CASE STUDY – 01 RESTAURANT ORDER MANAGEMENT

Introduction

In the competitive restaurant industry, effective order management is vital for enhancing customer satisfaction and optimizing operational efficiency. By leveraging database analytics, a restaurant can gain valuable insights into customer preferences, order patterns, and menu performance. This enables the establishment to make informed decisions that improve service delivery, streamline operations, and increase revenue.

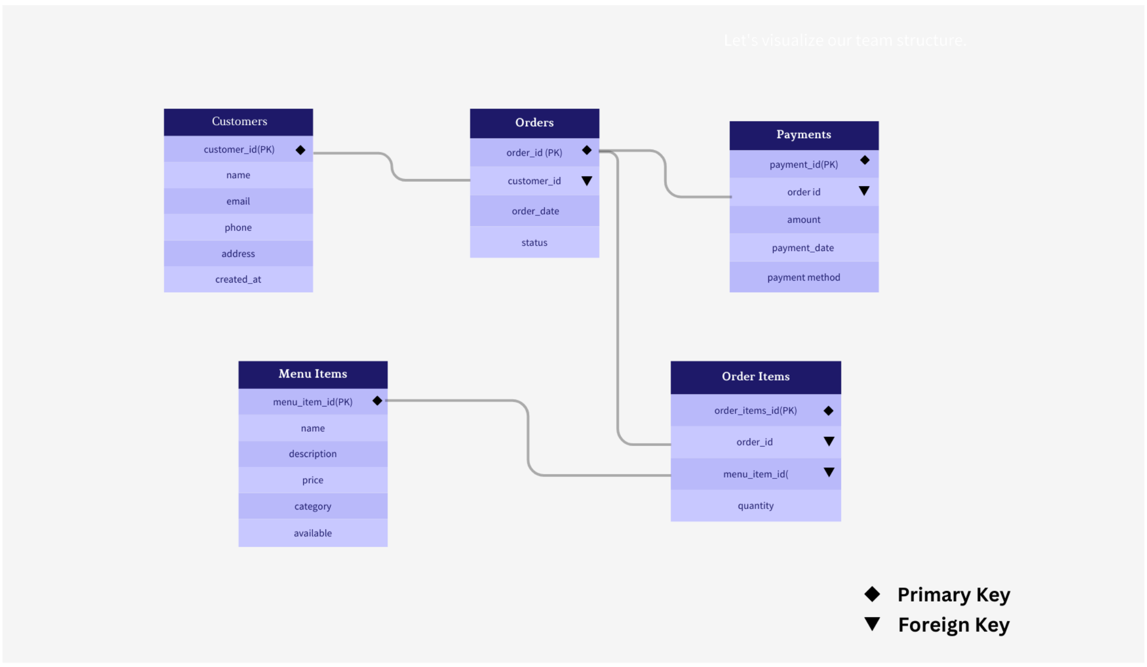
Problem Statement

This problem statement focuses on using data to gain insights into customer preferences, spending patterns, and menu performance, with the ultimate goal of enhancing the restaurant's operations and profitability.

Key Datasets

The case study utilizes the following key datasets:

- Customers
- Menu_Items
- Order_Items
- Orders
- Payments



DATASETS:

```
Create kj_resto;

Use kj_resto;

select * from Customers;

select * from Orders;

select * from Order_Items;

select * from Menu_Items;

select * from Payments;
```

— Create table Customers —

Create Table & Insert Data:

```
CREATE TABLE Customers (
   customer_id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(100) NOT NULL,
   email VARCHAR(100) UNIQUE NOT NULL,
   phone VARCHAR(15),
   address VARCHAR(255),
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
INSERT INTO Customers (name, email, phone, address, created_at) VALUES
('Kamal', 'kamal.raj@example.com', '123-456-7890', '123 Main St, Anytown, IND',
'2023-01-01 10:00:00'),
('Hiran', 'hiran.lal@example.com', '098-765-4321', '456 Elm St, Othertown, IND', '2023-02-01
11:00:00'),
('Gautham', 'gautham.venkat@example.com', '555-555-5555', '789 Oak St, Sometown, IND',
'2023-03-01 12:00:00'),
('Karthi', 'karthi.charan@example.com', '444-444-4444', '101 Pine St, Yourtown, IND',
'2023-04-01 13:00:00'),
('Sargunal', 'sargunal.bala@example.com', '333-333-3333', '202 Birch St, Mytown, IND',
'2023-05-01 14:00:00'),
('Aarav', 'aarav.sharma@example.com', '666-666-6666', '123 Cedar St, Anytown, IND',
'2023-06-01 15:00:00'),
('Divya', 'divya.rao@example.com', '777-777-7777', '456 Maple St, Othertown, IND',
'2023-07-01 16:00:00'),
('Ravi', 'ravi.kumar@example.com', '888-888-8888', '789 Ash St, Sometown, IND',
'2023-08-01 17:00:00'),
('Pooja', 'pooja.verma@example.com', '999-999-9999', '101 Birch St, Yourtown, IND',
'2023-09-01 18:00:00'),
('Vikas', 'vikas.malhotra@example.com', '111-222-3333', '202 Cedar St, Mytown, IND',
'2023-10-01 19:00:00');
```

— Create table Menu_Items —

Create Table & Insert Data:

```
CREATE TABLE Menu_Items (
    menu_item_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    price DECIMAL(10, 2),
    category VARCHAR(50),
    available BOOLEAN
);
INSERT INTO Menu_Items (name, description, price, category, available) VALUES
('Pizza', 'Delicious cheese pizza', 9.99, 'Food', TRUE),
('Burger', 'Juicy beef burger', 7.99, 'Food', TRUE),
('Pasta', 'Creamy Alfredo pasta', 8.99, 'Food', TRUE),
('Salad', 'Fresh garden salad', 5.99, 'Food', TRUE),
('Soda', 'Refreshing cola drink', 1.99, 'Beverage', TRUE),
('Chicken Wings', 'Spicy chicken wings', 12.99, 'Food', TRUE),
('Ice Cream', 'Vanilla ice cream', 3.99, 'Dessert', TRUE),
('Steak', 'Grilled chicken steak', 19.99, 'Food', TRUE),
('Coffee', 'Hot brewed coffee', 2.99, 'Beverage', TRUE),
('Chocolate Cake', 'Rich chocolate cake', 6.99, 'Dessert', TRUE);
```

— Create table Order_Items —

Create Table & Insert Data:

```
CREATE TABLE Order_Items (
    order_item_id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT,
    menu_item_id INT,
    quantity INT,
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),
    FOREIGN KEY (menu_item_id) REFERENCES Menu_Items(menu_item_id)
);
INSERT INTO Order_Items (order_id, menu_item_id, quantity) VALUES
(1, 1, 2),     -- Order 1 contains 2 Pizza
(2, 2, 1),     -- Order 2 contains 1 Burger
(3, 3, 1),     -- Order 3 contains 1 Pasta
(4, 4, 3),     -- Order 4 contains 3 Salad
(5, 5, 3),     -- Order 5 contains 3 Soda
(6, 6, 4),     -- Order 6 contains 4 Chicken Wings
(7, 7, 2),     -- Order 7 contains 2 Ice Cream
(8, 8, 1),     -- Order 8 contains 1 Steak
(9, 9, 3),     -- Order 9 contains 3 Coffee
(10, 10, 2); -- Order 10 contains 2 Chocolate Cake
```

— Create table Orders —

Create Table & Insert Data:

```
CREATE TABLE Orders (
   order_id INT AUTO_INCREMENT PRIMARY KEY,
   customer_id INT,
   order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   status VARCHAR(50),
   FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);
INSERT INTO Orders (customer_id, order_date, status) VALUES
(1, '2023-01-02 10:00:00', 'Completed'),
(2, '2023-02-02 11:00:00', 'Pending'),
(3, '2023-03-02 12:00:00', 'Shipped'),
(4, '2023-04-02 13:00:00', 'Cancelled'),
(5, '2023-05-02 14:00:00', 'Processing'),
(6, '2023-06-02 15:00:00', 'Completed'),
(7, '2023-07-02 16:00:00', 'Pending'),
(8, '2023-08-02 17:00:00', 'Shipped'),
(9, '2023-09-02 18:00:00', 'Cancelled'),
(10, '2023-10-02 19:00:00', 'Processing');
```

— Create table Payments —

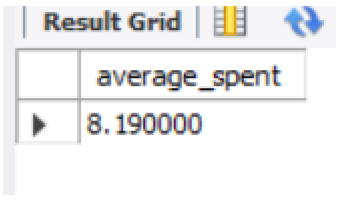
Create Table & Insert Data:

```
CREATE TABLE Payments (
   payment_id INT AUTO_INCREMENT PRIMARY KEY,
   order_id INT,
   amount DECIMAL(10, 2),
   payment_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   payment_method VARCHAR(50),
   FOREIGN KEY (order_id) REFERENCES Orders(order_id)
);
INSERT INTO Payments (order_id, amount, payment_date, payment_method) VALUES
(1, 9.99, '2023-01-03 10:00:00', 'Credit Card'),
(2, 7.99, '2023-02-03 11:00:00', 'UPI'),
(3, 8.99, '2023-03-03 12:00:00', 'Cash'),
(4, 5.99, '2023-04-03 13:00:00', 'Debit Card'),
(5, 1.99, '2023-05-03 14:00:00', 'Credit Card'),
(6, 12.99, '2023-06-03 15:00:00', 'Credit Card'),
(7, 3.99, '2023-07-03 16:00:00', 'UPI'),
(8, 19.99, '2023-08-03 17:00:00', 'Cash'),
(9, 2.99, '2023-09-03 18:00:00', 'Debit Card'),
(10, 6.99, '2023-10-03 19:00:00', 'Credit Card');
```

Case Study Questions & Answers:

1. To find the average amount spent:

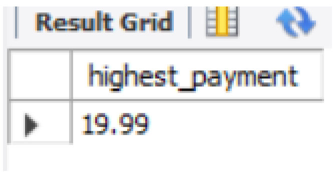
SELECT AVG(amount) AS average_spent
FROM Payments;



	average_spent
▶	8.190000

2.To find the Menu_Item with Highest Price:

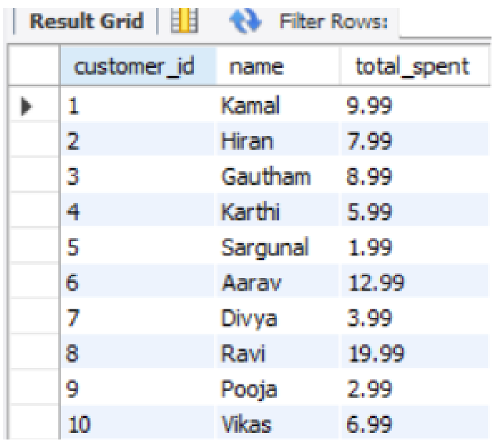
Find the menu item with the highest price:
SELECT name, price
FROM Menu_Items
ORDER BY price DESC
LIMIT 1;



	highest_payment
▶	19.99

3. To find the total amount spent by each customer:

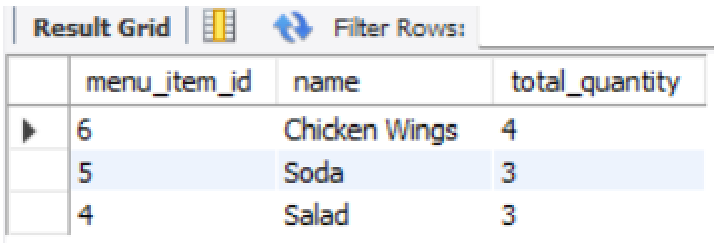
SELECT c.customer_id, c.name, SUM(p.amount) AS total_spent
FROM Customers c
JOIN Orders o ON c.customer_id = o.customer_id
JOIN Payments p ON o.order_id = p.order_id
GROUP BY c.customer_id, c.name;



	customer_id	name	total_spent
▶	1	Kamal	9.99
	2	Hiran	7.99
	3	Gautham	8.99
	4	Karthi	5.99
	5	Sargunal	1.99
	6	Aarav	12.99
	7	Divya	3.99
	8	Ravi	19.99
	9	Pooja	2.99
	10	Vikas	6.99

4. To list the top 3 most ordered menu items:

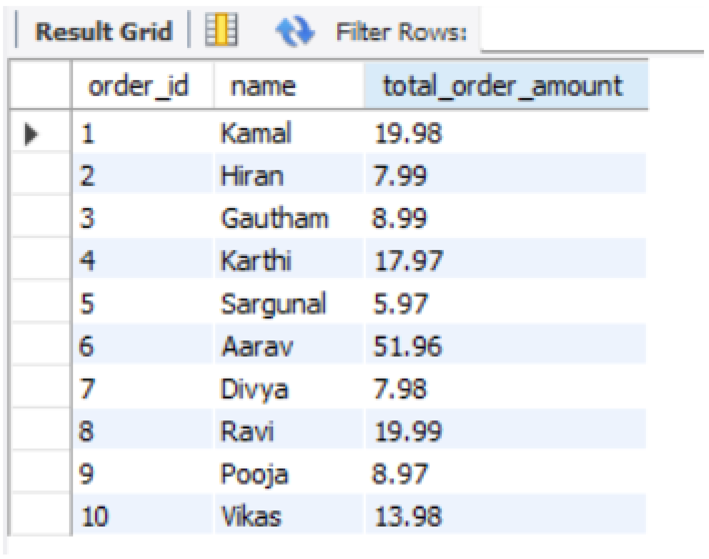
```
SELECT mi.menu_item_id, mi.name, SUM(oi.quantity) AS total_quantity
FROM Menu_Items mi
JOIN Order_Items oi ON mi.menu_item_id = oi.menu_item_id
GROUP BY mi.menu_item_id, mi.name
ORDER BY total_quantity DESC
LIMIT 3;
```



menu_item_id	name	total_quantity
6	Chicken Wings	4
5	Soda	3
4	Salad	3

5. To list all orders along with the customer's name and the total amount of the order:

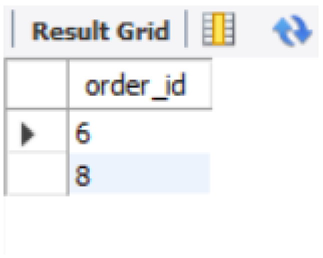
```
SELECT o.order_id, c.name, SUM(oi.quantity * mi.price) AS total_order_amount
FROM Orders o
JOIN Customers c ON o.customer_id = c.customer_id
JOIN Order_Items oi ON o.order_id = oi.order_id
JOIN Menu_Items mi ON oi.menu_item_id = mi.menu_item_id
GROUP BY o.order_id, c.name;
```



order_id	name	total_order_amount
1	Kamal	19.98
2	Hiran	7.99
3	Gautham	8.99
4	Karthi	17.97
5	Sargunal	5.97
6	Aarav	51.96
7	Divya	7.98
8	Ravi	19.99
9	Pooja	8.97
10	Vikas	13.98

6. To list all orders that include a menu item with a price greater than $10:

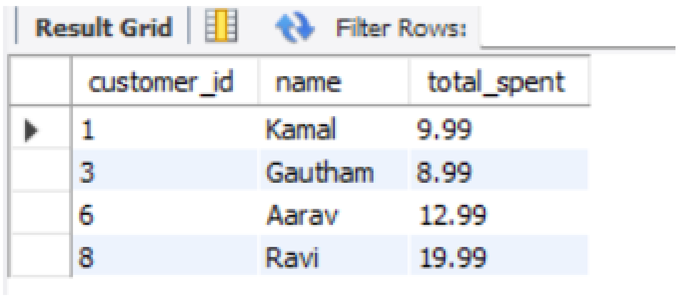
```
SELECT DISTINCT o.order_id
FROM Orders o
JOIN Order_Items oi ON o.order_id = oi.order_id
JOIN Menu_Items mi ON oi.menu_item_id = mi.menu_item_id
WHERE mi.price > 10.00;
```



order_id
6
8

7. To find customers who have spent more than the average amount spent per order:

```
SELECT c.customer_id, c.name, SUM(p.amount) AS total_spent
FROM Customers c
JOIN Orders o ON c.customer_id = o.customer_id
JOIN Payments p ON o.order_id = p.order_id
GROUP BY c.customer_id, c.name
HAVING SUM(p.amount) > (SELECT AVG(amount) FROM Payments);
```



customer_id	name	total_spent
1	Kamal	9.99
3	Gautham	8.99
6	Aarav	12.99
8	Ravi	19.99