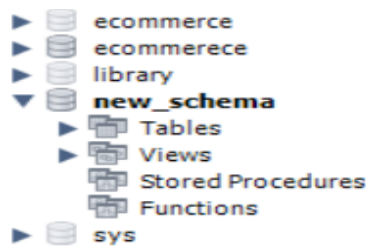


SQL QUERIES AND OUTPUTS

-- create database ecommerce

OUTPUT:



-- Create Tables and Inserting Data

-- Create Customer Table

create table customers (Customer_ID integer primary key, Name text, Email text, Country text);

select * from customers;

insert into customers values

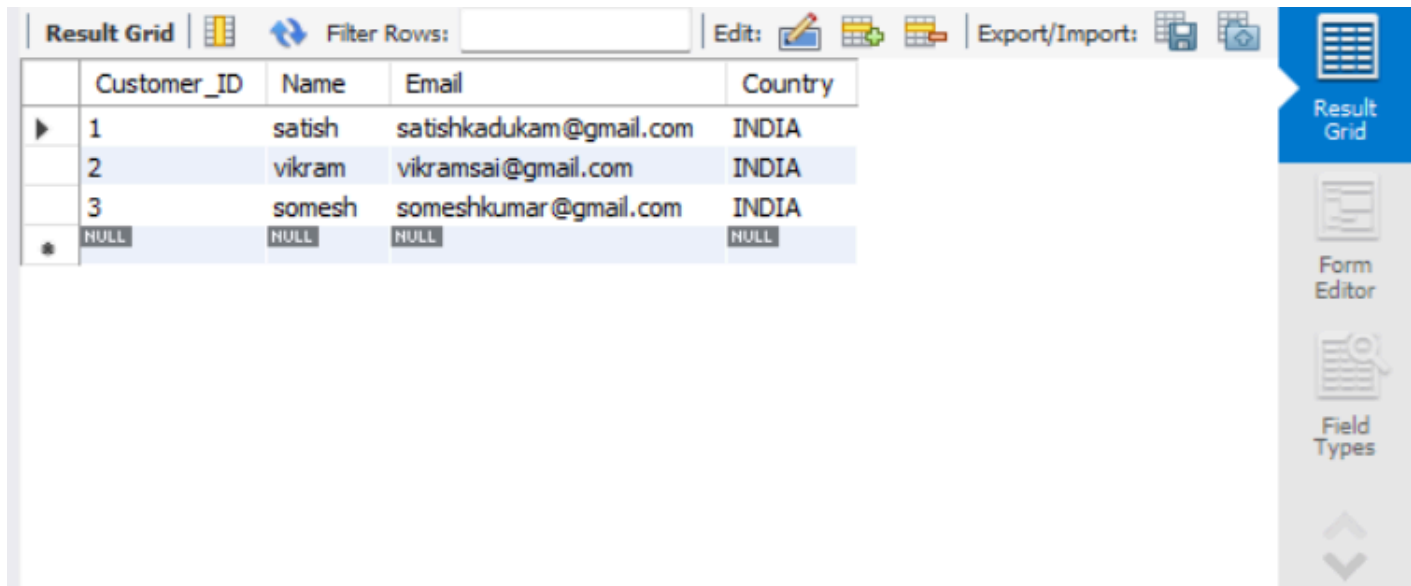
(1,'satish', 'satishkadukam@gmail.com', 'INDIA'),

(2,'vikram', 'vikramsai@gmail.com', 'INDIA'),

(3,'somesh', 'someshkumar@gmail.com', 'INDIA');

select * from customers;

OUTPUT:

A screenshot of a database management tool's 'Result Grid' window. The window displays the results of a SQL query. The table has four columns: 'Customer_ID', 'Name', 'Email', and 'Country'. There are three rows of data, corresponding to the three records inserted. The first row has values 1, satish, satishkadukam@gmail.com, and INDIA. The second row has values 2, vikram, vikramsai@gmail.com, and INDIA. The third row has values 3, somesh, someshkumar@gmail.com, and INDIA. The table is displayed in a grid format with a blue header and alternating row colors. The tool's interface includes a toolbar with icons for 'Filter Rows', 'Edit', and 'Export/Import', and a sidebar with buttons for 'Result Grid', 'Form Editor', and 'Field Types'.

Customer_ID	Name	Email	Country
1	satish	satishkadukam@gmail.com	INDIA
2	vikram	vikramsai@gmail.com	INDIA
3	somesh	someshkumar@gmail.com	INDIA

-- create Products Table

create table products (

Product_ID integer primary key, Name text, Price real);

select * from products;

insert into products values

```
(1, 'Laptop', 1200.00),
(2, 'Phone', 800.00),
(3, 'HeadPhones', 150.00);
select * from products;
```

OUTPUT:

Product_ID	Name	Price
1	Laptop	1200
2	Phone	800
3	HeadPhones	150
NULL	NULL	NULL

-- create orders table

```
create table orders (
Order_ID integer primary key,
Customer_ID integer,
Product_ID integer,
Quantity integer,
Order_Date date,
foreign key (Customer_ID) references
customers (Customer_ID),
foreign key (Product_ID) references
products (Product_ID));
```

insert into orders values

```
(1, 1, 1, 1, '2025-01-15'),
(2, 1, 3, 2, '2025-01-17'),
(3, 2, 2, 1, '2025-02-10'),
(4, 3, 1, 1, '2025-03-05');
select * from orders;
```

OUTPUT:

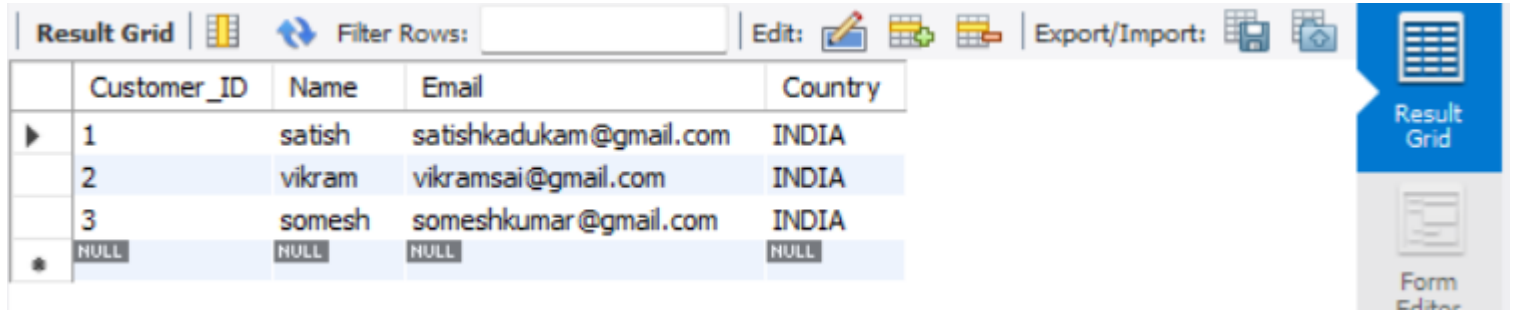
Order_ID	Customer_ID	Product_ID	Quantity	Order_Date
1	1	1	1	2025-01-15
2	1	3	2	2025-01-17
3	2	2	1	2025-02-10
4	3	1	1	2025-03-05
NULL	NULL	NULL	NULL	NULL

-- USING SELECT WHERE ORDER BY Queries

-- Get all customers from INDIA

```
select * from customers where Country = 'INDIA';
```

OUTPUT:

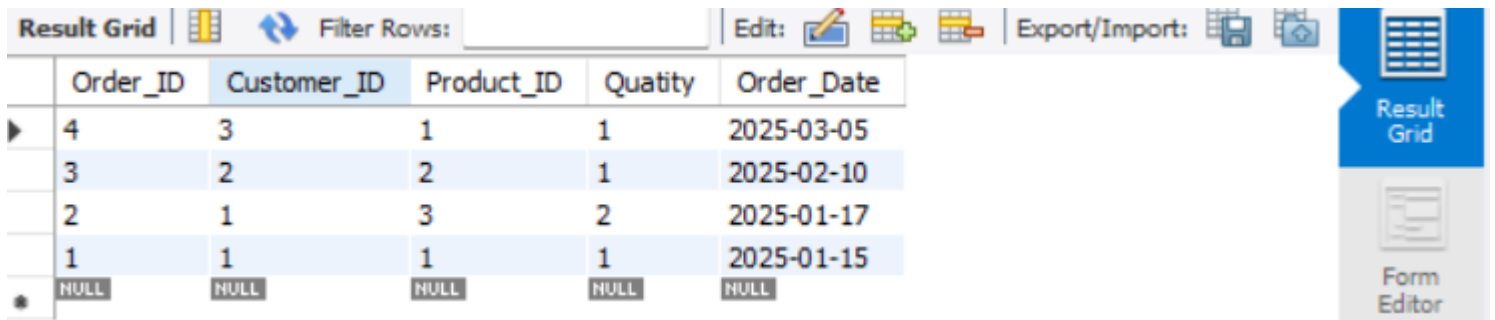


Customer_ID	Name	Email	Country
1	satish	satishkadukam@gmail.com	INDIA
2	vikram	vikramsai@gmail.com	INDIA
3	somesh	someshkumar@gmail.com	INDIA
NULL	NULL	NULL	NULL

-- Get all orders, ordered by order date (latest first)

```
select * from orders order by Order_Date desc;
```

OUTPUT:



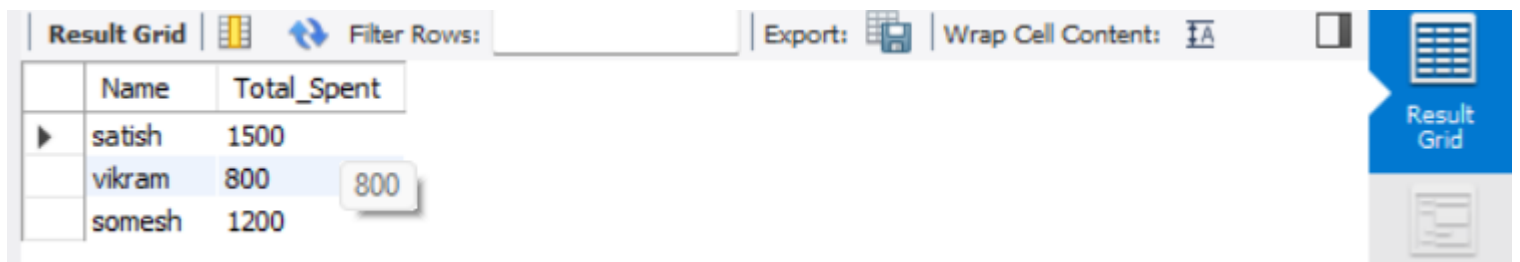
Order_ID	Customer_ID	Product_ID	Quantity	Order_Date
4	3	1	1	2025-03-05
3	2	2	1	2025-02-10
2	1	3	2	2025-01-17
1	1	1	1	2025-01-15
NULL	NULL	NULL	NULL	NULL

-- USING GROUP BY and AGGREGATE FUNCTIONS SUM AVG

-- Total Revenue per Customer

```
select c.Name, sum(p.Price * o.Quatity) as Total_Spent  
from orders o  
join customers c on o.Customer_ID = c.Customer_ID  
join products p on o.Product_ID = p.Product_ID  
group by c.Name;
```

OUTPUT:



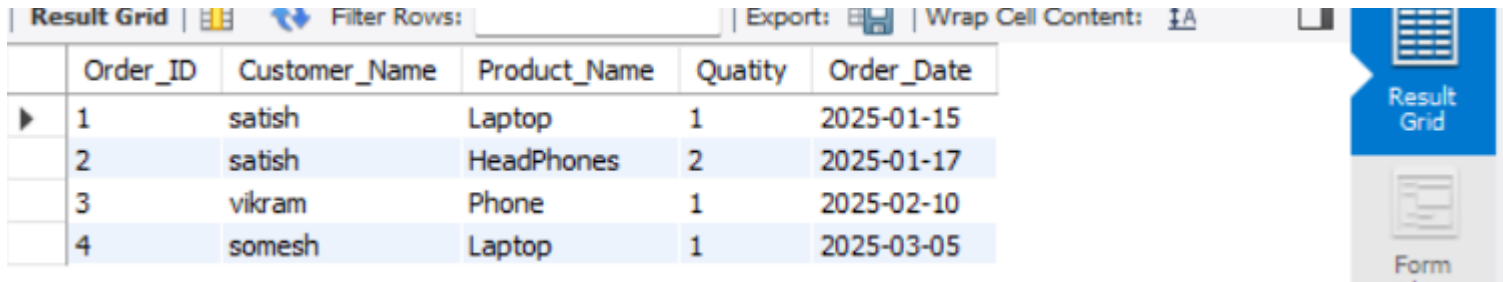
Name	Total_Spent
satish	1500
vikram	800
somesh	1200

-- USING JOINS INNER LEFT RIGHT

-- Inner Join Show all orders with customer and product details

```
select o.Order_ID, c.Name as Customer_Name,  
p.Name as Product_Name, o.Quantity, o.Order_Date from orders o  
join customers c on o.Customer_ID = c.Customer_ID  
join products p on o.Product_ID = p.Product_ID;
```

OUTPUT:



The screenshot shows a database application interface. At the top, there is a toolbar with buttons for 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'. Below the toolbar is a table with 6 columns: Order_ID, Customer_Name, Product_Name, Quantity, and Order_Date. The table contains 4 rows of data. To the right of the table is a vertical sidebar with a 'Result Grid' button and a 'Form' button.

	Order_ID	Customer_Name	Product_Name	Quantity	Order_Date
▶	1	satish	Laptop	1	2025-01-15
	2	satish	HeadPhones	2	2025-01-17
	3	vikram	Phone	1	2025-02-10
	4	somesh	Laptop	1	2025-03-05

-- Left Join Show all Customers and their orders (if any)

```
select c.Name, o.Order_ID  
from customers c  
left join orders o on c.Customer_ID = o.Customer_ID;
```

-- Note :

-- RIGHT JOIN (not supported in SQLite; for PostgreSQL/MySQL)

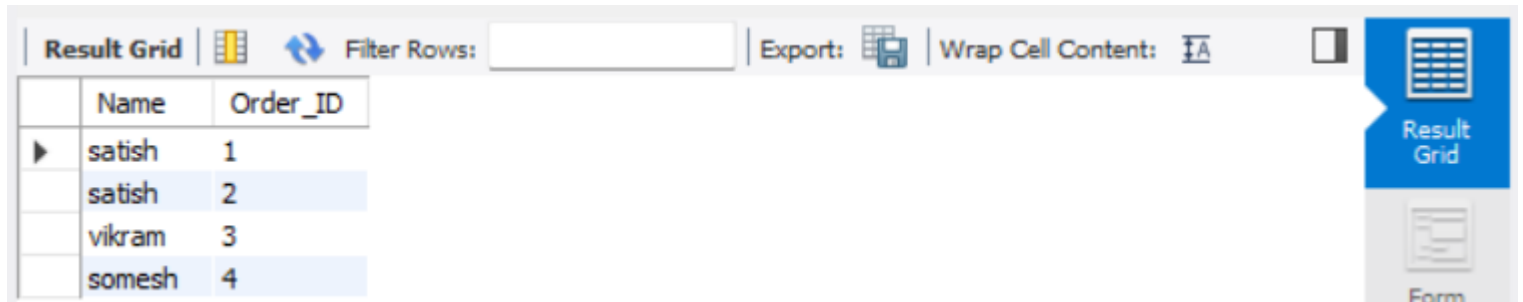
-- Show all orders with customers (same as LEFT JOIN flipped)

-- SELECT o.Order_ID, c.Name

-- FROM orders o

-- RIGHT JOIN customers c ON c.Customer_ID = o.Customer_ID;

OUTPUT:



The screenshot shows a database application interface. At the top, there is a toolbar with buttons for 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'. Below the toolbar is a table with 3 columns: Name, Order_ID, and an empty column. The table contains 4 rows of data. To the right of the table is a vertical sidebar with a 'Result Grid' button and a 'Form' button.

	Name	Order_ID
▶	satish	1
	satish	2
	vikram	3
	somesh	4

-- USING SUB-QUERIS

-- Customers who Spent more than Average total spend

```
SELECT Name  
FROM customers  
WHERE Customer_ID IN (
```

```

SELECT o.Customer_ID
FROM orders o
JOIN products p ON o.Product_ID = p.Product_ID
GROUP BY o.Customer_ID
HAVING SUM(p.Price * o.Quatity) > (
    SELECT AVG(Total_Spent)
    FROM (
        SELECT o.Customer_ID, SUM(p.Price * o.Quatity) AS Total_Spent
        FROM orders o
        JOIN products p ON o.Product_id = p.Product_id
        GROUP BY o.Customer_ID
    ) AS subquery_alias
)
);

```

OUTPUT:

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	Name
▶	satish
	somesh

Result Grid

-- USING VIEW

-- View Creation

-- Create a view for detailed order information

```

create view Order_Details as
select o.Order_ID, c.Name as Customer_Name,
p.Name as Product_Name, p.Price, o.Quatity,
(p.Price * o.Quatity) as Total_Price
from orders o
join customers c on o.Customer_ID = c.Customer_ID
join products p on o.Product_ID = p.Product_ID;

```

OUTPUT:

View: **order_details**

Columns:

Order_ID	int
Customer_Name	text
Product_Name	text
Price	double
Quantity	int
Total_Price	double

-- Query Optimization Tip

```
create index idx_Customer_ID on
orders(Customer_ID);
create index idx_Product_ID ON
orders(Product_ID);
```