**Coding Assessment-SQL**

CREATE DATABASE burger\_bash;

USE burger\_bash;

CREATE TABLE burger\_names (

burger\_id INT PRIMARY KEY,

burger\_name VARCHAR(100)

);

INSERT INTO burger\_names VALUES

(1, 'Beef Burger'),

(2, 'Vegetarian Burger'),

(3, 'Chicken Burger'),

(4, 'Meatlovers Burger');

CREATE TABLE burger\_runner (

runner\_id INT PRIMARY KEY,

registration\_date DATE

);

INSERT INTO burger\_runner VALUES

(1, '2022-01-01'),

(2, '2022-01-08'),

(3, '2022-01-15');

--DROP TABLE IF EXISTS runner\_orders;

--DROP TABLE IF EXISTS customer\_orders;

CREATE TABLE customer\_orders (

order\_id INT,

customer\_id INT,

burger\_id INT,

exclusions VARCHAR(100),

extras VARCHAR(100),

order\_time DATETIME

);

CREATE TABLE runner\_orders (

order\_id INT,

runner\_id INT,

pickup\_time DATETIME,

distance VARCHAR(20),

duration VARCHAR(20),

cancellation VARCHAR(20)

);

INSERT INTO customer\_orders (order\_id, customer\_id, burger\_id, exclusions, extras, order\_time) VALUES

(101, 1, 1, NULL, 'Cheese', '2022-01-05 12:00:00'),

(102, 1, 2, 'Onion', NULL, '2022-01-05 13:10:00'),

(103, 2, 2, NULL, NULL, '2022-01-06 14:05:00'),

(104, 3, 4, NULL, 'Bacon', '2022-01-07 12:05:00'),

(105, 2, 3, 'Lettuce', NULL, '2022-01-08 11:30:00'),

(106, 1, 4, NULL, NULL, '2022-01-08 14:23:00');

INSERT INTO runner\_orders (order\_id, runner\_id, pickup\_time, distance, duration, cancellation) VALUES

(101, 1, '2022-01-05 12:30:00', '5km', '30min', NULL),

(102, 2, '2022-01-05 13:40:00', '3km', '20min', NULL),

(103, 1, '2022-01-06 14:31:00', '6km', '33min', 'Cancelled'),

(104, 3, '2022-01-07 12:32:00', '4km', '25min', NULL),

(105, 2, NULL, NULL, NULL, 'Cancelled'),

(106, 2, '2022-01-08 14:56:00', '2km', '10min', NULL);

SELECT COUNT(\*) AS total\_burgers\_ordered FROM customer\_orders;

SELECT COUNT(DISTINCT order\_id) AS unique\_customer\_orders FROM customer\_orders;

SELECT runner\_id, COUNT(\*) AS successful\_orders

FROM runner\_orders

WHERE cancellation IS NULL

GROUP BY runner\_id;

SELECT b.burger\_name, COUNT(\*) AS burgers\_delivered

FROM customer\_orders c

JOIN runner\_orders r ON c.order\_id = r.order\_id

JOIN burger\_names b ON c.burger\_id = b.burger\_id

WHERE r.cancellation IS NULL

GROUP BY b.burger\_name;

SELECT customer\_id,

SUM(CASE WHEN burger\_id = 2 THEN 1 ELSE 0 END) AS Vegetarian\_Burger,

SUM(CASE WHEN burger\_id = 4 THEN 1 ELSE 0 END) AS Meatlovers\_Burger

FROM customer\_orders

GROUP BY customer\_id;

**Using Joins and Aggregate Functions**

**-- 1. List all delivered burger orders along with burger names and runner IDs**

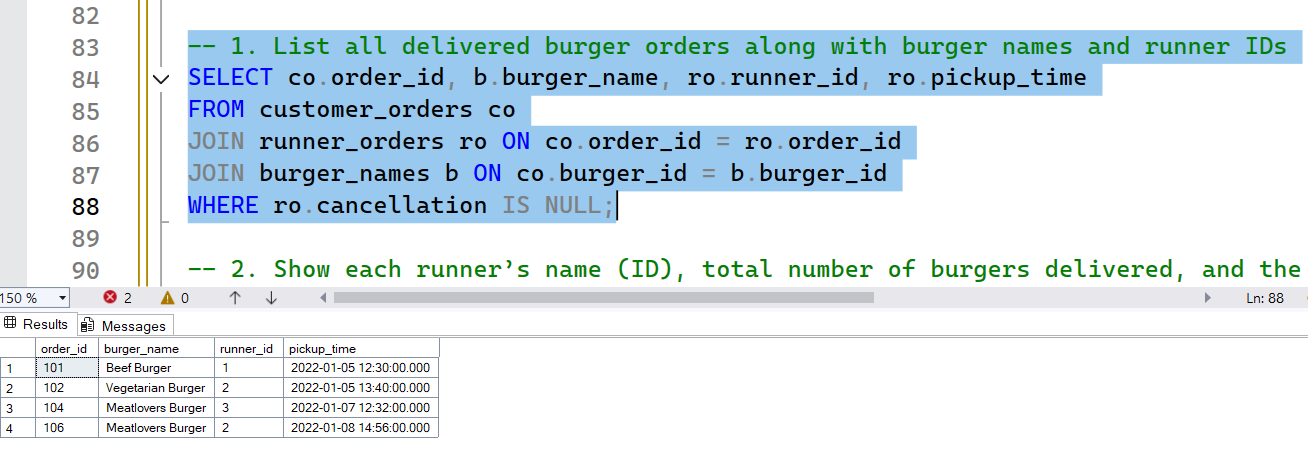
SELECT co.order\_id, b.burger\_name, ro.runner\_id, ro.pickup\_time

FROM customer\_orders co

JOIN runner\_orders ro ON co.order\_id = ro.order\_id

JOIN burger\_names b ON co.burger\_id = b.burger\_id

WHERE ro.cancellation IS NULL;



**-- 2.** **Show each runner’s name (ID), total number of burgers delivered, and the burger types they've delivered**

SELECT ro.runner\_id, b.burger\_name, COUNT(\*) AS total\_deliveries

FROM runner\_orders ro

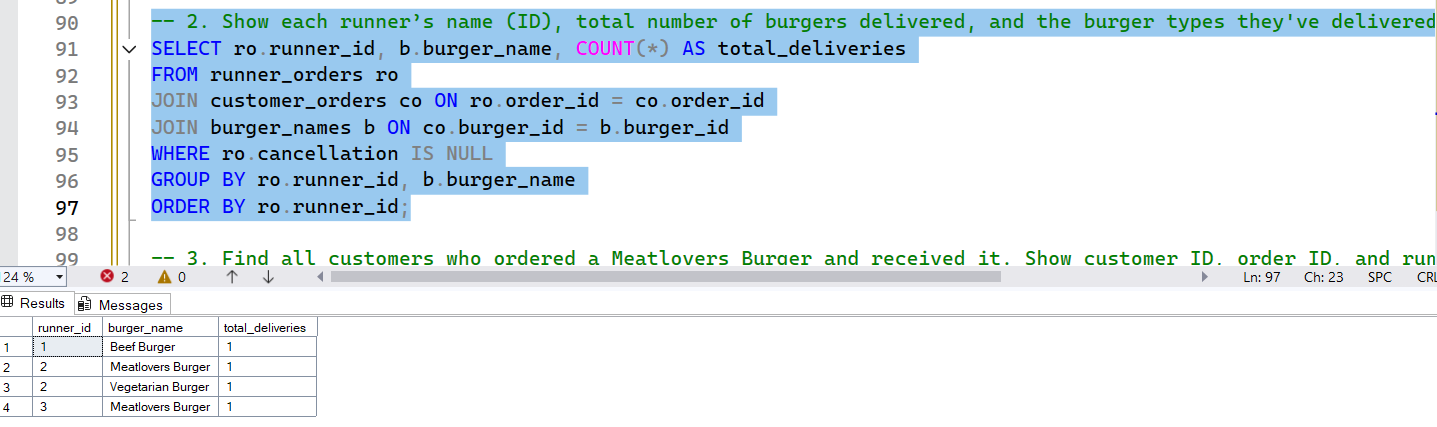
JOIN customer\_orders co ON ro.order\_id = co.order\_id

JOIN burger\_names b ON co.burger\_id = b.burger\_id

WHERE ro.cancellation IS NULL

GROUP BY ro.runner\_id, b.burger\_name

ORDER BY ro.runner\_id;



**-- 3. Find all customers who ordered a Meatlovers Burger and received it. Show customer ID, order ID, and runner.**

SELECT co.customer\_id, co.order\_id, ro.runner\_id

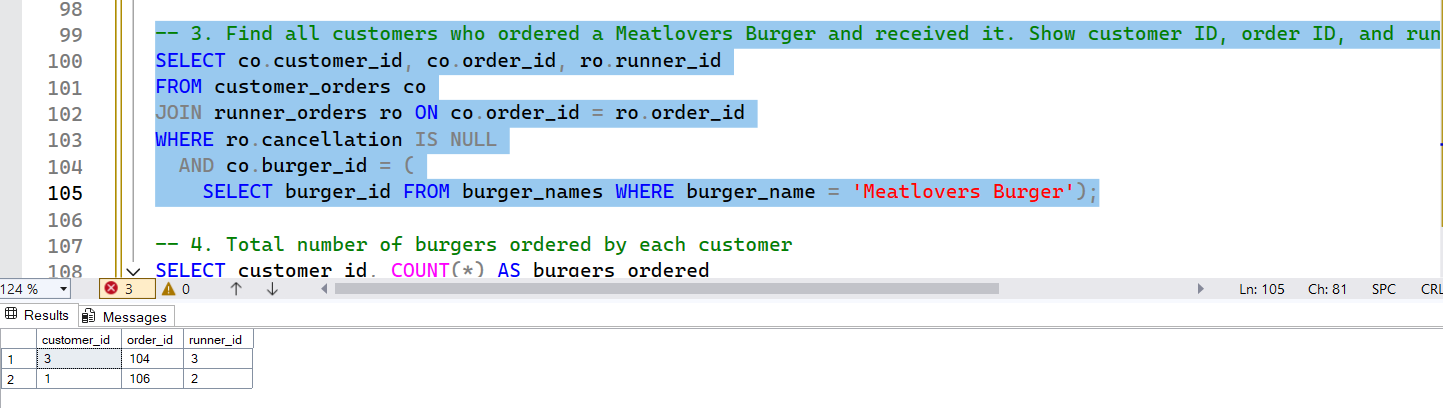
FROM customer\_orders co

JOIN runner\_orders ro ON co.order\_id = ro.order\_id

WHERE ro.cancellation IS NULL

AND co.burger\_id = (

SELECT burger\_id FROM burger\_names WHERE burger\_name = 'Meatlovers Burger');



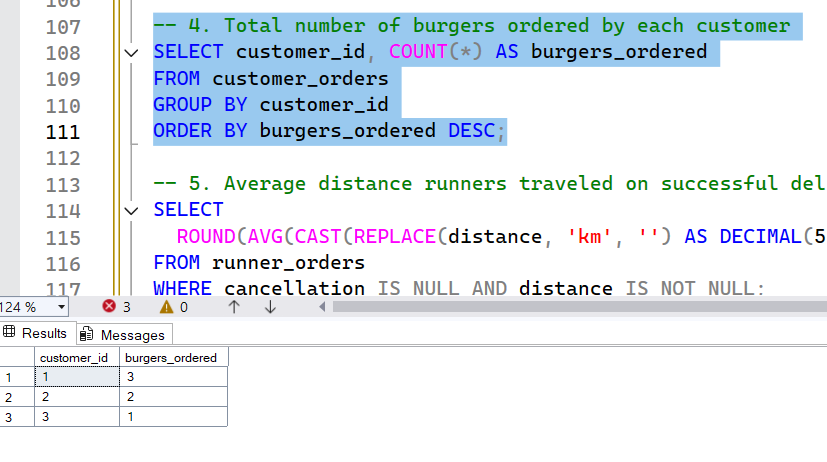
**-- 4. Total number of burgers ordered by each customer**

SELECT customer\_id, COUNT(\*) AS burgers\_ordered

FROM customer\_orders

GROUP BY customer\_id

ORDER BY burgers\_ordered DESC;



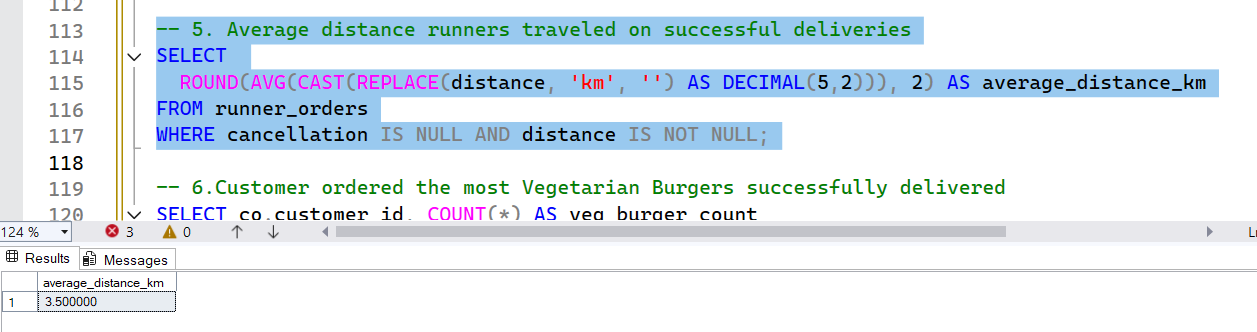
**-- 5. Average distance runners travelled on successful deliveries**

SELECT

ROUND(AVG(CAST(REPLACE(distance, 'km', '') AS DECIMAL(5,2))), 2) AS average\_distance\_km

FROM runner\_orders

WHERE cancellation IS NULL AND distance IS NOT NULL;



**-- 6. Customer ordered the most Vegetarian Burgers successfully delivered**

SELECT co.customer\_id, COUNT(\*) AS veg\_burger\_count

FROM customer\_orders co

JOIN runner\_orders ro ON co.order\_id = ro.order\_id

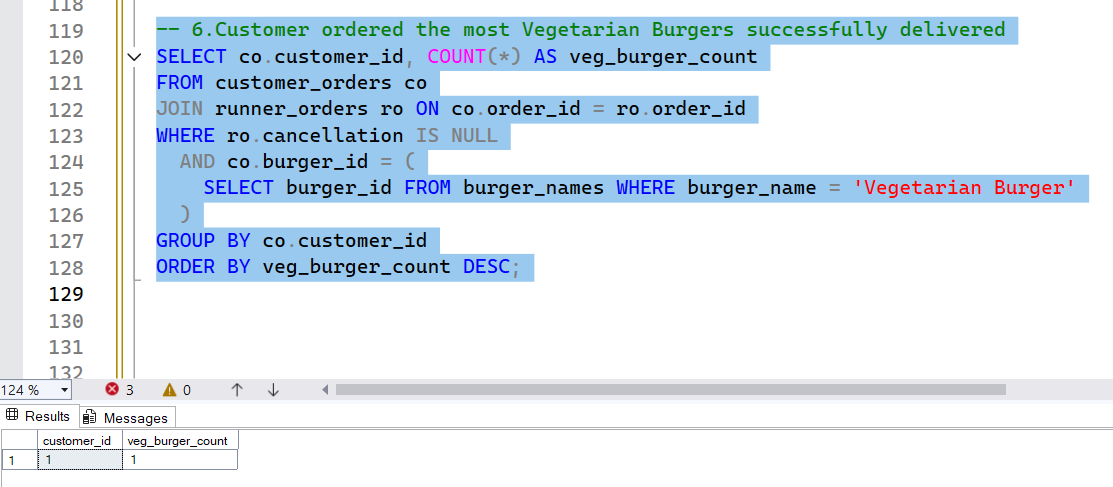
WHERE ro.cancellation IS NULL

AND co.burger\_id = (

SELECT burger\_id FROM burger\_names WHERE burger\_name = 'Vegetarian Burger' )

GROUP BY co.customer\_id

ORDER BY veg\_burger\_count DESC;



**Submitted by**

Siva Balan T