SQL Data Analysis Report — Roll Orders Dataset

Tools Used: SQL Server, SQL Queries

Dataset Components:

- Customer Orders (`customer_order`)

- Rolls (`rolls`)

- Roll Recipes (`rolls_recipes`)

- Ingredients (`ingredients`)

- Drivers (`drivers`)

1. Objective

To extract meaningful insights from roll-based order data using SQL queries in SQL Server. The analysis includes:

- Ingredient usage
- Roll popularity
- Customer behavior
- Delivery patterns
- Recipe compositions

2. Summary Insights

Metric	**Insight**
Most Ordered Roll	Non-Veg Roll (based on customer_order count)
Most Used Ingredient	BBQ Chicken or Cheese (based on frequency)
Rolls with Max Ingredients	Non-Veg Roll (8 ingredients)
Ingredient Never Used	[e.g., Tomato Sauce] (not part of any roll)
Most Popular Ingredient Combo	Common full recipe of Non-Veg Roll
Ingredient to Remove	Ingredient with lowest usage in orders
Roll Affected Most by Cheese Removal	Roll where Cheese is ordered most
Customers with Highest Orders	Identified by grouping by customer_id
Driver Order Stats	Orders placed after their registration date

4. Key Analytical Questions & Answers

Q1. List all rolls with their ingredients (names)

```
SELECT
  r.Rolls_Name,
  STRING_AGG(i.Ingredients_name, ', ') AS Ingredients_List
FROM Rolls r
JOIN Rolls_Recipes rr ON r.Rolls_id = rr.Roll_id
CROSS APPLY STRING_SPLIT(rr.Ingredients, ',') AS s
JOIN Ingredients i ON i.Ingredients_id = CAST(s.value AS INT)
GROUP BY r.Rolls_Name;
```

Q2. Get total ingredients used in each roll

```
SELECT
  r.Rolls_Name,
  COUNT(*) AS Ingredient_Count
FROM Rolls r
JOIN Rolls_Recipes rr ON r.Rolls_id = rr.Roll_id
CROSS APPLY STRING_SPLIT(rr.Ingredients, ',') AS s
GROUP BY r.Rolls_Name;
```

Q3. Which ingredients are used in both rolls?

```
SELECT i.Ingredients_name
FROM Ingredients i
WHERE i.Ingredients_id IN (
  SELECT value FROM Rolls_Recipes
  WHERE Roll_id = 1
  CROSS APPLY STRING_SPLIT(Ingredients, ',')
)
AND i.Ingredients_id IN (
  SELECT value FROM Rolls_Recipes
  WHERE Roll_id = 2
  CROSS APPLY STRING_SPLIT(Ingredients, ',')
);
```

Q4. Which ingredients are used only in Veg Roll?

```
SELECT i.Ingredients_name
FROM Ingredients i
WHERE i.Ingredients_id IN (
  SELECT value FROM Rolls_Recipes
  WHERE Roll_id = 2
  CROSS APPLY STRING_SPLIT(Ingredients, ',')
)
AND i.Ingredients_id NOT IN (
  SELECT value FROM Rolls_Recipes
  WHERE Roll_id = 1
  CROSS APPLY STRING_SPLIT(Ingredients, ',')
);
```

Q5. Which ingredients have never been used in any roll?

```
SELECT Ingredients_name
FROM Ingredients
WHERE Ingredients_id NOT IN (
  SELECT DISTINCT CAST(value AS INT)
  FROM Rolls_Recipes
  CROSS APPLY STRING_SPLIT(Ingredients, ',')
);
```

Q6. Which roll includes more than 5 ingredients?

```
SELECT r.Rolls_Name, COUNT(*) AS Ingredient_Count
FROM Rolls r
JOIN Rolls_Recipes rr ON r.Rolls_id = rr.Roll_id
CROSS APPLY STRING_SPLIT(rr.Ingredients, ',') AS s
GROUP BY r.Rolls_Name
HAVING COUNT(*) > 5;
```

Q7. Most popular ingredient (used in most customer orders)

```
SELECT TOP 1 i.Ingredients_name, COUNT(*) AS freq
FROM customer_order c
JOIN Rolls_Recipes rr ON c.roll_id = rr.Roll_id
CROSS APPLY STRING_SPLIT(rr.Ingredients, ',') AS s
JOIN Ingredients i ON i.Ingredients_id = CAST(s.value AS INT)
GROUP BY i.Ingredients_name
ORDER BY freq DESC;
```

Q8. Most popular ingredient combo

```
SELECT TOP 1 rr.Ingredients, COUNT(*) AS freq
FROM customer_order c
JOIN Rolls_Recipes rr ON c.roll_id = rr.Roll_id
GROUP BY rr.Ingredients
ORDER BY freq DESC;
```

Q9. If Cheese (ID = 4) is removed, which roll is most affected?

```
SELECT TOP 1 r.Rolls_Name, COUNT(*) AS Cheese_Orders
FROM customer_order c
JOIN Rolls_Recipes rr ON rr.Roll_id = c.roll_id
CROSS APPLY STRING_SPLIT(rr.Ingredients, ',') AS s
JOIN Rolls r ON r.Rolls_id = rr.Roll_id
WHERE CAST(s.value AS INT) = 4
GROUP BY r.Rolls_Name
ORDER BY Cheese_Orders DESC;
```

Q10. Orders placed after driver registration

```
SELECT d.driver_id, COUNT(*) AS Orders_Post_Registration
FROM Drivers d
JOIN customer_order c ON c.driver_id = d.driver_id
WHERE c.order_date > d.Reg_date
GROUP BY d.driver_id;
```

Q11. Trend: Date-wise total ingredients used

```
SELECT
   c.order_date,
   COUNT(*) AS Ingredients_Used
FROM customer_order c
JOIN Rolls_Recipes rr ON rr.Roll_id = c.roll_id
CROSS APPLY STRING_SPLIT(rr.Ingredients, ',') AS s
GROUP BY c.order_date
ORDER BY c.order_date;
```

Q12. Report: Roll Name | Ingredients Count | Ingredients List

```
SELECT
   r.Rolls_Name,
   COUNT(*) AS Ingredients_Count,
   STRING_AGG(i.Ingredients_name, ', ') AS Ingredient_List
FROM Rolls r
JOIN Rolls_Recipes rr ON rr.Roll_id = r.Rolls_id
CROSS APPLY STRING_SPLIT(rr.Ingredients, ',') AS s
JOIN Ingredients i ON i.Ingredients_id = CAST(s.value AS INT)
GROUP BY r.Rolls_Name;
```

Q12. Find which rolls include “Mushrooms” as an ingredient.

```
SELECT r.Rolls_name, i.Ingredients_name
FROM Rolls r
INNER JOIN Rolls_recepie rr ON rr.Rolls_id = r.Roll_id
CROSS APPLY STRING_SPLIT(rr.Ingredients, ',') AS sr
INNER JOIN Ingredients i ON i.Ingredients_id = CAST(sr.value AS INT)
WHERE i.Ingredients_name = 'Mushrooms';
```

Q13. Get a list of ingredients that are used in more than one roll.

```
Select i.Ingredients_name from Ingredients i where i.Ingredients_id in(select value from(select
rr.Roll_id,
cast(value as int)as value from Rolls_Recipes rr
cross apply string_split(rr.Ingredients,','))as sub
Group by value
having count(distinct roll_id)>1);
```

Q14. Show all orders along with the roll name and the ingredients included (expanded).

```
SELECT
   cr.order_id,
   cr.customer_id,
   cr.roll_id,
   rr.Rolls_Name AS Roll_Name,
   i.Ingredients_name AS Ingredient
FROM customer_order AS cr
INNER JOIN Rolls_Recipes r ON r.Roll_id = cr.roll_id
CROSS APPLY STRING_SPLIT(r.Ingredients, ',') AS s
INNER JOIN Ingredients i ON i.Ingredients_id = CAST(s.value AS INT)
INNER JOIN Rolls rr ON rr.Rolls_id = r.Roll_id
```

```
ORDER BY cr.order_id;
```

Q15. For each roll, show the count of veg and non-veg ingredients.

Assumption: Ingredients with Ingredients_id 1–6 are Non-Veg, and 7–12 are Veg

```
SELECT
   r.Rolls_Name,
   SUM(CASE WHEN i.Ingredients_id BETWEEN 1 AND 6 THEN 1 ELSE 0 END) AS Non_Veg_Count,
   SUM(CASE WHEN i.Ingredients_id BETWEEN 7 AND 12 THEN 1 ELSE 0 END) AS Veg_Count
FROM Rolls r
INNER JOIN Rolls_Recipes rr ON r.Rolls_id = rr.Roll_id
CROSS APPLY STRING_SPLIT(rr.Ingredients, ',') AS s
INNER JOIN Ingredients i ON i.Ingredients_id = CAST(s.value AS INT)
GROUP BY r.Rolls_Name;

--Which ingredients have never been used in any roll?
SELECT Ingredients_name
FROM Ingredients
WHERE Ingredients_id NOT IN (
   SELECT DISTINCT CAST(value AS INT)
   FROM Rolls_Recipes
   CROSS APPLY STRING_SPLIT(Ingredients, ',')
);
```

Q16. Which ingredient has been included in the highest number of customer orders?

```
SELECT TOP 1
   i.Ingredients_name,
   COUNT(DISTINCT cr.order_id) AS total_orders
FROM customer_order AS cr
INNER JOIN Rolls_Recipes r ON r.Roll_id = cr.roll_id
CROSS APPLY STRING_SPLIT(r.Ingredients, ',') AS s
INNER JOIN Ingredients i ON i.Ingredients_id = CAST(s.value AS INT)
GROUP BY i.Ingredients_name
ORDER BY total_orders DESC;
```

Q17. Rank ingredients by frequency of appearance in orders (most used first).

```
SELECT
   i.Ingredients_name,
   COUNT(DISTINCT cr.order_id) AS total_orders,
   RANK() OVER (ORDER BY COUNT(DISTINCT cr.order_id) DESC) AS ingredient_rank
FROM customer_order AS cr
INNER JOIN Rolls_Recipes r ON r.Roll_id = cr.roll_id
CROSS APPLY STRING_SPLIT(r.Ingredients, ',') AS s
INNER JOIN Ingredients i ON i.Ingredients_id = CAST(s.value AS INT)
GROUP BY i.Ingredients_name
ORDER BY ingredient_rank;
```

Q18. Which customer has ordered the roll that contains the maximum number of ingredients the most number of times?

```
WITH MaxIngredientRoll AS (
   SELECT TOP 1 Roll_id
   FROM Rolls_Recipes
   CROSS APPLY STRING_SPLIT(Ingredients, ',')
   GROUP BY Roll_id
   ORDER BY COUNT(*) DESC
),
CustomerOrderCounts AS (
   SELECT customer_id, COUNT(*) AS order_count
   FROM customer_order
   WHERE roll_id = (SELECT Roll_id FROM MaxIngredientRoll)
   GROUP BY customer_id
)
SELECT TOP 1 customer_id, order_count
FROM CustomerOrderCounts
ORDER BY order_count DESC
```

Q19. Get the top 3 most frequently used ingredients across all orders.

```
SELECT TOP 3
   i.Ingredients_name,
   COUNT(*) AS frequency
FROM customer_order c
INNER JOIN Rolls_Recipes rr ON c.roll_id = rr.Roll_id
CROSS APPLY STRING_SPLIT(rr.Ingredients, ',') AS s
INNER JOIN Ingredients i ON i.Ingredients_id = CAST(s.value AS INT)
GROUP BY i.Ingredients_name
ORDER BY frequency DESC;
```

Q20. If you had to remove one ingredient (used least), which would it be?

```
SELECT TOP 1
   i.Ingredients_name,
   COUNT(*) AS usage_count
FROM customer_order c
INNER JOIN Rolls_Recipes rr ON c.roll_id = rr.Roll_id
CROSS APPLY STRING_SPLIT(rr.Ingredients, ',') AS s
INNER JOIN Ingredients i ON i.Ingredients_id = CAST(s.value AS INT)
GROUP BY i.Ingredients_name
ORDER BY usage_count ASC;
```

Q21. how many rolls were ordered

```
select count(roll_id) as no_of_ordered_roll from customer_order;
```

Q22. Which customer has given maximum orders

```
select top 2 customer_id,count(*) as total_roll from customer_order
group by customer_id
order by total_roll desc;
```

Q23. In how many rolls extra items included

```
select order_id ,count(*) as extra_items_included_Inroll from customer_order
where extra_items_included is not null
group by order_id
order by extra_items_included_Inroll desc;
```

Q23. On which date max order has done by customer

```
select cast(order_date as date),count(*) as max_order_date from customer_order
group by cast(order_date as date)
order by max_order_date desc;
```

Q24. Duplicate order done by the customer

```
select [order_id]
```

```
      ,[customer_id]
      ,[roll_id]
      ,[order_date]
              ,count(*) as duplicate_order
              from customer_order
group by [order_id],[customer_id],[roll_id],[order_date]
having count(*)>1;
```

Q25. Unique customer who has done order

```
select count(distinct [customer_id]) from customer_order;
```

Q26. No. of rolls has ordered by each customer

```
select customer_id, count(roll_id) as no_of_rollId from customer_order
group by customer_id
order by no_of_rollId desc;
```

Q27. How many times roll_id=2 has ordered

```
select count(*) from customer_order
where roll_id=2;
```

Q28. In which roll_id, not included item is not null

```
select count(roll_id) from customer_order
where not_include_items is not null;
```

Q29. Total orders per day according to order_date

```
select cast(order_date AS DATE), count(*) as total_order from customer_order
group by CAST(order_date AS DATE)
order by total_order desc;
```

Q30. On which date most extra items are included

```
select cast(order_date AS DATE),count(extra_items_included) as more_extra_items_included_date
from customer_order
group by cast(order_date AS DATE)
order by more_extra_items_included_date desc
```

Q31. In which order extra items included and not_included_items both are not present

```
select distinct order_id from customer_order
where extra_items_included is null and not_include_items is null
```

Q32. Which customer has done the first order

```
select top 1 order_id, customer_id from customer_order
order by order_date asc;
```

Q33. What are the last 3 orders

```
Select top 3 * from customer_order
order by cast(order_date as date) desc;
```

Conclusion

This SQL-based analysis provides a comprehensive view of how ingredients, rolls, and customer behavior interact in a food delivery system. Insights like popular items, usage trends, and delivery efficiency can drive business decisions.