🍕 Case Study #2B: Pizza Runner

B. Runner and Customer Experience

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

Steps:

- Use the `DATEPART(WEEK, registration_date)` function to extract the week number for each runner's registration date.
- Count the number of `runner_id` entries for each week.
- Group the results by the extracted week number.

```
SELECT
  DATEPART(WEEK, registration_date) AS registration_week,
  COUNT(runner_id) AS runner_signup
FROM runners
GROUP BY DATEPART(WEEK, registration_date);
```

Answer:

- Week 1 of January 2021: 2 new runners signed up.
- Week 2 and Week 3 of January 2021: 1 new runner signed up each.

registration_week	runner_signup
1	2
2	1
3	1

---

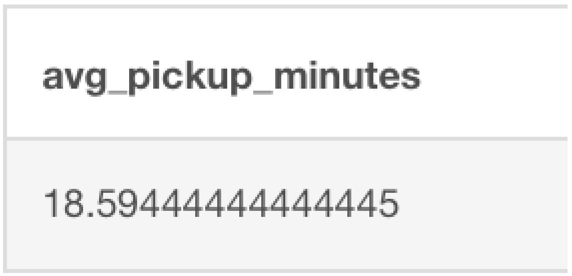
2. What was the average time in minutes it took for each runner to arrive at the Pizza Runner HQ to pick up the order?

Steps:

- Create a CTE named `time_taken_cte` to calculate the time difference (in minutes) between `order_time` and `pickup_time` for each order using the `DATEDIFF` function.
- Filter to exclude cases where the `pickup_minutes` is less than or equal to 1.
- Calculate the average of the `pickup_minutes` column.

```
WITH time_taken_cte AS (
  SELECT
    c.order_id,
    c.order_time,
    r.pickup_time,
    EXTRACT(EPOCH FROM (r.pickup_time - c.order_time)) / 60 A
S pickup_minutes
  FROM customer_orders_temp AS c
  JOIN runner_orders_temp AS r
    ON c.order_id = r.order_id
  WHERE r.distance > 0
)
SELECT
  AVG(pickup_minutes) AS avg_pickup_minutes
FROM time_taken_cte
WHERE pickup_minutes > 1;
```

Answer: The average time taken by runners to arrive at Pizza Runner HQ to pick up the order is 18 minutes.



avg_pickup_minutes
18.594444444444445

3. Is there a relationship between the number of pizzas and the time it takes to prepare the order?

Steps:

- Create a CTE named `prep_time_cte` to calculate the number of pizzas in each order and the preparation time in minutes.
- Calculate the average preparation time for each pizza count.
- Filter out orders where `prep_time_minutes` is less than or equal to 1.

```
WITH prep_time_cte AS (
  SELECT
    c.order_id,
    COUNT(c.pizza_id) AS pizza_order,
    c.order_time,
    r.pickup_time,
    EXTRACT(EPOCH FROM (r.pickup_time - c.order_time)) / 60 A
S prep_time_minutes
  FROM customer_orders_temp AS c
  JOIN runner_orders_temp AS r
    ON c.order_id = r.order_id
  WHERE r.distance > 0
  GROUP BY c.order_id, c.order_time, r.pickup_time
)
SELECT
  pizza_order,
  AVG(prep_time_minutes) AS avg_prep_time_minutes
FROM prep_time_cte
WHERE prep_time_minutes > 1
GROUP BY pizza_order;
```

Answer:

- A single pizza order takes an average of 12 minutes to prepare.
- An order with 2 pizzas takes 18 minutes on average.
- An order with 3 pizzas takes 30 minutes on average.

pizza_order	avg_prep_time_minutes
3	29.283333333333335
2	18.375
1	12.356666666666667

4. What is the average distance covered by each customer?

Steps:

- Join the `customer_orders` and `runner_orders` tables on `order_id`.
- Calculate the average `distance` for each customer using the `AVG` function.
- Filter to exclude orders with zero or null distances.

```
SELECT
  c.customer_id,
  ROUND(AVG(r.distance::NUMERIC), 2) AS avg_distance
FROM customer_orders_temp AS c
JOIN runner_orders_temp AS r
  ON c.order_id = r.order_id
WHERE r.distance > 0
GROUP BY c.customer_id
ORDER BY c.customer_id;
```

Answer:

- Customer 104 has the shortest average distance (10 km).
- Customer 105 has the longest average distance (25 km).

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

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

Steps:

- Filter `runner_orders` to include only rows where `duration` is not null or empty.
- Use the `MAX` and `MIN` functions to calculate the longest and shortest delivery times, respectively.
- Subtract the shortest time from the longest time.

```
SELECT
  MAX(duration::NUMERIC) - MIN(duration::NUMERIC) AS delivery
_time_difference
FROM runner_orders
WHERE duration NOT LIKE ' ';
```

Answer: The difference between the longest (40 minutes) and shortest (10 minutes) delivery times is 30 minutes.

delivery_time_difference
30

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

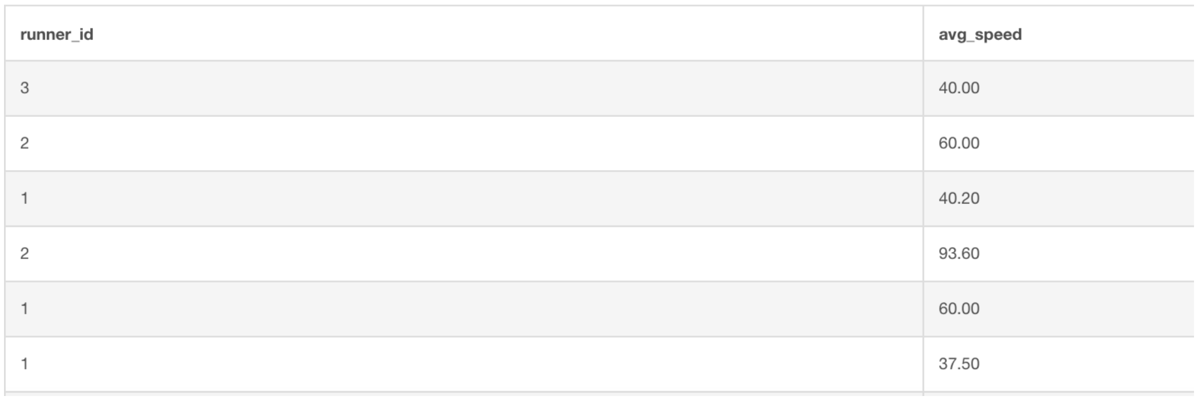
Steps:

- Join the `runner_orders` and `customer_orders` tables on `order_id`.
- Calculate the average speed for each runner as `distance / duration` converted to hours.
- Use the `ROUND` function to round the speed values and group by `runner_id`.

```
SELECT
  r.runner_id,
  ROUND((r.distance::NUMERIC / (r.duration::NUMERIC / 60)),
2) AS avg_speed
FROM runner_orders_temp AS r
JOIN customer_orders_temp AS c
  ON r.order_id = c.order_id
WHERE r.distance::NUMERIC > 0 AND r.duration::NUMERIC > 0
GROUP BY r.runner_id, r.distance, r.duration;
```

Answer:

- Runner 1's speed ranges from 37.5 km/h to 60 km/h.
- Runner 2's speed fluctuates from 35.1 km/h to 93.6 km/h.
- Runner 3 maintains a consistent speed of 40 km/h.



runner_id	avg_speed
3	40.00
2	60.00
1	40.20
2	93.60
1	60.00
1	37.50

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

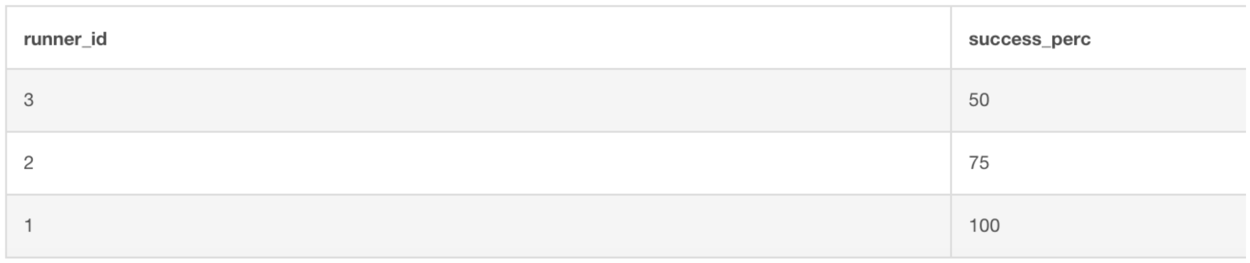
Steps:

- Use a `CASE` statement to count successful deliveries (`distance > 0`) for each runner.
- Divide the successful deliveries by the total number of orders and multiply by 100 to calculate the percentage.
- Round the result using the `ROUND` function.

```
SELECT
  runner_id,
  ROUND(100 * SUM(
    CASE WHEN distance > 0 THEN 1
    ELSE 0 END) / COUNT(*), 0) AS success_perc
FROM runner_orders_temp
GROUP BY runner_id;
```

Answer:

- Runner 1: 100% successful deliveries.
- Runner 2: 75% successful deliveries.
- Runner 3: 50% successful deliveries.



runner_id	success_perc
3	50
2	75
1	100