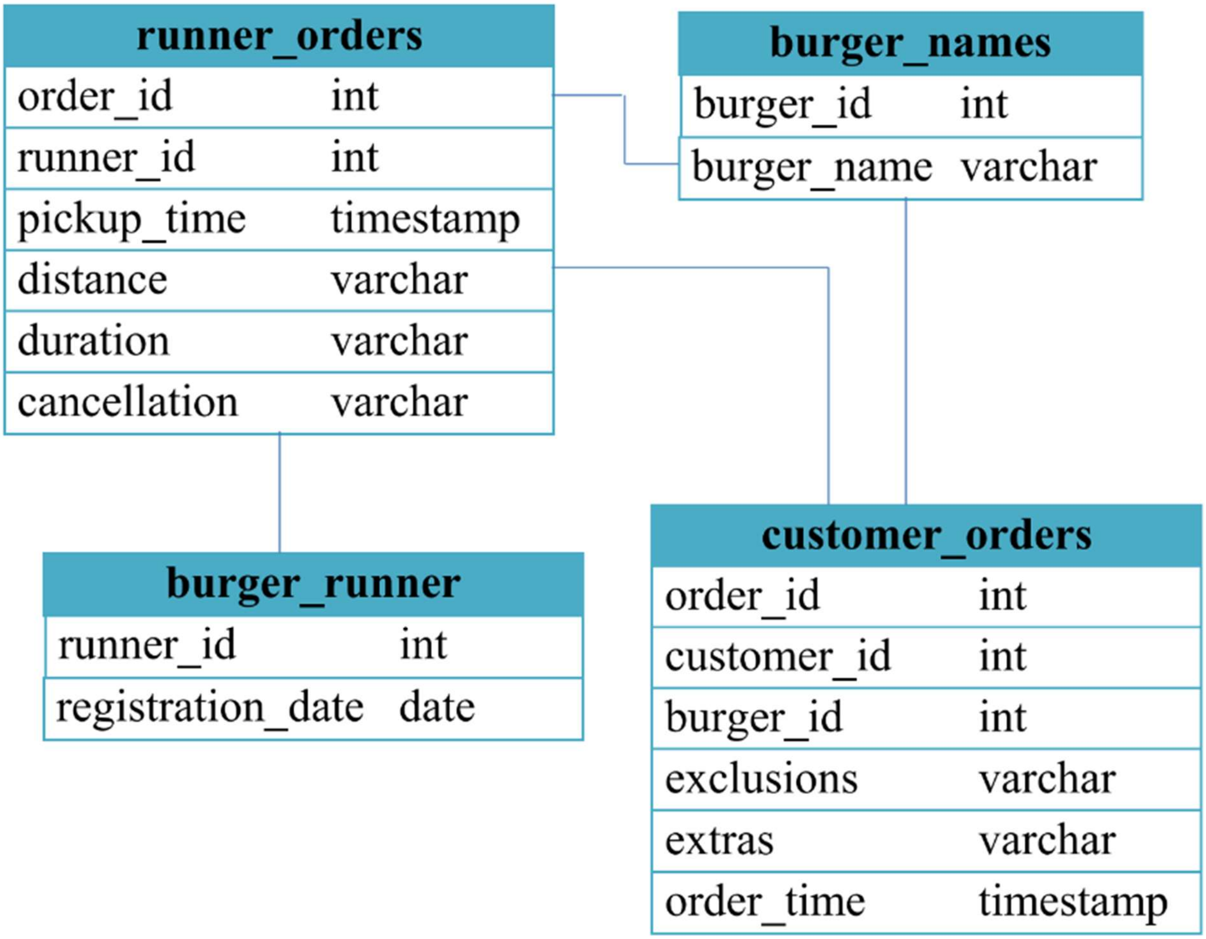
SQL Case Study 2: Burger Bash

CASE:

Have to develop Database to store all the transactions made in an online burger ordering system. In database terms four transaction takes place. They are while adding new burger variety, while burger runner is added into database, while customer ordering and while delivering the order. There are four tables present. They are runner_orders, burger_names, burger_runner, customer_orders. Schema of the database is shown below.

SCHEMA:



runner_orders	
order_id	int
runner_id	int
pickup_time	timestamp
distance	varchar
duration	varchar
cancellation	varchar

burger_names	
burger_id	int
burger_name	varchar

burger_runner	
runner_id	int
registration_date	date

customer_orders	
order_id	int
customer_id	int
burger_id	int
exclusions	varchar
extras	varchar
order_time	timestamp

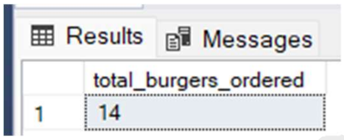
SOLUTION

```
CREATE DATABASE Case_Study;
USE Case_Study;

                                    --TABLE CREATION

CREATE TABLE burger_names(
   burger_id   INT PRIMARY KEY ,       burger_name VARCHAR(10) NOT NULL
);

CREATE TABLE burger_runner(
   runner_id   INT PRIMARY KEY,        registration_date date NOT NULL
);

CREATE TABLE customer_orders(
order_id    INT  NOT NULL,     customer_id INT NOT NULL,   burger_id    INT  NOT NULL,
exclusions  VARCHAR(4),        extras  VARCHAR(4),   order_time  timestamp NOT NULL
);

ALTER TABLE customer_orders
ADD FOREIGN KEY (burger_id) REFERENCES burger_names(burger_id);

CREATE TABLE runner_orders(
order_id     INT PRIMARY KEY,          runner_id    INT  NOT NULL,   pickup_time  timestamp,
distance     VARCHAR(7),        duration     VARCHAR(10),       cancellation VARCHAR(23)
);

ALTER TABLE runner_orders
ADD FOREIGN KEY (runner_id) REFERENCES burger_runner(runner_id);

                                --ADD DATA INTO DATABASE

INSERT INTO burger_names(burger_id,burger_name) VALUES (1,'Meatlovers');
INSERT INTO burger_names(burger_id,burger_name) VALUES (2,'Vegetarian');

INSERT INTO burger_runner VALUES (1,'2021-01-01');
INSERT INTO burger_runner VALUES (2,'2021-01-03');
INSERT INTO burger_runner VALUES (3,'2021-01-08');
INSERT INTO burger_runner VALUES (4,'2021-01-15');

INSERT INTO customer_orders VALUES (1,101,1,NULL,NULL,'2021-01-01 18:05:02');
INSERT INTO customer_orders VALUES (2,101,1,NULL,NULL,'2021-01-01 19:00:52');
INSERT INTO customer_orders VALUES (3,102,1,NULL,NULL,'2021-01-02 23:51:23');
INSERT INTO customer_orders VALUES (3,102,2,NULL,NULL,'2021-01-02 23:51:23');
INSERT INTO customer_orders VALUES (4,103,1,'4',NULL,'2021-01-04 13:23:46');
INSERT INTO customer_orders VALUES (4,103,1,'4',NULL,'2021-01-04 13:23:46');
INSERT INTO customer_orders VALUES (4,103,2,'4',NULL,'2021-01-04 13:23:46');
INSERT INTO customer_orders VALUES (5,104,1,NULL,'1','2021-01-08 21:00:29');
INSERT INTO customer_orders VALUES (6,101,2,NULL,NULL,'2021-01-08 21:03:13');
INSERT INTO customer_orders VALUES (7,105,2,NULL,'1','2021-01-08 21:20:29');
INSERT INTO customer_orders VALUES (8,102,1,NULL,NULL,'2021-01-09 23:54:33');
INSERT INTO customer_orders VALUES (9,103,1,'4','1, 5','2021-01-10 11:22:59');
INSERT INTO customer_orders VALUES (10,104,1,NULL,NULL,'2021-01-11 18:34:49');
INSERT INTO customer_orders VALUES (10,104,1,'2, 6','1, 4','2021-01-11 18:34:49');

INSERT INTO runner_orders VALUES (1,1,'2021-01-01 18:15:34','20km','32 minutes',NULL);
INSERT INTO runner_orders VALUES (2,1,'2021-01-01 19:10:54','20km','27 minutes',NULL);
INSERT INTO runner_orders VALUES (3,1,'2021-01-03 00:12:37','13.4km','20 mins',NULL);
INSERT INTO runner_orders VALUES (4,2,'2021-01-04 13:53:03','23.4','40',NULL);
INSERT INTO runner_orders VALUES (5,3,'2021-01-08 21:10:57','10','15',NULL);
INSERT INTO runner_orders VALUES (6,3,NULL,NULL,NULL,'Restaurant Cancellation');
INSERT INTO runner_orders VALUES (7,2,'2021-01-08 21:30:45','25km','25mins',NULL);
INSERT INTO runner_orders VALUES (8,2,'2021-01-10 00:15:02','23.4 km','15 minute',NULL);
INSERT INTO runner_orders VALUES (9,2,NULL,NULL,NULL,'Customer Cancellation');
INSERT INTO runner_orders VALUES (10,1,'2021-01-11 18:50:20','10km','10minutes',NULL);
```

QUESTIONS AND SOLUTIONS

1. How many burgers were ordered?

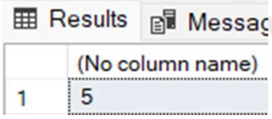
```
SELECT COUNT(*) AS total_burgers_ordered FROM customer_orders;
```



	total_burgers_ordered
1	14

2. How many unique customer orders were made?

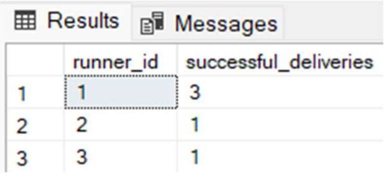
```
SELECT  COUNT( DISTINCT customer_id) from customer_orders;
```



	(No column name)
1	5

3. How many successful orders were delivered by each runner?

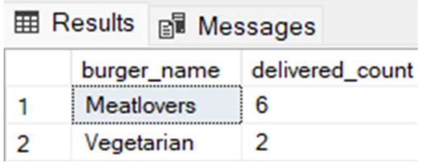
```
SELECT runner_id, COUNT(order_id) AS successful_deliveries FROM runner_orders WHERE
cancellation IS NULL GROUP BY runner_id;
```



	runner_id	successful_deliveries
1	1	3
2	2	1
3	3	1

4. How many of each type of burger was delivered?

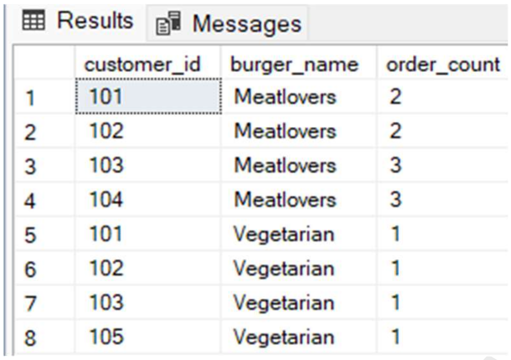
```
SELECT b.burger_name, COUNT(co.order_id) AS delivered_count FROM customer_orders co
JOIN burger_names b ON co.burger_id = b.burger_id JOIN runner_orders ro
ON co.order_id = ro.order_id WHERE ro.cancellation IS NULL GROUP BY b.burger_name;
```



	burger_name	delivered_count
1	Meatlovers	6
2	Vegetarian	2

5. How many Vegetarian and Meatlovers were ordered by each customer?

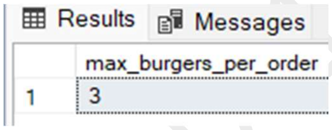
```
SELECT co.customer_id, b.burger_name, COUNT(co.order_id) AS order_count
FROM customer_orders co JOIN burger_names b ON co.burger_id = b.burger_id
GROUP BY co.customer_id, b.burger_name;
```



	customer_id	burger_name	order_count
1	101	Meatlovers	2
2	102	Meatlovers	2
3	103	Meatlovers	3
4	104	Meatlovers	3
5	101	Vegetarian	1
6	102	Vegetarian	1
7	103	Vegetarian	1
8	105	Vegetarian	1

6. What was the maximum number of burgers delivered in a single order?

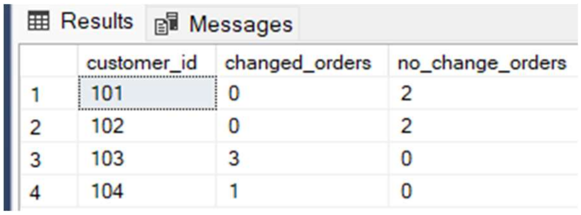
```
SELECT MAX(order_count) AS max_burgers_per_order FROM (
  SELECT order_id, COUNT(*) AS order_count   FROM customer_orders   GROUP BY order_id
) AS order_counts;
```



	max_burgers_per_order
1	3

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

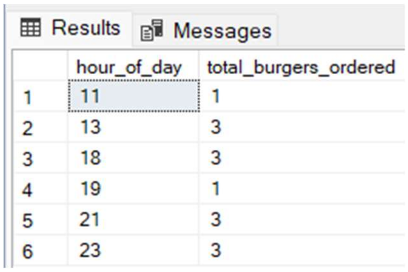
```
SELECT   customer_id,
  SUM(CASE WHEN exclusions IS NOT NULL OR extras IS NOT NULL THEN 1 ELSE 0 END) AS
changed_orders,
  COUNT(*) - SUM(CASE WHEN exclusions IS NOT NULL OR extras IS NOT NULL THEN 1 ELSE
0 END) AS no_change_orders
FROM customer_orders JOIN runner_orders ON customer_orders.order_id = runner_orders.order_id
WHERE runner_orders.cancellation IS NULL GROUP BY customer_id;
```



	customer_id	changed_orders	no_change_orders
1	101	0	2
2	102	0	2
3	103	3	0
4	104	1	0

8. What was the total volume of burgers ordered for each hour of the day?

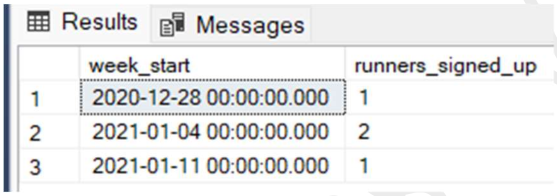
```
SELECT DATEPART(HOUR, order_time) AS hour_of_day,  COUNT(*) AS total_burgers_ordered
FROM   customer_orders GROUP BY DATEPART(HOUR, order_time);
```



	hour_of_day	total_burgers_ordered
1	11	1
2	13	3
3	18	3
4	19	1
5	21	3
6	23	3

9. How many runners signed up for each 1 week period?

```
SELECT   DATEADD(WEEK, DATEDIFF(WEEK, 0, registration_date), 0) AS week_start,
  COUNT(*) AS runners_signed_up FROM   burger_runner GROUP BY
  DATEADD(WEEK, DATEDIFF(WEEK, 0, registration_date), 0);
```



	week_start	runners_signed_up
1	2020-12-28 00:00:00.000	1
2	2021-01-04 00:00:00.000	2
3	2021-01-11 00:00:00.000	1

10. What was the average distance travelled for each customer?

```
SELECT customer_id, AVG(CAST(REPLACE(distance, 'km', '') AS FLOAT)) AS avg_distance_km
FROM runner_orders JOIN customer_orders ON runner_orders.order_id = customer_orders.order_id
GROUP BY customer_id;
```