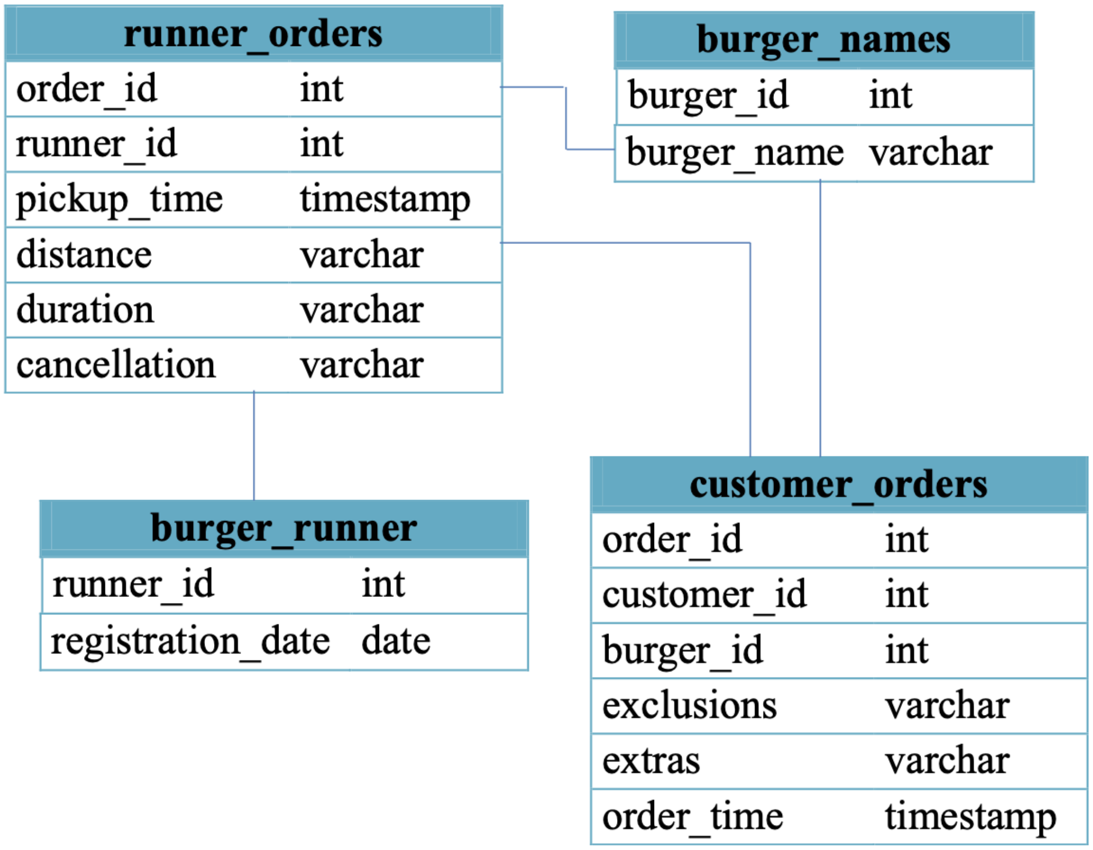
SCHEMA USED



1. How many burgers were ordered?

```
SELECT COUNT(*) AS burgers_ordered
from runner_orders
```

2. How many unique customer orders were made?

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

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

```
select (count(order_id) - count(cancellation)) AS Successful_delivered_order
from runner_orders
```

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

```
select B.burger_name,(count(R.order_id)- count(R.cancellation)) AS Number_of_burger
FROM burger_names AS B INNER JOIN customer_orders AS C
ON C.burger_id =B.burger_id
INNER JOIN runner_orders AS R ON
R.order_id=C.order_id
GROUP BY burger_name
```

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

```
select B.burger_name,(count(R.order_id)- count(R.cancellation)) AS Number_of_burger
FROM burger_names AS B INNER JOIN customer_orders AS C
ON C.burger_id =B.burger_id
INNER JOIN runner_orders AS R ON
R.order_id=C.order_id
GROUP BY burger_name
```

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

```
SELECT C.order_id,count(C.order_id)
FROM customer_orders AS C INNER JOIN runner_orders AS R
ON C.order_id=R.order_id
INNER JOIN burger_names AS B ON
C.burger_id=B.burger_id
GROUP BY C.order_id,R.cancellation
Having R.cancellation IS NULL
```

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

```
SELECT C.customer_id,
sum(case when C.exclusions <>'' or C.extras <>'' then 1 else 0 end) AS changes
FROM customer_orders AS C INNER JOIN runner_orders AS R
using (order_id)
```

```
where R.distance !='0'
GROUP BY C.customer_id
order by C.customer_id
```

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

```
SELECT extract(HOUR from order_time) AS time,count(order_id)
from customer_orders
group by extract(HOUR from order_time)
```

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

```
SELECT extract(WEEK from registration_date) as reg_week,count(runner_id) as
runner_singup
FROM burger_runner
group by extract(WEEK from registration_date)
```

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

```
SELECT ROUND(AVG(REPLACE(R.distance, 'km', '')::DOUBLE
PRECISION)::NUMERIC, 2) as Average_Distance, C.customer_id
FROM customer_orders AS C JOIN runner_orders AS R USING (order_id)
WHERE R.duration != '0'
GROUP BY C.customer_id
```