Prac 1

"Design a data warehouse for a retail store to analyze sales trends, customer purchases, and product performance. The warehouse will provide insights into sales by time period, customer demographics, and product categories."

- **Design the Star Schema:**
- In a **Star Schema**, the central **fact table** (e.g., `Sales`) is surrounded by **dimension tables** (e.g., `Customers`, `Products`, `Time`).

Example design:

- **Fact Table: `Sales`**
  - `sales_id` (Primary Key)
  - `product_id` (Foreign Key to Product dimension)
  - `customer_id` (Foreign Key to Customer dimension)
  - `time_id` (Foreign Key to Time dimension)
  - `quantity_sold`
  - `sales_amount`
- **Dimension Tables:**
  - **`Products`**: `product_id`, `product_name`, `category`, `price`
  - **`Customers`**: `customer_id`, `name`, `age`, `gender`, `location`
  - **`Time`**: `time_id`, `date`, `month`, `year`

Practical 2: Implementation of Dimension and Fact Tables

1. **Fact Table (`Sales`):**

   ```
   CREATE TABLE Sales (
       sales_id INT PRIMARY KEY,
       product_id INT,
       customer_id INT,
       time_id INT,
       quantity_sold INT,
       sales_amount DECIMAL(10, 2),
       FOREIGN KEY (product_id) REFERENCES Products(product_id),
       FOREIGN KEY (customer_id) REFERENCES Customers(customer_id),
       FOREIGN KEY (time_id) REFERENCES Time(time_id)
   );
   ```

2. **Dimension Tables:**

   - **`Products`** table:

     ```
     CREATE TABLE Products (
         product_id INT PRIMARY KEY,
         product_name VARCHAR(255),
         category VARCHAR(255),
         price DECIMAL(10, 2)
     );
     ```

   - **`Customers`** table:

```
CREATE TABLE Customers (
    customer_id INT PRIMARY KEY,
    name VARCHAR(255),
    age INT,
    gender VARCHAR(10),
    location VARCHAR(255)
);
```

- **`Time`** table:

```
CREATE TABLE Time (
    time_id INT PRIMARY KEY,
    date DATE,
    month VARCHAR(20),
    year INT
);
```

Practical 3:

1. Schema Design (Dimension and Fact Tables)

We will use the schema designed in Practical 1. Here's a reminder of the structure:

Fact Table (`Sales`):

```
sql
Copy code
CREATE TABLE Sales (
    sales_id INT PRIMARY KEY,
    product_id INT,
    customer_id INT,
    time_id INT,
    quantity_sold INT,
    sales_amount DECIMAL(10, 2),
    FOREIGN KEY (product_id) REFERENCES Products(product_id),
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id),
    FOREIGN KEY (time_id) REFERENCES Time(time_id)
);
```

Dimension Tables:

- **`Products` table:**

```
sql
Copy code
CREATE TABLE Products (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(255),
    category VARCHAR(255),
    price DECIMAL(10, 2)
);
```

- **`Customers` table:**

```
sql
Copy code
CREATE TABLE Customers (
    customer_id INT PRIMARY KEY,
    name VARCHAR(255),
    age INT,
    gender VARCHAR(10),
    location VARCHAR(255)
);
```

- **Time table:**

```
    sql
    Copy code
    CREATE TABLE Time (
        time_id INT PRIMARY KEY,
        date DATE,
        month VARCHAR(20),
        year INT
    );
```

Slice:

```
SELECT * FROM Sales

JOIN Time ON Sales.time_id = Time.time_id

WHERE year = 2023;
```

Dice:

```
SELECT * FROM Sales

JOIN Products ON Sales.product_id = Products.product_id

JOIN Customers ON Sales.customer_id = Customers.customer_id

JOIN Time ON Sales.time_id = Time.time_id

WHERE Products.category = 'Electronics'

AND Customers.age BETWEEN 20 AND 30

AND Time.year = 2023;
```

Rollup:

```
SELECT Time.year, SUM(Sales.sales_amount) AS total_sales

FROM Sales

JOIN Time ON Sales.time_id = Time.time_id

GROUP BY Time.year;
```

Practical 7:

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.datasets import load_iris

# Load the Iris dataset
iris = load_iris()
data = pd.DataFrame(iris.data, columns=iris.feature_names)
data['target'] = iris.target

# Features and target
X = data.drop(columns=['target'])  # Features (input)
y = data['target']  # Target (output)

# Split the dataset into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the Decision Tree classifier
clf = DecisionTreeClassifier()

# Train the classifier on the training data
clf.fit(X_train, y_train)

# Use the trained classifier to make predictions on the test data
y_pred = clf.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")

# Detailed classification report
print("Classification Report:\n", classification_report(y_test, y_pred))
```

Practical 8

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs

# Generate synthetic data for clustering
X, y = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=42)
```

```
# Visualize the dataset
plt.scatter(X[:, 0], X[:, 1], s=50)
plt.title('Dataset for Clustering')
plt.show()

# Initialize the K-Means model with 4 clusters
kmeans = KMeans(n_clusters=4, random_state=42)

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

# Get the cluster centers
centers = kmeans.cluster_centers_

# Predict the cluster for each data point
y_kmeans = kmeans.predict(X)

# Plot the clusters and their centers
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')
plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.75, marker='X')
plt.title('K-Means Clustering Results')
plt.show()
```