ANNAI MIRA COLLEGE OF ENGINEERING AND TECHNOLOGY

NH-46, Chennai-Bengaluru National Highways, Arappakkam,

Ranipet-632517, Tamil Nadu, India

Telephone: 04172-292925 Fax: 04172-292926

Email:amcet.rtet@gmail.com/info@amcet.in Web: www.amcet.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CCS341 – DATA WAREHOUSING LABORATORY

Name : …….

Register Number :…….

Year & Branch :…….

Semester :……

Academic Year :…….....

ANNAI MIRA COLLEGE OF ENGINEERING AND TECHNOLOGY

NH-46, Chennai-Bengaluru National Highways, Arappakkam,

Ranipet-632517, TamilNadu, India

Telephone: 04172-292925 Fax: 04172-292926

CERTIFICATE

This is to Certify that the Bonafide record of the practical work done by…… Register Number……. of IIIrd year B.E (Computer Science and Engineering) submitted for the B.E-Degree practical examination(VIth Semester) in **CCS341 – DATA WAREHOUSING LABORATORY** during the academic year **2023 – 2024.**

Staff in –Charge **Head of the Department**

Submitted for the practical examination held on ----------

Internal Examiner **External Examiner**

TABLE OF CONTENT

EX NO: 01	DATA EXPLORATION AND INTEGRATION WITH WEKA
DATE:	

Aim:

The goal of this lab is to install and familiarize with Weka. To demonstrate the available features in pre-processing, we will use the Weather dataset.

Procedure:

Step1: Download and install Weka.

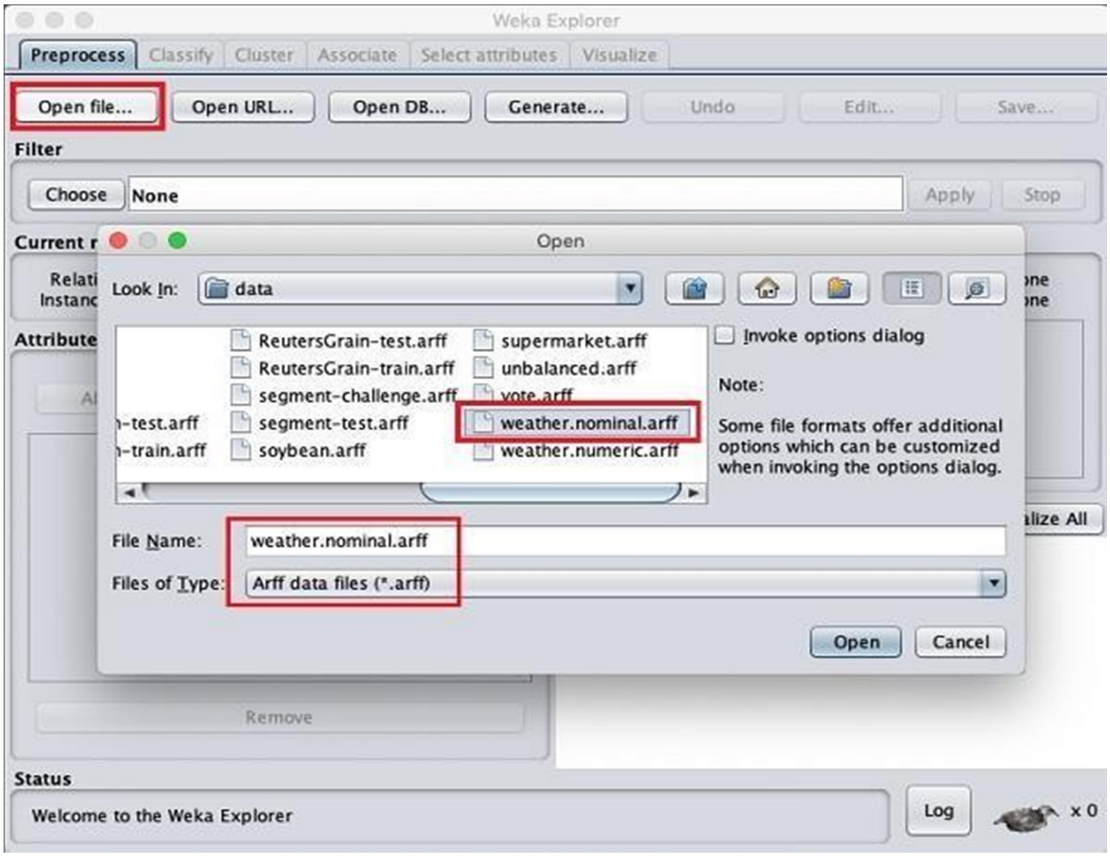
Step2: Open Weka and have a look at the interface. It is an open-source project written in Java from the University of Waikato.

Step 3: Click on the Explorer button on the right side.

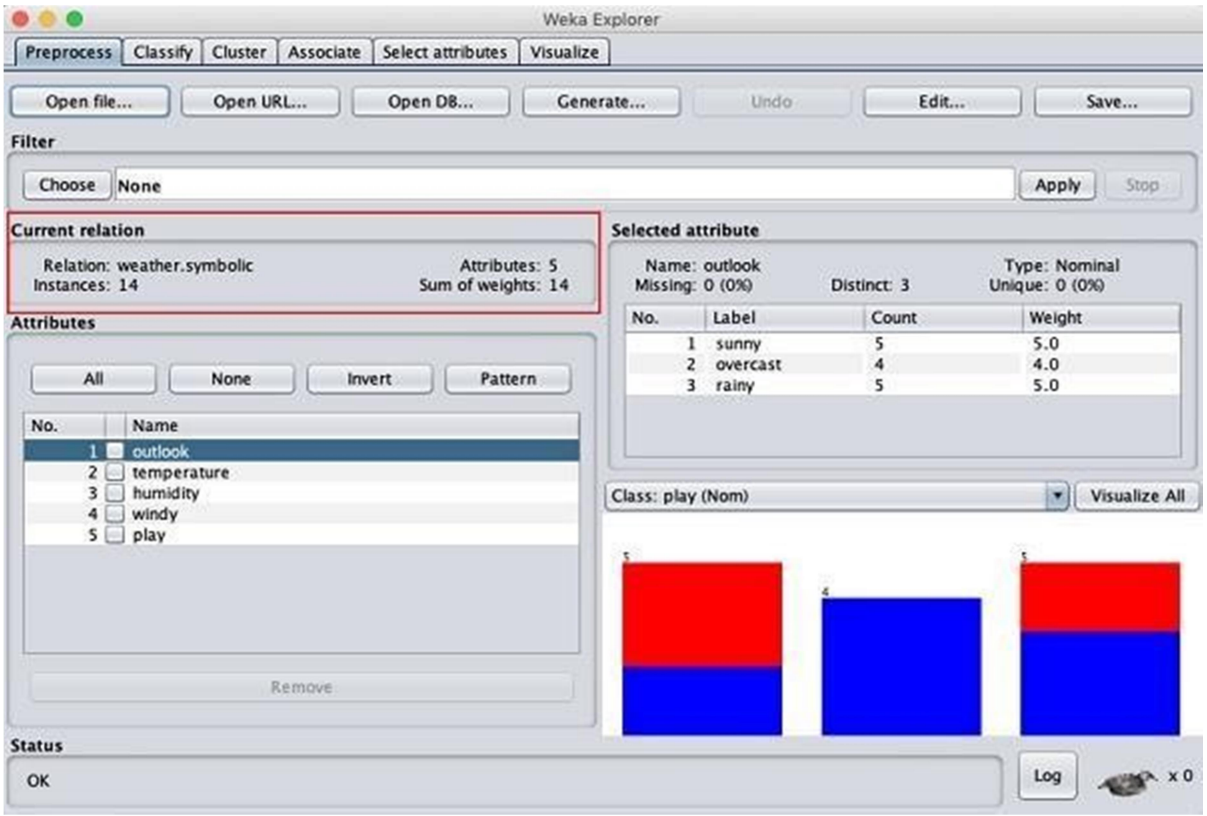
Step 4: Weka comes with a number of small datasets. Those files are located at C:\Program Files\Weka3-8 (If it is installed at this location. Or else, search for Weka-3-8 to find the installation location).

In this folder, there is a subfolder named 'data'. Open that folder to see all files that comes with Weka.

Using the ... **Open file** option under the **Pre-process** tag select the **weather-nominal.arff** file.



When opening the file, the screen looks like this.



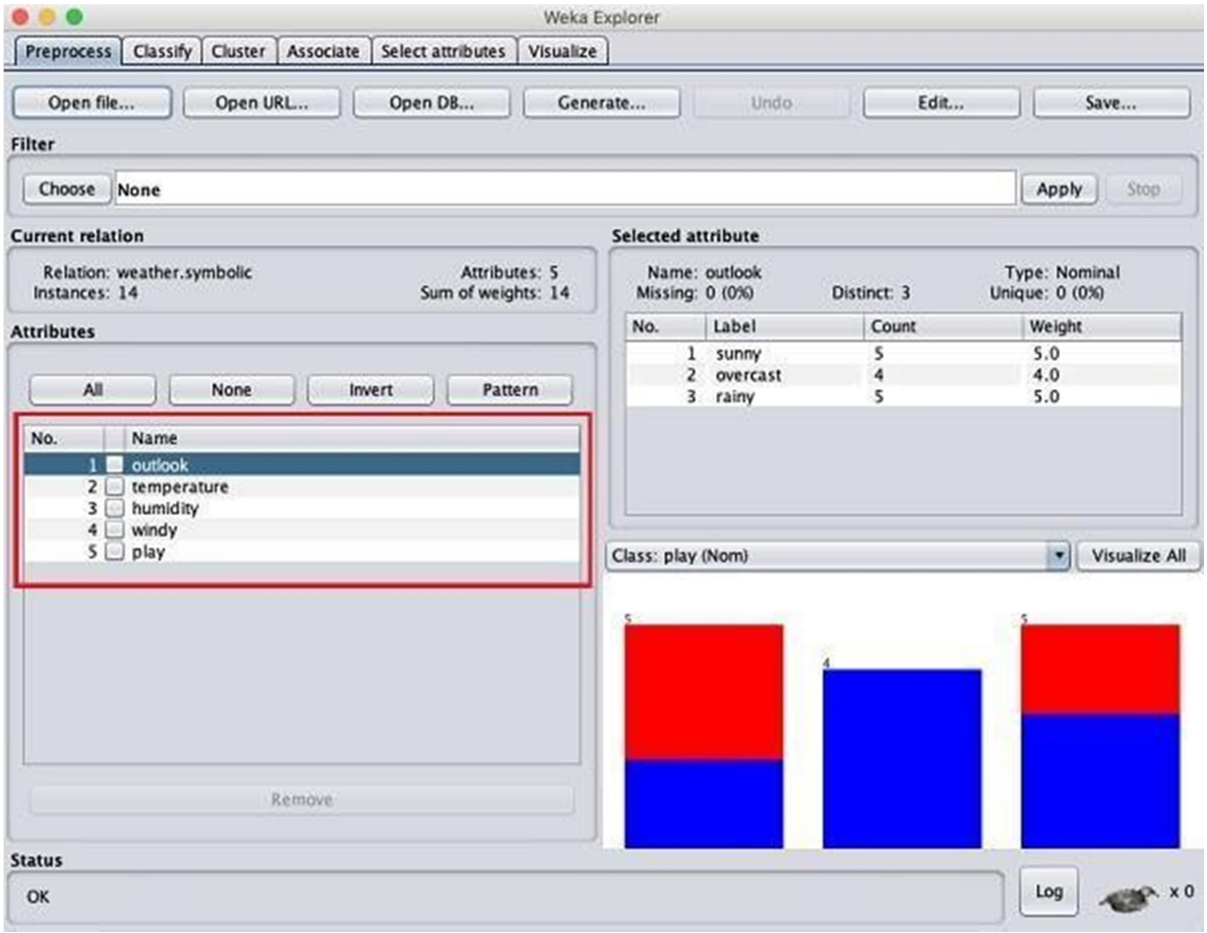
Step 5: Check different tabs to familiarize with the tool.

Understanding Data

Let us first look at the highlighted **Current relation** sub window. It shows the name of the dataset that is currently loaded. You can infer two points from this sub window.

- There are 14 instance – the number of rows in the table.
- The table contains 5 attributes – the fields, which are discussed in the upcoming sections.

On the left side, notice the **Attributes** sub window that displays the various fields in the database.



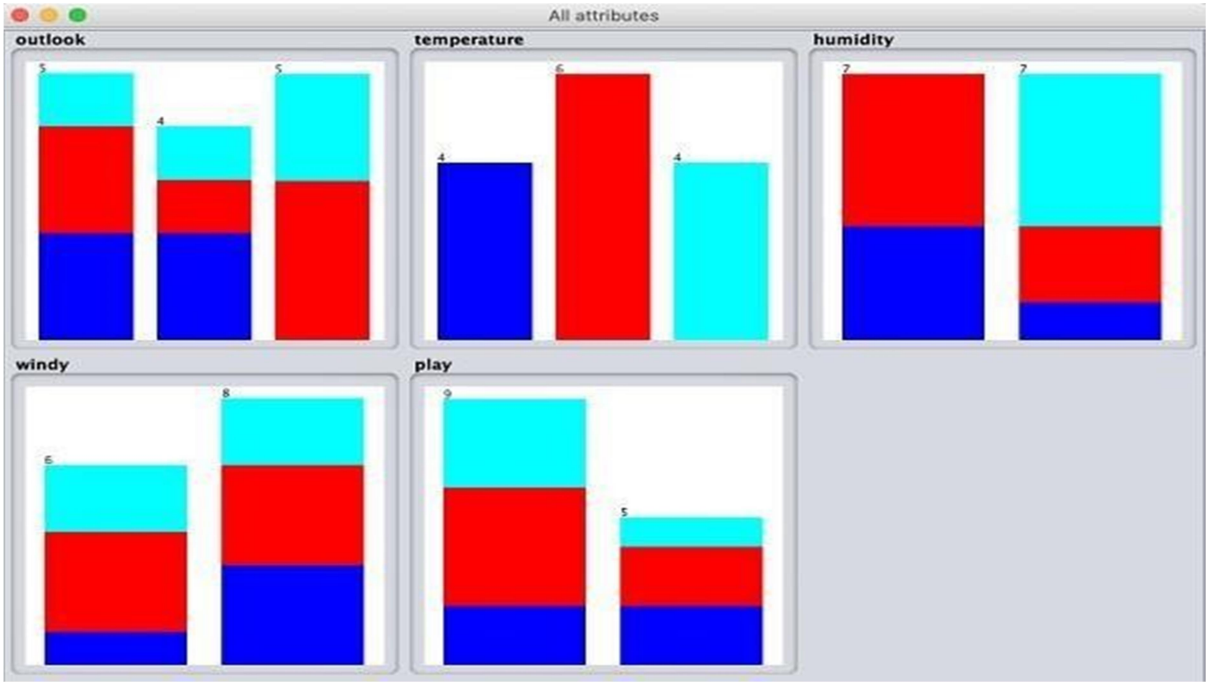
In the **Selected Attribute** sub window, you can observe the following −

- The name and the type of the attribute are displayed.
- The type for the **temperature** attribute is **Nominal**.
- The number of **Missing** values is zero.

- There are three distinct values with no unique value.

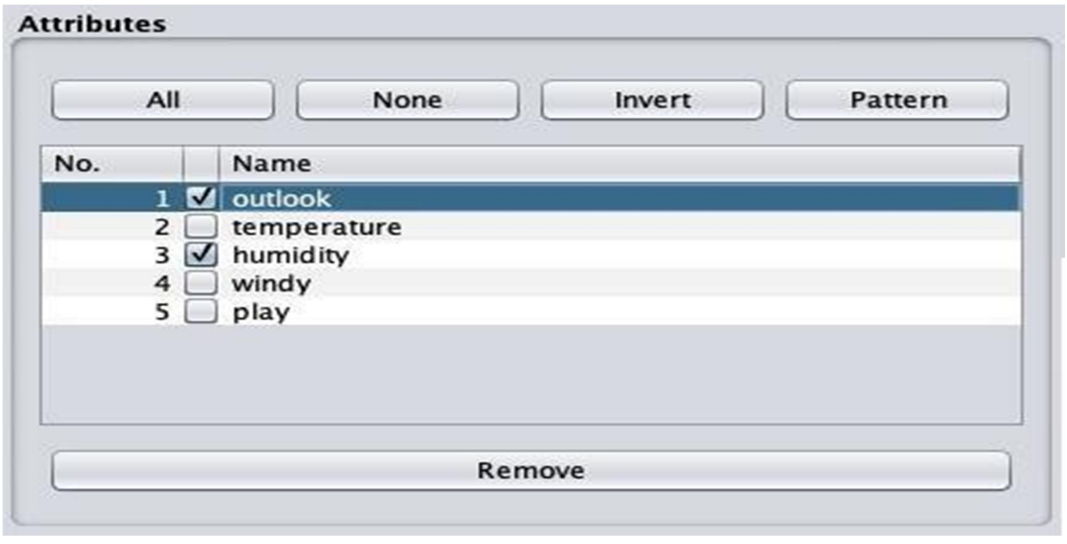
- The table underneath this information shows the nominal values for this field as hot, mild and cold.
- It also shows the count and weight in terms of a percentage for each nominal value.

At the bottom of the window, you see the visual representation of the **class** values. If you click on the **Visualize All** button, you will be able to see all features in one single window as shown here



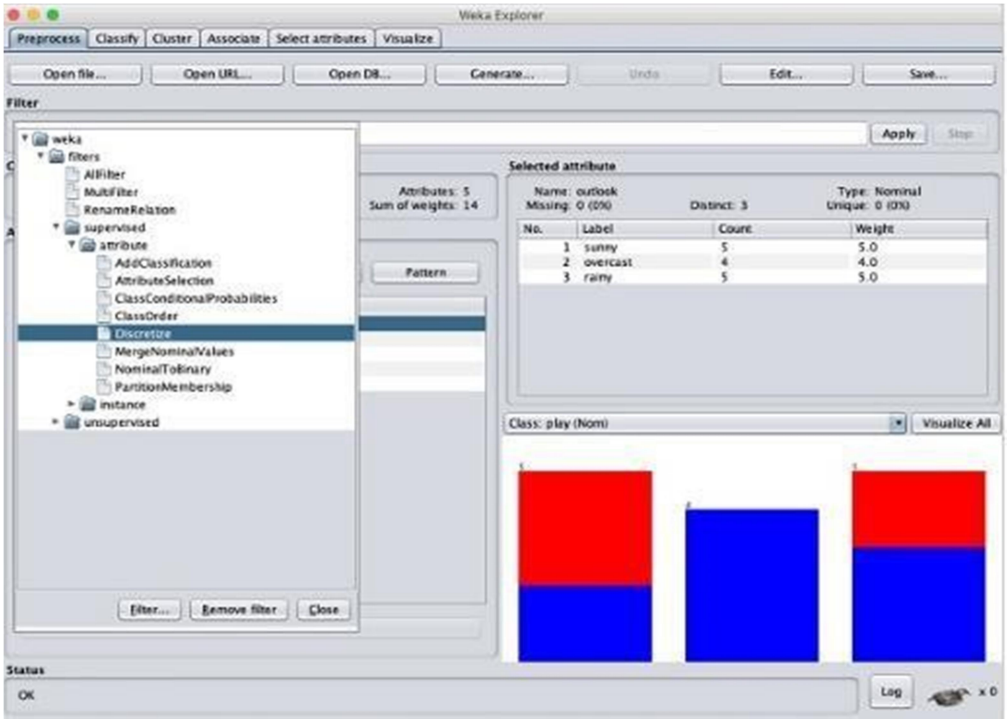
Removing Attributes:

Many a time, the data that you want to use for model building comes with many irrelevant fields. For example, the customer database may contain his mobile number which is relevant in analysing his credit rating.



Applying Filters:

Some of the machine learning techniques such as association rule mining requires categorical data. To illustrate the use of filters, we will use **weather-numeric.arff** database that contains two **numeric** attributes - **temperature** and **humidity**.



Result:

Thus, the data exploration and integration with WEKA is successfully executed.

EX NO: 02	APPLY WEKA TOOL FOR DATA VALIDATION
DATE:	

Aim:

To implement data validation using Weka.

Procedure:

Step 1: Launch Weka Explorer

- Open Weka and select the "Explorer" from the Weka GUI Chooser.

Step 2: Load the dataset

- Click on the "Open file" button and select "datasets" > "iris.arff" from the Weka installation directory. This will load the Iris dataset.

Step 3: Split your data into training and testing sets. Under the "Classify" tab, click on the "Choose" button next to the "Test options" area and select a testing method. Weka offers options like cross-validation, percentage split, and user-defined test set. Configure the options according to your needs.

Step 4: Select a classifier algorithm. Weka offers a wide range of algorithms for classification, regression, clustering, and other tasks. Under the "Classify" tab, click on the "Choose" button next to the "Classifier" area and choose an algorithm. Configure its parameters, if needed.

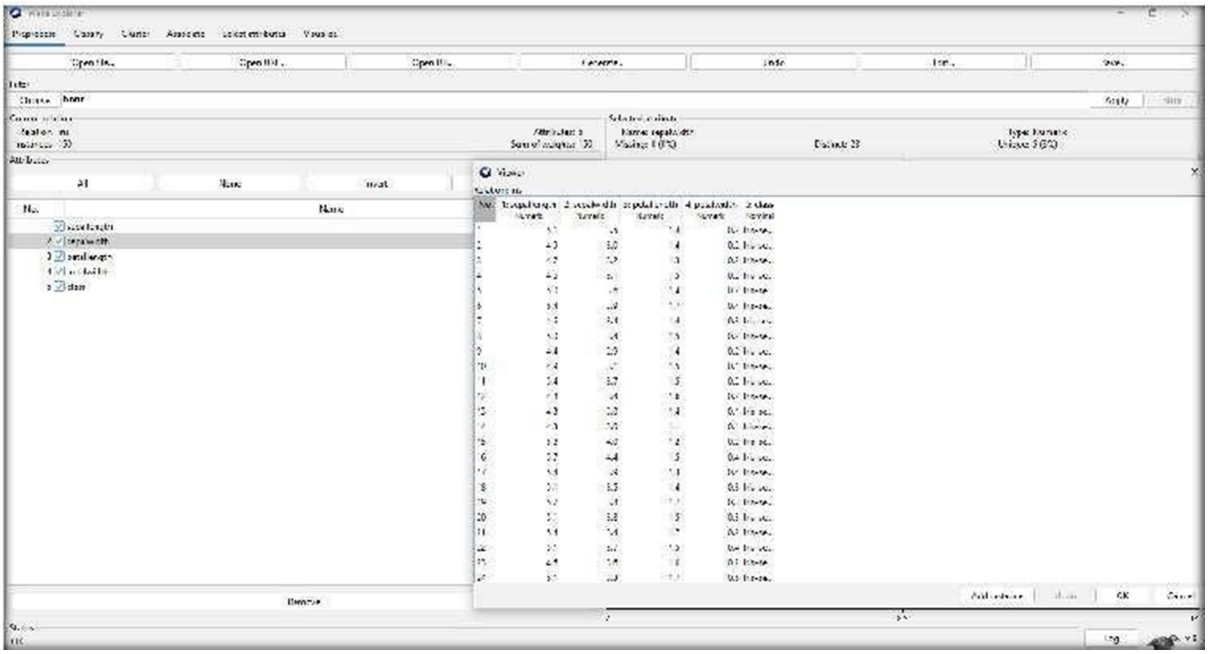
Step 5: Click on the "Start" button under the "Classify" tab to run the training and testing process. Weka will train the model on the training set and test its performance on the testing set using the selected algorithm.

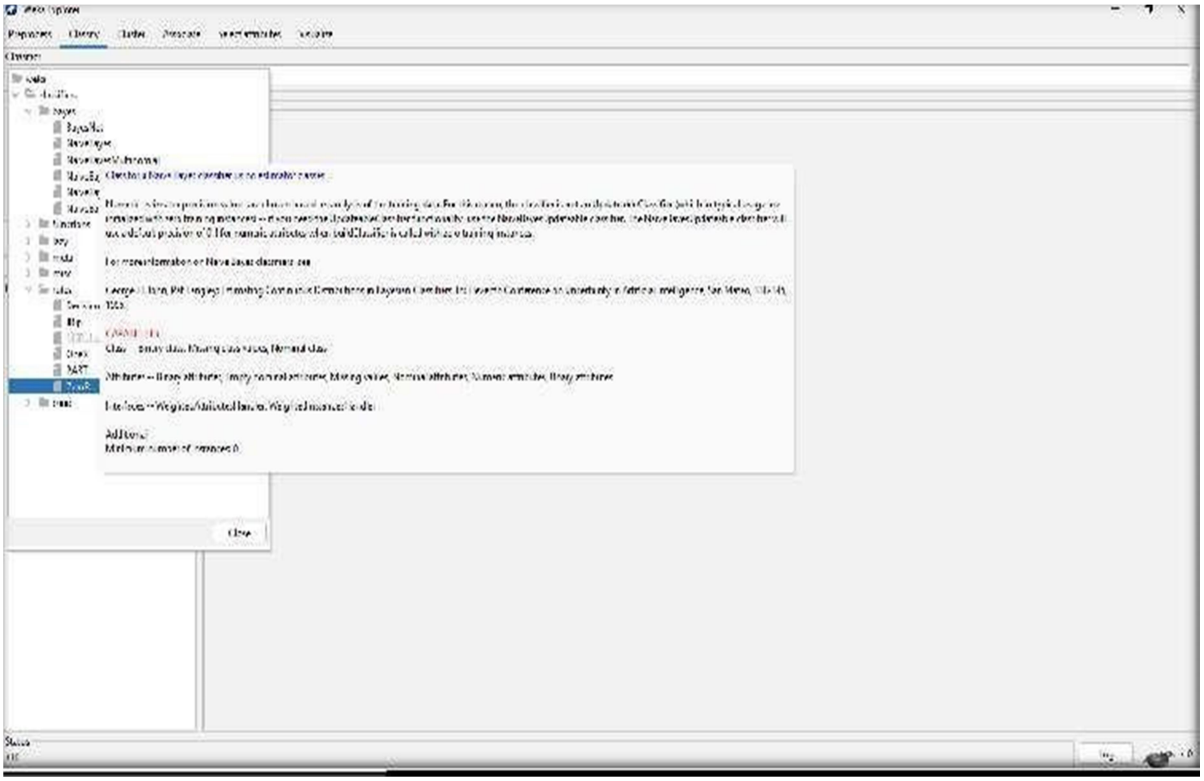
Validation Techniques:

Cross-Validation: Go to the "Classify" tab and choose a classifier. Then, under the "Test options," select the type of cross-validation you want to perform (e.g., 10-fold cross validation). Click "Start" to run the validation.

Train-Test Split: You can also split your data into a training set and a test set. Use the "Supervised" tab to train a model on the training set and evaluate its performance on the test set.

Output





Step 6: Evaluate the model's performance. Once the process finishes, Weka will display various performance measures like accuracy, precision, recall, and ROC curve (for classification tasks) or RMSE and MAE (for regression tasks). These measures can be found in the "Result list" on the right side of the window.

Step 7: Analyse the results and interpret them. Examine the performance measures to assess the model's quality and suitability for your dataset. Compare different models or validation methods if you have tried more than one.

Step 8: Repeat steps 4-7 with different algorithms or validation methods if desired. This will help you compare the performance of different models and choose the best one.

Result:

Thus, the simple data validation and testing dataset using Weka was implemented.

EX NO: 03	PLAN THE ARCHITECTURE FOR REAL TIME APPLICATION
DATE:	

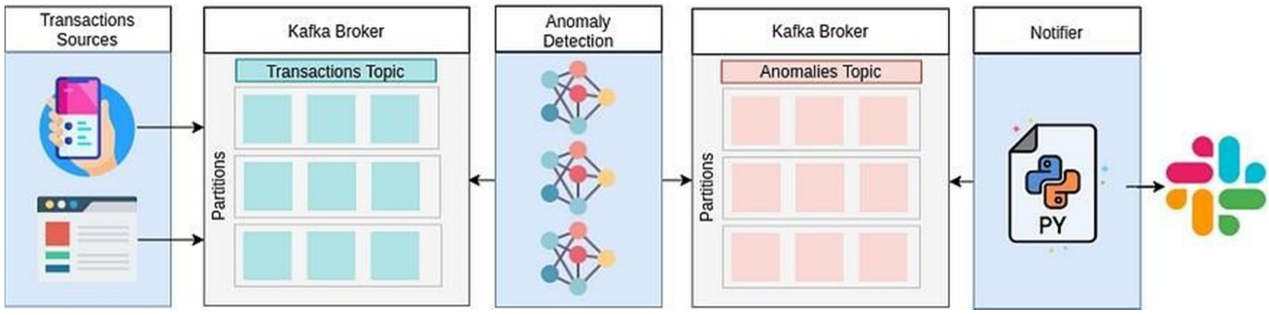
Aim:

To make real-time predictions with incoming stream data from Apache Kafka, and to implement notification messages for credit card transactions, GPS logs, system consumption metrics.

Project ideas:

- Train an anomaly detection algorithm using unsupervised machine learning.
- Create a new data producer that sends the transactions to a Kafka topic.
- Read the data from the Kafka topic to make the prediction using the trained ml model.
- If the model detects that the transaction is not an inlier, send it to another Kafka topic.
- Create the last consumer that reads the anomalies and sends an alert to a Slack channel.

Architecture:



Procedure:

Step 1: Project Structure:

i) First, Check the Settings.Py; It Has Some Variables to Set, Like the Kafka Broker Host and Port; Leave the Ones by Default (Listening On Localhost And Default Ports Of Kafka And Zookeeper).

ii) The Streaming/Utils.Py File Contains the Configurations to Create Kafka Consumers and Producers.

iii) Install The Requirements

Step 2: Train the Model

i) To generate random data; it will have two variables

ii) Isolation Forest model to detect the outliers; (To isolate the data points by

tracing random lines over one of the (sampled) variables' axes and, after several iterations, measure how "hard" was to isolate each observation).

Step 3: Create the Topics

i) "transactions," where the producer will send new transaction records. **ii)** "anomalies," the module that detects anomalies will send the data, and the last consumer will read it to send a slack notification:

Step 4: Transaction Producer

i) To generate the first producer that will send new data to the Kafka topic "transactions"; use the confluent-Kafka package; in the file streaming/producer.py.

ii) Producer will send data to a Kafka topic, with a probability of OUTLIERS_GENERATION_PROBABILITY; the data will come from an "outlier generator," will send an auto-incremental id, the data needed for the machine learning model and the current time in UTC.

Step 5: Outlier Detector Consumer

i) To make the predictions, and filter the outliers. Done in the streaming/anomalies_detector.py file

ii) The consumer read messages from the "transactions" topic and a consumer sent outliers to the "anomalies".

Step 6: Slack notification

i) To take some actions with these detected outliers; in a real-life scenario, it could block a transaction, scale a server, generate a recommendation, send an alert to an administrative user.

Program:

Step1: pip install -r requirements.txt.

Step2: Train and build the model

```
from joblib import dump
import numpy as np
from sklearn.ensemble import IsolationForest

rng = np.random.RandomState(42)
# Generate train data
X = 0.3 * rng.randn(500, 2)
X_train = np.r_[X + 2, X - 2]
X_train = np.round(X_train, 3)

# fit the model
clf = IsolationForest(n_estimators=50, max_samples=500, random_state=rng, contamination=0.01)
clf.fit(X_train)

dump(clf, './isolation_forest.joblib')
```

Step 3: Create the topics

kafka-topics.sh --zookeeper localhost:2181 --topic transactions --create --partitions 3 --replication-factor 1 kafka-topics.sh --zookeeper localhost:2181 --topic anomalies --create --partitions 3 --replication-factor 1

Step 4: Transaction producer

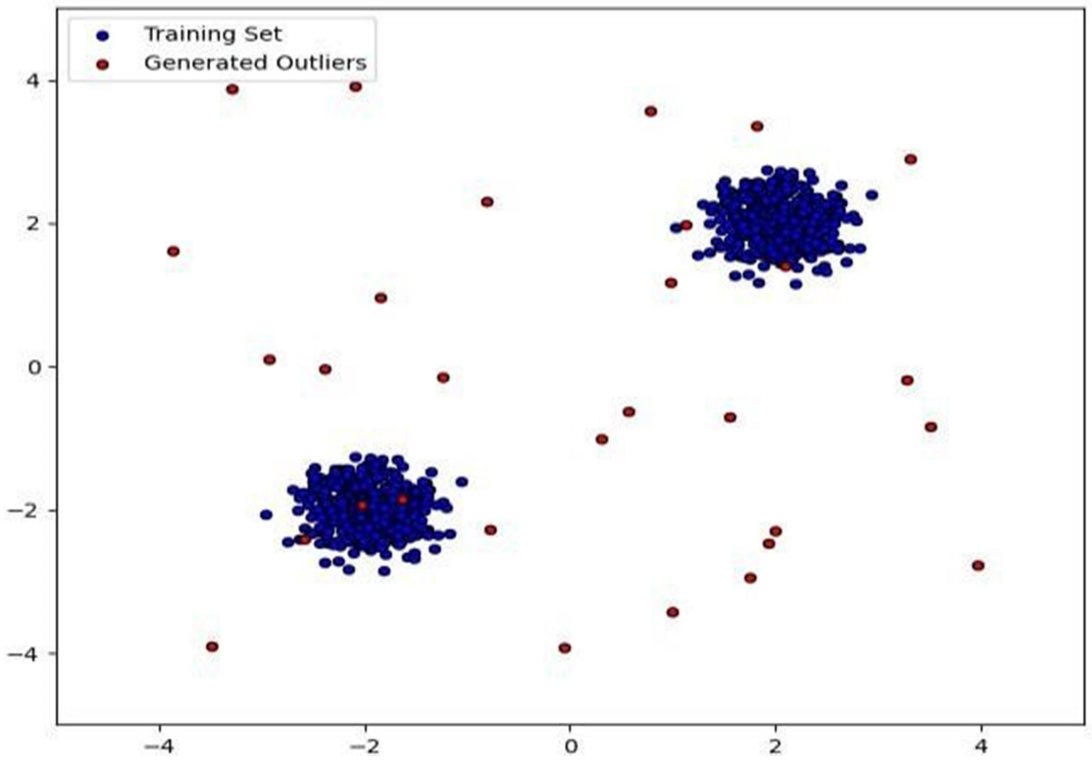
```
import json
import random
import time
from datetime import datetime
import numpy as np
from settings import TRANSACTIONS_TOPIC, DELAY, OUTLIERS_GENERATION_PROBABILITY
from streaming.utils import create_producer
```

```
_id = 0
producer = create_producer()

if producer is not None:
    while True:
        # Generate some abnormal observations
        if random.random() <= OUTLIERS_GENERATION_PROBABILITY:
            X_test = np.random.uniform(low=-4, high=4, size=(1, 2)).tolist()
        else:
            X = 0.3 * np.random.randn(1, 2)
            X_test = (X + np.random.choice(a=[2, -2], size=1, p=[0.5, 0.5]))
            X_test = np.round(X_test, 3).tolist()

        current_time = datetime.utcnow().isoformat()

        record = {"id": _id, "data": X_test, "current_time": current_time}
        record = json.dumps(record).encode("utf-8")

        producer.produce(topic=TRANSACTIONS_TOPIC,
                         value=record,
                         key=str(_id))
        producer.flush()
        _id += 1
        time.sleep(DELAY)
```

kafka-console-consumer.sh --bootstrap-server 127.0.0.1:9092 --topic transactions

Output

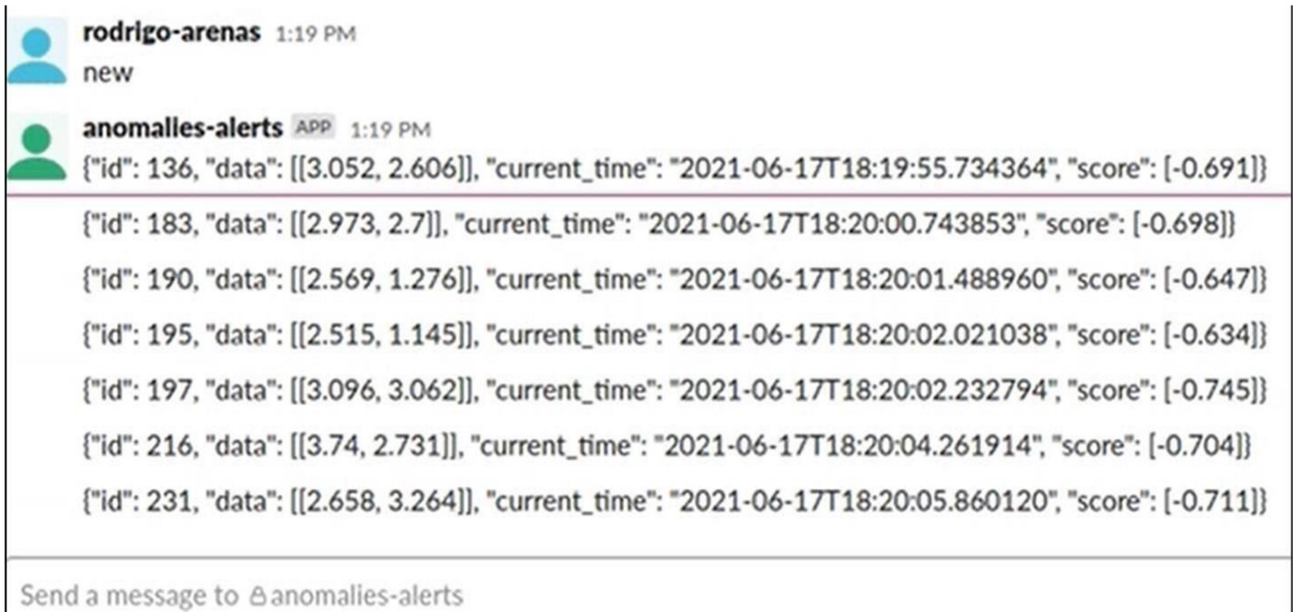
Anamoly detection

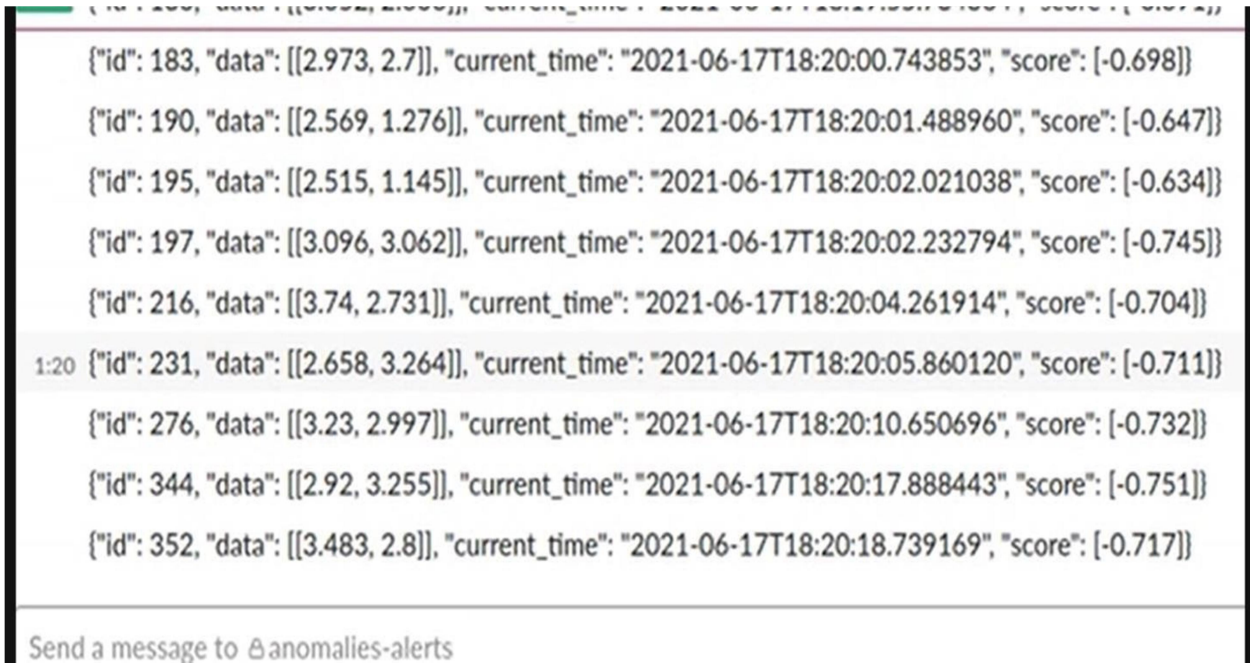


Step 5: Outlier Detector Consumer.

```
import json
import os
from joblib import load
import logging
from multiprocessing import Process
import numpy as np
from streaming.utils import create_producer, create_consumer
from settings import (TRANSACTIONS_TOPIC, TRANSACTIONS_CONSUMER_GROUP,
                      ANOMALIES_TOPIC, NUM_PARTITIONS)

model_path = os.path.abspath('../model/isolation_forest.joblib')
clf = load(model_path)


def detect():
    consumer = create_consumer(topic=TRANSACTIONS_TOPIC,
                               group_id=TRANSACTIONS_CONSUMER_GROUP)

    producer = create_producer()

    while True:
        message = consumer.poll(timeout=50)
        if message is None:
            continue
        if message.error():
            logging.error("Consumer error: {}".format(message.error()))
            continue

            # Message that came from producer
            record = json.loads(message.value().decode('utf-8'))
            data = record["data"]

            prediction = clf.predict(data)

            # If an anomaly comes in, send it to anomalies topic
            if prediction[0] == -1:
                score = clf.score_samples(data)
                record["score"] = np.round(score, 3).tolist()

                _id = str(record["id"])
                record = json.dumps(record).encode("utf-8")

                producer.produce(topic=ANOMALIES_TOPIC,
                                 value=record,
                                 key=_id)
                producer.flush()

        consumer.close()

# One consumer per partition
for _ in range(NUM_PARTITIONS):
    p = Process(target=detect)
    p.start()
```

CHATBOT ALERT NOTIFICATION





Step6: Slack notification

```
import logging
from slack import WebClient
from slack.errors import SlackApiError

from settings import (SLACK_API_TOKEN, SLACK_CHANNEL, ANOMALIES_TOPIC,
                      ANOMALIES_CONSUMER_GROUP)
from streaming.utils import create_consumer

client = WebClient(token=SLACK_API_TOKEN)
consumer = create_consumer(topic=ANOMALIES_TOPIC,
                           group_id=ANOMALIES_CONSUMER_GROUP)

while True:
    message = consumer.poll()
    if message is None:
        continue
    if message.error():
        logging.error("Consumer error: {}".format(message.error()))
        continue

    # Message that came from producer
    record = message.value().decode('utf-8')

    try:
        # Send message to slack channel
        response = client.chat_postMessage(channel=SLACK_CHANNEL,
                                           text=record)
    except SlackApiError as e:

        print(e.response["error"])

    consumer.commit()
```

Result:

Thus, the prediction of Real-time anomaly detection with Apache Kafka and Python has executes the streaming/Bot alerts notification.

EX NO: 04 (A)	QUERY FOR STAR SCHEMA USING SQL SERVER MANAGEMENT STUDIO
DATE:	

Aim:

To execute and verify query for star schema using SQL Server Management Studio.

Procedure:

Step 1: Install SQLEXPR and SQLManagementStudio

Step 2: Launch SQL Server Management Studio

Step 3: Create new database and write query for creating Star schema table

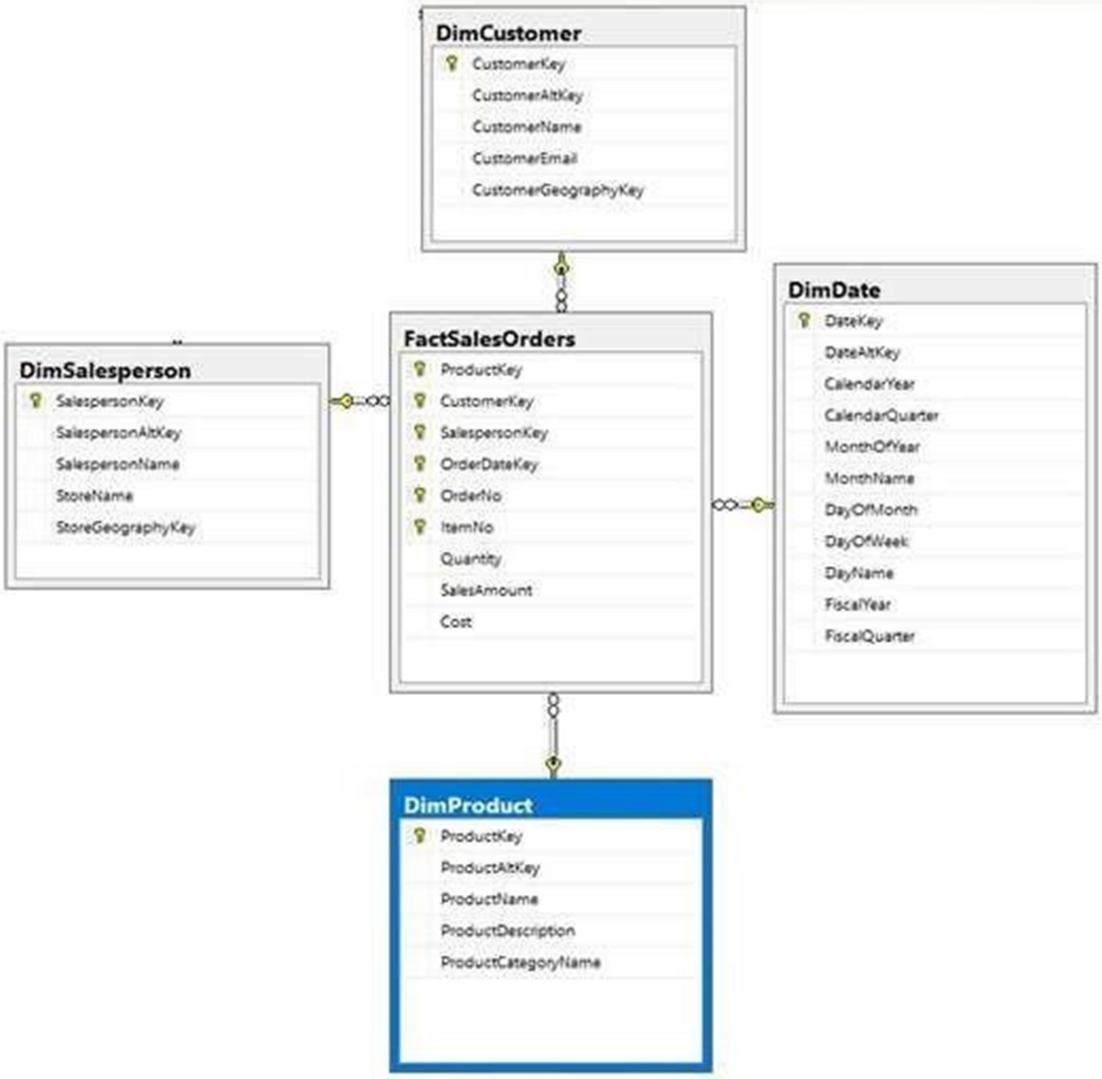
Step 4: Execute the query for schema

Step 5: Explore the database diagram for Star schema

Query for Star Schema

```
USE DemoGO

CREATE TABLE DimProduct

(ProductKey int identity NOT NULL PRIMARY KEY NONCLUSTERED,

ProductAltKey nvarchar(10) NOT NULL,

ProductName nvarchar(50) NULL, ProductDescription nvarchar(100) NULL, ProductCategoryName nvarchar(50))

GO

CREATE TABLE DimCustomer
(CustomerKey int identity NOT NULL PRIMARY KEY NONCLUSTERED,

CustomerAltKey nvarchar(10) NOT NULL, CustomerName nvarchar(50) NULL, CustomerEmail nvarchar(100) NULL,

CustomerGeographyKey int NULL)

GO
```

Output


DimCustomer
CustomerKey
CustomerAltKey
CustomerName
CustomerEmail
CustomerGeographyKey
DimDate
DateKey
DateAltKey
CalendarYear
CalendarQuarter
MonthOfYear
MonthName
DayOfMonth
DayOfWeek
DayName
FiscalYear
FiscalQuarter
DimSalesperson
SalespersonKey
SalespersonAltKey
SalespersonName
StoreName
StoreGeographyKey
FactSalesOrders
ProductKey
CustomerKey
SalespersonKey
OrderDateKey
OrderNo
ItemNo
Quantity
SalesAmount
Cost
DimProduct
ProductKey
ProductAltKey
ProductName
ProductDescription
ProductCategoryName

CREATE TABLE DimSalesperson
(SalespersonKey int identity NOT NULL PRIMARY KEY NONCLUSTERED,

SalespersonAltKey nvarchar(10) NOT NULL,

SalespersonName nvarchar(50) NULL, StoreName nvarchar(50) NULL,

SalespersonGeographyKey int NULL) GO

CREATE TABLE DimDate

(DateKey int NOT NULL PRIMARY KEY NONCLUSTERED, DateAltKey datetime NOT

NULL, CalendarYear int NOT NULL, CalendarQuarter int NOT NULL, MonthOfYear int NOT NULL, [MonthName]nvarchar(15) NOT NULL, [DayOfMonth]int NOT NULL,

[DayOfWeek]int NOT NULL, [DayName]nvarchar(15) NOT NULL, FiscalYear int NOT NULL, FiscalQuarter int NOT NULL)

GO

CREATE TABLE FactSalesOrders
(ProductKey int NOT NULL REFERENCES DimProduct(ProductKey), CustomerKey int NOT NULL REFERENCES DimCustomer(CustomerKey), SalespersonKey int NOT NULL REFERENCES DimSalesperson(SalespersonKey), OrderDateKey int NOT NULL

REFERENCES DimDate(DateKey),

OrderNo int NOT NULL, ItemNo int NOT NULL, Quantity int NOT NULL,

SalesAmount money NOT NULL,

Cost money NOT NULL

Result:

Thus, the Query for Star Schema was created and executed successfully.

EX NO: 04 (B)	QUERY FOR SNOWFLAKE SCHEMA USING SQL SERVER MANAGEMENT STUDIO
DATE:	

Aim:

To execute and verify query for Snowflake schema using SQL Server Management Studio.

Procedure:

Step 1: Install SQLEXPR and SQLManagementStudio

Step 2: LaunchSQL Server Management Studio

Step 3: Create new database and write query for creating Star schema table

Step 4: Execute the query

Step 5: Explore the database diagram for SnowFlake schema

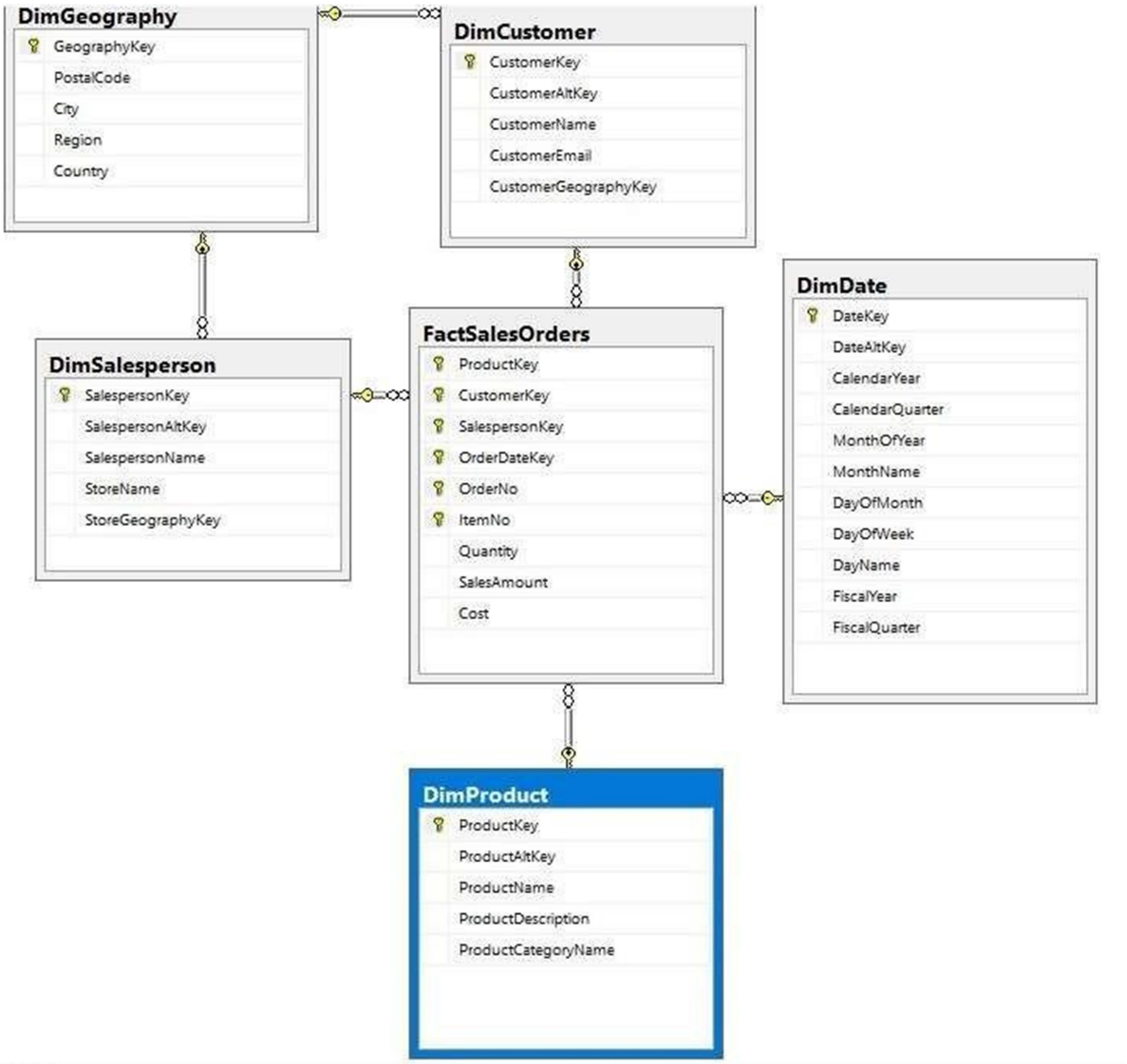
Step 6: Connect the Geography table with Salesperson & Product Geography key

Query for SnowFlake Schema

```
USE Demo
GO
CREATE TABLE DimProduct
(ProductKey int identity NOT NULL PRIMARY KEY NONCLUSTERED,
ProductAltKey nvarchar(10) NOT NULL,
ProductName nvarchar(50) NULL, ProductDescription nvarchar(100) NULL, ProductCategoryName nvarchar(50))
GO
CREATE TABLE DimCustomer
(CustomerKey int identity NOT NULL PRIMARY KEY NONCLUSTERED,
CustomerAltKey nvarchar(10) NOT NULL,
```

CustomerName nvarchar(50) NULL, CustomerEmail nvarchar(100) NULL, CustomerGeographyKey int NULL) GO

Output



CREATE TABLE DimSalesperson

(SalespersonKey int identity NOT NULL PRIMARY KEY NONCLUSTERED,

SalespersonAltKey nvarchar(10) NOT NULL,

SalespersonName nvarchar(50) NULL, StoreName nvarchar(50) NULL,

SalespersonGeographyKey int NULL) GO

CREATE TABLE DimGeography
(GeographyKey int identity NOT NULL PRIMARY KEY NONCLUSTERED, PostalCode nvarchar(15) NULL,

City nvarchar(50) NULL, Region nvarchar(50) NULL, Country nvarchar(50) NULL) GO

CREATE TABLE FactSalesOrders
(ProductKey int NOT NULL REFERENCES DimProduct(ProductKey), CustomerKey int NOT NULL REFERENCES DimCustomer(CustomerKey), SalespersonKey int NOT NULL

REFERENCES DimSalesperson(SalespersonKey), OrderNo int NOT NULL,

ItemNo int NOT NULL, Quantity int NOT NULL, SalesAmount money NOT NULL, Cost money NOT NULL

CONSTRAINT[PK_FactSalesOrders] PRIMARY KEY NONCLUSTERED ([ProductKey],[CustomerKey],[SalesPersonKey],[OrderNo], [ItemNo]

))

Result:

Thus, the Query for Snowflake Schema was created and executed successfully.

EX NO: 05	DESIGN DATA WAREHOUSE FOR REAL TIME APPLICATIONS
DATE:	

Aim:

To design and execute data warehouse for real time application using SQL Server Management Studio.

Procedure:

Step 1: Launch SQL Server Management Studio

Step 2: Explore the created database

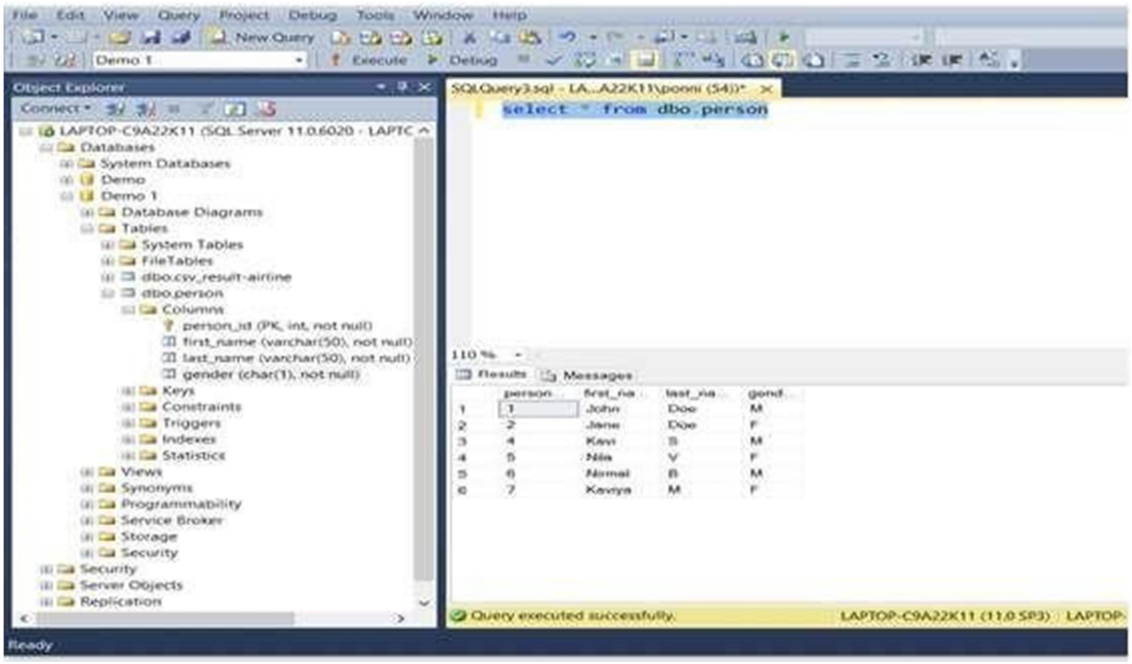
Step 3: 3.1 Right-click on the table name and click on the Edit top 200 rows option.

3.2. Enter the data inside the table or use the top 1000 rows option and enter the query.

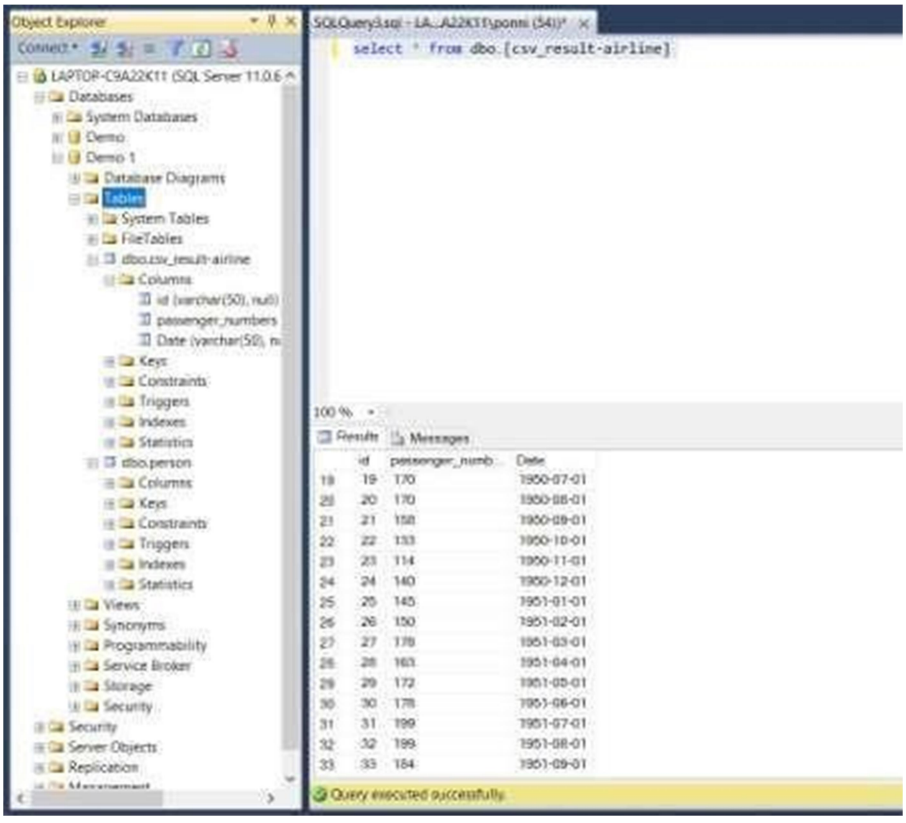
Step 4: Execute the query, and the data will be updated in the table.

Step 5: Right-click on the database and click on the tasks option. Use the import data option to import files to the database.

Output



Import CSV file



Sample Query

```
INSERT INTO dbo.person(first_name,last_name,gender) VALUES
('Kavi','S','M'), ('Nila','V','A'), ('Nirmal','B','M'), ('Kaviya','M','F');

SELECT * FROM dbo.person
```

Result:

Thus, the data warehouse for real-time applications was designed successfully.

EX NO: 06	ANALYSE THE DIMENSIONAL MODELING
DATE:	

Aim:

To implement the creation of table dimensions and analysis of data model.

Procedure:

Step1: Identify the Business Process
Step2: Identify the Grain
Step3: Identify the Dimensions
Step4: Identify the Facts
Step5: Build the Schema

Implementation:

Create the data
warehouse
create database
TopHireDW go
use TopHireDW
go
-- Create Date Dimension
if exists (select * from sys.tables
where name = 'DimDate') drop table
DimDate go
create table DimDate
(DateKey int not null primary key,
[Year] varchar(7), [Month] varchar(7), [Date] date,
DateString varchar(10)) go
-- Populate
Date
Dimension
truncate
table
DimDate go
declare @i int, @Date date, @StartDate date, @EndDate date, @DateKey int,
@DateString varchar(10), @Year
varchar(4), @Month varchar(7),
@Date1 varchar(20)
set @StartDate
= '2006-01-01'
set @EndDate =

'2016-12-31' set
@Date =

insert into DimDate (DateKey, [Year], [Month], [Date], DateString) values (0, 'Unknown', 'Unknown', '0001-01-01', 'Unknown') --The unknown row while @Date <= @EndDate

begin set @DateString = convert(varchar(10), @Date, 20) set @DateKey = convert(int, replace(@DateString,'-','')) set @Year = left(@DateString,4) set @Month = left(@DateString, 7)

insert into DimDate (DateKey, [Year], [Month], [Date], DateString) values (@DateKey, @Year, @Month, @Date, @DateString) set @Date = dateadd(d, 1, @Date) end go

select * from DimDate

-- Create Customer dimension

if exists (select * from sys.tables where name = 'DimCustomer') drop table DimCustomer go

create table DimCustomer

(CustomerKey int not null identity(1,1) primary key,

CustomerId varchar(20) not null,

CustomerName varchar(30), DateOfBirth date, Town varchar(50),

TelephoneNo varchar(30), DrivingLicenceNo varchar(30), Occupation varchar(30)

)

go

insert into DimCustomer (CustomerId, CustomerName, DateOfBirth, Town, TelephoneNo, DrivingLicenceNo, Occupation) select * from HireBase.dbo.Customer select * from DimCustomer -- Create Van dimension

if exists (select * from sys.tables where name = 'DimVan') drop table DimVan go

create table DimVan

(VanKey int not null identity(1,1) primary key,

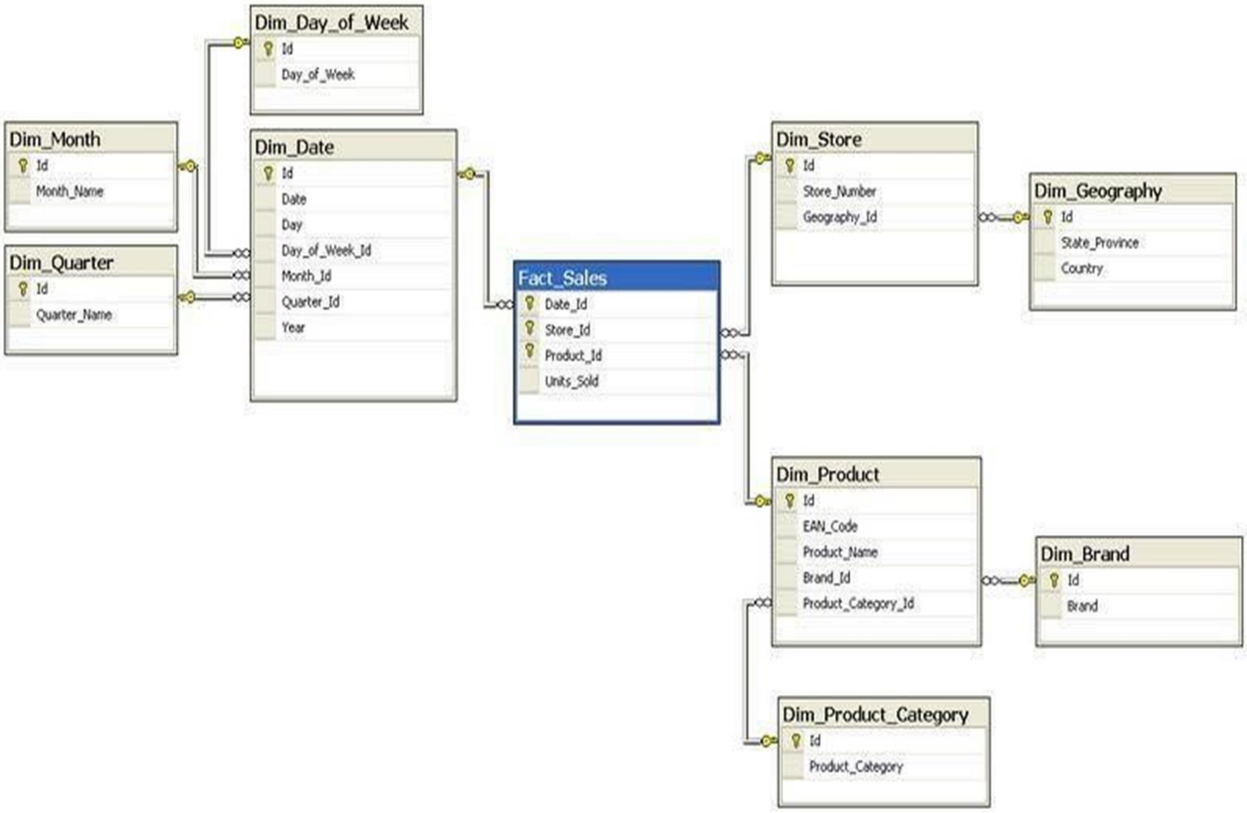
RegNo varchar(10) not null,

Make varchar(30), Model varchar(30), [Year] varchar(4),

Colour varchar(20), CC int, Class varchar(10)

Output

Snowflake Schema image source of dimensional data model



```
)
go
insert into DimVan (RegNo, Make, Model, [Year], Colour,
CC, Class) select * from HireBase.dbo.Van go select *
from DimVan -- Create Hire fact table
if exists (select * from sys.tables where name =
'FactHire') drop table FactHire go
create table FactHire
( SnapshotDateKey int not null, --Daily periodic snapshot fact table
HireDateKey int not null, CustomerKey int not null, VanKey int not null, --
Dimension Keys
HireId varchar(10) not null, --Degenerate Dimension
NoOfDays int, VanHire money, SatNavHire money,
Insurance money, DamageWaiver money, TotalBill money
)
go
select * from FactHire
```

tool with MySQL query

Result:

Thus, the implementation and analysed of data dimension model using weka implemented successfully.

EX NO: 07	CASE STUDY USING OLAP
DATE:	

Aim:

To evaluate the implementation and impact of OLAP technology in a real-world business context, analysing its effectiveness in enhancing data analysis, decision-making, and overall operational efficiency**.**

Introduction:

OLAP stands for **On-Line Analytical Processing**. OLAP is a classification of software technology which authorizes analysts, managers, and executives to gain insight into information through fast, consistent, interactive access in a wide variety of possible views of data that has been transformed from raw information to reflect the real dimensionality of the enterprise as understood by the clients .It is used to analyse business data from different points of view. Organizations collect and store data from multiple data sources, such as websites, applications, smart meters, and internal systems.

Methodology

OLAP (Online Analytical Processing) methodology refers to the approach and techniques used to design, create, and use OLAP systems for efficient multidimensional data analysis. Here are the key components and steps involved in the OLAP methodology:

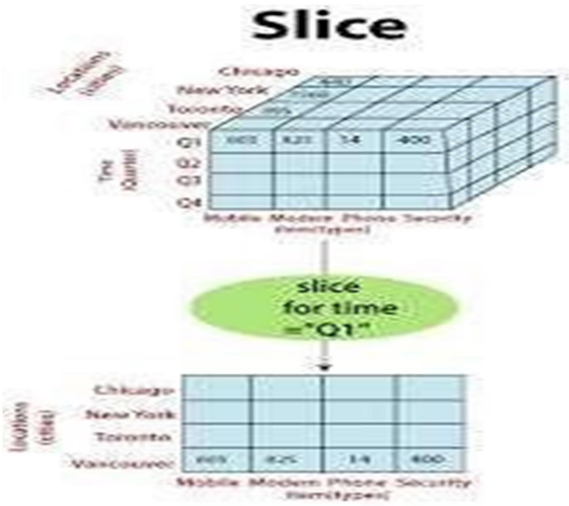
1. **Requirement Analysis:**

The process begins with understanding the specific analytical requirements of the users. Analysts and stakeholders define the dimensions, measures, hierarchies, and data sources that will be part of the OLAP system. This step is crucial to ensure that the OLAP system meets the business needs.

2. **Dimensional Modeling:**

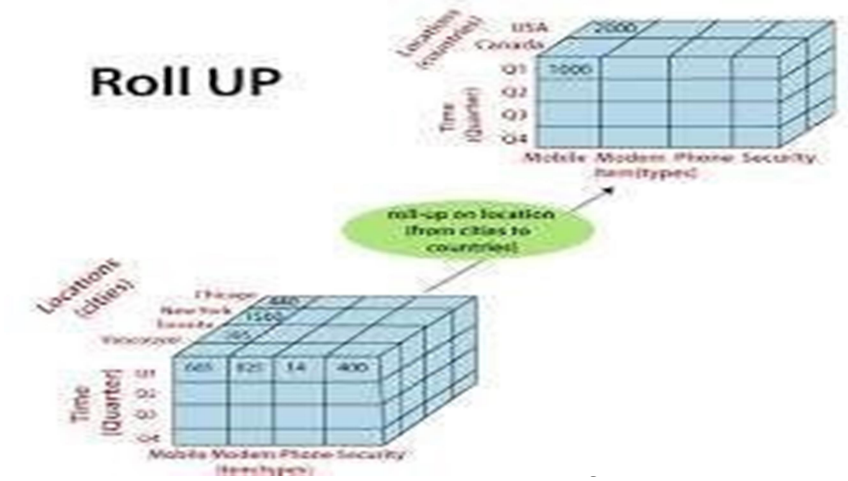
Dimension tables are designed to represent attributes like time, geography, and product categories. Fact tables contain the numerical data (measures) and the keys to dimension tables.

Slice



Dice

Roll Up



3. **Star Schema:**

This is a common design in OLAP systems where the fact table is at the center, connected to dimension tables.

4. **Data Extraction and Transformation:**

Data is extracted from various sources, cleaned, and transformed into a format suitable for OLAP analysis. This may involve data aggregation, cleansing, and integration.

5. **Data Loading:**

The prepared data is loaded into the OLAP database or cube. This step includes populating the dimension and fact tables and creating the data cube structure.

Operations in OLAP

In OLAP (Online Analytical Processing), operations are the fundamental actions performed on multidimensional data cubes to retrieve, analyze, and present data in a way that facilitates decision-making and data exploration. The main operations in OLAP are:

1. **Slice:** Slicing involves selecting a single dimension from a multidimensional cube to view a specific "slice" of the data. For example, you can slice the cube to view sales data for a particular month, product category, or region.

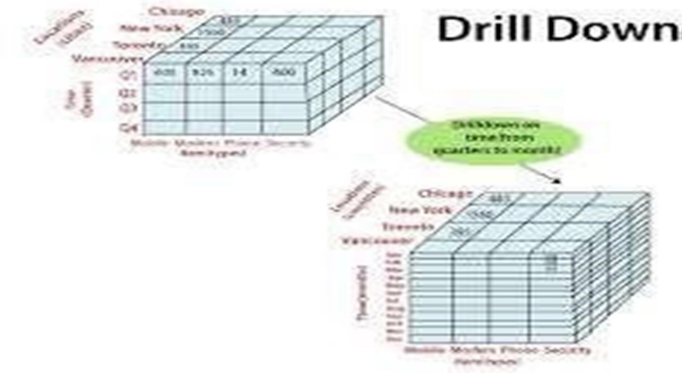
2. **Dice:** Dicing is the process of selecting specific values from two or more dimensions to create a subcube. It allows you to focus on a particular combination of attributes. For example, you can dice the cube to view sales data for a specific product category and region within a certain time frame.

3. **Roll-up (Drill-up):** Roll-up allows you to move from a more detailed level of data to a higher-level summary. For instance, you can roll up from daily sales data to monthly or yearly sales data, aggregating the information.

4. **Drill-down (Drill-through):** Drill-down is the opposite of roll-up, where you move from a higher-level summary to a more detailed view of the data. For example, you can drill down from yearly sales data to quarterly, monthly, and daily data, getting more granularity.

Pivot

Drill Down



5. **Pivot (Rotate):** Pivoting involves changing the orientation of the cube, which means swapping dimensions to view the data from a different perspective. This operation is useful for exploring data in various ways.

6. **Slice and Dice:** Combining slicing and dicing allows you to select specific values from different dimensions to create sub cubes. This operation helps you focus on a highly specific subset of the data.

7. **Drill-across**: Drill-across involves navigating between cubes that are related but have different dimensions or hierarchies. It allows users to explore data across different OLAP cubes.

8. **Data Filtering:** In OLAP, you can filter data to view only specific data points or subsets that meet certain criteria. This operation is useful for narrowing down data to what is most relevant for analysis.

Real time example

One of the real time examples of olap is Market Basket Analysis. Let us discuss in detail about the example

Market Basket Analysis:

- A data mining technique, is typically performed using algorithms like Apriori, FP- growth, or Eclat. These algorithms are designed to discover associations or patterns in transaction data, such as retail sales.
- While traditional OLAP (Online Analytical Processing) is not the primary tool for market basket analysis, it can play a supporting role. Here's how OLAP can complement market basket analysis in more detail:

1. Data Integration:

Gather and integrate transaction data from various sources, such as point-of-sale systems, e-commerce platforms, or other transactional databases. Clean and pre-process the data, ensuring that it is in a format suitable for analysis.

2. Data Modeling:

Design a data model that will be used in the OLAP cube. In the context of market basket analysis, consider the following dimensions and measures:

Dimensions:

Time (e.g., day, week, month)

Products (individual items or product categories) Customers (if you want to analyse customer behaviour).

Measures:

- The count of transactions containing specific items or itemsets.
  The count of products in each transaction.
- Any other relevant metrics, such as revenue, quantity, or profit.

3. Data Loading:

Load the integrated and preprocessed transaction data into the OLAP cube. Ensure that the cube is regularly updated to reflect the most recent data.

4. OLAP Cube Design:

Define hierarchies and relationships within the cube to enable effective analysis. For instance, you might have hierarchies that allow drilling down from product categories to individual products.

5. Market Basket Analysis:

Although OLAP cubes are not designed for direct market basket analysis, they can facilitate it in several ways.

Conclusion

OLAP is a powerful technology for businesses and organizations seeking data insights, informed decisions, and performance improvement. It enables multidimensional data analysis, especially in complex, data-intensive environments. It is a crucial technology for organizations seeking to gain insights from their data and make informed decisions. It empowers businesses to analyse data efficiently and effectively, offering a competitive advantage in today's data-driven world.

EX.NO: 08	CASE STUDY USING OLTP
DATE:	

Aim:

Develop an OLTP system that enables the e-commerce company to process a high volume of online orders, track inventory, manage customer information, and handle financial transactions in real-time, ensuring data integrity and providing a seamless shopping experience for customers.

Introduction:

In today's digital age, businesses across various industries are relying heavily on technology to streamline their operations and provide seamless services to their customers. One crucial aspect of this technological transformation is the development and implementation of efficient Online Transaction Processing (OLTP) systems. This case study delves into the design and implementation of an OLTP system for a fictional e-commerce company, "TechTrend Electronics," and examines the key considerations, challenges, and aims associated with such a project. This case study aims to showcase the process of developing an OLTP system tailored to TechTrend Electronics' unique requirements. The objective is to ensure that the company can efficiently handle a multitude of real-time transactions while maintaining data accuracy and providing a seamless shopping experience for its customers.

Methodology:

The methodology for developing an OLTP (Online Transaction Processing) system for a case study involves a systematic approach to designing, implementing, and testing the system. Below is a step-by-step methodology for creating an OLTP system for a case study, using the fictional e-commerce company "Tech Trend Electronics" as an example:

1. Database Design:

Develop a well-structured relational database schema that aligns with the business requirements. Normalize the data to eliminate redundancy and ensure data consistency. Create entity-relationship diagrams and define data models for key entities like customers, products, orders, payments, and inventory.

2. Technology Selection:

Choose appropriate technologies for the database management system (e.g., MySQL, PostgreSQL, Oracle) and programming languages (e.g., Java, Python, C#) for the OLTP system. Evaluate and select suitable frameworks, libraries, and tools that align with the chosen technologies.

3. System Architecture:

Design the system's architecture, which may include multiple application layers, a web interface, and a database layer. Implement a layered architecture, separating concerns for scalability, maintainability, and security.

4. User Authentication and Authorization:

Implement user authentication mechanisms to secure access to the system for both customers and staff. Define access control policies and user roles (e.g., customers, administrators, and employees) based on the principle of least privilege.

5. Transaction Processing Logic:

Develop the transaction processing logic, including handling order placement, inventory management, and payment processing in real-time. Ensure that transactions adhere to the ACID properties for data integrity.

6. Security Measures:

Implement security measures to protect customer data, financial information, and the system itself. Use encryption for sensitive data and ensure that the system is protected against common security threats (e.g., SQL injection, cross-site scripting).

7. Deployment and Monitoring:

Deploy the OLTP system in a production environment.

Implement monitoring tools to track system performance, identify bottlenecks, and generate reports for system administrators.

8. Maintenance and Updates:

Establish a plan for system maintenance and regular updates to address issues, enhance functionality, and adapt to changing business needs.

Real World Example

In a real-world scenario, let's consider an e-commerce platform as an example of an OLTP system. The platform processes millions of transactions every day.

Here's a breakdown of how the system functions: Users can browse through the website, add products to their carts, and complete the checkout process. As a user completes the checkout process, a new transaction is created. This transaction contains information about the products purchased, the buyer's details, the shipping address, and other relevant data. The system generates an invoice for the buyer and sends it via email. The system generates transaction reports, such as daily sales summaries or sales by product category, for internal use and management. In this scenario, the e-commerce platform acts as an OLTP system, with its transaction processing capabilities and the real-time updates to inventory and order details being key components. Here's an alternative approach using OLAP: Aggregate sales data across all time and geographical locations, making it available for reporting and analysis.

Allow business managers to run complex analytical queries on this data, such as calculating average sales by product category, comparing sales trends between different regions, or identifying top-performing sales channels. Use OLAP tools like data warehouses and data cubes to enable fast, real-time access to aggregated data and to simplify the process of running complex analytical queries.

By leveraging OLAP capabilities, businesses can gain insights into their sales performance, identify trends and patterns, and make data-driven decisions. This can ultimately lead to increased revenue, better customer service, and more efficient use of resources.

Conclusion:

In conclusion, OLTP systems play a pivotal role in modern business operations, facilitating real-time transaction processing, data integrity, and customer interactions. These systems are designed for high concurrency, low-latency, and consistent data access, making them essential for day-to-day operations in various industries, such as finance, e-commerce, healthcare, and more.

Overall, OLTP systems are the backbone of modern business operations, ensuring the seamless execution of day-to-day transactions and delivering a positive customer experience.

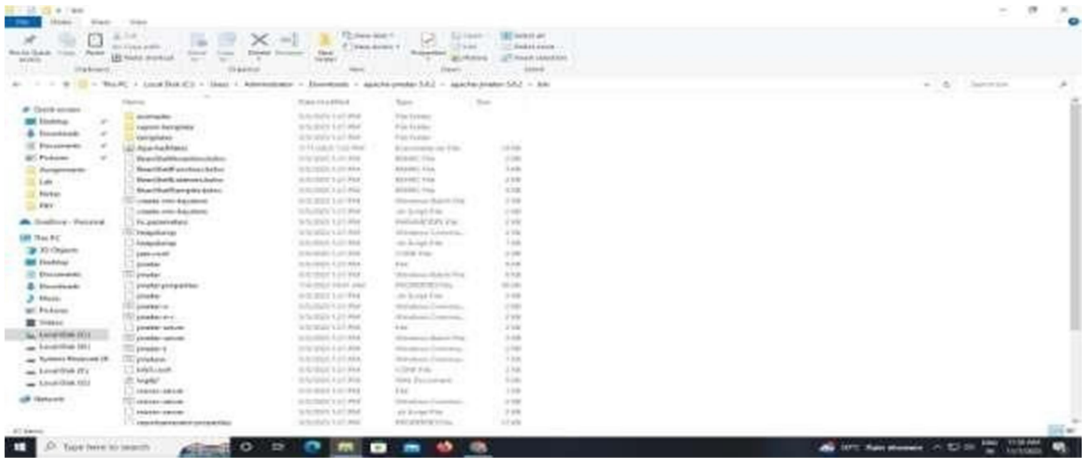
EX NO: 09	IMPLEMENTATION OF WAREHOUSE TESTING.
DATE:	

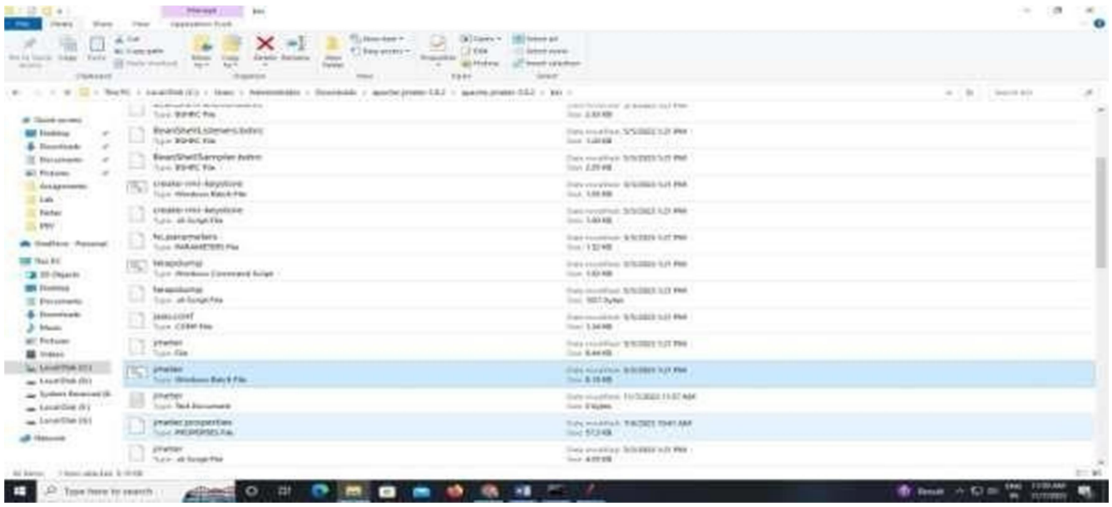
Aim:

To perform load testing using JMeter and interact with a SQL Server database using SQL Management Studio, you'll need to set up JMeter to send SQL queries to the database and collect the results for analysis.

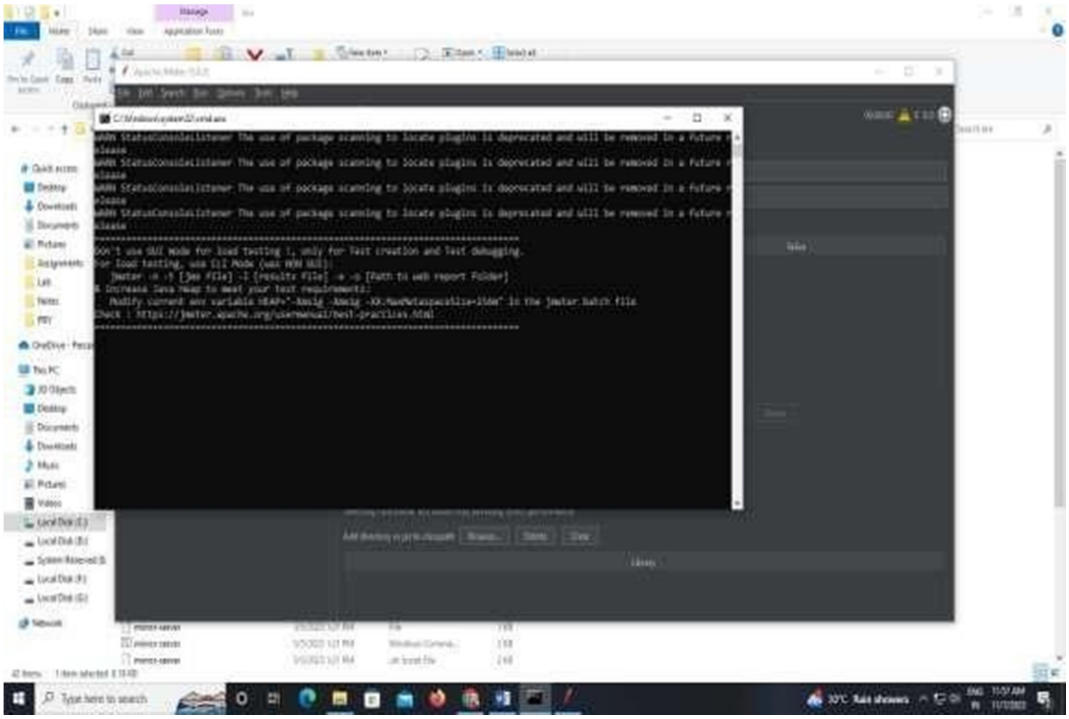
Procedure:

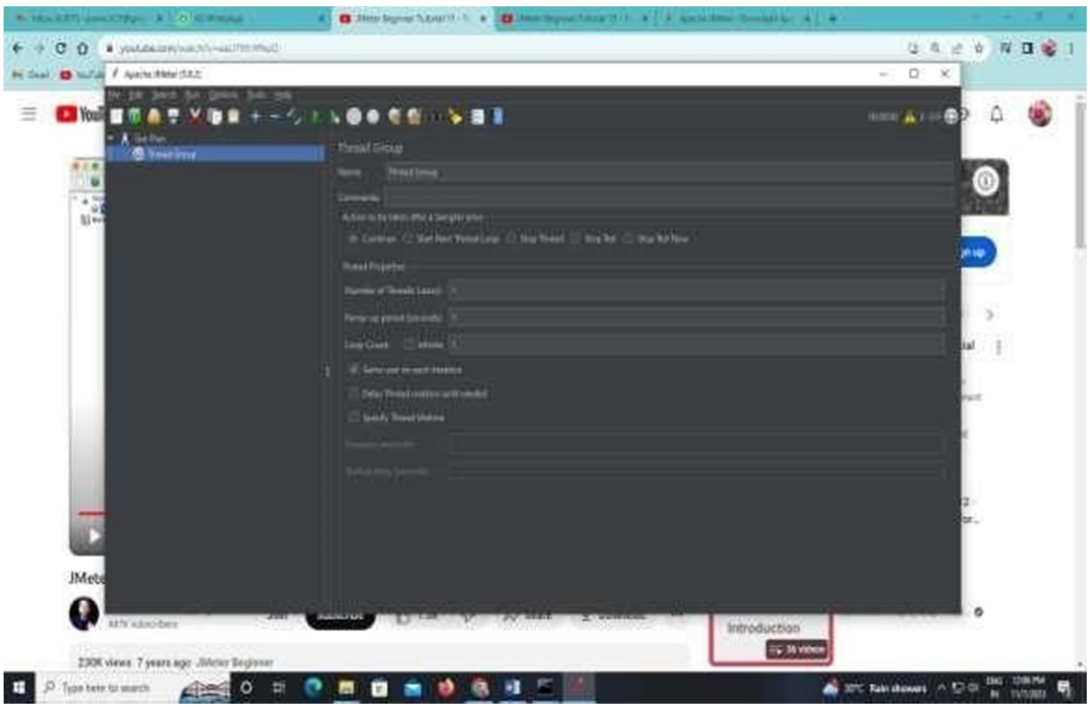
1. Install Required Software:
    - Install JMeter: Download and install JMeter from the official Apache JMeter website.
    - Install SQL Server and SQL Management Studio: If you haven't already, set up SQL Server and SQL Management Studio to manage your database.



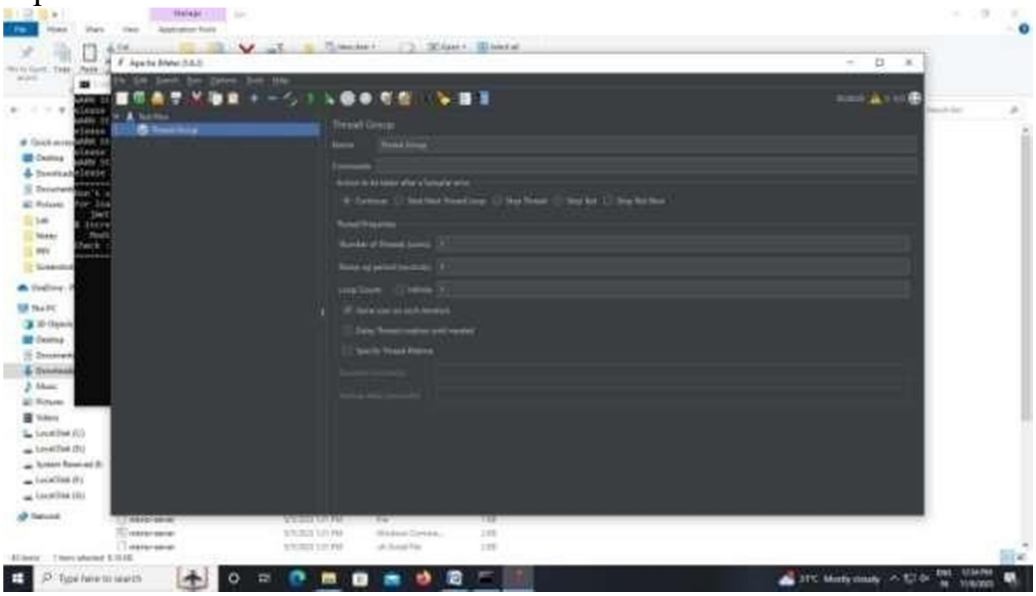


2. Create a Test Plan in JMeter:
    - Launch JMeter and create a new Test Plan.

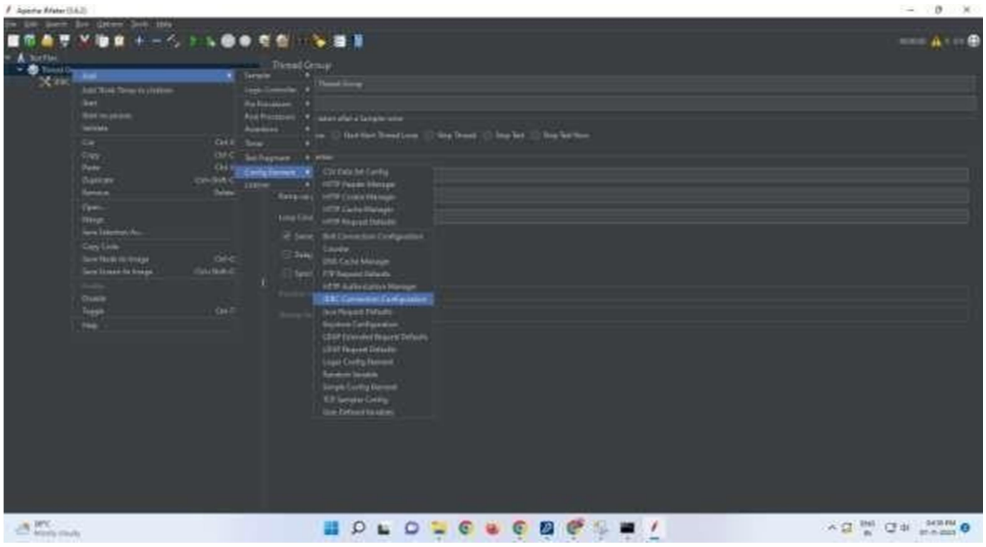




3. Add Thread Group:
    - Add a Thread Group to your Test Plan to simulate the number of users and requests.

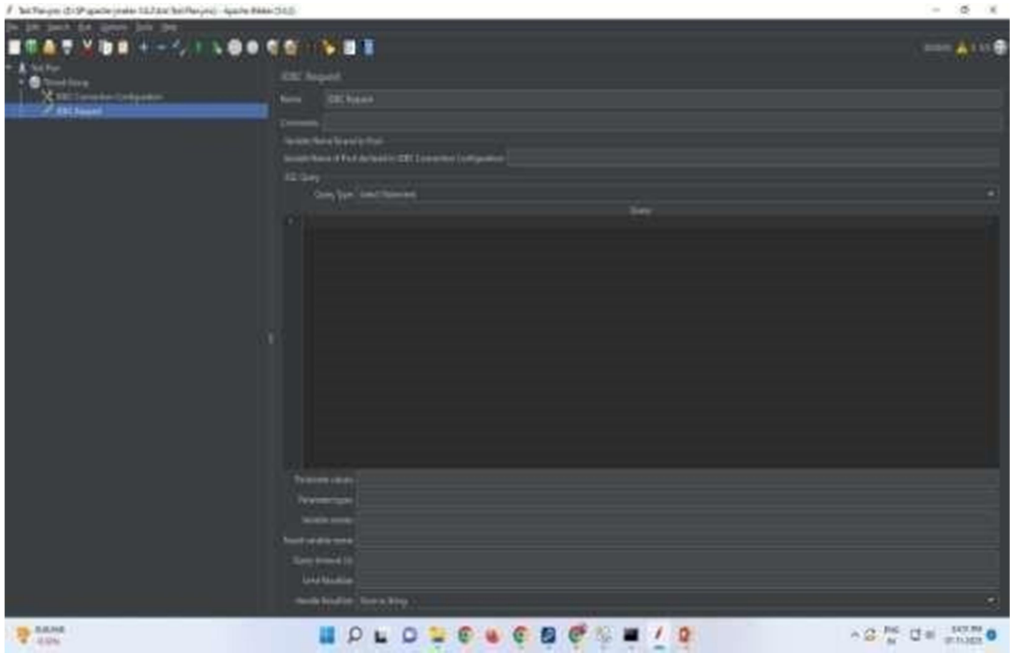


4. Add JDBC Connection Configuration:
    - Add a JDBC Connection Configuration element to your Thread Group. Configure it with the database connection details, such as the JDBC URL, username, and password. This element will allow JMeter to connect to your SQL Server database.



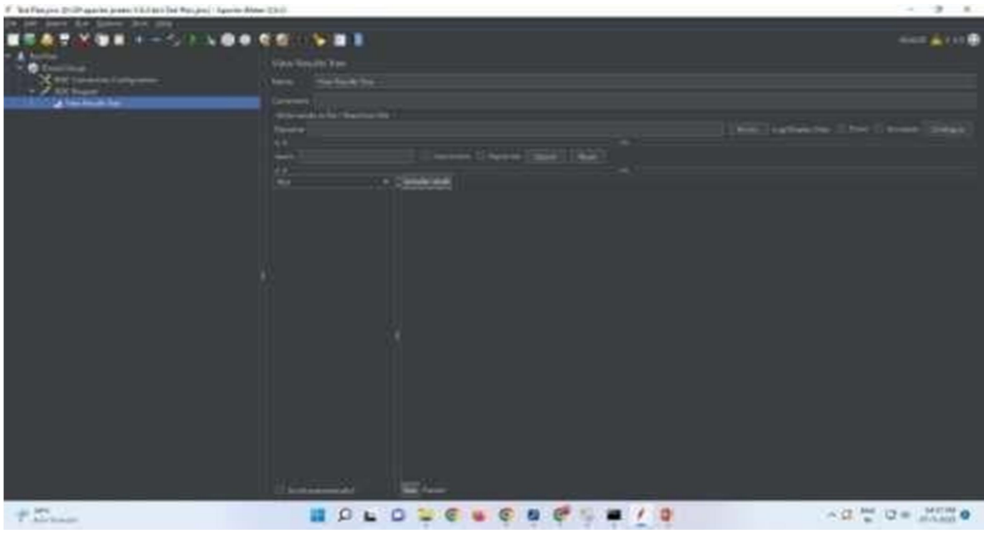
5. Add a JDBC Request Sampler:

- Add a JDBC Request sampler to your Thread Group. This sampler will contain your SQL query.
- Configure the JDBC Request sampler with the JDBC Connection Configuration created in the previous step.
- Enter your SQL query in the "Query" field of the JDBC Request sampler.

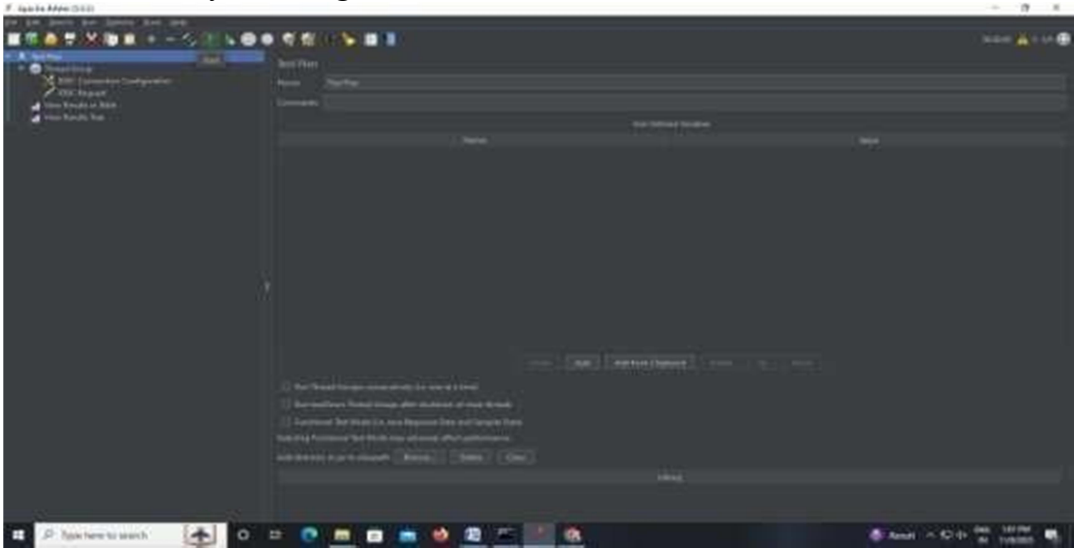


6.Add Listeners:

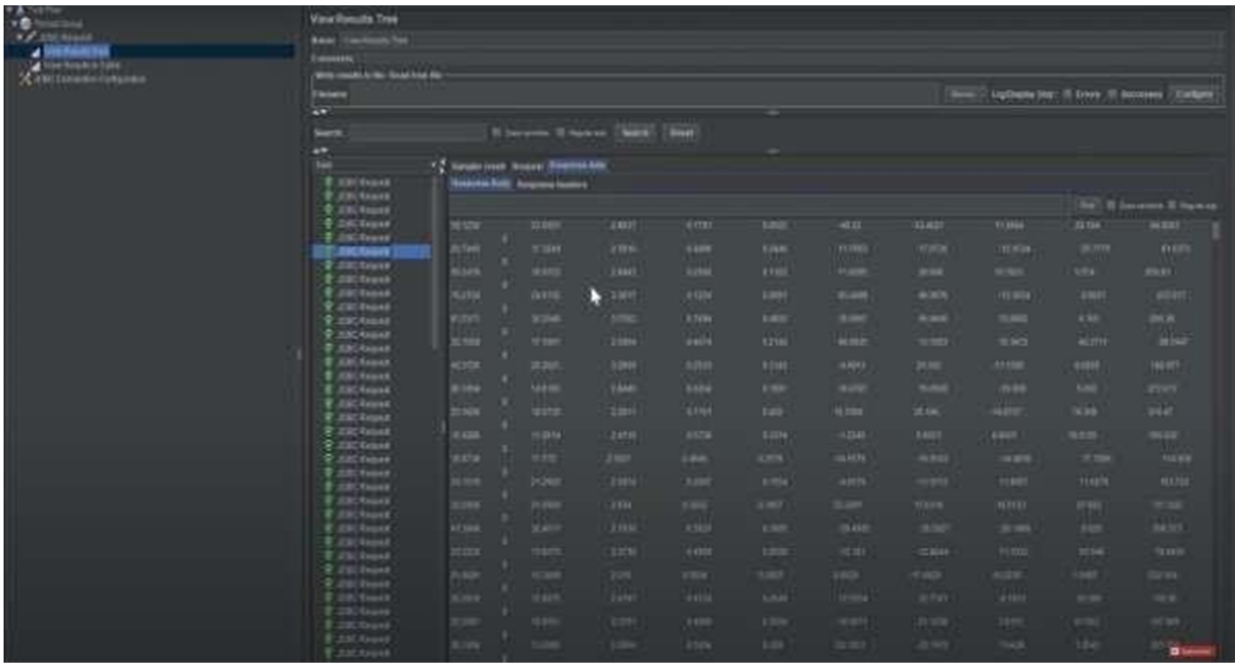
- Add listeners to your Test Plan to collect and view the test results. Common listeners include View Results Tree, Summary Report, and Response Times Over Time.



7. Configure Your Test Plan:
    - Configure the number of threads (virtual users), ramp-up time, and loop count in the Thread Group to simulate the desired load.
8. Run the Test:
    - Start the test by clicking the "Run" button in JMeter.



9. View and Analyse Results:
    - After the test has completed, you can view and analyse the results using the listeners you added. You can analyse response times, errors, and other performance metrics.



10. Optimize and Fine-Tune:

- Based on the results, you can optimize your SQL queries and JMeter test plan to finetune the performance of your database.

Conclusion

Using JMeter in conjunction with SQL Management Studio can be a powerful combination for load testing and performance analysis of applications that rely on SQL Server databases. This approach allows you to simulate a realistic user load, send SQL queries to the database, and evaluate the system's performance under various conditions.

JMeter in combination with SQL Management Studio provides a robust solution for assessing the performance of applications that rely on SQL Server databases. Through thorough testing, analysis, and optimization, you can ensure your application is capable of delivering a reliable and responsive experience to users even under heavy load conditions.