical mapping to represent complex data relationships and patterns effectively.
Interactivity: Explanation of interactive features within Tableau/PowerBI such as filters, parameters, and actions, enabling users to dynamically explore and interact with visualizations for deeper insights.
Dashboard Design Principles: Discussion on principles of effective dashboard design including layout, color schemes, and storytelling elements to create impactful and user-friendly dashboards.
Collaboration and Sharing: Overview of collaboration features in Tableau/PowerBI for sharing insights, collaborating with team members, and publishing dashboards for broader consumption.
Storytelling with Data: Introduction to techniques for crafting data-driven narratives that engage stakeholders, communicate insights, and drive decision-making using visualizations and storytelling elements.

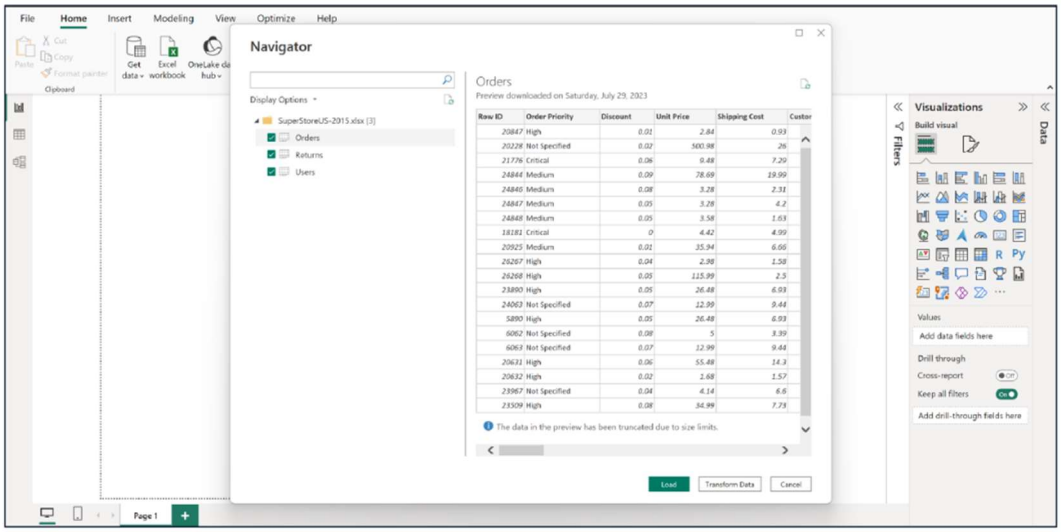
Dataset link: https://powerbidocs.com/wp-content/uploads/2019/11/SuperStoreUS-2015.xlsx

Code:

Step 1: Connect to Your Data Source

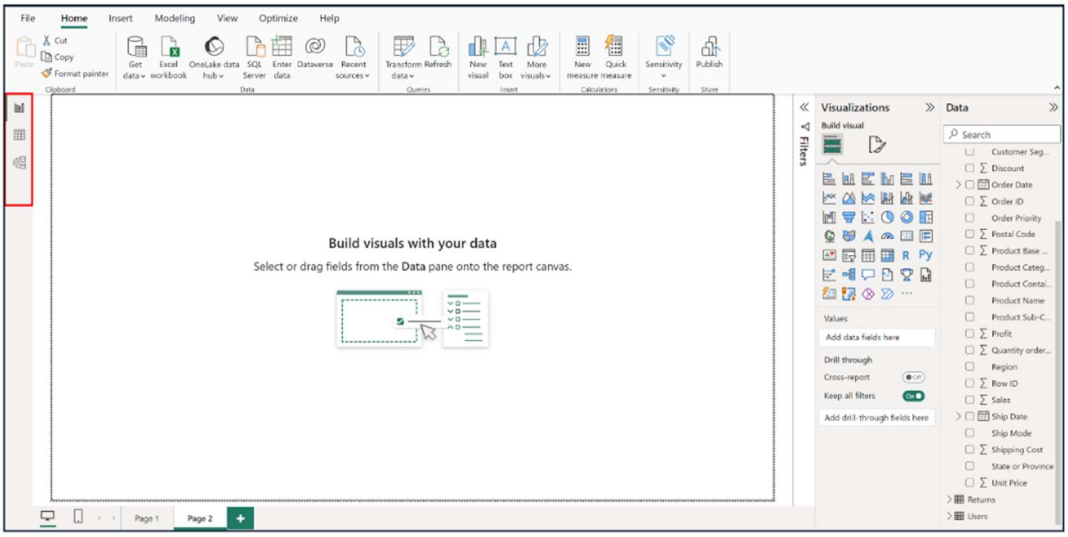
The first step in creating your Sales Dashboard using Power BI is to connect to your data source. For this project, we will utilize a sales dataset from a USA-based superstore for the year 2015. You can also download this dataset from here. This dataset contains comprehensive sales information, including product details, categories, customer segments, regions, order dates, sales, profits, etc. Establishing this data connection is the foundation upon which we will build insightful visualizations and metrics to showcase sales performance and trends.

To connect to a data store using Power BI -
Open Power BI Desktop.
Click "Home" > "Get Data" > select your data source (e.g., Excel, CSV, database).
Browse and select your dataset, then follow the prompts to configure the connection and load data into Power BI.



Step 2: Create A New Report

The next step in our journey to build a Sales Dashboard using Power BI is to create a new report canvas. After loading your data, click on the "Report View" tab (located on the left side) in Power BI Desktop. This will open a blank canvas where you can start designing your dashboard. Here, you'll add various visualizations, charts, and KPIs to represent your sales data effectively. This canvas serves as a dynamic workspace to arrange and customize elements that will form the basis of your interactive sales dashboard.



Step 3: Add Visualizations

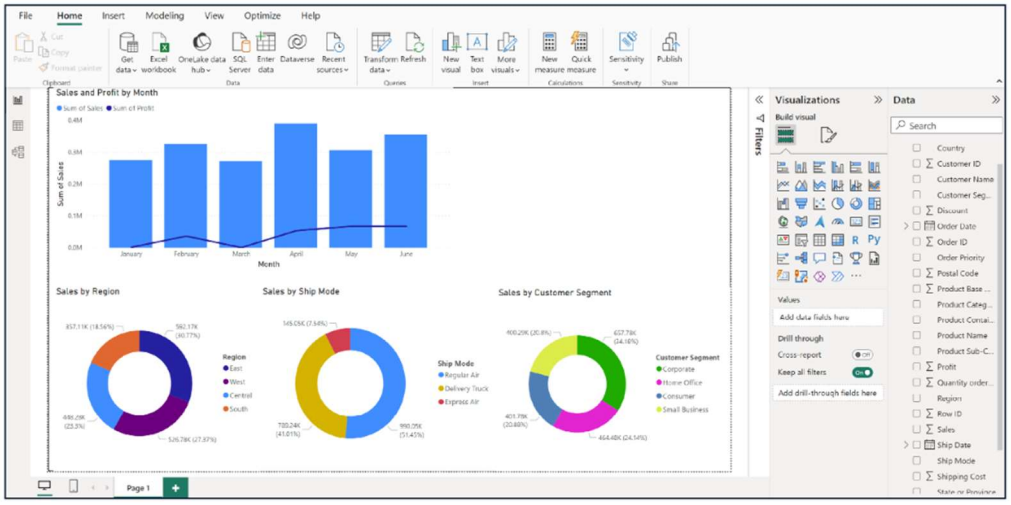
In this pivotal step of our Sales Dashboard creation process, we'll add a wide range of visualizations to convey vital insights from our data. To create these visualizations in Power BI –

Drag and drop the required fields onto the visualization area.
Choose the appropriate visualization type (bar, line, or pie chart).
Customize the visualizations by configuring labels, colors, and other settings.

In this project, we will focus on below visualizations -

To visualize sales and profit trends over time, we'll add a combination bar and line chart. The bars will represent monthly sales, while the line will represent profit. This dual-axis visualization will provide a clear depiction of how sales and profit correlate month by month, enabling us to spot trends and patterns.

For a comprehensive understanding of our sales distribution, we'll employ pie charts. These charts will depict the proportion of sales across various segments, including regions, customer segments, and shipping modes.



Step 4: Add the KPIs to the Dashboard

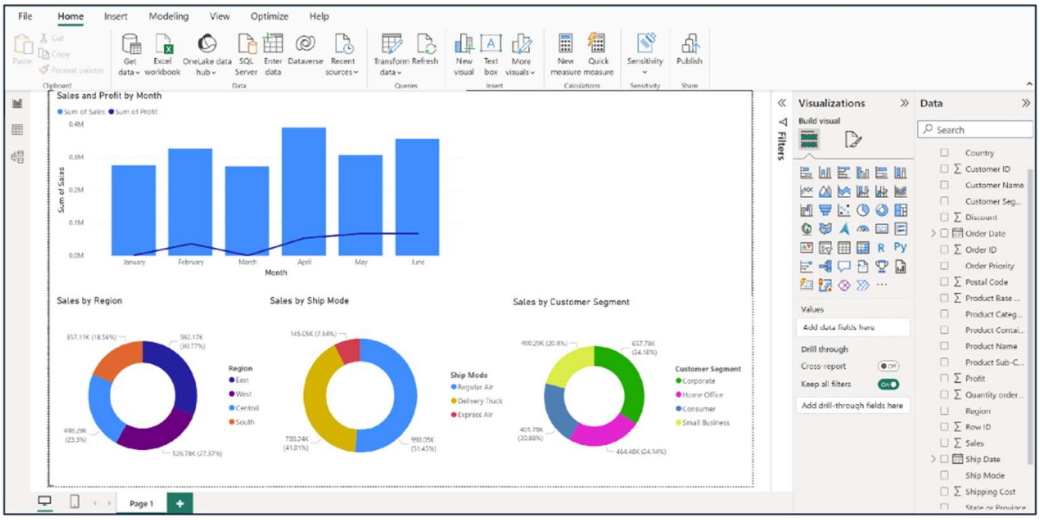
To add Key Performance Indicators (KPIs) to the dashboard, we'll utilize card visualizations in Power BI -

Click on "Card" from the Visualizations pane.

Drag and drop the relevant field onto the "Values" section of the card.

Customize the card by adding appropriate titles and formatting.

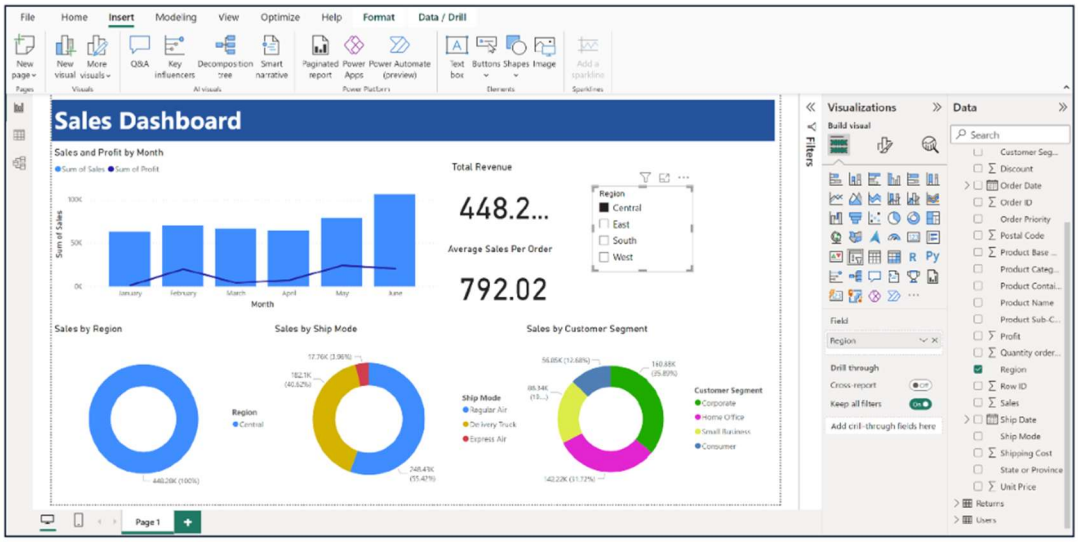
For instance, you can create KPI cards to display metrics like Total Revenue, Average Order Value, and Year-over-Year Growth. These concise visual elements provide immediate insights into essential performance indicators, enhancing the dashboard's impact and usability. In this project, we will add two KPIs, total sales and average sales per order, in the dashboard.



Step 5: Add Filters and Interactivity

Adding filters and interactivity to the sales dashboard is crucial as it empowers users to explore the data, focus on specific aspects, and gain deeper insights. Filters allow users to dynamically adjust data views, making the dashboard adaptable to various scenarios. Here's how to achieve it in Power BI -

From the "Visualizations" pane, select "Slicer."
Drag fields relevant to filtering (e.g., region, order date) onto the slicer.
Users can now interactively filter data by selecting options within the slicer.
This feature enables users to dynamically refine the displayed data dynamically, gaining deeper insights and customizing their analysis according to specific criteria. In this dashboard, we will add a filter based on the region feature. We have also added a title bar in the canvas, as shown below.



Conclusion:

The utilization of Tableau/PowerBI for data visualization and analysis presents a robust solution for extracting actionable insights from diverse datasets. By seamlessly connecting to various data sources and employing advanced visualization techniques, these tools empower users to construct informative charts and dashboards. Overall, Tableau/PowerBI serve as invaluable assets in transforming raw data into actionable intelligence, making them indispensable tools in the modern data-driven landscape.

ASSIGNMENT 8

Title: Exploring Student Grade Calculation with MapReduce

Aim: To develop a MapReduce program for efficiently calculating the grades of students, showcasing the applicability of MapReduce in educational data processing.

Objective:

1. Understand the MapReduce framework and its application in processing educational data.
2. Implement a scalable solution for calculating student grades using MapReduce, accommodating varying class sizes and grading criteria.
3. Explore the parallel processing capabilities of MapReduce to efficiently process large volumes of student performance data.

Theory:

MapReduce Paradigm: Introduction to the MapReduce programming model and its suitability for distributed data processing tasks.

Student Grade Calculation: Explanation of the process involved in determining student grades based on assessment scores and grading criteria.

Mapper Function: Description of the role of the mapper function in processing student records and emitting intermediate key-value pairs.

Reducer Function: Explanation of the reducer function's responsibility in aggregating intermediate results and generating the final grade outcomes.

Data Processing Pipelines: Overview of how MapReduce facilitates the construction of data processing pipelines for transforming raw student data into actionable insights.

Scalability: Discussion on how MapReduce enables scalability by distributing computation across multiple nodes, allowing for efficient processing of student performance data in large classrooms.

Data Aggregation: Explanation of how MapReduce aggregates individual student scores to calculate overall grades, considering different weighting schemes for assessments.

Data Quality Assurance: Brief overview of techniques for ensuring data quality and integrity in the student performance dataset, such as data validation and error handling.

Normalization and Aggregation: Explanation of techniques for normalizing student scores and aggregating them to calculate overall grades, considering various assessment types and grading criteria.

Data Preprocessing: Overview of data preprocessing steps such as data cleaning, transformation, and filtering to ensure the quality and consistency of input data before processing with MapReduce.

Grade Distribution Analysis: Introduction to the analysis of grade distributions using MapReduce, allowing educators to identify trends and patterns in student performance across different assessments.
Performance Optimization: Discussion on techniques for optimizing the performance of MapReduce programs for student grade calculation, including data partitioning strategies and task scheduling algorithms.
Data Privacy and Security: Brief overview of data privacy and security considerations in educational data processing, including methods for anonymizing student records and protecting sensitive information.
Feedback Generation: Description of how MapReduce can be used to generate personalized feedback for students based on their performance, enabling targeted interventions to support learning outcomes.
Integration with Learning Management Systems: Introduction to integrating MapReduce programs for student grade calculation with existing learning management systems, facilitating seamless data exchange and analysis.
Real-Time Monitoring: Brief explanation of real-time monitoring capabilities in MapReduce frameworks, enabling educators to monitor student progress and intervene as needed to support student learning.

Code:

```
from collections import defaultdict
def map_function(student_scores):
    """
    Emits tuples of student_id and score.
    """
    for student_id, score in student_scores:
        yield student_id, score

def reduce_function(mapped_data):
    """
    Assigns grades based on scores.
    """
    grades = {}
    for student_id, score in mapped_data:
        if score > 80:
            grade = 'A'
        elif score > 60:
            grade = 'B'
        elif score > 40:
            grade = 'C'
```

```
        else:
            grade = 'D'
        grades[student_id] = grade
    return grades

def map_reduce(student_scores):
    """
    Simulates the MapReduce process for assigning grades to students based on scores.
    """
    # Map phase
    mapped_data = list(map_function(student_scores))

    # Reduce phase
    grades = reduce_function(mapped_data)

    return grades

# Example student scores (student_id, score)
student_scores = [
    (1, 85),
    (2, 75),
    (3, 55),
    (4, 35),
    (5, 90)
]

# Find grades for students
grades = map_reduce(student_scores)

# Print grades for each student
for student_id, grade in grades.items():
    print(f"Student {student_id} has been assigned grade {grade}.")
```

```
Student 1 has been assigned grade A.
Student 2 has been assigned grade B.
Student 3 has been assigned grade C.
Student 4 has been assigned grade D.
Student 5 has been assigned grade A.
```

Conclusion:

The MapReduce program for student grade calculation showcases the scalability and efficiency of this approach in educational data processing. Through parallel processing, it efficiently handles large student datasets, offering valuable insights into student achievement.

ASSIGNMENT 9

Title: Analyzing Titanic Data with MapReduce: Gender-based Analysis of Casualties.

Aim: To develop a MapReduce program for analyzing Titanic ship data to determine the average age of deceased males and the number of deceased females in each passenger class.

Objective:

1. Understand the MapReduce framework and its application in processing and analyzing large datasets.
2. Implement a scalable solution to analyze Titanic ship data, focusing on gender-based analysis of casualties.
3. Calculate the average age of deceased males and determine the count of deceased females in each passenger class using MapReduce.

Theory:

MapReduce Paradigm: Introduction to the MapReduce programming model and its suitability for distributed data processing tasks.

Titanic Ship Data Analysis: Explanation of the dataset containing information about Titanic passengers, including their gender, age, and passenger class.

Mapper Function: Description of the mapper function's role in parsing Titanic ship data and emitting intermediate key-value pairs for further analysis.

Reducer Function: Explanation of the reducer function's responsibility in aggregating intermediate results and generating the final analysis outcomes.

Data Filtering: Overview of techniques for filtering Titanic ship data to focus only on relevant information, such as filtering by gender and survival status.

Data Aggregation: Explanation of how MapReduce aggregates individual passenger data to calculate the average age of deceased males and determine the count of deceased females in each passenger class.

Parallel Processing: Discussion on how MapReduce enables parallel processing of Titanic ship data across multiple nodes, allowing for efficient analysis of large datasets.

Data Partitioning: Explanation of how Titanic ship data can be partitioned and distributed across mapper nodes based on key attributes such as passenger ID or class, ensuring balanced workload distribution.

Join Operations: Overview of join operations in MapReduce to combine Titanic ship data with auxiliary datasets, such as passenger information or survival status, for comprehensive analysis.

Conditional Filtering: Discussion on conditional filtering techniques in MapReduce to selectively process Titanic ship data based on specified criteria, such as gender and survival status.
Data Serialization: Introduction to data serialization techniques for efficiently storing and transferring Titanic ship data between mapper and reducer nodes, optimizing data processing performance.
MapReduce Combiners: Description of how combiners in MapReduce can be utilized to perform local aggregation of intermediate results on mapper nodes, reducing data transmission overhead and improving performance.
Iterative Algorithms: Brief explanation of how iterative algorithms, such as calculating averages or counts, can be implemented in MapReduce using iterative MapReduce job chaining or distributed cache mechanisms.
Error Handling: Overview of error handling strategies in MapReduce to handle data parsing errors, missing values, or other exceptions encountered during Titanic ship data analysis, ensuring robustness and reliability of the program.
MapReduce Frameworks: Discussion on popular MapReduce frameworks such as Apache Hadoop and Apache Spark, which provide high-level abstractions and tools for developing, deploying, and managing MapReduce programs.

Code:

```
import pandas as pd
def map_reduce_with_pandas(input_file):
    # Load the dataset
    df = pd.read_csv(input_file)

    # Map: Filter deceased males and transform data for average age calculation
    deceased_males = df[(df['Survived'] == 0) & (df['Sex'] == 'male')]

    # Reduce: Calculate average age of deceased males
    average_age_deceased_males = deceased_males['Age'].mean()

    # Map: Filter deceased females and transform data for count by class
    deceased_females_by_class = df[(df['Survived'] == 0) & (df['Sex'] == 'female')]

    # Reduce: Count deceased females by class
    count_deceased_females_by_class =
deceased_females_by_class['Class'].value_counts()
    return average_age_deceased_males, count_deceased_females_by_class

# Example usage
```

```
input_file = 'titanic_data.csv'  # Update this to the path of your Titanic dataset CSV file
average_age, female_class_count = map_reduce_with_pandas(input_file)
print(f"Average age of males who died: {average_age:.2f}")
print("Number of deceased females in each class:")
print(female_class_count)
```

```
Average age of males who died: 31.62
Number of deceased females in each class:
Class
3    72
2     6
1     3
Name: count, dtype: int64
```

Conclusion:

The MapReduce program for Titanic ship data analysis showcases its scalability and efficiency in handling complex tasks. By leveraging parallel processing, it efficiently calculates the average age of deceased males and determines the count of deceased females in each passenger class.

ASSIGNMENT 10

Title: Exploring MongoDB: Installation, Database Creation, and CRUD Operations.

Aim: To provide a comprehensive guide for installing MongoDB, creating databases and collections, and performing CRUD operations on documents.

Objective:
1. Understand the installation process of MongoDB and its basic configuration settings.
2. Learn how to create databases and collections in MongoDB to organize data efficiently.
3. Master the CRUD operations (Create, Read, Update, Delete) for managing documents within MongoDB collections.

Theory:
MongoDB Overview: Introduction to MongoDB as a NoSQL document-oriented database, highlighting its advantages and use cases.
Installation Process: Explanation of the steps required to download, install, and configure MongoDB on various operating systems.
Database and Collection Creation: Description of how to create databases and collections in MongoDB using the MongoDB Shell or graphical user interfaces (GUIs) like MongoDB Compass.
CRUD Operations: Overview of the four fundamental CRUD operations in MongoDB:
- Insert: Adding new documents to a collection.
- Query: Retrieving documents from a collection based on specified criteria.
- Update: Modifying existing documents in a collection.
- Delete: Removing documents from a collection.

Step 1: MongoDB Installation on Windows: Download the MongoDB Community Server from the MongoDB Download Center. Run the installer and follow the setup wizard. Add MongoDB's bin folder to the PATH environment variable for easy command-line access.
https://www.mongodb.com/try/download/community

Step 2: Create a Database and Collection:

- Switch to Your New Database:

  use myNewDatabase

- Create a Collection by Inserting a Document:

db.myNewCollection.insertOne({name: "John Doe", age: 30})

MongoDB creates the database and collection upon inserting the first document.

Step 3: CRUD Operations

- Create (Insert Document): Insert a single document:

db.myNewCollection.insertOne({name: "Jane Doe", age: 25})

- Read (Query Document): Find one document:

db.myNewCollection.findOne({name: "John Doe"})

• Update Document: Update a single document:

db.myNewCollection.update One ({name: "John Doe"}, {$set: {age: 31}})

- Delete Document: Delete a single document:

db.myNewCollection.deleteOne({name: "Bob"})

Conclusion:

The guide provides a step-by-step approach to installing MongoDB, creating databases and collections, and performing CRUD operations on documents. By mastering these fundamental operations, users can harness the power and flexibility of MongoDB for storing and managing data efficiently. This serves as a foundation for further exploration of MongoDB's advanced features and capabilities in application development and data management.