1. Create the tables below in the database. Use foreign keys and primary keys as required.
   a. Create a table called as student with the following columns student_id, first_name, last_name ,birthdate , department_id ,address_id .

```sql
INSERT INTO student (student_id, first_name, last_name, birthdate, department_id, address_id) VALUES
(1, 'John', 'Doe', '1995-04-15', 1, 1),
(2, 'Jane', 'Smith', '1996-07-22', 2, 2),
(3, 'Alice', 'Johnson', '1994-11-30', 3, 3),
(4, 'Michael', 'Brown', '1997-02-19', 4, 4),
(5, 'Sophia', 'Davis', '1998-01-05', 5, 5),
(6, 'Daniel', 'Wilson', '1995-06-10', 6, 6),
(7, 'Olivia', 'Martinez', '1997-11-25', 1, 7),
(8, 'Ethan', 'Miller', '1996-03-30', 2, 8);

select * from student;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

| student_id | first_name | last_name | birthdate | department_id | address_id |
|---|---|---|---|---|---|
| 1 | John | Doe | 1995-04-15 | 1 | 1 |
| 2 | Jane | Smith | 1996-07-22 | 2 | 2 |
| 3 | Alice | Johnson | 1994-11-30 | 3 | 3 |
| 4 | Michael | Brown | 1997-02-19 | 4 | 4 |
| 5 | Sophia | Davis | 1998-01-05 | 5 | 5 |
| 6 | Daniel | Wilson | 1995-06-10 | 6 | 6 |
| 7 | Olivia | Martinez | 1997-11-25 | 1 | 7 |
| 8 | Ethan | Miller | 1996-03-30 | 2 | 8 |
| NULL | NULL | NULL | NULL | NULL | NULL |

   b. Create Address table with following columns address_id , street_address, city, State, postal_code

Limit to 1000 rows

```sql
INSERT INTO address (address_id, street_address, city, state, postal_code) VALUES
(1, '123 Elm St', 'Springfield', 'IL', '62701'),
(2, '456 Oak St', 'Decatur', 'IL', '62521'),
(3, '789 Pine St', 'Champaign', 'IL', '61820'),
(4, '102 Birch Rd', 'Peoria', 'IL', '61602'),
(5, '205 Cedar Ave', 'Chicago', 'IL', '60601'),
(6, '310 Maple Dr', 'Urbana', 'IL', '61801'),
(7, '415 Oak Blvd', 'Champaign', 'IL', '61821'),
(8, '520 Pine Rd', 'Carbondale', 'IL', '62901');
select * from address;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap C

| address_id | street_address | city | state | postal_code |
|---|---|---|---|---|
| 1 | 123 Elm St | Springfield | IL | 62701 |
| 2 | 456 Oak St | Decatur | IL | 62521 |
| 3 | 789 Pine St | Champaign | IL | 61820 |
| 4 | 102 Birch Rd | Peoria | IL | 61602 |
| 5 | 205 Cedar Ave | Chicago | IL | 60601 |
| 6 | 310 Maple Dr | Urbana | IL | 61801 |
| 7 | 415 Oak Blvd | Champaign | IL | 61821 |
| 8 | 520 Pine Rd | Carbondale | IL | 62901 |
| NULL | NULL | NULL | NULL | NULL |

c. Create department table department_id, department name. Make sure you are using the right data type against all the columns.

```
INSERT INTO department (department_id, department_name) VALUES
(1, 'Computer Science'),
(2, 'Mechanical Engineering'),
(3, 'Electrical Engineering'),
(4, 'Civil Engineering'),
(5, 'Mathematics'),
(6, 'Biology');

select * from department;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

| department_id | department_name |
|---|---|
| 1 | Computer Science |
| 2 | Mechanical Engineering |
| 3 | Electrical Engineering |
| 4 | Civil Engineering |
| 5 | Mathematics |
| 6 | Biology |
| NULL | NULL |

2. Use Sample data from [sampledata.txt]() to insert data into the database
3. Write a query to find the total number of students.

```
select count(student_id) as Total_Number_of_Students from student;
```

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

| Total_Number_of_Students |
|---|
| 8 |

4. Write a query to find which department john belongs to.

```
select d.department_name
from department d Left join student s
on s.department_id=d.department_id
where first_name="John";
```

Result Grid | Filter Rows: | Expo

| department_name |
|---|
| Computer Science |

5. List All Departments with Their Number of Students (Including Departments with No Students)

```sql
SELECT d.department_name, COUNT(s.student_id) AS student_count
FROM department d
LEFT JOIN student s ON d.department_id = s.department_id
GROUP BY d.department_name;
```

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

| department_name | student_count |
|---|---|
| Computer Science | 2 |
| Mechanical Engineering | 2 |
| Electrical Engineering | 1 |
| Civil Engineering | 1 |
| Mathematics | 1 |
| Biology | 1 |

6. Select all students with their department and address.

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

| student_id | first_name | last_name | birthdate | department_id | address_id | department_name | city | street_address |
|---|---|---|---|---|---|---|---|---|
| 1 | John | Doe | 1995-04-15 | 1 | 1 | Computer Science | Springfield | 123 Elm St |
| 3 | Alice | Johnson | 1994-11-30 | 3 | 3 | Electrical Engineering | Champaign | 789 Pine St |
| 4 | Michael | Brown | 1997-02-19 | 4 | 4 | Civil Engineering | Peoria | 102 Birch Rd |
| 5 | Sophia | Davis | 1998-01-05 | 5 | 5 | Mathematics | Chicago | 205 Cedar Ave |
| 7 | Olivia | Martinez | 1997-11-25 | 1 | 7 | Computer Science | Champaign | 415 Oak Blvd |

7. Find all students who are in the 'Computer Science' department

```
select s.*
from student s Right join department d on s.department_id=d.department_id
where d.department_name="Computer Science";
```

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

| student_id | first_name | last_name | birthdate | department_id | address_id |
|---|---|---|---|---|---|
| 1 | John | Doe | 1995-04-15 | 1 | 1 |
| 7 | Olivia | Martinez | 1997-11-25 | 1 | 7 |

8. Update Jane's city name to New York.

```
UPDATE address
SET city = 'New York'
WHERE address_id = (
    SELECT address_id
    FROM student
    WHERE first_name = 'Jane' AND last_name = 'Smith'
);

select city from address;
```

esult Grid | Filter Rows: | Export: | Wrap Cell C

| city |
|---|
| Springfield |
| New York |
| Champaign |
| Peoria |
| Chicago |
| Urbana |
| Champaign |
| Carbondale |

9. Delete a student from the student table.

```
--
DELETE FROM student
WHERE student_id = 1;
select * from student;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

| student_id | first_name | last_name | birthdate | department_id | address_id |
|---|---|---|---|---|---|
| 2 | Jane | Smith | 1996-07-22 | 2 | 2 |
| 3 | Alice | Johnson | 1994-11-30 | 3 | 3 |
| 4 | Michael | Brown | 1997-02-19 | 4 | 4 |
| 5 | Sophia | Davis | 1998-01-05 | 5 | 5 |
| 6 | Daniel | Wilson | 1995-06-10 | 6 | 6 |
| 7 | Olivia | Martinez | 1997-11-25 | 1 | 7 |
| 8 | Ethan | Miller | 1996-03-30 | 2 | 8 |
| NULL | NULL | NULL | NULL | NULL | NULL |

10. Select all students with their department and address in New York.

```
--
SELECT s.*, d.department_name, a.*
FROM student s
JOIN department d ON s.department_id = d.department_id
JOIN address a ON s.address_id = a.address_id
WHERE a.city = 'New York';
```

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

| student_id | first_name | last_name | birthdate | department_id | address_id | department_name | address_id | street_address | city |
|---|---|---|---|---|---|---|---|---|---|
| 2 | Jane | Smith | 1996-07-22 | 2 | 2 | Mechanical Engineering | 2 | 456 Oak St | New |

11. Count how many students are in each department

```sql
--
SELECT d.department_name, COUNT(s.student_id) AS student_count
FROM department d
LEFT JOIN student s ON d.department_id = s.department_id
GROUP BY d.department_name;
```

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

| department_name | student_count |
|---|---|
| Computer Science | 1 |
| Mechanical Engineering | 2 |
| Electrical Engineering | 1 |
| Civil Engineering | 1 |
| Mathematics | 1 |
| Biology | 1 |

12. Find students who live in 'Springfield'

```sql
--
SELECT s.*
FROM student s
JOIN address a ON s.address_id = a.address_id
WHERE a.city = 'Springfield';
```

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

| student_id | first_name | last_name | birthdate | department_id | address_id |
|---|---|---|---|---|---|
| 1 | John | Doe | 1995-04-15 | 1 | 1 |

13. Select students whose birthday falls in February

```
SELECT *
FROM student
WHERE MONTH(birthdate) = 2;
```

Result Grid | Filter Rows: | Edit: | Export/Impo

| student_id | first_name | last_name | birthdate | department_id | address_id |
|---|---|---|---|---|---|
| 4 | Michael | Brown | 1997-02-19 | 4 | 4 |
| NULL | NULL | NULL | NULL | NULL | NULL |

14. Get the department and address details for a specific student, example john

```
SELECT s.*, d.department_name, a.*
FROM student s
JOIN department d ON s.department_id = d.department_id
JOIN address a ON s.address_id = a.address_id
WHERE s.first_name = 'John';
```

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

| student_id | first_name | last_name | birthdate | department_id | address_id | department_name | address_id | street_address | city |
|---|---|---|---|---|---|---|---|---|---|
| 1 | John | Doe | 1995-04-15 | 1 | 1 | Computer Science | 1 | 123 Elm St | Springfiel |

15. Find all students who are born within 1995 to 1998

```
SELECT *
FROM student
WHERE YEAR(birthdate) BETWEEN 1995 AND 1998;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Co

| student_id | first_name | last_name | birthdate | department_id | address_id |
|---|---|---|---|---|---|
| 1 | John | Doe | 1995-04-15 | 1 | 1 |
| 2 | Jane | Smith | 1996-07-22 | 2 | 2 |
| 4 | Michael | Brown | 1997-02-19 | 4 | 4 |
| 5 | Sophia | Davis | 1998-01-05 | 5 | 5 |
| 6 | Daniel | Wilson | 1995-06-10 | 6 | 6 |
| 7 | Olivia | Martinez | 1997-11-25 | 1 | 7 |
| 8 | Ethan | Miller | 1996-03-30 | 2 | 8 |
| NULL | NULL | NULL | NULL | NULL | NULL |

16. List all students and their corresponding department names, sorted by department

```
--
SELECT s.*, d.department_name
FROM student s
JOIN department d ON s.department_id = d.department_id
ORDER BY d.department_name;
```

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

| student_id | first_name | last_name | birthdate | department_id | address_id | department_name |
|---|---|---|---|---|---|---|
| 6 | Daniel | Wilson | 1995-06-10 | 6 | 6 | Biology |
| 4 | Michael | Brown | 1997-02-19 | 4 | 4 | Civil Engineering |
| 1 | John | Doe | 1995-04-15 | 1 | 1 | Computer Science |
| 7 | Olivia | Martinez | 1997-11-25 | 1 | 7 | Computer Science |
| 3 | Alice | Johnson | 1994-11-30 | 3 | 3 | Electrical Engineering |
| 5 | Sophia | Davis | 1998-01-05 | 5 | 5 | Mathematics |
| 2 | Jane | Smith | 1996-07-22 | 2 | 2 | Mechanical Engineering |
| 8 | Ethan | Miller | 1996-03-30 | 2 | 8 | Mechanical Engineering |

17. Find the number of students in each department who are living in 'Champaign'

```
--
SELECT d.department_name, COUNT(s.student_id) AS student_count
FROM student s
JOIN department d ON s.department_id = d.department_id
JOIN address a ON s.address_id = a.address_id
WHERE a.city = 'Champaign'
GROUP BY d.department_name;
```

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

| department_name | student_count |
|---|---|
| Electrical Engineering | 1 |
| Computer Science | 1 |

18. Retrieve the names of students who live on 'Pine' street

```
SELECT s.first_name, s.last_name
FROM student s
JOIN address a ON s.address_id = a.address_id
WHERE a.street_address LIKE '%Pine%';
```

Result Grid | Filter Rows: | Export:

| first_name | last_name |
|---|---|
| Alice | Johnson |
| Ethan | Miller |

19. Update the department of a student with student_id = 6 to 'Mechanical Engineering'

```
UPDATE student
SET department_id = (
    SELECT department_id FROM department WHERE department_name = 'Mechanical Engineering'
)
WHERE student_id = 6;
select * from student;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Con

| student_id | first_name | last_name | birthdate | department_id | address_id |
|---|---|---|---|---|---|
| 1 | John | Doe | 1995-04-15 | 1 | 1 |
| 2 | Jane | Smith | 1996-07-22 | 2 | 2 |
| 3 | Alice | Johnson | 1994-11-30 | 3 | 3 |
| 4 | Michael | Brown | 1997-02-19 | 4 | 4 |
| 5 | Sophia | Davis | 1998-01-05 | 5 | 5 |
| 6 | Daniel | Wilson | 1995-06-10 | 2 | 6 |
| 7 | Olivia | Martinez | 1997-11-25 | 1 | 7 |
| 8 | Ethan | Miller | 1996-03-30 | 2 | 8 |
| NULL | NULL | NULL | NULL | NULL | NULL |

20. Find the student(s) who live in the city 'Chicago' and are in the 'Mathematics' department

```
SELECT s.*
FROM student s
JOIN department d ON s.department_id = d.department_id
JOIN address a ON s.address_id = a.address_id
WHERE a.city = 'Chicago' AND d.department_name = 'Mathematics';
```

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

| student_id | first_name | last_name | birthdate | department_id | address_id |
|---|---|---|---|---|---|
| 5 | Sophia | Davis | 1998-01-05 | 5 | 5 |

21. List all students who have an address in 'Urbana' or 'Peoria'

```sql
SELECT s.*
FROM student s
JOIN address a ON s.address_id = a.address_id
WHERE a.city IN ('Urbana', 'Peoria');
```

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

| student_id | first_name | last_name | birthdate | department_id | address_id |
|---|---|---|---|---|---|
| 4 | Michael | Brown | 1997-02-19 | 4 | 4 |
| 6 | Daniel | Wilson | 1995-06-10 | 2 | 6 |

22. Find the student with the highest student_id

```sql
SELECT *
FROM student
ORDER BY student_id DESC
LIMIT 1;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

| student_id | first_name | last_name | birthdate | department_id | address_id |
|---|---|---|---|---|---|
| 8 | Ethan | Miller | 1996-03-30 | 2 | 8 |
| NULL | NULL | NULL | NULL | NULL | NULL |

23. Find all students who are not in the 'Computer Science' department

```sql
SELECT s.*
FROM student s
JOIN department d ON s.department_id = d.department_id
WHERE d.department_name != 'Computer Science';
```

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

| student_id | first_name | last_name | birthdate | department_id | address_id |
|---|---|---|---|---|---|
| 2 | Jane | Smith | 1996-07-22 | 2 | 2 |
| 6 | Daniel | Wilson | 1995-06-10 | 2 | 6 |
| 8 | Ethan | Miller | 1996-03-30 | 2 | 8 |
| 3 | Alice | Johnson | 1994-11-30 | 3 | 3 |
| 4 | Michael | Brown | 1997-02-19 | 4 | 4 |
| 5 | Sophia | Davis | 1998-01-05 | 5 | 5 |

24. Count the total number of addresses in the 'Champaign' city

```
--
SELECT COUNT(*) AS total_addresses
FROM address
WHERE city = 'Champaign';
```

Result Grid | Filter Rows: | Export:

| total_addresses |
|---|
| 2 |

25. Find the name of the student who lives at '520 Pine Rd'

```
SELECT s.first_name, s.last_name
FROM student s
JOIN address a ON s.address_id = a.address_id
WHERE a.street_address = '520 Pine Rd';
```

Result Grid | Filter Rows: | Export:

| first_name | last_name |
|---|---|
| Ethan | Miller |

26. Get the average age of students in the 'Electrical Engineering' department

```
--
SELECT AVG(TIMESTAMPDIFF(YEAR, birthdate, CURDATE())) AS average_age
FROM student s
JOIN department d ON s.department_id = d.department_id
WHERE d.department_name = 'Electrical Engineering';
```

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

| average_age |
|---|
| 30.0000 |

27. List the students, their department, and the city where they live, but only for those in departments starting with 'M'

```sql
SELECT s.first_name, s.last_name, d.department_name, a.city
FROM student s
JOIN department d ON s.department_id = d.department_id
JOIN address a ON s.address_id = a.address_id
WHERE d.department_name LIKE 'M%';
```

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

| first_name | last_name | department_name | city |
|---|---|---|---|
| Jane | Smith | Mechanical Engineering | New York |
| Daniel | Wilson | Mechanical Engineering | Urbana |
| Ethan | Miller | Mechanical Engineering | Carbondale |
| Sophia | Davis | Mathematics | Chicago |

28. Delete a student from the 'Mechanical Engineering' department

```sql
DELETE FROM student
WHERE department_id = (
    SELECT department_id FROM department WHERE department_name = 'Mechanical Engineering'
);

select * from student;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

| student_id | first_name | last_name | birthdate | department_id | address_id |
|---|---|---|---|---|---|
| 1 | John | Doe | 1995-04-15 | 1 | 1 |
| 3 | Alice | Johnson | 1994-11-30 | 3 | 3 |
| 4 | Michael | Brown | 1997-02-19 | 4 | 4 |
| 5 | Sophia | Davis | 1998-01-05 | 5 | 5 |
| 7 | Olivia | Martinez | 1997-11-25 | 1 | 7 |
| NULL | NULL | NULL | NULL | NULL | NULL |

Download order.sql

Open PG Admin and open query tool and select any database of your choice.

Click on “Open file” and select order.sql from your device and execute it.

Questions:

1. Retrieve All Orders with Their Customer Details and Current Status

```sql
--
SELECT
    o.order_id,
    c.customer_id,
    c.first_name,
    c.last_name,
    c.email,
    s.status_name,
    o.order_date,
    o.total_amount
FROM order_schema.orders o
JOIN order_schema.customer c ON o.customer_id = c.customer_id
JOIN order_schema.status s ON o.status_id = s.status_id;
```

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

| order_id | customer_id | first_name | last_name | email | status_name | order_date | total_amount |
|---|---|---|---|---|---|---|---|
| 1 | 1 | John | Doe | john.doe@example.com | Shipped | 2025-02-15 | 1499.98 |
| 4 | 1 | John | Doe | john.doe@example.com | Cancelled | 2025-02-18 | 149.99 |
| 2 | 2 | Jane | Smith | jane.smith@example.com | Pending | 2025-02-16 | 199.99 |
| 3 | 3 | Emily | Jones | emily.jones@example.com | Shipped | 2025-02-17 | 499.99 |

2. Get the Total Value of Orders for a Given Customer in a Specific Time Period

```sql
--
SELECT
    c.customer_id,
    c.first_name,
    c.last_name,
    SUM(o.total_amount) AS total_order_value
FROM order_schema.orders o
JOIN order_schema.customer c ON o.customer_id = c.customer_id
WHERE o.order_date BETWEEN '2025-02-15' AND '2025-02-18'
  AND c.customer_id = 1  -- Replace with desired customer ID
GROUP BY c.customer_id, c.first_name, c.last_name;
```

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

| customer_id | first_name | last_name | total_order_value |
|---|---|---|---|
| 1 | John | Doe | 1649.97 |

3. Find the Most Expensive Order by Customer

```sql
SELECT customer_id, order_id, total_amount
FROM order_schema.orders
WHERE (customer_id, total_amount) IN (
    SELECT customer_id, MAX(total_amount)
    FROM order_schema.orders
    GROUP BY customer_id
);
```

Result Grid | Filter Rows: | Edit:

| customer_id | order_id | total_amount |
|---|---|---|
| 1 | 1 | 1499.98 |
| 2 | 2 | 199.99 |
| 3 | 3 | 499.99 |
| NULL | NULL | NULL |

4. Find the Total Revenue for Each Product Based on Orders

```sql
SELECT
    p.product_id,
    p.product_name,
    SUM(oi.quantity * oi.price) AS total_revenue
FROM order_schema.order_items oi
JOIN order_schema.product p ON oi.product_id = p.product_id
GROUP BY p.product_id, p.product_name;
```

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

| product_id | product_name | total_revenue |
|---|---|---|
| 1 | Laptop | 999.99 |
| 2 | Smartphone | 999.98 |
| 3 | Headphones | 449.97 |

5. Write a query to retrieve the order ID, customer ID, and the total amount of each order. If the total amount is null, display '0.00' instead.

```sql
--
SELECT
    order_id,
    customer_id,
    COALESCE(total_amount, 0.00) AS total_amount
FROM order_schema.orders;
```

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

| order_id | customer_id | total_amount |
|---|---|---|
| 1 | 1 | 1499.98 |
| 2 | 2 | 199.99 |
| 3 | 3 | 499.99 |
| 4 | 1 | 149.99 |

6. Retrieve the Order History of a Specific Customer Along with Product Details

```sql
SELECT
    oh.order_id,
    oh.status_change_date,
    oh.status_description,
    p.product_name,
    oi.quantity,
    oi.price
FROM order_schema.order_history oh
JOIN order_schema.orders o ON oh.order_id = o.order_id
JOIN order_schema.customer c ON o.customer_id = c.customer_id
JOIN order_schema.order_items oi ON o.order_id = oi.order_id
JOIN order_schema.product p ON oi.product_id = p.product_id
WHERE c.customer_id = 1
```

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

| order_id | status_change_date | status_description | product_name | quantity | price |
|---|---|---|---|---|---|
| 1 | 2025-02-15 | Order Placed | Laptop | 1 | 999.99 |
| 1 | 2025-02-15 | Order Placed | Smartphone | 1 | 499.99 |
| 1 | 2025-02-16 | Payment Processed | Laptop | 1 | 999.99 |
| 1 | 2025-02-16 | Payment Processed | Smartphone | 1 | 499.99 |
| 4 | 2025-02-18 | Order Placed | Headphones | 1 | 149.99 |

7. Get the Average Order Value Per Customer in the Last 30 Days.

```sql
SELECT c.customer_id, c.first_name, c.last_name,
       AVG(o.total_amount) AS avg_order_value
FROM order_schema.customer c
JOIN order_schema.orders o ON c.customer_id = o.customer_id
WHERE o.order_date >= CURDATE() - INTERVAL 30 DAY
GROUP BY c.customer_id, c.first_name, c.last_name;
```

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

| customer_id | first_name | last_name | avg_order_value |
|---|---|---|---|

8. Get the Top 5 Products with the Highest Number of Orders.

```sql
SELECT
    p.product_id,
    p.product_name,
    COUNT(DISTINCT oi.order_id) AS number_of_orders
FROM order_schema.order_items oi
JOIN order_schema.product p ON oi.product_id = p.product_id
GROUP BY p.product_id, p.product_name
ORDER BY number_of_orders DESC
LIMIT 5;
```

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

| product_id | product_name | number_of_orders |
|---|---|---|
| 2 | Smartphone | 2 |
| 3 | Headphones | 2 |
| 1 | Laptop | 1 |

9. Get the Customers Who Have Not Placed Any Orders in the Last 60 Days

```sql
SELECT c.customer_id, c.first_name, c.last_name
FROM order_schema.customer c
LEFT JOIN order_schema.orders o
  ON c.customer_id = o.customer_id
  AND o.order_date >= CURDATE() - INTERVAL 60 DAY
WHERE o.order_id IS NULL;
```

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

| customer_id | first_name | last_name |
|---|---|---|
| 1 | John | Doe |
| 2 | Jane | Smith |
| 3 | Emily | Jones |

10. List the Orders with Products Ordered More Than Once, Sorted by Order Date

```sql
);

SELECT
    o.order_id,
    o.order_date,
    p.product_name,
    oi.quantity
FROM order_schema.order_items oi
JOIN order_schema.orders o ON oi.order_id = o.order_id
JOIN order_schema.product p ON oi.product_id = p.product_id
WHERE oi.quantity > 1
ORDER BY o.order_date;
```

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

| order_id | order_date | product_name | quantity |
|---|---|---|---|
| 2 | 2025-02-16 | Headphones | 2 |

11. Retrieve the Number of Orders and Total Revenue for Each Status

```sql
SELECT s.status_name, COUNT(o.order_id) AS order_count, SUM(o.total_amount) AS total_revenue
FROM order_schema.orders o
JOIN order_schema.status s ON o.status_id = s.status_id
GROUP BY s.status_name;
```

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

| status_name | order_count | total_revenue |
|---|---|---|
| Shipped | 2 | 1999.97 |
| Pending | 1 | 199.99 |
| Cancelled | 1 | 149.99 |

12. Find Customers Who Have Ordered More Than a Specific Product (e.g., "Laptop")

```sql
SELECT DISTINCT c.customer_id, c.first_name, c.last_name
FROM order_schema.customer c
JOIN order_schema.orders o ON c.customer_id = o.customer_id
JOIN order_schema.order_items oi ON o.order_id = oi.order_id
JOIN order_schema.product p ON oi.product_id = p.product_id
WHERE p.product_name = 'Laptop';
```

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

| customer_id | first_name | last_name |
|---|---|---|
| 1 | John | Doe |

13. Find the Products That Have Never Been Ordered

```sql
SELECT p.*
FROM order_schema.product p
LEFT JOIN order_schema.order_items oi ON p.product_id = oi.product_id
WHERE oi.product_id IS NULL;
```

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

| product_id | product_name | price | stock_quantity |
|---|---|---|---|
| 4 | Monitor | 199.99 | 75 |

14. Get the Total Quantity of Products Ordered in the Last 7 Days

```sql
SELECT p.product_name, SUM(oi.quantity) AS total_quantity
FROM order_schema.order_items oi
JOIN order_schema.orders o ON oi.order_id = o.order_id
JOIN order_schema.product p ON oi.product_id = p.product_id
WHERE o.order_date >= CURDATE() - INTERVAL 7 DAY
GROUP BY p.product_name;
```

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

| product_name | total_quantity |
|---|---|

15. Create a view named product_details that includes all columns from the product table.

```sql
CREATE VIEW order_schema.product_details AS
SELECT * FROM order_schema.product;
```

16. Create a view named order_summary that includes the order_id, customer_id, order_date, total_amount, and status_name (from the status table) for each order.

```sql
CREATE VIEW order_schema.order_summary AS
SELECT o.order_id, o.customer_id, o.order_date, o.total_amount, s.status_name
FROM order_schema.orders o
JOIN order_schema.status s ON o.status_id = s.status_id;
```