# RESTAURANT DATABASE

## INTRODUCTION

The restaurant ordering system is very important for a restaurant to be successful because it affects how happy customers are and how much money the restaurant makes. The old way of writing orders on paper can be slow and mistakes can happen, which can make customers upset and cause the restaurant to lose money. To solve these problems, our SQL project on the restaurant ordering system aims to provide a comprehensive solution to manage the restaurant's ordering process efficiently and accurately.

We made a plan (called a *schema*) that uses different tables to store information about customers, their orders, menu items, and staff.

The **customer table** stores important information like their name, address, phone number and email.
The **menu item table** has information about all the food and drinks the restaurant serves, like their names and prices.
The **orders table** is used to store data on customer orders, which include order ID, order date, customer ID and total cost of the order.
The **staff table** stores data on the restaurant staff, including staff ID, name, position, hourly rate, and hire date.

Our SQL code includes create table statements for each table, which define the table's structure by specifying its attributes, such as the data types, size, and constraints that need to be followed.

The tables' *primary keys* are used to ensure data integrity and to link the tables using *foreign keys.*

There is a *one-to-many* relationship between the "*customer_id*" column in the "**customer**" table and the "*customer_id*" column in the "**order**" table.
This means that each customer can have many orders placed by them, but each order can only belong to one customer. This relationship is established through the use of foreign keys, where the "*customer_id"* column in the "**order**" table references the *"customer_id"* column in the "**customer**" table.

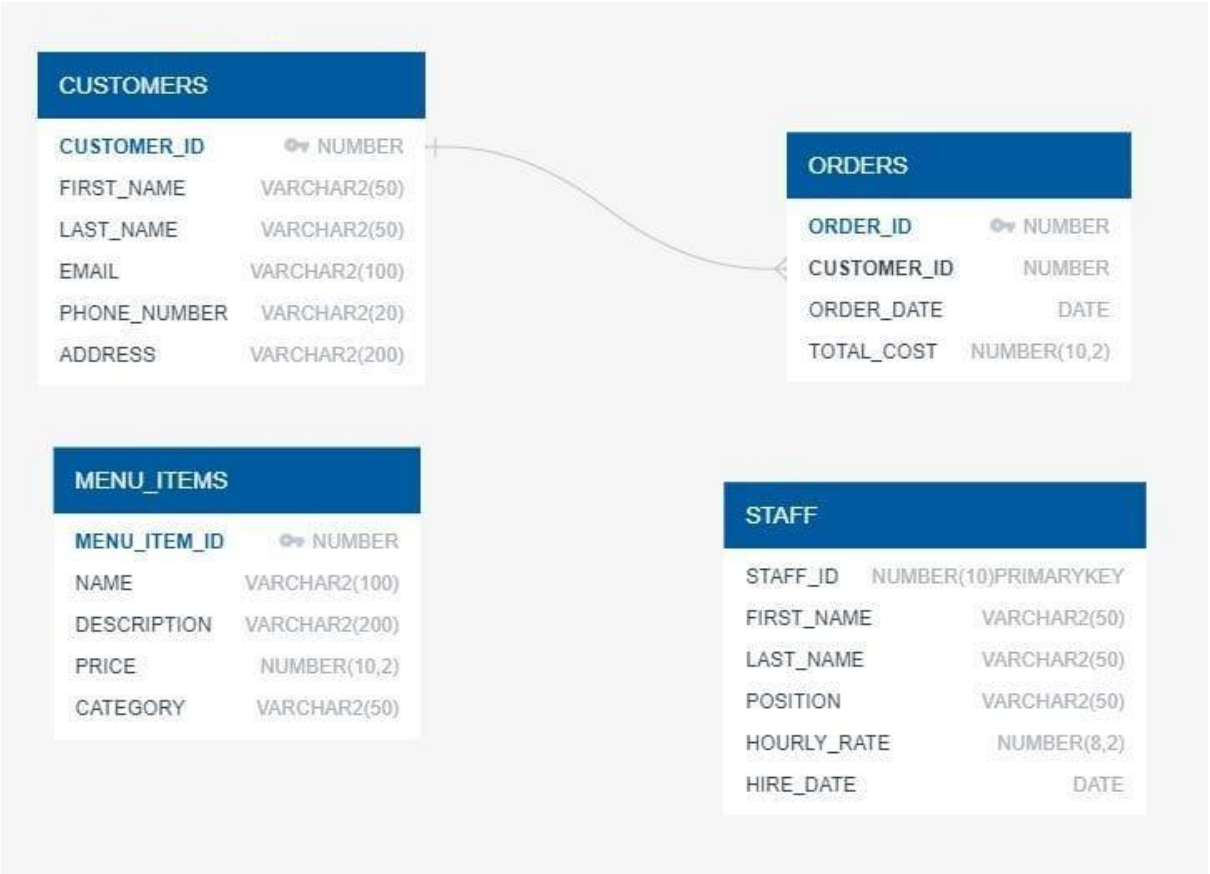
To sum up, our restaurant ordering SQL project is a helpful and easy-to-use solution that helps restaurants manage their orders, menu items, and staff with accuracy and ease. The project can be adjusted as needed, with tables, features, and rules being added or removed with ease. This flexible design provides a practical and economical solution for managing restaurant operations and increasing customer happiness and income.

# ER DIAGRAM

An entity-relationship diagram (**ERD**) is a *visual* representation of entities, attributes, and relationships between them in a database.

ERDs are useful for visualizing and designing the structure of a database. By using an ERD, we can get a better understanding of the relationships between the different entities in our database and ensure that they are properly defined and optimized for efficient querying and data retrieval.

Below is an E-R Diagram of our *database*



## SQL Commands are mainly categorized into the following categories

- 👉 Data Definition Language ( **DDL**)
  Keywords : CREATE, DROP, ALTER, TRUNCATE

- 👉 Data Manipulation Language ( **DML** )
  Keywords : INSERT, UPDATE, DELETE

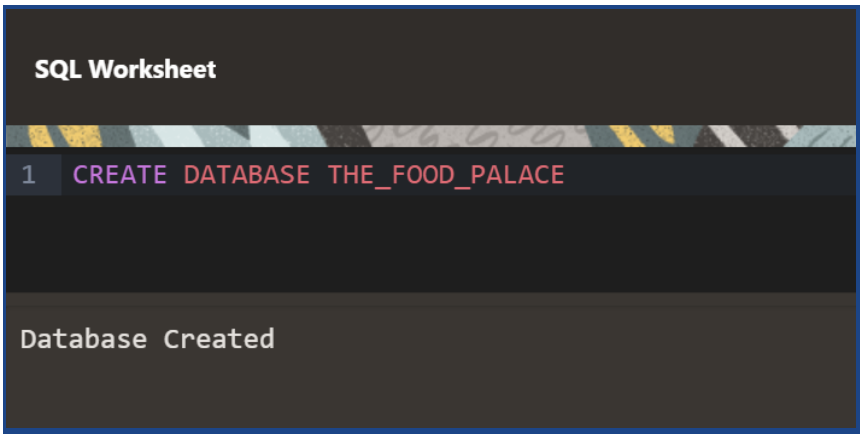
- 👉 Data Query Language ( **DQL** )
  Keyword : SELECT

- 👉 Data Control Language ( **DCL** )
  Keywords : GRANT, REVOKE

- 👉 Transaction Control Language ( **TCL** )
  Keywords : COMMIT, ROLLBACK

My project demonstrates how these keywords are used for various purposes

To begin, I will create a *database* for our restaurant, The Food Palace



## Creating tables :

### 👧 Customers

SQL Worksheet

```sql
CREATE TABLE Customers (
   customer_id NUMBER PRIMARY KEY,
   first_name VARCHAR2(50),
   last_name VARCHAR2(50),
   email VARCHAR2(100),
   phone_number VARCHAR2(20),
   address VARCHAR2(200)
);
```

```
Table created.
```

### 🧾 Orders

SQL Worksheet

```sql
CREATE TABLE Orders (
   order_id NUMBER PRIMARY KEY,
   customer_id NUMBER,
   order_date DATE,
   total_cost NUMBER(10,2),
   CONSTRAINT fk_customer_id FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);
```

```
Table created.
```

### 📕 Menu Items

SQL Worksheet

```sql
CREATE TABLE Menu_Items (
   menu_item_id NUMBER PRIMARY KEY,
   name VARCHAR2(100),
   description VARCHAR2(200),
   price NUMBER(10,2),
   category VARCHAR2(50)
);
```

```
Table created.
```

### 👷 Staff Info

**SQL Worksheet**

```sql
CREATE TABLE Staff (
    staff_id NUMBER(5) PRIMARY KEY,
    first_name VARCHAR2(50),
    last_name VARCHAR2(50),
    position VARCHAR2(50),
    hourly_rate NUMBER(8, 2),
    hire_date DATE
);
```

```
Table created.
```

---

## Inserting data in tables :

### ➔ Customers

**SQL Worksheet**

```sql
INSERT INTO Customers (customer_id, first_name, last_name, email, phone_number, address)
VALUES (1, 'John', 'Wick', 'johnwick@example.com', '123-456-7890', '123 Main St');

INSERT INTO Customers (customer_id, first_name, last_name, email, phone_number, address)
VALUES (2, 'Jane', 'Austin', 'janeaustin@example.com', '123-456-7891', '456 Oak St');

INSERT INTO Customers (customer_id, first_name, last_name, email, phone_number, address)
VALUES (3, 'Bob', 'Builder', 'bobbuilder@example.com', '123-456-7892', '789 Elm St');

INSERT INTO Customers (customer_id, first_name, last_name, email, phone_number, address)
VALUES (4, 'Samantha', 'Jones', 'sjones@example.com', '123-456-7893', '246 Maple St');
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

## ➔ Orders

SQL Worksheet

```sql
INSERT INTO Orders (order_id, customer_id, order_date, total_cost)
VALUES (1, 1, TO_DATE('2022-01-01', 'YYYY-MM-DD'), 25.00);

INSERT INTO Orders (order_id, customer_id, order_date, total_cost)
VALUES (2, 2, TO_DATE('2022-02-01', 'YYYY-MM-DD'), 35.00);

INSERT INTO Orders (order_id, customer_id, order_date, total_cost)
VALUES (3, 3, TO_DATE('2022-03-01', 'YYYY-MM-DD'), 20.00);

INSERT INTO Orders (order_id, customer_id, order_date, total_cost)
VALUES (4, 4, TO_DATE('2022-04-01', 'YYYY-MM-DD'), 15.00);
```

```
1 row(s) inserted.
```

## ➔ Menu Items

SQL Worksheet  Clear  Find  Actions

```sql
INSERT INTO Menu_Items (menu_item_id, name, description, price, category)
VALUES (1, 'Butter Chicken', 'Tender chicken cooked in a creamy tomato-based sauce with butter and spices', 12.99, 'Entree');

INSERT INTO Menu_Items (menu_item_id, name, description, price, category)
VALUES (2, 'Chana Masala', 'Chickpeas cooked in a spicy tomato-based sauce with onions and cilantro', 9.99, 'Vegetarian');

INSERT INTO Menu_Items (menu_item_id, name, description, price, category)
VALUES (3, 'Aloo Gobi', 'Potatoes and cauliflower cooked in a blend of spices', 10.99, 'Vegetarian');

INSERT INTO Menu_Items (menu_item_id, name, description, price, category)
VALUES (4, 'Palak Paneer', 'Cubes of paneer cheese in a creamy spinach sauce with spices', 11.99, 'Vegetarian');
```

```
1 row(s) inserted.
```

## ➔ Staff Info

SQL Worksheet

```sql
INSERT INTO Staff (staff_id, first_name, last_name, position, hourly_rate, hire_date)
VALUES (1, 'Samidha', 'Guru', 'Server', 12.50, TO_DATE('2022-01-15', 'yyyy-mm-dd'));

INSERT INTO Staff (staff_id, first_name, last_name, position, hourly_rate, hire_date)
VALUES (2, 'David', 'Smith', 'Chef', 20.00, TO_DATE('2022-02-01', 'yyyy-mm-dd'));

INSERT INTO Staff (staff_id, first_name, last_name, position, hourly_rate, hire_date)
VALUES (3, 'Jasmine', 'Punjabi', 'Server', 12.50, TO_DATE('2022-02-15', 'yyyy-mm-dd'));

INSERT INTO Staff (staff_id, first_name, last_name, position, hourly_rate, hire_date)
VALUES (4, 'Mohammed', 'Ali', 'Cook', 15.00, TO_DATE('2022-03-01', 'yyyy-mm-dd'));
```

```
1 row(s) inserted.
```

## Querying Data :

- ## Customers

SQL Worksheet

```
SELECT * FROM CUSTOMERS;
```

| CUSTOMER_ID | FIRST_NAME | LAST_NAME | EMAIL | PHONE_NUMBER | ADDRESS |
|---|---|---|---|---|---|
| 1 | John | Wick | johnwick@example.com | 123-456-7890 | 123 Main St |
| 2 | Jane | Austin | janeaustin@example.com | 123-456-7891 | 456 Oak St |
| 3 | Bob | Builder | bobbuilder@example.com | 123-456-7892 | 789 Elm St |
| 4 | Samantha | Jones | sjones@example.com | 123-456-7893 | 246 Maple St |
| 5 | James | Brown | jbrown@example.com | 123-456-7894 | 135 Oak Ln |
| 6 | Amy | Lee | alee@example.com | 123-456-7895 | 369 Pine Ave |

- ## Orders

SQL Worksheet

```
SELECT * FROM ORDERS;
```

| ORDER_ID | CUSTOMER_ID | ORDER_DATE | TOTAL_COST |
|---|---|---|---|
| 1 | 1 | 01-JAN-22 | 25 |
| 2 | 2 | 01-FEB-22 | 35 |
| 3 | 3 | 01-MAR-22 | 20 |
| 4 | 4 | 01-APR-22 | 15 |
| 5 | 5 | 01-MAY-22 | 45 |
| 6 | 6 | 01-JUN-22 | 22 |

## ■ Menu Items

SQL Worksheet

```
SELECT * FROM MENU_ITEMS;
```

| MENU_ITEM_ID | NAME | DESCRIPTION | PRICE | CATEGORY |
|---|---|---|---|---|
| 1 | Butter Chicken | Tender chicken cooked in a creamy tomato-based sauce with butter and spices | 12.99 | Entree |
| 2 | Chana Masala | Chickpeas cooked in a spicy tomato-based sauce with onions and cilantro | 9.99 | Vegetarian |
| 3 | Aloo Gobi | Potatoes and cauliflower cooked in a blend of spices | 10.99 | Vegetarian |
| 4 | Palak Paneer | Cubes of paneer cheese in a creamy spinach sauce with spices | 11.99 | Vegetarian |
| 5 | Tandoori Chicken | Chicken marinated in yogurt and spices, cooked in a clay oven | 13.99 | Entree |

## ■ Staff Info

SQL Worksheet

```
SELECT * FROM STAFF;
```

| STAFF_ID | FIRST_NAME | LAST_NAME | POSITION | HOURLY_RATE | HIRE_DATE |
|---|---|---|---|---|---|
| 1 | Samidha | Guru | Server | 12.5 | 15-JAN-22 |
| 2 | David | Smith | Chef | 20 | 01-FEB-22 |
| 3 | Jasmine | Punjabi | Server | 12.5 | 15-FEB-22 |
| 4 | Mohammed | Ali | Cook | 15 | 01-MAR-22 |
| 5 | Nina | Dobrev | Server | 12.5 | 15-MAR-22 |

## ★ Tables in database

SQL Worksheet

```
SELECT TABLE_NAME FROM USER_TABLES;
```

| TABLE_NAME |
|---|
| CUSTOMERS |
| MENU_ITEMS |
| ORDERS |
| STAFF |

## Altering Table :

- **Adding Column**

SQL Worksheet

```sql
ALTER TABLE CUSTOMERS
ADD BIRTH_DATE DATE;
```

```
Table altered.
```

- **Changing Datatype**

SQL Worksheet

```sql
ALTER TABLE CUSTOMERS
MODIFY ADDRESS VARCHAR2(100);
```

```
Table altered.
```

- **Renaming Column**

SQL Worksheet

```sql
ALTER TABLE CUSTOMERS
RENAME COLUMN PHONE_NUMBER TO MOBILE_NUMBER;
```

```
Table altered.
```

- **Deleting Column**

SQL Worksheet

```sql
ALTER TABLE CUSTOMERS
DROP COLUMN BIRTH_DATE;
```

```
Table altered.
```

- **Renaming Table**

SQL Worksheet

```sql
ALTER TABLE STAFF
RENAME TO STAFF_INFO;
```

```
Table altered.
```

- **Truncating table** (Deleting the data, but keeping the table structure)

SQL Worksheet

```sql
TRUNCATE TABLE example_table;
```

```
Table truncated.
```

- **Deleting Table**

SQL Worksheet

```sql
DROP TABLE example_table;
```

Table dropped.

---

## Using Order By Clause

### ⏫ Ascending

SQL Worksheet

```sql
SELECT * FROM CUSTOMERS
ORDER BY FIRST_NAME ASC;
```

| CUSTOMER_ID | FIRST_NAME | LAST_NAME | EMAIL | MOBILE_NUMBER | ADDRESS |
|---|---|---|---|---|---|
| 9 | Alex | Dadd | adadd@example.com | 123-456-7898 | 789 Oak St |
| 6 | Amy | Lee | alee@example.com | 123-456-7895 | 369 Pine Ave |
| 3 | Bob | Builder | bobbuilder@example.com | 123-456-7892 | 789 Elm St |
| 8 | Emily | Paris | eparis@example.com | 123-456-7897 | 457 Birch Rd |
| 5 | James | Brown | jbrown@example.com | 123-456-7894 | 135 Oak Ln |

### ⏬ Descending

SQL Worksheet

```sql
SELECT * FROM CUSTOMERS
ORDER BY FIRST_NAME DESC;
```

| CUSTOMER_ID | FIRST_NAME | LAST_NAME | EMAIL | MOBILE_NUMBER | ADDRESS |
|---|---|---|---|---|---|
| 4 | Samantha | Jones | sjones@example.com | 123-456-7893 | 246 Maple St |
| 7 | Michael | Davis | mdavis@example.com | 123-456-7896 | 802 Cedar Blvd |
| 10 | Lily | Chen | lchen@example.com | 123-456-7899 | 102 Elm St |
| 1 | John | Wick | johnwick@example.com | 123-456-7890 | 123 Main St |
| 2 | Jane | Austin | janeaustin@example.com | 123-456-7891 | 456 Oak St |

## ⏫ On multiple Columns

SQL Worksheet

```sql
SELECT * FROM CUSTOMERS
ORDER BY CUSTOMER_ID, ADDRESS;
```

| CUSTOMER_ID | FIRST_NAME | LAST_NAME | EMAIL | MOBILE_NUMBER | ADDRESS |
|---|---|---|---|---|---|
| 1 | John | Wick | johnwick@example.com | 123-456-7890 | 123 Main St |
| 2 | Jane | Austin | janeaustin@example.com | 123-456-7891 | 456 Oak St |
| 3 | Bob | Builder | bobbuilder@example.com | 123-456-7892 | 789 Elm St |
| 4 | Samantha | Jones | sjones@example.com | 123-456-7893 | 246 Maple St |
| 5 | James | Brown | jbrown@example.com | 123-456-7894 | 135 Oak Ln |

---

## Fetching limited rows

SQL Worksheet

```sql
SELECT * FROM STAFF_INFO
FETCH FIRST 5 ROWS ONLY;
```

| STAFF_ID | FIRST_NAME | LAST_NAME | POSITION | HOURLY_RATE | HIRE_DATE |
|---|---|---|---|---|---|
| 1 | Samidha | Guru | Server | 12.5 | 15-JAN-22 |
| 2 | David | Smith | Chef | 20 | 01-FEB-22 |
| 3 | Jasmine | Punjabi | Server | 12.5 | 15-FEB-22 |
| 4 | Mohammed | Ali | Cook | 15 | 01-MAR-22 |
| 5 | Nina | Dobrev | Server | 12.5 | 15-MAR-22 |

## Fetching Unique (Distinct) Values

SQL Worksheet

```
SELECT DISTINCT(POSITION)
FROM STAFF_INFO;
```

| POSITION |
| --- |
| Server |
| Chef |
| Cook |

---

## Filtering Results using WHERE Clause

- **Using Comparison Operator**

SQL Worksheet

```
SELECT * FROM STAFF_INFO
WHERE HOURLY_RATE = 12.5;
```

| STAFF_ID | FIRST_NAME | LAST_NAME | POSITION | HOURLY_RATE | HIRE_DATE |
| --- | --- | --- | --- | --- | --- |
| 1 | Samidha | Guru | Server | 12.5 | 15-JAN-22 |
| 3 | Jasmine | Punjabi | Server | 12.5 | 15-FEB-22 |
| 5 | Nina | Dobrev | Server | 12.5 | 15-MAR-22 |
| 7 | Lila | Dcruz | Server | 12.5 | 15-APR-22 |

Display STAFF with 12.5 Hourly Rate

SQL Worksheet

```sql
SELECT * FROM STAFF_INFO
WHERE POSITION = 'Server';
```

| STAFF_ID | FIRST_NAME | LAST_NAME | POSITION | HOURLY_RATE | HIRE_DATE |
|---|---|---|---|---|---|
| 1 | Samidha | Guru | Server | 12.5 | 15-JAN-22 |
| 3 | Jasmine | Punjabi | Server | 12.5 | 15-FEB-22 |
| 5 | Nina | Dobrev | Server | 12.5 | 15-MAR-22 |
| 7 | Lila | Dcruz | Server | 12.5 | 15-APR-22 |
| 9 | Sarah | Ali | Server | 12.5 | 15-MAY-22 |

Display Server Staff

SQL Worksheet

```sql
SELECT STAFF_ID, FIRST_NAME, LAST_NAME, HIRE_DATE
FROM STAFF_INFO
WHERE HIRE_DATE <= '01-MAR-22';
```

| STAFF_ID | FIRST_NAME | LAST_NAME | HIRE_DATE |
|---|---|---|---|
| 1 | Samidha | Guru | 15-JAN-22 |
| 2 | David | Smith | 01-FEB-22 |
| 3 | Jasmine | Punjabi | 15-FEB-22 |
| 4 | Mohammed | Ali | 01-MAR-22 |

Display Staff hired before 1st March 2022

- **On multiple columns**

SQL Worksheet

```sql
SELECT *
FROM CUSTOMERS
WHERE FIRST_NAME = 'James' AND LAST_NAME = 'Brown';
```

| CUSTOMER_ID | FIRST_NAME | LAST_NAME | EMAIL | MOBILE_NUMBER | ADDRESS |
|---|---|---|---|---|---|
| 5 | James | Brown | jbrown@example.com | 123-456-7894 | 135 Oak Ln |

SQL Worksheet

```sql
SELECT *
FROM STAFF_INFO
WHERE POSITION = 'Cook' OR HOURLY_RATE = 20;
```

| STAFF_ID | FIRST_NAME | LAST_NAME | POSITION | HOURLY_RATE | HIRE_DATE |
|---|---|---|---|---|---|
| 2 | David | Smith | Chef | 20 | 01-FEB-22 |
| 4 | Mohammed | Ali | Cook | 15 | 01-MAR-22 |
| 6 | James | Bond | Chef | 20 | 01-APR-22 |
| 8 | Amir | Khan | Cook | 15 | 01-MAY-22 |
| 10 | Kevin | Masifar | Chef | 20 | 01-JUN-22 |

Display Cooks or Staff with 20 Hourly Rate

## Alias (Temporary Name)

SQL Worksheet

```sql
SELECT FIRST_NAME AS NAME, MOBILE_NUMBER AS PHONE
FROM CUSTOMERS;
```

| NAME | PHONE |
|---|---|
| John | 123-456-7890 |
| Jane | 123-456-7891 |
| Bob | 123-456-7892 |
| Samantha | 123-456-7893 |
| James | 123-456-7894 |

---

## Between Operator

SQL Worksheet

```sql
SELECT *
FROM ORDERS
WHERE TOTAL_COST BETWEEN 25 AND 45;
```

| ORDER_ID | CUSTOMER_ID | ORDER_DATE | TOTAL_COST |
|---|---|---|---|
| 1 | 1 | 01-JAN-22 | 25 |
| 2 | 2 | 01-FEB-22 | 35 |
| 5 | 5 | 01-MAY-22 | 45 |
| 9 | 9 | 01-SEP-22 | 28 |

Orders with total cost between 25 and 45

## In Operator

SQL Worksheet

```
SELECT *
FROM MENU_ITEMS
WHERE MENU_ITEM_ID IN (1,2,3);
```

| MENU_ITEM_ID | NAME | DESCRIPTION | PRICE | CATEGORY |
|---|---|---|---|---|
| 1 | Butter Chicken | Tender chicken cooked in a creamy tomato-based sauce with butter and spices | 12.99 | Entree |
| 2 | Chana Masala | Chickpeas cooked in a spicy tomato-based sauce with onions and cilantro | 9.99 | Vegetarian |
| 3 | Aloo Gobi | Potatoes and cauliflower cooked in a blend of spices | 10.99 | Vegetarian |

Show Menu items with id 1, 2 or 3

---

## DML Examples

- **INSERT**

SQL Worksheet

```
INSERT INTO MENU_ITEMS (MENU_ITEM_ID,NAME,DESCRIPTION,PRICE,CATEGORY)
VALUES (11,'Prawns Biryani','Aromatic basmati rice dum cooked with prawns, herbs & spice powders',15.99,'Seafood');
```

```
1 row(s) inserted.
```

- **UPDATE**

SQL Worksheet

```
UPDATE STAFF_INFO
SET FIRST_NAME = 'Chimmu', LAST_NAME = 'Singh'
WHERE STAFF_ID = 1;
```

```
1 row(s) updated.
```

- **DELETE**

SQL Worksheet

```sql
DELETE FROM MENU_ITEMS
WHERE MENU_ITEM_ID = 11;
```

1 row(s) deleted.

---

## Aggregate Functions

- **COUNT**

SQL Worksheet

```sql
SELECT COUNT(FIRST_NAME) AS TOTAL_COUNT
FROM CUSTOMERS;
```

| TOTAL_COUNT |
| --- |
| 10 |

Count of total customers

- **MAX**

SQL Worksheet

```sql
SELECT MAX(HOURLY_RATE) AS HIGHEST_RATE
FROM STAFF_INFO;
```

| HIGHEST_RATE |
| --- |
| 20 |

Highest Rate of Staff

- **MIN**

SQL Worksheet

```sql
SELECT MIN(PRICE) AS LOWEST_PRICE
FROM MENU_ITEMS;
```

| LOWEST_PRICE |
| --- |
| 2.99 |

Cheapest Menu item

- **AVG**

SQL Worksheet

```sql
SELECT AVG(HOURLY_RATE) AS AVERAGE
FROM STAFF_INFO;
```

| AVERAGE |
| --- |
| 15.25 |

- **SUM**

SQL Worksheet

```sql
SELECT SUM(PRICE) AS TOTAL_PRICE
FROM MENU_ITEMS;
```

| TOTAL_PRICE |
| --- |
| 90.9 |

Total price of all items

---

## GROUP BY

SQL Worksheet

```sql
SELECT COUNT(*) AS COUNT, POSITION
FROM STAFF_INFO
GROUP BY POSITION;
```

| COUNT | POSITION |
| --- | --- |
| 5 | Server |
| 3 | Chef |
| 2 | Cook |

Count of staff per position

## GROUP BY with HAVING

SQL Worksheet

```sql
SELECT MAX(TOTAL_COST) AS MAX_COST_DAY, ORDER_DATE
FROM ORDERS
GROUP BY ORDER_DATE
HAVING MAX(TOTAL_COST) > 20;
```

| MAX_COST_DAY | ORDER_DATE |
|---|---|
| 22 | 01-JUN-22 |
| 33 | 01-OCT-22 |
| 25 | 01-JAN-22 |
| 45 | 01-MAY-22 |
| 35 | 01-FEB-22 |

Most expensive order per day, considering cost of order should be more than 20

---

## LIKE Operator

SQL Worksheet

```sql
SELECT *
FROM CUSTOMERS
WHERE FIRST_NAME LIKE 'J%';
```

| CUSTOMER_ID | FIRST_NAME | LAST_NAME | EMAIL | MOBILE_NUMBER | ADDRESS |
|---|---|---|---|---|---|
| 1 | John | Wick | johnwick@example.com | 123-456-7890 | 123 Main St |
| 2 | Jane | Austin | janeaustin@example.com | 123-456-7891 | 456 Oak St |
| 5 | James | Brown | jbrown@example.com | 123-456-7894 | 135 Oak Ln |

Customers whose name start with **"J"**

## UNION

SQL Worksheet

```sql
SELECT FIRST_NAME AS ALL_NAMES FROM CUSTOMERS
UNION
SELECT FIRST_NAME AS ALL_NAMES FROM STAFF_INFO;
```

| ALL_NAMES |
|---|
| Alex |
| Amir |
| Amy |
| Bob |
| David |

Display ALL names in Customers and Staff

---

# JOINS

- ## INNER JOIN

SQL Worksheet

```sql
SELECT FIRST_NAME ||' '|| LAST_NAME AS FULL_NAME, ORDER_DATE, TOTAL_COST
FROM CUSTOMERS C
INNER JOIN ORDERS O
ON C.CUSTOMER_ID = O.ORDER_ID;
```

| FULL_NAME | ORDER_DATE | TOTAL_COST |
|---|---|---|
| John Wick | 01-JAN-22 | 25 |
| Jane Austin | 01-FEB-22 | 35 |
| Bob Builder | 01-MAR-22 | 20 |
| Samantha Jones | 01-APR-22 | 15 |

Show Customers, their order date and order cost

- **LEFT JOIN**

SQL Worksheet

```
SELECT C.FIRST_NAME, C.LAST_NAME, O.TOTAL_COST
FROM CUSTOMERS C LEFT JOIN ORDERS O ON O.CUSTOMER_ID = C.CUSTOMER_ID;
```

| FIRST_NAME | LAST_NAME | TOTAL_COST |
|---|---|---|
| John | Wick | 25 |
| Jane | Austin | 35 |
| Bob | Builder | 20 |
| Samantha | Jones | 15 |

Show All Customers and their order cost

---

# SUBQUERY

SQL Worksheet

Clear Find Actions Save Run

```
SELECT * FROM MENU_ITEMS WHERE PRICE > (SELECT AVG(PRICE) FROM MENU_ITEMS);
```

| MENU_ITEM_ID | NAME | DESCRIPTION | PRICE | CATEGORY |
|---|---|---|---|---|
| 1 | Butter Chicken | Tender chicken cooked in a creamy tomato-based sauce with butter and spices | 12.99 | Entree |
| 2 | Chana Masala | Chickpeas cooked in a spicy tomato-based sauce with onions and cilantro | 9.99 | Vegetarian |
| 3 | Aloo Gobi | Potatoes and cauliflower cooked in a blend of spices | 10.99 | Vegetarian |
| 4 | Palak Paneer | Cubes of paneer cheese in a creamy spinach sauce with spices | 11.99 | Vegetarian |
| 5 | Tandoori Chicken | Chicken marinated in yogurt and spices, cooked in a clay oven | 13.99 | Entree |
| 6 | Chicken Tikka Masala | Tender pieces of chicken in a spicy tomato-based sauce with cream and spices | 12.99 | Entree |

Menu Items that are costlier than the average

SQL Worksheet

Clear Find Actions Save

```
SELECT * FROM MENU_ITEMS WHERE PRICE = (SELECT MAX(PRICE) FROM MENU_ITEMS WHERE PRICE < (SELECT MAX(PRICE) FROM MENU_ITEMS));
```

| MENU_ITEM_ID | NAME | DESCRIPTION | PRICE | CATEGORY |
|---|---|---|---|---|
| 1 | Butter Chicken | Tender chicken cooked in a creamy tomato-based sauce with butter and spices | 12.99 | Entree |
| 6 | Chicken Tikka Masala | Tender pieces of chicken in a spicy tomato-based sauce with cream and spices | 12.99 | Entree |

2nd most expensive item(s) on the menu

---

# VIEWS

## ➢ Creating a VIEW

SQL Worksheet

```sql
CREATE VIEW STAFF_INFO_VIEW
AS SELECT STAFF_ID, FIRST_NAME, LAST_NAME, POSITION
FROM STAFF_INFO;
```

View created.

## ➢ Displaying the VIEW

SQL Worksheet

```sql
SELECT * FROM STAFF_INFO_VIEW;
```

| STAFF_ID | FIRST_NAME | LAST_NAME | POSITION |
|---|---|---|---|
| 1 | Samidha | Guru | Server |
| 2 | David | Smith | Chef |
| 3 | Jasmine | Punjabi | Server |
| 4 | Mohammed | Ali | Cook |
| 5 | Nina | Dobrev | Server |