DATABASE MANAGEMENT SYSTEM ASSIGNMENT

- 28th May 2025
- Abhik Chakraborty (Abhik.Chakraborty@bounteous.com)

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.**

   Solution:

```sql
CREATE TABLE student (
    student_id SERIAL PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    birthdate DATE,
    department_id INT REFERENCES department(department_id),
    address_id INT REFERENCES address(address_id)
);
```

```sql
CREATE TABLE address (
    address_id SERIAL PRIMARY KEY,
    street_address VARCHAR(255),
    city VARCHAR(100),
    state VARCHAR(50),
    postal_code VARCHAR(20)
);
```

```sql
CREATE TABLE department (
    department_id SERIAL PRIMARY KEY,
    department_name VARCHAR(100) NOT NULL
);

show tables;
```

Result Grid | Filter Rows: | Export: | Wrap

Tables_in_assignment
address
department
student

2. **Use Sample data from sampledata.txt to insert data into the database.**

Answer:

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

```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');
```

```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);
```

3. **Write a query to find the total number of students.**

Answer:

```sql
SELECT COUNT(*) FROM student;
```

Result Grid Filter Rows:

	COUNT(*)
▶	8

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

Answer:

```
SELECT d.department_name
FROM student s
JOIN department d ON s.department_id = d.department_id
```

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

department_name
Computer Science

John belong to Computer Science department.

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

Answer:

```
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.**

Answer:

```
SELECT s.*, d.department_name, a.city, a.street_address
FROM student s
JOIN department d ON s.department_id = d.department_id
JOIN address a ON s.address_id = a.address_id;
```

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
2	Jane	Smith	1996-07-22	2	2	Mechanical Engineering	Decatur	456 Oak 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
6	Daniel	Wilson	1995-06-10	6	6	Biology	Urbana	310 Maple Dr
7	Olivia	Martinez	1997-11-25	1	7	Computer Science	Champaign	415 Oak Blvd
8	Ethan	Miller	1996-03-30	2	8	Mechanical Engineering	Carbondale	520 Pine Rd

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

Answer:

```
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.**

Answer:

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

```
SELECT s.first_name, s.last_name, a.city
FROM student s
JOIN address a ON s.address_id = a.address_id
WHERE s.first_name = 'Jane';
```

Result Grid | Filter Rows: | Export: | Wra

first_name	last_name	city
Jane	Smith	New York

9. Delete a student from the student table.

Answer:

Before deleting:

```
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

After deleting:

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

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

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

John with student_id = 1 is deleted.

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

Answer:

```
SELECT s.*, 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:

student_id	first_name	last_name	birthdate	department_id	address_id	department_name	city
2	Jane	Smith	1996-07-22	2	2	Mechanical Engineering	New York

11. Count how many students are in each department

Answer:

```
SELECT d.department_name, COUNT(s.student_id) AS 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	count
Computer Science	1
Mechanical Engineering	2
Electrical Engineering	1
Civil Engineering	1
Mathematics	1
Biology	1

12. Find students who live in 'Springfield'

Answer:

```
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

There is no student from city = Springfield.

13. Select students whose birthday falls in February

Answer:

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

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

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

Answer:

Since John is deleted form database because of earlier query, took the example of Jane

```
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 = 'Jane';
```

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

department_name	address_id	street_address	city	state	postal_code
Mechanical Engineering	2	456 Oak St	New York	IL	62521

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

Answer:

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

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

student_id	first_name	last_name	birthdate	department_id	address_id
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

Answer:

```
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
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'

Answer:

```sql
SELECT d.department_name, COUNT(*) AS 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 Conter

department_name	count
Electrical Engineering	1
Computer Science	1

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

Answer:

```sql
SELECT s.*
FROM student s
JOIN address a ON s.address_id = a.address_id
WHERE LOWER(a.street_address) LIKE '%pine%';
```

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

student_id	first_name	last_name	birthdate	department_id	address_id
3	Alice	Johnson	1994-11-30	3	3
8	Ethan	Miller	1996-03-30	2	8

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

Answer:

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

Query to check:

```
SELECT s.student_id, s.first_name, s.department_id, d.department_name
FROM student s
JOIN department d ON s.department_id = d.department_id
WHERE s.student_id = 6;
```

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

student_id	first_name	department_id	department_name
6	Daniel	2	Mechanical Engineering

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

Answer:

```
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'

Answer:

```
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

Answer:

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

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

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

Answer:

```
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
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
8	Ethan	Miller	1996-03-30	2	8

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

Answer:

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

Result Grid | Filter Rows:

COUNT(*)
2

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

Answer:

```sql
SELECT s.*
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:

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

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

Answer:

```sql
SELECT AVG(TIMESTAMPDIFF(YEAR, s.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'

Answer:

```
SELECT s.*, 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:

student_id	first_name	last_name	birthdate	department_id	address_id	department_name	city
2	Jane	Smith	1996-07-22	2	2	Mechanical Engineering	New York
5	Sophia	Davis	1998-01-05	5	5	Mathematics	Chicago
6	Daniel	Wilson	1995-06-10	2	6	Mechanical Engineering	Urbana
8	Ethan	Miller	1996-03-30	2	8	Mechanical Engineering	Carbondale

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

Answer:

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

Verification Query:

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

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

student_id	first_name	last_name	birthdate	department_id	address_id
6	Daniel	Wilson	1995-06-10	2	6
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**

   Answer:

```sql
SELECT o.order_id, o.order_date, o.total_amount,
       c.first_name, c.last_name, 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	status_name
1	2025-02-15	1499.98	John	Doe	Shipped
4	2025-02-18	149.99	John	Doe	Cancelled
2	2025-02-16	199.99	Jane	Smith	Pending
3	2025-02-17	499.99	Emily	Jones	Shipped

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

   Answer:

```sql
SELECT SUM(total_amount) AS total_value
FROM order_schema.orders
WHERE customer_id = 1
  AND order_date BETWEEN '2025-02-15' AND '2025-02-18';
```

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

total_value
1649.97

3. Find the Most Expensive Order by Customer

Answer:

```sql
SELECT customer_id, MAX(total_amount) AS max_order_amount
FROM order_schema.orders
GROUP BY customer_id;
```

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

customer_id	max_order_amount
1	1499.98
2	199.99
3	499.99

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

Answer:

```sql
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

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.**

Answer:

```
SELECT order_id, customer_id,
IFNULL(total_amount, 0.00) AS total_amount
FROM order_schema.orders;
```

Result Grid | Filter Rows: | Export:

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**

Answer:

```
SELECT o.order_id, o.order_date, p.product_name, oi.quantity, oi.price
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 o.customer_id = 1;
```

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

order_id	order_date	product_name	quantity	price
1	2025-02-15	Laptop	1	999.99
1	2025-02-15	Smartphone	1	499.99
4	2025-02-18	Headphones	1	149.99

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

Answer:

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

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

customer_id	avg_order_value

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

Answer:

```sql
SELECT p.product_name, COUNT(DISTINCT oi.order_id) AS num_orders
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 num_orders DESC
LIMIT 5;
```

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

product_name	num_orders
Headphones	2
Smartphone	2
Laptop	1

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

Answer:

```
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 Content:

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**

Answer:

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

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

order_id	order_date
2	2025-02-16

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

Answer:

```sql
SELECT s.status_name,
COUNT(o.order_id) AS num_orders,
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	num_orders	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")

Answer:

```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' AND oi.quantity > 1;
```

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

customer_id	first_name	last_name

13.Find the Products That Have Never Been Ordered

Answer:

```
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

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

Answer:

```
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.**

Answer:

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

SELECT * FROM order_schema.product_details;
```

Result Grid | Filter Rows: | Export: | W

product_id	product_name	price	stock_quantity
1	Laptop	999.99	50
2	Smartphone	499.99	100
3	Headphones	149.99	200
4	Monitor	199.99	75

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.**

Answer:

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

```sql
SELECT * FROM order_summary;
```

Result Grid | Filter Rows: | Export: | Wrap C

order_id	customer_id	order_date	total_amount	status_name
1	1	2025-02-15	1499.98	Shipped
3	3	2025-02-17	499.99	Shipped
2	2	2025-02-16	199.99	Pending
4	1	2025-02-18	149.99	Cancelled