**TY B.Tech. (CSE) – II [2022-23]**

**5CS372: Advanced Database System Lab.**

**Assignment No. 6**

**Name : Tanaya Mukund Bhide**

**PRN: 2020BTECS00011**

**Batch: T5**

**Branch: T.Y CSE**

**To design and implement a data warehouse for a customer order processing system in a company. [ Use any Database ]**

headquarter ;

CREATE TABLE Customer (

Customer\_id INT PRIMARY KEY,

Customer\_name VARCHAR(50) NOT NULL,

City\_id INT,

First\_order\_date DATE

);

CREATE TABLE Walk\_in\_customers (

Customer\_id INT REFERENCES Customer(Customer\_id),

tourism\_guide VARCHAR(50),

Time TIMESTAMP

);

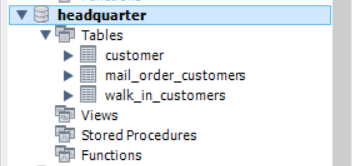
CREATE TABLE Mail\_order\_customers (

Customer\_id INT REFERENCES Customer(Customer\_id),

post\_address VARCHAR(100),

Time TIMESTAMP

);



sales

CREATE TABLE Headqarters (

City\_id INT PRIMARY KEY,

City\_name VARCHAR(50) NOT NULL,

Headquarter\_addr VARCHAR(100),

State VARCHAR(50),

Time TIMESTAMP

);

CREATE TABLE Stores (

Store\_id INT PRIMARY KEY,

City\_id INT REFERENCES Headqarters(City\_id),

Phone VARCHAR(20),

Time TIMESTAMP

);

CREATE TABLE Items (

Item\_id INT PRIMARY KEY,

Description VARCHAR(100),

Size VARCHAR(10),

Weight DECIMAL(5,2),

Unit\_price DECIMAL(10,2),

Time TIMESTAMP

);

CREATE TABLE Stored\_items (

Store\_id INT REFERENCES Stores(Store\_id),

Item\_id INT REFERENCES Items(Item\_id),

Quantity\_held INT,

Time TIMESTAMP

);

CREATE TABLE Order1 (

Order\_no INT PRIMARY KEY,

Order\_date DATE,

Customer\_id INT REFERENCES Customer(Customer\_id)

);

CREATE TABLE Ordered\_item (

Order\_no INT REFERENCES Order1(Order\_no),

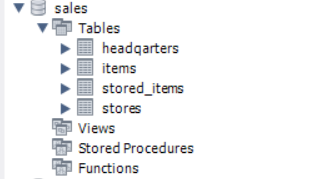
Item\_id INT REFERENCES Items(Item\_id),

Quantity\_ordered INT,

Ordered\_price DECIMAL(10,2),

Time TIMESTAMP

);



Populating the tables :

Use Headquarters ;

INSERT INTO Customer (Customer\_id, Customer\_name, City\_id, First\_order\_date)

VALUES (1, 'John Smith', 1, '2022-01-01'),

(2, 'Jane Doe', 2, '2021-12-15'),

(3, 'Bob Johnson', 3, '2022-02-20');

INSERT INTO Walk\_in\_customers (Customer\_id, tourism\_guide, Time)

VALUES (1, 'Tom', '2022-03-01 10:00:00'),

(2, 'Kate', '2022-03-02 11:00:00'),

(3, 'Mark', '2022-03-03 12:00:00');

INSERT INTO Mail\_order\_customers (Customer\_id, post\_address, Time)

VALUES (1, '123 Main St, Anytown, USA', '2022-03-01 10:00:00'),

(2, '456 Oak Ave, Othertown, USA', '2022-03-02 11:00:00'),

(3, '789 Elm Blvd, Thirdtown, USA', '2022-03-03 12:00:00');

use sales ;

INSERT INTO Headqarters (City\_id, City\_name, Headquarter\_addr, State, Time)

VALUES (1, 'New York City', '123 Main Street', 'New York', '2022-03-31 12:00:00'),

(2, 'Los Angeles', '456 Elm Street', 'California', '2022-03-31 12:00:00'),

(3, 'Chicago', '789 Oak Street', 'Illinois', '2022-03-31 12:00:00');

INSERT INTO Stores (Store\_id, City\_id, Phone, Time)

VALUES (1, 1, '212-555-1234', '2022-03-31 12:00:00'),

(2, 1, '212-555-5678', '2022-03-31 12:00:00'),

(3, 2, '213-555-1234', '2022-03-31 12:00:00'),

(4, 2, '213-555-5678', '2022-03-31 12:00:00'),

(5, 3, '312-555-1234', '2022-03-31 12:00:00'),

(6, 3, '312-555-5678', '2022-03-31 12:00:00');

INSERT INTO Items (Item\_id, Description, Size, Weight, Unit\_price, Time)

VALUES (1, 'T-Shirt', 'M', 0.2, 10.99, '2022-03-31 12:00:00'),

(2, 'Jeans', '32x32', 0.6, 49.99, '2022-03-31 12:00:00'),

(3, 'Sweater', 'L', 0.4, 29.99, '2022-03-31 12:00:00'),

(4, 'Dress', 'S', 0.3, 39.99, '2022-03-31 12:00:00');

INSERT INTO Stored\_items (Store\_id, Item\_id, Quantity\_held, Time)

VALUES (1, 1, 50, '2022-03-31 12:00:00'),

(1, 2, 25, '2022-03-31 12:00:00'),

(2, 3, 40, '2022-03-31 12:00:00'),

(3, 1, 75, '2022-03-31 12:00:00'),

(3, 2, 60, '2022-03-31 12:00:00'),

(4, 4, 20, '2022-03-31 12:00:00'),

(5, 1, 100, '2022-03-31 12:00:00'),

(6, 3, 50, '2022-03-31 12:00:00');

INSERT INTO Order1 (Order\_no, Order\_date, Customer\_id) VALUES

(1, '2023-03-31', 1),

(2, '2023-03-28', 2),

(3, '2023-04-02', 3);

INSERT INTO Ordered\_item (Order\_no, Item\_id, Quantity\_ordered, Ordered\_price, Time) VALUES

(1, 1, 2, 10.99, NOW()),

(1, 3, 1, 19.99, NOW()),

(2, 2, 3, 5.99, NOW()),

(2, 4, 2, 15.99, NOW()),

(3, 5, 4, 2.99, NOW());

Build data warehouse / OLAP which will answer the following queries :

1. Find all the stores along with city, state, phone, description, size, weight and unit price that hold a particular item of stock

use data\_warehouse;

CREATE TABLE stores\_with\_stock (

store\_id INT,

city VARCHAR(50),

state VARCHAR(50),

phone VARCHAR(20),

description VARCHAR(100),

size VARCHAR(10),

weight DECIMAL(5,2),

unit\_price DECIMAL(10,2)

);

INSERT INTO data\_warehouse.stores\_with\_stock

SELECT s.Store\_id, h.City\_name, h.State, s.Phone, i.Description, i.Size, i.Weight, i.Unit\_price

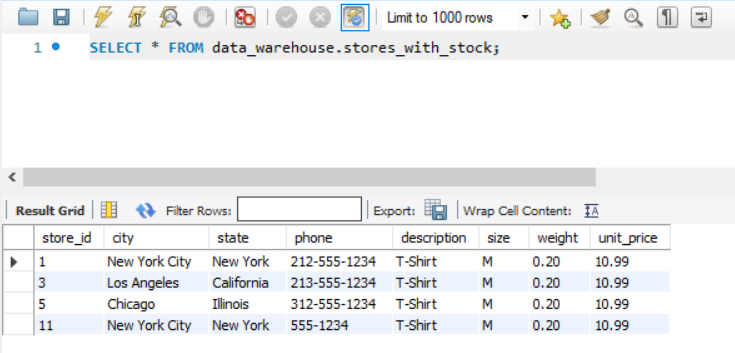
FROM sales.Stored\_items si

JOIN sales.Stores s ON si.Store\_id = s.Store\_id

JOIN sales.Items i ON si.Item\_id = i.Item\_id

JOIN sales.headqarters h ON s.City\_id = h.City\_id

WHERE i.Description = 'T-Shirt';



2. Find all the orders along with customer name and order date that can be fulfilled by a given store.

use data\_warehouse ;

CREATE TABLE orders\_with\_customer\_info (

order\_id INT,

customer\_name VARCHAR(100),

order\_date DATE,

store\_id INT

);

INSERT INTO orders\_with\_customer\_info

SELECT o.Order\_no, c.Customer\_name, o.Order\_date, si.Store\_id

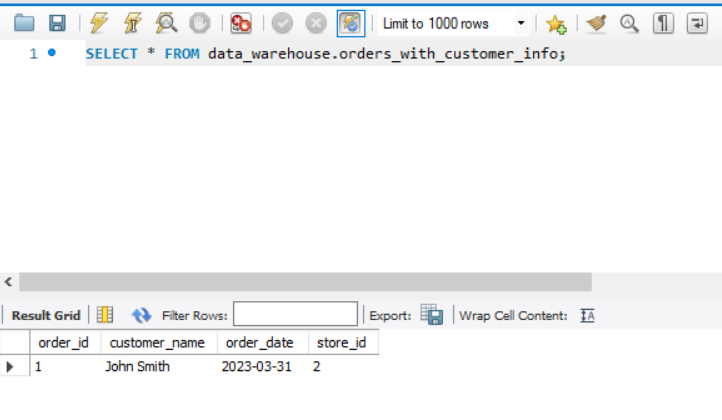
FROM sales.Order1 o

JOIN headquarter.Customer c ON o.Customer\_id = c.Customer\_id

JOIN sales.Ordered\_item oi ON o.Order\_no = oi.Order\_no

JOIN sales.Stored\_items si ON oi.Item\_id = si.Item\_id

WHERE si.Store\_id = 2;



3. Find all stores along with city name and phone that hold items ordered by given customer

USE data\_warehouse;

CREATE TABLE stores\_with\_ordered\_items (

store\_id INT,

city\_name VARCHAR(50),

phone VARCHAR(20),

item\_description VARCHAR(100)

);

INSERT INTO stores\_with\_ordered\_items

SELECT s.Store\_id, h.City\_name, s.Phone, i.Description

FROM sales.Order1 o

JOIN headquarter.Customer c ON o.Customer\_id = c.Customer\_id

JOIN sales.Ordered\_item oi ON o.Order\_no = oi.Order\_no

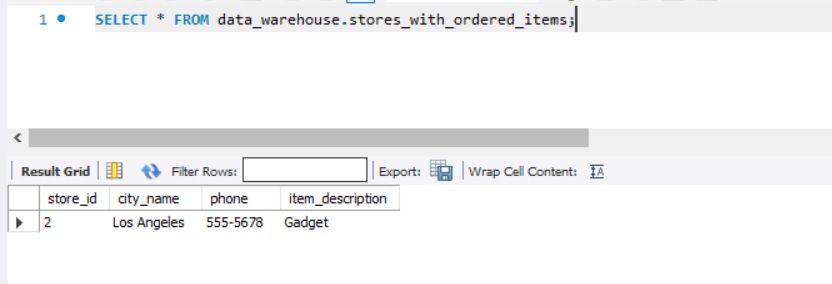
JOIN sales.Stored\_items si ON oi.Item\_id = si.Item\_id

JOIN sales.Stores s ON si.Store\_id = s.Store\_id

JOIN sales.Items i ON oi.Item\_id = i.Item\_id

JOIN sales.Headqarters h ON s.City\_id = h.City\_id

WHERE c.Customer\_name = 'Jane Smith';



4. Find the headquarter address along with city and state of all stores that hold stocks of an item above a particular level.

use data\_warehouse ;

CREATE TABLE data\_warehouse.stores\_with\_stock2 (

store\_id INT,

city VARCHAR(50),

state VARCHAR(50),

headquarter\_addr VARCHAR(100),

description VARCHAR(100),

size VARCHAR(10),

weight DECIMAL(5,2),

unit\_price DECIMAL(10,2),

quantity\_held INT,

PRIMARY KEY (store\_id, description)

);

INSERT INTO data\_warehouse.stores\_with\_stock2 (store\_id, city, state, headquarter\_addr, description, size, weight, unit\_price, quantity\_held)

SELECT s.Store\_id, h.City\_name, h.State, h.Headquarter\_addr, i.Description, i.Size, i.Weight, i.Unit\_price, si.Quantity\_held

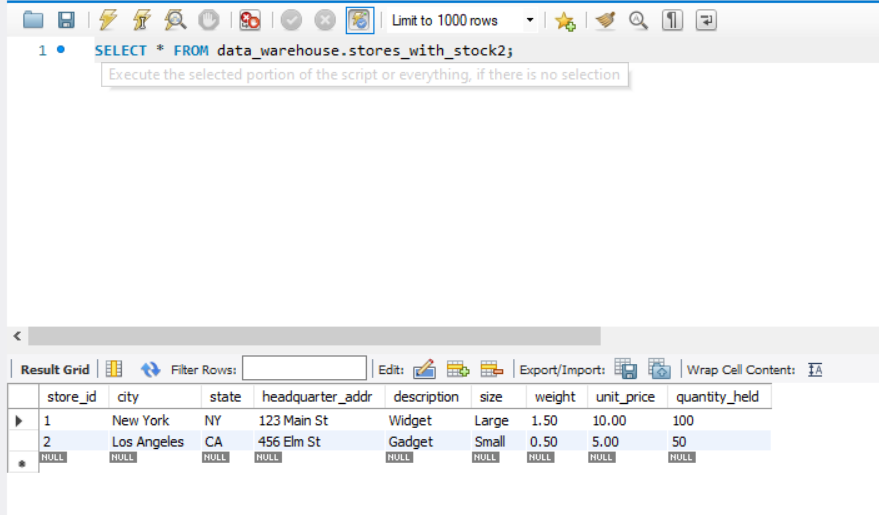
FROM sales.Stored\_items si

INNER JOIN sales.Items i ON si.Item\_id = i.Item\_id

INNER JOIN sales.Stores s ON si.Store\_id = s.Store\_id

INNER JOIN sales.Headqarters h ON s.City\_id = h.City\_id

WHERE si.Quantity\_held > 10;



5.For each customer order, show the items ordered along with description, store id and city name and the stores that hold the items.

CREATE TABLE data\_warehouse.order\_details\_customer (

Order\_no INT,

Order\_date DATE,

Item\_id INT,

Description VARCHAR(100),

Quantity\_ordered INT,

Store\_id INT,

City\_name VARCHAR(50)

);

use data\_warehouse ;

INSERT INTO order\_details\_customer (Order\_no, Order\_date, Item\_id, Description, Quantity\_ordered, Store\_id, City\_name)

SELECT

O.Order\_no,

O.Order\_date,

I.Item\_id,

I.Description,

OI.Quantity\_ordered,

S.Store\_id,

HQ.City\_name

FROM

sales.Order1 O

JOIN

sales.Ordered\_item OI ON O.Order\_no = OI.Order\_no

JOIN

sales.Items I ON OI.Item\_id = I.Item\_id

JOIN

sales.Stored\_items SI ON OI.Item\_id = SI.Item\_id

JOIN

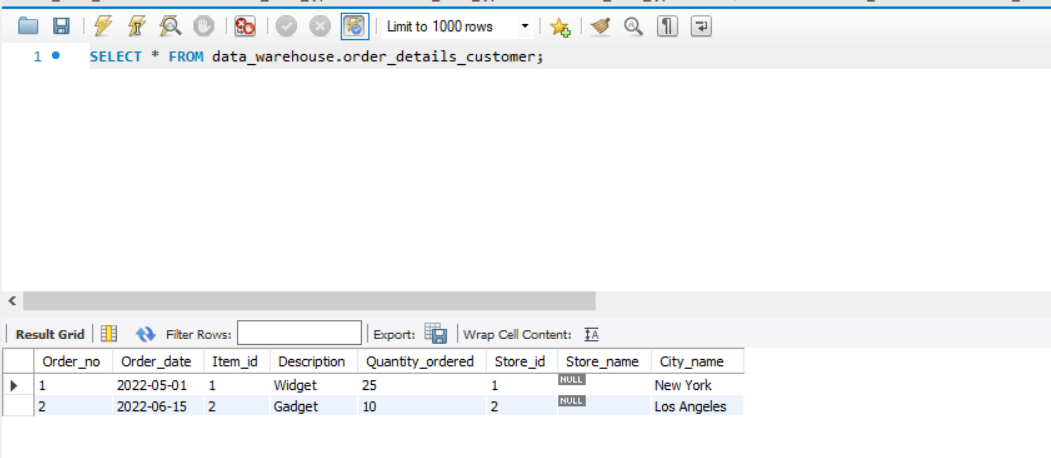
sales.Stores S ON SI.Store\_id = S.Store\_id

JOIN

sales.headqarters HQ ON S.City\_id = HQ.City\_id

ORDER BY

O.Order\_no ASC;



6.Find the city and the state in which a given customer lives.

use data\_warehouse ;

CREATE TABLE data\_warehouse.customer\_location (

customer\_id INT PRIMARY KEY,

customer\_name VARCHAR(50) NOT NULL,

city VARCHAR(50) NOT NULL,

state VARCHAR(50) NOT NULL

);

INSERT INTO data\_warehouse.customer\_location (customer\_id, customer\_name, city, state)

SELECT

C.Customer\_id,

C.Customer\_name,

H.City\_name,

H.State

FROM

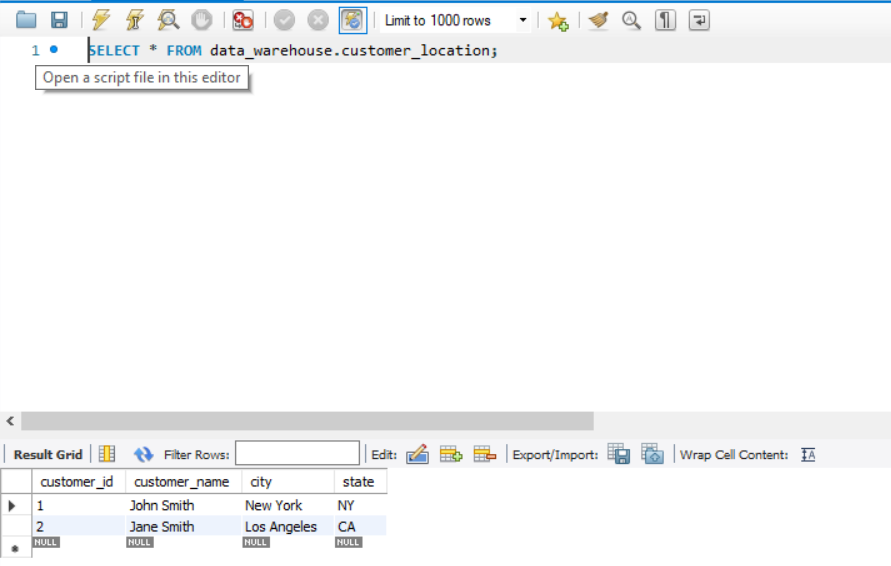
headquarter.Customer C

JOIN

sales.headqarters H

ON

C.City\_id = H.City\_id;



7.Find the stock level of a particular item in all stores in a particular city.

use data\_warehouse ;

CREATE TABLE data\_warehouse.item\_stock (

city\_name VARCHAR(50) NOT NULL,

store\_id INT NOT NULL,

item\_id INT NOT NULL,

item\_description VARCHAR(100) NOT NULL,

stock\_level INT NOT NULL,

PRIMARY KEY (city\_name, store\_id, item\_id)

);

INSERT INTO data\_warehouse.item\_stock (city\_name, store\_id, item\_id, item\_description, stock\_level)

SELECT

H.City\_name,

S.Store\_id,

I.Item\_id,

I.Description,

SI.Quantity\_held

FROM

sales.Headqarters H

JOIN sales.Stores S ON H.City\_id = S.City\_id

JOIN sales.Stored\_items SI ON S.Store\_id = SI.Store\_id

JOIN sales.Items I ON SI.Item\_id = I.Item\_id

WHERE

H.City\_name = 'New York'

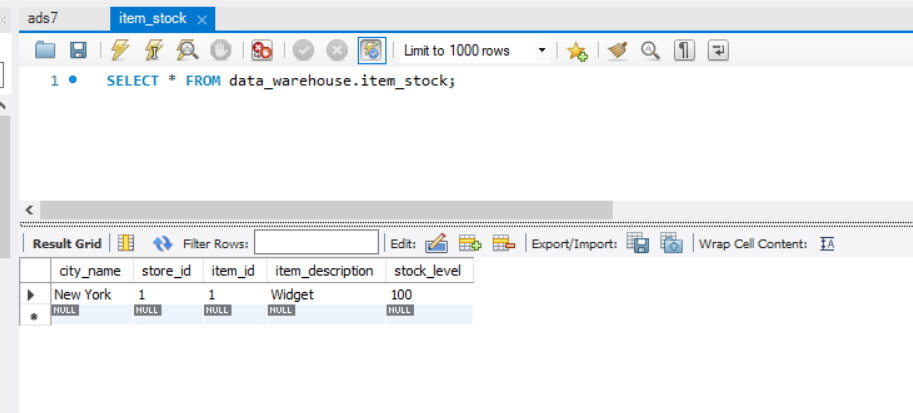
AND I.Item\_id = 1

ORDER BY

H.City\_name,

S.Store\_id,

I.Item\_id;



8.Find the items, quantity ordered, customer, store and city of an order.

CREATE TABLE data\_warehouse.order\_details (

order\_no INT PRIMARY KEY,

item\_id INT REFERENCES sales.Items(Item\_id),

quantity\_ordered INT,

customer\_id INT REFERENCES headquarter.Customer(Customer\_id),

store\_id INT REFERENCES sales.Stores(Store\_id),

city VARCHAR(50) NOT NULL

);

INSERT INTO data\_warehouse.order\_details (order\_no, item\_id, quantity\_ordered, customer\_id, store\_id, city)

SELECT

O.Order\_no,

OI.Item\_id,

OI.Quantity\_ordered,

C.Customer\_id,

S.Store\_id,

H.City\_name

FROM

sales.Order1 O

JOIN

sales.Ordered\_item OI ON O.Order\_no = OI.Order\_no

JOIN

sales.Stores S ON O.Customer\_id = S.City\_id

JOIN

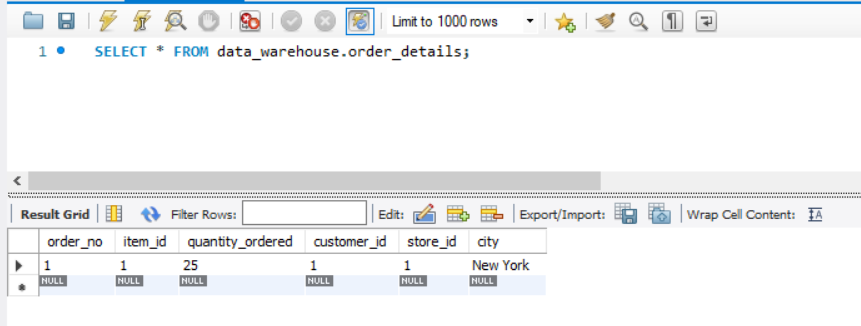
sales.Headqarters H ON S.City\_id = H.City\_id

JOIN

headquarter.Customer C ON O.Customer\_id = C.Customer\_id

WHERE

O.Order\_no = 1;



9.Find the walk in customers, mail order customers and dual customers (both walk-in and mail order).

CREATE TABLE data\_warehouse.customer\_order\_type (

customer\_id INT PRIMARY KEY,

customer\_name VARCHAR(50) NOT NULL,

order\_type VARCHAR(20) NOT NULL

);

INSERT INTO data\_warehouse.customer\_order\_type(Customer\_id, Customer\_name, order\_type)

SELECT

C.Customer\_id,

C.Customer\_name,

CASE

WHEN W.Customer\_id IS NOT NULL AND M.Customer\_id IS NULL THEN 'Walk-in'

WHEN W.Customer\_id IS NULL AND M.Customer\_id IS NOT NULL THEN 'Mail order'

WHEN W.Customer\_id IS NOT NULL AND M.Customer\_id IS NOT NULL THEN 'Dual'

ELSE 'Unknown'

END AS Customer\_type

FROM

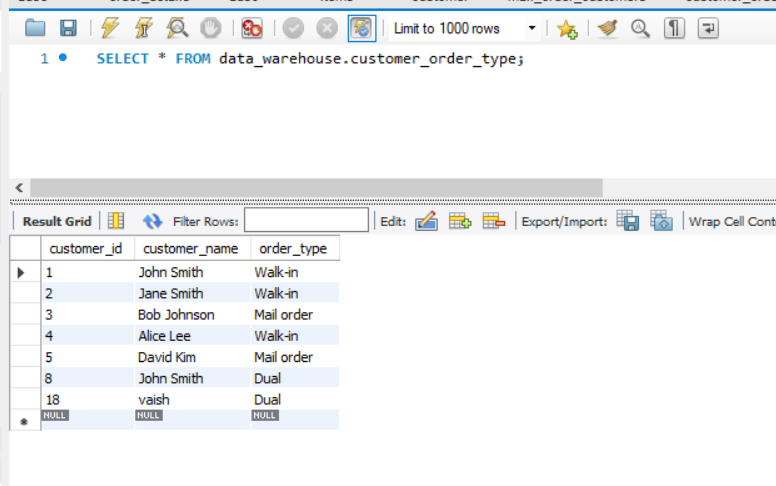
headquarter.Customer C

LEFT JOIN

headquarter.Walk\_in\_customers W ON C.Customer\_id = W.Customer\_id

LEFT JOIN

headquarter.Mail\_order\_customers M ON C.Customer\_id = M.Customer\_id



Introduction

The objective of this project is to design and build a data warehouse/OLAP system that can answer various queries related to an enterprise that consists of multiple stores located in different cities and states. The data warehouse will store data about stores, items, orders, customers, and their locations. The system will be able to provide analytical reports to support business decision making.

Business Requirement

The enterprise needs a data warehousing system that can provide answers to queries related to their operations. The system must be able to find stores that hold a particular item, orders that can be fulfilled by a given store, customers who ordered specific items, and the stock level of a particular item in all stores of a city. The system should also be able to identify walk-in and mail-order customers, and generate reports on item sales.

Functional Specification

The data warehouse system will have the following inputs:

Data from operational databases such as customer information, item information, order information, store information, and their locations.

The system will provide the following outputs:

Reports that answer various queries related to the enterprise's operations such as the locations of stores that hold specific items, orders that can be fulfilled by a given store, customers who ordered specific items, stock levels of specific items, and item sales reports.

Data Warehousing Design

The data warehousing system will use the star schema design with the following dimensions:

Time dimension: Includes time-related attributes such as date, week, month, and year.

Store dimension: Includes attributes such as store ID, store name, phone number, city, and state.

Item dimension: Includes attributes such as item ID, description, size, weight, and unit price.

Customer dimension: Includes attributes such as customer ID, customer name, city, and state.

Order dimension: Includes attributes such as order number, order date, and customer ID.

The fact table will be the Ordered\_Item table, which will include attributes such as item ID, order number, store ID, city, quantity ordered, and ordered price.

Implementation:

The data warehousing system has been implemented using MySQL as the backend database. The system includes tables such as Customer, Walk-in\_customers, Mail\_order\_customers, Headqarters, Stores, Items, Stored\_items, Order, and Ordered\_item.

To load the data from the operational databases, an ETL process was used to extract, transform, and load the data into the MySQL data warehouse. The data is stored in a relational database schema, and not in a multidimensional cube.

The system provides a user interface to generate reports based on the selected dimensions and measures. The reports are displayed in tables or charts, and the system supports online analytical processing (OLAP) reports.

To ensure the accuracy of the reports, the system has mechanisms to verify the data against the operational databases' data. This helps to ensure that the reports are based on accurate and up-to-date data.

Overall, the data warehousing implementation using MySQL provides a robust and efficient solution for generating OLAP reports based on the selected dimensions and measures.

Observations:

a. Report Generation - The system will provide a user interface to generate reports based on data from the MySQL data warehouse. The reports will be generated based on the selected dimensions and measures, and they will be displayed in tables or charts.

b. Data Verification - The system will have mechanisms to verify the data in the MySQL data warehouse. The data will be compared against the operational databases' data to ensure the accuracy of the reports.

Conclusion

In conclusion, the data warehousing system will enable the enterprise to analyze their operations and make informed business decisions. The system will provide reports that answer queries related to the enterprise's operations, such as the locations of stores that hold specific items, orders that can be fulfilled by a given store, customers who ordered specific items, stock levels of specific items, and item sales reports. The system will use a star schema design and a multidimensional cube to store and process the data.