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!

The exclusions and extras columns will need to be cleaned up before using them in your queries.

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

3	102	2		NaN	2021-01-02 23:51:23
4	103	1	4		2021-01-04 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
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
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

CASE STUDY QUESTION

A. Pizza Metrics

1. How many pizzas were ordered?
2. How many unique customer orders were made?
3. How many successful orders were delivered by each runner?
4. How many of each type of pizza was delivered?
5. How many Vegetarian and Meatlovers were ordered by each customer?
6. What was the maximum number of pizzas delivered in a single order?
7. For each customer, how many delivered pizzas had at least 1 change and how many had no changes?
8. How many pizzas were delivered that had both exclusions and extras?
9. What was the total volume of pizzas ordered for each hour of the day?
10. What was the volume of orders for each day of the week?

B. Runner and Customer Experience

1. How many runners signed up for each 1 week period? (i.e. week starts 2021-01-01)
2. What was the average time in minutes it took for each runner to arrive at the Pizza Runner HQ to pickup the order?
3. Is there any relationship between the number of pizzas and how long the order takes to prepare?
4. What was the average distance travelled for each customer?
5. What was the difference between the longest and shortest delivery times for all orders?
6. What was the average speed for each runner for each delivery and do you notice any trend for these values?
7. What is the successful delivery percentage for each runner?

ANSWER – Đặng Thành Thái

❖ Clean the data

- ***Covert the string null in customer_orders into null value***

```
SELECT order_id, customer_id, pizza_id,
        CASE
                WHEN exclusions = ‘null’   THEN   null
                ELSE exclusions
        END     AS   exclusions,
        CASE
                WHEN extras = ‘null’   THEN   null
                ELSE extras
        END     AS   extras,
        order_time
INTO   cleaned_co
FROM customer_orders;
```

- ***Change the data type and value of table runner_orders***

```
SELECT order_id, runner_id,
        CAST ( CASE
                        WHEN distance = ‘null’  THEN  null
                        ELSE TRIM (‘km’ from distance)
                END     AS   INT)   AS   distance,
        CAST ( CASE
                        WHEN duration = ‘null’  THEN  null
                        ELSE   SUBSTRING (duration, 1, 2)
```

```
            END    AS  INT)   AS  duration,
        CASE
            WHEN cancellation  IN  (‘null’, ‘ ’)   THEN  null
            ELSE cancellation
        END    AS cancellation
INTO   cleaned_ro
FROM runner_orders;
```

❖ Pizza Metrics

```
Q1:     SELECT   COUNT (pizza_id)  AS  number_of_ordered
        FROM  cleaned_co;
Q2:     SELECT   COUNT (DISTINCT  order_id)  AS unique_order
        FROM  cleaned_co;
Q3:     SELECT runner_id,
                SUM  ( CASE
                        WHEN cancellation  IS  null  THEN  1
                        ELSE  0
                    END)   AS  total_ordered
        FROM cleaned_ro
        GROUP BY runner_id;
Q4:     SELECT c.pizza_id,
                SUM ( CASE
                        WHEN  r.cancellation  IS  null  THEN  1
                        ELSE  0
                    END)   AS  pizza_delivered
        FROM  cleaned_co   c,
            cleaned_ro   r
        WHERE  c.order_id = r.order_id
        GROUP BY c.pizza_id;
Q5:     SELECT c.customer_id, p.pizza_names, COUNT(c.pizza_id)  AS  number_order
        FROM cleaned_co  c,
            pizza_names  p
        WHERE c.pizza_id = p.pizza_id
        GROUP BY c.customer_id, p.pizza_names;
Q6:     SELECT c.order_id, COUNT (r.runner_id)  AS  number_pizza
        FROM  cleaned_co  c,
            cleaned_ro  r
        WHERE c.order_id = r.order_id
            AND  r.cancellation  IS  null
        GROUP BY c.order_id
        ORDER BY number_pizza  DESC  LIMIT  1;
Q7:     SELECT c.customer_id,
             SUM ( CASE
                     WHEN c.exclusion  IS NOT null OR  c.extras  IS NOT null  THEN  1
                     ELSE  0
                 END)  AS  pizza_change
        FROM  cleaned_co  c,
            cleaned_ro  r
        WHERE  c.order_id = r.order_id
             AND  r.cancellation  IS  null
        GROUP BY c.customer_id;
```

Q8:
```
SELECT  COUNT (c.pizza_id)
FROM  cleaned_co  c,
      cleaned_ro  r
WHERE  c.order_id = r.order_id
       AND  c.exclusions  IS NOT null
       AND  c.extras  IS NOT null
       AND  r.cancellation  IS null;
```

Q9:
```
SELECT  DEPART (hour, order_time)  AS hour_day,
        COUNT (order_id)  AS number_order
FROM  cleaned_co
GROUP BY DEPART (hour, order_time) ;
```

Q10:
```
SELECT  FORMAT (order_time, ‘dddd’)  AS week_day,
         COUNT (order_id)  AS week_day_order
FROM cleaned_co
GROUP BY FORMAT (order_time, ‘dddd’);
```

❖ **RUNNER & CUSTOMER**

Q1:
```
SELECT  DEPART (week, registration_date)  AS  week,
         COUNT (runner_id)  AS  number_resign
FROM runners
GROUP BY  DEPART (week, registration_date) ;
```

Q2:
```
SELECT  r.runner_id,
        AVG (DATEDIFF (minute, c.order_time, r.pickup_time)  AS  avg_time
FROM  cleaned_co  c,
      cleaned_ro  r
WHERE  c.order_id = r.order_id
```

```
        GROUP BY  r.runner_id;

Q3:     WITH  relation  AS  (

                SELECT  c.order_id,

                        COUNT (pizza_id)  AS  number_pizza,

                        DATEDIFF (minute, c.order_time, r.pickup_time)  AS  time_order

                        DATEDIFF (minute, c.order_time, r.pickup_time)  AS  time_pizza

                FROM  cleaned_co  c,

                      cleaned_ro  r

                WHERE c.order_id = r.order_id

                GROUP BY  c.order_id, c.order_time, r.pickup_time  )


        SELECT number_pizza,

                AVG (time_order)  AS  avg_time_taken,

                AVG (time_pizza)  AS  avg_time_pizza

        FROM  relation

        GROUP BY  number_pizza;

Q4:     SELECT  c.customer_id,

                AVG (r.distance)  AS avg_distance

        FROM cleaned_co  c,

                cleaned_ro  r

        WHERE  c.order_id = r.order_id

        GROUP BY  c.customer_id

Q5:     SELECT  MAX (duration)  AS  max_time,

                 MIN (duration)  AS  min_time,

                 MAX (duration) - MIN (duration)  AS difference

        FROM cleaned_ro
```

```
Q6:     SELECT  runner_id, distance, duration,
                 ROUND (distance*60/duration, 2)  AS  speed
        FROM cleaned_ro
        WHERE  cancellation  IS  null
        ORDER BY runner_id, speed

Q7:     WITH db_1  AS  (
                SELECT runner_id,
                        COUNT (runner_id)  AS  su_delivered
                FROM cleaned_ro
                WHERE  cancellation  IS  null)


        SELECT  r.runner_id,
                CAST(d.su_delivered AS float)*100/ CAST( COUNT (r.runner_id) AS float) AS percent
        FROM  cleaned_ro  r,
                db_1  d
        WHERE r.runner_id = d.runner_id
        GROUP BY r.runner_id
```