Dataset: -

```
5 •  select * from customer_orders;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	customer_id	pizza_id	exclusions	extras	order_time
1	101	1			2020-01-01 18:05:02
2	101	1			2020-01-01 19:00:52
3	102	1			2020-01-02 23:51:23
3	102	2		NULL	2020-01-02 23:51:23
4	103	1	4		2020-01-04 13:23:46
4	103	1	4		2020-01-04 13:23:46
4	103	2	4		2020-01-04 13:23:46
5	104	1	null	1	2020-01-08 21:00:29
6	101	2	null	null	2020-01-08 21:03:13
7	105	2	null	1	2020-01-08 21:20:29
8	102	1	null	null	2020-01-09 23:54:33
9	103	1	4	1, 5	2020-01-10 11:22:59
10	104	1	null	null	2020-01-11 18:34:49
10	104	1	2, 6	1, 4	2020-01-11 18:34:49

Table 1: - Customer Orders

```
5 •  select * from pizza_toppings;
```

Result Grid | Filter Rows:

topping_id	topping_name
1	Bacon
2	BBQ Sauce
3	Beef
4	Cheese
5	Chicken
6	Mushrooms
7	Onions
8	Pepperoni
9	Peppers
10	Salami
11	Tomatoes
12	Tomato Sauce

Table 2: - Pizza Toppings

```
5 •  select * from pizza_recipes;
```

Result Grid | Filter Rows:

pizza_id	toppings
1	1, 2, 3, 4, 5, 6, 8, 10
2	4, 6, 7, 9, 11, 12

Table 3: - Pizza Recipes

```
5 •  select * from runner_orders;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	runner_id	pickup_time	distance	duration	cancellation
1	1	2020-01-01 18:15:34	20km	32 minutes	
2	1	2020-01-01 19:10:54	20km	27 minutes	
3	1	2020-01-03 00:12:37	13.4km	20 mins	NULL
4	2	2020-01-04 13:53:03	23.4	40	NULL
5	3	2020-01-08 21:10:57	10	15	NULL
6	3	null	null	null	Restaurant Cancellation
7	2	2020-01-08 21:30:45	25km	25mins	null
8	2	2020-01-10 00:15:02	23.4 km	15 minute	null
9	2	null	null	null	Customer Cancellation
10	1	2020-01-11 18:50:20	10km	10minutes	null

Table 4: - Runner orders

```
4 •  select * from runners;
```

Result Grid | Filter Rows:

runner_id	registration_date
1	2021-01-01
2	2021-01-03
3	2021-01-08
4	2021-01-15

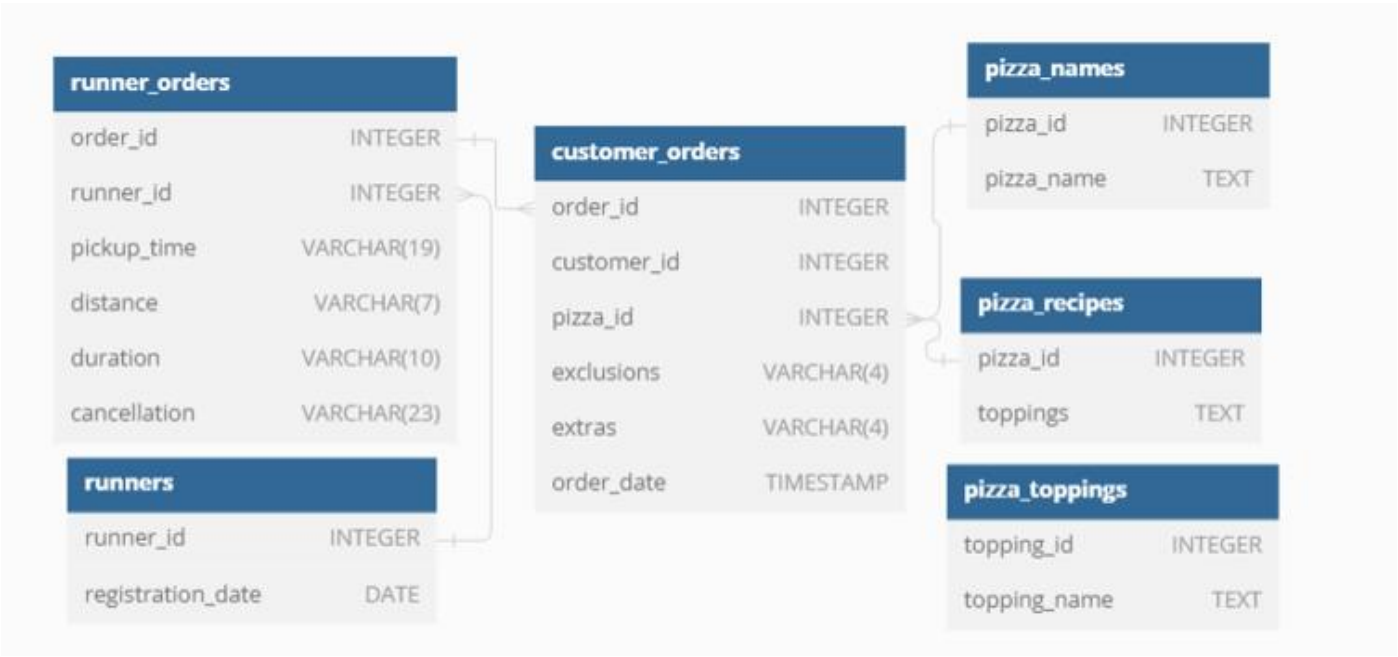
Table 5: - Runners

```
5 •  select * from pizza_names;
```

Result Grid | Filter Rows:

pizza_id	pizza_name
1	Meatlovers
2	Vegetarian

Table 6: - Pizza Names



Entity Relationship Diagram

Data Cleaning and Transformation: -

In CUSTOMER_TABLE columns exclusions and extras are inconsistent. In order to make it consistent replacing blank and text null with null in a temp table so that we didn't lose any data.

```
CREATE TABLE customer_orders_temp AS
  SELECT
    order_id,
    customer_id,
    pizza_id,
    CASE
      WHEN exclusions = '' OR exclusions = 'null' THEN null
      ELSE exclusions
    END AS exclusions,
    CASE
      WHEN extras = '' OR extras = 'null' THEN null
      ELSE extras
    END AS extras,
    order_time
FROM customer_orders;
```

Table: 7. CUSTOMER_ORDER_TEMP: -

order_id	customer_id	pizza_id	exclusions	extras	order_time
1	101	1	NULL	NULL	2020-01-01 18:05:02
2	101	1	NULL	NULL	2020-01-01 19:00:52
3	102	1	NULL	NULL	2020-01-02 23:51:23
3	102	2	NULL	NULL	2020-01-02 23:51:23
4	103	1	4	NULL	2020-01-04 13:23:46
4	103	1	4	NULL	2020-01-04 13:23:46
4	103	2	4	NULL	2020-01-04 13:23:46
5	104	1	NULL	1	2020-01-08 21:00:29
6	101	2	NULL	NULL	2020-01-08 21:03:13
7	105	2	NULL	1	2020-01-08 21:20:29
8	102	1	NULL	NULL	2020-01-09 23:54:33
9	103	1	4	1, 5	2020-01-10 11:22:59
10	104	1	NULL	NULL	2020-01-11 18:34:49
10	104	1	2, 6	1, 4	2020-01-11 18:34:49

Now for RUNNER_OREDER table done the similar things as above additionally removes prefix in duration and distance columns and finally need to update as you can see in below screenshot the PICKUP TIME, DISTANCE and DURATION to DATETIME,INT and INT respectively.

```
desc runner_orders_temp;
```

Result Grid | Filter Rows: | Export: | Wrap

Field	Type	Null	Key	Default	Extra
order_id	int	YES		NULL	
runner_id	int	YES		NULL	
pickup_time	varchar(19)	YES		NULL	
distance	varchar(7)	YES		NULL	
duration	varchar(10)	YES		NULL	
cancellation	varchar(23)	YES		NULL	

Script 1:

```
create table runner_orders_temp as (
select
    order_id,
    runner_id,
    case when pickup_time in ('','null') then null
         else pickup_time end as pickup_time,
    case when distance in ('','null') then null
         when distance like '%km' then trim('km' from distance)
         else distance end as distance,
    case when duration in ('','null') then null
         when duration like '%mins' then trim('mins' from duration)
         when duration like '%minute' then trim('minute' from duration)
         when duration like '%minutes' then trim('minutes' from duration)
         else duration end as duration,
    case when cancellation in ('','null') then null
                else cancellation end as cancellation
from runner_orders);
```

Script 2: Modifying DATATYPE

```
ALTER TABLE runner_orders_temp
modify COLUMN pickup_time datetime,
modify COLUMN distance decimal(0,1),
modify COLUMN duration integer;
```

Table: 8. Runner Order Temp

```
142 •   select * from runner_orders_temp;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	runner_id	pickup_time	distance	duration	cancellation
1	1	2020-01-01 18:15:34	20.0	32	NULL
2	1	2020-01-01 19:10:54	20.0	27	NULL
3	1	2020-01-03 00:12:37	13.4	20	NULL
4	2	2020-01-04 13:53:03	23.4	40	NULL
5	3	2020-01-08 21:10:57	10.0	15	NULL
6	3	NULL	NULL	NULL	Restaurant Cancellation
7	2	2020-01-08 21:30:45	25.0	25	NULL
8	2	2020-01-10 00:15:02	23.4	15	NULL
9	2	NULL	NULL	NULL	Customer Cancellation
10	1	2020-01-11 18:50:20	10.0	10	NULL

Data Analysis: -

Creating a view

As required to join CUSTOMER ORDER and RUNNER ORDER tables frequently. So for ease have created a view.

```
-- Creating a view
CREATE VIEW delivered_orders AS
SELECT * FROM customer_orders_temp
JOIN runner_orders_temp
USING (order_id)
WHERE distance IS NOT NULL;
```

Table 9: Delivered Orders

```
select * from delivered_orders;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	customer_id	pizza_id	exclusions	extras	order_time	runner_id	pickup_time	distance	duration	cancellation
1	101	1	NULL	NULL	2020-01-01 18:05:02	1	2020-01-01 18:15:34	20.0	32	NULL
2	101	1	NULL	NULL	2020-01-01 19:00:52	1	2020-01-01 19:10:54	20.0	27	NULL
3	102	1	NULL	NULL	2020-01-02 23:51:23	1	2020-01-03 00:12:37	13.4	20	NULL
3	102	2	NULL	NULL	2020-01-02 23:51:23	1	2020-01-03 00:12:37	13.4	20	NULL
4	103	1	4	NULL	2020-01-04 13:23:46	2	2020-01-04 13:53:03	23.4	40	NULL
4	103	1	4	NULL	2020-01-04 13:23:46	2	2020-01-04 13:53:03	23.4	40	NULL
4	103	2	4	NULL	2020-01-04 13:23:46	2	2020-01-04 13:53:03	23.4	40	NULL
5	104	1	NULL	1	2020-01-08 21:00:29	3	2020-01-08 21:10:57	10.0	15	NULL
7	105	2	NULL	1	2020-01-08 21:20:29	2	2020-01-08 21:30:45	25.0	25	NULL
8	102	1	NULL	NULL	2020-01-09 23:54:33	2	2020-01-10 00:15:02	23.4	15	NULL
10	104	1	NULL	NULL	2020-01-11 18:34:49	1	2020-01-11 18:50:20	10.0	10	NULL
10	104	1	2, 6	1, 4	2020-01-11 18:34:49	1	2020-01-11 18:50:20	10.0	10	NULL

B) Pizza Metrics

Q1. How many pizzas were ordered?

```
SELECT COUNT(pizza_id) AS ordered_pizza
FROM customer_orders_temp;
```

Result Grid | Filter Rows: | Export:

ordered_pizza
14

Q2. How many unique customer orders were made?

```
SELECT COUNT(DISTINCT(order_id)) AS unique_customer_orders
FROM customer_orders_temp;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

unique_customer_orders
10

Q3. How many successful orders were delivered by each runner?

```sql
SELECT runner_id, COUNT(runner_id) AS order_count
FROM runner_orders_temp
WHERE DISTANCE IS NOT NULL
GROUP BY runner_id
ORDER BY order_count DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

runner_id	order_count
1	4
2	3
3	1

Q4. How many of each type of pizza was delivered?

```sql
SELECT p.pizza_name pizza_name, COUNT(c.pizza_id) pizza_delivered
FROM runner_orders_temp as r
JOIN customer_orders_temp as c  on r.order_id = c.order_id
JOIN pizza_names as p on c.pizza_id = p.pizza_id
WHERE r.distance IS NOT NULL
GROUP BY p.pizza_name;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

pizza_name	pizza_delivered
Meatlovers	9
Vegetarian	3

Q5. How many VEGETARIAN and MEATLOVERS were ordered by each customer?

```sql
with cte as(
    SELECT c.customer_id customer_id, p.pizza_name pizza_name, COUNT(c.customer_id) AS pizza_count
    FROM customer_orders_temp c
    JOIN pizza_names p on c.pizza_id = p.pizza_id
    GROUP BY c.customer_id, p.pizza_name
    ORDER BY c.customer_id)
SELECT customer_id,
    sum(case when pizza_name = 'Meatlovers' then pizza_count else 0 end) as Meatlovers,
    sum(case when pizza_name = 'Vegetarian' then pizza_count else 0 end) as Vegetarian
    from cte
    group by customer_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	Meatlovers	Vegetarian
101	2	1
102	2	1
103	3	1
104	3	0
105	0	1

Q6. What was the maximum number of pizzas delivered in a single order?

```
65  With ranking AS (
66    SELECT order_id, COUNT(order_id) AS pizza_count,
67    RANK() OVER(ORDER BY COUNT(order_id) DESC) rn
68    FROM customer_orders_temp
69    JOIN runner_orders_temp
70    USING (order_id)
71    WHERE DISTANCE IS NOT NULL
72    GROUP BY order_id)
73  SELECT order_id, pizza_count FROM ranking
74  WHERE ranking.rn = 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	pizza_count
4	3

Q7. For each customer, how many delivered pizzas had at least 1 change, and how many had no changes?

```
97  SELECT customer_id,
98    COUNT(
99      CASE WHEN exclusions <> '' OR extras <> '' THEN 1
100     END
101   ) AS changed,
102   COUNT(
103     CASE WHEN exclusions = '' AND extras = '' THEN 1
104     END
105   ) AS unchanged
106 FROM delivered_orders
107 GROUP BY customer_id
108 ORDER BY customer_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	changed	unchanged
101	0	0
102	0	0
103	3	0
104	2	0
105	1	0

Insight: Only 40% of the customers (i.e. Customer 101 and 102) took their pizzas with the standard set of toppings. The others were open to trying out other toppings.

Q8. How many pizzas were delivered that had both exclusions and extras?

```
113  SELECT COUNT(*) AS pizza_having_exclusions_n_extras
114  FROM delivered_orders
115  WHERE exclusions is not null
116  AND extras is not null;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

pizza_having_exclusions_n_extras
1

Insight: Only 1 pizza without any changes

Q9. What was the total volume of pizzas ordered for each hour of the day?

```
122 •  SELECT EXTRACT(HOUR FROM order_time) AS hour_of_day,
123        count(pizza_id) AS pizza_count
124    FROM customer_orders_temp
125    GROUP BY hour_of_day
126    ORDER BY hour_of_day;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content

hour_of_day	pizza_count
11	1
13	3
18	3
19	1
21	3
23	3

Insight: From the above we can infer that the 13th (1pm), 18th (6pm), 21th (9pm) and 23rd (11pm) hours are the busiest of the day 11th (11am) and 19th (7pm) hours are the least busy

Q10. What was the volume of orders for each day of the week?

```
133 •  SELECT dayname(order_time) AS day_of_week,
134        COUNT(pizza_id) AS pizza_count
135    FROM customer_orders_temp
136    GROUP BY day_of_week
137    ORDER BY day_of_week;
```

Result Grid | Filter Rows: | Export: | Wra

day_of_week	pizza_count
Friday	1
Saturday	5
Thursday	3
Wednesday	5

Insight: With 5 orders each, Saturdays and Wednesdays are the busiest days of the week while Friday is the least busy.

B) Runner and Customer Experience

Q1. How many runners signed up for each 1 week period? (I.e. week starts 2021-01-01)

```
156 •  SELECT EXTRACT(WEEK FROM registration_date + 3) AS week_of_year,
157        COUNT(runner_id)
158    FROM runners
159    GROUP BY week_of_year
160    ORDER BY week_of_year;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

week_of_year	COUNT(runner_id)
1	2
2	1
3	1

Insight: Week 1 has the most runners (2) signed up.

Q2. What was the average time in minutes it took for each runner to arrive at the Pizza Runner HQ to pick up the order?

```sql
WITH order_time_diff AS (
    SELECT DISTINCT order_id, TIMESTAMPDIFF(SECOND, order_time, pickup_time) AS time_diff
    FROM delivered_orders
)
SELECT SEC_TO_TIME(ROUND(AVG(time_diff) + 30)) AS rounded_avg_time_diff
FROM order_time_diff;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

rounded_avg_time_diff
00:16:29

Insight: It takes each runner 16 minutes on the average to pick up the order.

Q3. Is there any relationship between the number of pizzas and how long the order takes to prepare?

```sql
WITH orders_group AS (
    SELECT order_id, COUNT(order_id) AS pizza_count,
           TIMESTAMPDIFF(minute, order_time, pickup_time) AS time_diff
    FROM delivered_orders
    GROUP BY order_id, pickup_time, order_time
    ORDER BY order_id
)
SELECT pizza_count, AVG(time_diff) AS avg_time_diff_minute
FROM orders_group
GROUP BY pizza_count;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

pizza_count	avg_time_diff_minute
3	29.0000
2	18.0000
1	12.0000

Insights: From the above, the more the pizzas contained in an order, the longer it takes for that order to be ready.

Q4. What was the average distance traveled for each customer?

```sql
SELECT customer_id,
    ROUND(AVG(distance), 2) as avg_distance
FROM delivered_orders
GROUP BY customer_id
ORDER BY avg_distance DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell C

customer_id	avg_distance
105	25.00
103	23.40
101	20.00
102	16.73
104	10.00

Insights: Customer 105 stays farthest (25km) while Customer 104 stays closest (10km).

Q5. What was the difference between the longest and shortest delivery times for all orders?

```
208 •   SELECT MAX(duration)- MIN(duration) AS delivery_time_diff
209     FROM runner_orders_temp;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

delivery_time_diff
30

Q6. What was the average speed for each runner for each delivery?

```
218 •   SELECT DISTINCT order_id, runner_id,
219       round(distance / (duration/60), 2) AS average_speed_km_per_hr
220     FROM delivered_orders
221     ORDER BY runner_id, average_speed_km_per_hr;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	runner_id	average_speed_km_per_hr
1	1	37.50
3	1	40.20
2	1	44.44
10	1	60.00
4	2	35.10
7	2	60.00
8	2	93.60
5	3	40.00

Insights: 0f concern is Runner 2's speed. There is a large variance between the lowest (35.1km/hr.) and highest speeds (93.6km/hr.). This should be investigated.

Q7. What is the successful delivery percentage for each runner?

```
227 •   SELECT runner_id,
228       round(count(distance)/ count(runner_id) * 100) AS delivery_percentage
229     FROM runner_orders_temp
230     GROUP BY runner_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

runner_id	delivery_percentage
1	100
2	75
3	50

Insights: Runner 1 has highest percentage of successful deliveries (100%) while Runner 3 has the least (50%). But it's important to note that it's beyond the control of the runner as either the customer or the restaurant can cancel orders.

Conclusions: -

- Modify the standard pizza as only 40% pizza delivered without any changes.
- Saturdays and Wednesday are the busy day so can hire more runner on those days.
- There is large variance between the lowest and highest speeds of Runner 2's speed. That requires further investigation.