Restaurant Order Analysis – SQL Project

Duration: 3 Months
Tool Used: Microsoft SQL Server

Objective:

The purpose of this analysis is to understand restaurant sales, customer spending, and peak order times to help improve business performance in the upcoming quarter.

Table 1: MENU_ITEMS
Contains information about the food items available in the restaurant.

```
SELECT * FROM MENU_ITEMS
```

	menu_item_id	item_name	category	price
1	101	Hamburger	American	12.9499998092651
2	102	Cheeseburger	American	13.9499998092651
3	103	Hot Dog	American	9
4	104	Veggie Burger	American	10.5
5	105	Mac & Cheese	American	7
6	106	French Fries	American	7
7	107	Orange Chicken	Asian	16.5
8	108	Tofu Pad Thai	Asian	14.5
9	109	Korean Beef Bowl	Asian	17.9500007629395
10	110	Pork Ramen	Asian	17.9500007629395
11	111	California Roll	Asian	11.9499998092651
12	112	Salmon Roll	Asian	14.9499998092651
13	113	Edamame	Asian	5
14	114	Potstickers	Asian	9
15	115	Chicken Tacos	Mexican	11.9499998092651
16	116	Steak Tacos	Mexican	12.9499998092651

Table 2: ORDER_DETAILS
Contains information related to customer orders.

```
SELECT * FROM ORDER_DETAILS
```

	order_details_id	order_id	order_date	order_time	item_id
1	1	1	2023-01-01	11:38:36.0000000	109
2	2	2	2023-01-01	11:57:40.0000000	108
3	3	2	2023-01-01	11:57:40.0000000	124
4	4	2	2023-01-01	11:57:40.0000000	117
5	5	2	2023-01-01	11:57:40.0000000	129
6	6	2	2023-01-01	11:57:40.0000000	106
7	7	3	2023-01-01	12:12:28.0000000	117
8	8	3	2023-01-01	12:12:28.0000000	119
9	9	4	2023-01-01	12:16:31.0000000	117
10	10	5	2023-01-01	12:21:30.0000000	117
11	11	6	2023-01-01	12:29:36.0000000	101
12	12	6	2023-01-01	12:29:36.0000000	114
13	13	7	2023-01-01	12:50:37.0000000	123
14	14	8	2023-01-01	12:51:37.0000000	123
15	15	9	2023-01-01	12:52:01.0000000	108
16	16	9	2023-01-01	12:52:01.0000000	126

Restaurant Order Analysis – SQL Project

Data Cleaning Process

-- Checking Null Values

```
select
sum(case when order_id is null then 1 else 0 end) as order_id_nulls,
sum(case when order_date is null then 1 else 0 end) as order_date_nulls,
sum(case when order_time is null then 1 else 0 end) as order_time_nulls,
sum(case when item_id is null then 1 else 0 end) as item_id_null
from order_details
```

Result

	order_id_nulls	order_date_nulls	order_time_nulls	item_id_null
1	0	0	0	137

Item_id contains 137 NULL values

All other columns do not contain NULL values

Key Performance Indicators (KPI'S)

-- Total Orders

```
select count(distinct(order_id)) as Total_Order from order_details
```

	Total_Order
1	5370

-- Total Revenue

```
select round(sum(m.price),2) as total_amount from menu_items m
inner join order_details o on m.menu_item_id = o.item_id
```

	total_amount
1	159217.9

-- Average Order Value

```
select round(avg(Total_spend),2) as Avg_Order_Value from (select sum(m.price) as
Total_spend from menu_items m
inner join order_details o on m.menu_item_id = o.item_id
group by order_id) t
```

	Avg_Order_Value
1	29.8

Restaurant Order Analysis – SQL Project

-- Highest Order Value

```
select round(max(Total),2) as Highest_Order from (select sum(m.price) as Total
from menu_items m
inner join order_details o on m.menu_item_id = o.item_id
group by order_id)t
```

	Highest_Order
1	192.15

-- Lowest Order Value

```
select round(min(Total),2) as Lowest_Order from (select sum(m.price) as Total from
menu_items m
inner join order_details o on m.menu_item_id = o.item_id
group by order_id)t
```

	Lowest_Order
1	5

-- Granular Requirements

Menu Item Performance Analysis

-- Item Wise Quantity Sold and Revenue

```
select m.item_name, count(o.item_id) as Quantity_Sold,
round(sum(m.price),2) as Total_Revenue from order_details o
inner join menu_items m on m.menu_item_id = o.item_id
group by o.item_id,m.item_name
order by Total_Revenue desc
```

Result

	item_name	Quantity_Sold	Total_Revenue
1	Korean Beef Bowl	588	10554.6
2	Spaghetti & Meatballs	470	8436.5
3	Tofu Pad Thai	562	8149
4	Cheeseburger	583	8132.85
5	Hamburger	622	8054.9
6	Orange Chicken	456	7524
7	Eggplant Parmesan	420	7119
8	Steak Torta	489	6821.55
9	Chicken Parmesan	364	6533.8
10	Pork Ramen	360	6462
11	Chicken Burrito	455	5892.25
12	Mushroom Ravioli	359	5564.5
13	Spaghetti	367	5321.5
14	Steak Burrito	354	5292.3
15	Meat Lasagna	273	4900.35
16	Salmon Roll	324	4843.8

Restaurant Order Analysis – SQL Project

-- Top 5 High Performing Items

```
select top 5 m.item_name, count(o.item_id) as Quantity_Sold,
round(sum(m.price),2) as Total_Revenue from order_details o
inner join menu_items m on m.menu_item_id = o.item_id
WHERE o.item_id IS NOT NULL
group by o.item_id,m.item_name
order by Total_Revenue desc
```

Result

	item_name	Quantity_Sold	Total_Revenue
1	Korean Beef Bowl	588	10554.6
2	Spaghetti & Meatballs	470	8436.5
3	Tofu Pad Thai	562	8149
4	Cheeseburger	583	8132.85
5	Hamburger	622	8054.9

-- Low Performing Items

```
select top 2 m.item_name, count(o.item_id) as Quantity_Sold,
round(sum(m.price),2) as Total_Revenue from order_details o
inner join menu_items m on m.menu_item_id = o.item_id
group by o.item_id,m.item_name
order by Total_Revenue asc
```

Result

	item_name	Quantity_Sold	Total_Revenue
1	Chicken Tacos	123	1469.85
2	Potstickers	205	1845

-- Category Performance Analysis

-- Total Orders, Revenue and Average_order_Price per Category

```
select m.category, count(o.item_id) as Total_Order,
round(sum(m.price),2) as Revenue,
round(avg(price),2) as Average_Order_Price from menu_items m
inner join order_details o on m.menu_item_id = o.item_id
group by m.category
order by Revenue desc
```

Result

Restaurant Order Analysis – SQL Project

	category	Total_Order	Revenue	Average_Order_Price
1	Italian	2948	49462.7	16.78
2	Asian	3470	46720.65	13.46
3	Mexican	2945	34796.8	11.82
4	American	2734	28237.75	10.33

-- Time-Based Order Analysis

-- top 5 peak hour

```
select top 5 datepart(hour,order_time) as Peak_Hour,
count(distinct(order_id)) as Total_Order from order_details
group by datepart(hour,order_time)
order by Total_Order desc
```

Result

	Peak_Hour	Total_Order
1	12	647
2	17	623
3	18	596
4	13	595
5	19	498

-- lowest peak hour

```
select top 2 datepart(hour,order_time) as Lowest_Order_Hour,
count(distinct(order_id)) as Total_Order from order_details
group by datepart(hour,order_time)
order by Total_Order asc
```

Result

	Lowest_Order_Hour	Total_Order
1	10	2
2	23	4

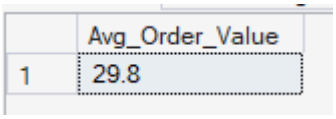
-- Customer Spending Analysis

-- Average Spend per Order

```
select round(avg(Total_spend),2) as Avg_Order_Value from (select sum(m.price) as
Total_spend from menu_items m
inner join order_details o on m.menu_item_id = o.item_id
group by order_id) t
```

Result

Restaurant Order Analysis – SQL Project

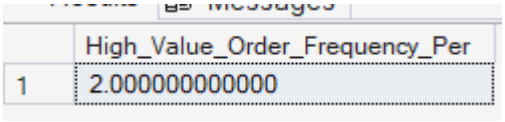


	Avg_Order_Value
1	29.8

-- High_Value_Order_Frequence

```
select round(count(case when total_Amount >= 100 then 1 end) * 100.0 / count(*),2)
as High_Value_Order_Frequency_Per from
(select sum(m.price) as total_Amount from menu_items m
inner join order_details o on m.menu_item_id = o.item_id
group by o.order_id)t
```

Result



	High_Value_Order_Frequency_Per
1	2.000000000000

Key Insights

- A small number of top-selling items generate the majority of revenue.
- Average Order Value is around **$30**, which can be increased through combo offers.
- Peak hours require proper staffing to maintain service quality.
- Low-performing items should be reviewed for pricing or removal.
- High-value orders (>$100) are limited, indicating scope for loyalty programs.