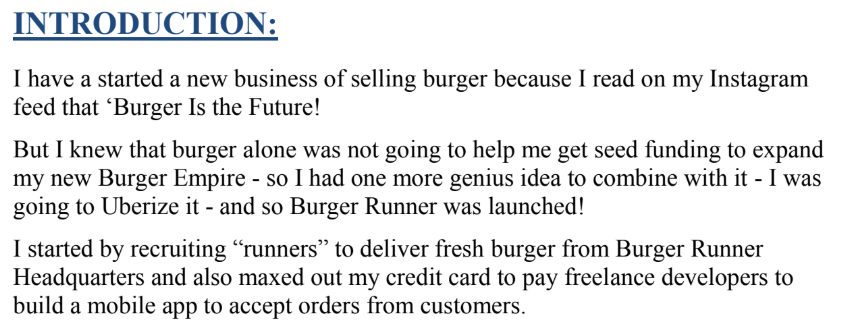
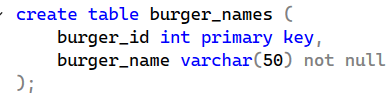
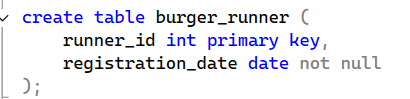
**CODING ASSIGNMENT**

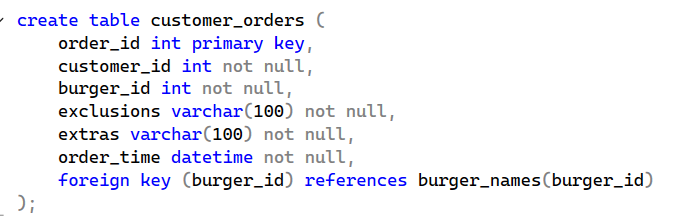
**BURGER BASH**

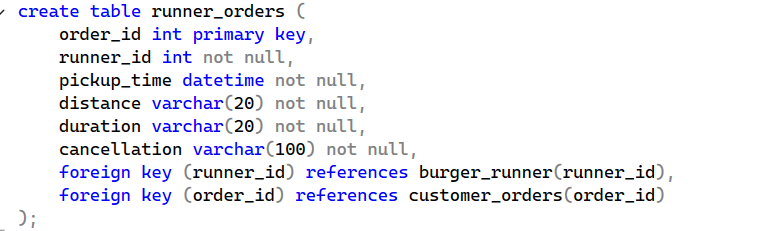


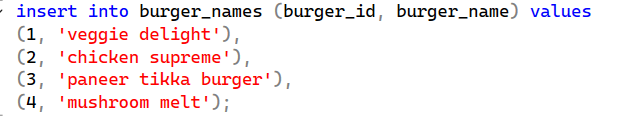


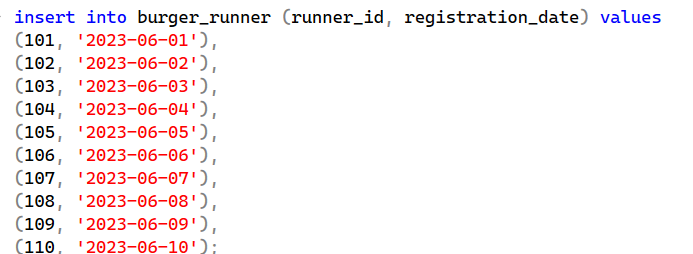


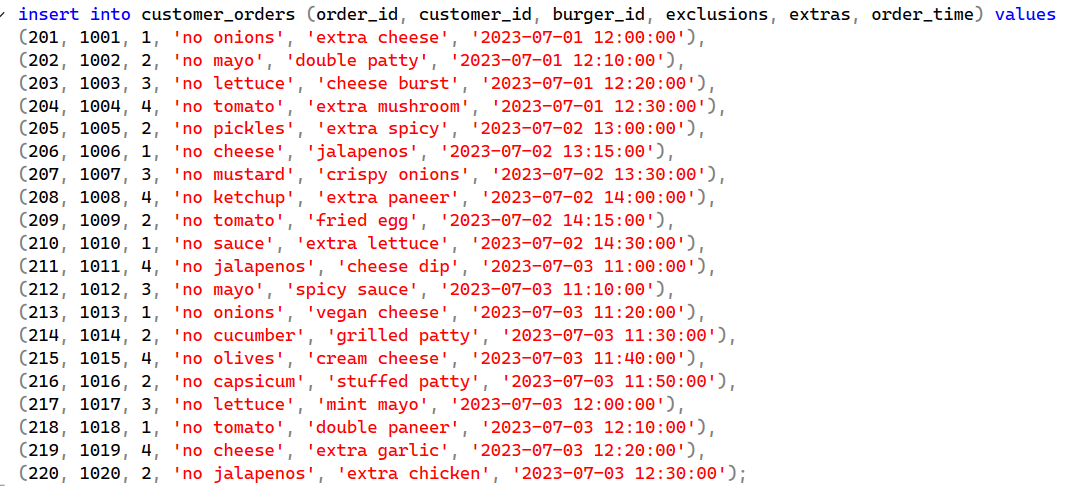


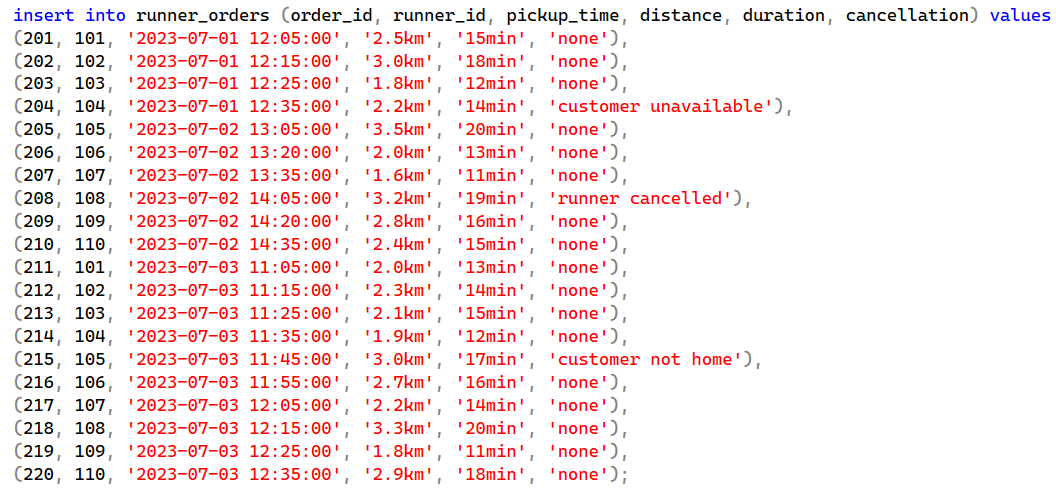






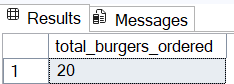






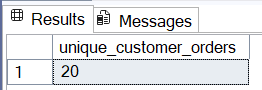
**-- 1. Count how many burgers were ordered**

select count(\*) as total\_burgers\_ordered from customer\_orders;



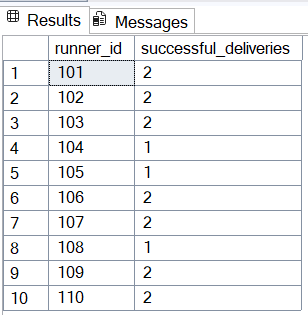
**-- 2. Count how many unique customer orders were made**

select count(distinct order\_id) as unique\_customer\_orders from customer\_orders;



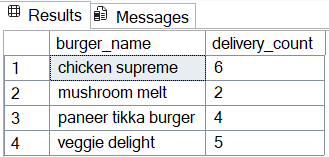
**-- 3. Count how many successful orders were delivered by each runner**

select runner\_id, count(\*) as successful\_deliveries from runner\_orders where cancellation is null or lower(cancellation) = 'none' group by runner\_id;



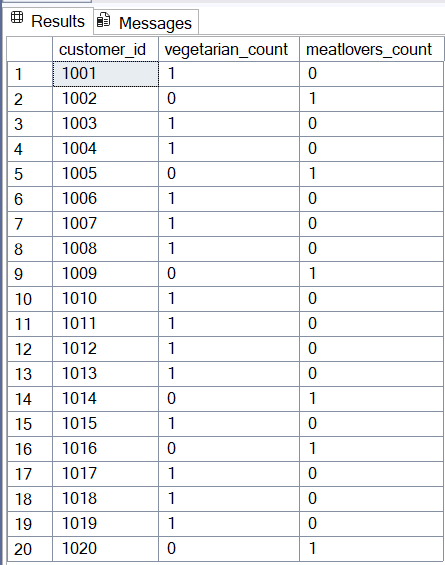
**-- 4. Count how many of each type of burger was delivered**

select b.burger\_name, count(\*) as delivery\_count 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 or lower(r.cancellation) = 'none' group by b.burger\_name;



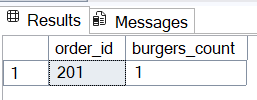
**-- 5. Count how many Vegetarian and Meatlovers burgers were ordered by each customer**

select customer\_id, sum(case when burger\_id in (1, 3, 4) then 1 else 0 end) as vegetarian\_count, sum(case when burger\_id = 2 then 1 else 0 end) as meatlovers\_count from customer\_orders group by customer\_id;



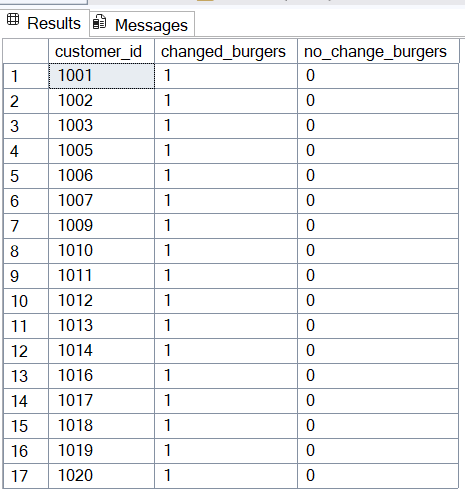
**-- 6. Get the maximum number of burgers delivered in a single order**

select top 1 c.order\_id, count(\*) as burgers\_count from customer\_orders c join runner\_orders r on c.order\_id = r.order\_id where r.cancellation is null or lower(r.cancellation) = 'none' group by c.order\_id order by burgers\_count desc;



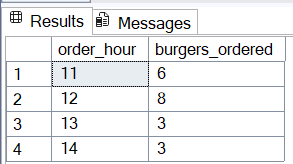
**-- 7. For each customer, count how many delivered burgers had at least 1 change and how many had no changes**

select customer\_id, sum(case when (exclusions is not null and exclusions <> '') or (extras is not null and extras <> '') then 1 else 0 end) as changed\_burgers, sum(case when (exclusions is null or exclusions = '') and (extras is null or extras = '') then 1 else 0 end) as no\_change\_burgers from customer\_orders c join runner\_orders r on c.order\_id = r.order\_id where r.cancellation is null or lower(r.cancellation) = 'none' group by customer\_id;



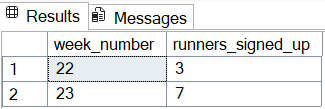
**-- 8. Show the total volume of burgers ordered for each hour of the day**

select datepart(hour, order\_time) as order\_hour, count(\*) as burgers\_ordered from customer\_orders group by datepart(hour, order\_time) order by order\_hour;



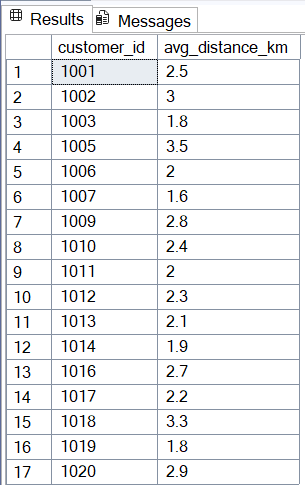
**-- 9. Show how many runners signed up for each 1 week period**

select datepart(week, registration\_date) as week\_number, count(\*) as runners\_signed\_up from burger\_runner group by datepart(week, registration\_date) order by week\_number;



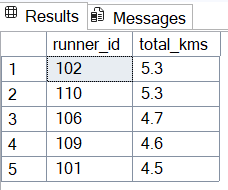
**-- 10. Get the average distance travelled for each customer**

select c.customer\_id, avg(cast(replace(r.distance, 'km', '') as float)) as avg\_distance\_km from customer\_orders c join runner\_orders r on c.order\_id = r.order\_id where r.cancellation is null or lower(r.cancellation) = 'none' group by c.customer\_id;



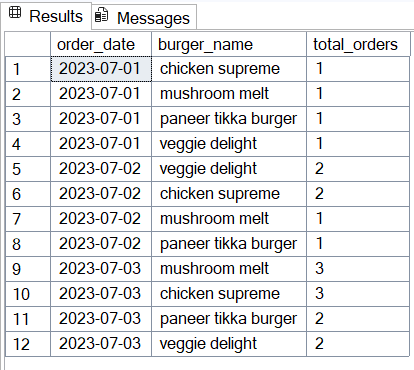
**-- Complex Query 1: Most active runners with max total distance**

SELECT TOP 5 r.runner\_id, SUM(CAST(REPLACE(r.distance, 'km', '') AS FLOAT)) AS total\_kms FROM runner\_orders r WHERE r.cancellation IS NULL OR LOWER(r.cancellation) = 'none' GROUP BY r.runner\_id ORDER BY total\_kms DESC;



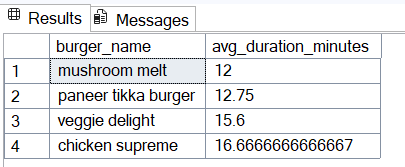
**-- Complex Query 2: Burger type popularity per day**

SELECT CAST(order\_time AS DATE) AS order\_date, b.burger\_name, COUNT(\*) AS total\_orders FROM customer\_orders c JOIN burger\_names b ON c.burger\_id = b.burger\_id GROUP BY CAST(order\_time AS DATE), b.burger\_name ORDER BY order\_date, total\_orders DESC;



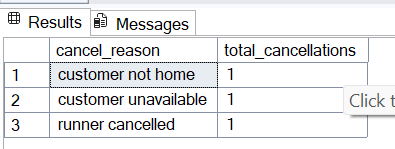
**-- Complex Query 3: Average duration of delivery by burger type (only successful)**

SELECT b.burger\_name, AVG(CAST(REPLACE(r.duration, 'min', '') AS FLOAT)) AS avg\_duration\_minutes 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 OR LOWER(r.cancellation) = 'none' GROUP BY b.burger\_name ORDER BY avg\_duration\_minutes;



**-- Complex Query 4: Number of cancelled orders by reason**

SELECT LOWER(cancellation) AS cancel\_reason, COUNT(\*) AS total\_cancellations FROM runner\_orders WHERE cancellation IS NOT NULL AND LOWER(cancellation) <> 'none' GROUP BY LOWER(cancellation);



**-- Complex Query 5: Rank runners by number of deliveries per day**

SELECT CAST(pickup\_time AS DATE) AS delivery\_date, runner\_id, COUNT(*) AS deliveries, RANK() OVER (PARTITION BY CAST(pickup\_time AS DATE) ORDER BY COUNT(*) DESC) AS delivery\_rank FROM runner\_orders WHERE cancellation IS NULL OR LOWER(cancellation) = 'none' GROUP BY CAST(pickup\_time AS DATE), runner\_id ORDER BY delivery\_date, delivery\_rank;

