Script

-- Step 1: Create the Database

CREATE DATABASE SalesDB;

GO

-- Use the Database

USE SalesDB;

GO

-- Step 2: Create the Table

CREATE TABLE SalesData (

OrderID INT PRIMARY KEY,

OrderDate DATE,

Region VARCHAR(50),

Product VARCHAR(100),

Category VARCHAR(50),

SalesAmount DECIMAL(10,2),

Quantity INT,

Profit DECIMAL(10,2)

);

GO

-- Step 3: Insert Sample Data

INSERT INTO SalesData (OrderID, OrderDate, Region, Product, Category, SalesAmount, Quantity, Profit)

VALUES

(101, '2024-01-01', 'North', 'Laptop', 'Electronics', 1200, 2, 200),

(102, '2024-01-02', 'South', 'Smartphone', 'Electronics', 800, 3, 150),

(103, '2024-01-03', 'East', 'Washing Machine', 'Appliances', 1500, 1, 300),

(104, '2024-01-04', 'West', 'Refrigerator', 'Appliances', 2000, 1, 400),

(105, '2024-01-05', 'North', 'TV', 'Electronics', 1800, 2, 350),

(106, '2024-01-06', 'South', 'Microwave', 'Appliances', 700, 1, 100),

(107, '2024-01-07', 'East', 'Laptop', 'Electronics', 1300, 1, 250),

(108, '2024-01-08', 'West', 'Air Conditioner', 'Appliances', 2200, 1, 500),

(109, '2024-01-09', 'North', 'Smartphone', 'Electronics', 900, 2, 180),

(110, '2024-01-10', 'South', 'TV', 'Electronics', 1900, 1, 400);

GO

-- Step 4: Verify the Data

SELECT \* FROM SalesData;

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

rainfall <- c(799,1174.8,865.1,1334.6,635.4,918.5,685.5,998.6,784.2,985,882.8,1071)

# Convert it to a time series object.

rainfall.timeseries <- ts(rainfall,start = c(2012,1),frequency = 12)

# Print the timeseries data

print(rainfall.timeseries)

# Give the chart file a

# Plot a graph of the time series.

plot(rainfall.timeseries)

# Save the file.

dev.off()

plot(rainfall.timeseries)

**Perform the data clustering using clustering algorithm.**

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)

dev.off()

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

Linear regression on the given data warehouse data using R/Python.

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

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

# Apply the lm() function.

relation <- lm(y~x)

print(relation)

# Values of height

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

# Values of width

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

# Apply the lm() function

relation <- lm(y~x)

print(summary(relation))

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

# The response vector.

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

# Apply the lm() function.

relation <- lm(y~x)

a <- data.frame(x = 170)

result <- predict(relation,a)

print(result)

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)

# Give the chart file a name.

png(file = "linearregression.png")

# Plot the chart.

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")

# Save the file.

dev.off()

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")

# PRATICAL-6

AIM:- Perform the logistic regression on the given data warehouse data using R/Python.

# Step 1: Install and load required packages

if (!require(readxl)) install.packages('readxl', dependencies = TRUE)

if (!require(caTools)) install.packages('caTools', dependencies = TRUE)

if (!require(ROCR)) install.packages('ROCR', dependencies = TRUE)

library(readxl)

library(caTools)

library(ROCR)

# Step 2: Load the Excel data using read\_excel

quality <- read\_excel("C:/Users/sumit singh/Downloads/quality\_data.xlsx")

# Step 3: Explore the data

str(quality)

table(quality$PoorCare)

# Step 4: Split the data into training and testing sets

set.seed(88)

split <- sample.split(quality$PoorCare, SplitRatio = 0.75)

qualityTrain <- subset(quality, split == TRUE)

qualityTest <- subset(quality, split == FALSE)

# Step 5: Perform logistic regression

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

summary(QualityLog)

# Step 6: Predict on training data

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

summary(predictTrain)

# Step 7: Evaluate the model

confusion\_matrix <- table(qualityTrain$PoorCare, predictTrain > 0.5)

print(confusion\_matrix)

# Step 8: Calculate accuracy

correct\_predictions <- sum(diag(confusion\_matrix))

total\_predictions <- sum(confusion\_matrix)

accuracy\_train <- correct\_predictions / total\_predictions

print(paste("Training Accuracy:", accuracy\_train))

# Step 9: Plot ROC Curve

ROCRpred <- prediction(predictTrain, qualityTrain$PoorCare)

ROCRperf <- performance(ROCRpred, "tpr", "fpr")

plot(ROCRperf, colorize = TRUE, print.cutoffs.at = seq(0, 1, 0.1), text.adj = c(-0.2, 1.7))

AIM:- 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

# Correct file path with proper quotes

file\_path = r"C:\Users\sumit singh\Documents\studentData.csv"

# Read the CSV file

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

# Display the first 5 rows of the dataset

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

print(df.head())

# Display dataset summary

print("\nDataset Summary:")

print(df.info())

# Display basic statistics

print("\nBasic Statistics:")

print(df.describe())

# Check for missing values in each column

print("\nMissing Values in Each Column:")

print(df.isnull().sum())

# Display column names

print("\nColumn Names:")

print(df.columns)

# If 'Salary' column exists, display the average salary

if 'Salary' in df.columns:

print(f"\nAverage Salary: {df['Salary'].mean():.2f}")

# If 'Age' column exists, display the youngest and oldest person's age

if 'Age' in df.columns:

print(f"\nYoungest Person's Age: {df['Age'].min()}")

print(f"Oldest Person's Age: {df['Age'].max()}")

# If 'Gender' column exists, display the gender distribution

if 'Gender' in df.columns:

print("\nGender Distribution:")

print(df['Gender'].value\_counts())

# Final message

print("\n⬛ Data Analysis Completed Successfully!")

8 A . Perform data visualization using Python on any sales data.

# Step 1: Install Required Libraries

# Run this command in terminal if not installed: pip install pandas numpy matplotlib seaborn

# Step 2: Import Libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# Step 3: Generate Sample Sales Data

dates = pd.date\_range(start='2023-01-01', periods=100, freq='D')

categories = ['Electronics', 'Clothing', 'Groceries', 'Furniture']

data = {

'Date': np.random.choice(dates, 200),

'Category': np.random.choice(categories, 200),

'Sales\_Amount': np.random.randint(100, 1000, 200),

'Units\_Sold': np.random.randint(1, 20, 200)

}

df = pd.DataFrame(data)

# Convert Date column to datetime format

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

# Summary Statistics

print("Summary Statistics:")

print(df.describe())

# Step 4: Visualizations

# Sales Trend Over Time

plt.figure(figsize=(12, 6))

sns.lineplot(data=df.groupby('Date')['Sales\_Amount'].sum().reset\_index(), x='Date', y='Sales\_Amount')

plt.title('Sales Trend Over Time')

plt.xlabel('Date')

plt.ylabel('Total Sales')

plt.xticks(rotation=45)

plt.grid(True)

plt.show()

# Sales by Category

plt.figure(figsize=(10, 5))

sns.barplot(data=df.groupby('Category', as\_index=False)['Sales\_Amount'].sum(), x='Category', y='Sales\_Amount', palette='viridis')

plt.title('Total Sales by Category')

plt.xlabel('Category')

plt.ylabel('Sales Amount')

plt.show()

# Distribution of Sales Amounts

plt.figure(figsize=(10, 5))

sns.histplot(df['Sales\_Amount'], bins=20, kde=True, color='blue')

plt.title('Distribution of Sales Amounts')

plt.xlabel('Sales Amount')

plt.ylabel('Frequency')

plt.grid(True)

plt.show()

# Scatter Plot: Sales Amount vs. Units Sold

plt.figure(figsize=(10, 5))

sns.scatterplot(data=df, x='Units\_Sold', y='Sales\_Amount', hue='Category', palette='coolwarm')

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

plt.xlabel('Units Sold')

plt.ylabel('Sales Amount')

plt.grid(True)

plt.show()

pip install pandas numpy matplotlib seaborn

AIM: Create the Data staging area for the selected database using SQL

**-- Step 1: Create Staging Database**

**CREATE DATABASE IF NOT EXISTS Sales\_Staging;**

**USE Sales\_Staging;**

**-- Step 2: Create Staging Table**

**CREATE TABLE IF NOT EXISTS Staging\_Sales (**

**SalesID INT PRIMARY KEY,**

**OrderDate DATE,**

**ProductName VARCHAR(100),**

**Category VARCHAR(50),**

**Region VARCHAR(50),**

**SalesAmount DECIMAL(10,2),**

**Profit DECIMAL(10,2),**

**Quantity INT,**

**LoadDate DATETIME DEFAULT CURRENT\_TIMESTAMP,**

**BatchID INT**

**);**

**-- Step 3: Load Raw Data into the Staging Table**

**INSERT INTO Staging\_Sales (SalesID, OrderDate, ProductName, Category, Region, SalesAmount, Profit, Quantity, BatchID)**

**VALUES**

**(1, '2024-01-01', 'Laptop', 'Electronics', 'North', 1200.00, 200.00, 3, 101),**

**(2, '2024-01-02', 'Smartphone', 'Electronics', 'South', 800.00, 150.00, 2, 101),**

**(3, '2024-01-03', 'Tablet', 'Electronics', 'East', 600.00, 100.00, 5, 101);**

**-- Step 4: Perform Data Cleansing & Transformation**

**-- Remove Duplicates**

**DELETE FROM Staging\_Sales**

**WHERE SalesID NOT IN (**

**SELECT MIN(SalesID)**

**FROM Staging\_Sales**

**GROUP BY OrderDate, ProductName, Region**

**);**

**-- Handle Null Values**

**UPDATE Staging\_Sales**

**SET Profit = 0**

**WHERE Profit IS NULL;**

**-- Step 5: Transfer Clean Data to the Final Sales Table**

**CREATE TABLE IF NOT EXISTS Final\_Sales (**

**SalesID INT PRIMARY KEY,**

**OrderDate DATE,**

**ProductName VARCHAR(100),**

**Category VARCHAR(50),**

**Region VARCHAR(50),**

**SalesAmount DECIMAL(10,2),**

**Profit DECIMAL(10,2),**

**Quantity INT**

**);**

**INSERT INTO Final\_Sales (SalesID, OrderDate, ProductName, Category, Region, SalesAmount, Profit, Quantity)**

**SELECT SalesID, OrderDate, ProductName, Category, Region, SalesAmount, Profit, Quantity**

**FROM Staging\_Sales;**

**-- Step 6: Archive or Delete Processed Data**

**DELETE FROM Staging\_Sales WHERE BatchID = 101;**

**-- End of Script**