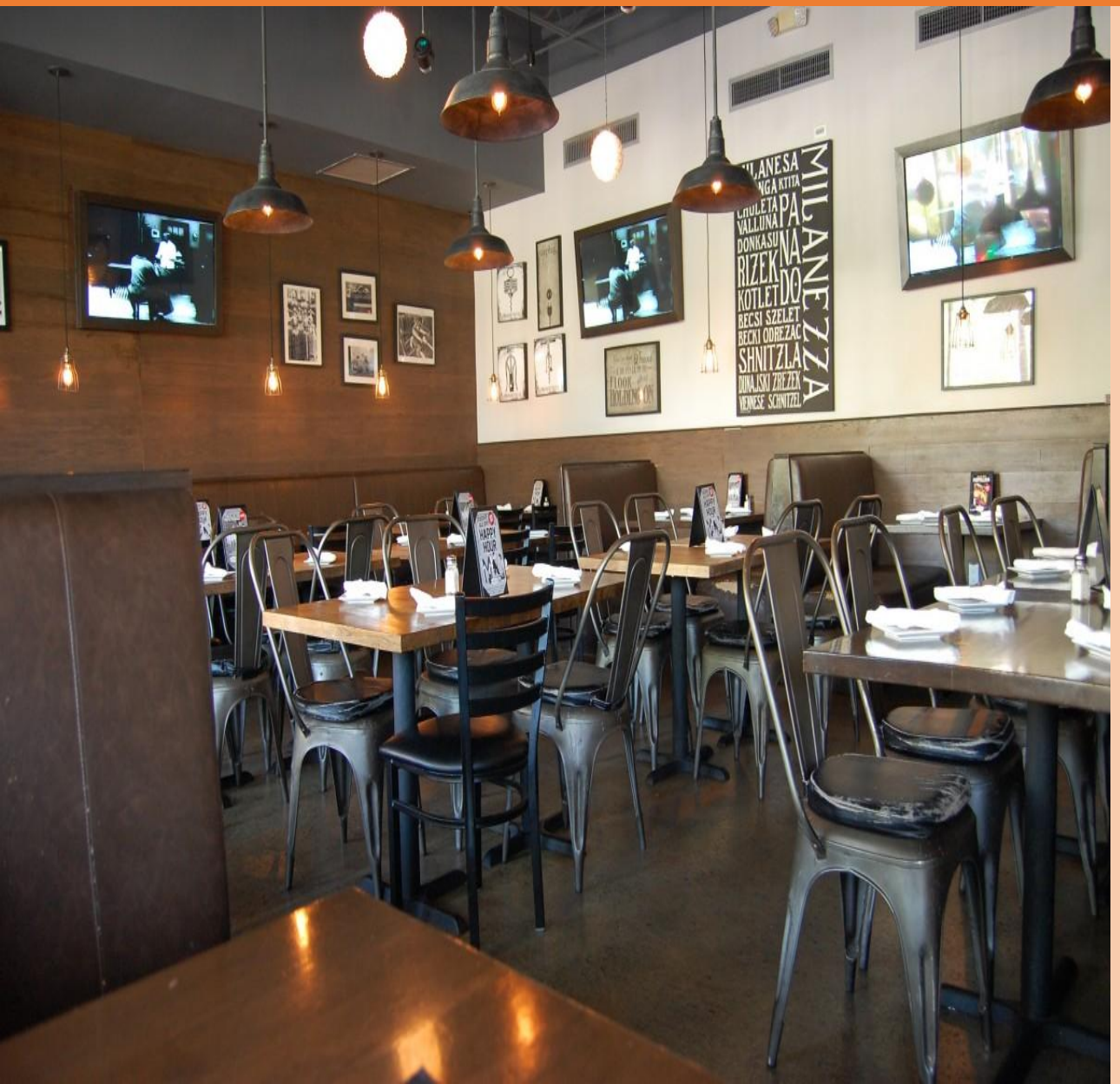
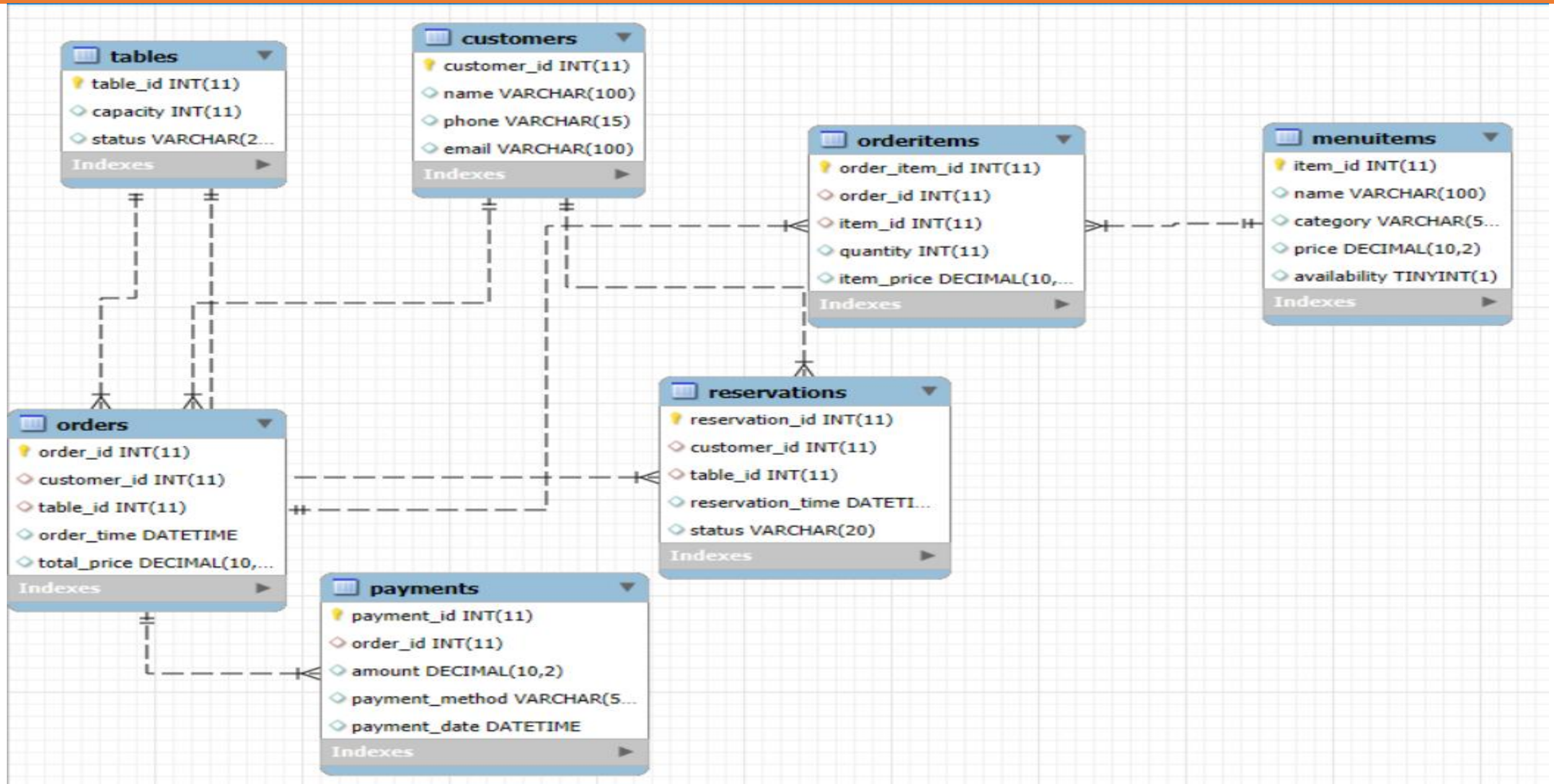
Restaurant Management System

ABSTRACT :



- This project is about creating a Restaurant Management System using SQL. The system helps manage daily restaurant tasks like booking tables, taking food orders, managing the menu, handling employees, and processing payments.
- We have built a database with tables for customers, orders, menu items, employees, and more. Using SQL, we can connect this data to find useful information, like total sales or most popular dishes. We also use stored procedures and triggers to make some tasks automatic, like updating table status or recording payments when an order is completed.
- This project helps improve the restaurant’s operations and shows how SQL can be used in real-life situations to organize and manage business data.

ER- DIAGRAM


tables
table_id INT(11)
capacity INT(11)
status VARCHAR(2...
Indexes
customers
customer_id INT(11)
name VARCHAR(100)
phone VARCHAR(15)
email VARCHAR(100)
Indexes
orderitems
order_item_id INT(11)
order_id INT(11)
item_id INT(11)
quantity INT(11)
item_price DECIMAL(10,...
Indexes
menuitems
item_id INT(11)
name VARCHAR(100)
category VARCHAR(5...
price DECIMAL(10,2)
availability TINYINT(1)
Indexes
reservations
reservation_id INT(11)
customer_id INT(11)
table_id INT(11)
reservation_time DATETI...
status VARCHAR(20)
Indexes
orders
order_id INT(11)
customer_id INT(11)
table_id INT(11)
order_time DATETIME
total_price DECIMAL(10,...
Indexes
payments
payment_id INT(11)
order_id INT(11)
amount DECIMAL(10,2)
payment_method VARCHAR(5...
payment_date DATETIME
Indexes

CONTENTS OF TABLE

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

customer_id	name	phone	email
1	Aarav Mehta	9123456780	aarav@email.com
2	Kiara Singh	9988776655	kiara@email.com
3	Rohan Patel	9876543211	rohan@email.com
4	Simran Kaur	9811122233	simran@email.com
5	Rahul Verma	9331122233	rahul@email.com
6	Ishita Das	9456781234	ishita@email.com
7	Yash Malhotra	9551236789	yash@email.com
8	Anaya Reddy	9789012345	anaya@email.com
9	Manav Joshi	9623412789	manav@email.com
10	Tanya Desai	9881234567	tanya@email.com
11	Dev Sharma	9111222333	dev@email.com
12	Pooja Joshi	9223344556	pooja@email.com
13	Amit Saxena	9345678910	amit@email.com
14	Riya Banerjee	9789011223	riya@email.com
15	Kabir Singh	9561237890	kabir@email.com
16	Nisha Menon	9123456000	nisha@email.com
17	Sarthak Rao	9678901234	sarthak@email.com
18	Ira Kapoor	9887766554	ira@email.com
19	Mohit Ahuja	9345674321	mohit@email.com
20	Tanvi Iyer	9445566778	tanvi@email.com
21	Parth Yadav	9834452211	parth@email.com
22	Jaya Mishra	9556677880	jaya@email.com
23	Karan Oberoi	9483345678	karan@email.com
24	Snehal Shah	9123451111	snehal@email.com

Select * from Customers;

CONTENTS OF TABLE

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

table_id	capacity	status
1	2	Available
2	4	Occupied
3	6	Reserved
4	2	Available
5	4	Available
6	6	Occupied
7	2	Reserved
8	4	Available
9	6	Available
10	2	Occupied
11	2	Available
12	4	Reserved
13	6	Occupied
14	2	Reserved
15	4	Available
16	6	Available
17	4	Occupied
18	2	Available
19	6	Reserved
20	2	Occupied
21	4	Available
22	6	Available
23	2	Reserved
24	4	Occupied

Select * from Tables;

CONTENTS OF TABLE

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

item_id	name	category	price	availability
1	Paneer Butter Masala	Main Course	9.50	1
2	Tomato Soup	Starter	4.00	1
3	Mango Lassi	Beverage	2.50	1
4	Veg Biryani	Main Course	8.25	1
5	Tandoori Roti	Bread	1.50	1
6	Masala Dosa	Main Course	6.75	1
7	Cold Coffee	Beverage	3.25	1
8	Spring Rolls	Starter	5.00	1
9	Butter Naan	Bread	2.00	1
10	Gulab Jamun	Dessert	3.00	1
11	Dal Tadka	Main Course	7.50	1
12	Vegetable Soup	Starter	3.50	1
13	Masala Chaas	Beverage	2.00	1
14	Chole Bhature	Main Course	8.00	1
15	Butter Naan	Bread	2.25	1
16	Aloo Paratha	Bread	4.00	1
17	Mango Lassi	Beverage	3.25	1
18	Paneer Tikka	Starter	6.50	1
19	Methi Thepla	Bread	3.75	1
20	Gulab Jamun	Dessert	2.50	1
21	Masala Pulao	Main Course	7.00	1
22	Veg Pakora	Starter	4.25	1
23	Cold Coffee	Beverage	3.00	1
24	Veg Korma	Main Course	8.50	1

```
select * from MenuItems;
```

CONTENTS OF TABLE

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

employee_id	name	role	shift_time
1	Neha Kapoor	Waiter	Morning
2	Aditya Roy	Chef	Evening
3	Sneha Sharma	Manager	Full Day
4	Vikram Rana	Waiter	Evening
5	Priya Nair	Chef	Morning
6	Kunal Sethi	Waiter	Night
7	Divya Chauhan	Host	Full Day
8	Ravi Kumar	Waiter	Morning
9	Meena Pillai	Chef	Night
10	Ankit Gupta	Cleaner	Night
11	Ravi Kapoor	Waiter	Morning
12	Amit Sharma	Chef	Evening
13	Pooja Iyer	Manager	Full Day
14	Sanjay Verma	Waiter	Night
15	Neha Patel	Chef	Morning
16	Vikram Joshi	Cleaner	Night
17	Divya Kapoor	Host	Full Day
18	Priya Nair	Waiter	Morning
19	Ankit Mehta	Chef	Evening
20	Rohan Sethi	Manager	Full Day
21	Tanu Sharma	Waiter	Night
22	Sonal Reddy	Chef	Morning
23	Manav Verma	Cleaner	Night
24	Ayesha Khan	Host	Full Day

Employees 20

```
select * from Employees;
```

CONTENTS OF TABLE

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

order_id	customer_id	table_id	order_time	total_price
1	1	2	2025-05-01 18:10:00	16.00
2	2	3	2025-05-01 18:25:00	12.75
3	3	5	2025-05-01 18:40:00	18.50
4	4	1	2025-05-01 19:00:00	11.00
5	5	4	2025-05-01 19:15:00	22.75
6	6	6	2025-05-01 19:30:00	10.00
7	7	7	2025-05-01 19:50:00	15.25
8	8	8	2025-05-01 20:10:00	14.00
9	9	9	2025-05-01 20:30:00	17.75
10	10	10	2025-05-01 20:50:00	19.00
11	11	1	2025-05-02 12:00:00	13.25
12	12	2	2025-05-02 12:20:00	17.50
13	13	3	2025-05-02 12:40:00	11.75
14	14	4	2025-05-02 13:00:00	14.00
15	15	5	2025-05-02 13:30:00	20.25
16	16	6	2025-05-02 13:45:00	15.00
17	17	7	2025-05-02 14:00:00	18.50
18	18	8	2025-05-02 14:20:00	16.75
19	19	9	2025-05-02 14:45:00	22.00
20	20	10	2025-05-02 15:10:00	12.00
21	1	1	2025-05-03 12:15:00	14.50
22	2	2	2025-05-03 12:30:00	10.25
23	3	3	2025-05-03 12:45:00	19.00
24	4	4	2025-05-03 13:00:00	17.75

```
select * from Orders;
```

CONTENTS OF TABLE

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

order_item_id	order_id	item_id	quantity	item_price
1	1	1	1	9.50
2	1	3	1	2.50
3	1	2	1	4.00
4	2	4	1	8.25
5	2	5	2	1.50
6	3	6	1	6.75
7	3	7	1	3.25
8	3	10	2	3.00
9	4	8	1	5.00
10	4	9	2	2.00
11	11	1	1	9.50
12	11	5	2	1.50
13	12	2	1	4.00
14	12	3	1	2.50
15	13	4	1	8.25
16	13	10	1	3.00
17	14	6	1	6.75
18	14	7	1	3.25
19	15	1	2	9.50
20	15	9	2	2.00
21	16	8	1	5.00
22	16	10	1	3.00
23	17	4	1	8.25
24	17	3	2	2.50

```
select * from OrderItems;
```

CONTENTS OF TABLE

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

reservation_id	customer_id	table_id	reservation_time	status
15	5	5	2025-05-05 18:00:00	Completed
16	6	6	2025-05-05 18:30:00	Confirmed
17	7	7	2025-05-05 19:00:00	Pending
18	8	8	2025-05-05 19:30:00	Confirmed
19	9	9	2025-05-06 18:00:00	Cancelled
20	10	10	2025-05-06 18:30:00	Completed
21	1	1	2025-05-06 19:00:00	Confirmed
22	2	2	2025-05-06 19:30:00	Confirmed
23	3	3	2025-05-07 18:00:00	Pending
24	4	4	2025-05-07 18:30:00	Completed
25	5	5	2025-05-07 19:00:00	Confirmed
26	6	6	2025-05-07 19:30:00	Pending
27	7	7	2025-05-08 18:00:00	Completed
28	8	8	2025-05-08 18:30:00	Confirmed
29	9	9	2025-05-08 19:00:00	Cancelled
30	10	10	2025-05-08 19:30:00	Confirmed
NULL	NULL	NULL	NULL	NULL

```
select * from Reservations;
```

CONTENTS OF TABLE

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

payment_id	order_id	amount	payment_method	payment_date
1	1	16.00	Card	2025-05-01 18:20:00
2	2	12.75	Cash	2025-05-01 18:35:00
3	3	18.50	UPI	2025-05-01 18:55:00
4	4	11.00	Cash	2025-05-01 19:10:00
5	5	22.75	Card	2025-05-01 19:25:00
6	6	10.00	UPI	2025-05-01 19:45:00
7	7	15.25	Card	2025-05-01 20:00:00
8	8	14.00	Cash	2025-05-01 20:20:00
9	9	17.75	Card	2025-05-01 20:40:00
10	10	19.00	UPI	2025-05-01 21:00:00
11	11	13.25	Card	2025-05-02 00:00:00
12	12	17.50	Cash	2025-05-02 00:00:00
13	13	11.75	UPI	2025-05-02 00:00:00
14	14	14.00	Card	2025-05-02 00:00:00
15	15	20.25	UPI	2025-05-02 00:00:00
16	16	15.00	Cash	2025-05-02 00:00:00
17	17	18.50	Card	2025-05-02 00:00:00
18	18	16.75	UPI	2025-05-02 00:00:00
19	19	22.00	Cash	2025-05-02 00:00:00
20	20	12.00	UPI	2025-05-02 00:00:00
21	21	14.50	Card	2025-05-03 00:00:00
22	22	10.25	Cash	2025-05-03 00:00:00
23	23	19.00	UPI	2025-05-03 00:00:00
24	24	17.75	Card	2025-05-03 00:00:00

```
select * from Payments;
```

SUBQUERY

A subquery is a query that appears inside another query statement

SUBQUERY

Result Grid | Filter Rows:

name	order_id	total_price
Mohit Ahuja	19	22.00
Kabir Singh	15	20.25
Aarav Mehta	1	16.00
Dev Sharma	11	13.25

1. Find Customers who ordered the most expensive item

select Customers.name, Orders.order_id, Orders.total_price from Customers, Orders, OrderItems WHERE Customers.customer_id = Orders.customer_id and Orders.order_id = OrderItems.order_id and OrderItems.item_id = (select item_id FROM MenuItems where price = (select max(price) from MenuItems))order by Orders.total_price desc;

This query identifies **customers** who have ordered the **most expensive item** available on the menu

SUBQUERY

Result Grid | Filter Rows:

name	price
Butter Naan	2.00
Cold Coffee	3.25
Gulab Jamun	3.00
Mango Lassi	2.50
Masala Dosa	6.75
Paneer Butter Masala	9.50
Spring Rolls	5.00
Tandoori Roti	1.50
Tomato Soup	4.00
Veg Biryani	8.25

2. Find Menu Items that were ordered more than once

select MenuItems.name, MenuItems.price from MenuItems where MenuItems.item_id IN (select item_id from OrderItems group by item_id having count(*) > 1)order by MenuItems.name;

This finds the **menu items** that were ordered more than **once**, based on the orderitem table.

SUBQUERY

Result Grid | Filter Rows: | Export:

name	order_id	total_price
Rahul Verma	5	22.75
Mohit Ahuja	19	22.00
Rahul Verma	25	21.50
Kabir Singh	15	20.25
Rohan Patel	23	19.00
Tanya Desai	10	19.00
Sarthak Rao	17	18.50
Rohan Patel	3	18.50
Anaya Reddy	28	18.25
Manav Joshi	9	17.75
Simran Kaur	24	17.75
Pooja Joshi	12	17.50
Ira Kapoor	18	16.75
Tanya Desai	30	16.75
Aarav Mehta	1	16.00

3. Find Customers who made orders with a total greater than ₹20

select Customers.name, Orders.order_id, Orders.total_price from Customers, Orders where Customers.customer_id = Orders.customer_id and Orders.total_price > (select avg(total_price) from Orders)order by Orders.total_price desc;

This query lists the **customers** who placed orders where the **total price** is greater than the **average price** of all orders.

JOINS

Joins indicate how SQL Server should use data from one table to select the rows in another table

JOINS

Result Grid | Filter Rows: | Exp

order_id	customer_name	table_id
1	Aarav Mehta	2
2	Kiara Singh	3
3	Rohan Patel	5
4	Simran Kaur	1
5	Rahul Verma	4
6	Ishita Das	6
7	Yash Malhotra	7
8	Anaya Reddy	8
9	Manav Joshi	9
10	Tanya Desai	10
11	Dev Sharma	1
12	Pooja Joshi	2
13	Amit Saxena	3
14	Riya Banerjee	4
15	Kabir Singh	5
16	Nisha Menon	6
17	Sarthak Rao	7
18	Ira Kapoor	8
19	Mohit Ahuja	9
20	Tanvi Iyer	10
21	Aarav Mehta	1
22	Kiara Singh	2
23	Rohan Patel	3

1. INNER JOIN: List all orders with customer names and table numbers

select Orders.order_id, Customers.name as customer_name, Tables.table_idfrom Ordersinner join Customers on Orders.customer_id = Customers.customer_idinner join Tables on Orders.table_id = Tables.table_id;

Shows all orders that have valid customer and table information. Filters out unmatched data.

JOINS

Result Grid | Filter Rows:

name	reservation_time	status
Aarav Mehta	2025-05-01 17:00:00	Reserved
Ishita Das	2025-05-01 19:30:00	Reserved
Rohan Patel	2025-05-04 20:00:00	Reserved
Yash Malhotra	2025-05-05 19:00:00	Reserved
Rohan Patel	2025-05-07 18:00:00	Reserved
Yash Malhotra	2025-05-08 18:00:00	Reserved

2.Show reservations with customer names and table status

select Customers.name, Reservations.reservation_time, Tables.statusfrom Reservationsinner join Customers on Reservations.customer_id = Customers.customer_idinner join Tables on Reservations.table_id = Tables.table_idwhere Tables.status = 'Reserved';

Displays reservations where the **table is marked as ‘Reserved’**, along with the **customer name** and **reservation time**.

JOINS

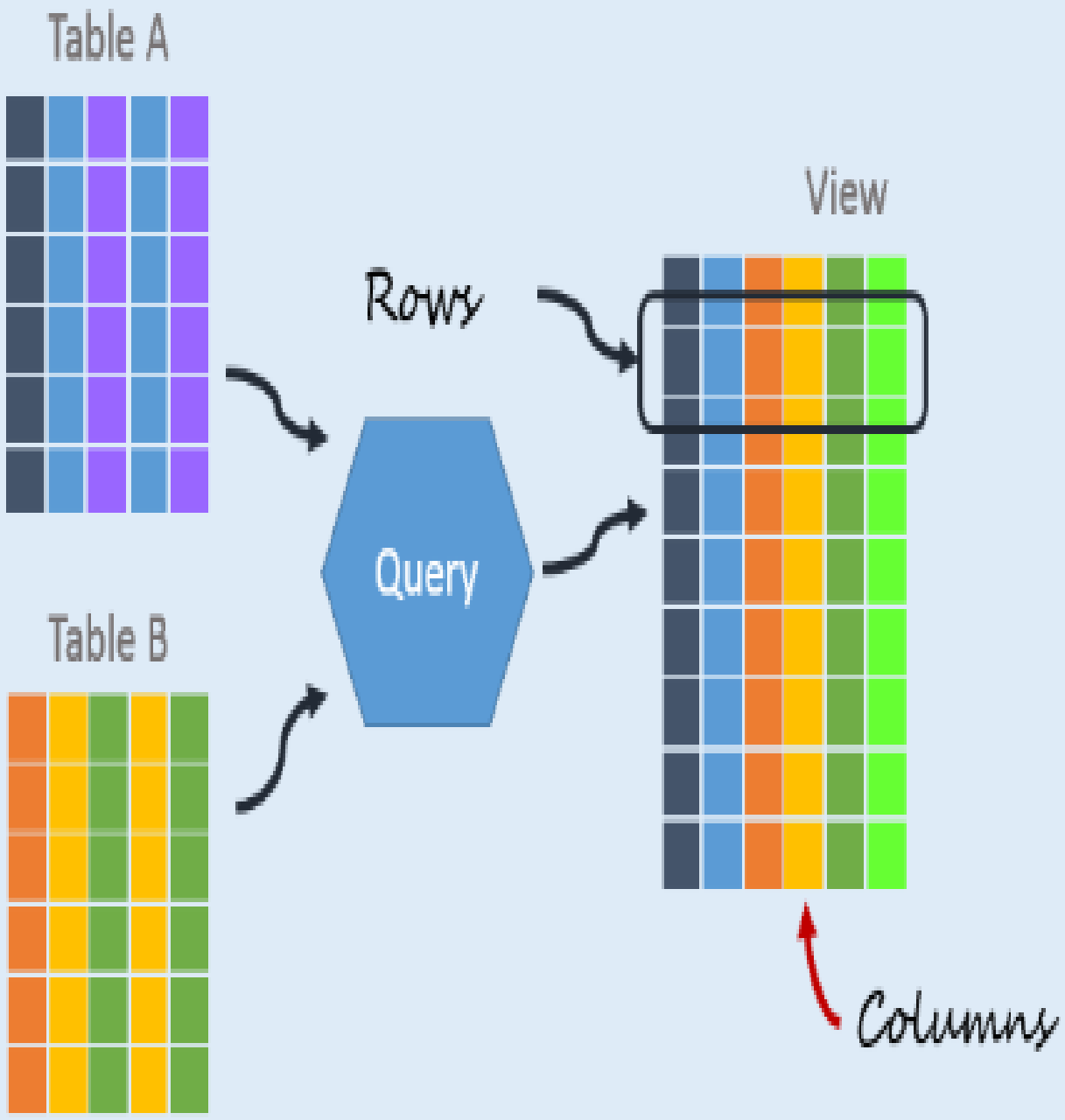
Result Grid | Filter Rows:

name	order_id	quantity
Gulab Jamun	3	2
Spring Rolls	4	1
Butter Naan	4	2
Paneer Butter Masala	11	1
Tandoori Roti	11	2
Tomato Soup	12	1
Mango Lassi	12	1
Veg Biryani	13	1
Gulab Jamun	13	1
Masala Dosa	14	1
Cold Coffee	14	1
Paneer Butter Masala	15	2
Butter Naan	15	2
Spring Rolls	16	1
Gulab Jamun	16	1
Veg Biryani	17	1
Mango Lassi	17	2
Masala Dosa	18	2
Tomato Soup	18	1
Paneer Butter Masala	19	1
Gulab Jamun	19	2
Cold Coffee	20	1
Tandoori Roti	20	3

3.Show all menu items and any order they appeared in

select MenuItems.name, OrderItems.order_id, OrderItems.quantity from MenuItems left join OrderItems on MenuItems.item_id = OrderItems.item_id where OrderItems.order_id is not null;

Lists all **menu items** that have been **ordered at least once**. Menu items not yet ordered are excluded.



VIEW

View is a virtual table based on the result-set of an SQL statement

VIEW

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

customer_name	order_id	total_price	order_time
Aarav Mehta	1	16.00	2025-05-01 18:10:00
Kiara Singh	2	12.75	2025-05-01 18:25:00
Rohan Patel	3	18.50	2025-05-01 18:40:00
Simran Kaur	4	11.00	2025-05-01 19:00:00
Rahul Verma	5	22.75	2025-05-01 19:15:00
Ishita Das	6	10.00	2025-05-01 19:30:00
Yash Malhotra	7	15.25	2025-05-01 19:50:00
Anaya Reddy	8	14.00	2025-05-01 20:10:00
Manav Joshi	9	17.75	2025-05-01 20:30:00
Tanya Desai	10	19.00	2025-05-01 20:50:00
Dev Sharma	11	13.25	2025-05-02 12:00:00
Pooja Joshi	12	17.50	2025-05-02 12:20:00
Amit Saxena	13	11.75	2025-05-02 12:40:00
Riya Banerjee	14	14.00	2025-05-02 13:00:00
Kabir Singh	15	20.25	2025-05-02 13:30:00
Nisha Menon	16	15.00	2025-05-02 13:45:00
Sarthak Rao	17	18.50	2025-05-02 14:00:00
Ira Kapoor	18	16.75	2025-05-02 14:20:00
Mohit Ahuja	19	22.00	2025-05-02 14:45:00
Tanvi Iyer	20	12.00	2025-05-02 15:10:00
Aarav Mehta	21	14.50	2025-05-03 12:15:00
Kiara Singh	22	10.25	2025-05-03 12:30:00
Rohan Patel	23	19.00	2025-05-03 12:45:00
Simran Kaur	24	17.75	2025-05-03 13:00:00
Rahul Verma	25	21.50	2025-05-03 13:15:00

Customer Order Summary

```
create or replace  view  CustomerOrderSummary asselect
Customers.name as
customer_name,Orders.order_id,Orders.total_price,
Orders.order_timefrom Customersjoin Orders on
Customers.customer_id = Orders.customer_id;
select * from CustomerOrderSummary;
```

Quickly see each customer’s orders, total amount, and time.

VIEW

Result Grid | Filter Rows:

name	times_ordered
Butter Naan	2
Cold Coffee	3
Gulab Jamun	4
Mango Lassi	3
Masala Dosa	3
Paneer Butter Masala	4
Spring Rolls	2
Tandoori Roti	3
Tomato Soup	3
Veg Biryani	3

View: Popular Menu Items (Ordered More Than Once)

```
create or replace view PopularItems as select
MenuItems.name,count(OrderItems.item_id) as times_ordered
from MenuItems
join OrderItems on MenuItems.item_id =
OrderItems.item_idgroup by MenuItems.namehaving
count(OrderItems.item_id) > 1;
select * from PopularItems;
```

Identify best-selling or frequently ordered menu items.

VIEW

Result Grid | Filter Rows:

employee_name	role	shift_time
Amit Sharma	Chef	Evening
Ankit Mehta	Chef	Evening
Vikram Rana	Waiter	Evening
Meera Bhat	Chef	Evening
Aditya Roy	Chef	Evening
Pooja Iyer	Manager	Full Day
Divya Kapoor	Host	Full Day
Rajeev Shah	Manager	Full Day
Sneha Sharma	Manager	Full Day
Divya Chauhan	Host	Full Day
Rohan Sethi	Manager	Full Day
Ayesha Khan	Host	Full Day
Tanvi Desai	Waiter	Morning
Priya Nair	Waiter	Morning
Neha Patel	Chef	Morning
Neha Kapoor	Waiter	Morning
Ravi Kapoor	Waiter	Morning
Ravi Kumar	Waiter	Morning
Nisha Malhotra	Chef	Morning
Priya Nair	Chef	Morning
Sonal Reddy	Chef	Morning
Karan Ahuja	Waiter	Night
Manav Verma	Cleaner	Night
Tanu Sharma	Waiter	Night
Vikram Joshi	Cleaner	Night

Employee Shift Overview

create or replace view EmployeeShiftView as select name as employee_name,role, shift_time from Employees order by shift_time;
select * from EmployeeShiftView;

View all employees organized by their shift for easy scheduling.

Conclusion:

.Identified customers who ordered the **most expensive menu item.**

. Found menu items that were **ordered more than once**, showing how frequently certain items are chosen by customers.

. Listed customers who placed **high-value orders**, meaning their total order price was **above the average** of all orders.

. Displays only **orders with matching customer and table details**, removing any incomplete or unmatched records.

. Shows **reservations with 'Reserved' tables**, including the **customer's name** and **reservation time.**

. Displays **menu items that were ordered at least once**, leaving out items that haven’t been ordered yet.

. Provides a quick view of **each customer’s orders**, including the **total amount** and **order time**.

. Highlights **popular menu items** that were **ordered most often** by customers.

. Organizing employees by shift in a table format helps you visually sort them for easy reference when planning schedules. Simple sorting and filtering features can further streamline this process.

Prepared by Priyanka Jaiswal
From ITVedant

Thank you