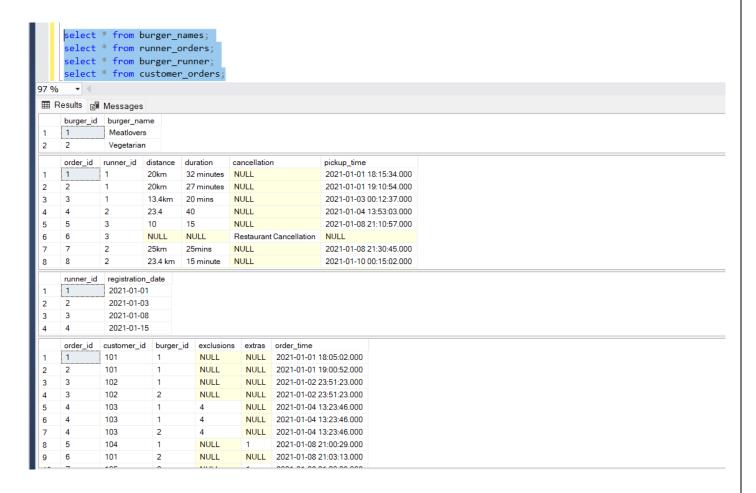
Coding Challenge – 1 (Sql) Burger Bash

Name: Aathirainathan P

Date: 08-11-2024

I. Creating the Database & Tables:



The four tables that were necessary were created.

II. Topics:

1. Querying Data by Using Joins and Subqueries & Subtotal:

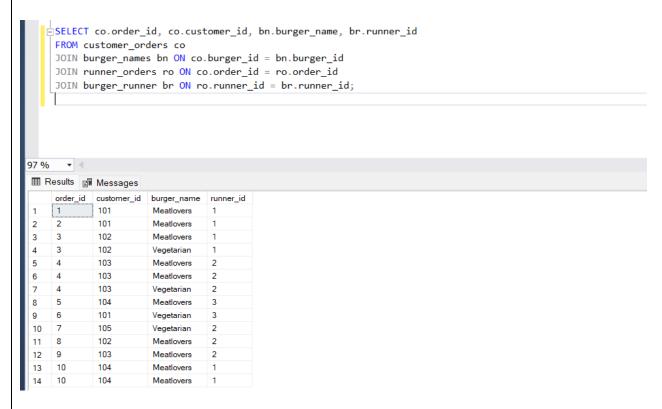
SQL Joins:

SQL Joins are used to combine rows from two or more tables based on related fields between them. When we work with multiple tables, joins allow us to retrieve related data across those tables.

> Types:

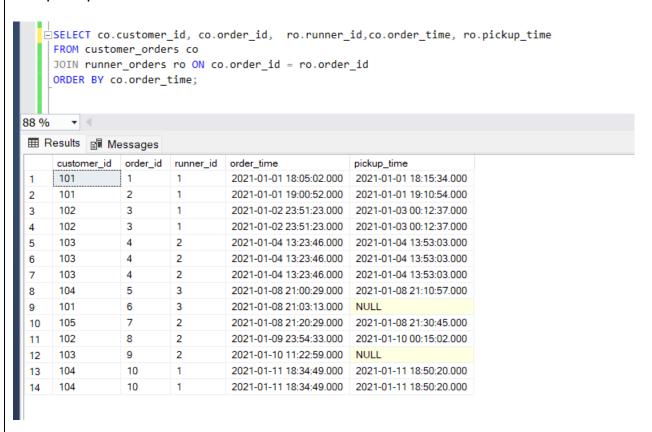
- 1. Inner Join
- 2. Left Join
- 3. Right Join
- 4. Outer Join
- 5. Cross Join
- 6. Self Join

Query 1: Get the list of orders with order id, customer id, runner id and burger names:



This query joins four tables () to return the details of customer orders along with **burger names** and runner IDs.

Query 2: Get a list of customers and their corresponding runner details along with order time and pickup time.

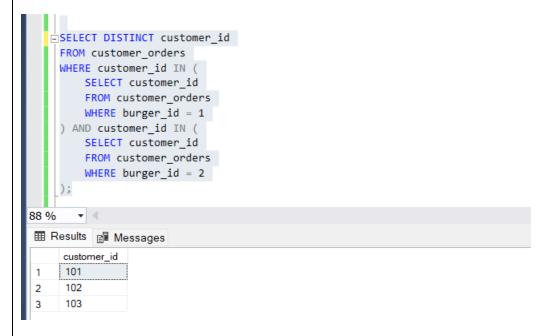


This query returns customer order details along with the **runner ID and pickup time**, showing the chronological order of orders.

SubQuery:

- A **subquery** is a query within another query. Subqueries can be used to perform operations that depend on the results of another query.
- ➤ A correlated subquery is a subquery that refers to columns of the outer query, making it dependent on the outer query.

Query 3: Find the customers who have ordered both burgers:



This query finds customers who have ordered **both burger ID 1** and burger **ID 2** by using two subqueries. The outer query checks if a customer exists in both subquery results.

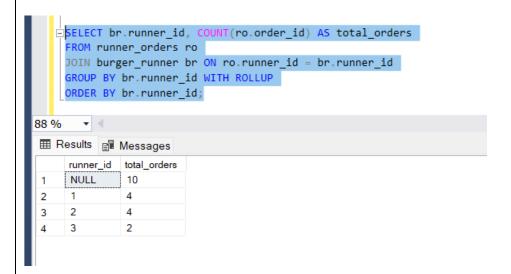
Query 4: Find the runner who handled the most orders:

This query uses a subquery to first calculate the order count for each runner, and then finds the runner with the **highest number of orders**.

Subtotals:

A **subtotal** is a partial sum, which is grouped by certain attributes. Subtotals can be calculated using **group by and aggregate functions** like SUM(), AVG(), COUNT(), etc.

Query 5: Calculate the total number of orders handled by each runner and show the subtotal for each runner:



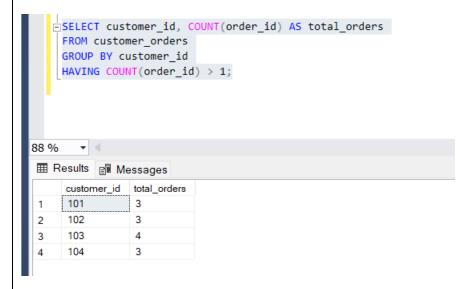
This query returns the total number of orders handled by each runner by grouping the results by runner_id and using the COUNT() function to calculate the number of orders per runner

2. Manipulating Data Using SQL Commands: GROUP BY and HAVING Clause

Groupby and Having Clause:

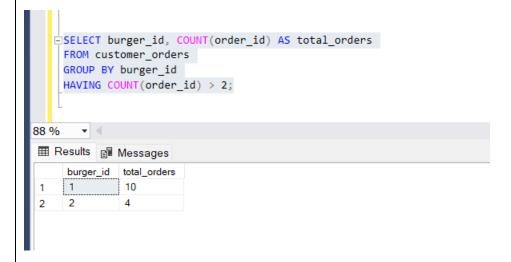
- The GROUP BY clause is used to arrange identical data into groups, usually with aggregate functions like **SUM()**, **COUNT()**, **AVG()**, etc. The HAVING clause is used to filter the results after the grouping, whereas WHERE filters the rows before grouping.
- ➤ WHERE is used to filter rows **before** aggregation.
- ➤ HAVING is used to filter groups **after** aggregation.

Query 6: Get the number of orders per customer where the total is greater than 1.



This query groups the **customer_orders by customer_id** and filters the results using **HAVING** to only include customers who have placed more than one order.

Query 7: Get the number of orders per burger type, only for those with more than 2 orders.



This query groups the orders by **burger_id** and filters them to only include burger types with **more than two orders**.

Submitted by:

Aathirainathan P