MySql Project-2

Zomato Clone

ZOMATO CLONE – PROJECT SCENARIOS

- Scenario 1: Display all orders with customer details
- Scenario 2: Display all users including those who have not placed any orders
- Scenario 3: Display all restaurants including those with no orders
- Scenario 4: Find users who live in the same city
- Scenario 5: Show orders from highly rated active restaurants
- Scenario 6: Count total orders for each restaurant
- Scenario 7: Calculate revenue and average order value per restaurant

ZOMATO CLONE – PROJECT SCENARIOS

- Scenario 8: City-wise order count and total spending
- Scenario 9: Find the most expensive menu item
- Scenario 10: Log every new order insertion
- Scenario 11: Create a view for user order summary
- Scenario 12: Stored procedure to get orders of a restaurant
- Scenario 13: Improve search performance using index

Scenario 1: Display all orders with customer details

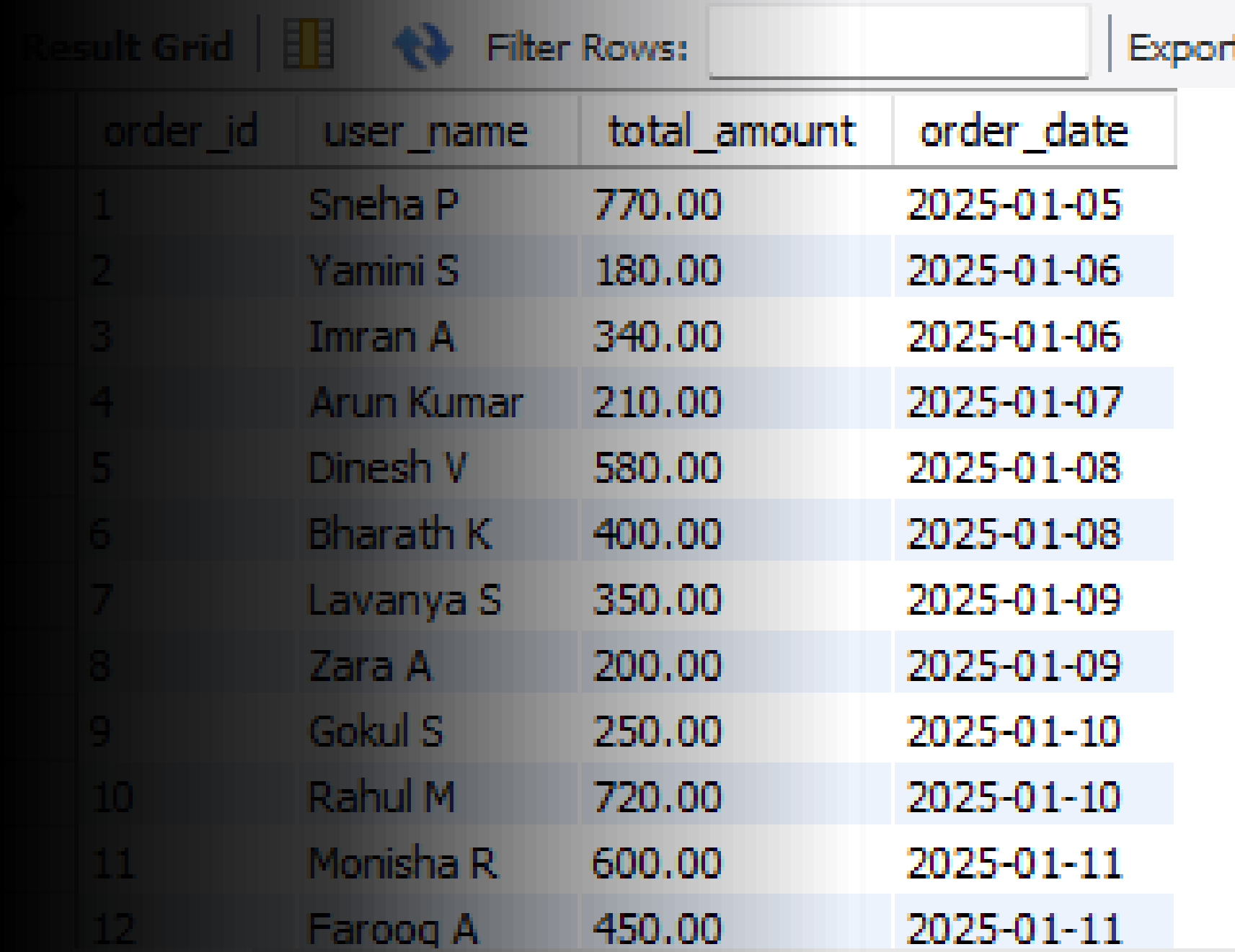
- select

  o.order_id, u.name as user_name,

  o.total_amount, o.order_date

  from orders o

  inner join users u on o.user_id = u.user_id;



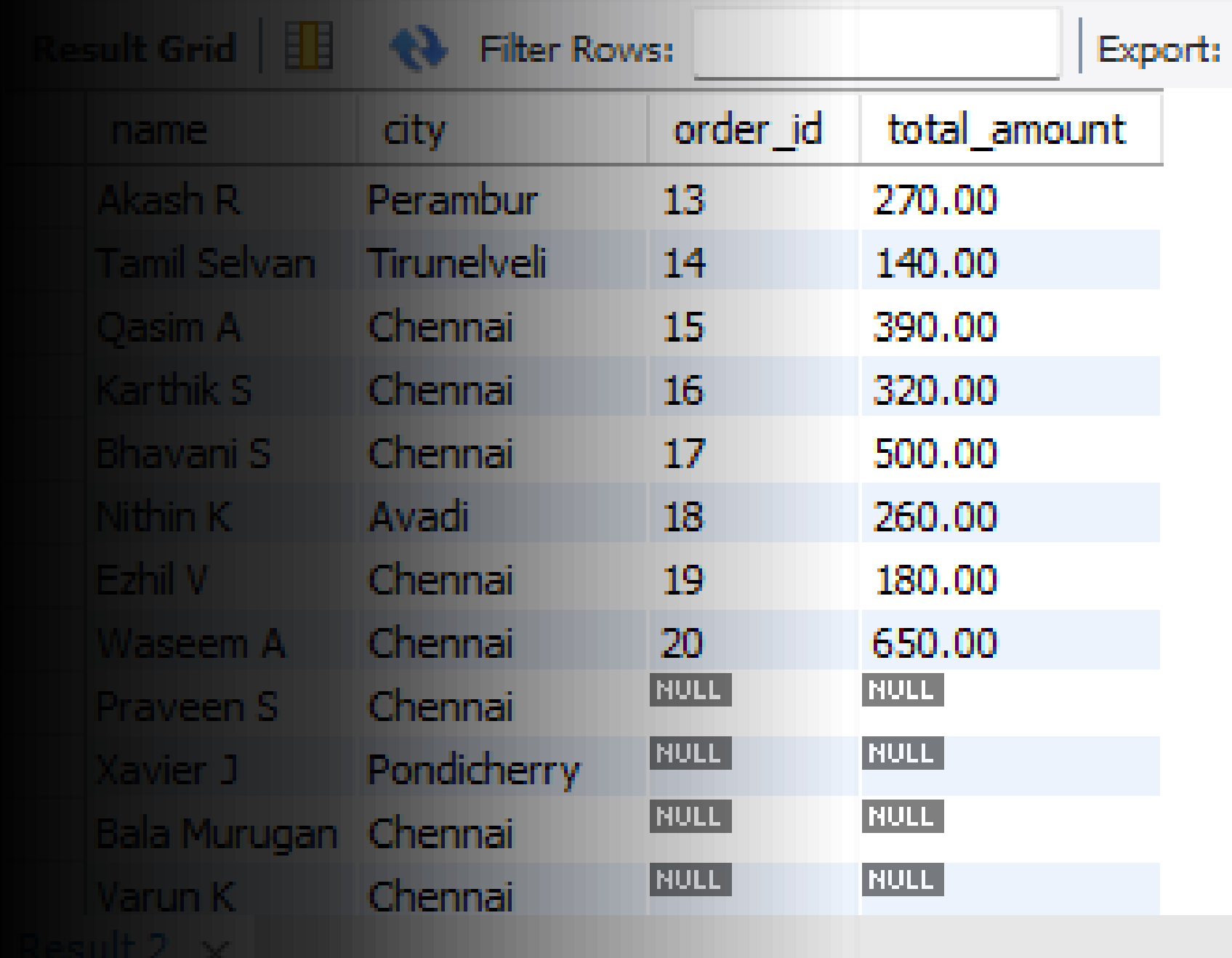
Scenario 2: Display all users including those who have not placed any orders

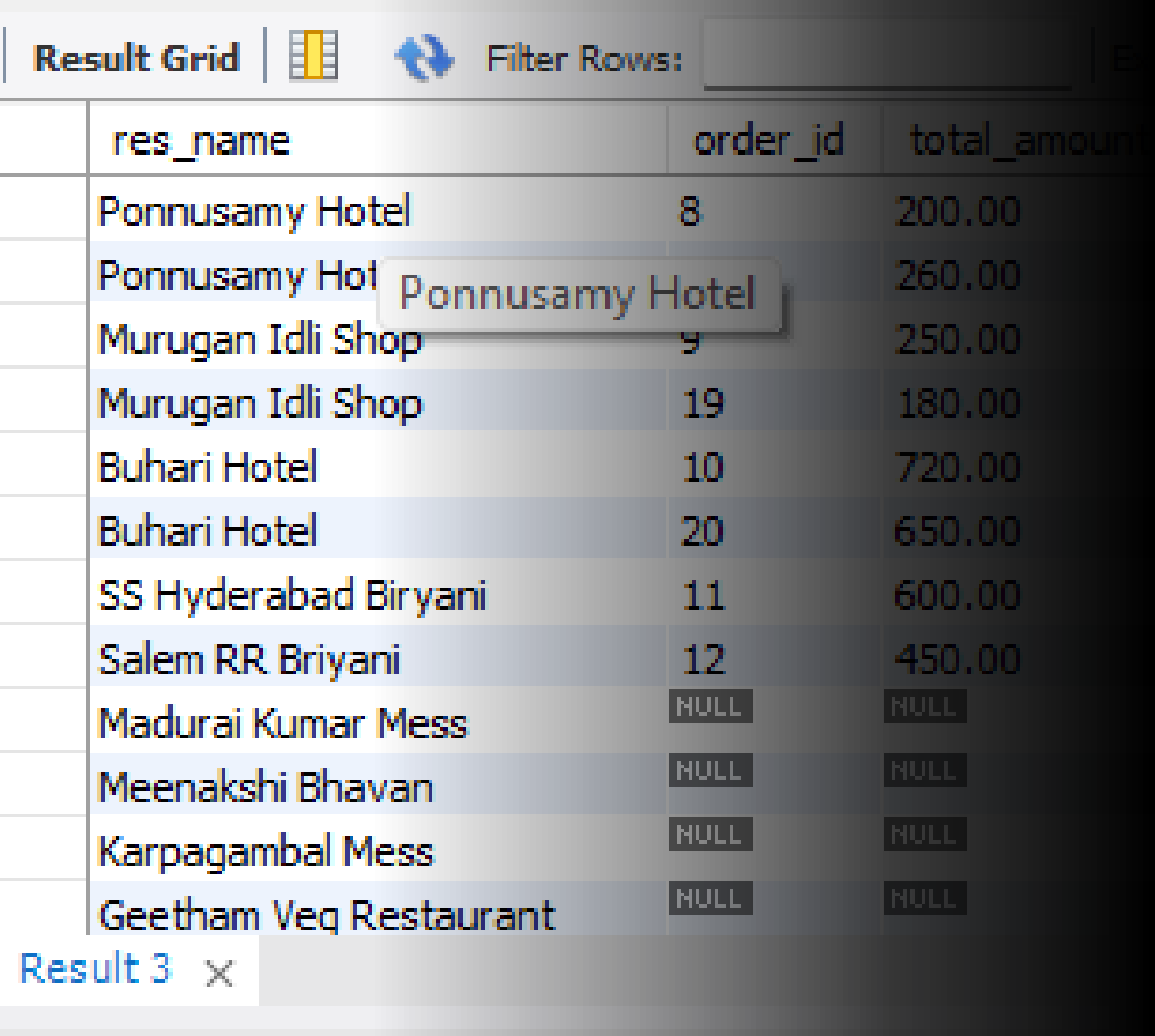
- select

u.name, u.city, o.order_id, o.total_amount

from users u

left join orders o on u.user_id = o.user_id;



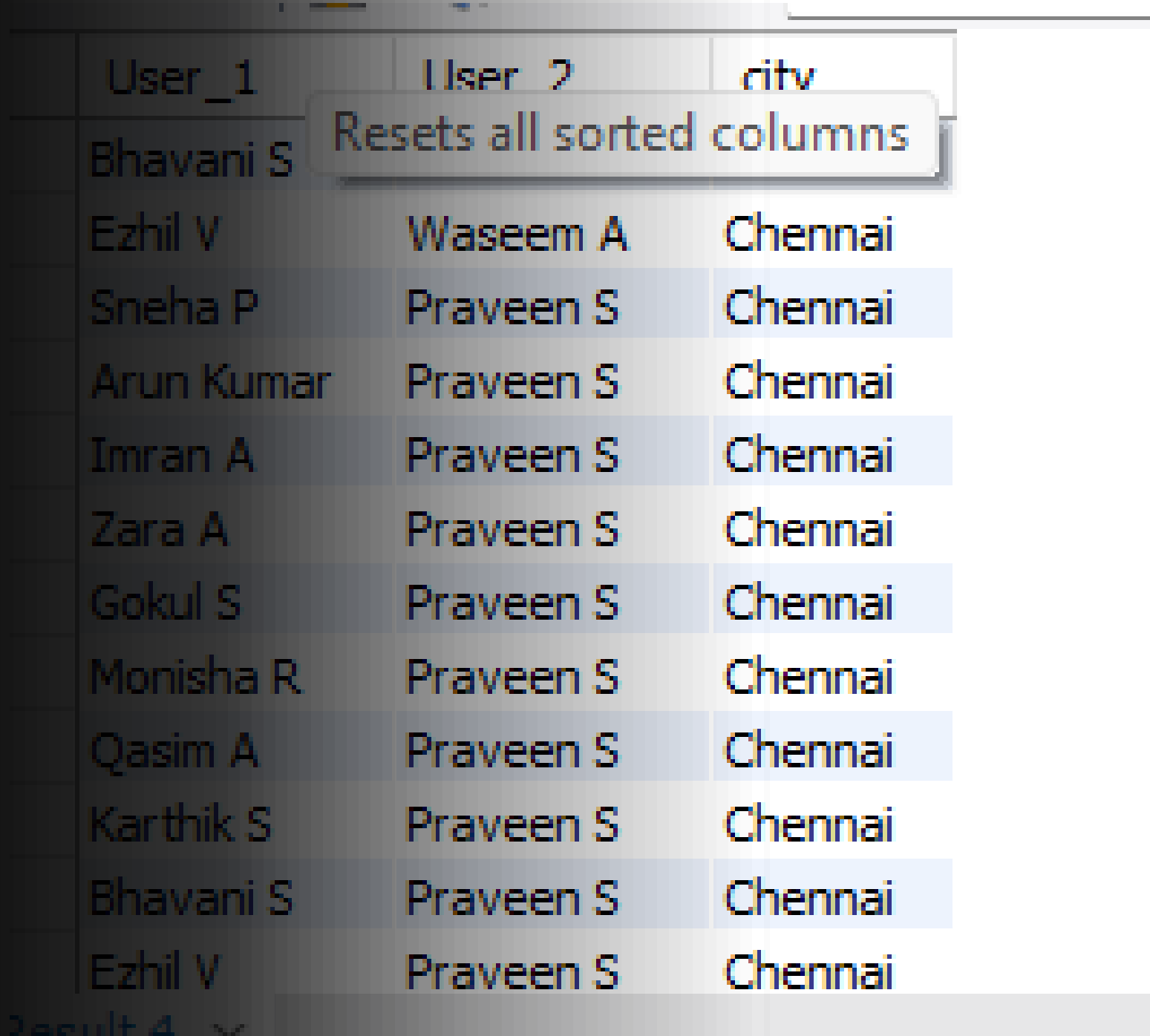


Scenario 3: Display all restaurants including those with no orders

- select
- r.res_name, o.order_id, o.total_amount
- from orders o
- right join restaurants r on o.res_id = r.res_id;

Scenario 4: Find users who live in the same city

- SELECT
- a.name AS User_1,
- b.name AS User_2,
- a.city
- FROM users b
- INNER JOIN users a ON a.city = b.city
- WHERE a.user_id < b.user_id;



Scenario 5: Show orders from highly rated active restaurants

```
SELECT
    o.order_id,
    u.name AS user_name,
    r.res_name,
    r.rating,
    r.is_active
FROM orders o
INNER JOIN users u ON o.user_id = u.user_id
INNER JOIN restaurants r ON o.res_id = r.res_id
WHERE r.rating > 4.0 AND r.is_active = true;
```

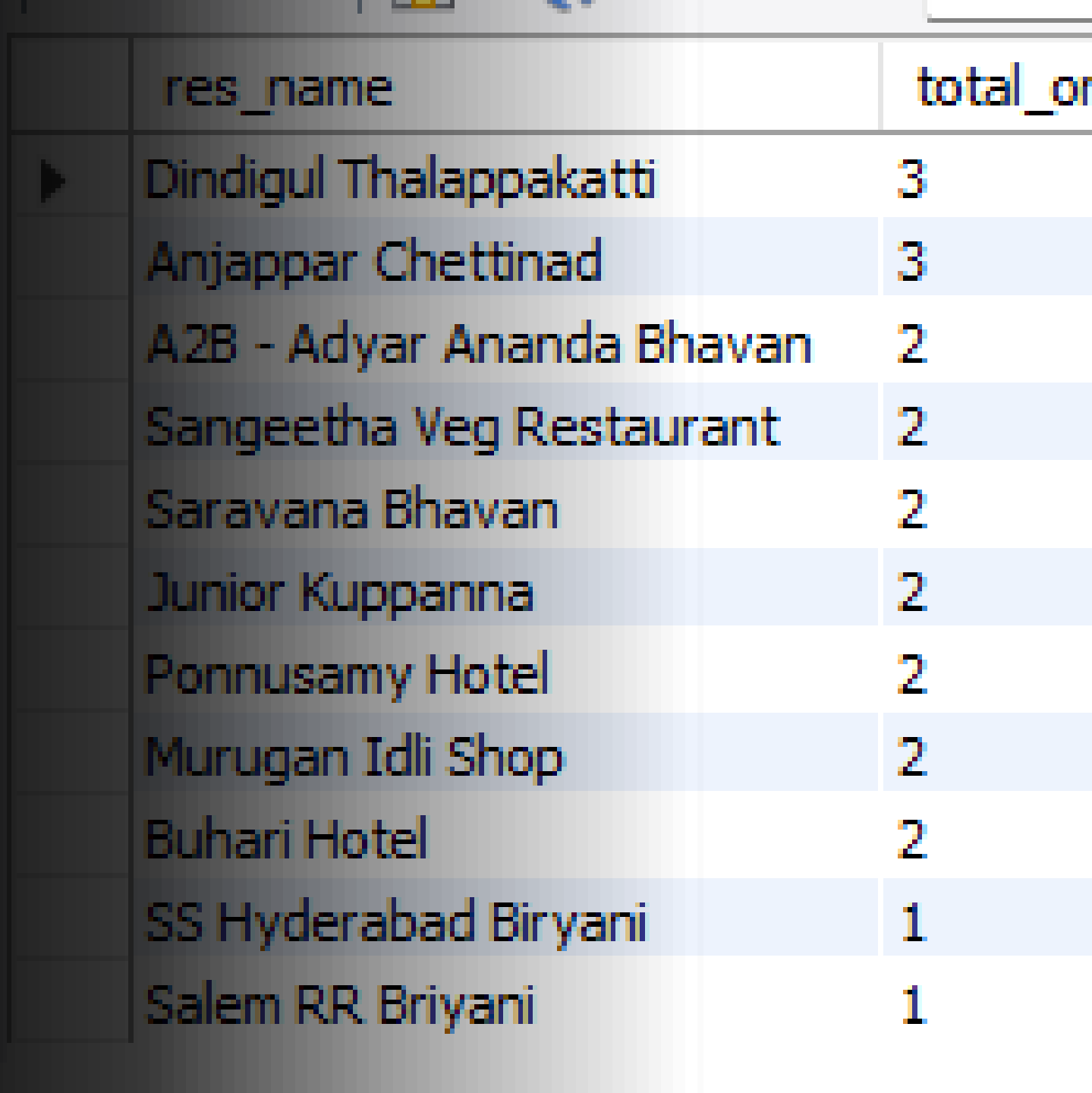
Result Grid | Filter Rows:

order_id	user_name	res_name
1	Sneha P	Dindigul Thalappakatti
7	Lavanya S	Dindigul Thalappakatti
21	Imran A	Dindigul Thalappakatti
2	Yamini S	A2B - Adyar Ananda Bhavan
13	Akash R	A2B - Adyar Ananda Bhavan
4	Arun Kumar	Sangeetha Veg Restaurant
14	Tamil Selvan	Sangeetha Veg Restaurant
3	Imran A	Saravana Bhavan
15	Qasim A	Saravana Bhavan
5	Dinesh V	Junior Kuppanna
16	Karthik S	Junior Kuppanna
8	Zara A	Ponnusamy Hotel

Result 5

Scenario 6: Count total orders for each restaurant

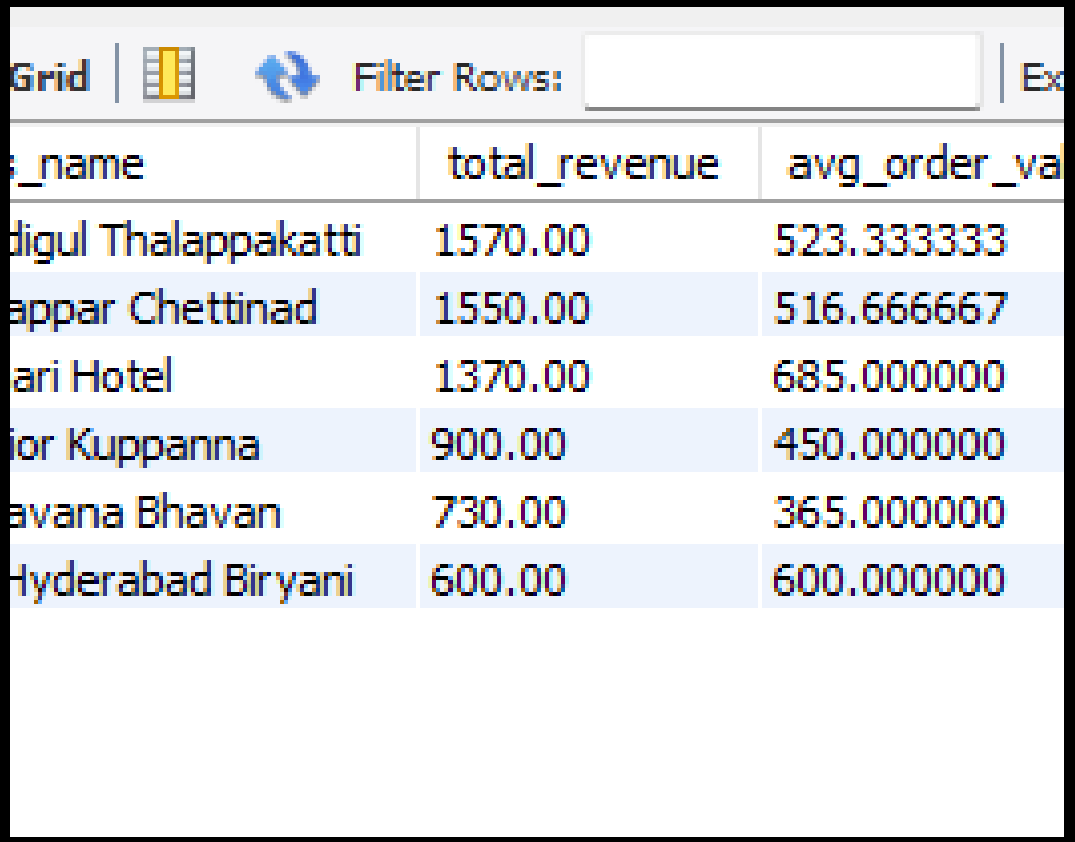
- SELECT
- r.res_name,
- COUNT(o.order_id) AS total_orders
- FROM orders o
- INNER JOIN restaurants r ON o.res_id = r.res_id
- GROUP BY r.res_name
- ORDER BY total_orders DESC;



res_name	total_or
Dindigul Thalappakatti	3
Anjappar Chettinad	3
A2B - Adyar Ananda Bhavan	2
Sangeetha Veg Restaurant	2
Saravana Bhavan	2
Junior Kuppanna	2
Ponnusamy Hotel	2
Murugan Idli Shop	2
Buhari Hotel	2
SS Hyderabad Biryani	1
Salem RR Briyani	1

Scenario 7: Calculate revenue and average order value per restaurant

- SELECT
- r.res_name,
- SUM(o.total_amount) AS total_revenue,
- AVG(o.total_amount) AS avg_order_value
- FROM orders o
- INNER JOIN restaurants r ON o.res_id = r.res_id
- GROUP BY r.res_name
- HAVING total_revenue > 500
- ORDER BY total_revenue DESC;

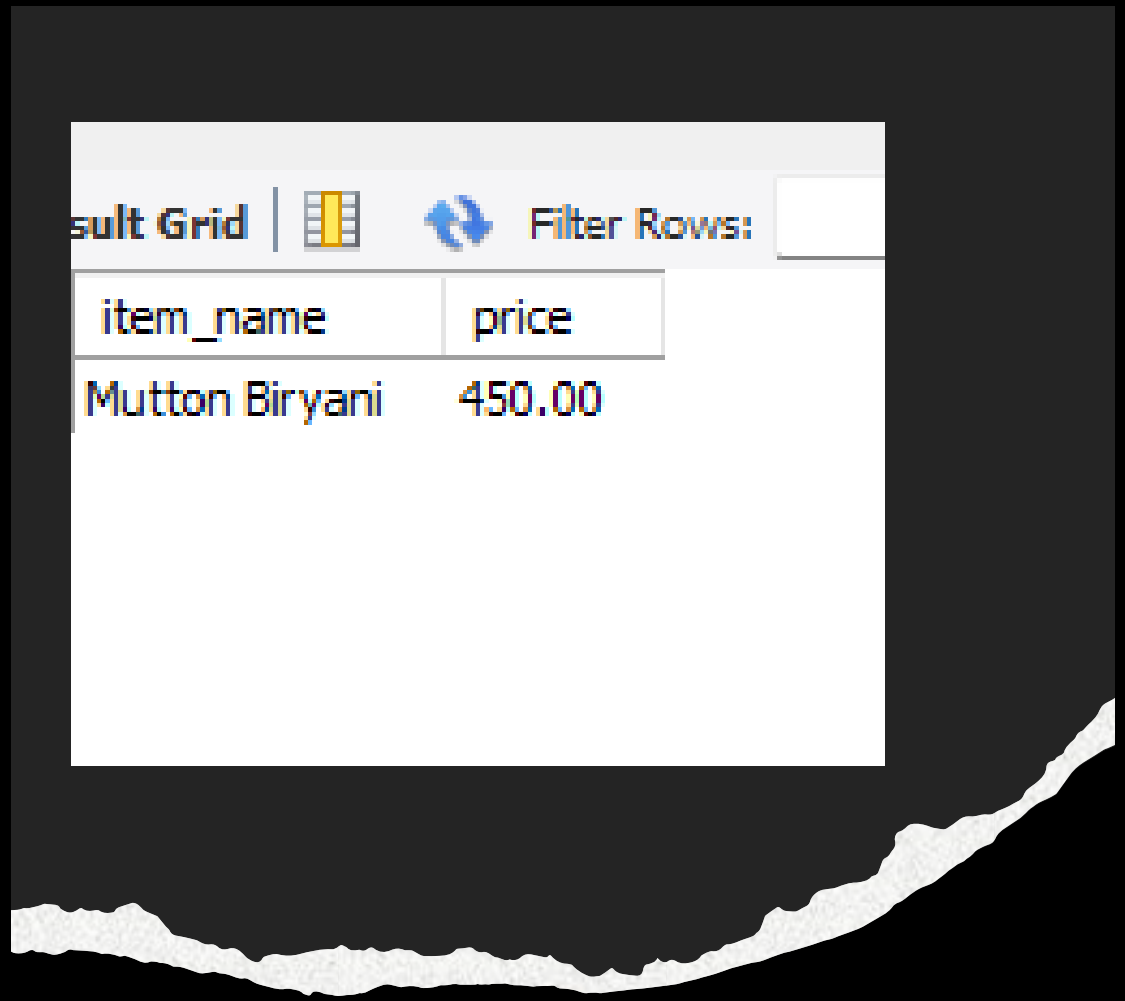


_name	total_revenue	avg_order_va
digul Thalappakatti	1570.00	523.333333
appar Chettinad	1550.00	516.666667
ari Hotel	1370.00	685.000000
ior Kuppanna	900.00	450.000000
avana Bhavan	730.00	365.000000
Hyderabad Biryani	600.00	600.000000

Scenario 8: City-wise order count and total spending

- SELECT
- u.city,
- COUNT(o.order_id) AS total_orders,
- SUM(o.total_amount) AS total_spent
- FROM users u
- LEFT JOIN orders o ON u.user_id = o.user_id
- GROUP BY u.city
- ORDER BY total_orders DESC;

city	total_orders	total_spent
Chennai	12	4860.00
Kanchipuram	2	1230.00
Chromepet	1	180.00
Krishnagiri	1	400.00
Tambaram	1	350.00
Guduvanchery	1	720.00
Mumbai	1	450.00
Perambur	1	270.00
Tirunelveli	1	140.00
Avadi	1	260.00
Pondicherry	0	NULL
Tiruvallur	0	NULL



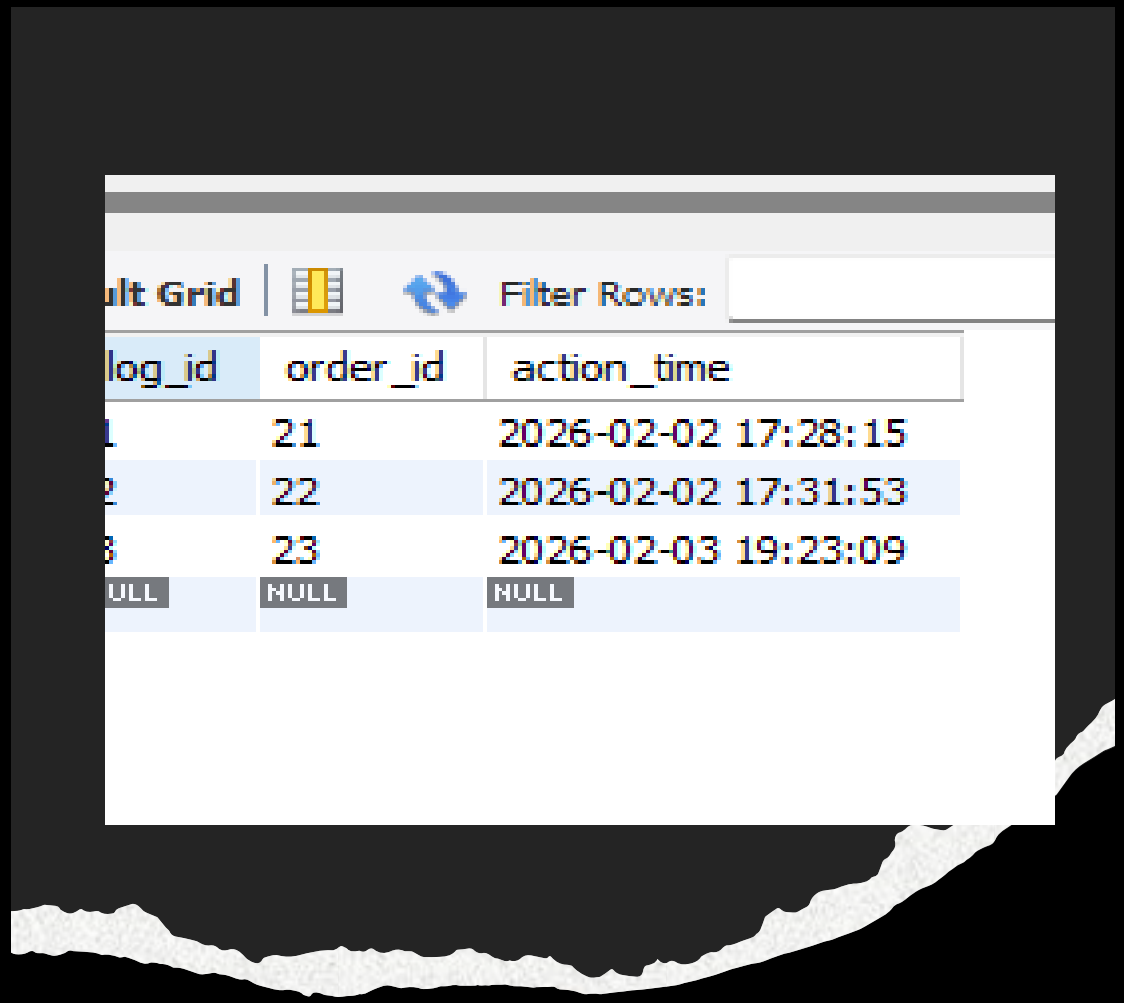
Scenario 9: Find the most expensive menu item

- SELECT
- item_name, price
- FROM menu_items
- WHERE price = (SELECT MAX(price) FROM menu_items);

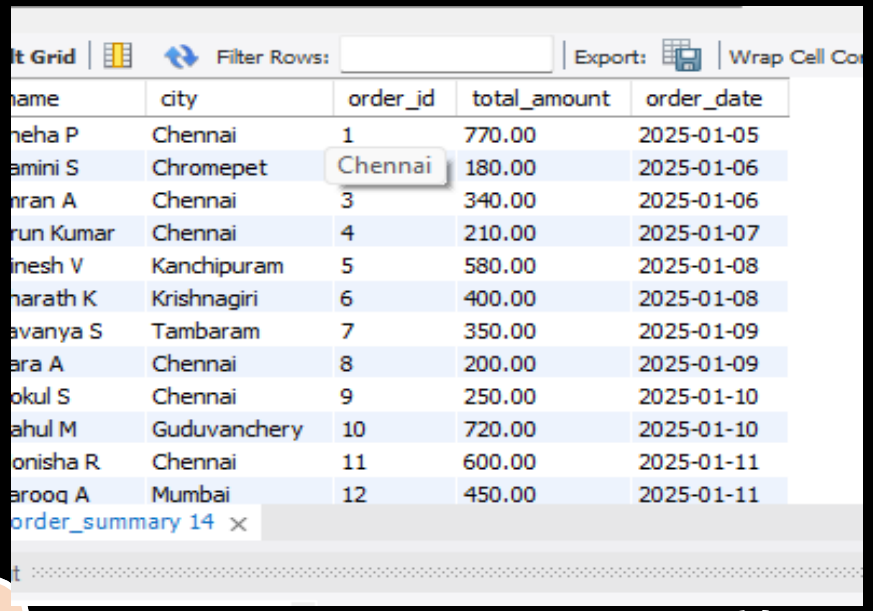
Scenario 10: Log every new order insertion

- INSERT INTO orders (user_id, res_id, total_amount, order_date)
- VALUES (9, 7, 850.00, '2026-02-17');

- SELECT * FROM order_logs;



Scenario 11: Create a view for user order summary

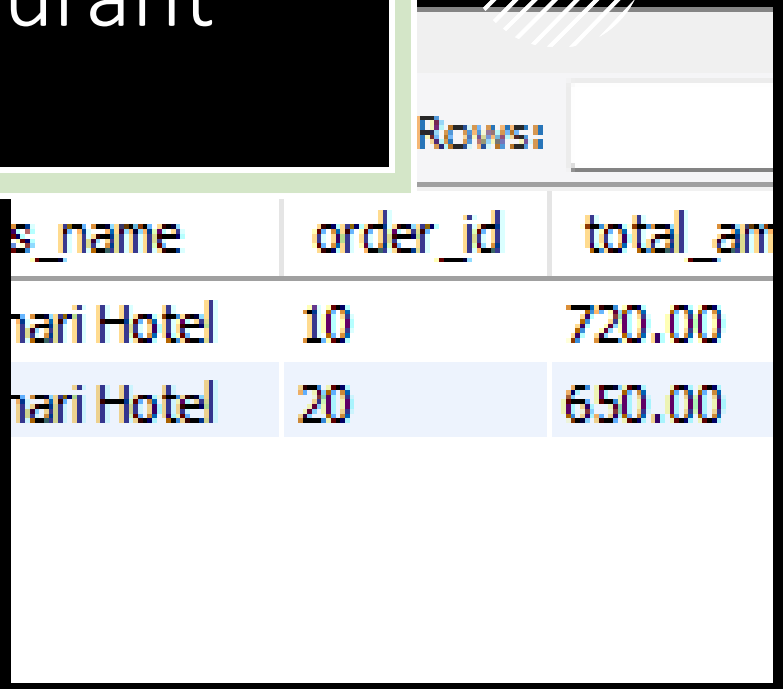


name	city	order_id	total_amount	order_date
neha P	Chennai	1	770.00	2025-01-05
amini S	Chromepet	2	180.00	2025-01-06
nran A	Chennai	3	340.00	2025-01-06
run Kumar	Chennai	4	210.00	2025-01-07
inesh V	Kanchipuram	5	580.00	2025-01-08
narath K	Krishnagiri	6	400.00	2025-01-08
avanya S	Tambaram	7	350.00	2025-01-09
ara A	Chennai	8	200.00	2025-01-09
okul S	Chennai	9	250.00	2025-01-10
ahul M	Guduvanchery	10	720.00	2025-01-10
onisha R	Chennai	11	600.00	2025-01-11
aroog A	Mumbai	12	450.00	2025-01-11

- CREATE VIEW user_order_summary AS
- SELECT
- u.name,
- u.city,
- o.order_id,
- o.total_amount,
- o.order_date
- FROM users u
- JOIN orders o ON u.user_id = o.user_id;

- SELECT * FROM user_order_summary;

Scenario 12: Stored procedure to get orders of a restaurant



- DELIMITER $$
- CREATE PROCEDURE GetRestaurantOrders(IN resName VARCHAR(100))
- BEGIN
- SELECT r.res_name, o.order_id, o.total_amount
- FROM restaurants r
- JOIN orders o ON r.res_id = o.res_id
- WHERE r.res_name = resName;
- END $$
- DELIMITER ;

- CALL GetRestaurantOrders('Buhari Hotel');

user_id	name	email	phone	city
9	Gokul S	gokul33@gmail.com	9876543242	Chennai
NULL	NULL	NULL	NULL	NULL

Scenario 13: Improve search performance using index

- CREATE INDEX idx_user_email ON users(email);
- SELECT *FROM users WHERE email = 'gokul33@gmail.com';

Thank You