Manav Shah

Q 1:

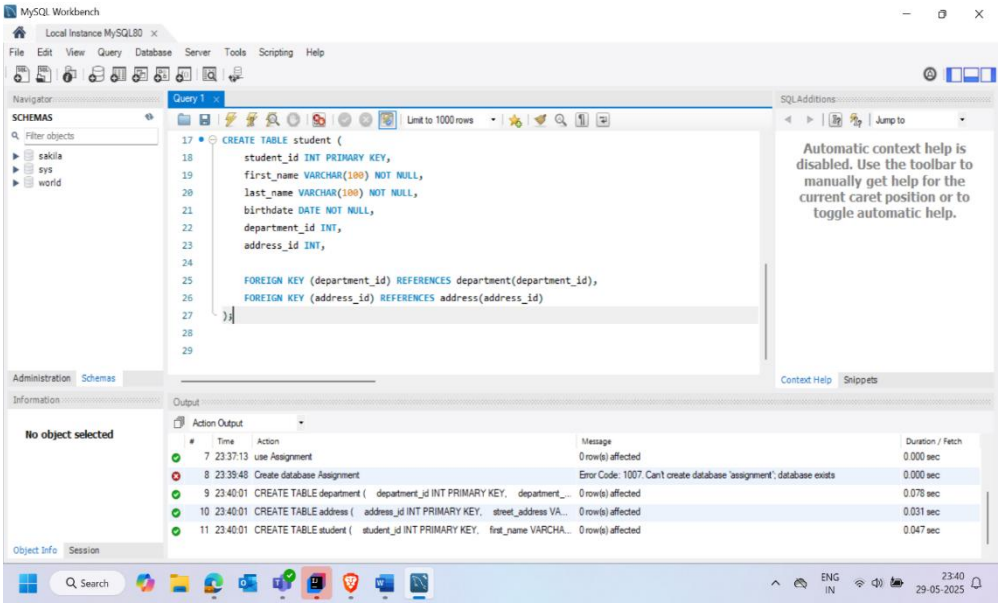
1. **Create the tables below in the database. Use foreign keys and primary keys as required.**

a. **Create a table called as student with the following columns student_id, first_name, last_name ,birthdate , department_id ,address_id .**

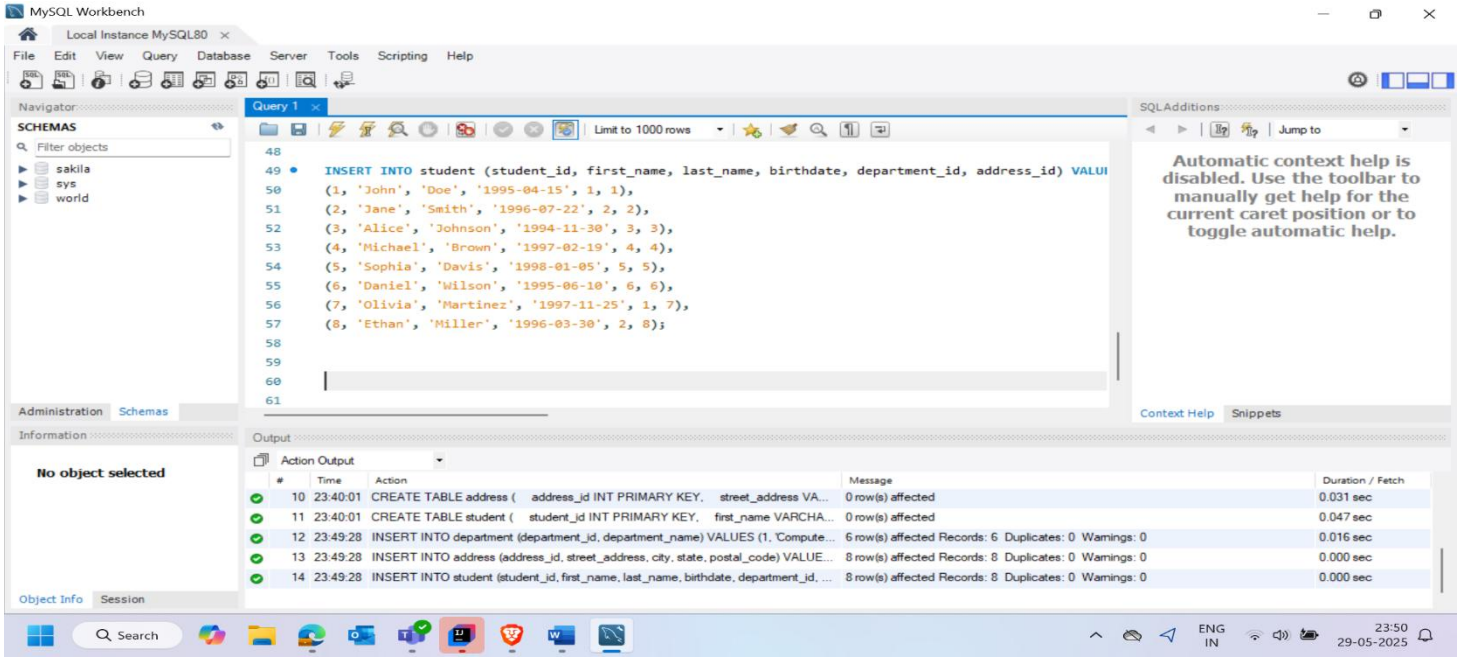
b. **Create Address table with following columns address_id , street_address, city, State, postal_code**

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

```
Create database Assignment;
use Assignment;

CREATE TABLE department (
  department_id INT PRIMARY KEY,
  department_name VARCHAR(100) NOT NULL
);

CREATE TABLE address (
  address_id INT PRIMARY KEY,
  street_address VARCHAR(150),
  city VARCHAR(100),
  state VARCHAR(50),
  postal_code VARCHAR(20)
);

CREATE TABLE student (
  student_id INT PRIMARY KEY,
  first_name VARCHAR(100) NOT NULL,
  last_name VARCHAR(100) NOT NULL,
  birthdate DATE NOT NULL,
  department_id INT,
  address_id INT,

  FOREIGN KEY (department_id) REFERENCES department(department_id),
  FOREIGN KEY (address_id) REFERENCES address(address_id)
);
```



2:



Write a query to find the total number of students.

Write a query to find which department john belongs to.

```
SELECT d.department_name
FROM student s
JOIN department d ON s.department_id = d.department_id
WHERE s.first_name = 'John';

SELECT d.department_name, COUNT(s.student_id) AS student_count
```

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

department_name
Computer Science

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

```
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;
```

ult 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 departm

```
SELECT s.first_name, s.last_name, d.department_name, a.street_address, 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;

SELECT s.*
```

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

first_name	last_name	department_name	street_address	city
John	Doe	Computer Science	123 Elm St	Springfield
Olivia	Martinez	Computer Science	415 Oak Blvd	Champaign
Jane	Smith	Mechanical Engineering	456 Oak St	Decatur
Ethan	Miller	Mechanical Engineering	520 Pine Rd	Carbondale
Alice	Johnson	Electrical Engineering	789 Pine St	Champaign
Michael	Brown	Civil Engineering	102 Birch Rd	Peoria
Sophia	Davis	Mathematics	205 Cedar Ave	Chicago
Daniel	Wilson	Biology	310 Maple Dr	Urbana

7.

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

```
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
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' LIMIT 1
);
```

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

9. Delete a student from the student table.

```
DELETE FROM student
WHERE student_id = 8;  -- Replace with desired student_id
```

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

```
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 a.city = 'New York';
```

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

first_name	last_name	department_name	city

11. Count how many students are in each department

```
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	1
Electrical Engineering	1
Civil Engineering	1
Mathematics	1
Biology	1

12. Find students who live in 'Springfield'

```
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/Import: | Wrap Cell Content:

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 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:

department_name	address_id	street_address	city	state	postal_code
Computer Science	1	123 Elm St	Springfield	IL	62701

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

```
SELECT * FROM student
WHERE birthdate BETWEEN '1995-01-01' AND '1998-12-31';
```

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

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
NULL	NULL	NULL	NULL	NULL	NULL

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

```
SELECT s.first_name, s.last_name, 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:

first_name	last_name	department_name
Daniel	Wilson	Biology
Michael	Brown	Civil Engineering
John	Doe	Computer Science
Olivia	Martinez	Computer Science
Alice	Johnson	Electrical Engineering
Sophia	Davis	Mathematics
Jane	Smith	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 Content:

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: | Wrap Cell Content:

first_name	last_name
Alice	Johnson

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;
```

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 d.department_name = 'Mathematics' AND a.city = 'Chicago';
```

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'

```
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

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

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

student_id	first_name	last_name	birthdate	department_id	address_id
7	Olivia	Martinez	1997-11-25	1	7
NULL	NULL	NULL	NULL	NULL	NULL

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

```
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
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 champaign_addresses
FROM address
WHERE city = 'Champaign';
```

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

champaign_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: | Wrap Cell Content:

first_name	last_name

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

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

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

avg_age
30.0000

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

```
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	Decatur
Daniel	Wilson	Mechanical Engineering	Urbana
Sophia	Davis	Mathematics	Chicago

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

```
DELETE FROM student
WHERE department_id = (
    SELECT department_id FROM department
    WHERE department_name = 'Mechanical Engineering'
)
LIMIT 1;
```

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, o.order_date, o.total_amount,
   c.first_name, c.last_name, c.email,
   s.status_name
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	order_date	total_amount	first_name	last_name	email	status_name
1	2025-02-15	1499.98	John	Doe	[illegible]	Shipped
4	2025-02-18	149.99	John	Doe	[illegible]	[illegible]celled
2	2025-02-16	199.99	Jane	Smith	jane.smith@example.com	Pending
3	2025-02-17	499.99	Emily	Jones	emily.jones@example.com	Shipped

Export recordset to an external file

Result 1

Output

Action Output

#	Time	Action	Message
56	19:51:56	CREATE TABLE order_schema.order_items (order_item_id INT PRIMARY KEY, order_id INT, product_id INT, quantity IN...	0 row(s) affected
57	19:51:56	INSERT INTO order_schema.order_items (order_item_id, order_id, product_id, quantity, price) VALUES (1, 1, 1, 1, 999.99), -- 1 L...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0

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

```sql
SELECT SUM(total_amount) AS total_value
FROM order_schema.orders
WHERE customer_id = 1
  AND order_date BETWEEN '2025-02-01' AND '2025-02-28';
```

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

total_value
1649.97

3. Find the Most Expensive Order by Customer

```
SELECT customer_id, order_id, total_amount
FROM order_schema.orders o
WHERE total_amount = (
    SELECT MAX(total_amount)
    FROM order_schema.orders o2
    WHERE o2.customer_id = o.customer_id
);
```

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

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

orders 3

Output

Action Output

#	Time	Action	Message
61	19:54:00	SELECT SUM(total_amount) AS total_value FROM order_schema.orders WHERE customer_id = 1 AND order_date BETWEEN '2...	1 row(s) returned

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

```
SELECT 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_name;
```

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

product_name	total_revenue
Laptop	999.99
Smartphone	999.98
Headphones	449.97

Result 4

Output

Action Output

#	Time	Action	Message
62	19:54:23	SELECT customer_id, order_id, total_amount FROM order_schema.orders o WHERE total_amount = (SELECT MAX(total_amoun...	3 row(s) returned

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.

```
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.order_items oi ON o.order_id = oi.order_id
JOIN order_schema.product p ON oi.product_id = p.product_id
WHERE o.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	2025-02-16	Payment Processed	Laptop	1	999.99
1	2025-02-15	Order Placed	Smartphone	1	499.99
1	2025-02-16	Payment Processed	Smartphone	1	499.99
4	2025-02-18	Order Placed	Headphones	1	149.99

Result 8

Output

Action Output

#	Time	Action	Message
66	19:57:09	SELECT p.product_name, SUM(oi.quantity * oi.price) AS total_revenue FROM order_schema.order_items oi JOIN order_schema.pro...	3 row(s) returned
67	19:57:26	SELECT oh.order_id, oh.status_change_date, oh.status_description, p.product_name, oi.quantity, oi.price FROM order_schema...	5 row(s) returned

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 average_order_value
FROM order_schema.customer c
JOIN order_schema.orders o ON c.customer_id = o.customer_id
WHERE o.order_date >= CURRENT_DATE - 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	average_order_value

Result 9

Output

Action Output

#	Time	Action	Message
67	19:57:26	SELECT oh.order_id, oh.status_change_date, oh.status_description, p.product_name, oi.quantity, oi.price FROM order_schema...	5 row(s) returned
68	19:57:43	SELECT c.customer_id, c.first_name, c.last_name, AVG(o.total_amount) AS average_order_value FROM order_schema.custom...	0 row(s) returned

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

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

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

product_name	order_count
Smartphone	2
Headphones	2
Laptop	1

Result 10

Output

Action Output

#	Time	Action	Message
68	19:57:43	SELECT c.customer_id, c.first_name, c.last_name, AVG(o.total_amount) AS average_order_value FROM order_schema.custom...	0 row(s) returned
69	19:58:09	SELECT p.product_name, COUNT(oi.order_id) AS order_count FROM order_schema.order_items oi JOIN order_schema.product p O...	3 row(s) returned

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

```sql
SELECT c.*
FROM order_schema.customer c
WHERE c.customer_id NOT IN (
    SELECT customer_id
    FROM order_schema.orders
    WHERE order_date >= CURRENT_DATE - INTERVAL 60 DAY
);
```

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

customer_id	first_name	last_name	email	phone_number
1	John	Doe	john.doe@example.com	555-1234
2	Jane	Smith	jane.smith@example.com	555-5678
3	Emily	Jones	emily.jones@example.com	555-8765
NULL	NULL	NULL	NULL	NULL

customer 11

Output

Action Output

#	Time	Action	Message
69	19:58:09	SELECT p.product_name, COUNT(oi.order_id) AS order_count FROM order_schema.order_items oi JOIN order_schema.product p O...	3 row(s) returned
70	19:58:43	SELECT c.* FROM order_schema.customer c WHERE c.customer_id NOT IN (SELECT customer_id FROM order_schema.ord...	3 row(s) returned

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.orders o
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 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

Result 12

Output

Action Output

#	Time	Action	Message
70	19:58:43	SELECT c.* FROM order_schema.customer c WHERE c.customer_id NOT IN (SELECT customer_id FROM order_schema.ord...	3 row(s) returned
71	19:59:07	SELECT o.order_id, o.order_date, p.product_name, oi.quantity FROM order_schema.orders o JOIN order_schema.order_items oi ON...	1 row(s) returned

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

Result 13

Output

Action Output

#	Time	Action	Message
71	19:59:07	SELECT o.order_id, o.order_date, p.product_name, oi.quantity FROM order_schema.orders o JOIN order_schema.order_items oi ON...	1 row(s) returned
72	19:59:31	SELECT s.status_name, COUNT(o.order_id) AS order_count, SUM(o.total_amount) AS total_revenue FROM order_schema.orders o ...	3 row(s) returned

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

```
SELECT 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'
GROUP BY c.customer_id, c.first_name, c.last_name
HAVING SUM(oi.quantity) > 1;
```

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

customer_id	first_name	last_name

Result 14

Output

Action Output

#	Time	Action	Message
72	19:59:31	SELECT s.status_name, COUNT(o.order_id) AS order_count, SUM(o.total_amount) AS total_revenue FROM order_schema.orders o ...	3 row(s) returned
73	19:59:53	SELECT c.customer_id, c.first_name, c.last_name FROM order_schema.customer c JOIN order_schema.orders o ON c.customer_id ...	0 row(s) returned

13. Find the Products That Have Never Been Ordered

```
SELECT p.product_id, p.product_name
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
4	Monitor

Result 15

Output

Action Output

#	Time	Action	Message
73	19:59:53	SELECT c.customer_id, c.first_name, c.last_name FROM order_schema.customer c JOIN order_schema.orders o ON c.customer_id ...	0 row(s) returned
74	20:00:18	SELECT p.product_id, p.product_name FROM order_schema.product p LEFT JOIN order_schema.order_items oi ON p.product_id = ...	1 row(s) returned

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

```
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 >= CURRENT_DATE - INTERVAL 7 DAY
GROUP BY p.product_name;
```

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

product_name	total_quantity

Result 16

Output

Action Output

#	Time	Action	Message
74	20:00:18	SELECT p.product_id, p.product_name FROM order_schema.product p LEFT JOIN order_schema.order_items oi ON p.product_id = ...	1 row(s) returned
75	20:00:44	SELECT p.product_name, SUM(oi.quantity) AS total_quantity FROM order_schema.order_items oi JOIN order_schema.orders o ON ...	0 row(s) returned

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

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

Action Output

#	Time	Action	Message
75	20:00:44	SELECT p.product_name, SUM(oi.quantity) AS total_quantity FROM order_schema.order_items oi JOIN order_schema.orders o ON ...	0 row(s) returned
76	20:01:09	CREATE VIEW order_schema.product_details AS SELECT * FROM order_schema.product	0 row(s) affected

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.

```
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;
```

Output

Action Output

#	Time	Action	Message
76	20:01:09	CREATE VIEW order_schema.product_details AS SELECT * FROM order_schema.product	0 row(s) affected
77	20:01:39	CREATE VIEW order_schema.order_summary AS SELECT o.order_id, o.customer_id, o.order_date, o.total_amount, s.status_name ...	0 row(s) affected