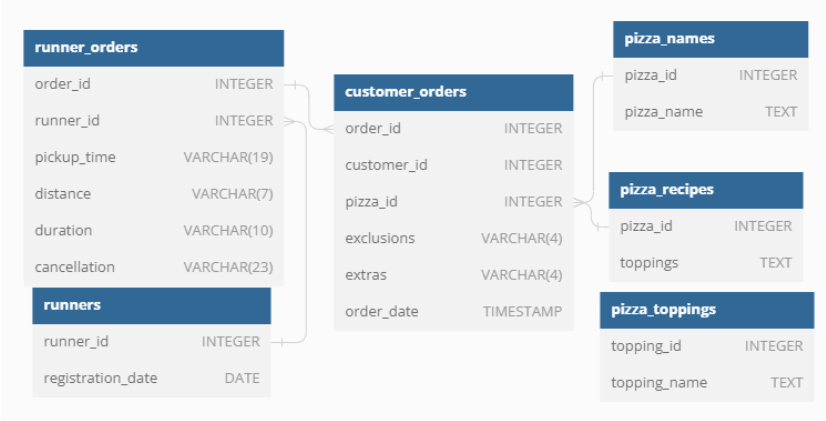
[SQL] Customer Order and Delivery Service Operations

작성자 : 김유민

Introduction

Pizza Runner, a uberized company making & delivering fresh pizza, requires assistance to clean data and apply some calculation for metrics to better direct his runners and optimise its operations.

Entity Relationship Diagram



Datasets used

Table 1: `runners`

The `runners` table shows the `registration_date` for each new runner.

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

Table 2: `customer_orders`

Customer `pizza orders` are captured in the `customer_orders` table with 1 row for each individual pizza that is part of the order.

The `pizza_id` relates to the type of pizza which was ordered whilst the `exclusions` are the `ingredient_id` values which should be removed from the pizza and the `extras` are the `ingredient_id` values which need to be added to the pizza.

Note that customers can order multiple pizzas in a single order with varying exclusions and extras values even if the pizza is the same type!

order_id	customer_id	pizza_id	exclusions	extras	order_time
1	101	1			2021-01-01 18:05:02
2	101	1			2021-01-01 19:00:52
3	102	1			2021-01-02 23:51:23

order_id	customer_id	pizza_id	exclusions	extras	order_time
3	102	2		NaN	2021-01-02 23:51:23
4	103	1	4		2020-01-04 13:23:46
4	103	1	4		2021-01-04 13:23:46
4	103	2	4		2021-01-04 13:23:46
5	104	1	null	1	2021-01-08 21:00:29
6	101	2	null	null	2021-01-08 21:03:13
7	105	2	null	1	2021-01-08 21:20:29
8	102	1	null	null	2021-01-09 23:54:33
9	103	1	4	1, 5	2021-01-10 11:22:59
10	104	1	null	null	2021-01-11 18:34:49
10	104	1	2, 6	1, 4	2021-01-11 18:34:49

Table 3: `runner_orders`

After each orders are received through the system - they are assigned to a runner - however not all orders are fully completed and can be cancelled by the restaurant or the customer. The `pickup_time` is the timestamp at which the runner arrives at the Pizza Runner headquarters to pick up the freshly cooked pizzas. The `distance` and `duration` fields are related to how far and long the runner had to travel to deliver the order to the respective customer.

order_id	runner_id	pickup_time	distance	duration	cancellation
1	1	2021-01-01 18:15:34	20km	32 minutes	
2	1	2021-01-01 19:10:54	20km	27 minutes	
3	1	2021-01-03 00:12:37	13.4km	20 mins	NaN
4	2	2021-01-04 13:53:03	23.4	40	NaN
5	3	2021-01-08 21:10:57	10	15	NaN
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: `pizza_names`

At the moment - Pizza Runner only has 2 pizzas available the Meat Lovers or Vegetarian!

pizza_id	pizza_name
1	Meat Lovers
2	Vegetarian

Data Cleansing

`customer_orders` table

- In the exclusions and extras columns, there are blank spaces and null values. So, we should unify the values to 'NULL' .
- (Qurey)

```
DROP TABLE IF EXISTS customer_orders_temp;

CREATE TEMPORARY TABLE customer_orders_temp AS
SELECT order_id,
       customer_id,
       pizza_id,
```

```
        CASE
            WHEN exclusions = '' THEN NULL
            WHEN exclusions = 'null' THEN NULL
            ELSE exclusions
        END AS exclusions,
        CASE
            WHEN extras = '' THEN NULL
            WHEN extras = 'null' THEN NULL
            ELSE extras
        END AS extras,
        order_time
FROM pizza_runner.customer_orders;

SELECT * FROM customer_orders_temp;
```

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			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		1	2020-01-08 21:00:29
6	101	2			2020-01-08 21:03:13
7	105	2		1	2020-01-08 21:20:29
8	102	1			2020-01-09 23:54:33
9	103	1	4	1, 5	2020-01-10 11:22:59
10	104	1			2020-01-11 18:34:49
10	104	1	2, 6	1, 4	2020-01-11 18:34:49

`runner_orders` table

- The pickup_time, distance, duration and cancellation columns in runner_orders table will need to be cleaned up before using them in the queries
- In the pickup_time column, there are null values.
- In the distance column, there are null values. It contains unit - km. The 'km' must also be stripped
- In the duration column, there are null values. The 'minutes', 'mins' 'minute' must be stripped
- In the cancellation column, there are blank spaces and null values.
- (Query)

```
DROP TABLE IF EXISTS runner_orders_temp;

CREATE TEMPORARY TABLE runner_orders_temp AS

SELECT order_id,
       runner_id,
       CASE
           WHEN pickup_time LIKE 'null' THEN NULL
           ELSE pickup_time
       END AS pickup_time,
       CASE
           WHEN distance LIKE 'null' THEN NULL
           ELSE CAST(regexp_replace(distance, '[a-z]+', '') AS FLOAT)
       END AS distance,
       CASE
           WHEN duration LIKE 'null' THEN NULL
           ELSE CAST(regexp_replace(duration, '[a-z]+', '') AS FLOAT)
       END AS duration,
       CASE
           WHEN cancellation LIKE '' THEN NULL
           WHEN cancellation LIKE 'null' THEN NULL
           ELSE cancellation
       END AS cancellation
FROM pizza_runner.runner_orders;

SELECT * FROM runner_orders_temp;
```

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

Data Analyzing (1) - Order Metrics

1. How many pizzas were ordered?

- Query :

```
SELECT count(pizza_id) AS "total_order"
FROM pizza_runner.customer_orders_temp;
```

- Results : 14 orders were made right now.

total_order
14

2. How many unique customer orders were made?

- Query :

```
SELECT
  COUNT(DISTINCT order_id) AS 'n_order'
FROM customer_orders_temp;
```

- Results : 10 unique orders were made until now.

n_order
10

3. How many times has each customer orderd/reorderd pizza?

- Query :

```
-- distict time
SELECT customer_id, GROUP_CONCAT(distinct order_time)as order_date,
       count(distinct order_time) as n_order
FROM pizza_runner.customer_orders_temp
GROUP BY customer_id
ORDER BY customer_id;
```

```
-- distinct day
SELECT customer_id, GROUP_CONCAT(distinct date_format(order_time, '%Y-%m-%d')) as order_date,
       count(distinct(date_format(order_time, '%Y-%m-%d'))) as n_order
FROM pizza_runner.customer_orders_temp
GROUP BY customer_id
ORDER BY customer_id;
```

- Results : Almost every customer (except customer_id = ‘105’) reorder pizza.

customer_id	order_date	n_order
101	2020-01-01 18:05:02, 2020-01-01 19:00:52, 2020-01-08 21:03:13	3
102	2020-01-02 23:51:23, 2020-01-09 23:54:33	2
103	2020-01-04 13:23:46, 2020-01-10 11:22:59	2
104	2020-01-08 21:00:29, 2020-01-11 18:34:49	2
105	2020-01-08 21:20:29	1

customer_id	order_date	n_order
101	2020-01-01,2020-01-08	2
102	2020-01-02,2020-01-09	2
103	2020-01-04,2020-01-10	2
104	2020-01-08,2020-01-11	2
105	2020-01-08	1

4. How long does it takes to re-order pizza by each customer ?

- Query :

```
WITH Table1 AS (
SELECT customer_id, max(order_time) as last_order, min(order_time) as first_order, count(distinct order_time) as n_order
FROM pizza_runner.customer_orders_temp
GROUP BY customer_id
ORDER BY customer_id)

SELECT customer_id, CONCAT(ROUND(DATEDIFF( last_order,first_order)/(n_order-1),2),'days') as cycle
FROM Table1
WHERE n_order <> 1;
```

- Results : 4.875days (average for all customers) takes to reorder.

customer_id	cycle
101	3.50days
102	7.00days
103	6.00days
104	3.00days

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

- Query :

```
SELECT p.pizza_name,
  count(c.*) AS n_pizza_type
FROM pizza_runner.customer_orders_temp AS c
  LEFT JOIN pizza_runner.pizza_names AS p ON p.pizza_id = c.pizza_id
  LEFT JOIN pizza_runner.runner_orders_temp AS r ON c.order_id = r.order_id
WHERE cancellation IS NULL
GROUP BY p.pizza_name
ORDER BY n_pizza_type DESC;
```

- Results : meatlovers are orders 9 times, and vegetarian orders 3 times.

pizza_name	n_pizza_type
Meatlovers	9
Vegetarian	3

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

- Query :

```
SELECT customer_id,
       SUM(CASE
               WHEN pizza_id = 1 THEN 1
               ELSE 0
           END) AS 'meat_lovers',
       SUM(CASE
               WHEN pizza_id = 2 THEN 1
               ELSE 0
           END) AS 'vegetarian'
FROM pizza_runner.customer_orders_temp
GROUP BY customer_id
ORDER BY customer_id;
```

- Results : there are someone who ordered both, and who ordered only meat, who ordered only vegetarian.

customer_id	meat_lovers	vegetarian
101	2	1
102	2	1
103	3	1
104	3	0
105	0	1

7. Which pizza was the most popular for each customer?

- Query : Except 1 person, most loved ‘Meatlovers’ pizza.

```
WITH order_table AS (
SELECT customer_id, p.pizza_name, count(*) as ordertime,
       dense_rank() over(PARTITION BY customer_id ORDER BY count(*) desc) as ranking
FROM pizza_runner.customer_orders_temp AS c
  LEFT JOIN pizza_runner.pizza_names AS p ON p.pizza_id = c.pizza_id
GROUP BY customer_id, p.pizza_name)
```

```
SELECT customer_id, pizza_name, ordertime
FROM order_table
WHERE ranking = 1;
```

- Results :

customer_id	pizza_name	ordertime
101	Meatlovers	2
102	Meatlovers	2
103	Meatlovers	3
104	Meatlovers	3
105	Vegetarian	1

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

- Query :

```
SELECT customer_id,
       order_id,
       count(order_id) AS pizza_count
FROM pizza_runner.customer_orders_temp
GROUP BY order_id
ORDER BY pizza_count DESC
LIMIT 1;
```

- Results : The person, who ordered the most , ordered 3 pizza at one time.

customer_id	order_id	pizza_count
103	4	3

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

- Query :

```
SELECT hour(order_time) AS 'Hour',
       count(order_id) AS 'n_order',
       round(100*count(order_id) /sum(count(order_id)) over(), 2) AS 'ratio'
FROM pizza_runner.customer_orders_temp
GROUP BY 1
ORDER BY 1;
```

- Results : We can find the hourly pattern when consumer usually order pizza.

Hour	n_order	ratio
11	1	7.14
13	3	21.43
18	3	21.43
19	1	7.14
21	3	21.43
23	3	21.43

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

- The DAYOFWEEK() function returns the weekday index for a given date (1=Sunday, 2=Monday, 3=Tuesday, 4=Wednesday, 5=Thursday, 6=Friday, 7=Saturday)
- DAYNAME() returns the name of the week day
- Query :

```
SELECT dayname(order_time) AS 'Day Of Week',
       count(order_id) AS 'n_order',
       round(100*count(order_id) /sum(count(order_id)) over(), 2) AS 'ratio'
FROM pizza_runner.customer_orders_temp
GROUP BY 1
ORDER BY 2 DESC;
```

- Results : Also can find the weekly pattern when consumer usually order pizza.

Day of Week	n_order	ratio
Wednesday	5	35.71
Saturday	5	21.4335.71
Thursday	3	21.43
Friday	1	7.14

Data Analyzing (2) - Runner and Customer Experience

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

1. Create CTE and select
   - runner_id
   - Actual registration date
   - Starting week. Calculate the starting week by
     - subtract the registration_date from '2021-01-01'
     - get the remainder of subtraction above from dividing by 7 (days in week)
     - subtract registration_date from remainder to derive start week.
2. Select runner_id and starting_week from CTE
3. Group and Order by starting week.

- Query :

```
WITH runner_signups as (SELECT runner_id,
    registration_date,
    date_sub(registration_date, INTERVAL datediff(registration_date,'2021-01-01') %7 DAY)
            AS starting_week
FROM pizza_runner.runners )

SELECT starting_week,
  count(runner_id) AS n_runners
from runner_signups
GROUP BY starting_week
ORDER BY starting_week;
```

- Results :We can know the pattern when the runner first signed up the delivery service.

starting_week	n_runners
2021-01-01	2
2021-01-08	1
2021-01-15	1

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

- Query :

```
SELECT runner_id,
       round(avg(TIMESTAMPDIFF(MINUTE, order_time, pickup_time)), 2) avg_arrival_time
FROM runner_orders_temp
INNER JOIN customer_orders_temp USING (order_id)
WHERE cancellation IS NULL
GROUP BY runner_id;
```

- Results : It takes average 10 ~ 25 minutes for runners to arrive at pickup.

runner_id	avg_arrival_time
1	15.33
2	23.40
3	10.00

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

- Query :

```
WITH order_count_cte AS
  (SELECT order_id,
          COUNT(order_id) AS pizzas_order_count,
          TIMESTAMPDIFF(MINUTE, order_time, pickup_time) AS prep_time
   FROM runner_orders_temp
   INNER JOIN customer_orders_temp USING (order_id)
   WHERE cancellation IS NULL
   GROUP BY order_id)
SELECT pizzas_order_count,
       round(avg(prep_time), 2) AS prep_time
FROM order_count_cte
GROUP BY pizzas_order_count;
```

- Results : There is a positive correlation between the number of orders and prep time. The more pizzas are orders, the more it takes to be prepared.

pizzas_order_count	prep_time
1	12.00
2	18.00
3	29.00

4a. What was the average distance travelled for each customer?

- Query :

```
SELECT customer_id,
       round(avg(distance), 2) AS 'avg_distance'
FROM runner_orders_temp
INNER JOIN customer_orders_temp USING (order_id)
WHERE cancellation IS NULL
GROUP BY customer_id
ORDER BY customer_id;
```

- Results : customers are located in average 10 ~ 25km far away.

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

4b. What was the average distance travelled for each runner?

- Query :

```
SELECT runner_id,
       round(avg(distance), 2) AS 'avg_distance'
FROM runner_orders_temp
WHERE cancellation IS NULL
GROUP BY runner_id
ORDER BY runner_id;
```

- Results : we can monitor the runner's deliver distances.

runner_id	avg_distance
1	15.85
2	23.93
3	10.00

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

- Query :

```
SELECT MIN(duration) AS min_duration,
       MAX(duration) AS max_duration,
       MAX(duration) - MIN(duration) AS time_diff
FROM runner_orders_temp;
```

- Results : longest time was 40 minutes, and shortest was 10 minutes. It seems there is quite big gap.

min_duration	max_duration	time_diff
10	40	30

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

- Query :

```
SELECT runner_id,
       COUNT(pickup_time) AS delivered_orders,
       COUNT(*) AS total_orders,
       ROUND(100 * COUNT(pickup_time) / COUNT(*)) AS delivered_percentage
FROM runner_orders_temp
GROUP BY runner_id
ORDER BY runner_id;
```

- Results : We can monitor the runner's delivery quality.

runner_id	delivered_orders	total_orders	delivered_percentage
1	4	4	100.0
2	3	4	75.0
3	1	2	50.0