Database Assignment

Question 1: Create the tables below in the database

```
SHOW DATABASES;
CREATE DATABASE collegeDb;
USE collegeDb;

CREATE TABLE Address(
    address_id INT PRIMARY KEY,
    street_address VARCHAR(255),
    city VARCHAR(100),
    state VARCHAR(100),
    postal_code VARCHAR(20)
);

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');

CREATE TABLE Department(
    department_id INT PRIMARY KEY,
    department_name VARCHAR(100)
);

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

CREATE TABLE Student (
    student_id INT PRIMARY KEY,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    birthdate DATE,
    department_id INT,
    address_id INT,
    FOREIGN KEY(department_id) REFERENCES Department(department_id),
    FOREIGN KEY(address_id) REFERENCES Address(address_id)
);

-- DROP TABLE Student;  -- Optional cleanup

INSERT INTO Student (student_id, first_name, last_name, birthdate, depart
ment_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);
```

Question 2: Use sample data to insert into database

```
LOAD DATA INFILE 'C:/Users/Anshu/studentdata.csv'
INTO TABLE Student
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 0 LINES;
```

Question 3: Find total number of students

```
SELECT COUNT(*) FROM Student;
```

Question 4: Find which department John belongs to

```
SELECT
   s.first_name, s.last_name, d.department_name
FROM
   Student s
INNER JOIN
   Department d ON s.department_id = d.department_id
WHERE
   s.first_name = 'John';
```

Question 5: List all departments with their number of students

```
SELECT
   d.department_id, 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_id;
```

Question 6: Select all students with their department and address

```
SELECT s.first_name, s.last_name, d.department_name, a.city, a.state, a.po
stal_code
FROM Student s
LEFT JOIN Department d ON s.department_id = d.department_id
LEFT JOIN Address a ON s.address_id = a.address_id;
```

Question 7: Find all students in the Computer Science department

```
SELECT s.*
FROM Student s
LEFT JOIN Department d ON s.department_id = d.department_id
WHERE d.department_name = 'Computer Science';
```

Question 8: Update John's city name to New York

```
UPDATE Address a
JOIN Student s ON a.address_id = s.address_id
SET a.city = 'New York'
WHERE s.first_name = 'John';
```

Question 9: Delete a student from the Student table

```
DELETE FROM Student
WHERE student_id = 5;
```

Question 10: Select all students with department and address in New York

```
SELECT s.*, a.city, d.department_name
FROM Student s
LEFT JOIN Department d ON s.department_id = d.department_id
LEFT JOIN Address a ON s.address_id = a.address_id
WHERE a.city = 'New York';
```

Question 11: Count students in each department

```
SELECT d.department_name, COUNT(s.student_id)
FROM Department d
LEFT JOIN Student s ON d.department_id = s.department_id
GROUP BY d.department_name;
```

Question 12: Find students who live in Springfield

```
SELECT s.*, a.city
FROM Student s
INNER JOIN Address a ON s.address_id = a.address_id
WHERE a.city = 'Springfield';
```

Question 13: Select students born in February

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

Question 14: Get department and address of a specific student (e.g., John)

```
SELECT s.first_name, d.*, a.*
FROM Student s
INNER JOIN Department d ON d.department_id = s.department_id
INNER JOIN Address a ON s.address_id = a.address_id
WHERE s.first_name = 'John';
```

Question 15: Find students born between 1995 and 1998

```
SELECT *
FROM Student
WHERE YEAR(birthdate) > 1995 AND YEAR(birthdate) < 1998;
```

Question 16: List all students with department names, sorted by department

```
SELECT s.first_name, s.last_name, d.department_name
FROM Student s
LEFT JOIN Department d ON s.department_id = d.department_id
ORDER BY d.department_name;
```

Question 17: Find number of students in each department living in 'Champaign'

```
SELECT d.department_name, COUNT(s.student_id) AS student_count
FROM Student s
JOIN Address a ON s.address_id = a.address_id
JOIN Department d ON s.department_id = d.department_id
WHERE a.city = 'Champaign'
GROUP BY d.department_name;
```

Question 18: Retrieve names of students who live on 'Pine' Street

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

Question 19: Update department of 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;
```

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

```
SELECT s.*
FROM Student s
JOIN Address a ON s.address_id = a.address_id
JOIN Department d ON s.department_id = d.department_id
WHERE a.city = 'Chicago' AND d.department_name = 'Mathematics';
```

Question 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');
```

Question 22: Find the student with the highest student_id

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

Question 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';
```

Question 24: Count the total number of addresses in the 'Champaign' city

```
SELECT COUNT(*) AS address_count
FROM Address
WHERE city = 'Champaign';
```

Question 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';
```

Question 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';
```

Question 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%';
```

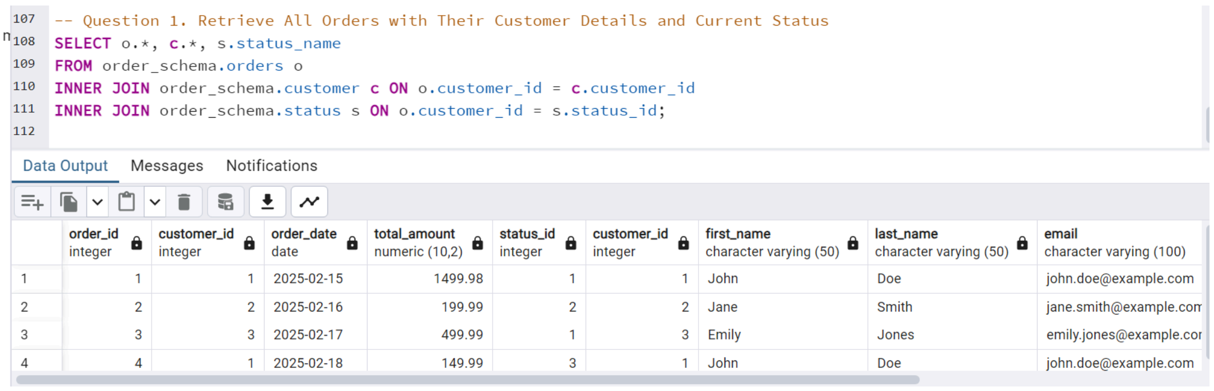
Question 28: Delete a student from the 'Mechanical Engineering' department

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

E-commerce Shop Database:

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

```
SELECT o.*, c.*, s.status_name
FROM order_schema.orders o
INNER JOIN order_schema.customer c ON o.customer_id = c.customer_id
INNER JOIN order_schema.status s ON o.status_id = s.status_id;
```

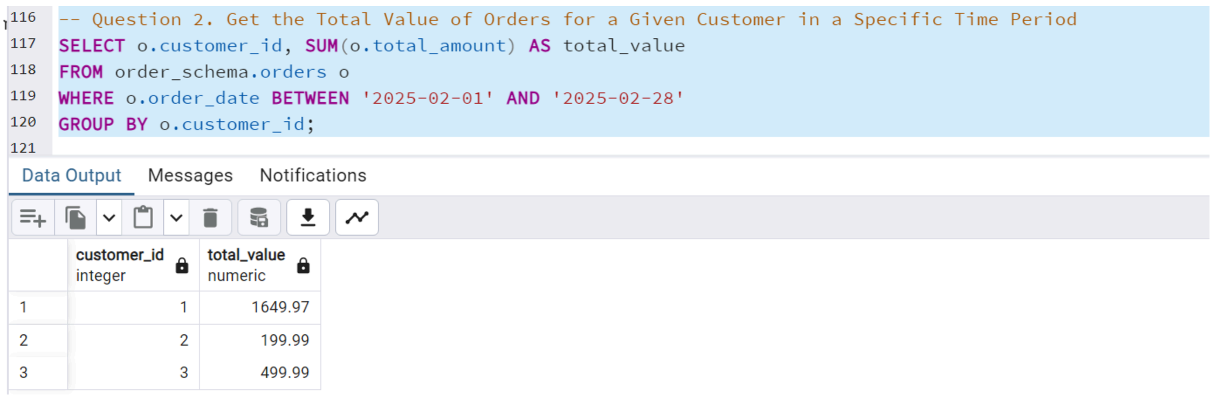


```
-- Question 1. Retrieve All Orders with Their Customer Details and Current Status
SELECT o.*, c.*, s.status_name
FROM order_schema.orders o
INNER JOIN order_schema.customer c ON o.customer_id = c.customer_id
INNER JOIN order_schema.status s ON o.customer_id = s.status_id;
```

	order_id integer	customer_id integer	order_date date	total_amount numeric (10,2)	status_id integer	customer_id integer	first_name character varying (50)	last_name character varying (50)	email character varying (100)
1	1	1	2025-02-15	1499.98	1	1	John	Doe	john.doe@example.com
2	2	2	2025-02-16	199.99	2	2	Jane	Smith	jane.smith@example.com
3	3	3	2025-02-17	499.99	1	3	Emily	Jones	emily.jones@example.cor
4	4	1	2025-02-18	149.99	3	1	John	Doe	john.doe@example.com

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

```
SELECT o.customer_id, SUM(o.total_amount) AS total_value
FROM order_schema.orders o
WHERE o.order_date BETWEEN '2025-02-01' AND '2025-02-28'
GROUP BY o.customer_id;
```

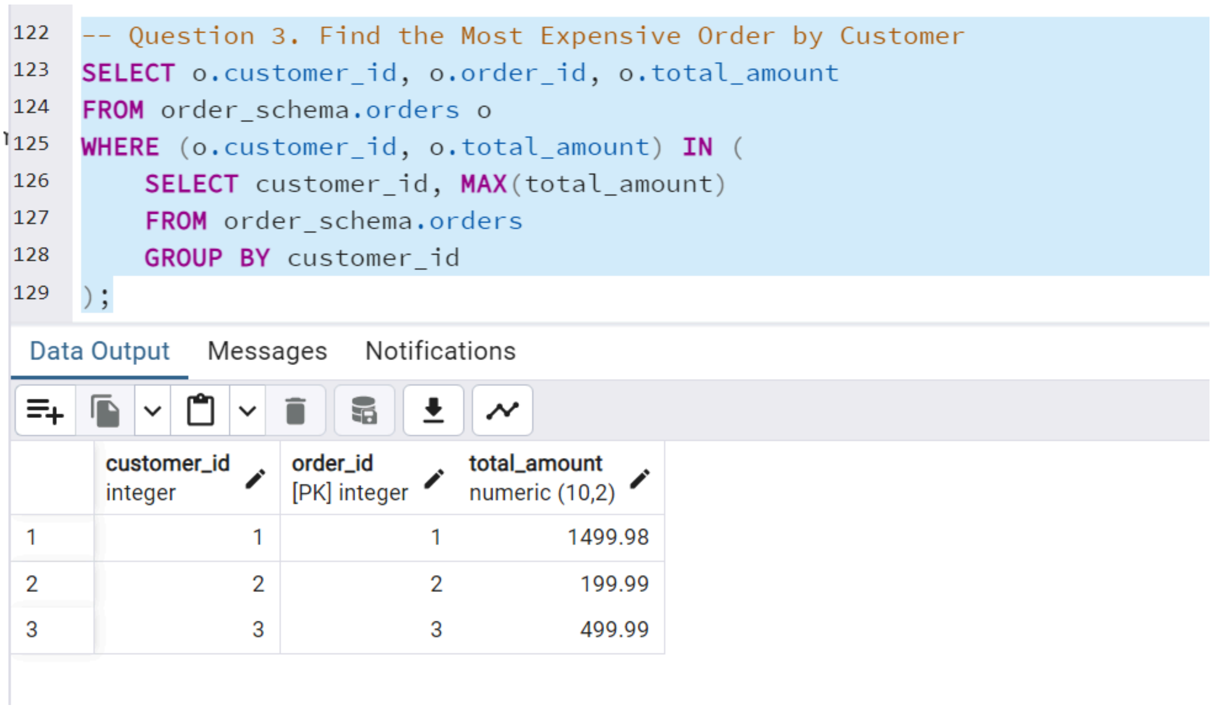


```
-- Question 2. Get the Total Value of Orders for a Given Customer in a Specific Time Period
SELECT o.customer_id, SUM(o.total_amount) AS total_value
FROM order_schema.orders o
WHERE o.order_date BETWEEN '2025-02-01' AND '2025-02-28'
GROUP BY o.customer_id;
```

	customer_id integer	total_value numeric
1	1	1649.97
2	2	199.99
3	3	499.99

Question 3. Find the Most Expensive Order by Customer

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

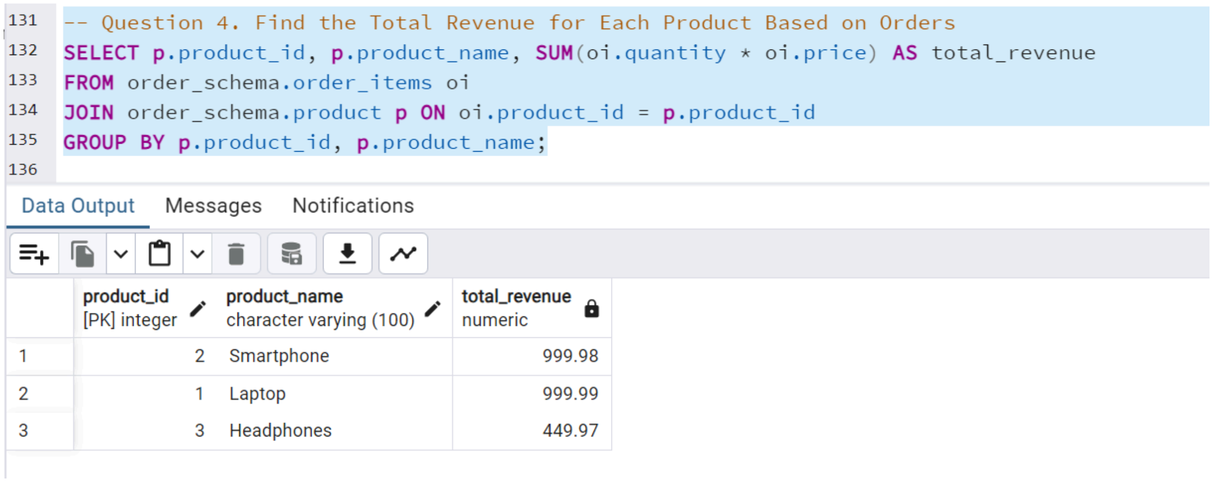


```
-- Question 3. Find the Most Expensive Order by Customer
SELECT o.customer_id, o.order_id, o.total_amount
FROM order_schema.orders o
WHERE (o.customer_id, o.total_amount) IN (
    SELECT customer_id, MAX(total_amount)
    FROM order_schema.orders
    GROUP BY customer_id
);
```

	customer_id integer	order_id [PK] integer	total_amount numeric (10,2)
1	1	1	1499.98
2	2	2	199.99
3	3	3	499.99

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

```
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;
```



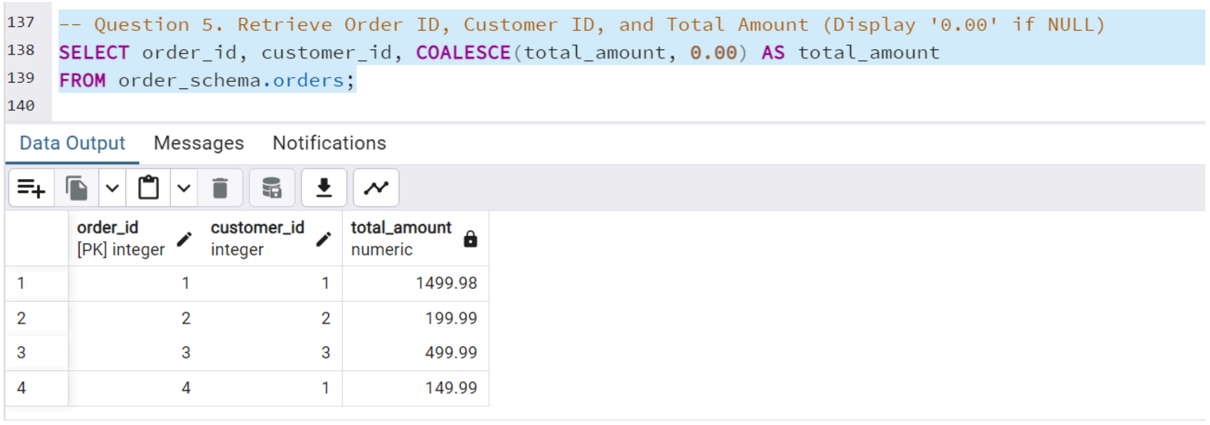
```
-- Question 4. Find the Total Revenue for Each Product Based on Orders
SELECT p.product_id, p.product_name, SUM(oi.quantity * oi.price) AS total_revenue
FROM order_schema.order_items oi
JOIN order_schema.product p ON oi.product_id = p.product_id
GROUP BY p.product_id, p.product_name;
```

Data Output Messages Notifications

	product_id [PK] integer	product_name character varying (100)	total_revenue numeric
1	2	Smartphone	999.98
2	1	Laptop	999.99
3	3	Headphones	449.97

Question 5. Retrieve Order ID, Customer ID, and Total Amount (Display '0.00' if NULL)

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



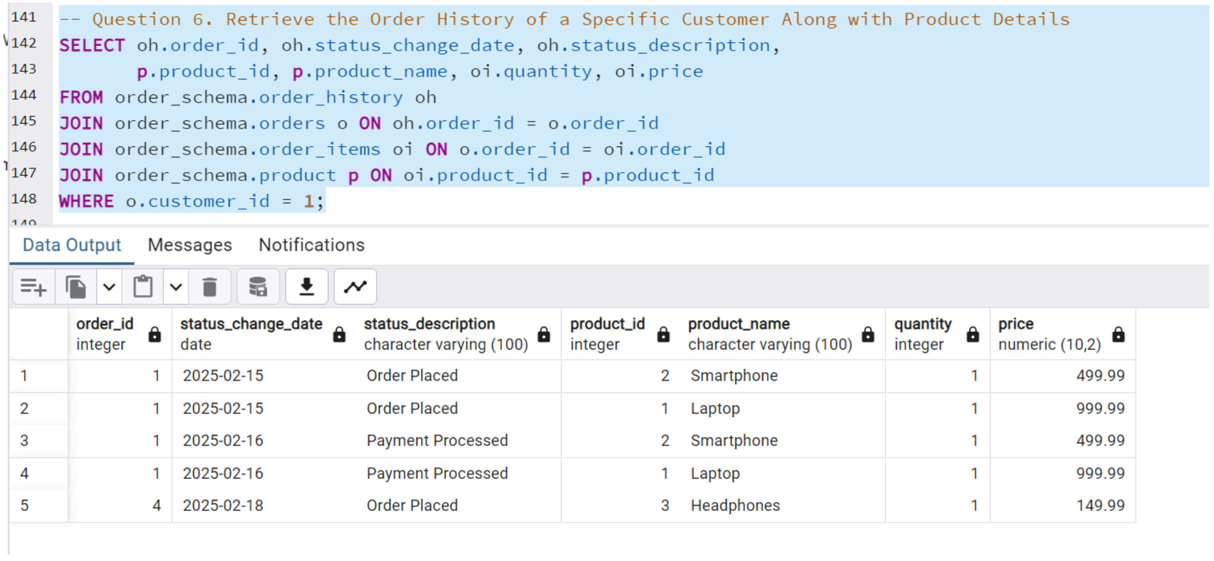
```
-- Question 5. Retrieve Order ID, Customer ID, and Total Amount (Display '0.00' if NULL)
SELECT order_id, customer_id, COALESCE(total_amount, 0.00) AS total_amount
FROM order_schema.orders;
```

Data Output Messages Notifications

	order_id [PK] integer	customer_id integer	total_amount numeric
1	1	1	1499.98
2	2	2	199.99
3	3	3	499.99
4	4	1	149.99

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

```
SELECT oh.order_id, oh.status_change_date, oh.status_description,
       p.product_id, 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;
```



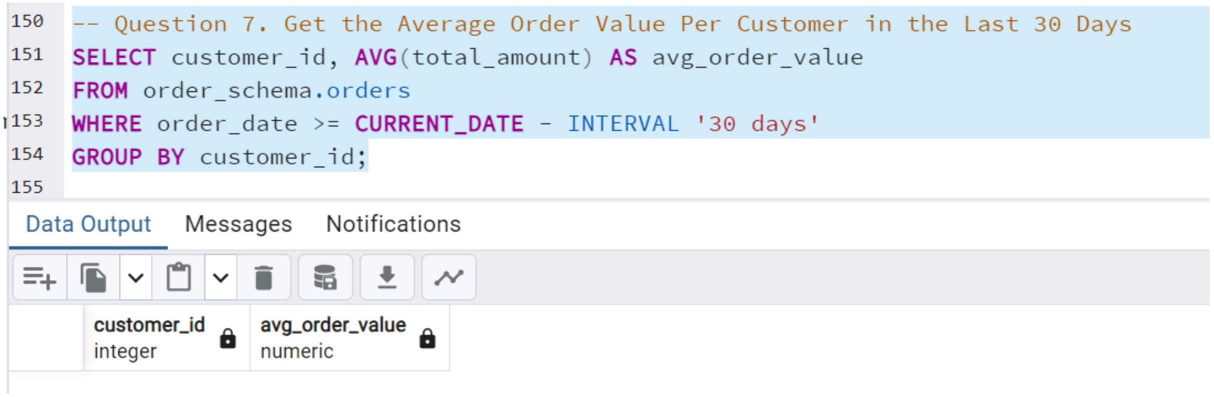
```
-- Question 6. Retrieve the Order History of a Specific Customer Along with Product Details
SELECT oh.order_id, oh.status_change_date, oh.status_description,
       p.product_id, p.product_name, oi.quantity, oi.price
FROM order_schema.order_history oh
JOIN order_schema.orders o ON oh.order_id = o.order_id
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;
```

Data Output Messages Notifications

	order_id integer	status_change_date date	status_description character varying (100)	product_id integer	product_name character varying (100)	quantity integer	price numeric (10,2)
1	1	2025-02-15	Order Placed	2	Smartphone	1	499.99
2	1	2025-02-15	Order Placed	1	Laptop	1	999.99
3	1	2025-02-16	Payment Processed	2	Smartphone	1	499.99
4	1	2025-02-16	Payment Processed	1	Laptop	1	999.99
5	4	2025-02-18	Order Placed	3	Headphones	1	149.99

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

```
SELECT customer_id, AVG(total_amount) AS avg_order_value
FROM order_schema.orders
WHERE order_date >= CURRENT_DATE - INTERVAL '30 days'
GROUP BY customer_id;
```



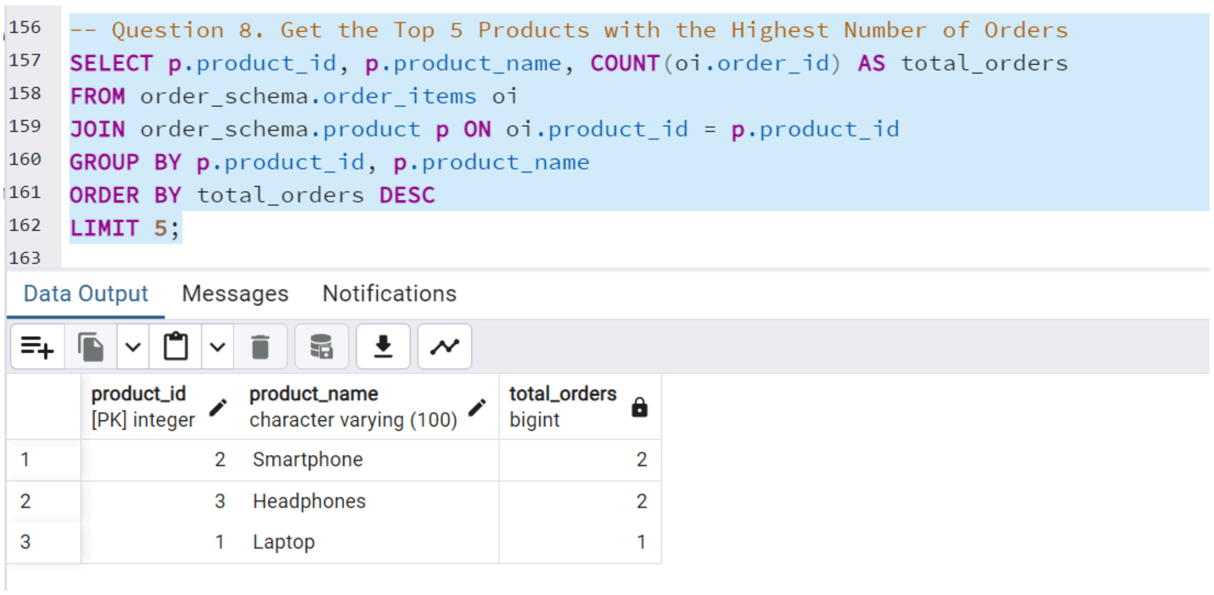
```
-- Question 7. Get the Average Order Value Per Customer in the Last 30 Days
SELECT customer_id, AVG(total_amount) AS avg_order_value
FROM order_schema.orders
WHERE order_date >= CURRENT_DATE - INTERVAL '30 days'
GROUP BY customer_id;
```

Data Output Messages Notifications

	customer_id integer	avg_order_value numeric

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

```
SELECT p.product_id, p.product_name, COUNT(oi.order_id) AS total_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 total_orders DESC
LIMIT 5;
```

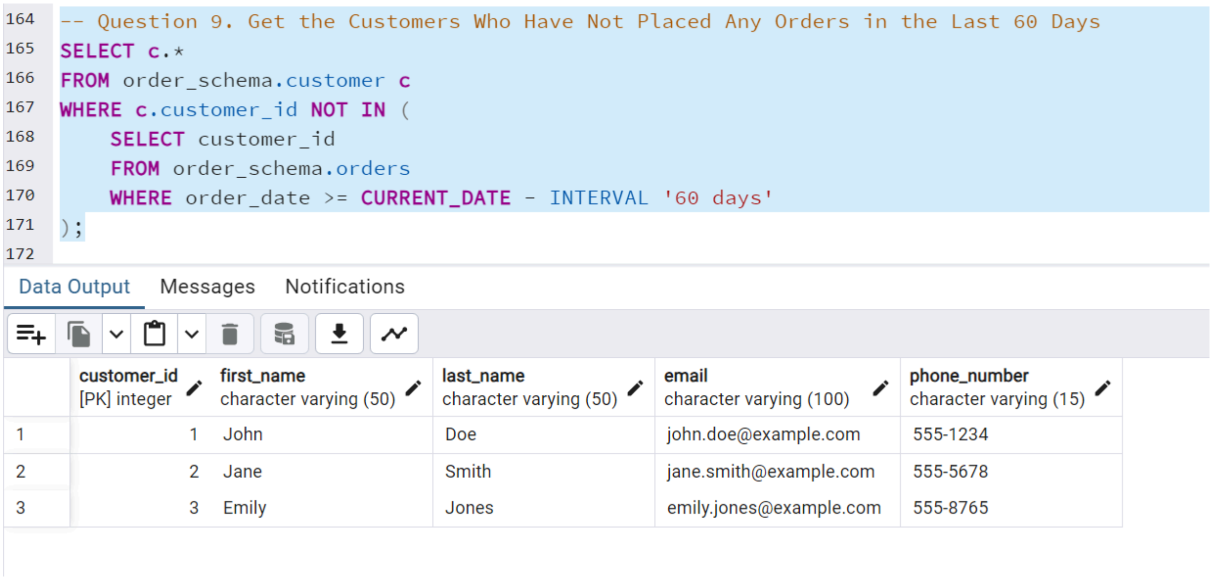


```
-- Question 8. Get the Top 5 Products with the Highest Number of Orders
SELECT p.product_id, p.product_name, COUNT(oi.order_id) AS total_orders
FROM order_schema.order_items oi
JOIN order_schema.product p ON oi.product_id = p.product_id
GROUP BY p.product_id, p.product_name
ORDER BY total_orders DESC
LIMIT 5;
```

	product_id [PK] integer	product_name character varying (100)	total_orders bigint
1	2	Smartphone	2
2	3	Headphones	2
3	1	Laptop	1

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

```
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 days'
);
```



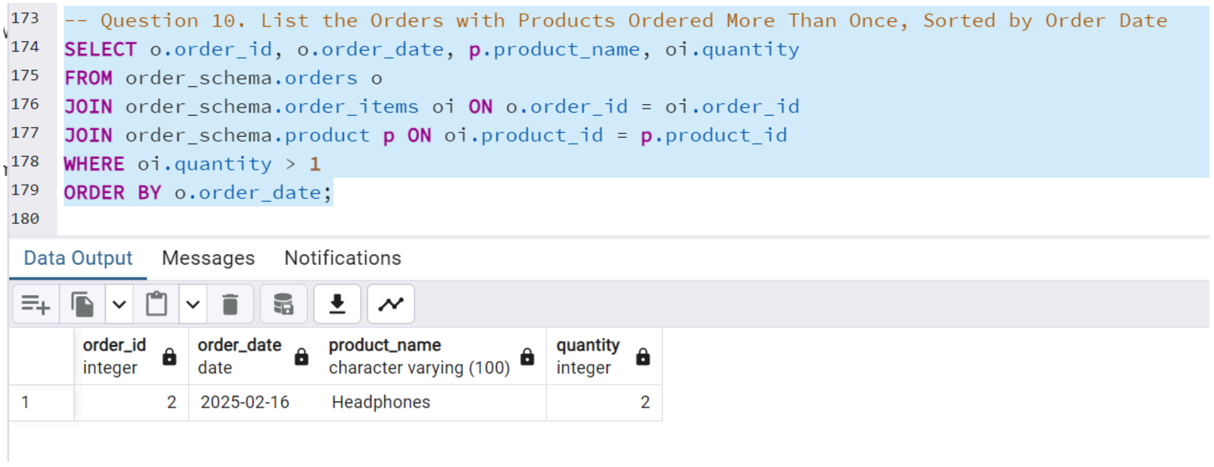
```
-- Question 9. Get the Customers Who Have Not Placed Any Orders in the Last 60 Days
SELECT c.*
FROM order_schema.customer c
WHERE c.customer_id NOT IN (
    SELECT customer_id
    FROM order_schema.orders
    WHERE order_date >= CURRENT_DATE - INTERVAL '60 days'
);
```

Data Output Messages Notifications

	customer_id [PK] integer	first_name character varying (50)	last_name character varying (50)	email character varying (100)	phone_number character varying (15)
1	1	John	Doe	john.doe@example.com	555-1234
2	2	Jane	Smith	jane.smith@example.com	555-5678
3	3	Emily	Jones	emily.jones@example.com	555-8765

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

```
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;
```



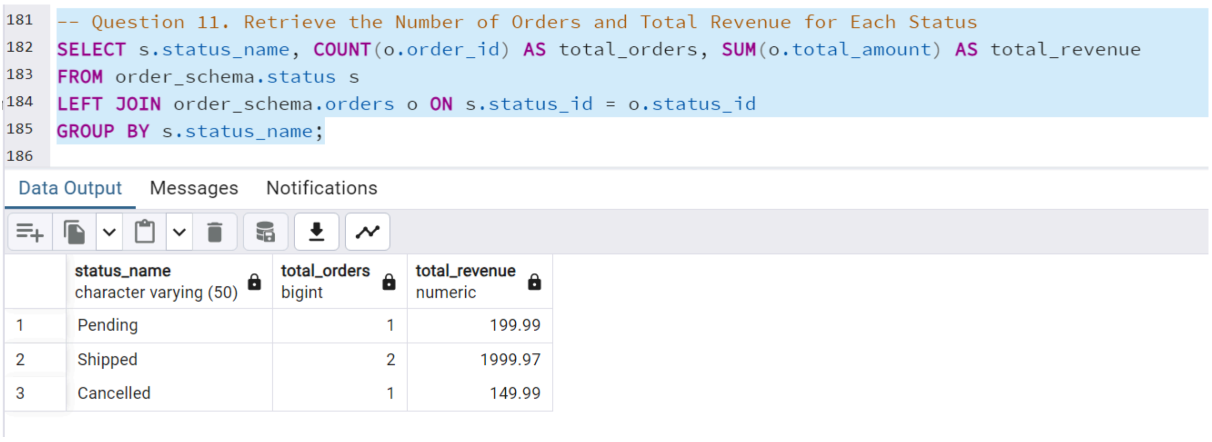
```
-- Question 10. List the Orders with Products Ordered More Than Once, Sorted by Order Date
SELECT o.order_id, o.order_date, p.product_name, oi.quantity
FROM order_schema.orders o
JOIN order_schema.order_items oi ON o.order_id = oi.order_id
JOIN order_schema.product p ON oi.product_id = p.product_id
WHERE oi.quantity > 1
ORDER BY o.order_date;
```

Data Output Messages Notifications

	order_id integer	order_date date	product_name character varying (100)	quantity integer
1	2	2025-02-16	Headphones	2

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

```
SELECT s.status_name, COUNT(o.order_id) AS total_orders, SUM(o.total_a
mount) AS total_revenue
FROM order_schema.status s
LEFT JOIN order_schema.orders o ON s.status_id = o.status_id
GROUP BY s.status_name;
```



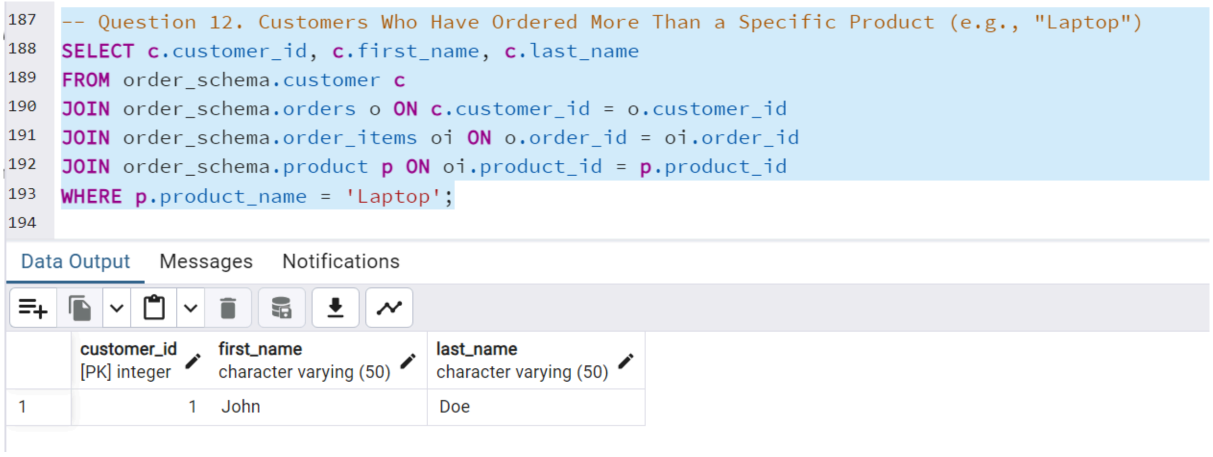
```
-- Question 11. Retrieve the Number of Orders and Total Revenue for Each Status
SELECT s.status_name, COUNT(o.order_id) AS total_orders, SUM(o.total_amount) AS total_revenue
FROM order_schema.status s
LEFT JOIN order_schema.orders o ON s.status_id = o.status_id
GROUP BY s.status_name;
```

Data Output Messages Notifications

	status_name character varying (50)	total_orders bigint	total_revenue numeric
1	Pending	1	199.99
2	Shipped	2	1999.97
3	Cancelled	1	149.99

Question 12. 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';
```



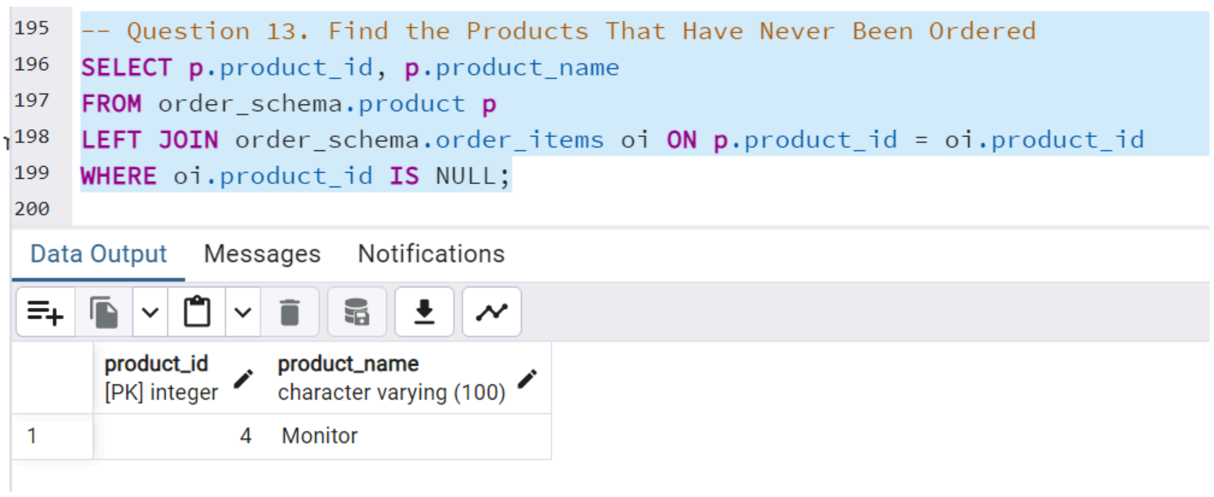
```
-- Question 12. Customers Who Have Ordered More Than a Specific Product (e.g., "Laptop")
SELECT c.customer_id, c.first_name, c.last_name
FROM order_schema.customer c
JOIN order_schema.orders o ON c.customer_id = o.customer_id
JOIN order_schema.order_items oi ON o.order_id = oi.order_id
JOIN order_schema.product p ON oi.product_id = p.product_id
WHERE p.product_name = 'Laptop';
```

Data Output Messages Notifications

	customer_id [PK] integer	first_name character varying (50)	last_name character varying (50)
1	1	John	Doe

Question 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;
```

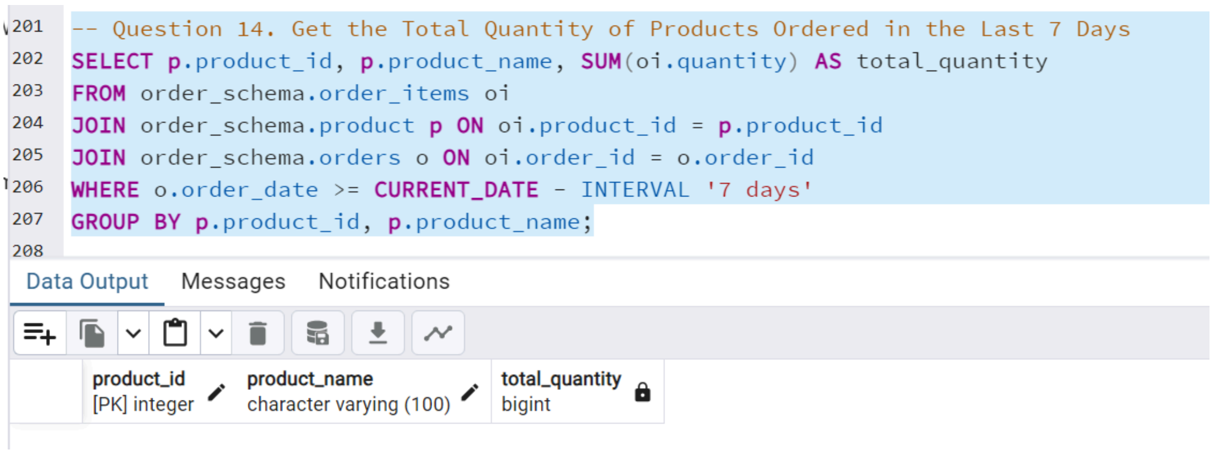


```
-- Question 13. Find the Products That Have Never Been Ordered
SELECT p.product_id, p.product_name
FROM order_schema.product p
LEFT JOIN order_schema.order_items oi ON p.product_id = oi.product_id
WHERE oi.product_id IS NULL;
```

	product_id [PK] integer	product_name character varying (100)
1	4	Monitor

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

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

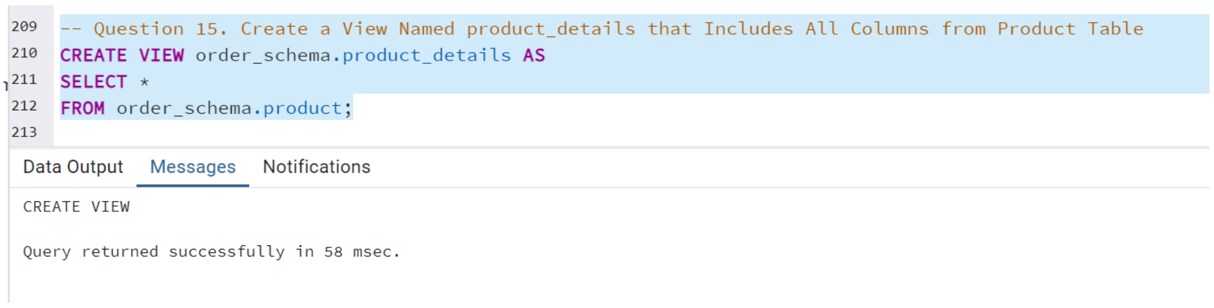


```
-- Question 14. Get the Total Quantity of Products Ordered in the Last 7 Days
SELECT p.product_id, p.product_name, SUM(oi.quantity) AS total_quantity
FROM order_schema.order_items oi
JOIN order_schema.product p ON oi.product_id = p.product_id
JOIN order_schema.orders o ON oi.order_id = o.order_id
WHERE o.order_date >= CURRENT_DATE - INTERVAL '7 days'
GROUP BY p.product_id, p.product_name;
```

	product_id [PK] integer	product_name character varying (100)	total_quantity bigint

Question 15. Create a View Named product_details that Includes All Columns from Product Table

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



```
-- Question 15. Create a View Named product_details that Includes All Columns from Product Table
CREATE VIEW order_schema.product_details AS
SELECT *
FROM order_schema.product;
```

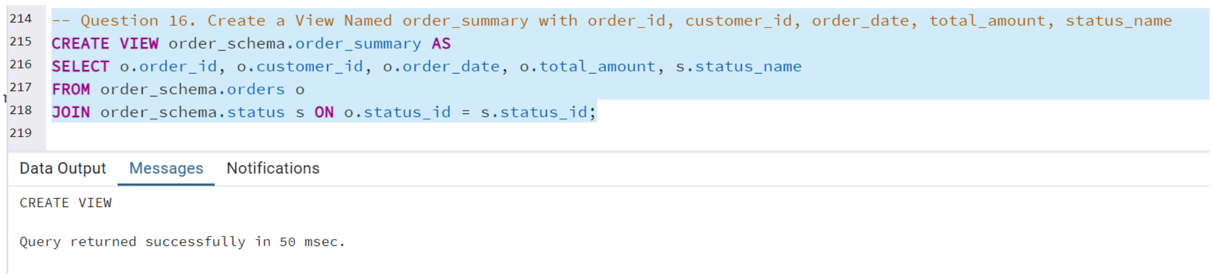
Data Output Messages Notifications

```
CREATE VIEW

Query returned successfully in 58 msec.
```

Question 16. Create a View Named order_summary with order_id, customer_id, order_date, total_amount, status_name

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



```
-- Question 16. Create a View Named order_summary with order_id, customer_id, order_date, total_amount, status_name
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;
```

Data Output Messages Notifications

```
CREATE VIEW

Query returned successfully in 50 msec.
```