PRACTICE
DATA WAREHOUSE

Practise I

Setup PostGRSQL and Easy Data Modeling

Aims

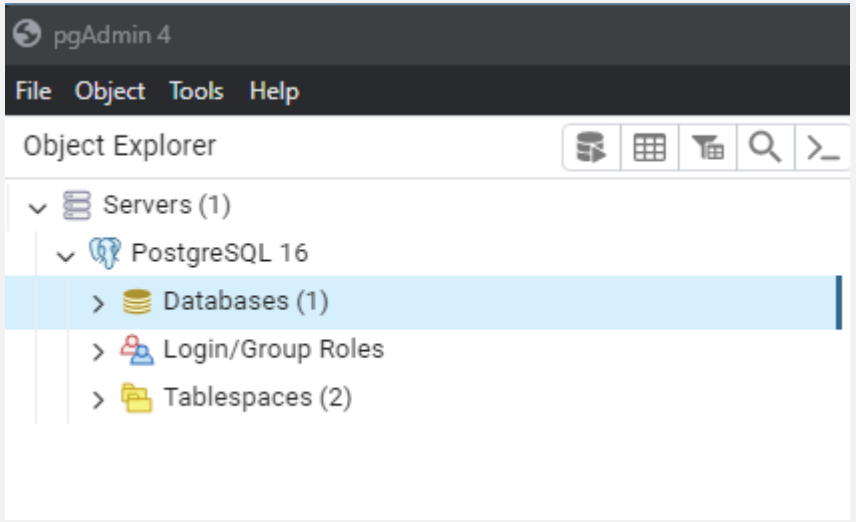
Setup Environment (Database)

Create tabels and relations from scratch

Create and execute SQL statements

PREPARATION
INSTALL POSTGRES SQL

- https://www.postgresql.org/download/
- Follow install guide, devault values, but no extra installation at the end
- Start pgAdmin 4



DATA WAREHOUSE - PRACTICE - DAY 1 - „DATA MODELING“

TASK

- **Create a simple data model for a fictional company named "OnlineShop." The data model should include basic entities and relationships for customer data, product information, and orders.**

1. Design the data model for the company "OnlineShop“ including relationships.
2. Identify the primary keys and foreign keys for each table.

Show 1 and 2 to lecturer

3. Create SQL DDL statements to create the tables in a relational database.
4. Insert some sample data into the tables to test the data model.

Show result

DATA WAREHOUSE - PRACTICE - DAY 1 - „DATA MODELING“

SOLUTION PART 11

1. Identification of entities and attributes:
   - **Customers: CustomerID, First Name, Last Name, Email, Address**
   - **Products: ProductID, Product Name, Description, Price, Category**
   - **Orders: OrderID, CustomerID (FK), ProductID (FK), Date, Quantity**
2. Table design:
   - **Customers (Customers): CustomerID (Primary Key), First Name, Last Name, Email, Address**
   - **Products (Products): ProductID (Primary Key), Product Name, Description, Price, Category**
   - **Orders (Orders): OrderID (Primary Key), CustomerID (Foreign Key), ProductID (Foreign Key), Date, Quantity**

3.Defining relationships:

- An order is assigned to a customer (1:n relationship between customers and orders).
- A product can be included in multiple orders (1:n relationship between products and orders).

4.Primary and Foreign Keys:

- Each table has a primary key that is unique.
- CustomerID and ProductID in the Orders table are foreign keys that reference the respective primary keys in the Customers and Products tables.

5.Extensions (optional):

- You can add additional attributes, such as customer registration date, order status, etc.

DATA WAREHOUSE - PRACTICE - DAY 1 - „DATA MODELING“

SOLUTION PART II

- CREATE TABLE Orders (
  OrderID INT PRIMARY KEY,
  CustomerID INT,
  ProductID INT,
  OrderDate DATE,
  Quantity INT,
  FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
  FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
  );

- CREATE TABLE Customers (
  CustomerID INT PRIMARY KEY,
  FirstName VARCHAR(50),
  LastName VARCHAR(50),
  Email VARCHAR(100),
  Address VARCHAR(255)
  );

- CREATE TABLE Products (
  ProductID INT PRIMARY KEY,
  ProductName VARCHAR(100),
  Description TEXT,
  Price DECIMAL(10, 2),
  Category VARCHAR(50)
  );

DATA WAREHOUSE - PRACTICE - DAY 1 - „DATA MODELING“

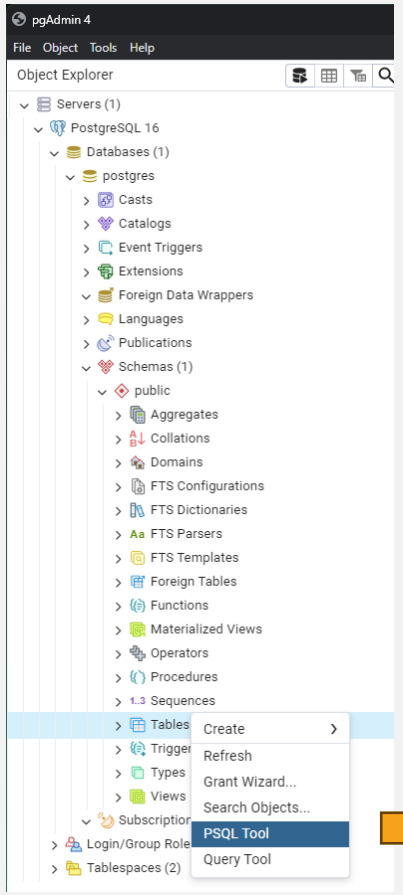
SOLUTION PART II

```
INSERT INTO Customers (CustomerID, FirstName, LastName, Email, Address) VALUES (1, 'John', 'Doe', 'john@example.com', '123 Main St‘);
```

```
INSERT INTO Products (ProductID, ProductName, Description, Price, Category) VALUES (1, 'Laptop', '15-inch Laptop', 999.99, 'Electronics‘);
```

```
INSERT INTO Orders (OrderID, CustomerID, ProductID, OrderDate, Quantity) VALUES (1, 1, 1, '2024-04-28', 2);
```

POSTGRSQL CREATE TABLE

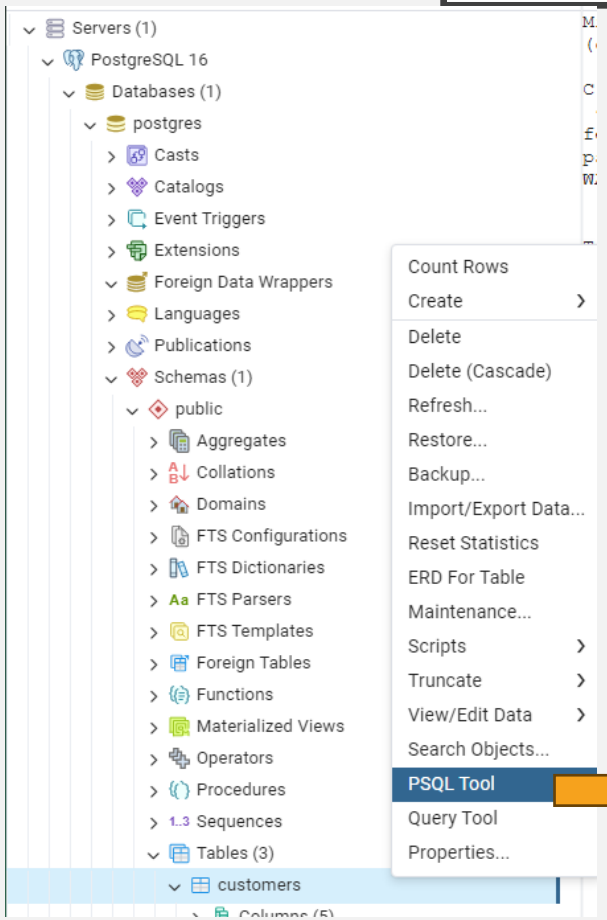



```
Microsoft Windows [Version 10.0.22631.3447]
(c) Microsoft Corporation. Alle Rechte vorbehalten.

C:\Program Files\PostgreSQL\16\pgAdmin 4\runtime>"C:\Program Files\PostgreSQL\16\pgAdmin
 4\runtime\psql.exe" "host=localhost port=5432 dbname=postgres user=postgres sslmode=pre
fer connect_timeout=10" 2>>&1
psql (16.2)
WARNING: Console code page (850) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

postgres=# CREATE TABLE Customers (
postgres(#     CustomerID INT PRIMARY KEY,
postgres(#     FirstName VARCHAR(50),
postgres(#     LastName VARCHAR(50),
postgres(#     Email VARCHAR(100),
postgres(#     Address VARCHAR(255)
postgres(# );
FEHLER:  Relation »customers« existiert bereits
postgres=#
postgres=# CREATE TABLE Products (
postgres(#     ProductID INT PRIMARY KEY,
postgres(#     ProductName VARCHAR(100),
postgres(#     Description TEXT,
postgres(#     Price DECIMAL(10, 2),
postgres(#     Category VARCHAR(50)
postgres(# );
FEHLER:  Relation »products« existiert bereits
postgres=#
postgres=# CREATE TABLE Orders (
postgres(#     OrderID INT PRIMARY KEY,
postgres(#     CustomerID INT,
postgres(#     ProductID INT,
postgres(#     OrderDate DATE,
postgres(#     Quantity INT,
postgres(#     FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
postgres(#     FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
postgres(# );
```

POSTGRSQL INSERT DATA




```
Microsoft Windows [Version 10.0.22631.3447]
(c) Microsoft Corporation. Alle Rechte vorbehalten.

C:\Program Files\PostgreSQL\16\pgAdmin 4\runtime>"C:\Program Files\PostgreSQL\16\pgAdmin
 4\runtime\psql.exe" "host=localhost port=5432 dbname=postgres user=postgres sslmode=pre
fer connect_timeout=10" 2>>&1
psql (16.2)
WARNING: Console code page (850) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

postgres=# INSERT INTO Customers (CustomerID, FirstName, LastName, Email, Address) VALUE
S (1, 'John', 'Doe', 'john@example.com', '123 Main St');
FEHLER:  doppelter Schlüsselwert verletzt Unique-Constraint »customers_pkey«
DETAIL:  Schlüssel »(customerid)=(1)« existiert bereits.
postgres=# INSERT INTO Products (ProductID, ProductName, Description, Price, Category) V
ALUES (1, 'Laptop', '15-inch Laptop', 999.99, 'Electronics');
FEHLER:  doppelter Schlüsselwert verletzt Unique-Constraint »products_pkey«
DETAIL:  Schlüssel »(productid)=(1)« existiert bereits.
postgres=# INSERT INTO Orders (OrderID, CustomerID, ProductID, OrderDate, Quantity) VALU
ES (1, 1, 1, '2024-04-28', 2);
FEHLER:  doppelter Schlüsselwert verletzt Unique-Constraint »orders_pkey«
DETAIL:  Schlüssel »(orderid)=(1)« existiert bereits.
postgres=#
```

PRACTICE
DATA WAREHOUSE

Practise 2

ETL

Aims

Extract data from different sources

Transform and clean data

Load data into database

DATA WAREHOUSE - PRACTICE 2 - „ETL“

TASK

Perform an ETL (Extraction, Transformation, Loading) exercise where you extract data from various sources, transform it, and load it into your data model. This can be done using simple tools like Excel or more complex tools like SQL. Note: Data may be flawed.

1. Extract customer data from a CSV file and product data from an JSON spreadsheet.
2. Transform the extracted data to prepare it for the data model (e.g., cleaning, formatting).
3. Load the transformed data into the corresponding tables of the previously created data model.

DATA WAREHOUSE - PRACTICE 2 - „ETL“

TASK

- **Customer.csv**

CustomerID,FirstName,LastName,Email,Address

1,John,Doe,john@example.com,123 Main St

2,Jane,Smith,jane@example.com,456 Oak St

3,,Johnson,michael@example.com,789 Elm St

4,Emily,Williams,,emily@example.com,101 Pine St

5,Chris,Brown,chris@example.com,

- **Product.json**

[{ "ProductID": 1, "ProductName": "Laptop", "Description": "15-inch Laptop", "Price": 999.99, "Category": "Electronics" }, { "ProductID": 2, "ProductName": "Smartphone", "Price": 499.99, "Category": "Electronics" }, { "ProductID": 3, "ProductName": "Headphones", "Description": "Wireless Headphones" }, { "ProductID": 4, "Description": "Introduction to Data Science", "Price": 29.99, "Category": "Books" }, { "ProductID": 5, "ProductName": "Tablet", "Description": "10-inch Tablet", "Price": 299.99, "Category": "Electronics" }]

DATA WAREHOUSE - PRACTICE 2 - „ETL“

SOLUTION

1. **Extraktion**
   Verwenden Sie geeignete Tools oder Programmiersprachen, um die Daten aus den Quellen zu extrahieren. Zum Beispiel können Python-Bibliotheken wie pandas für diese Aufgabe verwendet werden.

2. **Transformation**
   Verwenden Sie Skripts oder Funktionen, um die extrahierten Daten zu transformieren. Zum Beispiel können Sie in Python pandas verwenden, um Daten zu bereinigen und zu formatieren.

3. **Laden**
   Verwenden Sie SQL-Insert-Anweisungen oder Importfunktionen, um die transformierten Daten in die Datenbank des Datenmodells zu laden. Stellen Sie sicher, dass die Daten entsprechend den Tabellendefinitionen eingefügt werden.

In dieser modifizierten Version der customers.csv-Datei gibt es folgende Fehler:

- Im dritten Datensatz fehlt der Vorname.
- Im vierten Datensatz fehlt die Adresse.
- Im fünften Datensatz fehlt die E-Mail-Adresse.

In dieser modifizierten Version der products.csv-Datei gibt es folgende Fehler:

- Im zweiten Datensatz fehlt die Beschreibung.
- Im dritten Datensatz fehlt der Preis.
- Im vierten Datensatz fehlt der Produktname und die Kategorie.
- Im vierten Datensatz fehlt die Kategorie.

PRACTICE
DATA WAREHOUSE

Practise III

MultiDimensional Data

Create a real datawarehouse

Aims

Use a real business scenario to

I. Udentify facts and dimension

II., Create fact tables

III. Create dimension tables

DATA WAREHOUSE - PRACTICE 3 – „MULTIDIMENSIONAL DATA“

TASK DETAILS

- The Basic scenario

A retail company wants to set up a data warehouse to analyze its sales data. The company sells a variety of products through various distribution channels and branches. It wants to gain insights into sales, product sales, customer behavior and the success of sales campaigns.

- Task1: Think about facts and dimensions
  - Facts
    - Facts are the quantifiable data points to be analyzed. So, for quantifiable measurements. These could include, for example, sales, number of products sold, discounts, etc.
  - Dimensions
    - Dimensions are contextual information about the facts. These can be dimensions such as time, product, customer, store, seller, etc.

DATA WAREHOUSE - PRACTICE 3 – „MULTIDIMENSIONAL DATA“

SOLUTION

- **Facts**
- Sales, product units sold, discounts, costs.

- **Dimension**
- Time (date, month, year),
- product (product name, category, SKU),
- customer (customer name, region, customer type), store (branch number, location, store type),
- salesperson (name, employee ID, department).

TASK II: CREATE FACTS AND DIMENSIONS TABLES WITH DATA IN SQL

DATA WAREHOUSE - PRACTICE 3 – „MULTIDIMENSIONAL DATA“

SOLUTION

- Fact table for sales:
  - DateID (from time dimension),
  - ProductID (from product dimension),
  - CustomerID (from customer dimension),
  - BranchID (from branch dimension),
  - SellerID (from seller dimension), sales, discount, costs.
- Additional fact tables can be created for other quantifiable measurements, such as: E.g. sales units, costs, etc.

- Dimension table for time:
  - DateID, Date, Day, Month, Quarter, Year.

  Dimension table for product:
  - ProductID, Product Name, Category, SKU, Manufacturer.
- More dimension tables for customers, stores and sellers according to the identified dimensions.

DATA WAREHOUSE - PRACTICE 3 – „MULTIDIMENSIONAL DATA“

SOLUTION-CREATE DATABASE 1/3

Create fact-table for salesvalues

```
CREATE TABLE SalesFact (
    SaleID INT PRIMARY KEY,
    DateID INT,
    ProductID INT,
    CustomerID INT,
    StoreID INT,
    EmployeeID INT,
    Revenue DECIMAL(10, 2),
    Discount DECIMAL(10, 2),
    Cost DECIMAL(10, 2),
    FOREIGN KEY (DateID) REFERENCES TimeDimension(DateID),
    FOREIGN KEY (ProductID) REFERENCES ProductDimension(ProductID),
    FOREIGN KEY (CustomerID) REFERENCES CustomerDimension(CustomerID),
    FOREIGN KEY (StoreID) REFERENCES StoreDimension(StoreID),
    FOREIGN KEY (EmployeeID) REFERENCES EmployeeDimension(EmployeeID)
);
```

Create dimension-table for time

```
CREATE TABLE TimeDimension (
    DateID INT PRIMARY KEY,
    Date DATE,
    Day INT,
    Month INT,
    Quarter INT,
    Year INT
);
```

DATA WAREHOUSE - PRACTICE 3 – „MULTIDIMENSIONAL DATA“

SOLUTION-CREATE DATABASE 2/3

Create dimension-table for product

```
CREATE TABLE ProductDimension (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(100),
    Category VARCHAR(50),
    SKU VARCHAR(50),
    Manufacturer VARCHAR(100)
);
```

Create dimension-table for store

```
CREATE TABLE StoreDimension (
    StoreID INT PRIMARY KEY,
    StoreNumber INT,
    Location VARCHAR(100),
    StoreType VARCHAR(50)
);
```

DATA WAREHOUSE - PRACTICE 3 – „MULTIDIMENSIONAL DATA“

SOLUTION-CREATE DATABASE 3/3

Create dimension-table for employee

```
CREATE TABLE EmployeeDimension (
    EmployeeID INT PRIMARY KEY,
    EmployeeName VARCHAR(100),
    EmployeeNumber INT,
    Department VARCHAR(50)
);
```

DATA WAREHOUSE - PRACTICE 3 – „MULTIDIMENSIONAL DATA“

SOLUTION-EXAMPLE DATA FOR DATABASE 1/3

Data for fact-table: SalesFact

```
INSERT INTO SalesFact (SaleID, DateID,
ProductID, CustomerID, StoreID, EmployeeID,
Revenue, Discount, Cost)
VALUES
(1, 1, 1, 1, 1, 1, 1000.00, 50.00, 800.00),
(2, 2, 2, 2, 2, 2, 1500.00, 100.00, 1200.00),
(3, 3, 3, 3, 1, 3, 2000.00, 200.00, 1600.00);
```

Data for dimension-table: TimeDimension

```
INSERT INTO TimeDimension (DateID, Date,
Day, Month, Quarter, Year)
VALUES
(1, '2024-01-01', 1, 1, 1, 2024),
(2, '2024-01-02', 2, 1, 1, 2024),
(3, '2024-01-03', 3, 1, 1, 2024);
```

DATA WAREHOUSE - PRACTICE 3 – „MULTIDIMENSIONAL DATA“

SOLUTION-EXAMPLE DATA FOR DATABASE 2/3

Data for dimension-table : ProductDimension

```
INSERT INTO ProductDimension (ProductID,
ProductName, Category, SKU, Manufacturer)
VALUES
(1, 'Laptop', 'Electronics', 'LT1001', 'ABC Electronics'),
(2, 'Smartphone', 'Electronics', 'SP2001', 'XYZ Mobiles'),
(3, 'Headphones', 'Electronics', 'HP3001', 'PQR Audio');
```

Data for dimension-table : CustomerDimension

```
INSERT INTO CustomerDimension (CustomerID,
CustomerName, Region, CustomerType)
VALUES
(1, 'John Doe', 'North', 'Regular'),
(2, 'Jane Smith', 'South', 'Premium'),
(3, 'Michael Johnson', 'East', 'Regular');
```

DATA WAREHOUSE - PRACTICE 3 – „MULTIDIMENSIONAL DATA“

SOLUTION-EXAMPLE DATA FOR DATABASE 3/3

Data for dimension-table : "StoreDimension

```
INSERT INTO StoreDimension (StoreID, StoreNumber, Location, StoreType)
VALUES
(1, 101, 'New York', 'Flagship'),
(2, 202, 'Los Angeles', 'Outlet'),
(3, 303, 'Chicago', 'Franchise');
```

Data for dimension-table : "EmployeeDimension

```
INSERT INTO EmployeeDimension (EmployeeID, EmployeeName, EmployeeNumber, Department)
VALUES
(1, 'John Smith', 1001, 'Sales'),
(2, 'Jane Doe', 1002, 'Marketing'),
(3, 'Michael Johnson', 1003, 'Finance');
```

PRACTICE
DATA WAREHOUSE

Practise IV

Complex SQL Statements for Datawarehouse

Aims

Practise Data Mart View

TASKS 1-4

Task 1: Calculate Total Sales per Year

Write an SQL query to calculate the total sales for each year.

Task 2: Calculate Average Sales per Product Category

Write an SQL query to calculate the average sales for each product category.

Task 3: Calculate Sales by Region and Year

Write an SQL query to calculate the sales for each region and each year.

Task 4: Identify Products with the Highest Sales

Write an SQL query to identify the top 5 products by sales.

TASKS 5-8

Task 5: Analyze Sales by Individual Sellers

Write an SQL query to calculate the total sales for each seller.

Task 6: Identify Customers with the Highest Sales

Write an SQL query to identify the top 3 customers by sales.

Task 7: Calculate Sales and Discounts per Month

Write an SQL query to calculate the total sales and total discounts for each month.

Task 8: Calculate Sales Units per Product and Store

Write an SQL query to calculate the number of units sold for each product in each store.

TASK 1-SOLUTION

```
SELECT
    t.Year,
    SUM(sf.Revenue) AS TotalRevenue
FROM
    SalesFact sf
JOIN
    TimeDimension t ON sf.DateID = t.DateID
GROUP BY
    t.Year
ORDER BY
    t.Year;
```

TASK II-SOLUTION

```
SELECT
    pd.Category,
    AVG(sf.Revenue) AS AverageRevenue
FROM
    SalesFact sf
JOIN
    ProductDimension pd ON sf.ProductID = pd.ProductID
GROUP BY
    pd.Category
ORDER BY
    pd.Category;
```

TASK III-SOLUTION

```
SELECT
    cd.Region,
    t.Year,
    SUM(sf.Revenue) AS TotalRevenue
FROM
    SalesFact sf
JOIN
    CustomerDimension cd ON sf.CustomerID = cd.CustomerID
JOIN
    TimeDimension t ON sf.DateID = t.DateID
GROUP BY
    cd.Region, t.Year
ORDER BY
    cd.Region, t.Year;
```

TASK IV-SOLUTION

```
SELECT
    pd.ProductName,
    SUM(sf.Revenue) AS TotalRevenue
FROM
    SalesFact sf
JOIN
    ProductDimension pd ON sf.ProductID = pd.ProductID
GROUP BY
    pd.ProductName
ORDER BY
    TotalRevenue DESC
LIMIT
    5;
```

TASK V-SOLUTION

```
SELECT
    ed.EmployeeName,
    SUM(sf.Revenue) AS TotalRevenue
FROM
    SalesFact sf
JOIN
    EmployeeDimension ed ON sf.EmployeeID = ed.EmployeeID
GROUP BY
    ed.EmployeeName
ORDER BY
    TotalRevenue DESC;
```

TASK V1-SOLUTION

```
SELECT
    cd.CustomerName,
    SUM(sf.Revenue) AS TotalRevenue
FROM
    SalesFact sf
JOIN
    CustomerDimension cd ON sf.CustomerID = cd.CustomerID
GROUP BY
    cd.CustomerName
ORDER BY
    TotalRevenue DESC
LIMIT
    3;
```

TASK VII-SOLUTION

```
SELECT
    t.Year,
    t.Month,
    SUM(sf.Revenue) AS TotalRevenue,
    SUM(sf.Discount) AS TotalDiscount
FROM
    SalesFact sf
JOIN
    TimeDimension t ON sf.DateID = t.DateID
GROUP BY
    t.Year, t.Month
ORDER BY
    t.Year, t.Month;
```

TASK VIII-SOLUTION

```
SELECT
    pd.ProductName,
    sd.Location AS StoreLocation,
    SUM(sf.Quantity) AS TotalUnitsSold
FROM
    SalesFact sf
JOIN
    ProductDimension pd ON sf.ProductID = pd.ProductID
JOIN
    StoreDimension sd ON sf.StoreID = sd.StoreID
GROUP BY
    pd.ProductName, sd.Location
ORDER BY
    pd.ProductName, sd.Location;
```

FACT TABE FOR SALES (SALESFACT)

- CREATE TABLE SalesFact (SaleID INT PRIMARY KEY, DateID INT, ProductID INT, CustomerID INT, StoreID INT, EmployeeID INT, Revenue DECIMAL(10, 2), Discount DECIMAL(10, 2), Quantity INT, FOREIGN KEY (DateID) REFERENCES TimeDimension(DateID), FOREIGN KEY (ProductID) REFERENCES ProductDimension(ProductID), FOREIGN KEY (CustomerID) REFERENCES CustomerDimension(CustomerID), FOREIGN KEY (StoreID) REFERENCES StoreDimension(StoreID), FOREIGN KEY (EmployeeID) REFERENCES EmployeeDimension(EmployeeID));

DIMENSION TABLE FOR TIME (TIMEDIMENSION)

- CREATE TABLE TimeDimension (DateID INT PRIMARY KEY, Date DATE, Day INT, Month INT, Quarter INT, Year INT);

DIMENSION TABLE FOR PRODUCT (PRODUCTDIMENSION)

- CREATE TABLE ProductDimension (ProductID INT PRIMARY KEY, ProductName VARCHAR(100), Category VARCHAR(50), SKU VARCHAR(50), Manufacturer VARCHAR(100));

DIMENSION TABLE FOR CUSTOMER (CUSTOMERDIMENSION)

- CREATE TABLE CustomerDimension (CustomerID INT PRIMARY KEY, CustomerName VARCHAR(100), Region VARCHAR(50), CustomerType VARCHAR(50));

DIMENSION TABLE FOR STORE (STOREDIMENSION)

- CREATE TABLE StoreDimension (StoreID INT PRIMARY KEY, StoreNumber INT, Location VARCHAR(100), StoreType VARCHAR(50));

DIMENSION TABLE FOR EMPLOYEE (EMPLOYEEDIMENSION)

- CREATE TABLE EmployeeDimension (EmployeeID INT PRIMARY KEY, EmployeeName VARCHAR(100), EmployeeNumber INT, Department VARCHAR(50));

EXAMPLE DATA FOR "SALESFACT"

- INSERT INTO SalesFact (SaleID, DateID, ProductID, CustomerID, StoreID, EmployeeID, Revenue, Discount, Quantity) VALUES (1, 1, 1, 1, 1, 1, 1000.00, 50.00, 10), (2, 2, 2, 2, 2, 2, 1500.00, 100.00, 20), (3, 3, 3, 3, 1, 3, 2000.00, 200.00, 15), (4, 4, 4, 4, 2, 1, 2500.00, 150.00, 25), (5, 5, 1, 2, 3, 2, 3000.00, 100.00, 30);

EXAMPLE DATA FOR "TIMEDIMENSION"

- INSERT INTO TimeDimension (DateID, Date, Day, Month, Quarter, Year) VALUES (1, '2024-01-01', 1, 1, 1, 2024), (2, '2024-02-01', 1, 2, 1, 2024), (3, '2024-03-01', 1, 3, 1, 2024), (4, '2024-04-01', 1, 4, 2, 2024), (5, '2024-05-01', 1, 5, 2, 2024);

EXAMPLE DATA FOR "PRODUCTDIMENSION"

- INSERT INTO ProductDimension (ProductID, ProductName, Category, SKU, Manufacturer) VALUES (1, 'Laptop', 'Electronics', 'LT1001', 'ABC Electronics'), (2, 'Smartphone', 'Electronics', 'SP2001', 'XYZ Mobiles'), (3, 'Headphones', 'Electronics', 'HP3001', 'PQR Audio'), (4, 'Tablet', 'Electronics', 'TB4001', 'LMN Devices');

EXAMPLE DATA FOR "CUSTOMERDIMENSION"

- INSERT INTO CustomerDimension (CustomerID, CustomerName, Region, CustomerType) VALUES (1, 'John Doe', 'North', 'Regular'), (2, 'Jane Smith', 'South', 'Premium'), (3, 'Michael Johnson', 'East', 'Regular'), (4, 'Emily Davis', 'West', 'Premium');

EXAMPLE DATA FOR "STOREDIMENSION"

- INSERT INTO StoreDimension (StoreID, StoreNumber, Location, StoreType) VALUES (1, 101, 'New York', 'Flagship'), (2, 202, 'Los Angeles', 'Outlet'), (3, 303, 'Chicago', 'Franchise'), (4, 404, 'Houston', 'Flagship');

EXAMPLE DATA FOR "EMPLOYEEDIMENSION"

- INSERT INTO EmployeeDimension (EmployeeID, EmployeeName, EmployeeNumber, Department) VALUES (1, 'John Smith', 1001, 'Sales'), (2, 'Jane Doe', 1002, 'Marketing'), (3, 'Michael Johnson', 1003, 'Finance'), (4, 'Emily Davis', 1004, 'Sales');

PRACTICE
DATA WAREHOUSE

Practise 5

Facts

Aims

Create:

Fully Additive Facts: 15 minutes
Semi-Additive Facts: 15 minutes
Non-Additive Facts: 15 minutes
Schema Design Task: 15 minutes

Exercise 1: Fully Additive Facts

Problem: A retail company stores sales data in a data warehouse. The fact table Sales includes the following columns:

- DateKey (Foreign Key)
- StoreKey (Foreign Key)
- ProductKey (Foreign Key)
- QuantitySold (Number of units sold)
- TotalSales (Total sales amount in dollars)

Questions:

1. Write an SQL query to calculate the total sales (TotalSales) for each store (StoreKey) across all dates.
2. Write an SQL query to calculate the total quantity sold (QuantitySold) for each product (ProductKey) across all dates.

Exercise 2: Semi-Additive Facts

Problem: A banking institution tracks daily account balances in a data warehouse. The fact table AccountBalances includes the following columns:

- DateKey (Foreign Key)
- AccountKey (Foreign Key)
- EndOfDayBalance (Balance at the end of the day)

Questions:

1. Write an SQL query to calculate the end-of-day balance for each account (AccountKey) for the last date in the dataset.
2. Write an SQL query to calculate the average end-of-day balance for each account (AccountKey) over all dates.

Exercise 3: Non-Additive Facts

Problem: A survey company stores survey results in a data warehouse. The fact table SurveyResults includes the following columns:

- SurveyDateKey (Foreign Key)
- RespondentKey (Foreign Key)
- SatisfactionRating (Rating from 1 to 5)

Questions:

1. Write an SQL query to find the average satisfaction rating (SatisfactionRating) for each survey date (SurveyDateKey).
2. Write an SQL query to find the highest satisfaction rating (SatisfactionRating) for each respondent (RespondentKey).

Exercise 4: STAR SCHEMA

Task: Ask the students to create a star schema for a new data warehouse designed to track online course enrollments. The schema should include:

- A fact table with relevant metrics (e.g., number of enrollments, total revenue).
- Dimension tables for Course, Student, and Date.

Expected Outcome: Students should provide a schema diagram showing the fact table and its relationships with the dimension tables, including primary and foreign keys, and some example fields in each table.

SOLUTION EXERCISE 1

- SELECT StoreKey, SUM(TotalSales) AS TotalSales
- FROM Sales
- GROUP BY StoreKey;

- SELECT ProductKey, SUM(QuantitySold) AS TotalQuantitySold
- FROM Sales
- GROUP BY ProductKey;

SOLUTION EXERCISE 11

- SELECT AccountKey, EndOfDayBalance
- FROM AccountBalances
- WHERE DateKey = (SELECT MAX(DateKey) FROM AccountBalances);

- SELECT AccountKey, AVG(EndOfDayBalance) AS AverageBalance
- FROM AccountBalances
- GROUP BY AccountKey;

SOLUTION EXERCISE III

- SELECT SurveyDateKey, AVG(SatisfactionRating) AS AverageRating
- FROM SurveyResults
- GROUP BY SurveyDateKey;

- SELECT RespondentKey, MAX(SatisfactionRating) AS HighestRating
- FROM SurveyResults
- GROUP BY RespondentKey;

PRACTICE
DATA WAREHOUSE

Practise 7

Tutorial: From dimensional model to stunning report in Power BI Desktop

LINK FOR TUTORIAL

Step 1: Login in university pc or install PowerBI Desktop

Step 2: Open tutorial -

https://learn.microsoft.com/en-us/power-bi/create-reports/desktop-dimensional-model-report?source=recommendations

TUTORIAL 8
SCENARIO: ONLINE RETAIL SALES ANALYSIS

1. **Data Import**: Import the provided tables into Power BI Desktop. Check Tutorial 7

2. **Modeling**: Construct a snowflake schema using the provided entities and relationships. Ensure each dimension table (Customer, Product, Category, Subcategory, Manufacturer) is properly linked to the Sales Fact Table.

3. **Measure Creation**: Create measures such as Total Sales, Quantity Sold, Average Sales per Customer, etc., based on the Sales Fact Table.

4. **Visualization**: Design meaningful visualizations (e.g., sales trends over time, sales by category, top customers by sales amount) using the relationships established in your snowflake schema.

5. **Analysis**: Perform analysis to derive insights from the data. For example, identify best-selling products, analyze sales performance across different regions, or compare sales across different categories and subcategories.

ENTITIES

1. **Sales Fact Table**:
    1. Fields: OrderID (PK), OrderDate, CustomerID (FK), ProductID (FK), QuantitySold, SalesAmount
2. **Customer Dimension Table**:
    1. Fields: CustomerID (PK), CustomerName, CustomerCity, CustomerState, CustomerCountry
3. **Product Dimension Table**:
    1. Fields: ProductID (PK), ProductName, CategoryID (FK), SubcategoryID (FK), ManufacturerID (FK), UnitPrice
4. **Category Dimension Table**:
    1. Fields: CategoryID (PK), CategoryName
5. **Subcategory Dimension Table**:
    1. Fields: SubcategoryID (PK), SubcategoryName, CategoryID (FK)
6. **Manufacturer Dimension Table**:
    1. Fields: ManufacturerID (PK), ManufacturerName

RELATIONSHIPS

- **Sales Fact Table**:
  - FK relationship with **Customer Dimension Table** on CustomerID
  - FK relationship with **Product Dimension Table** on ProductID
- **Product Dimension Table**:
  - FK relationship with **Category Dimension Table** on CategoryID
  - FK relationship with **Subcategory Dimension Table** on SubcategoryID
  - FK relationship with **Manufacturer Dimension Table** on ManufacturerID

PRACTICE
DATA WAREHOUSE

Practise 8

Tutorial: How to Perform ETL Process Using Excel

Follow instructions on https://medium.com/@pratikshamadivala/how-to-perform-etl-process-using-excel-8db81966ffbe
1 Perform ETL
2 Check dimensional modeling