**Practical 1**

**Perform the analysis for the following:**

**a. Import the data warehouse data in Microsoft Excel and create the Pivot table and Pivot**

**Chart.**

Step 1: Open Excel → Go to Data tab → Click Get Data → From Database → From SQL Server Database.

Step 2: Enter:

Server Name: DESKTOP-010F9HT (or your server name)

Database: Sales\_DW

Step 3: Select tables (e.g., FactProductSales, DimProduct) → Click Load.

2. Create a Pivot Table

Step 1: Select the imported data → Go to Insert → PivotTable → Choose New Worksheet.

Step 2: Drag fields into areas:

Rows: Product (from DimProduct)

Columns: SalesTimeKey (or any date/time field)

Values: Sales\_Amount (sum/average)

Step 3: Customize calculations (e.g., Sum, Count, Average) in Value Field Settings.

3. Create a Pivot Chart

Step 1: Click inside the PivotTable → Go to Insert → PivotChart.

Step 2: Choose chart type (e.g., Column, Line, Pie).

Step 3: Format chart (titles, colors, legends) using Chart Tools.

4. Refresh Data (If Updated in SQL Server)

Right-click PivotTable → Refresh (or go to Data → Refresh All).

Key Shortcuts & Tips

✅ Quick PivotTable: Select data → Press Alt + N + V (Excel shortcut).

✅ Slicers for Filtering: Insert → Slicer (for interactive filtering).

✅ Group Dates: Right-click dates in PivotTable → Group → By Months/Quarters.

Final Output:

A summarized PivotTable (e.g., sales by product).

A visual PivotChart (e.g., trend analysis).

**b. Import the cube in Microsoft Excel and create the Pivot table and Pivot Chart to**

**perform data analysis.**

1. Connect Excel to SSAS Cube

Step 1: Open Excel → Data tab → Get Data → From Database → From Analysis Services.

Step 2: Enter:

Server Name: DESKTOP-010F9HT (or your SSAS server)

Credentials: Use Windows Authentication (default) or SQL login.

Step 3: Select the Cube (e.g., SalesDW2) → Click Next → Finish.

2. Choose Data View (PivotTable Report)

Select PivotTable Report → Place in New Worksheet or existing sheet → Click OK.

3. Build PivotTable from Cube

Step 1: In the PivotTable Fields pane:

Measures: Drag (e.g., Sales\_Amount) to Values.

Dimensions: Drag (e.g., Product Name, Date) to Rows/Columns.

Step 2: Use Hierarchies (e.g., Year → Quarter → Month for dates).

4. Create PivotChart

Click inside PivotTable → Insert → PivotChart → Choose type (e.g., Column, Line).

Customize with Chart Tools (titles, colors, filters).

5. Analyze with Cube Functions

Drill Down: Click +/- icons to expand/collapse hierarchies.

Slicers/Timelines: Insert from PivotTable Analyze tab for interactive filtering.

Key Features of Cube-Based PivotTables

✅ MDX Measures: Use cube calculations (e.g., [Measures].[Profit Margin]).

✅ Hierarchy Navigation: Drill into dimensions (e.g., Country → City → Store).

✅ Real-Time Refresh: Right-click → Refresh to update from SSAS.

Final Output:

Dynamic PivotTable with OLAP cube data.

Interactive PivotChart for visual analysis (e.g., sales trends by product/region).

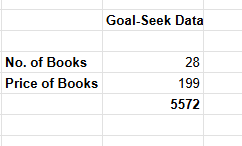
**Practical 2**

Apply the what – if Analysis for data visualization. Design and generate necessary reports

based on the data warehouse data. Use Excel

**If the practical is for Goal-Seek**

Step 1:Set up your sheet



Step 2:Keep F11 cell selected and go to **Data tab > What-If Analysis > Goal Seek**

Step 3: **Set cell:** F11  
 **To value:** 20000  
 **By changing cell:** F10

The value of the book has now changed to 20000 as you can see in F11 cel with the change in price as 20000 was the goal..

If you want to change the price of the book then:

**Set cell:** F11  
 **To value:** 20080  
 **By changing cell:** F9

The value of the book has now changed to 20080 as you can see in F11 cell with the change in book value as 20080 was the goal..

**If the practical is for Data-**

Step 1:Set up your sheet

Step 2: In a cell , enter: 2000

Step 3: Select the whole table (including blank Revenue column).

Step 4:Go to:

**Data tab > What-If Analysis > Data Table**

Step 5:In the dialog **Column input cell:** the cell where price is used in your formula

Step 6: Click OK - Excel fills in all revenue values!

**Practical 3**

**Perform the data classification using classification algorithm using R**

rainfall <- c(799,1174.8,865.1,1334.6,635.4,918.5,685.5,784.2,985,882.8,1071)

rainfall.timeseries <- ts(rainfall,start = c(2021,1),frequency = 12)

print(rainfall.timeseries)

png(file = "rainfall.png")

plot(rainfall.timeseries)

dev.off()

plot(rainfall.timeseries)

**Practical 4**

**Perform the data clustering using a clustering algorithm using R**

newiris <- iris

newiris$Species <- NULL

(kc <- kmeans(newiris,3))

table(iris$Species,kc$cluster)

plot(newiris[c("Sepal.Length","Sepal.Width")],col=kc$cluster)

points(kc$centers[,c("Sepal.Length","Sepal.Width")],col=1:3,pch=8,cex=2)

**Practical 5**

**Perform the Linear regression on the given data warehouse data using R**

x <- c(151,174,138,186,128,136,179,163,152,131)

y <- c(63,81,56,91,47,57,76,72,62,48)

relation <- lm(y~x)

png(file = "linearregression.png")

plot(y,x,col = "blue",main = "Height & Weight Regression",abline(lm(x~y)),cex=1.3, pch=16,

xlab="Weight in Kg", ylab="Height in cm")

dev.off()

**Practical 6**

**Perform the logistic regression on the given data warehouse data using R.**

Code:

quality<-read.csv("C:/quality.csv")

str(quality)

table(quality$PoorCare)

install.packages("caTools")

library(caTools)

qualityTrain = subset(quality,split==TRUE)

qualityTest =subset(quality,split==FALSE)

nrow(qualityTrain)

nrow(qualityTest)

QualityLog=glm(PoorCare~OfficeVisits + Narcotics,data=qualityTrain,family = binomial) >

summary(QualityLog)

predictTrain=predict(QualityLog,type = "response")

summary(predictTrain)

tapply(predictTrain, qualityTrain$PoorCare, mean)

table(qualityTrain$PoorCare,predictTrain>0.5)

table(qualityTrain$PoorCare,predictTrain>0.7)

table(qualityTrain$PoorCare,predictTrain<0.2)

install.packages("ROCR")

library(ROCR)

ROCRpred=prediction(predictTrain,qualityTrain$PoorCare)

ROCRperf=performance(ROCRpred,"tpr","fpr")

plot(ROCRperf)

plot(ROCRperf,colorize=TRUE)

plot(ROCRperf,colorize=TRUE,print.cutoffs.at=seq(0,1,by=0.1),text.adj=c(-0.2,10.7))

**Practical 7**

**Write a Python program to read data from a CSV file, perform simple data analysis, and**

**generate basic insights. (Use Pandas is a Python library).**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

file\_path = 'C:\WineQT.csv'

df = pd.read\_csv(file\_path)

summary\_stats = df.describe()

print("Summary Statistics")

print(summary\_stats)

correlation\_matrix = df.corr()

print("\nCorrelation Matrix: ")

print(correlation\_matrix)

quality\_distribution = df['quality'].value\_counts().sort\_index()

print("\nQuality Distribution: ")

print(quality\_distribution)

quality\_correlation = correlation\_matrix['quality'].sort\_values(ascending=False)

print("\nQuality Correlation: ")

print(quality\_correlation)

quality\_correlation = correlation\_matrix ['quality'].sort\_values (ascending=False)

print("\nCorrelation with Quality:")

print (quality\_correlation)

plt.figure (figsize=(10, 6))

plt.subplot (2, 2, 1)

sns.countplot (x='quality', data=df, palette='viridis')

plt.title('Quality Distribution')

plt.subplot(2, 2, 2)

sns.heatmap (correlation\_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)

plt.title('Correlation Matrix Heatmap')

plt.subplot (2, 2, 3)

sns.boxplot (x='quality', y='alcohol', data=df, palette='viridis')

plt.title('Alcohol vs Quality')

plt.subplot (2, 2, 4)

sns.boxplot (x='quality', y='density', data=df, palette='viridis')

plt.title('Density vs Quality')

plt.tight\_layout ()

plt.show()

**Practical 8**

**(a.Perform data visualization)**

**a. Perform data visualization using Python on any sales data.**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

def load\_data(csv\_file):

try:

data = pd.read\_csv('C:\sales\_data.csv', encoding="ISO-8859-1")

print("Data loaded successfully!")

print(data)

return data

except Exception as e:

print(f"Error loading data: {e}")

return None

def data\_summary(data):

print("\nFirst 5 rows of the data:")

print(data.head()) # Displays the first 5 rows

print("\nData Structure:")

print(data.info()) # Get information on the DataFrame such as column types and non-null

counts

print("\nStatistical summary of numeric columns:")

print(data.describe()) # Gives summary statistics for numeric columns

def plot\_sales\_over\_time(data):

data['Date'] = pd.to\_datetime(data['Date'])

sales\_by\_date = data.groupby('Date')['Sales\_Amount'].sum().reset\_index()

plt.figure(figsize=(10, 6))

sns.lineplot(x='Date', y='Sales\_Amount', data=sales\_by\_date, marker='o')

plt.title('Total Sales Over Time')

plt.xlabel('Date')

plt.ylabel('Total Sales ($)')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.show()

def plot\_sales\_by\_region(data):

plt.figure(figsize=(10, 6))

sns.boxplot(x='Region', y='Sales\_Amount', data=data)

plt.title('Sales Distribution by Region')

plt.xlabel('Region')

plt.ylabel('Sales Amount ($)')

plt.tight\_layout()

plt.show()

def plot\_sales\_vs\_quantity(data):

plt.figure(figsize=(10, 6))

sns.scatterplot(x='Quantity\_Sold', y='Sales\_Amount', data=data, hue='Product',

palette='viridis')

plt.title('Sales Amount vs Quantity Sold')

plt.xlabel('Quantity Sold')

plt.ylabel('Sales Amount ($)')

plt.tight\_layout()

plt.show()

def plot\_top\_products\_by\_sales(data):

top\_products = data.groupby('Product')['Sales\_Amount'].sum().reset\_index()

top\_products = top\_products.sort\_values('Sales\_Amount', ascending=False).head(10)

plt.figure(figsize=(10, 6))

sns.barplot(x='Sales\_Amount', y='Product', data=top\_products, palette='Blues\_d')

plt.title('Top 10 Products by Sales Amount')

plt.xlabel('Sales Amount ($)')

plt.ylabel('Product')

plt.tight\_layout()

plt.show()

def main():

csv\_file = 'sales\_data.csv' # Replace with your actual CSV file path

data = load\_data(csv\_file)

if data is not None:

data\_summary(data)

plot\_sales\_over\_time(data)

plot\_sales\_by\_region(data)

plot\_sales\_vs\_quantity(data)

plot\_top\_products\_by\_sales(data)

if \_\_name\_\_ == "\_\_main\_\_":

main()

b. Perform data visualization using PowerBI on any sales data.

**Step-by-Step Guide for Analyzing Sales Data in Power BI**

### **Step 1: Install and Open Power BI**

1. **Download & Install Power BI Desktop**
   * If you haven't installed it yet, download it from [Power BI Download](https://powerbi.microsoft.com/en-us/downloads/).
   * Install and open **Power BI Desktop**.

### **Step 2: Load the Sales Data**

1. Open **Power BI Desktop**.
2. Click on **"Home" > "Get Data" > "Text/CSV"**.
3. Browse and select the **sales\_data\_sample.csv** file you uploaded.
4. Click **Open**, then click **Load** (you can check the data preview before loading).

### **Step 3: Transform Data in Power Query Editor**

Before building reports, let's clean and format the data:

1. Click on **"Transform Data"** to open **Power Query Editor**.
2. Review columns and check for missing values:
   * **ADDRESSLINE2, STATE, TERRITORY** have missing values, so you can remove them or fill in with "Unknown".
   * To remove a column: **Right-click** on the column header > Click **Remove**.
3. Convert **ORDERDATE** column to **Date** format:
   * Select **ORDERDATE** column.
   * In the top menu, click **Transform > Data Type > Date**.
4. Ensure **SALES, PRICEEACH** are in **Decimal Number** format.
5. Click **"Close & Apply"**.

**Step 4: Create Basic Visualizations**

#### **1. Sales Performance Overview**

● **Total Sales & Orders**

1. Go to **Report View** (bottom left panel).

2. Drag **SALES** to a **Card Visual** (this shows total revenue).

3. Drag **ORDERNUMBER** to another **Card Visual** (shows total orders).

● **Sales Trend Over Time**

1. Drag **ORDERDATE** to **X-Axis** in a **Line Chart**.

2. Drag **SALES** to **Y-Axis** (shows sales over time).

#### **2. Sales by Product Line**

1. Insert a **Bar Chart**.
2. Drag **PRODUCTLINE** to **X-Axis**.
3. Drag **SALES** to **Y-Axis** (shows revenue per product category).

#### **3. Sales by Country**

1. Insert a **Map Visual** (Globe icon).
2. Drag **COUNTRY** to **Location** field.
3. Drag **SALES** to **Values** field (shows sales by country).

#### **4. Order Status Breakdown**

1. Insert a **Pie Chart**.
2. Drag **STATUS** to **Legend**.
3. Drag **ORDERNUMBER** to **Values** (shows percentage of orders in each status).

### **Step 5: Add Filters and Slicers**

● Click on **"Slicer"** and add **YEAR\_ID** to filter by year.

● Add **CUSTOMERNAME** to filter by customer.

● Add **DEALSIZE** to filter by deal size.

### **Step 6: Save & Publish Report**

1. Click **File > Save As** and save the Power BI report.
2. Click **Publish** to share on **Power BI Service** (if needed).

**Steps to Perform Data Visualization in Power BI**

1. **Import the Data:**
   * Open **Power BI Desktop**.
   * Click on **"Get Data" > "Excel"** and select the uploaded file.
   * Load the relevant sheets into Power BI.
2. **Data Preparation:**
   * Check for missing values or inconsistent data.
   * Apply transformations (if needed) using **Power Query Editor**.
3. **Create Key Visuals:**
   * **Sales Trends:** Use a **Line Chart** to show sales performance over time.
   * **Regional Sales Analysis:** Use a **Map Visual** to display sales by region.
   * **Top Products Sold:** Use a **Bar Chart** to highlight best-selling products.
   * **Revenue Breakdown:** Use a **Pie Chart** or **Treemap** to show sales by category.
   * **Customer Segmentation:** Use **Clustered Bar Charts** to group customers based on sales.
4. **Add Filters and Slicers:**
   * Include **Date Slicers** to filter data by month, quarter, or year.
   * Use **Dropdown Filters** for region, product category, or customer segment.
5. **Enhance with DAX Measures:**
   * Create calculated measures such as:

o Total Sales = SUM(Sales[Revenue])

o Sales Growth = ( [Total Sales] - PREVIOUSMONTH([Total Sales]) ) / PREVIOUSMONTH([Total Sales])

* + Use **KPIs** for profit margin and sales targets.

1. **Publish and Share:**
   * Save the report and publish it to **Power BI Service**.
   * Share with stakeholders via dashboards.

**Practical 9**

**Create the Data staging area for the selected database using SQL.**

1. Load Data into Power BI

Open Power BI Desktop → Home → Get Data → Excel/CSV/SQL Server.

Select your source file (e.g., sales\_data\_sample.xlsx) → Transform Data (to clean before loading).

2. Clean & Transform Data in Power Query Editor

Remove Unnecessary Columns: Right-click → Remove Columns.

Handle Missing Values:

Replace nulls: Transform → Replace Values (e.g., "Unknown" for text, 0 for numbers).

Remove blank rows: Home → Remove Rows → Remove Blank Rows.

Standardize Text:

Trim spaces: Transform → Format → Trim.

Proper case: Use Text.Proper([Column]) in Custom Column.

Fix Data Types: Ensure dates, numbers, and text are correctly assigned (e.g., Order\_Date as Date).

Duplicate Your Source Table (to make a staging table)

In the left panel (Queries list), right-click your source table

Click Duplicate

3. Create Staging Tables

Fact Table (e.g., FactSales):

Contains transactional data: Order\_ID, Product\_ID, Quantity, Sales\_Amount, Date.

Dimension Tables:

DimProducts: Product\_ID, Product\_Name, Category.

DimCustomers: Customer\_ID, Customer\_Name, Region.

DimDates: Create a date table with Date, Year, Quarter, Month.

4. Build a Date Table (If Missing)

let

StartDate = #date(2020, 1, 1),

EndDate = #date(2030, 12, 31),

DateList = List.Dates(StartDate, Duration.Days(EndDate - StartDate) + 1, #duration(1, 0, 0, 0)),

DateTable = Table.FromList(DateList, Splitter.SplitByNothing(), {"Date"}),

#"Added Columns" = Table.AddColumn(DateTable, "Year", each Date.Year([Date])),

#"Added Month" = Table.AddColumn(#"Added Columns", "Month", each Date.Month([Date]))

in

#"Added Month"

5. Set Relationships in Data Model

Go to Model View → Drag to link:

FactSales[Customer\_ID] → DimCustomers[Customer\_ID]

FactSales[Product\_ID] → DimProducts[Product\_ID]

FactSales[Date] → DimDates[Date]

6. Validate Data Quality

DAX Measures:

Total Sales = SUM(FactSales[Sales\_Amount])

Missing Customers = COUNTROWS(FILTER(FactSales, ISBLANK(FactSales[Customer\_ID])))

7. Save & Publish

File → Save (.pbix).

Publish to Power BI Service (optional for sharing).

Key Benefits of Data Staging

✅ Consistency: Clean, standardized data for accurate reporting.

✅ Performance: Optimized tables improve query speed.

✅ Scalability: Easily add new data sources or dimensions.

Output: A structured dataset ready for dashboards, reports, or further analysis in Power BI or SQL.

**Practical 10**

**Create the cube with suitable dimension and fact tables based on ROLAP, MOLAP and**

**HOLAP model.**

1. Launch SQL Server Data Tools (SSDT) & Create Project

Open SQL Server Data Tools (SSDT) → File → New → Project.

Select Business Intelligence → Analysis Services Project.

Name the project (e.g., Sales\_Analysis\_Cube) → Click OK.

2. Connect to Data Source (SQL Server Database)

In Solution Explorer, right-click Data Sources → New Data Source.

Click New → Set up a connection to your database:

Server Name: DESKTOP-010F9HT (or your SQL Server instance).

Database: Sales\_DW.

Test connection → Click OK.

Choose authentication (Windows or SQL) → Finish.

3. Create Data Source View (DSV)

Right-click Data Source Views → New Data Source View.

Select the Sales\_DW data source → Click Next.

Add tables:

Fact Table: FactProductSales.

Dimension Tables: DimProduct, DimDate, DimCustomer, etc.

Click Finish to generate the DSV.

4. Build the Cube Using Cube Wizard

Right-click Cubes → New Cube → Use existing tables.

Select the Fact Table (FactProductSales) → Click Next.

Choose Measures (e.g., Sales\_Amount, Quantity) → Click Next.

Select Related Dimensions (e.g., DimProduct, DimDate) → Click Next.

Name the cube (e.g., Sales\_Cube) → Click Finish.

5. Define Dimensions & Hierarchies

Double-click dimensions (e.g., DimDate) in Solution Explorer.

Drag attributes (e.g., Year, Quarter, Month) to Hierarchies pane.

Set attribute relationships (e.g., Year → Quarter → Month).

6. Deploy & Process the Cube

Right-click the project → Properties → Set Deployment Server (SSAS instance name).

Right-click project → Deploy (sends cube to SSAS server).

After deployment, right-click cube → Process → Full Process (loads data).

7. Verify in SQL Server Management Studio (SSMS)

Open SSMS → Connect to Analysis Services.

Navigate to Databases → Your cube (Sales\_Cube) should appear.

Browse data by right-clicking cube → Browse.

Key Outputs

✅ Cube Structure: Measures (e.g., Sales, Profit) + Dimensions (Product, Time).

✅ Hierarchies: Drill-down paths (e.g., Year → Quarter → Month).

✅ MDX Support: Enables complex queries (e.g., YoY growth calculations).

Tips

🔹 Optimize Performance: Partition large fact tables.

🔹 Use Aggregations: Pre-calculate summaries for faster queries.

🔹 Security: Configure roles in SSAS to restrict data access.