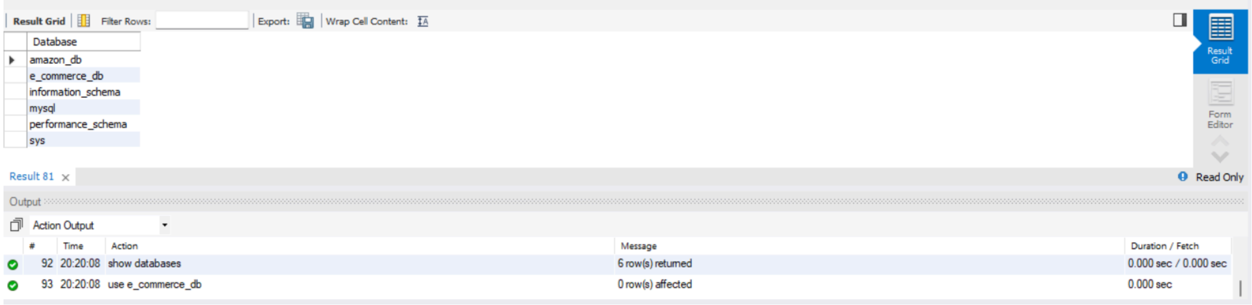
TASK - 3

Code:

show databases;

use e_commerce_db;

output:



	Database
▶	amazon_db
	e_commerce_db
	information_schema
	mysql
	performance_schema
	sys

	#	Time	Action	Message	Duration / Fetch
✓	92	20:20:08	show databases	6 row(s) returned	0.000 sec / 0.000 sec
✓	93	20:20:08	use e_commerce_db	0 row(s) affected	0.000 sec

Code:

show tables;

select * from customer;

select * from order_items;

select * from orders;

select * from payments;

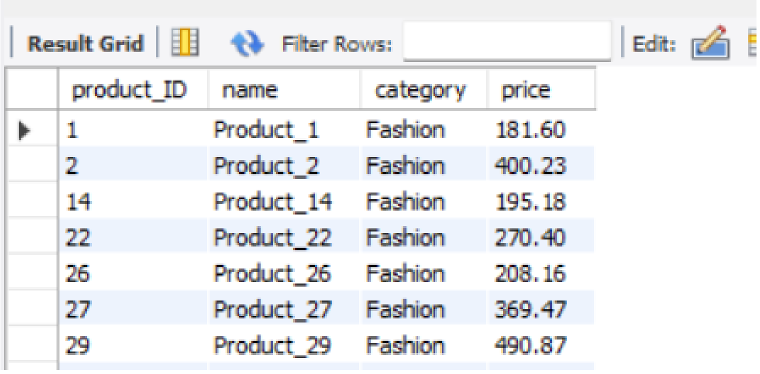
select * from products;

#Where clause

Code:

select * from products where category = "Fashion";

output:



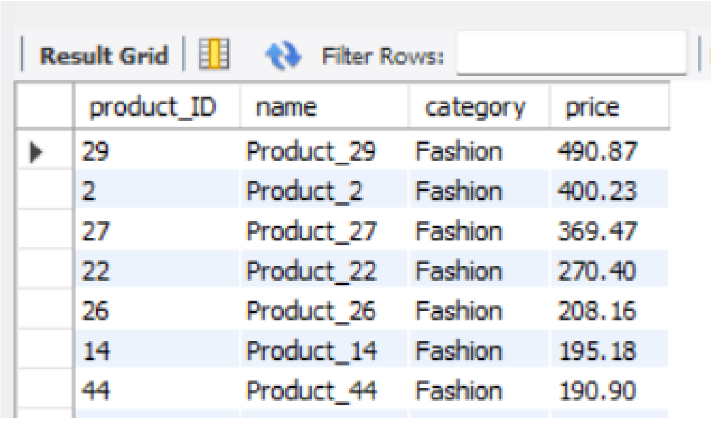
	product_ID	name	category	price
▶	1	Product_1	Fashion	181.60
	2	Product_2	Fashion	400.23
	14	Product_14	Fashion	195.18
	22	Product_22	Fashion	270.40
	26	Product_26	Fashion	208.16
	27	Product_27	Fashion	369.47
	29	Product_29	Fashion	490.87

#Order by clause

Code:

select * from products where category="Fashion" order by price desc;

output:



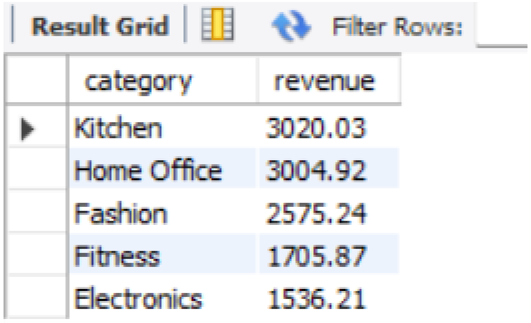
product_ID	name	category	price
29	Product_29	Fashion	490.87
2	Product_2	Fashion	400.23
27	Product_27	Fashion	369.47
22	Product_22	Fashion	270.40
26	Product_26	Fashion	208.16
14	Product_14	Fashion	195.18
44	Product_44	Fashion	190.90

# Group by top 5 category by revenue

Code:

select category,sum(price) as revenue from products group by category order by sum(price) desc limit 5;

output:



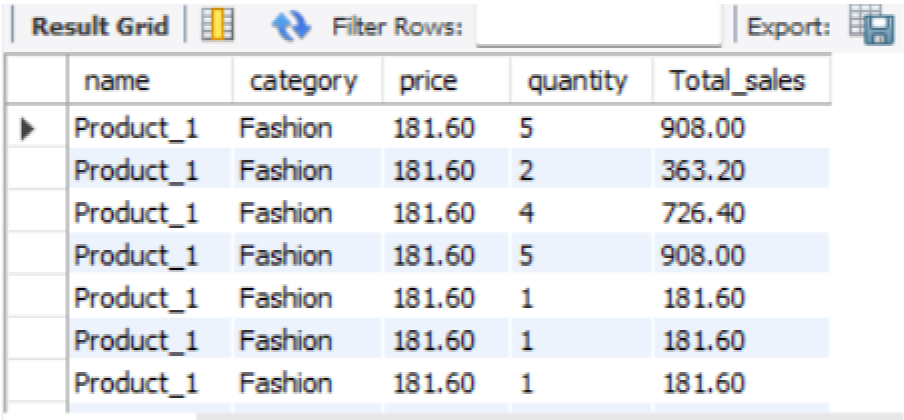
category	revenue
Kitchen	3020.03
Home Office	3004.92
Fashion	2575.24
Fitness	1705.87
Electronics	1536.21

#Inner join

Code:

select p.name,p.category,p.price,oi.quantity ,(p.price* oi.quantity) as Total_sales from products p inner join order_items oi on p.product_ID=oi.product_id;

output:



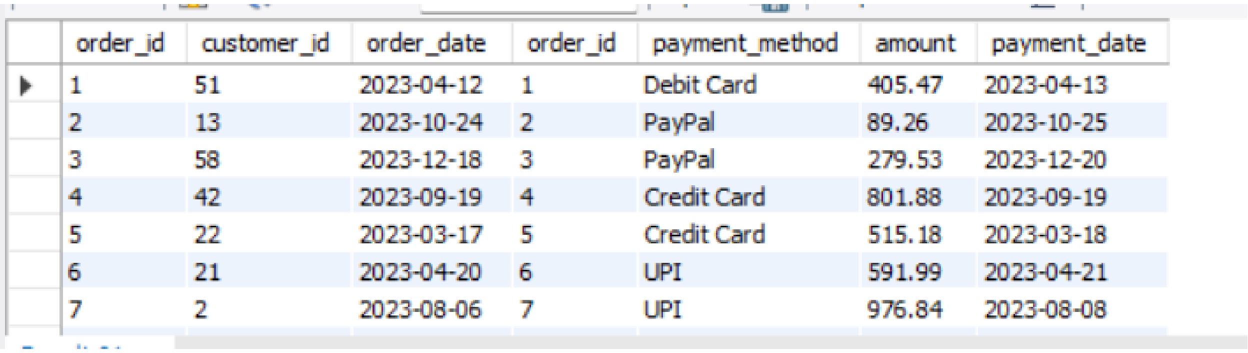
name	category	price	quantity	Total_sales
Product_1	Fashion	181.60	5	908.00
Product_1	Fashion	181.60	2	363.20
Product_1	Fashion	181.60	4	726.40
Product_1	Fashion	181.60	5	908.00
Product_1	Fashion	181.60	1	181.60
Product_1	Fashion	181.60	1	181.60
Product_1	Fashion	181.60	1	181.60

# Left join

Code:

```
select *  from orders o left join payments p on o.order_id= p.order_id;
```

output:



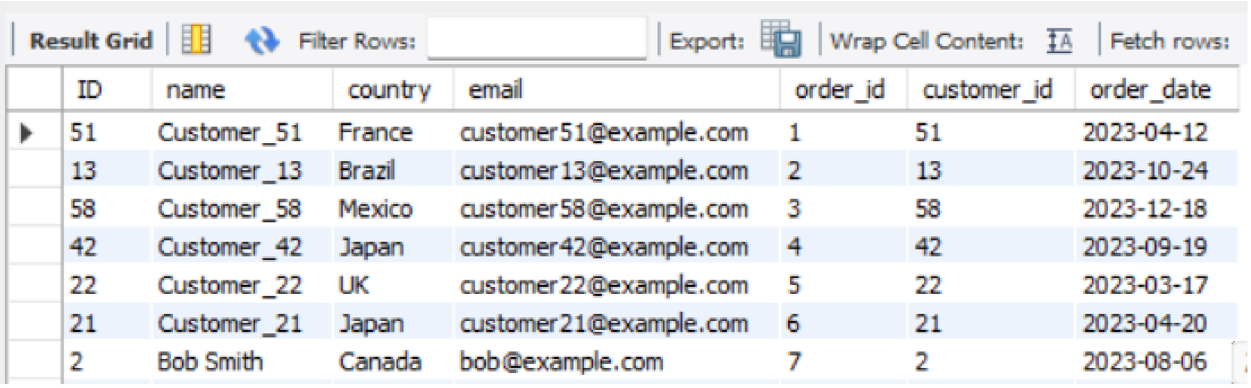
order_id	customer_id	order_date	order_id	payment_method	amount	payment_date
1	51	2023-04-12	1	Debit Card	405.47	2023-04-13
2	13	2023-10-24	2	PayPal	89.26	2023-10-25
3	58	2023-12-18	3	PayPal	279.53	2023-12-20
4	42	2023-09-19	4	Credit Card	801.88	2023-09-19
5	22	2023-03-17	5	Credit Card	515.18	2023-03-18
6	21	2023-04-20	6	UPI	591.99	2023-04-21
7	2	2023-08-06	7	UPI	976.84	2023-08-08

# Right join

Code:

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

output:



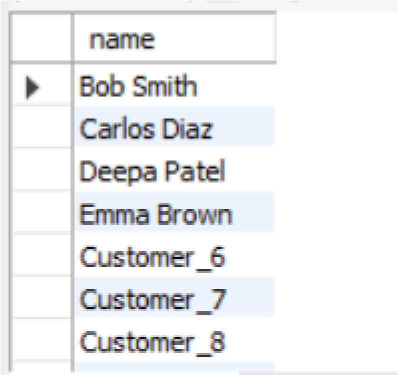
ID	name	country	email	order_id	customer_id	order_date
51	Customer_51	France	customer51@example.com	1	51	2023-04-12
13	Customer_13	Brazil	customer13@example.com	2	13	2023-10-24
58	Customer_58	Mexico	customer58@example.com	3	58	2023-12-18
42	Customer_42	Japan	customer42@example.com	4	42	2023-09-19
22	Customer_22	UK	customer22@example.com	5	22	2023-03-17
21	Customer_21	Japan	customer21@example.com	6	21	2023-04-20
2	Bob Smith	Canada	bob@example.com	7	2	2023-08-06

#Sub Query

Code:

```
select name from customer where ID in (select o.customer_id from orders o join payments p on
o.order_id=p.order_id where amount >=900) ;
```

output:



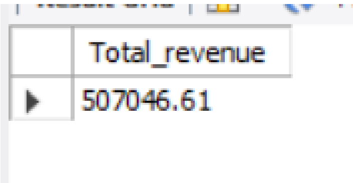
name
Bob Smith
Carlos Diaz
Deepa Patel
Emma Brown
Customer_6
Customer_7
Customer_8

# Aggregate function Sum and Avg

Code:

```
select sum(amount) as Total_revenue from payments;
```

output:

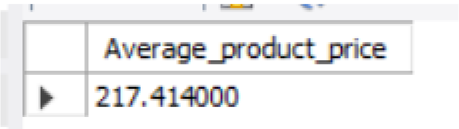


Total_revenue
507046.61

Code:

```
select avg(price) as Average_product_price  from products;
```

output:



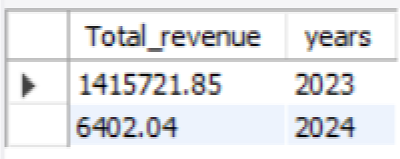
Average_product_price
217.414000

# Trend analysis by year

Code:

```
select sum(p.price*oi.quantity) as Total_revenue ,year(o.order_date) as years from products as p
inner join order_items as oi on p.product_id = oi.product_id
inner join orders o on oi.order_id =o.order_id group by year(o.order_date) ;
```

output:



Total_revenue	years
1415721.85	2023
6402.04	2024