TY B.Tech. (CSE) – II [2022-23]

5CS372: Advanced Database System Lab.

Assignment No. 6

Name : Tanaya Mukund Bhide

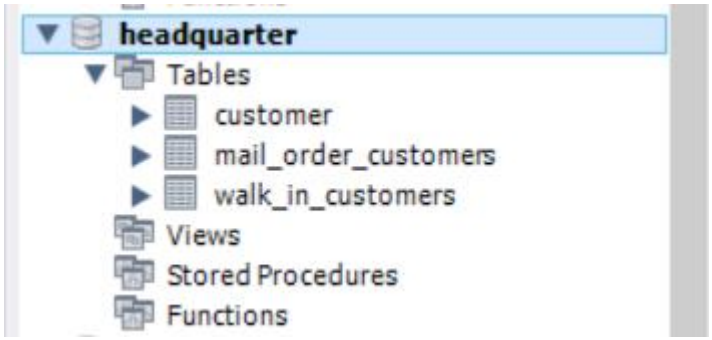
PRN: 2020BTECS00011

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
);
```

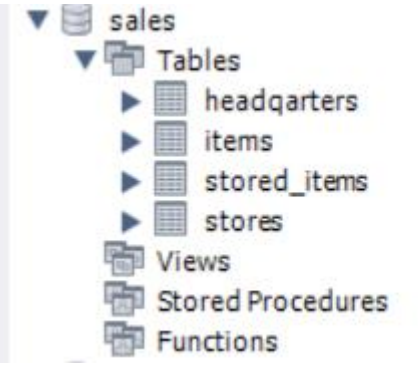


sales

```
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');
```

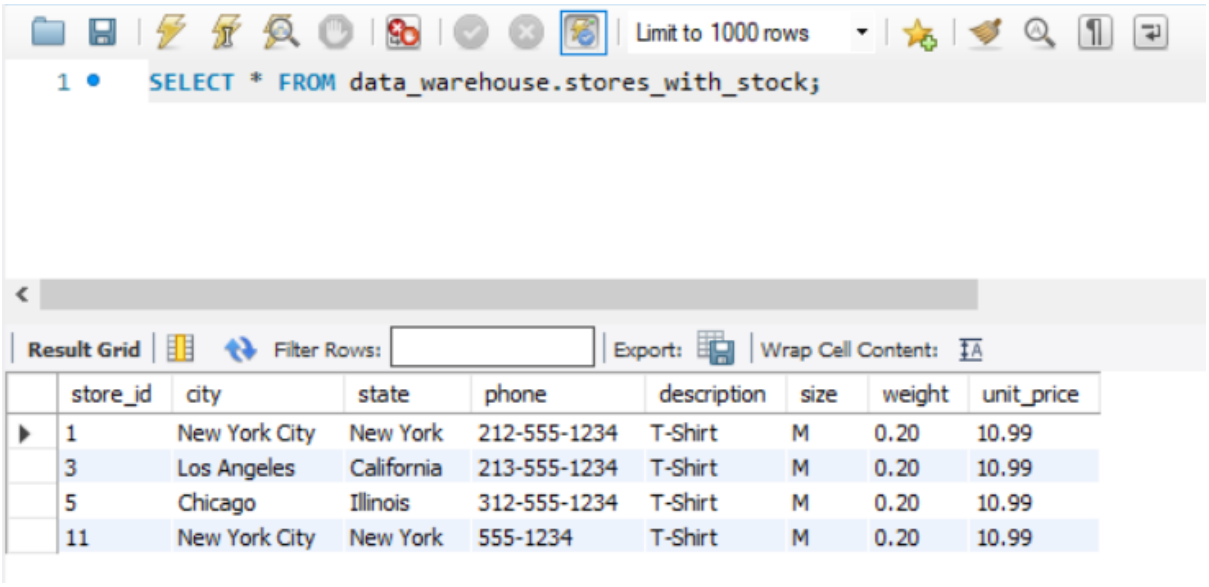
```
use sales ;

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

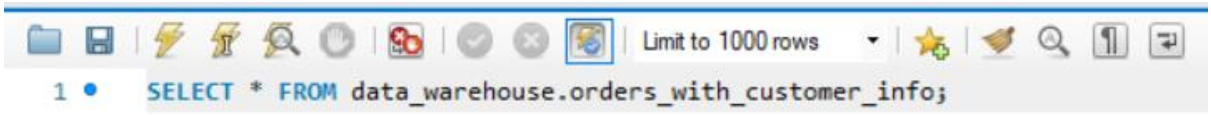
```
use data_warehouse;

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';
```



store_id	city	state	phone	description	size	weight	unit_price
1	New York City	New York	212-555-1234	T-Shirt	M	0.20	10.99
3	Los Angeles	California	213-555-1234	T-Shirt	M	0.20	10.99
5	Chicago	Illinois	312-555-1234	T-Shirt	M	0.20	10.99
11	New York City	New York	555-1234	T-Shirt	M	0.20	10.99

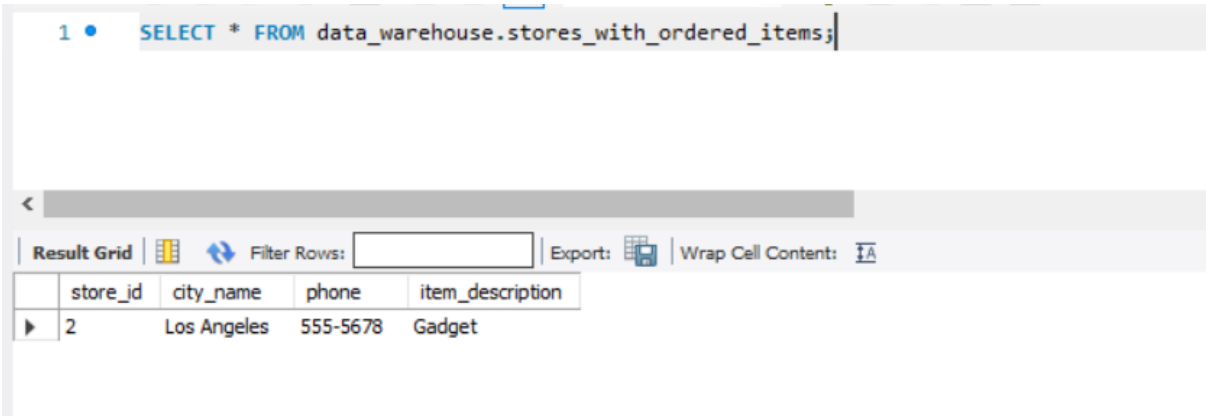
2. Find all the orders along with customer name and order date that can be fulfilled by a given store.

```
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;
```



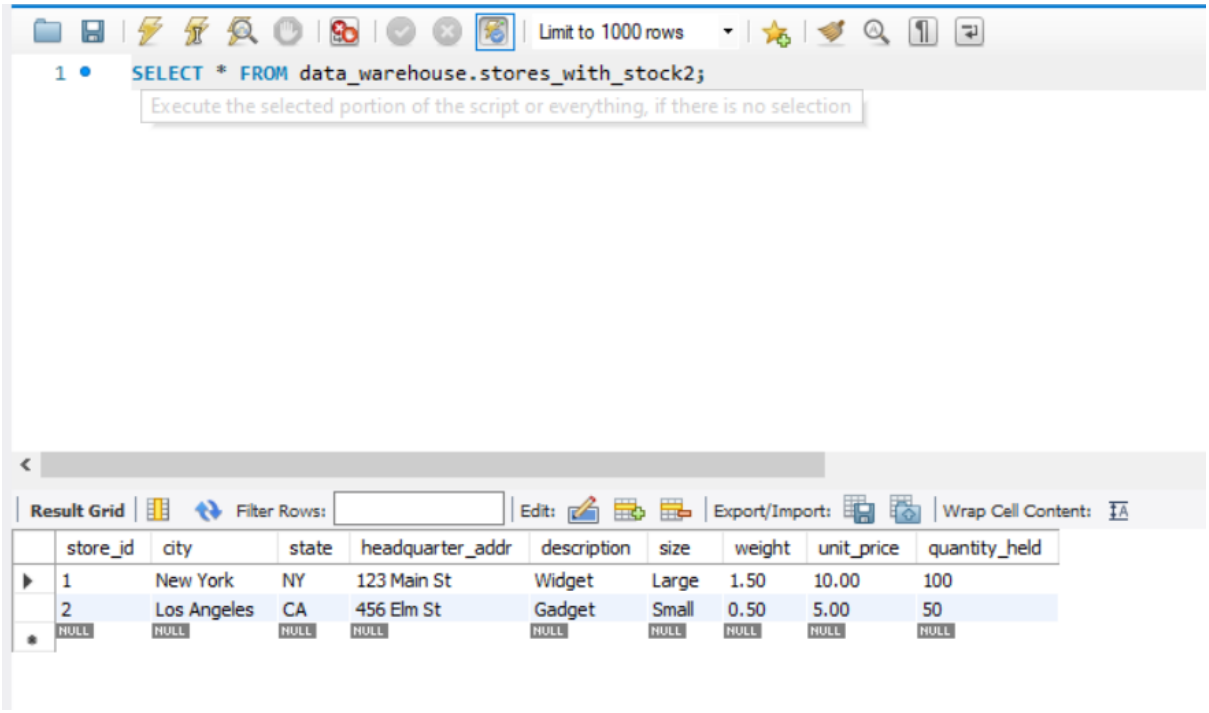
3. Find all stores along with city name and phone that hold items ordered by given customer

```
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.

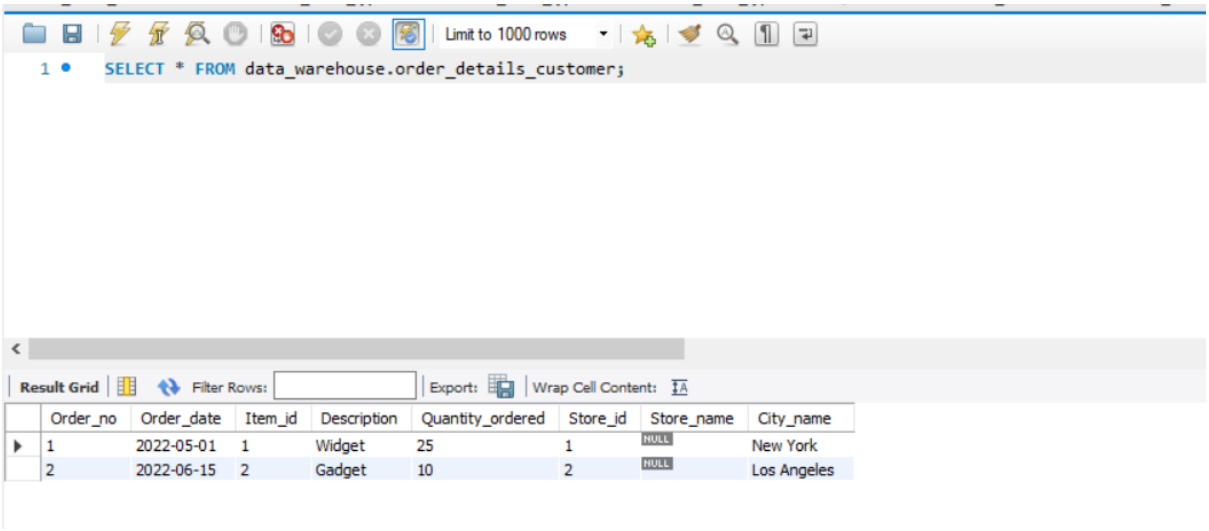
```
use data_warehouse ;


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;
```


Limit to 1000 rows
1
SELECT * FROM data_warehouse.stores_with_stock2;
Execute the selected portion of the script or everything, if there is no selection
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
store_id city state headquarter_addr description size weight unit_price quantity_held
1 New York NY 123 Main St Widget Large 1.50 10.00 100
2 Los Angeles CA 456 Elm St Gadget Small 0.50 5.00 50
NULL NULL NULL NULL NULL NULL NULL NULL NULL

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

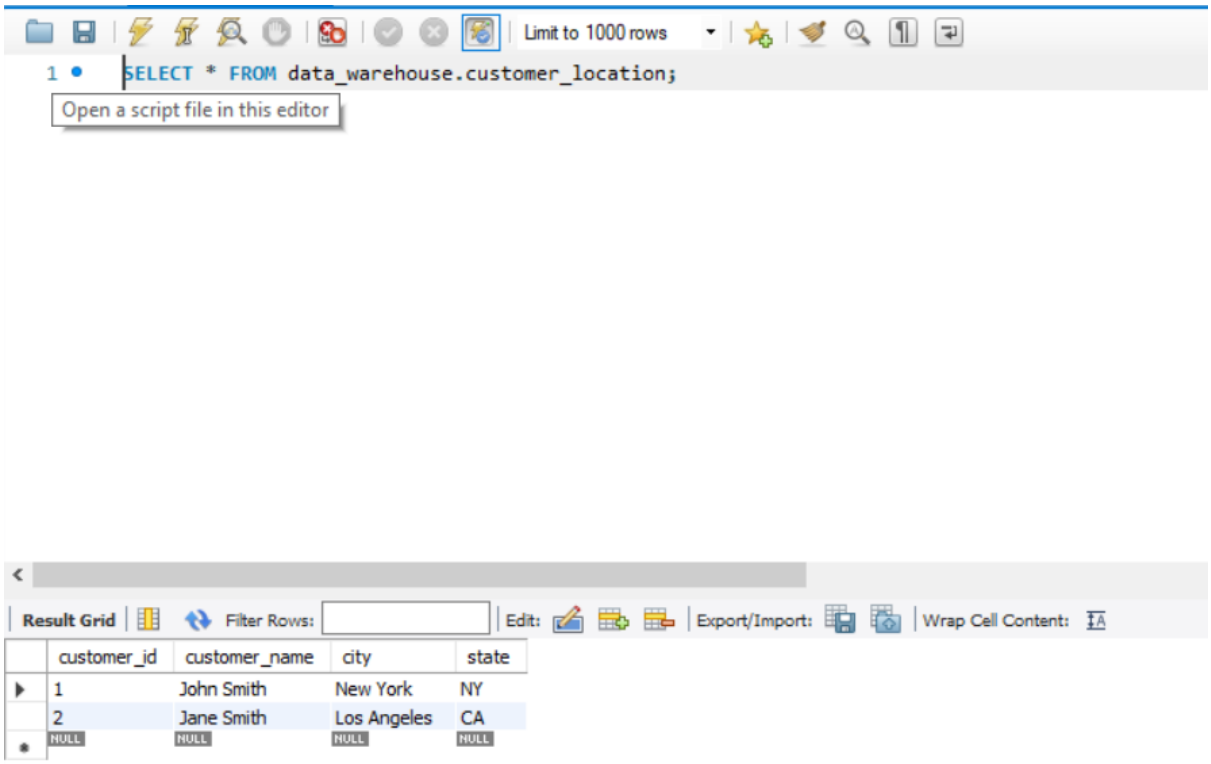
```
CREATE TABLE data_warehouse.order_details_customer (
    Order_no INT,
    Order_date DATE,
    Item_id INT,
    Description VARCHAR(100),
    Quantity_ordered INT,
    Store_id INT,
    City_name VARCHAR(50)
);
use data_warehouse ;
INSERT INTO order_details_customer (Order_no, Order_date, Item_id, 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.

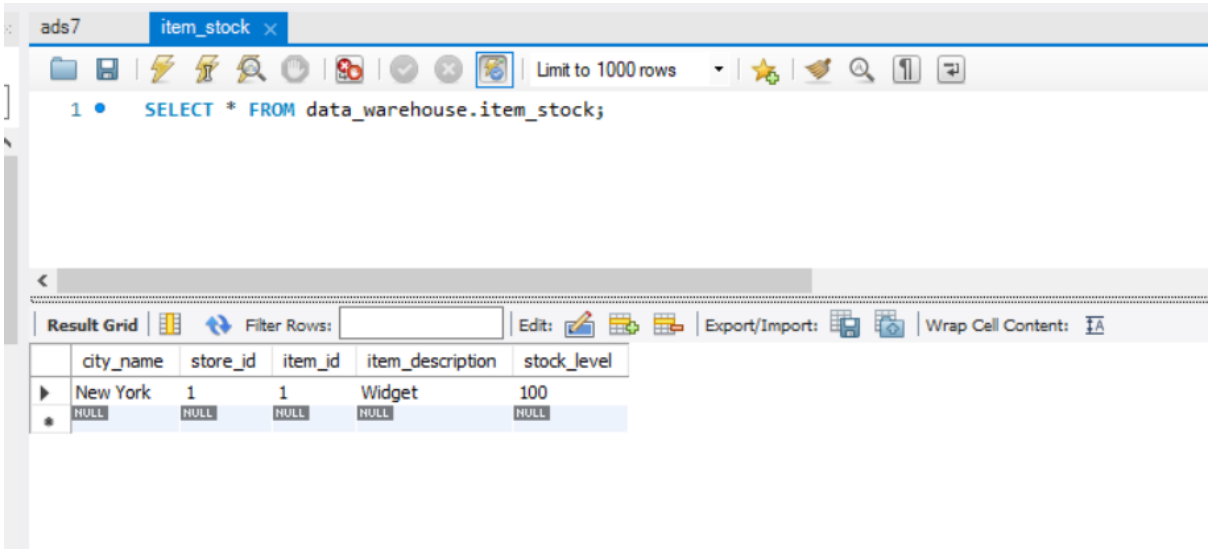
```
use data_warehouse ;

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;
```


Limit to 1000 rows
1
SELECT * FROM data_warehouse.customer_location;
Open a script file in this editor
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
customer_id
customer_name
city
state
1
John Smith
New York
NY
2
Jane Smith
Los Angeles
CA
NULL
NULL
NULL
NULL

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
```

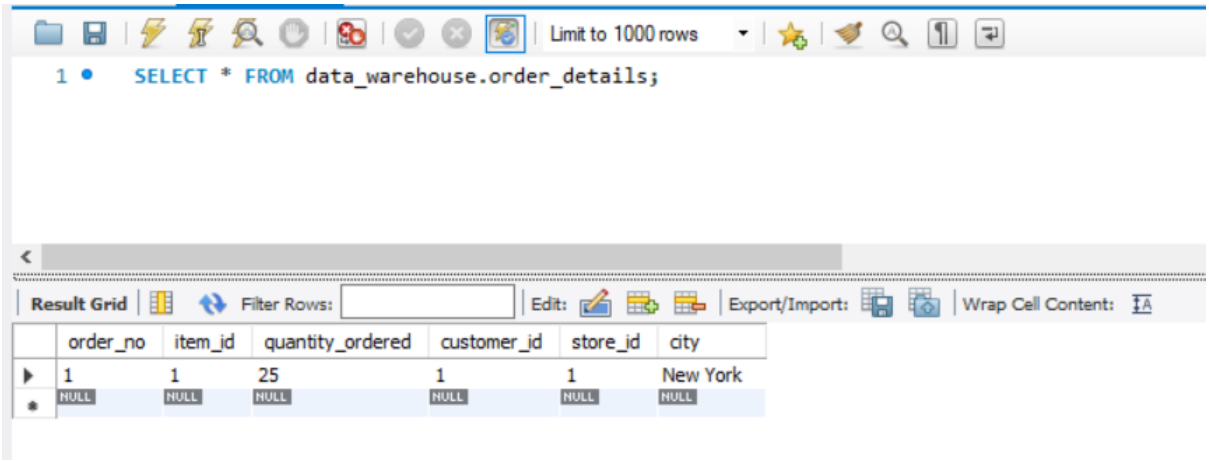
```
ORDER BY
    H.City_name,
    S.Store_id,
    I.Item_id;
```



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

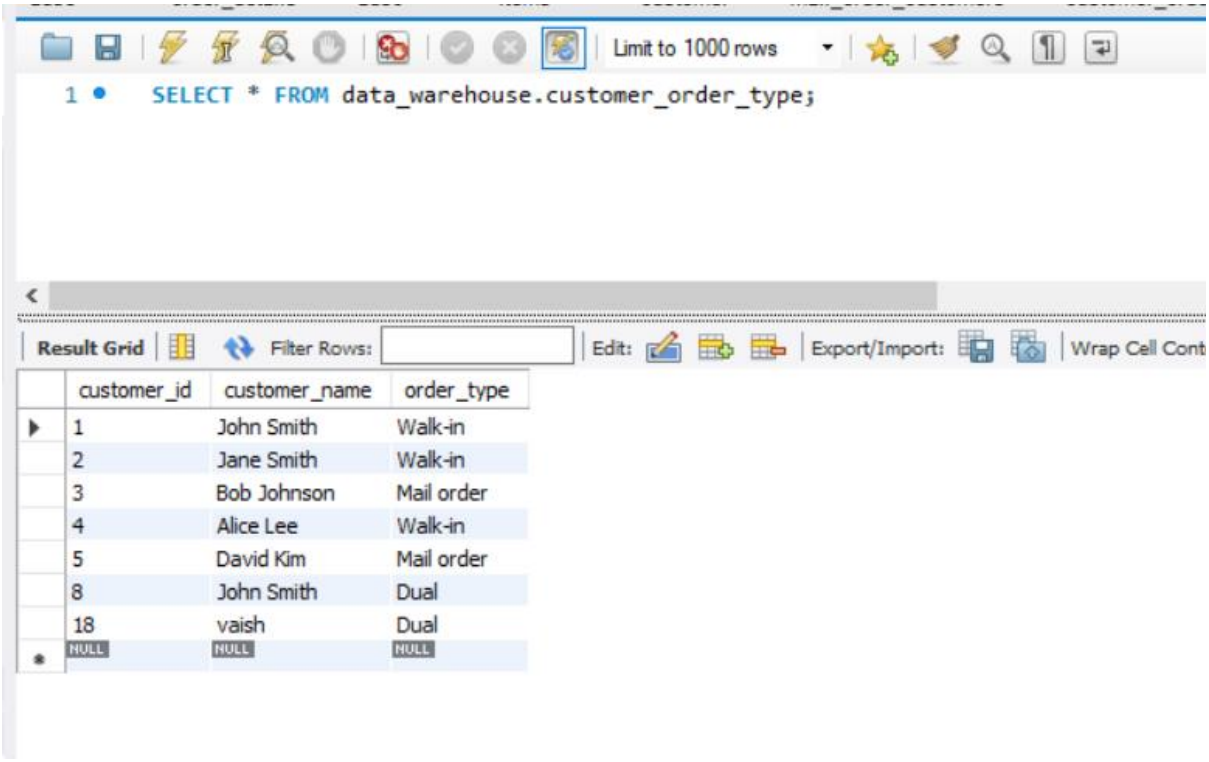
```
CREATE TABLE data_warehouse.order_details (
    order_no INT PRIMARY KEY,
    item_id INT REFERENCES sales.Items(Item_id),
    quantity_ordered INT,
    customer_id INT REFERENCES headquarter.Customer(Customer_id),
    store_id INT REFERENCES sales.Stores(Store_id),
    city VARCHAR(50) NOT NULL
);

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
```



customer_id	customer_name	order_type
1	John Smith	Walk-in
2	Jane Smith	Walk-in
3	Bob Johnson	Mail order
4	Alice Lee	Walk-in
5	David Kim	Mail order
8	John Smith	Dual
18	vaish	Dual
NULL	NULL	NULL

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.