```
--droping the database Ecommerce
Drop database if exists Ecommerce;


--creating the database Ecommerce
Create Database Ecommerce;


--droping the table customers
DROP TABLE CUSTOMERS;


--creating the table Orders
create table Customers(
customer_id INT PRIMARY KEY,
name VARCHAR(100),
email VARCHAR(100),
country VARCHAR(100),
signup_date DATE
);

--Retriving the data from customers Table
SELECT * FROM Customers;

--droping the table orders
Drop Table orders;
```

```
--creating the table Orders
CREATE TABLE ORDERS(
order_id INT PRIMARY KEY,
customer_id INT REFERENCES CUSTOMERS(CUSTOMER_ID),
product_id INT,
order_date DATE,
quantity INT,
total_amount DECIMAL(10,2)
);


--Retriving the data from orders Table
select * From orders;

--droping the table products
Drop Table products;

--createing the table products
create table products(
product_id INT,
product_name VARCHAR(100),
category VARCHAR(100),
price DECIMAL(10,2)
);

--Retriving the data from products Table
SELECT * FROM PRODUCTS;
```

--droping the table payments

Drop Table payments;

--createing the table payments

CREATE TABLE PAYMENTS(

payment_id INT,

order_id INT REFERENCES ORDERS(ORDER_ID),

payment_method VARCHAR(100),

payment_date DATE,

amount DECIMAL(10,2)

);

--Retriving the data from payments Table

select *From payments;

--A.Use SELECT, WHERE, ORDER BY, GROUP BY

select * From customers;

output:

customer_id	name	email	country	signup_date
1	John Smith	johnsmith@example.com	Germany	01-03-2023
2	Priya Sharma	priyasharma@example.com	UK	19-04-2023
3	Ahmed Khan	ahmedkhan@example.com	UAE	07-10-2023
4	Emily Davis	emilydavis@example.com	Canada	14-07-2023
5	Raj Patel	rajpatel@example.com	UK	05-11-2023
6	Sara Ali	saraali@example.com	India	29-07-2023
7	Michael Brown	michaelbrown@example.com	Canada	29-10-2023
8	Ananya Gupta	ananyagupta@example.com	India	07-10-2023
9	Omar Farouk	omarfarouk@example.com	Canada	12-03-2023
10	David Lee	davidlee@example.com	UAE	08-06-2023
11	Sophia Wong	sophiawong@example.com	USA	26-05-2023
12	Ibrahim Noor Fatima	ibrahimnoor@example.com	USA	31-08-2023
13	Hussain	fatimahussain@example.com	UK	26-03-2023
14	Chris Evans	chrisevans@example.com	USA	28-09-2023

15	Kavya Menon	kavyamenon@example.com	Australia	31-08-2023
16	James Miller	jamesmiller@example.com	USA	19-10-2023
17	Amir Qureshi	amirqureshi@example.com	UK	31-01-2023
18	Nina Rossi	ninarossi@example.com	Canada	16-10-2023
19	Liam Wilson	liamwilson@example.com	UAE	27-04-2023
20	Isabella Garcia	isabellagarcia@example.com	Canada	10-05-2023
21	Ethan Johnson	ethanjohnson@example.com	Canada	07-10-2023
22	Olivia Martinez	oliviamartinez@example.com	UK	03-08-2023
23	William Clark	williamclark@example.com	India	30-03-2023
24	Ava Taylor	avataylor@example.com	UAE	20-10-2023
25	Mia Rodriguez	miarodriguez@example.com	Australia	03-01-2023

```
select * From products

where price>500;
```

output:

product_id	product_name	category	price
101	Laptop	Electronics	850

```
select payment_method,sum(amount)

from payments

group by payment_method

order by sum(amount) desc;
```

output:

payment_method	sum
PayPal	6110
Net Banking	3420
UPI	3065
Debit Card	1110
Credit Card	775

--B.Use JOINS (INNER, LEFT, RIGHT)

--Inner join

```
select c.name,o.total_amount
from customers c
inner join
orders o
on
c.customer_id=o.customer_id
order by total_amount;
```

output:

name	total_amount
Liam Wilson	45
Kavya Menon	45
Isabella Garcia	60
Isabella Garcia	60
Kavya Menon	70
Ava Taylor	120
Ava Taylor	140
Olivia Martinez	140
William Clark	140
Olivia Martinez	175
Olivia Martinez	200
Omar Farouk	225
John Smith	300
David Lee	300
Fatima Hussain	300
Chris Evans	300
Ethan Johnson	300
Ethan Johnson	360

Liam Wilson	375
Ethan Johnson	375
David Lee	600
Emily Davis	600
John Smith	600
Fatima Hussain	600
Sara Ali	750
Sophia Wong	750
Emily Davis	750
Chris Evans	900
Amir Qureshi	1500
Emily Davis	3400

--left join

```
select (o.customer_id),sum(p.amount)
from orders o
left join
payments p
on
o.order_id=p.order_id
group by customer_id
order by customer_id,sum(p.amount);
```

output:

customer_id	sum
1	900
4	4750
6	750
9	225
10	900
11	750
13	900
14	1200
15	115
17	1500
19	420
20	120

21	1035
22	515
23	140
24	260

```
--Right Join
select o.order_date,p.payment_date
from orders o
right join
payments p
on
o.order_id=p.order_id
where p.amount>1000;
```

output:

order_date	payment_date
07-05-2023	07-05-2023
28-04-2023	28-04-2023

--C.Write subqueries

```
select max(total_amount) as second_highest_amount
from orders
where total_amount<(
select max(total_amount)
from orders
);
```

Output:

second_highest_amount
1500

-- D.Use aggregate functions (SUM, AVG)

--sum

```
select category,sum(price)

from products

group by category

order by sum(price) desc;
```

output:

category	sum
Electronics	1895
Furniture	210
Accessories	155
Fashion	75

--average

```
select product_name,avg(price)

from products

group by product_name

order by avg(price);
```

output:

product_name	avg
Backpack	35
Headphones	45
Chair	60
Shoes	75
Watch	120
Table	150
Printer	200
Camera	300
Smartphone	500
Laptop	850

--E.Create views for analysis

```
create or replace view customer_orders as

select

c.customer_id,

c.name,

o.order_id,

o.order_date,

o.total_amount

from customers c

inner join

orders o

on

c.customer_id = o.customer_id

order by total_amount;
```

output:

CREATE VIEW

Query returned successfully in 94 msec.

```
--Retriving the data from created view

select * From customer_orders;
```

output:

customer_id	name	order_id	order_date	total_amount
19	Liam Wilson	5003	13-05-2023	45
15	Kavya Menon	5024	26-05-2023	45
20	Isabella Garcia	5013	20-05-2023	60
20	Isabella Garcia	5021	02-05-2023	60
15	Kavya Menon	5028	05-06-2023	70

24	Ava Taylor	5018	07-04-2023	120
24	Ava Taylor	5029	28-06-2023	140
22	Olivia Martinez	5004	13-04-2023	140
23	William Clark	5011	01-06-2023	140
22	Olivia Martinez	5008	01-05-2023	175
22	Olivia Martinez	5019	15-04-2023	200
9	Omar Farouk	5023	10-04-2023	225
1	John Smith	5009	29-05-2023	300
10	David Lee	5030	02-06-2023	300
13	Fatima Hussain	5027	24-04-2023	300
14	Chris Evans	5015	04-04-2023	300
21	Ethan Johnson	5005	30-06-2023	300
21	Ethan Johnson	5006	09-06-2023	360
19	Liam Wilson	5026	04-06-2023	375
21	Ethan Johnson	5025	06-06-2023	375
10	David Lee	5014	22-06-2023	600
4	Emily Davis	5017	16-06-2023	600
1	John Smith	5007	11-04-2023	600
13	Fatima Hussain	5022	20-06-2023	600
6	Sara Ali	5010	08-04-2023	750
11	Sophia Wong	5001	13-05-2023	750
4	Emily Davis	5020	18-05-2023	750
14	Chris Evans	5012	01-04-2023	900
17	Amir Qureshi	5002	07-05-2023	1500
4	Emily Davis	5016	28-04-2023	3400

--F.Optimize queries with indexes

```
CREATE INDEX idx_orders_customer_id

ON orders(customer_id);
```

Output:

```
CREATE INDEX
```

Query returned successfully in 91 msec.