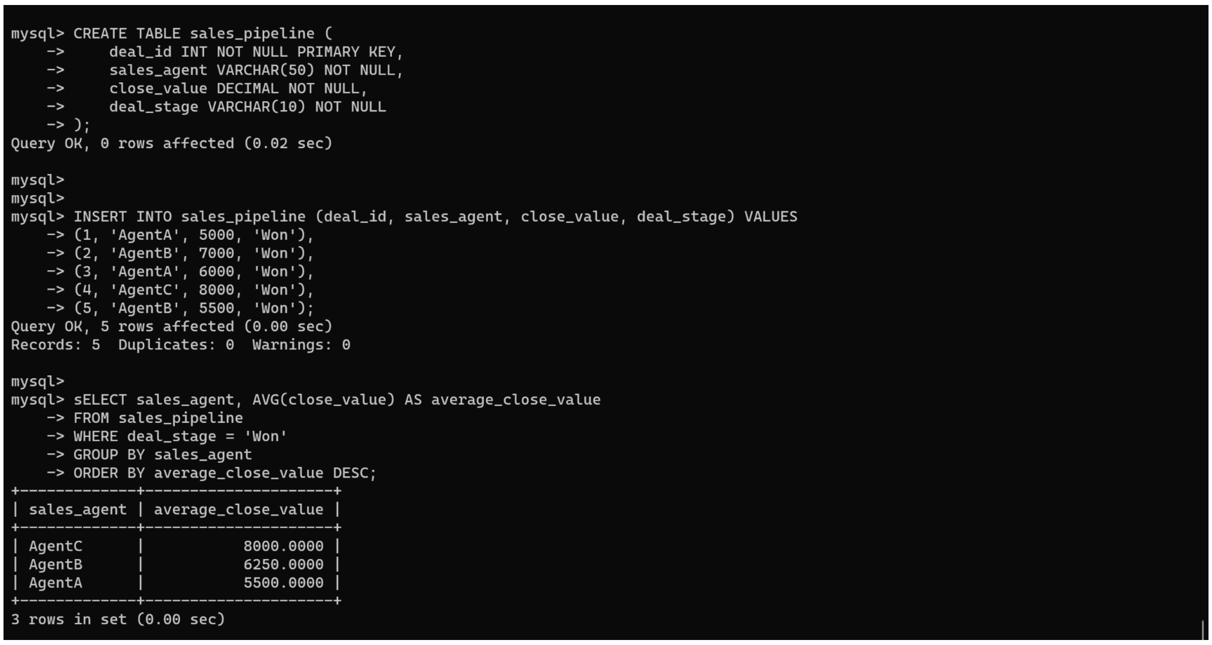
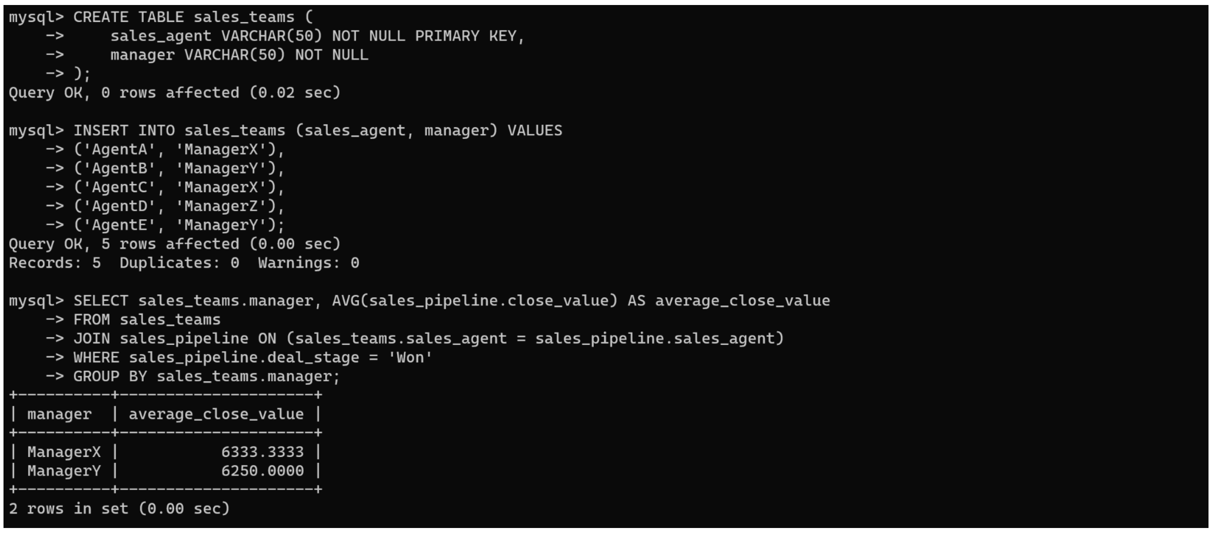
NAME : SHARATH CHANDRA
BATCH: DATA ENGINEERING
TOPIC: RULES AND RESTRICTIONS TO GROUP AND FILTER DATA IN SQL QUERIES, Order of Execution of SQL Queries,How to calculate Subtotals in SQL Queries,Differences Between UNION EXCEPT and INTERSECT Operators in SQL Server,Star Schema, Snowflake.

1)For GroupBy data in sql queries:
Example:Average Close Value By Agent Won Deals



```
mysql> CREATE TABLE sales_pipeline (
    ->     deal_id INT NOT NULL PRIMARY KEY,
    ->     sales_agent VARCHAR(50) NOT NULL,
    ->     close_value DECIMAL NOT NULL,
    ->     deal_stage VARCHAR(10) NOT NULL
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql>
mysql> INSERT INTO sales_pipeline (deal_id, sales_agent, close_value, deal_stage) VALUES
    -> (1, 'AgentA', 5000, 'Won'),
    -> (2, 'AgentB', 7000, 'Won'),
    -> (3, 'AgentA', 6000, 'Won'),
    -> (4, 'AgentC', 8000, 'Won'),
    -> (5, 'AgentB', 5500, 'Won');
Query OK, 5 rows affected (0.00 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql>
mysql> sELECT sales_agent, AVG(close_value) AS average_close_value
    -> FROM sales_pipeline
    -> WHERE deal_stage = 'Won'
    -> GROUP BY sales_agent
    -> ORDER BY average_close_value DESC;
+-------------+---------------------+
| sales_agent | average_close_value |
+-------------+---------------------+
| AgentC      |           8000.0000 |
| AgentB      |           6250.0000 |
| AgentA      |           5500.0000 |
+-------------+---------------------+
3 rows in set (0.00 sec)
```

We could even ascertain the average value of deals aggregated by manager

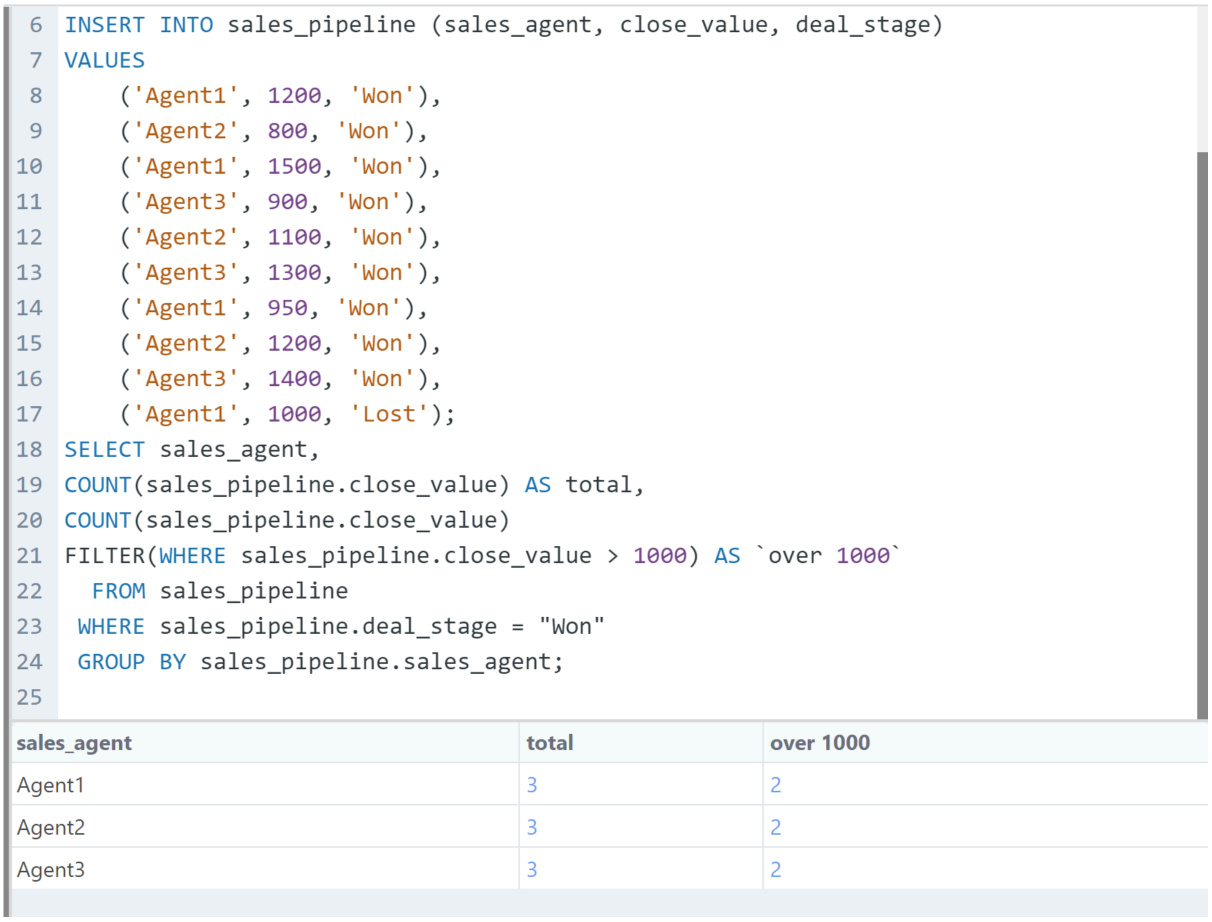


```
mysql> CREATE TABLE sales_teams (
    ->     sales_agent VARCHAR(50) NOT NULL PRIMARY KEY,
    ->     manager VARCHAR(50) NOT NULL
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO sales_teams (sales_agent, manager) VALUES
    -> ('AgentA', 'ManagerX'),
    -> ('AgentB', 'ManagerY'),
    -> ('AgentC', 'ManagerX'),
    -> ('AgentD', 'ManagerZ'),
    -> ('AgentE', 'ManagerY');
Query OK, 5 rows affected (0.00 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> SELECT sales_teams.manager, AVG(sales_pipeline.close_value) AS average_close_value
    -> FROM sales_teams
    -> JOIN sales_pipeline ON (sales_teams.sales_agent = sales_pipeline.sales_agent)
    -> WHERE sales_pipeline.deal_stage = 'Won'
    -> GROUP BY sales_teams.manager;
+----------+---------------------+
| manager  | average_close_value |
+----------+---------------------+
| ManagerX |           6333.3333 |
| ManagerY |           6250.0000 |
+----------+---------------------+
2 rows in set (0.00 sec)
```

–>For Filter data in sql queries:

example:Count Sales Over 1000

```
CREATE TABLE sales_pipeline (
    sales_agent VARCHAR(50),
    close_value DECIMAL(10, 2),
    deal_stage VARCHAR(10)
);
```

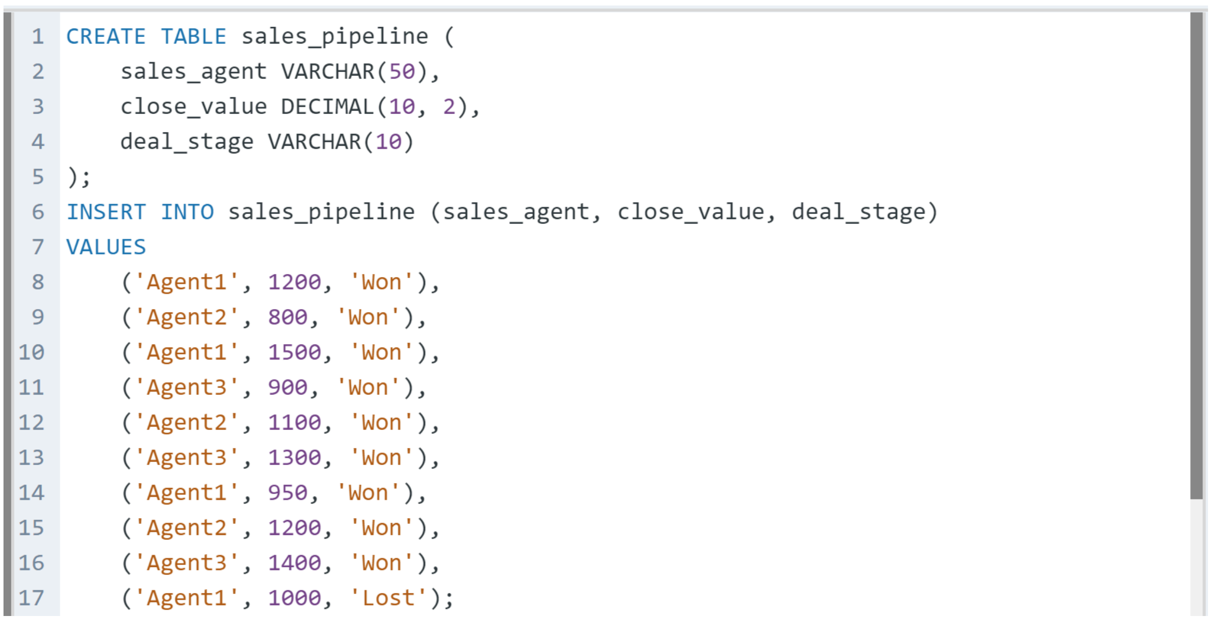


```sql
INSERT INTO sales_pipeline (sales_agent, close_value, deal_stage)
VALUES
    ('Agent1', 1200, 'Won'),
    ('Agent2', 800, 'Won'),
    ('Agent1', 1500, 'Won'),
    ('Agent3', 900, 'Won'),
    ('Agent2', 1100, 'Won'),
    ('Agent3', 1300, 'Won'),
    ('Agent1', 950, 'Won'),
    ('Agent2', 1200, 'Won'),
    ('Agent3', 1400, 'Won'),
    ('Agent1', 1000, 'Lost');
SELECT sales_agent,
COUNT(sales_pipeline.close_value) AS total,
COUNT(sales_pipeline.close_value)
FILTER(WHERE sales_pipeline.close_value > 1000) AS `over 1000`
  FROM sales_pipeline
 WHERE sales_pipeline.deal_stage = "Won"
 GROUP BY sales_pipeline.sales_agent;
```

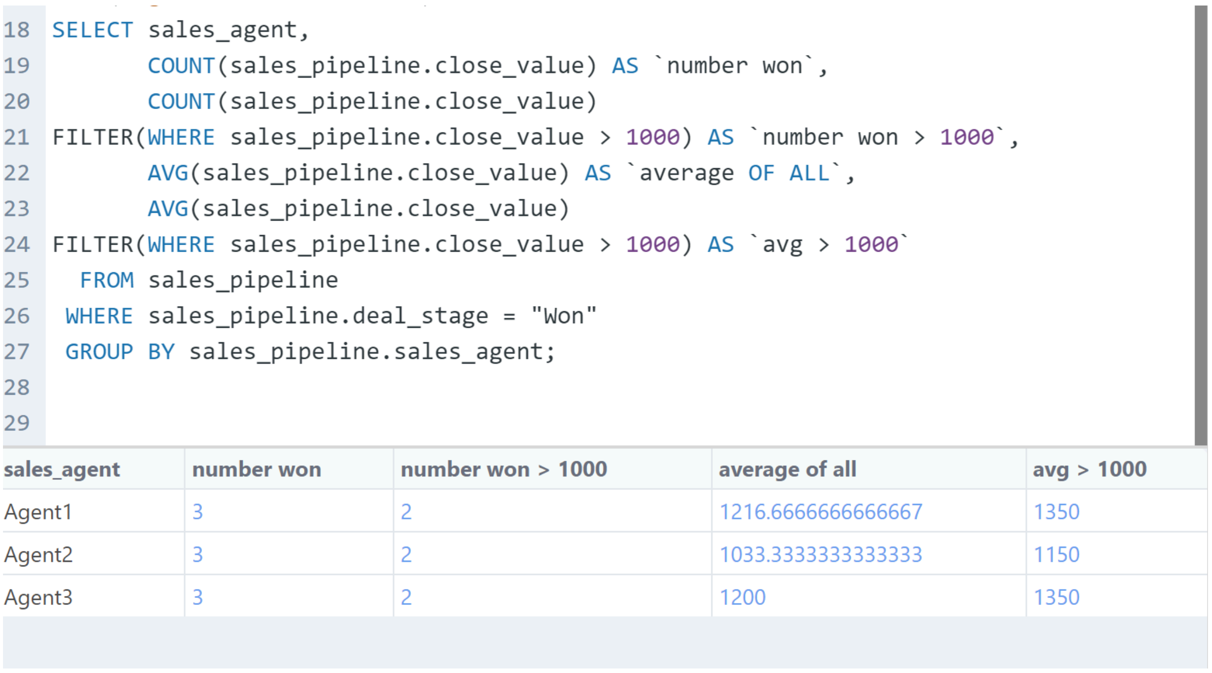
sales_agent	total	over 1000
Agent1	3	2
Agent2	3	2
Agent3	3	2

As we saw in the aggregate functions section, WHERE also limits the values in a query against which an aggregate function is run. FILTER is more flexible than WHERE because you can use more than one FILTER modifier in an aggregate query while you can only use only one WHERE clause.

Example: Sales Statistics



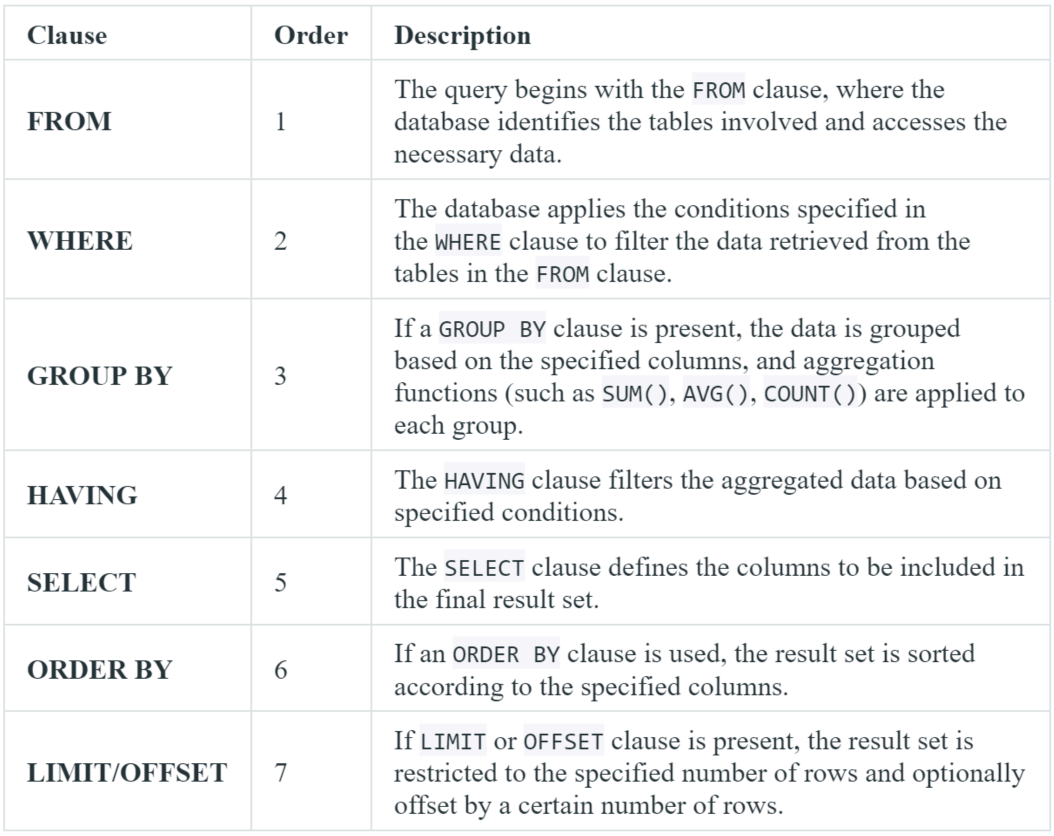
```sql
CREATE TABLE sales_pipeline (
    sales_agent VARCHAR(50),
    close_value DECIMAL(10, 2),
    deal_stage VARCHAR(10)
);
INSERT INTO sales_pipeline (sales_agent, close_value, deal_stage)
VALUES
    ('Agent1', 1200, 'Won'),
    ('Agent2', 800, 'Won'),
    ('Agent1', 1500, 'Won'),
    ('Agent3', 900, 'Won'),
    ('Agent2', 1100, 'Won'),
    ('Agent3', 1300, 'Won'),
    ('Agent1', 950, 'Won'),
    ('Agent2', 1200, 'Won'),
    ('Agent3', 1400, 'Won'),
    ('Agent1', 1000, 'Lost');
```



```sql
SELECT sales_agent,
       COUNT(sales_pipeline.close_value) AS `number won`,
       COUNT(sales_pipeline.close_value)
FILTER(WHERE sales_pipeline.close_value > 1000) AS `number won > 1000`,
       AVG(sales_pipeline.close_value) AS `average OF ALL`,
       AVG(sales_pipeline.close_value)
FILTER(WHERE sales_pipeline.close_value > 1000) AS `avg > 1000`
  FROM sales_pipeline
 WHERE sales_pipeline.deal_stage = "Won"
 GROUP BY sales_pipeline.sales_agent;
```

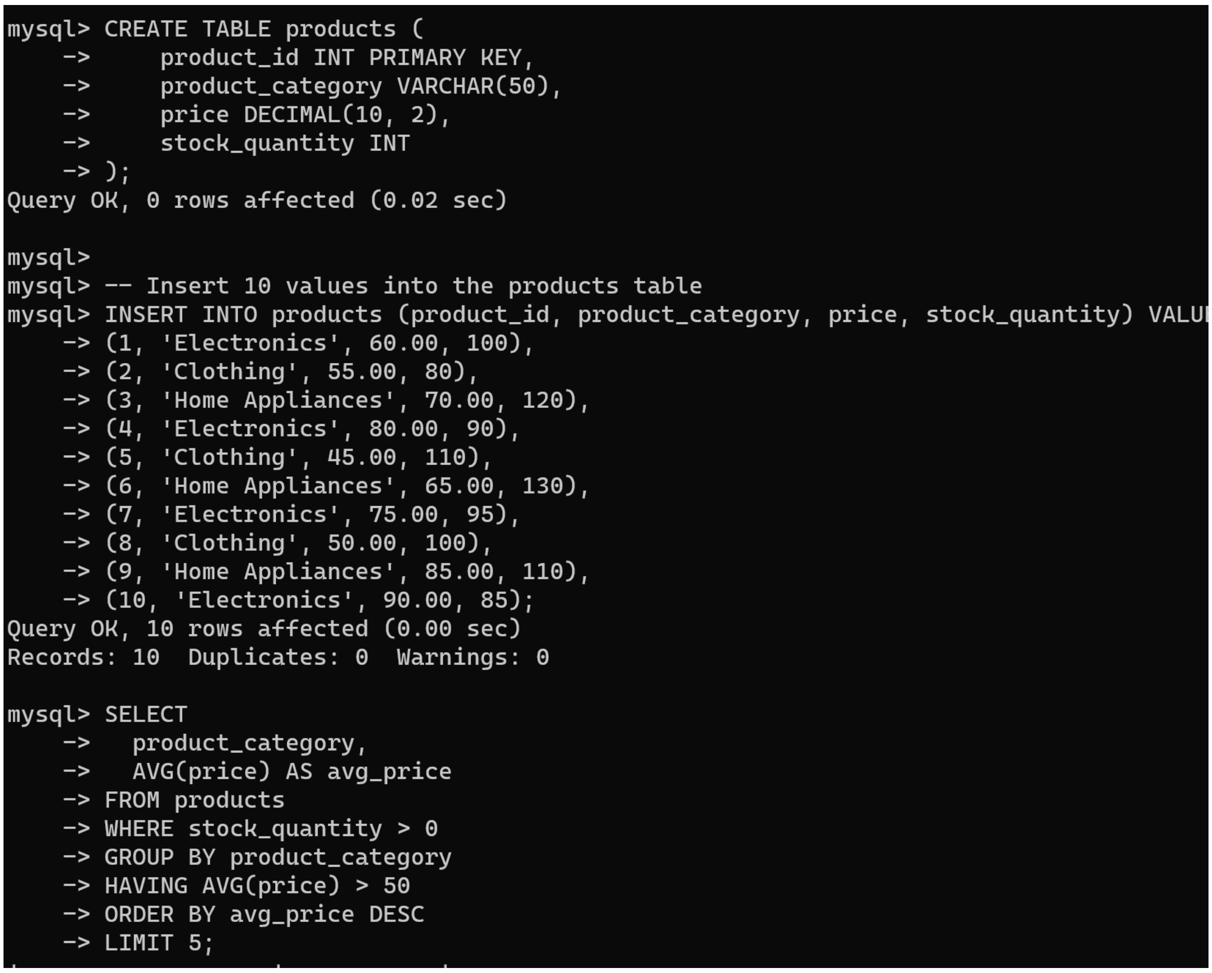
sales_agent	number won	number won > 1000	average of all	avg > 1000
Agent1	3	2	1216.6666666666667	1350
Agent2	3	2	1033.3333333333333	1150
Agent3	3	2	1200	1350

2)Order of Execution of SQL Queries:

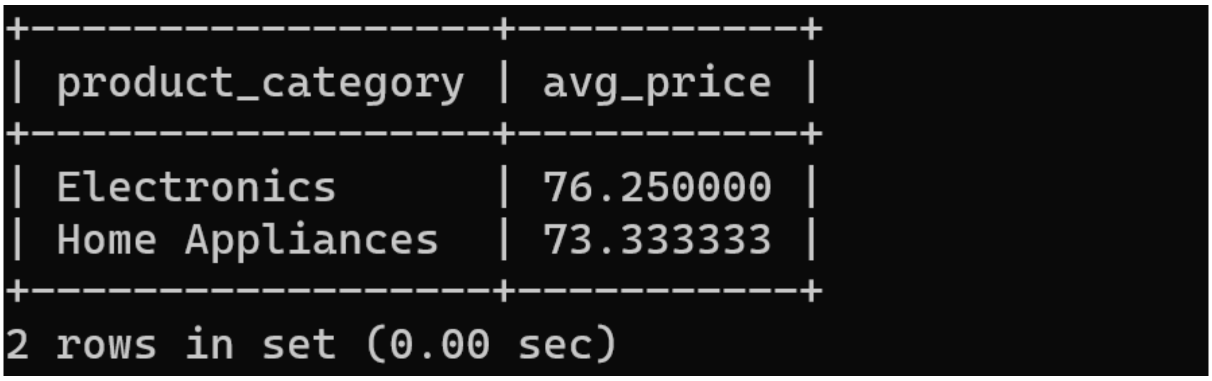


Clause	Order	Description
FROM	1	The query begins with the `FROM` clause, where the database identifies the tables involved and accesses the necessary data.
WHERE	2	The database applies the conditions specified in the `WHERE` clause to filter the data retrieved from the tables in the `FROM` clause.
GROUP BY	3	If a `GROUP BY` clause is present, the data is grouped based on the specified columns, and aggregation functions (such as `SUM()`, `AVG()`, `COUNT()`) are applied to each group.
HAVING	4	The `HAVING` clause filters the aggregated data based on specified conditions.
SELECT	5	The `SELECT` clause defines the columns to be included in the final result set.
ORDER BY	6	If an `ORDER BY` clause is used, the result set is sorted according to the specified columns.
LIMIT/OFFSET	7	If `LIMIT` or `OFFSET` clause is present, the result set is restricted to the specified number of rows and optionally offset by a certain number of rows.

Example:



```
mysql> CREATE TABLE products (
    ->     product_id INT PRIMARY KEY,
    ->     product_category VARCHAR(50),
    ->     price DECIMAL(10, 2),
    ->     stock_quantity INT
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> -- Insert 10 values into the products table
mysql> INSERT INTO products (product_id, product_category, price, stock_quantity) VALU
    -> (1, 'Electronics', 60.00, 100),
    -> (2, 'Clothing', 55.00, 80),
    -> (3, 'Home Appliances', 70.00, 120),
    -> (4, 'Electronics', 80.00, 90),
    -> (5, 'Clothing', 45.00, 110),
    -> (6, 'Home Appliances', 65.00, 130),
    -> (7, 'Electronics', 75.00, 95),
    -> (8, 'Clothing', 50.00, 100),
    -> (9, 'Home Appliances', 85.00, 110),
    -> (10, 'Electronics', 90.00, 85);
Query OK, 10 rows affected (0.00 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> SELECT
    ->   product_category,
    ->   AVG(price) AS avg_price
    -> FROM products
    -> WHERE stock_quantity > 0
    -> GROUP BY product_category
    -> HAVING AVG(price) > 50
    -> ORDER BY avg_price DESC
    -> LIMIT 5;
```



```
+------------------+-----------+
| product_category | avg_price |
+------------------+-----------+
| Electronics      | 76.250000 |
| Home Appliances  | 73.333333 |
+------------------+-----------+
2 rows in set (0.00 sec)
```

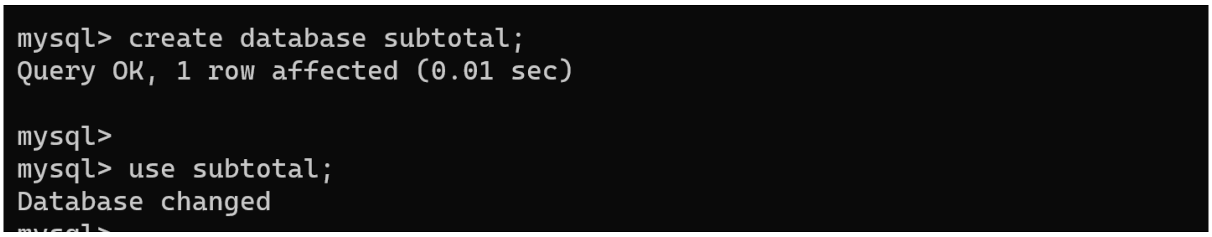
Here's how the SQL order of execution works for this query:

i)Retrieve data from the products table.

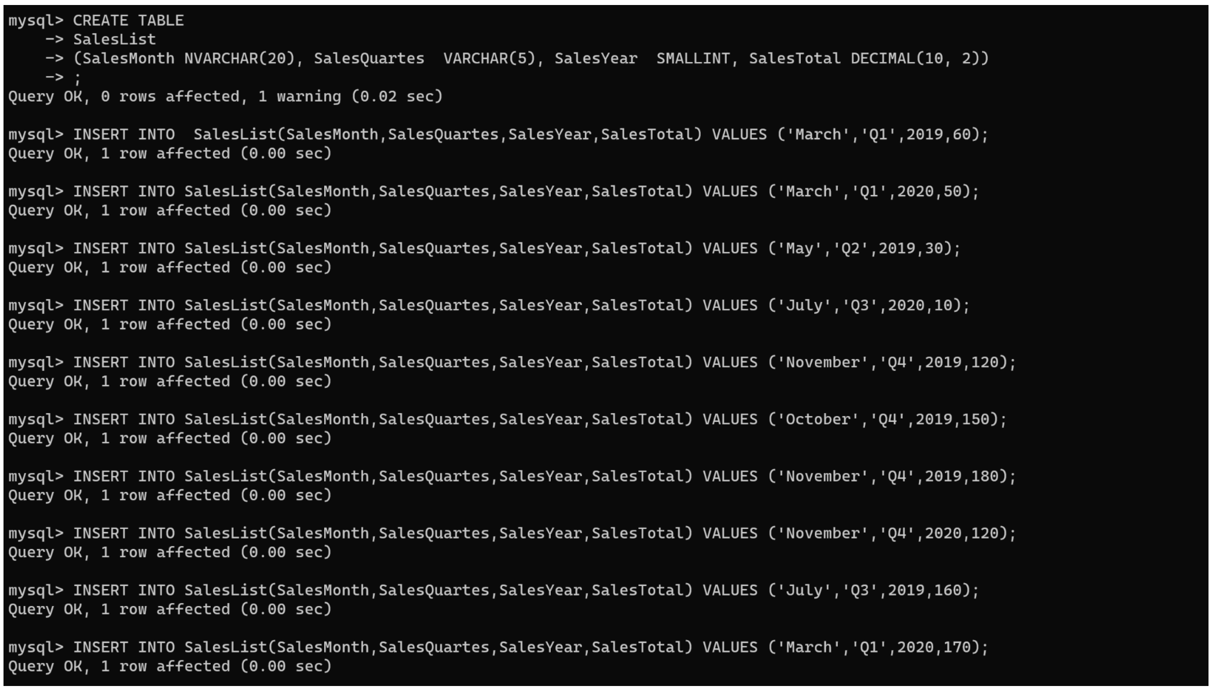
ii)Apply the filter condition in the WHERE clause to the data.

iii)Group the filtered data by the product_category column and calculate the average price for each group.
iv)Filter the grouped data using the HAVING clause condition.
v)Select the product_category column and the calculated average price for the final result set.
vi)Sort the result set based on the calculated average price in descending order.
vii)Limit the result set to a maximum of 5 rows.

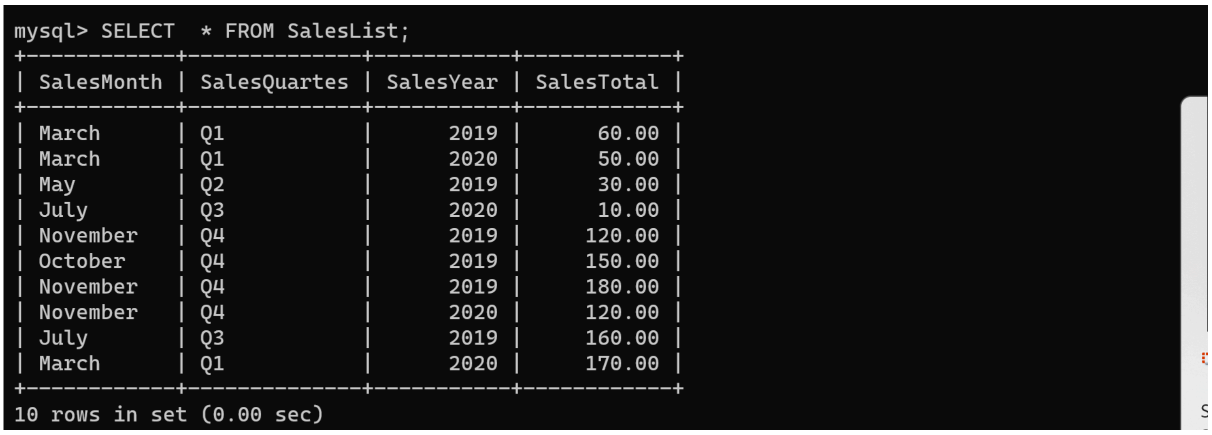
How to calculate Subtotals in SQL Queries:



```
mysql> create database subtotal;
Query OK, 1 row affected (0.01 sec)

mysql>
mysql> use subtotal;
Database changed
```

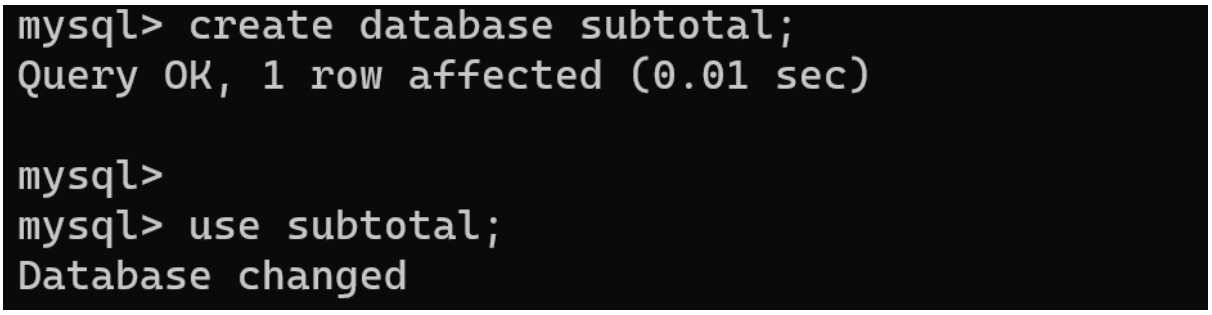


```
mysql> CREATE TABLE
    -> SalesList
    -> (SalesMonth NVARCHAR(20), SalesQuartes  VARCHAR(5), SalesYear  SMALLINT, SalesTotal DECIMAL(10, 2))
    -> ;
Query OK, 0 rows affected, 1 warning (0.02 sec)

mysql> INSERT INTO  SalesList(SalesMonth,SalesQuartes,SalesYear,SalesTotal) VALUES ('March','Q1',2019,60);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO SalesList(SalesMonth,SalesQuartes,SalesYear,SalesTotal) VALUES ('March','Q1',2020,50);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO SalesList(SalesMonth,SalesQuartes,SalesYear,SalesTotal) VALUES ('May','Q2',2019,30);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO SalesList(SalesMonth,SalesQuartes,SalesYear,SalesTotal) VALUES ('July','Q3',2020,10);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO SalesList(SalesMonth,SalesQuartes,SalesYear,SalesTotal) VALUES ('November','Q4',2019,120);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO SalesList(SalesMonth,SalesQuartes,SalesYear,SalesTotal) VALUES ('October','Q4',2019,150);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO SalesList(SalesMonth,SalesQuartes,SalesYear,SalesTotal) VALUES ('November','Q4',2019,180);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO SalesList(SalesMonth,SalesQuartes,SalesYear,SalesTotal) VALUES ('November','Q4',2020,120);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO SalesList(SalesMonth,SalesQuartes,SalesYear,SalesTotal) VALUES ('July','Q3',2019,160);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO SalesList(SalesMonth,SalesQuartes,SalesYear,SalesTotal) VALUES ('March','Q1',2020,170);
Query OK, 1 row affected (0.00 sec)
```

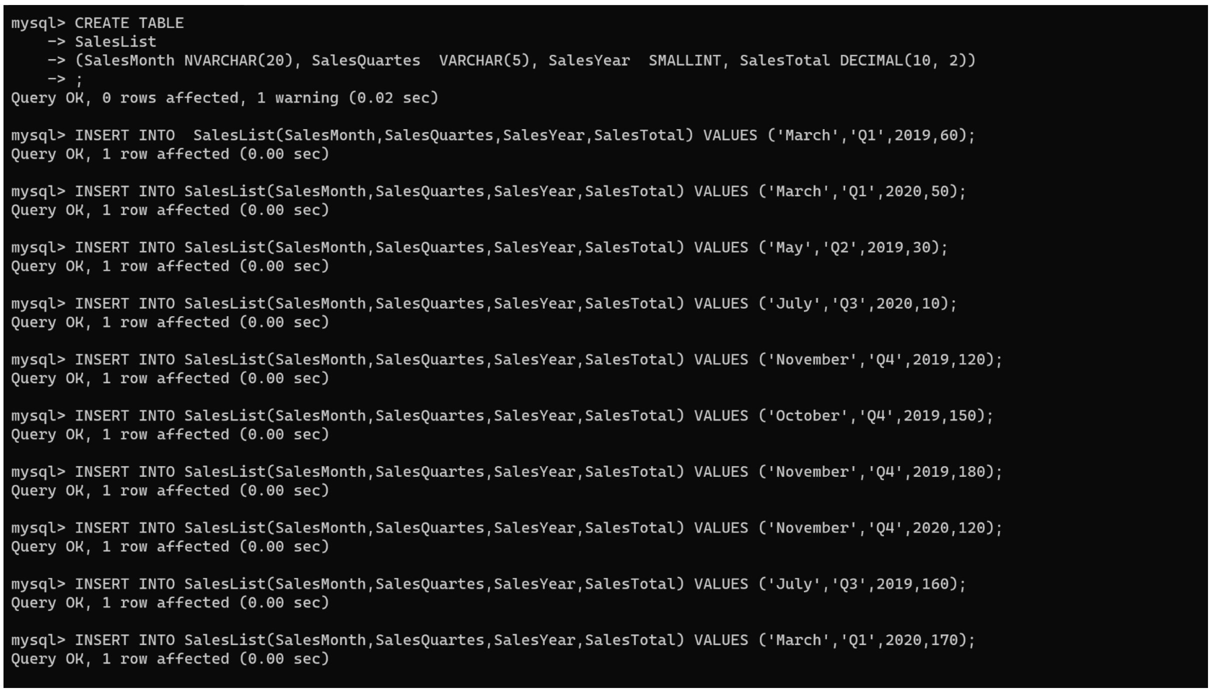


```
mysql> SELECT  * FROM SalesList;
+------------+--------------+-----------+------------+
| SalesMonth | SalesQuartes | SalesYear | SalesTotal |
+------------+--------------+-----------+------------+
| March      | Q1           |      2019 |      60.00 |
| March      | Q1           |      2020 |      50.00 |
| May        | Q2           |      2019 |      30.00 |
| July       | Q3           |      2020 |      10.00 |
| November   | Q4           |      2019 |     120.00 |
| October    | Q4           |      2019 |     150.00 |
| November   | Q4           |      2019 |     180.00 |
| November   | Q4           |      2020 |     120.00 |
| July       | Q3           |      2019 |     160.00 |
| March      | Q1           |      2020 |     170.00 |
+------------+--------------+-----------+------------+
10 rows in set (0.00 sec)
```

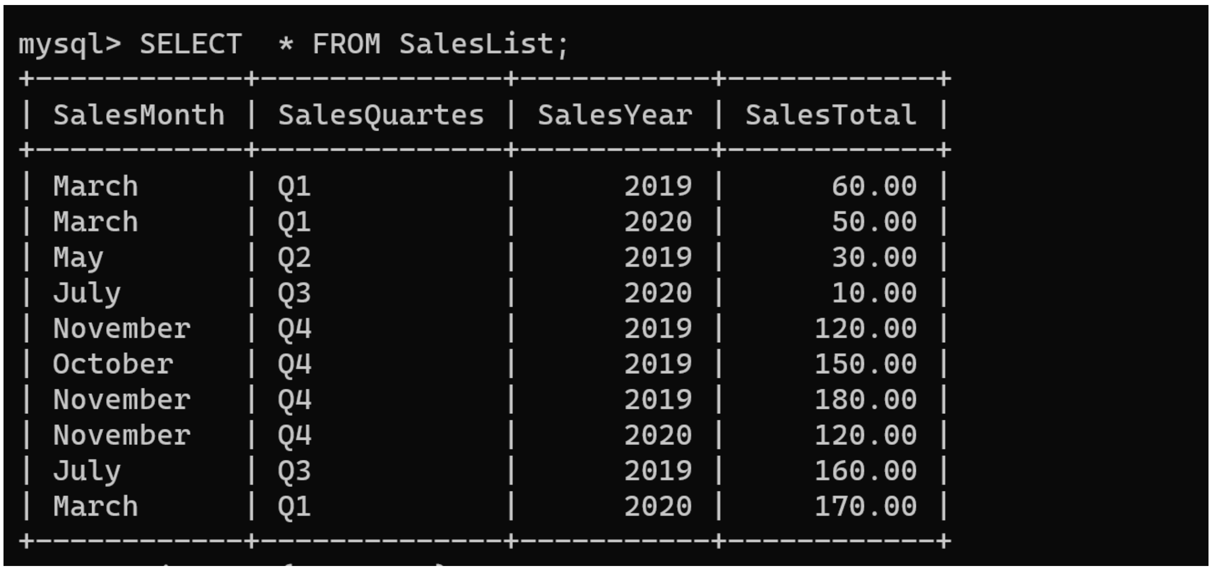
3)How to calculate Subtotals in SQL Queries:



```
mysql> create database subtotal;
Query OK, 1 row affected (0.01 sec)

mysql>
mysql> use subtotal;
Database changed
```



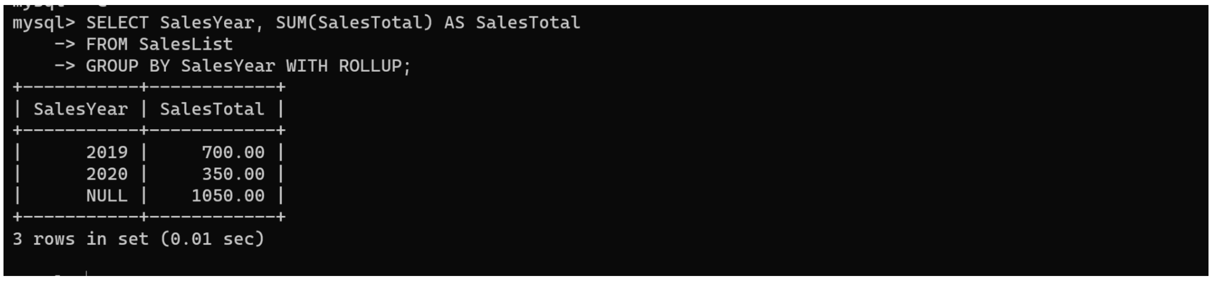
```
mysql> CREATE TABLE
    -> SalesList
    -> (SalesMonth NVARCHAR(20), SalesQuartes  VARCHAR(5), SalesYear  SMALLINT, SalesTotal DECIMAL(10, 2))
    -> ;
Query OK, 0 rows affected, 1 warning (0.02 sec)

mysql> INSERT INTO  SalesList(SalesMonth,SalesQuartes,SalesYear,SalesTotal) VALUES ('March','Q1',2019,60);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO SalesList(SalesMonth,SalesQuartes,SalesYear,SalesTotal) VALUES ('March','Q1',2020,50);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO SalesList(SalesMonth,SalesQuartes,SalesYear,SalesTotal) VALUES ('May','Q2',2019,30);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO SalesList(SalesMonth,SalesQuartes,SalesYear,SalesTotal) VALUES ('July','Q3',2020,10);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO SalesList(SalesMonth,SalesQuartes,SalesYear,SalesTotal) VALUES ('November','Q4',2019,120);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO SalesList(SalesMonth,SalesQuartes,SalesYear,SalesTotal) VALUES ('October','Q4',2019,150);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO SalesList(SalesMonth,SalesQuartes,SalesYear,SalesTotal) VALUES ('November','Q4',2019,180);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO SalesList(SalesMonth,SalesQuartes,SalesYear,SalesTotal) VALUES ('November','Q4',2020,120);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO SalesList(SalesMonth,SalesQuartes,SalesYear,SalesTotal) VALUES ('July','Q3',2019,160);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO SalesList(SalesMonth,SalesQuartes,SalesYear,SalesTotal) VALUES ('March','Q1',2020,170);
Query OK, 1 row affected (0.00 sec)
```



```
mysql> SELECT  * FROM SalesList;
+------------+--------------+-----------+------------+
| SalesMonth | SalesQuartes | SalesYear | SalesTotal |
+------------+--------------+-----------+------------+
| March      | Q1           |      2019 |      60.00 |
| March      | Q1           |      2020 |      50.00 |
| May        | Q2           |      2019 |      30.00 |
| July       | Q3           |      2020 |      10.00 |
| November   | Q4           |      2019 |     120.00 |
| October    | Q4           |      2019 |     150.00 |
| November   | Q4           |      2019 |     180.00 |
| November   | Q4           |      2020 |     120.00 |
| July       | Q3           |      2019 |     160.00 |
| March      | Q1           |      2020 |     170.00 |
+------------+--------------+-----------+------------+
```

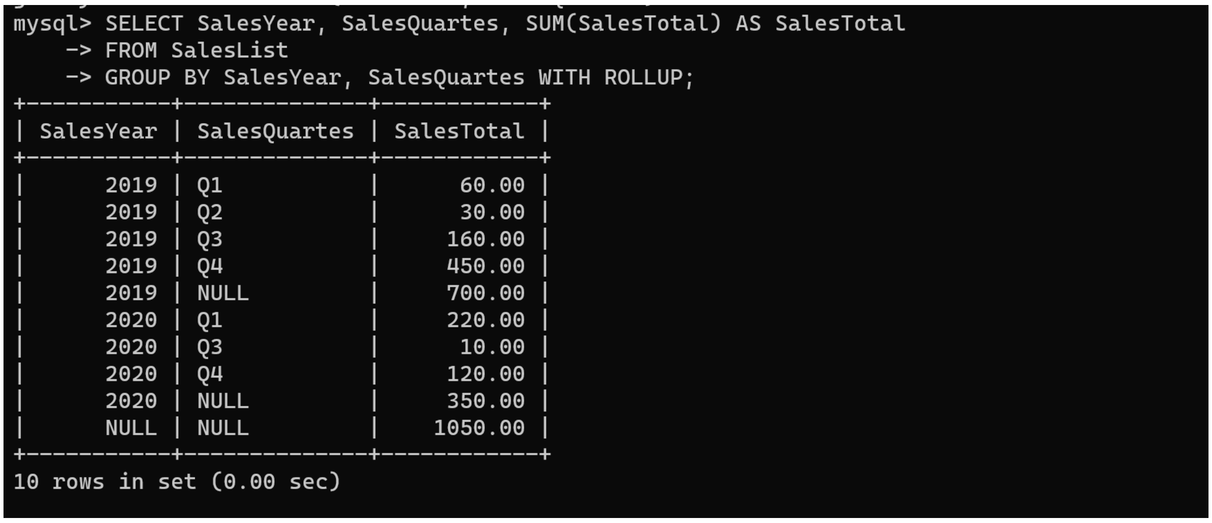
group by rollup - used to calculate subtotals:

it is giving sum of sales of each year as total sum , null is the grand total :



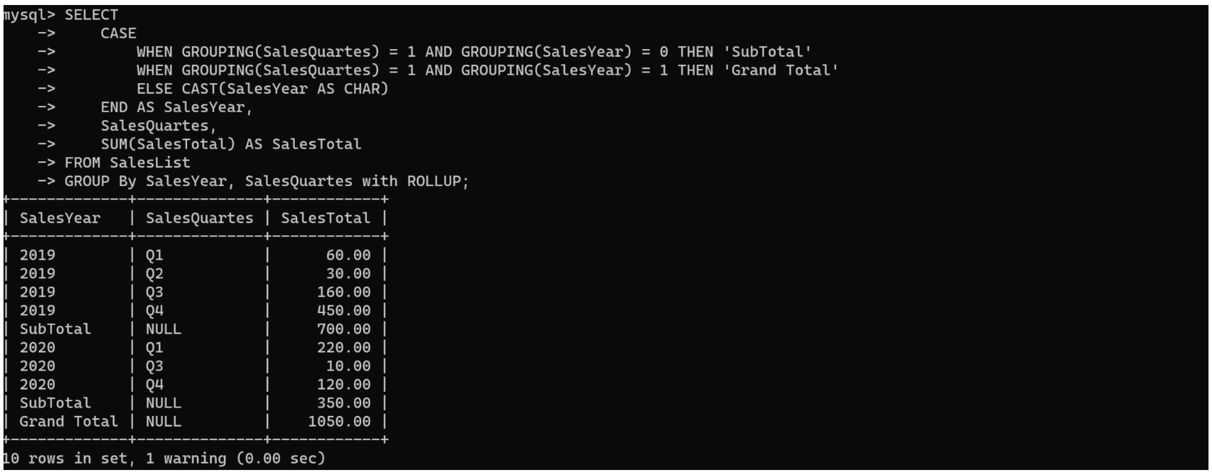
```
mysql> SELECT SalesYear, SUM(SalesTotal) AS SalesTotal
    -> FROM SalesList
    -> GROUP BY SalesYear WITH ROLLUP;
+-----------+------------+
| SalesYear | SalesTotal |
+-----------+------------+
|      2019 |     700.00 |
|      2020 |     350.00 |
|      NULL |    1050.00 |
+-----------+------------+
3 rows in set (0.01 sec)
```

null is the grand total:



```
mysql> SELECT SalesYear, SalesQuartes, SUM(SalesTotal) AS SalesTotal
    -> FROM SalesList
    -> GROUP BY SalesYear, SalesQuartes WITH ROLLUP;
+-----------+--------------+------------+
| SalesYear | SalesQuartes | SalesTotal |
+-----------+--------------+------------+
|      2019 | Q1           |      60.00 |
|      2019 | Q2           |      30.00 |
|      2019 | Q3           |     160.00 |
|      2019 | Q4           |     450.00 |
|      2019 | NULL         |     700.00 |
|      2020 | Q1           |     220.00 |
|      2020 | Q3           |      10.00 |
|      2020 | Q4           |     120.00 |
|      2020 | NULL         |     350.00 |
|      NULL | NULL         |    1050.00 |
+-----------+--------------+------------+
10 rows in set (0.00 sec)
```

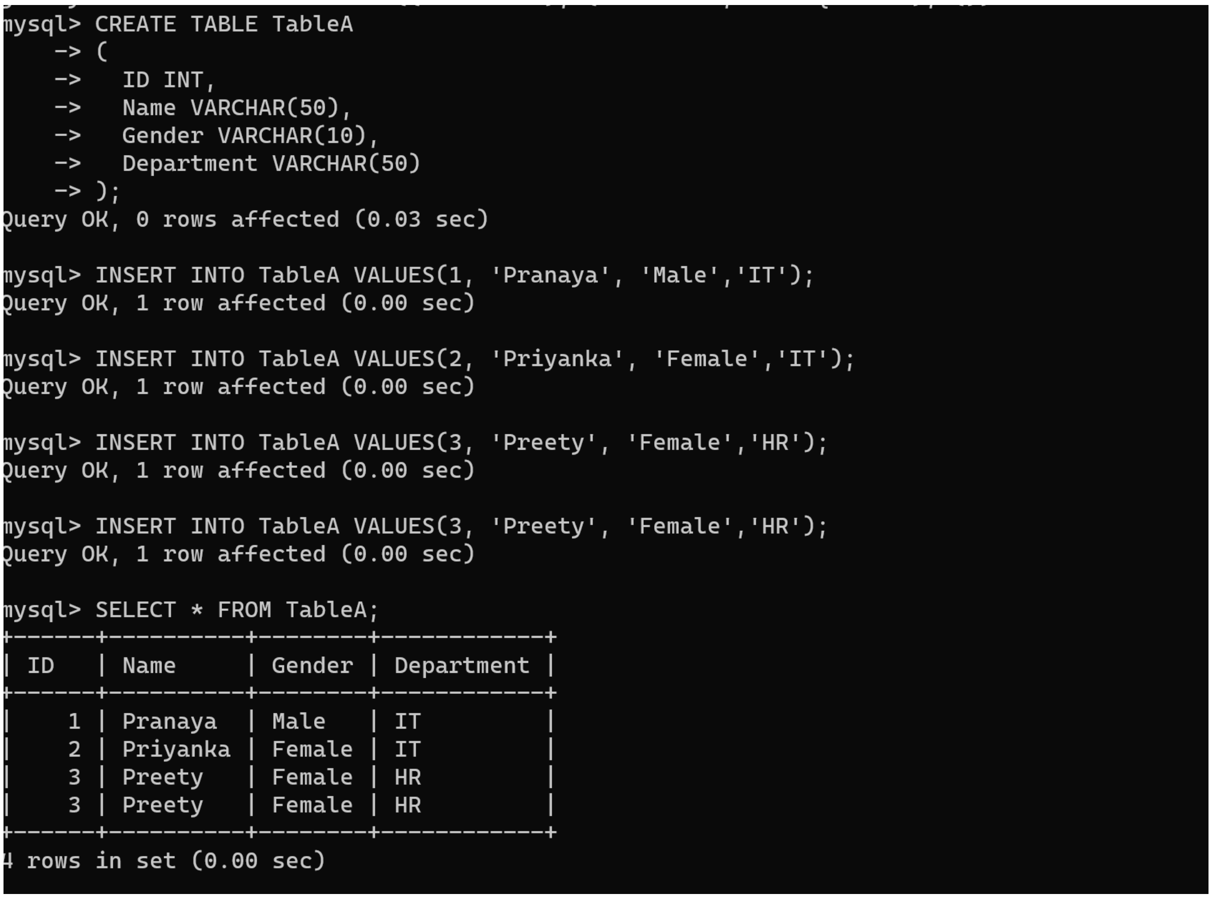
renaming the columns using switch case as subtotal and grand total:



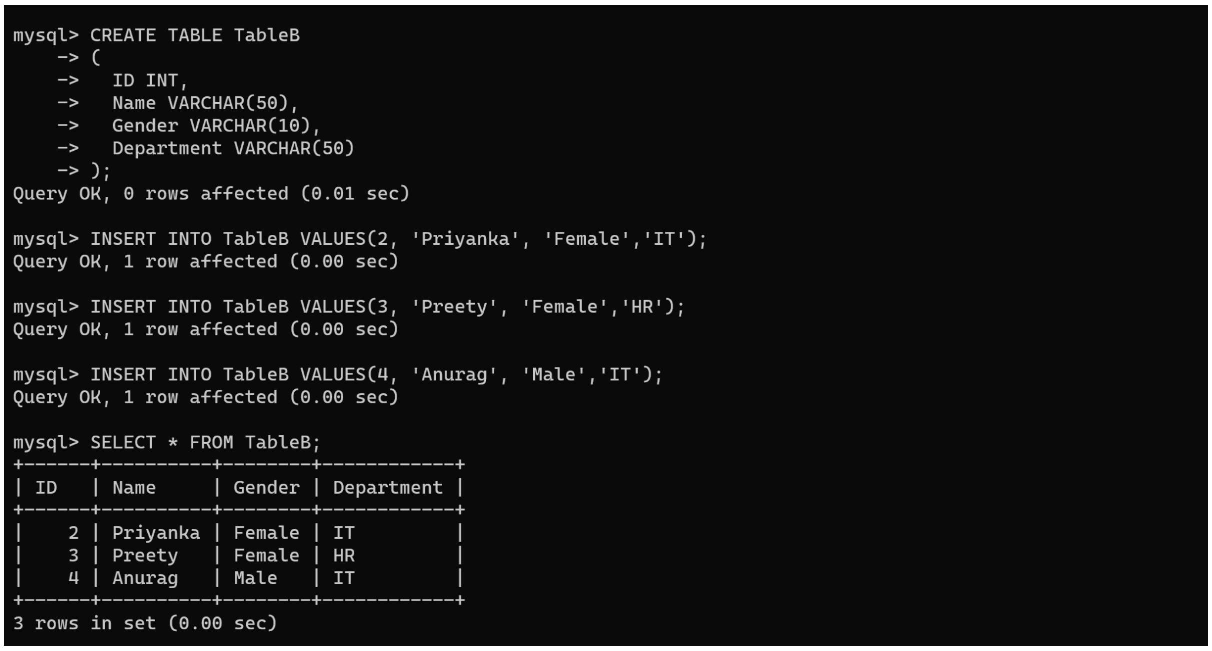
```
mysql> SELECT
    ->     CASE
    ->         WHEN GROUPING(SalesQuartes) = 1 AND GROUPING(SalesYear) = 0 THEN 'SubTotal'
    ->         WHEN GROUPING(SalesQuartes) = 1 AND GROUPING(SalesYear) = 1 THEN 'Grand Total'
    ->         ELSE CAST(SalesYear AS CHAR)
    ->     END AS SalesYear,
    ->     SalesQuartes,
    ->     SUM(SalesTotal) AS SalesTotal
    -> FROM SalesList
    -> GROUP By SalesYear, SalesQuartes with ROLLUP;
+-------------+--------------+------------+
| SalesYear   | SalesQuartes | SalesTotal |
+-------------+--------------+------------+
| 2019        | Q1           |      60.00 |
| 2019        | Q2           |      30.00 |
| 2019        | Q3           |     160.00 |
| 2019        | Q4           |     450.00 |
| SubTotal    | NULL         |     700.00 |
| 2020        | Q1           |     220.00 |
| 2020        | Q3           |      10.00 |
| 2020        | Q4           |     120.00 |
| SubTotal    | NULL         |     350.00 |
| Grand Total | NULL         |    1050.00 |
+-------------+--------------+------------+
10 rows in set, 1 warning (0.00 sec)
```

4)Differences Between UNION,EXCEPT and INTERSECT Operators in SQL Server.

Union: The UNION operator combines the result sets of two or more SELECT statements into a single result set.
Except: The EXCEPT operator returns all rows from the first SELECT statement that are absent in the second SELECT statement’s results.
Intersect: The INTERSECT operator returns all rows common to both SELECT statements.
Example:

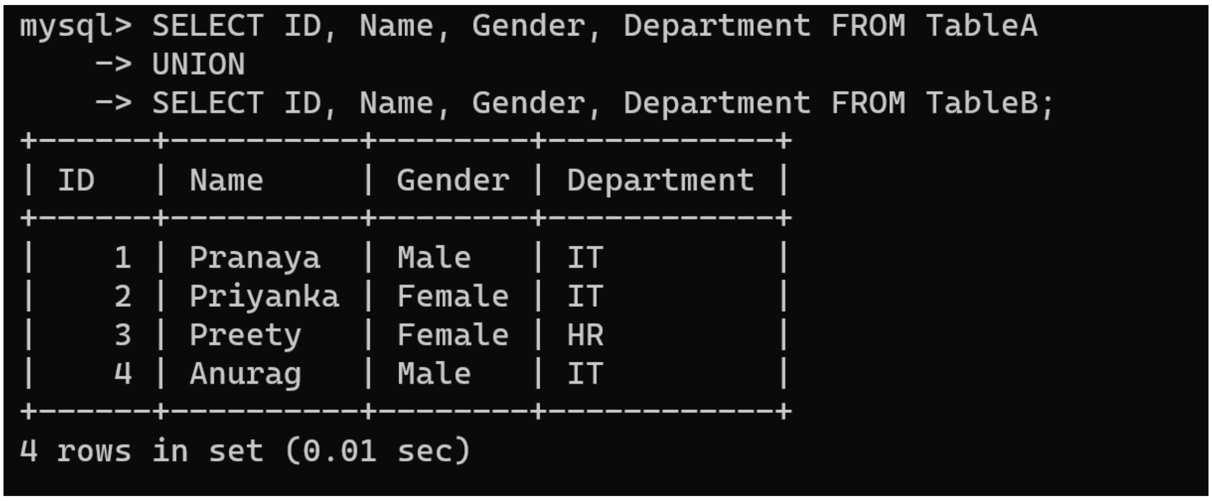


```
mysql> CREATE TABLE TableA
    -> (
    ->   ID INT,
    ->   Name VARCHAR(50),
    ->   Gender VARCHAR(10),
    ->   Department VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> INSERT INTO TableA VALUES(1, 'Pranaya', 'Male','IT');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO TableA VALUES(2, 'Priyanka', 'Female','IT');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO TableA VALUES(3, 'Preety', 'Female','HR');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO TableA VALUES(3, 'Preety', 'Female','HR');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM TableA;
+------+----------+--------+------------+
| ID   | Name     | Gender | Department |
+------+----------+--------+------------+
|    1 | Pranaya  | Male   | IT         |
|    2 | Priyanka | Female | IT         |
|    3 | Preety   | Female | HR         |
|    3 | Preety   | Female | HR         |
+------+----------+--------+------------+
4 rows in set (0.00 sec)
```



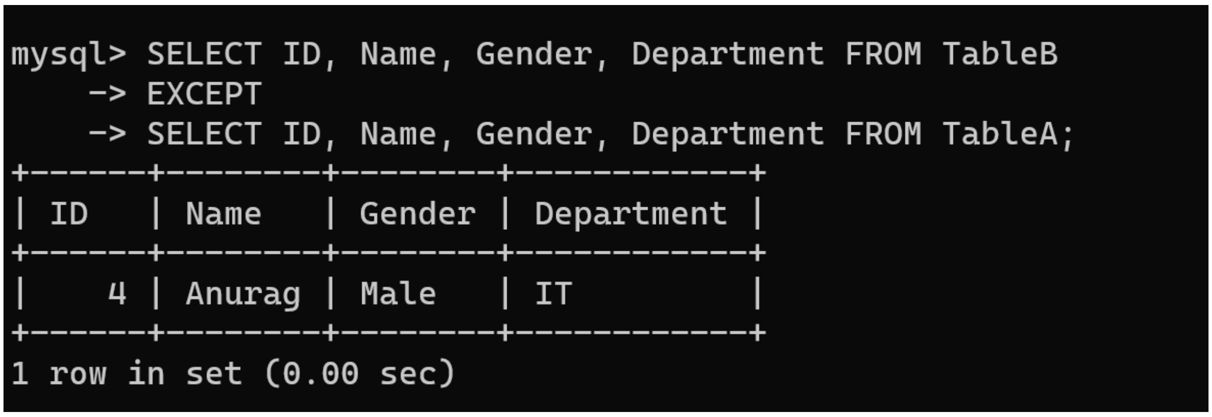
```
mysql> CREATE TABLE TableB
    -> (
    ->   ID INT,
    ->   Name VARCHAR(50),
    ->   Gender VARCHAR(10),
    ->   Department VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO TableB VALUES(2, 'Priyanka', 'Female','IT');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO TableB VALUES(3, 'Preety', 'Female','HR');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO TableB VALUES(4, 'Anurag', 'Male','IT');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM TableB;
+------+----------+--------+------------+
| ID   | Name     | Gender | Department |
+------+----------+--------+------------+
|    2 | Priyanka | Female | IT         |
|    3 | Preety   | Female | HR         |
|    4 | Anurag   | Male   | IT         |
+------+----------+--------+------------+
3 rows in set (0.00 sec)
```

UNION:



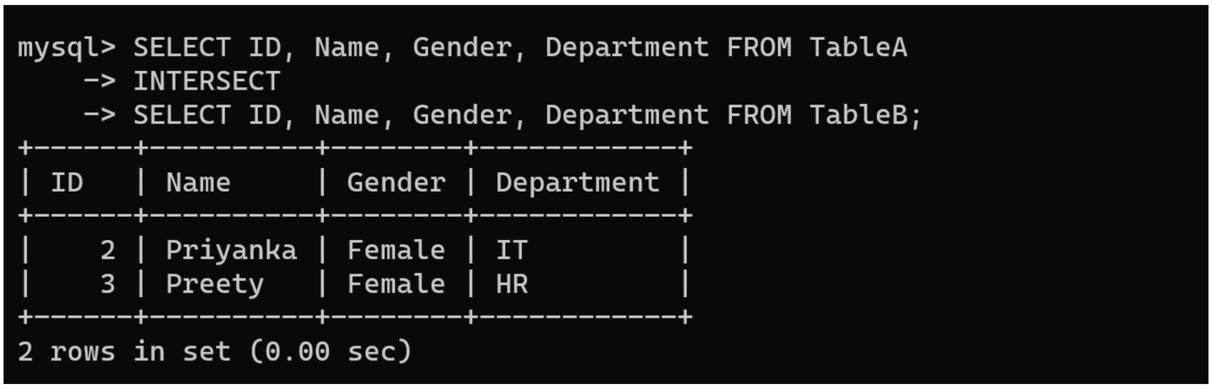
```
mysql> SELECT ID, Name, Gender, Department FROM TableA
    -> UNION
    -> SELECT ID, Name, Gender, Department FROM TableB;
+------+----------+--------+------------+
| ID   | Name     | Gender | Department |
+------+----------+--------+------------+
|    1 | Pranaya  | Male   | IT         |
|    2 | Priyanka | Female | IT         |
|    3 | Preety   | Female | HR         |
|    4 | Anurag   | Male   | IT         |
+------+----------+--------+------------+
4 rows in set (0.01 sec)
```

EXCEPT:



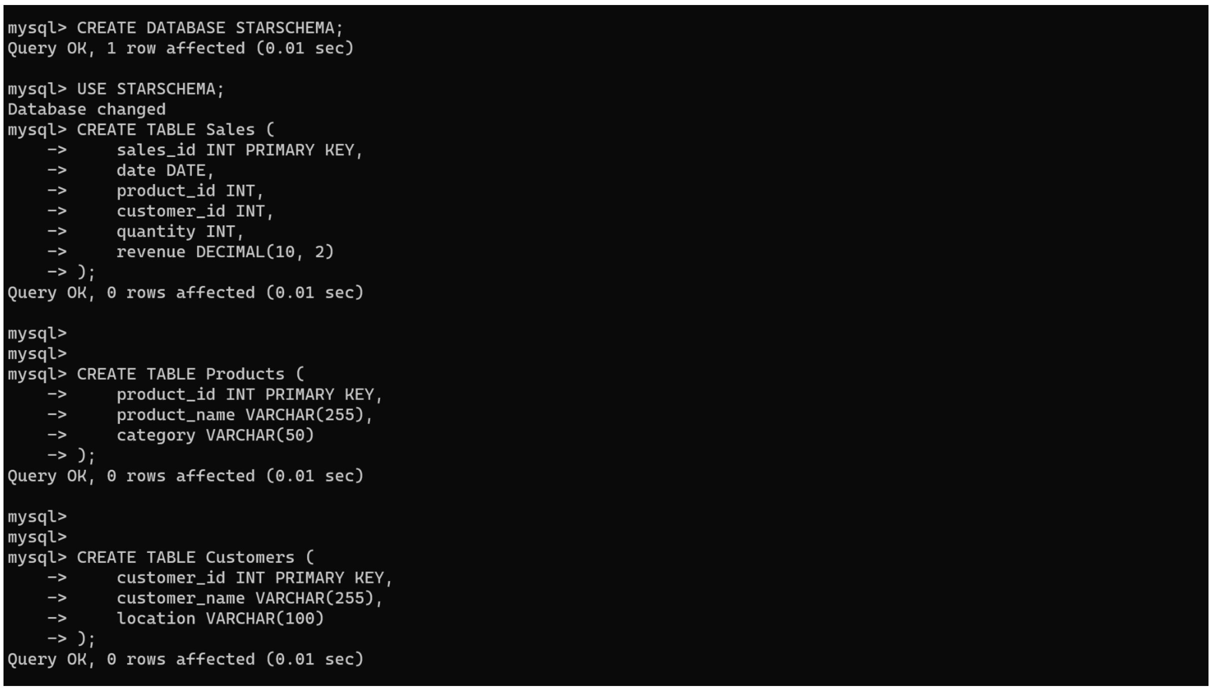
```
mysql> SELECT ID, Name, Gender, Department FROM TableB
    -> EXCEPT
    -> SELECT ID, Name, Gender, Department FROM TableA;
+------+--------+--------+------------+
| ID   | Name   | Gender | Department |
+------+--------+--------+------------+
|    4 | Anurag | Male   | IT         |
+------+--------+--------+------------+
1 row in set (0.00 sec)
```

INTERSECT:

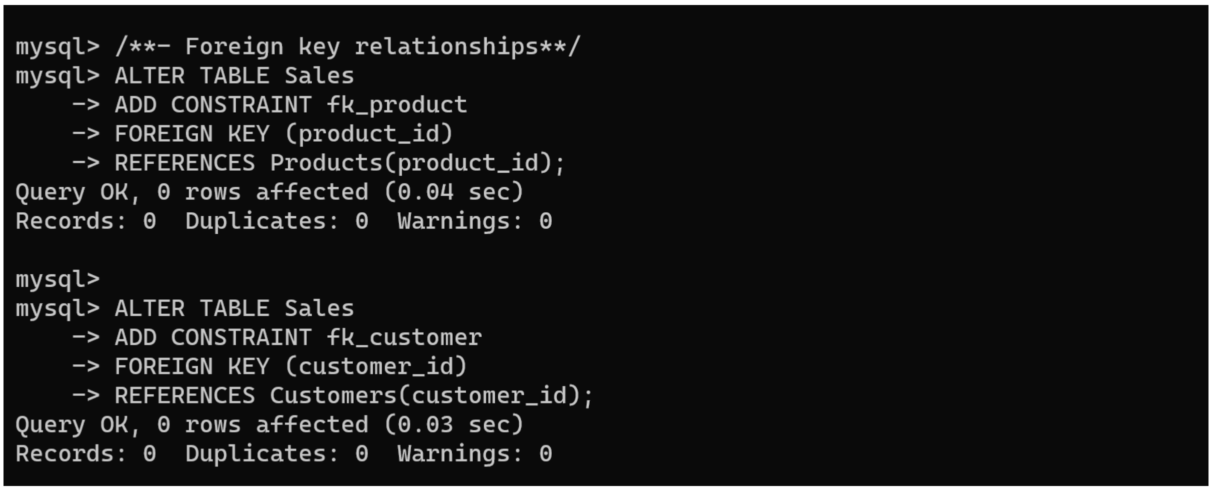


```
mysql> SELECT ID, Name, Gender, Department FROM TableA
    -> INTERSECT
    -> SELECT ID, Name, Gender, Department FROM TableB;
+------+----------+--------+------------+
| ID   | Name     | Gender | Department |
+------+----------+--------+------------+
|    2 | Priyanka | Female | IT         |
|    3 | Preety   | Female | HR         |
+------+----------+--------+------------+
2 rows in set (0.00 sec)
```

5)STAR SCHEMA:

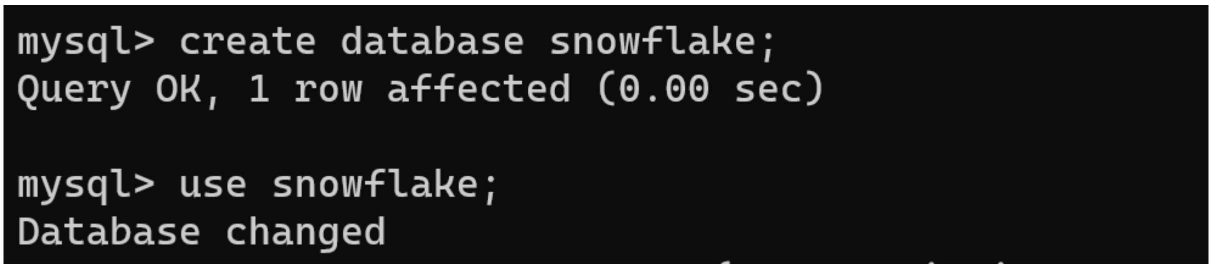


```
mysql> CREATE DATABASE STARSCHEMA;
Query OK, 1 row affected (0.01 sec)

mysql> USE STARSCHEMA;
Database changed
mysql> CREATE TABLE Sales (
    ->     sales_id INT PRIMARY KEY,
    ->     date DATE,
    ->     product_id INT,
    ->     customer_id INT,
    ->     quantity INT,
    ->     revenue DECIMAL(10, 2)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql>
mysql> CREATE TABLE Products (
    ->     product_id INT PRIMARY KEY,
    ->     product_name VARCHAR(255),
    ->     category VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql>
mysql> CREATE TABLE Customers (
    ->     customer_id INT PRIMARY KEY,
    ->     customer_name VARCHAR(255),
    ->     location VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.01 sec)
```

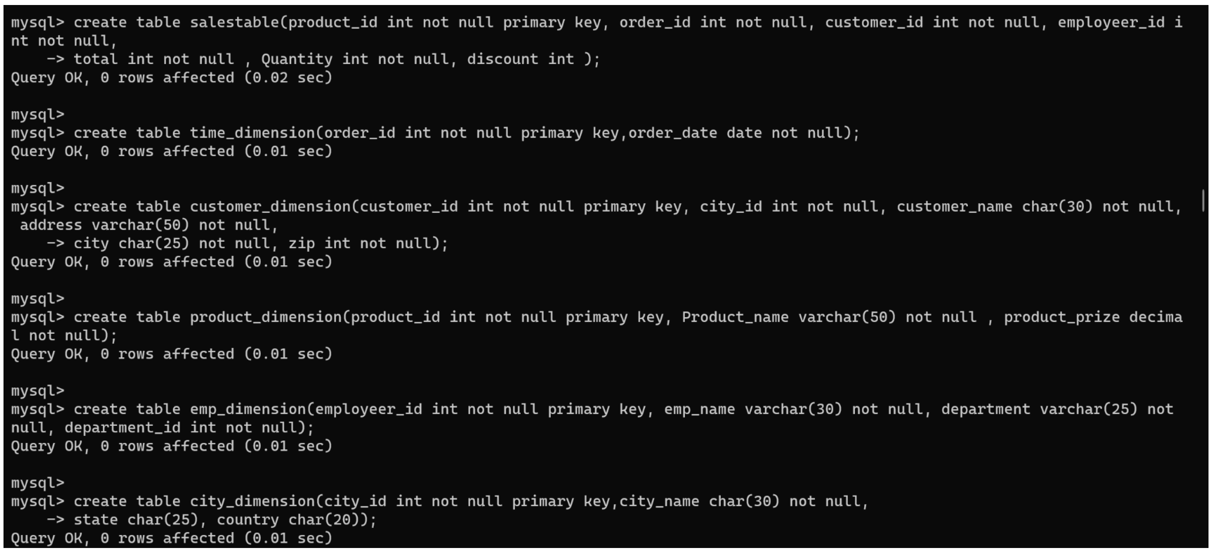


```
mysql> /**- Foreign key relationships**/
mysql> ALTER TABLE Sales
    -> ADD CONSTRAINT fk_product
    -> FOREIGN KEY (product_id)
    -> REFERENCES Products(product_id);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
mysql> ALTER TABLE Sales
    -> ADD CONSTRAINT fk_customer
    -> FOREIGN KEY (customer_id)
    -> REFERENCES Customers(customer_id);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

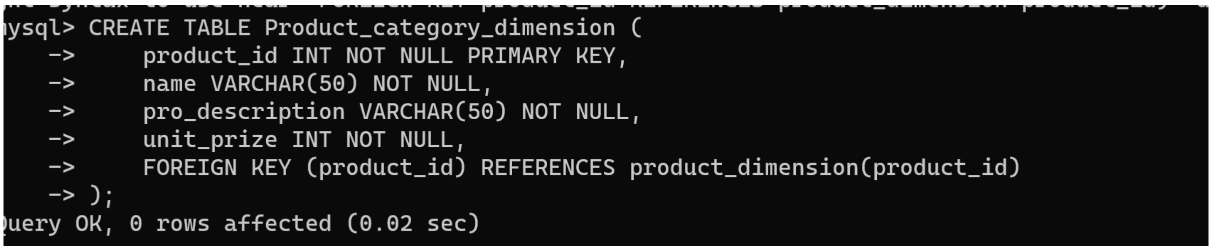
6)Snowflake:



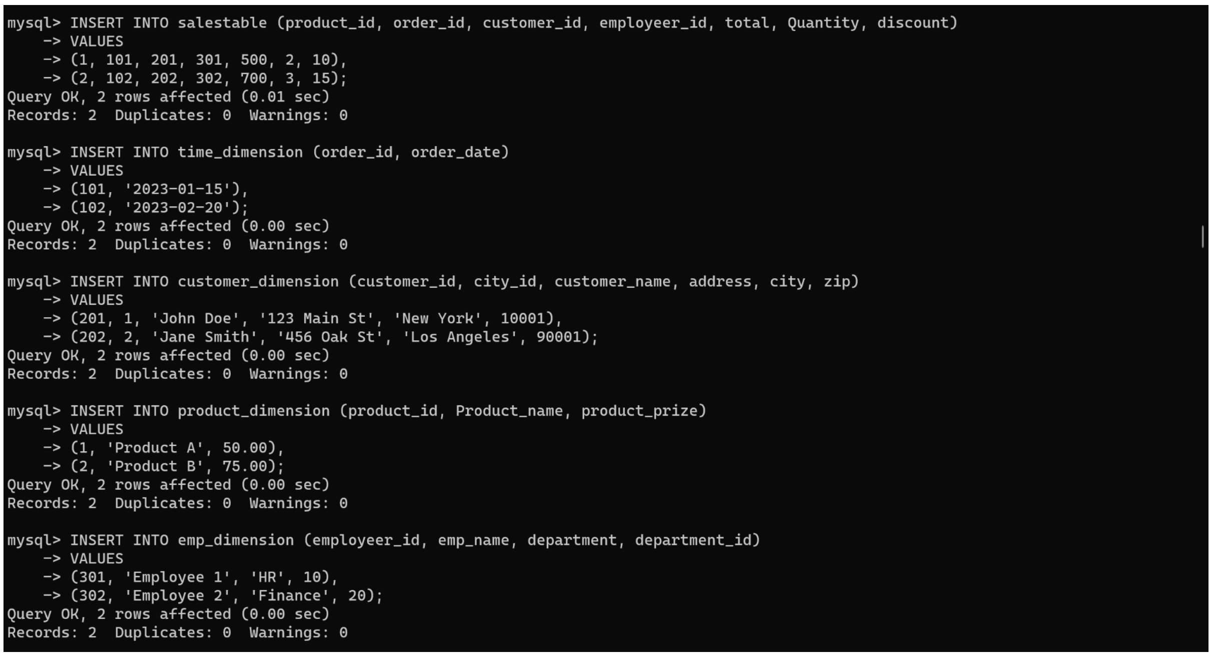
```
mysql> create database snowflake;
Query OK, 1 row affected (0.00 sec)

mysql> use snowflake;
Database changed
```



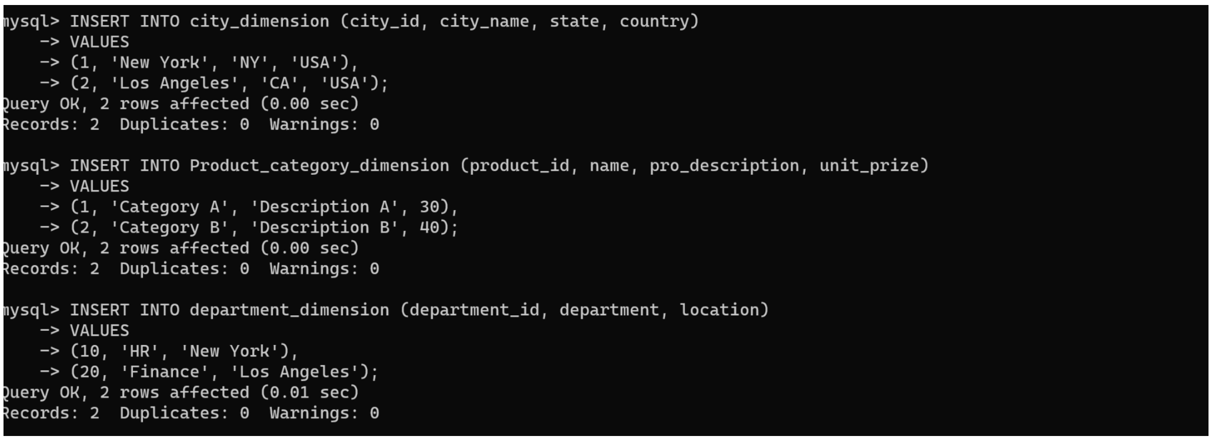
```
mysql> create table salestable(product_id int not null primary key, order_id int not null, customer_id int not null, employeer_id i
nt not null,
    -> total int not null , Quantity int not null, discount int );
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> create table time_dimension(order_id int not null primary key,order_date date not null);
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> create table customer_dimension(customer_id int not null primary key, city_id int not null, customer_name char(30) not null,
 address varchar(50) not null,
    -> city char(25) not null, zip int not null);
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> create table product_dimension(product_id int not null primary key, Product_name varchar(50) not null , product_prize decima
l not null);
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> create table emp_dimension(employeer_id int not null primary key, emp_name varchar(30) not null, department varchar(25) not
null, department_id int not null);
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> create table city_dimension(city_id int not null primary key,city_name char(30) not null,
    -> state char(25), country char(20));
Query OK, 0 rows affected (0.01 sec)
```



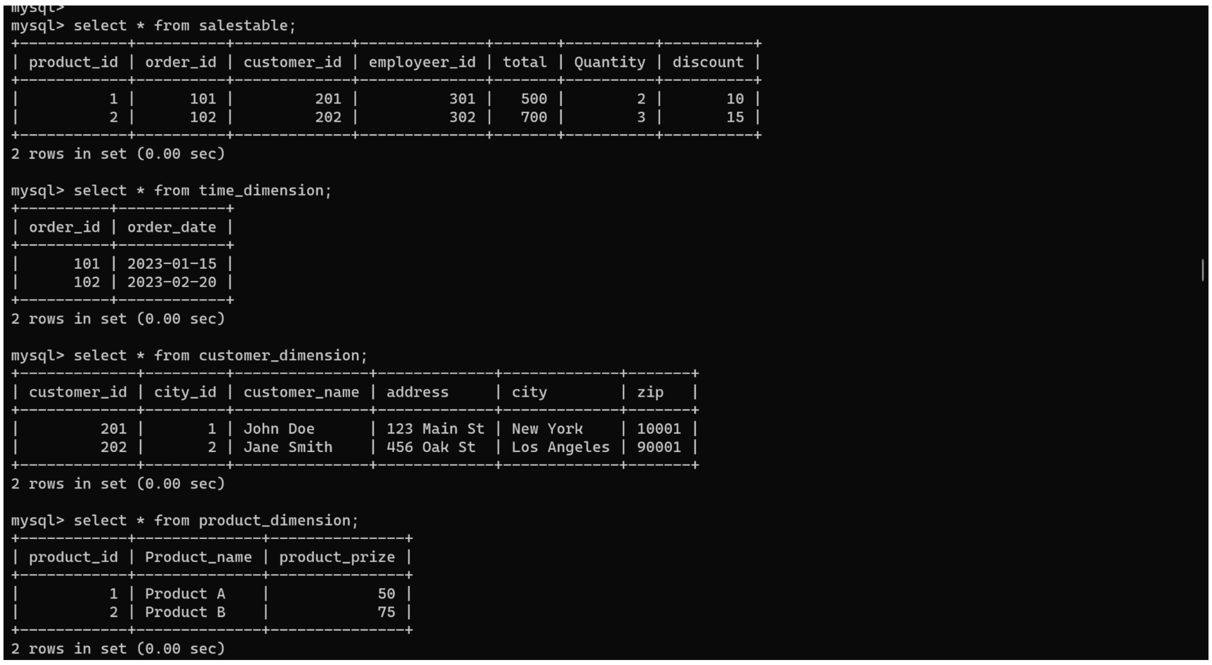
```
ysql> CREATE TABLE Product_category_dimension (
    ->     product_id INT NOT NULL PRIMARY KEY,
    ->     name VARCHAR(50) NOT NULL,
    ->     pro_description VARCHAR(50) NOT NULL,
    ->     unit_prize INT NOT NULL,
    ->     FOREIGN KEY (product_id) REFERENCES product_dimension(product_id)
    -> );
uery OK, 0 rows affected (0.02 sec)
```



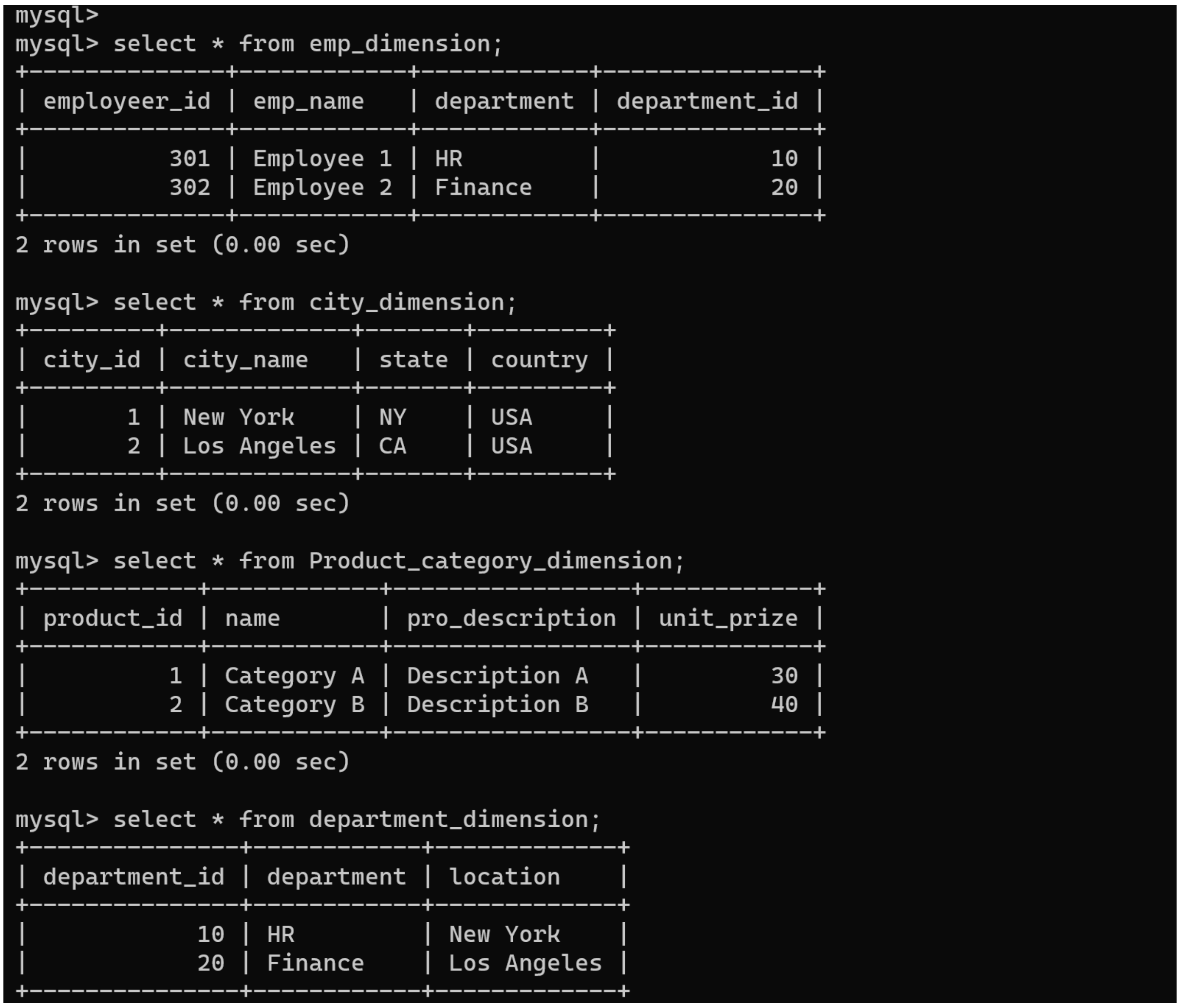
```
mysql> INSERT INTO salestable (product_id, order_id, customer_id, employeer_id, total, Quantity, discount)
    -> VALUES
    -> (1, 101, 201, 301, 500, 2, 10),
    -> (2, 102, 202, 302, 700, 3, 15);
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> INSERT INTO time_dimension (order_id, order_date)
    -> VALUES
    -> (101, '2023-01-15'),
    -> (102, '2023-02-20');
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> INSERT INTO customer_dimension (customer_id, city_id, customer_name, address, city, zip)
    -> VALUES
    -> (201, 1, 'John Doe', '123 Main St', 'New York', 10001),
    -> (202, 2, 'Jane Smith', '456 Oak St', 'Los Angeles', 90001);
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> INSERT INTO product_dimension (product_id, Product_name, product_prize)
    -> VALUES
    -> (1, 'Product A', 50.00),
    -> (2, 'Product B', 75.00);
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> INSERT INTO emp_dimension (employeer_id, emp_name, department, department_id)
    -> VALUES
    -> (301, 'Employee 1', 'HR', 10),
    -> (302, 'Employee 2', 'Finance', 20);
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0
```



```
mysql> INSERT INTO city_dimension (city_id, city_name, state, country)
    -> VALUES
    -> (1, 'New York', 'NY', 'USA'),
    -> (2, 'Los Angeles', 'CA', 'USA');
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Product_category_dimension (product_id, name, pro_description, unit_prize)
    -> VALUES
    -> (1, 'Category A', 'Description A', 30),
    -> (2, 'Category B', 'Description B', 40);
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> INSERT INTO department_dimension (department_id, department, location)
    -> VALUES
    -> (10, 'HR', 'New York'),
    -> (20, 'Finance', 'Los Angeles');
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0
```



```
mysql>
mysql> select * from salestable;
+------------+----------+-------------+--------------+-------+----------+----------+
| product_id | order_id | customer_id | employeer_id | total | Quantity | discount |
+------------+----------+-------------+--------------+-------+----------+----------+
|          1 |      101 |         201 |          301 |   500 |        2 |       10 |
|          2 |      102 |         202 |          302 |   700 |        3 |       15 |
+------------+----------+-------------+--------------+-------+----------+----------+
2 rows in set (0.00 sec)

mysql> select * from time_dimension;
+----------+------------+
| order_id | order_date |
+----------+------------+
|      101 | 2023-01-15 |
|      102 | 2023-02-20 |
+----------+------------+
2 rows in set (0.00 sec)

mysql> select * from customer_dimension;
+-------------+---------+---------------+-------------+-------------+-------+
| customer_id | city_id | customer_name | address     | city        | zip   |
+-------------+---------+---------------+-------------+-------------+-------+
|         201 |       1 | John Doe      | 123 Main St | New York    | 10001 |
|         202 |       2 | Jane Smith    | 456 Oak St  | Los Angeles | 90001 |
+-------------+---------+---------------+-------------+-------------+-------+
2 rows in set (0.00 sec)

mysql> select * from product_dimension;
+------------+--------------+---------------+
| product_id | Product_name | product_prize |
+------------+--------------+---------------+
|          1 | Product A    |            50 |
|          2 | Product B    |            75 |
+------------+--------------+---------------+
2 rows in set (0.00 sec)
```



```
mysql>
mysql> select * from emp_dimension;
+--------------+------------+------------+---------------+
| employeer_id | emp_name   | department | department_id |
+--------------+------------+------------+---------------+
|          301 | Employee 1 | HR         |            10 |
|          302 | Employee 2 | Finance    |            20 |
+--------------+------------+------------+---------------+
2 rows in set (0.00 sec)

mysql> select * from city_dimension;
+---------+-------------+-------+---------+
| city_id | city_name   | state | country |
+---------+-------------+-------+---------+
|       1 | New York    | NY    | USA     |
|       2 | Los Angeles | CA    | USA     |
+---------+-------------+-------+---------+
2 rows in set (0.00 sec)

mysql> select * from Product_category_dimension;
+------------+------------+-----------------+------------+
| product_id | name       | pro_description | unit_prize |
+------------+------------+-----------------+------------+
|          1 | Category A | Description A   |         30 |
|          2 | Category B | Description B   |         40 |
+------------+------------+-----------------+------------+
2 rows in set (0.00 sec)

mysql> select * from department_dimension;
+---------------+------------+-------------+
| department_id | department | location    |
+---------------+------------+-------------+
|            10 | HR         | New York    |
|            20 | Finance    | Los Angeles |
+---------------+------------+-------------+
```