

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;
```

Result Grid					Filter Rows:	Export
	order_id	user_name	total_amount	order_date		
	1	Sneha P	770.00	2025-01-05		
	2	Yamini S	180.00	2025-01-06		
	3	Imran A	340.00	2025-01-06		
	4	Arun Kumar	210.00	2025-01-07		
	5	Dinesh V	580.00	2025-01-08		
	6	Bharath K	400.00	2025-01-08		
	7	Lavanya S	350.00	2025-01-09		
	8	Zara A	200.00	2025-01-09		
	9	Gokul S	250.00	2025-01-10		
	10	Rahul M	720.00	2025-01-10		
	11	Monisha R	600.00	2025-01-11		
	12	Farooq A	450.00	2025-01-11		

Result 1 ×

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;
```

Result Grid					Filter Rows:		Export:
	name	city	order_id	total_amount			
	Akash R	Perambur	13	270.00			
	Tamil Selvan	Tirunelveli	14	140.00			
	Qasim A	Chennai	15	390.00			
	Karthik S	Chennai	16	320.00			
	Bhavani S	Chennai	17	500.00			
	Nithin K	Avadi	18	260.00			
	Ezhil V	Chennai	19	180.00			
	Waseem A	Chennai	20	650.00			
	Praveen S	Chennai	NULL	NULL			
	Xavier J	Pondicherry	NULL	NULL			
	Bala Murugan	Chennai	NULL	NULL			
	Varun K	Chennai	NULL	NULL			

Result 2 x

Result Grid



Filter Rows:

E

res_name	order_id	total_amount
Ponnusamy Hotel	8	200.00
Ponnusamy Hot		260.00
Murugan Idli Shop	9	250.00
Murugan Idli Shop	19	180.00
Buhari Hotel	10	720.00
Buhari Hotel	20	650.00
SS Hyderabad Biryani	11	600.00
Salem RR Briyani	12	450.00
Madurai Kumar Mess	NULL	NULL
Meenakshi Bhavan	NULL	NULL
Karpagambal Mess	NULL	NULL
Geetham Veg Restaurant	NULL	NULL

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;

Result 3 ✕

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 a
- INNER JOIN users b ON a.city = b.city
- WHERE a.user_id < b.user_id;

User_1	User_2	city
Bhavani S		
Ezhil V	Waseem A	Chennai
Sneha P	Praveen S	Chennai
Arun Kumar	Praveen S	Chennai
Imran A	Praveen S	Chennai
Zara A	Praveen S	Chennai
Gokul S	Praveen S	Chennai
Monisha R	Praveen S	Chennai
Qasim A	Praveen S	Chennai
Karthik S	Praveen S	Chennai
Bhavani S	Praveen S	Chennai
Ezhil V	Praveen S	Chennai

Resets all sorted columns

Result 4

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;

	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 ✕

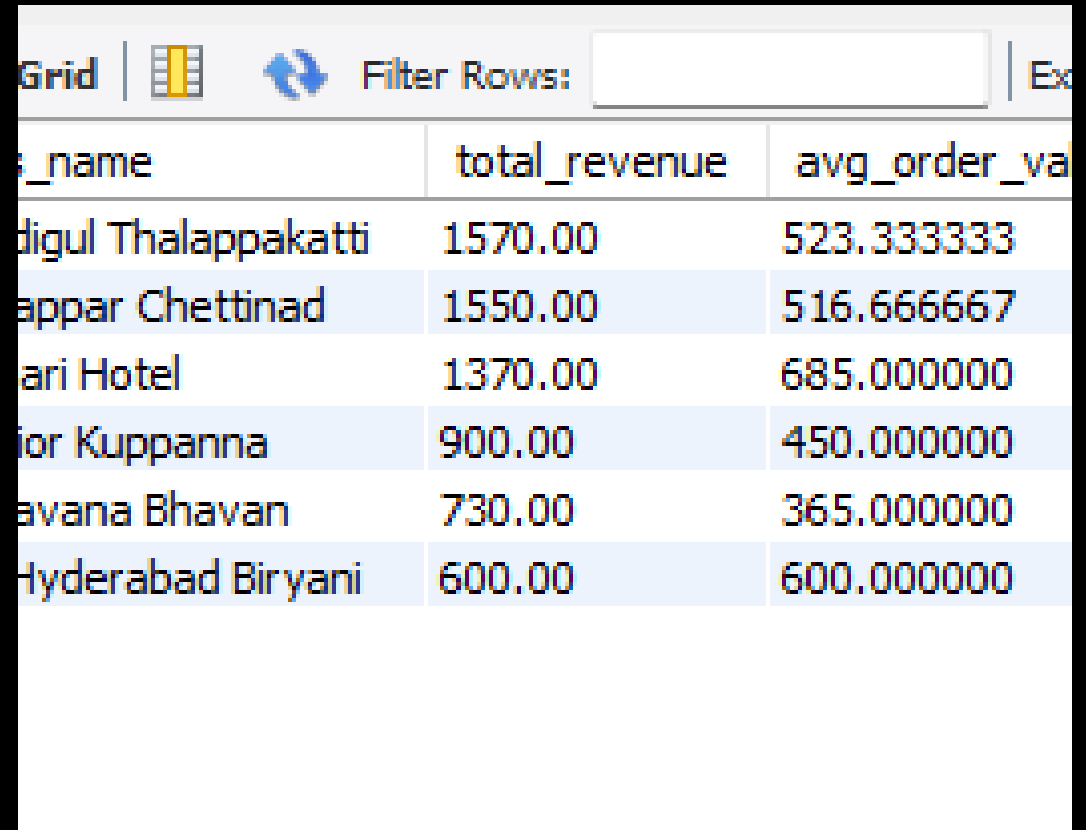
Scenario 6: Count total orders for each restaurant

- 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;



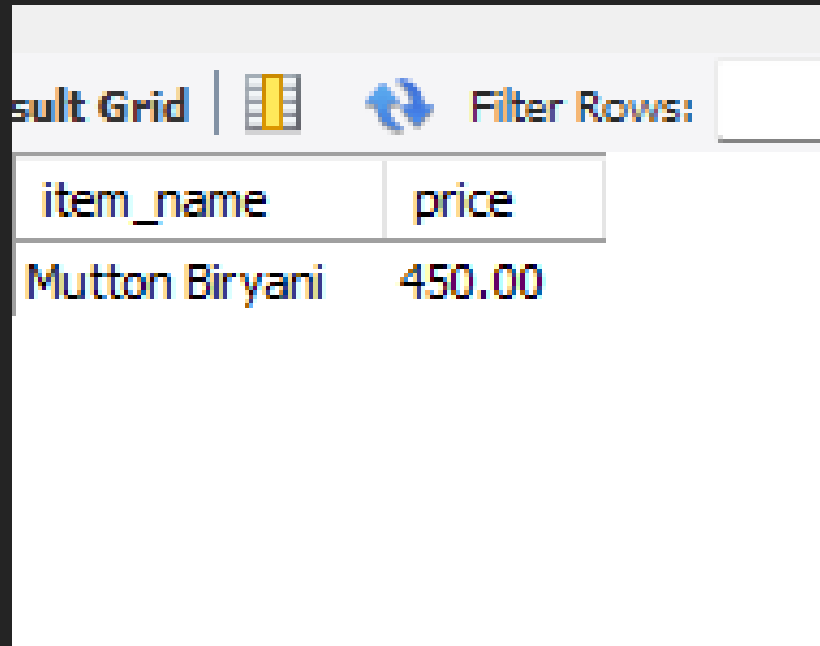
The screenshot shows a data grid interface with a toolbar at the top containing a 'Grid' button, a column selection icon, a refresh icon, and a 'Filter Rows:' input field. The table below has three columns: 'res_name', 'total_revenue', and 'avg_order_value'. It displays six rows of data, sorted by total revenue in descending order.

res_name	total_revenue	avg_order_value
Indigul Thalappakatti	1570.00	523.333333
Chappar 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



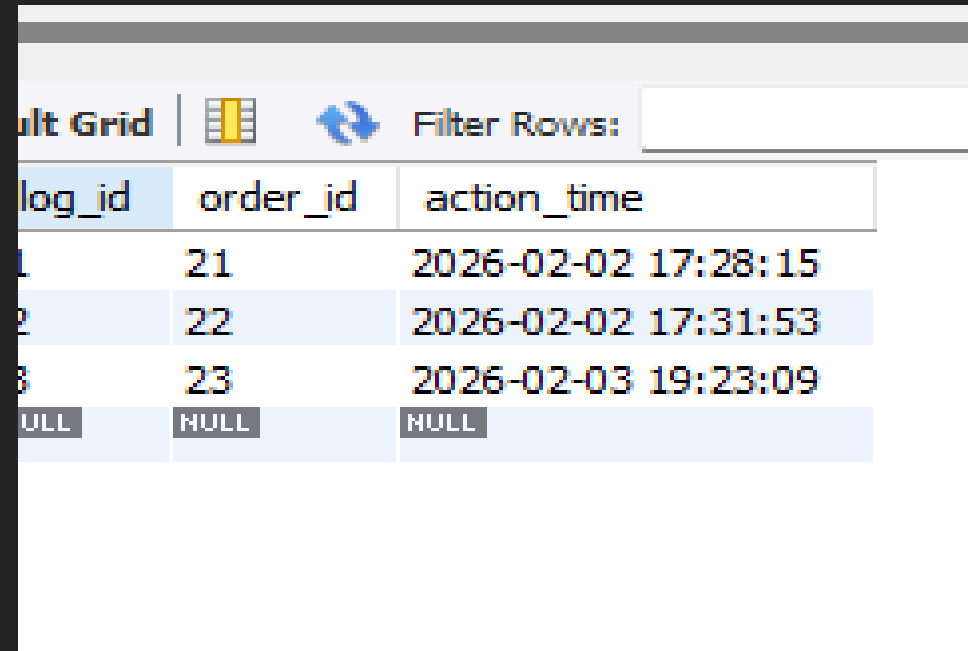
item_name	price
Mutton Biryani	450.00

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

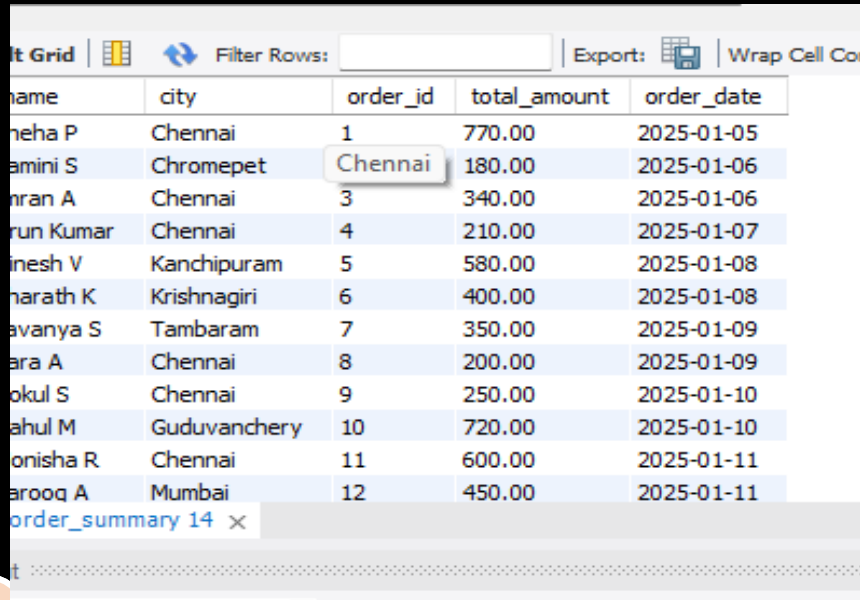
- INSERT INTO orders (user_id, res_id, total_amount, order_date)
- VALUES (9, 7, 850.00, '2026-02-17');
- SELECT * FROM order_logs;



log_id	order_id	action_time
1	21	2026-02-02 17:28:15
2	22	2026-02-02 17:31:53
3	23	2026-02-03 19:23:09
NULL	NULL	NULL

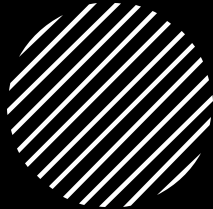


Scenario 11: Create a view for user order summary


A screenshot of a 'Table Grid' application window. It features a toolbar with icons for 'Filter Rows', 'Export', and 'Wrap Cell Content'. Below the toolbar is a table with 5 columns: 'name', 'city', 'order_id', 'total_amount', and 'order_date'. The table contains 12 rows of data. The 'city' column has a dropdown menu open, showing 'Chennai' selected. At the bottom of the window, there is a tab labeled 'order_summary 14' with a close button (x).

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
nesh V	Kanchipuram	5	580.00	2025-01-08
harath 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
arooq 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



res_name	order_id	total_amount
Buhari Hotel	10	720.00
Buhari Hotel	20	650.00

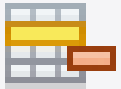
- DELIMITER \$\$
- CREATE PROCEDURE GetRestaurantOrders(IN resName VARCHAR(100))
- BEGIN
- SELECT r.res_name, o.order_id, o.total_amount
- FROM restaurants r
- JOIN orders o ON r.res_id = o.res_id
- WHERE r.res_name = resName;
- END \$\$
- DELIMITER ;
- CALL GetRestaurantOrders('Buhari Hotel');

Result Grid



Filter Rows:

Edit:



	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';`

The background is black and features several abstract geometric elements. In the top-left corner, there is a light orange semi-circle and two white zigzag lines. In the bottom-left corner, there is a light green circle. In the bottom-right corner, there is a light green semi-circle. A large black rectangle with a thin white border and a slightly offset orange border is centered on the page.

Thank You