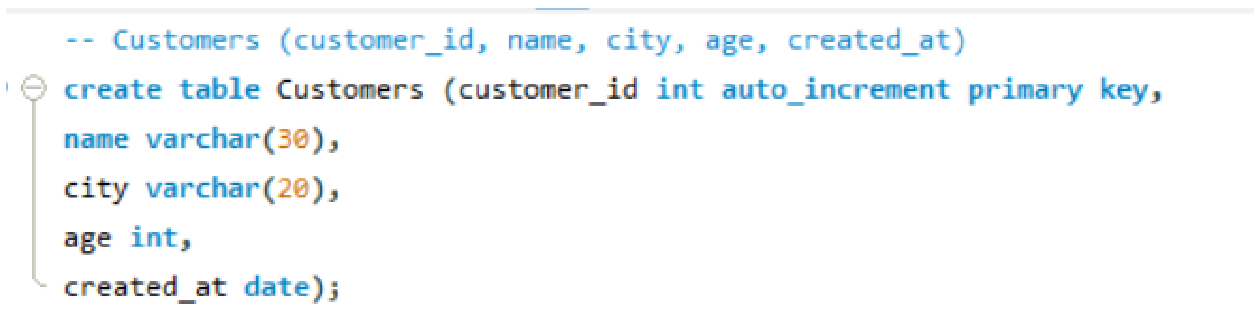
SQL Case Study Questions

- -Scenario-:

You are working for a -Retail Company- managing sales, customers, and products in a SQL database.

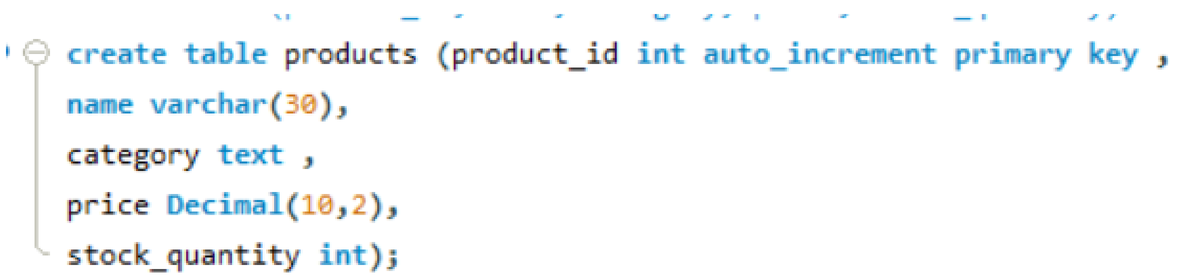
The database contains the following -tables-:

- -Customers (customer_id, name, city, age, created_at)-



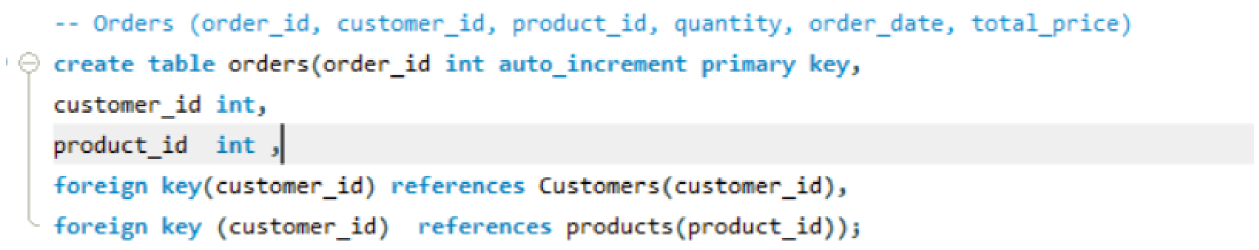
```sql
-- Customers (customer_id, name, city, age, created_at)
create table Customers (customer_id int auto_increment primary key,
name varchar(30),
city varchar(20),
age int,
created_at date);
```

- -Products (product_id, name, category, price, stock_quantity)-



```sql
create table products (product_id int auto_increment primary key ,
name varchar(30),
category text ,
price Decimal(10,2),
stock_quantity int);
```

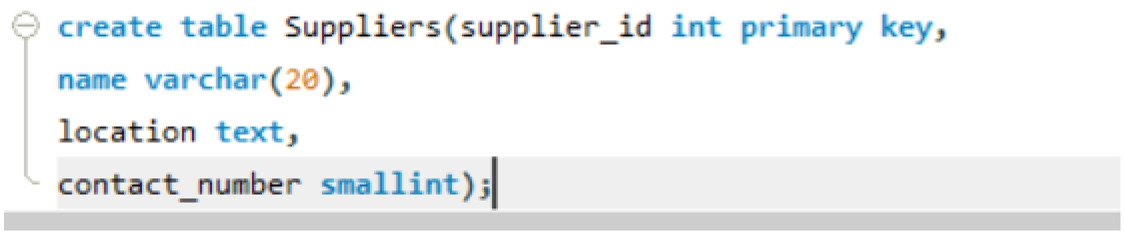
- -Orders (order_id, customer_id, product_id, quantity, order_date, total_price)-



```sql
-- Orders (order_id, customer_id, product_id, quantity, order_date, total_price)
create table orders(order_id int auto_increment primary key,
customer_id int,
product_id  int ,
foreign key(customer_id) references Customers(customer_id),
foreign key (customer_id)  references products(product_id));
```

-- DDL (Data Definition Language-

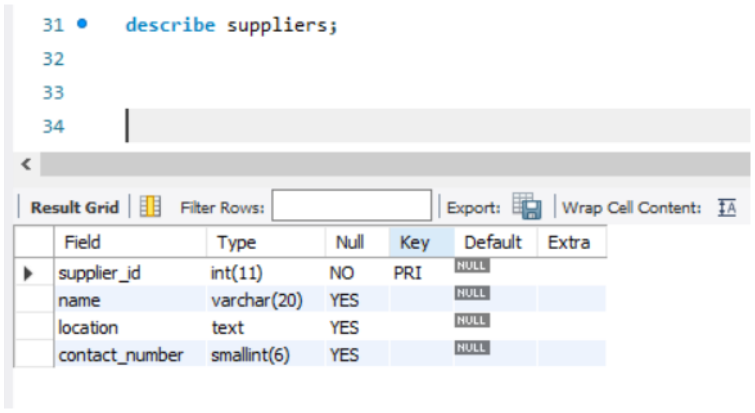
1. **Create a table- -Suppliers- with columns: -supplier_id (PK)-, -name-, -location-, -contact_number-.**



```sql
create table Suppliers(supplier_id int primary key,
name varchar(20),
location text,
contact_number smallint);
```

2. **Modify the column- -contact_number- in the -Suppliers- table to increase its length.**

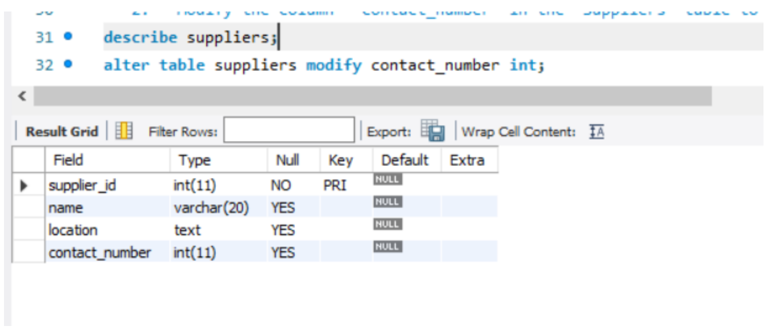
Before update contact_number:-



```sql
describe suppliers;
```

Field	Type	Null	Key	Default	Extra
supplier_id	int(11)	NO	PRI	NULL	
name	varchar(20)	YES		NULL	
location	text	YES		NULL	
contact_number	smallint(6)	YES		NULL	

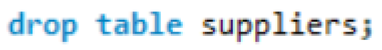
After update contact_number :-



```sql
-- 2. Modify the column contact_number in the Suppliers table to
describe suppliers;
alter table suppliers modify contact_number int;
```

Field	Type	Null	Key	Default	Extra
supplier_id	int(11)	NO	PRI	NULL	
name	varchar(20)	YES		NULL	
location	text	YES		NULL	
contact_number	int(11)	YES		NULL	

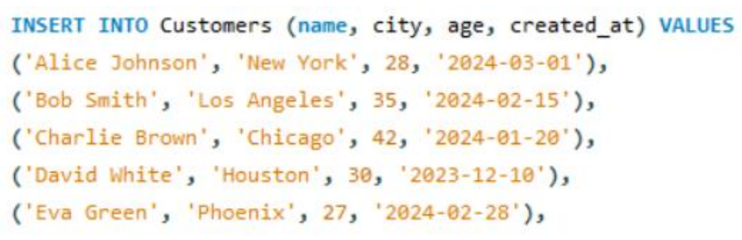
3. **Delete the table- -Suppliers- permanently.**



```sql
drop table suppliers;
```

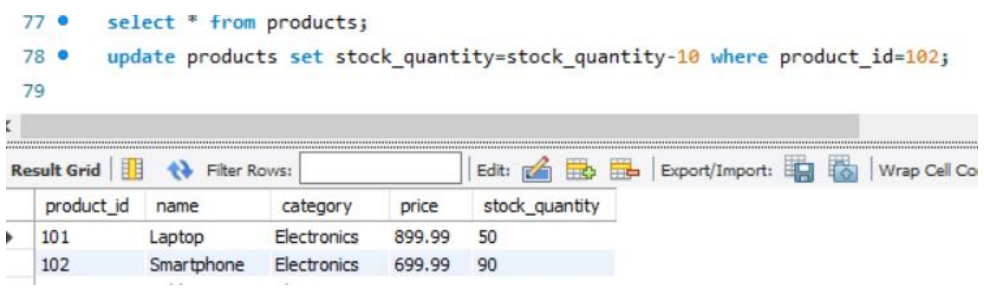
-- DML (Data Manipulation Language)-

4-Insert -5 sample records- into the -Customers- table.



```
INSERT INTO Customers (name, city, age, created_at) VALUES
('Alice Johnson', 'New York', 28, '2024-03-01'),
('Bob Smith', 'Los Angeles', 35, '2024-02-15'),
('Charlie Brown', 'Chicago', 42, '2024-01-20'),
('David White', 'Houston', 30, '2023-12-10'),
('Eva Green', 'Phoenix', 27, '2024-02-28'),
```

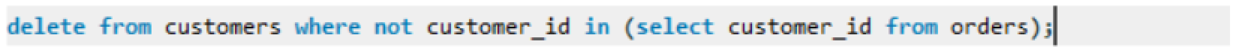
5-Update the -stock_quantity- in the -Products- table for -product_id = 102- by decreasing it by 10.



```
select * from products;
update products set stock_quantity=stock_quantity-10 where product_id=102;
```

product_id	name	category	price	stock_quantity
101	Laptop	Electronics	899.99	50
102	Smartphone	Electronics	699.99	90

6-Delete all customers who -haven't placed any orders- from the -Customers- table.



```
delete from customers where not customer_id in (select customer_id from orders);
```

-- ORDER BY, LIMIT, and OFFSET-

7□-Retrieve all products -ordered by price in descending order

--->select * from products order by price desc;

8□-Display the -first 5 most expensive products- from the -Products- table.

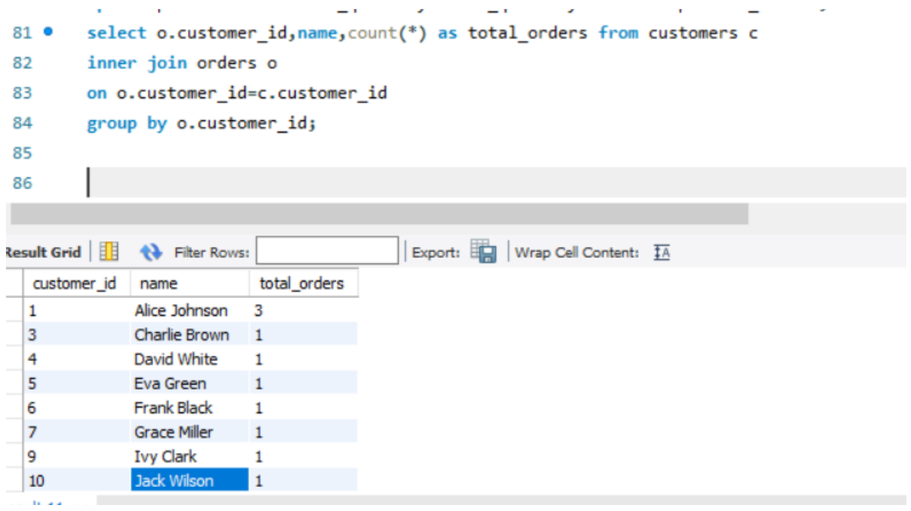
select * from products order by price desc limit 5;

9□-Retrieve the -next 5 most expensive products- (pagination using -OFFSET-).

select * from products order by price desc limit 5,5;

-- GROUP BY and HAVING-

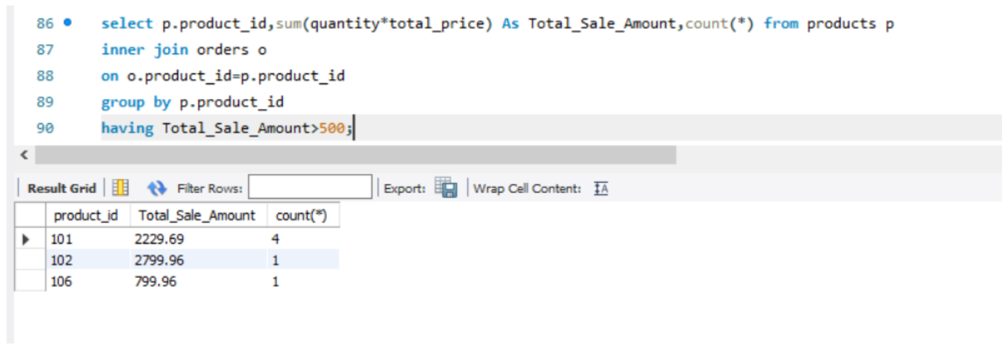
10- Find the total number of orders placed by each customer -(GROUP BY customer_id)-.



```sql
select o.customer_id,name,count(*) as total_orders from customers c
inner join orders o
on o.customer_id=c.customer_id
group by o.customer_id;
```

customer_id	name	total_orders
1	Alice Johnson	3
3	Charlie Brown	1
4	David White	1
5	Eva Green	1
6	Frank Black	1
7	Grace Miller	1
9	Ivy Clark	1
10	Jack Wilson	1

11-Find the -total sales amount- for each product where the total sales exceed -$500

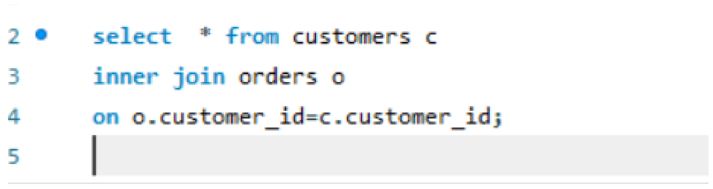


```sql
select p.product_id,sum(quantity*total_price) As Total_Sale_Amount,count(*) from products p
inner join orders o
on o.product_id=p.product_id
group by p.product_id
having Total_Sale_Amount>500;
```

product_id	Total_Sale_Amount	count(*)
101	2229.69	4
102	2799.96	1
106	799.96	1

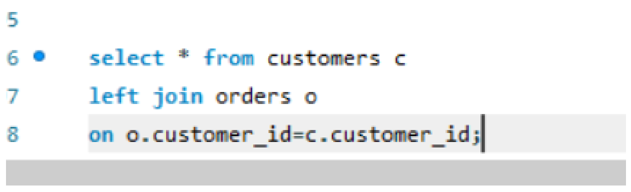
-- JOINS (INNER, LEFT, RIGHT, FULL OUTER)-

12-Retrieve a list of -customers who have placed orders-, including their order details (-INNER JOIN-).

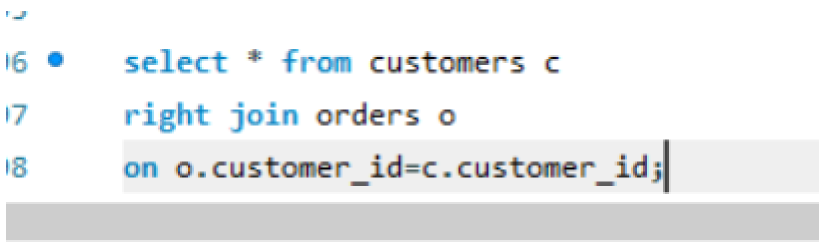


```sql
select  * from customers c
inner join orders o
on o.customer_id=c.customer_id;
```

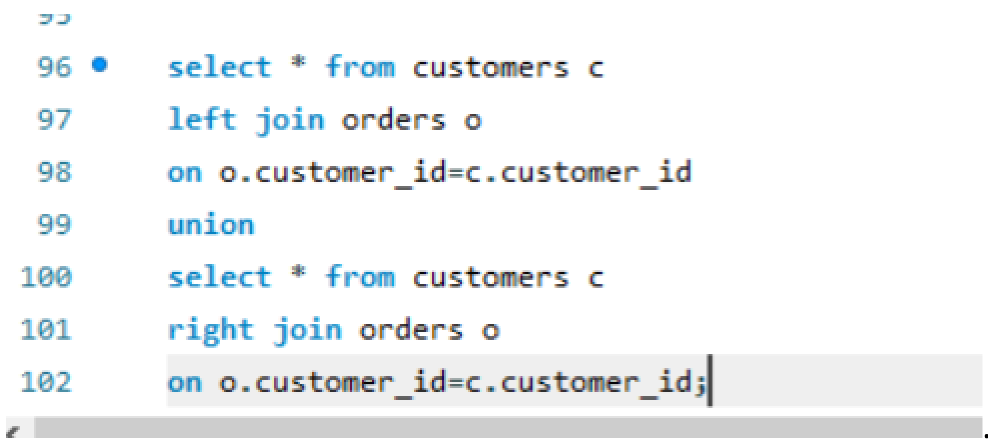
13-Retrieve a list of all customers and -their order details (if any), otherwise show NULL- (-LEFT JOIN-).


```
select * from customers c
left join orders o
on o.customer_id=c.customer_id;
```

14-Retrieve a list of all orders along with customer details, even if -some customers have not placed any orders- (-RIGHT JOIN-).

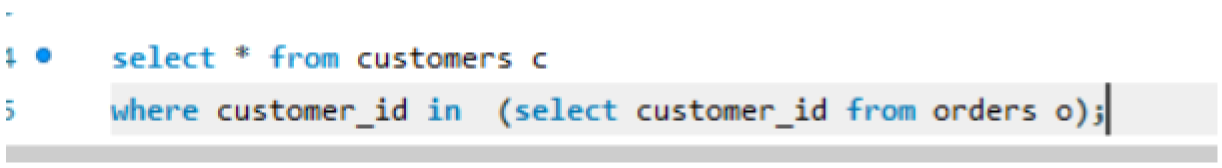

```
select * from customers c
right join orders o
on o.customer_id=c.customer_id;
```

15-Retrieve a list of -all customers and all orders-, including those -without a matching record- (-FULL OUTER JOIN-)


```
select * from customers c
left join orders o
on o.customer_id=c.customer_id
union
select * from customers c
right join orders o
on o.customer_id=c.customer_id;
```
.

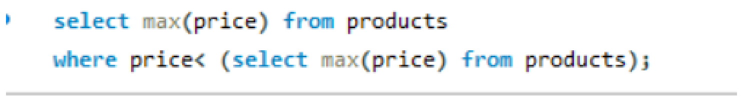
-- Subqueries-

16-Retrieve the details of customers -who have placed at least one order- (use a subquery inside -WHERE-).



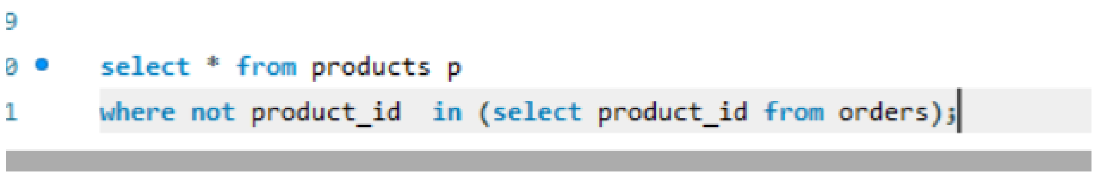
```sql
select * from customers c
where customer_id in  (select customer_id from orders o);
```

17-Find the -second-highest priced product- using a subquery.



```sql
select max(price) from products
where price< (select max(price) from products);
```

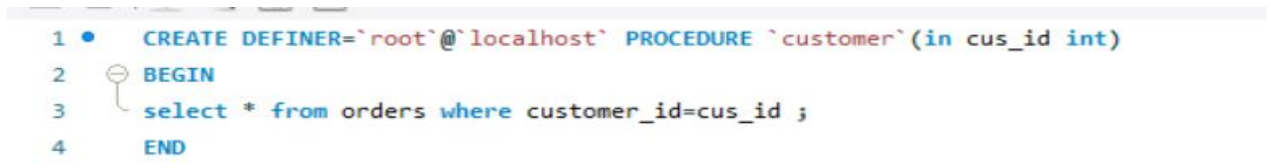
18-Retrieve the list of -products that have never been ordered- using a subquery.



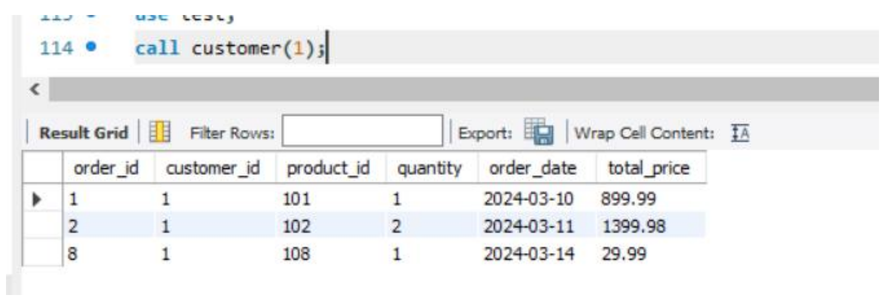
```sql
select * from products p
where not product_id  in (select product_id from orders);
```

-- Stored Procedures-

19-Write a -stored procedure- that takes a -customer_id- as input and returns all their orders.



```sql
CREATE DEFINER=`root`@`localhost` PROCEDURE `customer`(in cus_id int)
BEGIN
select * from orders where customer_id=cus_id ;
END
```



```sql
use test;
call customer(1);
```

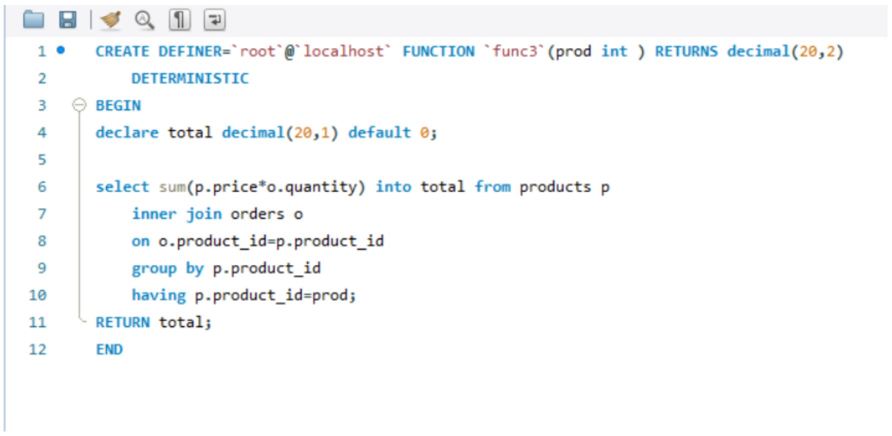
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	customer_id	product_id	quantity	order_date	total_price
1	1	101	1	2024-03-10	899.99
2	1	102	2	2024-03-11	1399.98
8	1	108	1	2024-03-14	29.99

-- User-Defined Functions (UDF)-

20-Create a -UDF- that takes a -product_id- and returns the total revenue generated from that product.

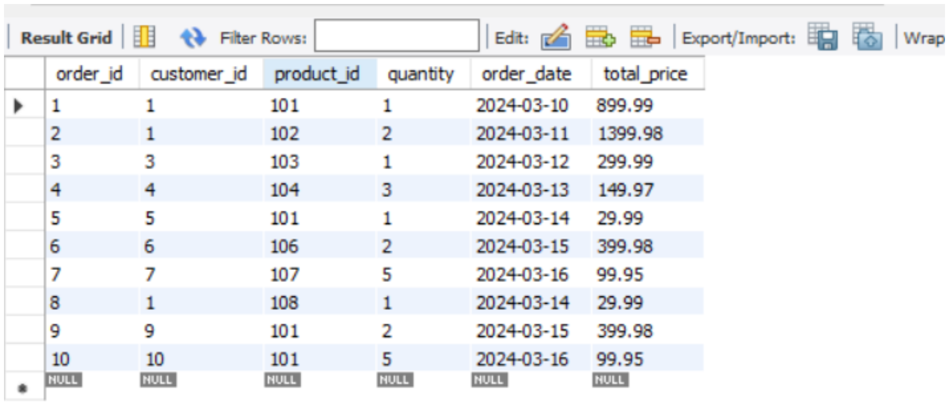
Code:-



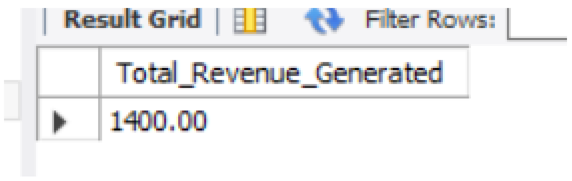
```
CREATE DEFINER=`root`@`localhost` FUNCTION `func3`(prod int ) RETURNS decimal(20,2)
    DETERMINISTIC
BEGIN
declare total decimal(20,1) default 0;

select sum(p.price*o.quantity) into total from products p
    inner join orders o
    on o.product_id=p.product_id
    group by p.product_id
    having p.product_id=prod;
RETURN total;
END
```

Output:



order_id	customer_id	product_id	quantity	order_date	total_price
1	1	101	1	2024-03-10	899.99
2	1	102	2	2024-03-11	1399.98
3	3	103	1	2024-03-12	299.99
4	4	104	3	2024-03-13	149.97
5	5	101	1	2024-03-14	29.99
6	6	106	2	2024-03-15	399.98
7	7	107	5	2024-03-16	99.95
8	1	108	1	2024-03-14	29.99
9	9	101	2	2024-03-15	399.98
10	10	101	5	2024-03-16	99.95
NULL	NULL	NULL	NULL	NULL	NULL



Total_Revenue_Generated
1400.00