**Project Title: Restaurant Data Analysis for Jomato. SQL Server Portfolio Project.**

**Introduction**

This project involves analyzing a dataset provided by a client who operates a food delivery platform similar to Jomato. The dataset contains information about various restaurants in a city, including restaurant types, cuisine types, ratings, and average costs. The goal is to extract valuable insights from this data using SQL Server, which can help the client in making data-driven decisions to improve their platform.

**Business Context**

The client, a food delivery platform, needs to understand the performance and characteristics of the restaurants listed on their platform. Insights into restaurant types, customer ratings, and average costs are crucial for optimizing their service offerings, enhancing customer satisfaction, and making strategic business decisions. By analyzing this data, the client can identify top-performing restaurants, assess customer preferences, and adjust their platform's algorithms to better serve their users.

**Tasks Performed**

1. **Creating a User-Defined Function: Adding 'Chicken' to 'Quick Bites'**
   * **Purpose**: The client wants to identify restaurants that offer quick chicken bites. To do this, we created a user-defined function that modifies the restaurant type by appending "Chicken" to any restaurant classified as "Quick Bites."
   * **SQL Techniques**:
     + Created a function using CREATE FUNCTION.
     + Used conditional logic (IF...ELSE) to modify the restaurant type.
   * **Query**:

CREATE FUNCTION addChickenIntoQuickBites(@restoType NVARCHAR(50))

RETURNS NVARCHAR(50)

AS

BEGIN

DECLARE @modifiedType NVARCHAR(50)

IF @restoType = 'Quick Bites'

SET @modifiedType = 'Quick Chicken Bites'

ELSE

SET @modifiedType = @restoType

RETURN @modifiedType

END

* + **Usage**:

SELECT RestaurantName, dbo.addChickenIntoQuickBites(RestaurantType) AS restoType

FROM jomato;

1. **Identifying the Restaurant with the Maximum Number of Ratings**
   * **Purpose**: The client needs to know which restaurant has received the most ratings, as this indicates popularity and customer engagement.
   * **SQL Techniques**:
     + Used a Common Table Expression (CTE) to find the maximum number of ratings.
     + Applied a WHERE clause to filter restaurants with this maximum rating.
   * **Query**:

WITH MaxRatings AS (

SELECT MAX([No of Rating]) AS MaxRating

FROM jomato

)

SELECT restaurantName, CuisinesType, [No.of.Rating], dbo.addChickenIntoQuickBites(RestaurantType) AS restoType

FROM Jomato

WHERE [No of Rating] = (SELECT MaxRating FROM MaxRatings);

1. **Creating a Rating Status Column**
   * **Purpose**: To categorize restaurants based on their rating, allowing the client to quickly assess the quality of restaurants on their platform.
   * **SQL Techniques**:
     + Used a CASE statement to create a new column that categorizes ratings into 'Excellent,' 'Good,' 'Average,' and 'Bad.'
   * **Query**:

SELECT \*,

CASE

WHEN Rating > 4 THEN 'Excellent'

WHEN Rating > 3.5 AND Rating <= 4 THEN 'Good'

WHEN Rating > 3 AND Rating <= 3.5 THEN 'Average'

ELSE 'Bad'

END AS RatingStatus

FROM Jomato;

1. **Calculating Ceil, Floor, and Absolute Values of Ratings**
   * **Purpose**: The client requested additional insights into the ratings, including their ceiling, floor, and absolute values, along with the current date and its components.
   * **SQL Techniques**:
     + Used SQL functions like CEILING, FLOOR, and ABS to calculate the required values.
     + Retrieved the current date using GETDATE() and extracted components like year, month, and day.
   * **Query**:

SELECT

RestaurantName,

Rating,

CEILING(Rating) AS CeilRating,

FLOOR(Rating) AS FloorRating,

ABS(Rating) AS AbsoluteRating,

GETDATE() AS CurrentDate,

YEAR(GETDATE()) AS Year,

DATENAME(MONTH, GETDATE()) AS monthName,

DAY(GETDATE()) AS Day

FROM Jomato;

1. **Displaying Restaurant Type and Total Average Cost Using ROLLUP**
   * **Purpose**: To calculate the average cost for each restaurant type and provide a summary that includes the total average cost across all types.
   * **SQL Techniques**:
     + Used the ROLLUP operator with GROUP BY to generate subtotals and a grand total for the average cost.
   * **Query**:

SELECT

RestaurantType,

AVG(AverageCost) AS TotalAvgCost

FROM

Jomato

GROUP BY

ROLLUP(RestaurantType);

**6. Creating a Stored Procedure: Filtering Restaurants with Table Booking Availability**

* **Purpose**: The client wanted to identify restaurants that offer table booking, filtering out those that do not.
* **SQL Techniques**:
  + Created a stored procedure using CREATE PROCEDURE.
  + Used a SELECT statement within the stored procedure to filter restaurants based on table booking availability.
  + **Query**:

CREATE PROCEDURE GetRestaurantsWithTableBooking

AS

BEGIN

SELECT

RestaurantName,

RestaurantType,

CuisinesType

FROM

Jomato

WHERE

TableBooking != 0;

END;

* **Usage**:

EXEC GetRestaurantsWithTableBooking;

**7. Creating a Transaction to Update and Rollback Cuisine Type**

* **Purpose**: To demonstrate the ability to perform updates within a transaction and roll them back if necessary, ensuring data integrity.
* **SQL Techniques**:
  + Used a BEGIN TRANSACTION block to start the transaction.
  + Performed an UPDATE to change the cuisine type from 'Cafe' to 'Cafeteria'.
  + Rolled back the transaction using ROLLBACK to revert the changes.
  + **Query**:

BEGIN TRANSACTION;

UPDATE Jomato

SET CuisinesType = 'Cafeteria'

WHERE CuisinesType = 'Cafe';

-- Check the result

SELECT \* FROM Jomato WHERE CuisinesType = 'Cafeteria';

-- Rollback the transaction

ROLLBACK;

**8. Generating Row Numbers and Finding Top 5 Areas with Highest Ratings**

* **Purpose**: The client wanted to identify the top 5 areas with the highest-rated restaurants.
* **SQL Techniques**:
  + Used the ROW\_NUMBER() function to generate row numbers for each record.
  + Filtered the top 5 areas based on the highest ratings.
  + **Query**:

SELECT

Area,

Rating,

ROW\_NUMBER() OVER (ORDER BY Rating DESC) AS RowNum

FROM

Jomato

WHERE

RowNum <= 5;

**9. Using a WHILE Loop to Display Numbers from 1 to 50**

* **Purpose**: Demonstrate control flow using a WHILE loop.
* **SQL Techniques**:
  + Used a WHILE loop to iterate and display numbers from 1 to 50.
  + **Query**:

DECLARE @Counter INT = 1;

WHILE @Counter <= 50

BEGIN

PRINT @Counter;

SET @Counter = @Counter + 1;

END;

**10. Creating a View to Store the Top 5 Highest-Rated Restaurants**

* **Purpose**: To create a persistent view that stores the top 5 highest-rated restaurants, making it easier for the client to access this information.
* **SQL Techniques**:
  + Used the CREATE VIEW statement to define a view that captures the top 5 highest-rated restaurants.
  + **Query**:

CREATE VIEW TopRatedRestaurants AS

SELECT

RestaurantName,

Rating

FROM

Jomato

ORDER BY

Rating DESC

LIMIT 5;

**11. Creating a Trigger to Notify on New Record Insertion**

* **Purpose**: To create a trigger that notifies the client whenever a new record is inserted into the Jomato table.
* **SQL Techniques**:
  + Used the CREATE TRIGGER statement to define a trigger that activates on INSERT.
  + **Query**:

CREATE TRIGGER NotifyNewRestaurantInsertion

ON Jomato

AFTER INSERT

AS

BEGIN

PRINT 'A new restaurant record has been inserted.';

END;

**Results and Insights**

1. **Modified Restaurant Types**:
   * The user-defined function successfully modified "Quick Bites" to "Quick Chicken Bites," allowing the client to easily identify restaurants offering chicken in this category.
2. **Top-Rated Restaurant**:
   * The query identified the restaurant with the maximum number of ratings, highlighting it as a potential benchmark for other restaurants on the platform.
3. **Rating Categorization**:
   * The RatingStatus column categorized restaurants based on their ratings, providing a quick overview of restaurant quality.
4. **Additional Rating Insights**:
   * The ceiling, floor, and absolute values of ratings were calculated, offering the client deeper insights into the rating distribution.
5. **Cost Analysis**:
   * The ROLLUP operation provided a summary of average costs for each restaurant type and a grand total, aiding the client in understanding the pricing trends across different types of restaurants.
6. **Filtered Restaurant List**:

* The stored procedure successfully filtered restaurants that offer table booking, allowing the client to focus on those with this feature.

7. **Transaction and Rollback**:

* The transaction was demonstrated successfully, showing the client how changes can be made and reverted if necessary, ensuring data accuracy.

1. **Top Areas by Ratings**:

* The top 5 areas with the highest-rated restaurants were identified, giving the client insights into prime locations for quality dining.

1. **WHILE Loop Execution**:

* The loop successfully displayed numbers from 1 to 50, demonstrating control flow capabilities in SQL.

10. **Persistent View of Top-Rated Restaurants**:

* The view provided an easy way for the client to access the top 5 highest-rated restaurants without needing to run complex queries repeatedly.

11. **Trigger for New Record Insertion**:

* The trigger effectively notified the client of new restaurant records being added, enhancing data monitoring.

**Conclusion**

This project successfully demonstrated the use of SQL Server to analyze restaurant data for a food delivery platform. By creating user-defined functions, categorizing data, and performing advanced queries, valuable insights were extracted that can help the client optimize their platform and make informed business decisions.