8 Week SQL Challenge - Week 2

Case Study #2 - Pizza Runner

Viswanadh Gandimenu · Aug 4, 2024

Introduction

Danny was scrolling through his Instagram feed when something really caught his eye - “80s Retro Styling and Pizza Is The Future!”

Danny was sold on the idea, but he knew that pizza alone was not going to help him get seed funding to expand his new Pizza Empire - so he had one more genius idea to combine with it - he was going to *Uberize* it - and so Pizza Runner was launched!

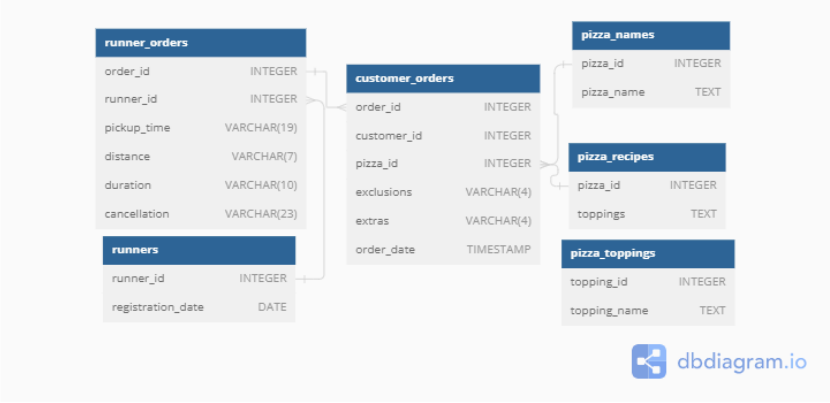
Danny started by recruiting “runners” to deliver fresh pizza from Pizza Runner Headquarters (otherwise known as Danny’s house) and also maxed out his credit card to pay freelance developers to build a mobile app to accept orders from customers.

Available Data

Because Danny had a few years of experience as a data scientist - he was very aware that data collection was going to be critical for his business’ growth.

He has prepared for us an entity relationship diagram of his database design but requires further assistance to clean his data and apply some basic calculations so he can better direct his runners and optimize Pizza Runner’s operations.

Entity Relationship Diagram



Case Study Questions

Before you start writing your SQL queries however - you might want to investigate the data, you may want to do something with some of those null values and data types in the customer_orders and runner_orders tables!

A. Pizza Metrics

1. **How many pizzas were ordered?**

```
SELECT COUNT(pizza_id) AS [No of pizzas]
FROM pizza_runner.customer_orders
```

	No of pizzas
1	14

2. **How many unique customer orders were made?**

```
SELECT DISTINCT customer_id AS [No of unique customers]
FROM pizza_runner.customer_orders
```

	No of unique customers
1	101
2	102
3	103
4	104
5	105

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

```
SELECT runner_id,COUNT(*) AS [No of successful orders]
FROM pizza_runner.customer_orders C
LEFT JOIN pizza_runner.runner_orders AS R
ON R.order_id = C.order_id
WHERE cancellation IS NULL
GROUP BY runner_id
```

	runner_id	No of successful orders
1	1	6
2	2	5
3	3	1

4. **How many of each type of pizza was delivered?**

```
SELECT C.pizza_id,pizza_name, COUNT(*) AS[No of pizza delivered]

FROM pizza_runner.customer_orders C

LEFT JOIN pizza_runner.runner_orders AS R

ON R.order_id = C.order_id

LEFT JOIN pizza_runner.pizza_names AS P ON

P.pizza_id = C.pizza_id

WHERE cancellation IS NULL

GROUP BY C.pizza_id,pizza_name
```

	pizza_id	pizza_name	No of pizza delivered
1	1	Meatlovers	9
2	2	Vegetarian	3

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

```
SELECT customer_id, SUM(IIF(pizza_name  = 'Meatlovers',1,0)) AS
[Meatlovers],  SUM(IIF(pizza_name  = 'Vegetarian',1,0)) AS
[Vegetarian]

FROM pizza_runner.customer_orders C

LEFT JOIN pizza_runner.runner_orders AS R

ON R.order_id = C.order_id

LEFT JOIN pizza_runner.pizza_names AS P ON

P.pizza_id = C.pizza_id

GROUP BY customer_id
```

	customer_id	Meatlovers	Vegetarian
1	101	2	1
2	102	2	1
3	103	3	1
4	104	3	0
5	105	0	1

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

```
SELECT TOP 1 COUNT(C.pizza_id) AS [MAX Pizzas sold in single
order]
```

```
FROM pizza_runner.customer_orders C

LEFT JOIN pizza_runner.runner_orders AS R

ON R.order_id = C.order_id

LEFT JOIN pizza_runner.pizza_names AS P ON

P.pizza_id = C.pizza_id

WHERE cancellation IS NULL

GROUP BY C.order_id

ORDER BY [MAX Pizzas sold in single order] DESC
```

	MAX Pizzas sold in single order
1	3

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

```
SELECT C.customer_id,

SUM(IIF(exclusions IS NOT NULL OR extras IS NOT NULL,1,0)) AS
[changes],

SUM(IIF(exclusions IS NULL AND extras IS NULL,1,0)) AS [No
changes]

FROM pizza_runner.customer_orders C

LEFT JOIN pizza_runner.runner_orders AS R

ON R.order_id = C.order_id

WHERE cancellation IS NULL

GROUP BY C.customer_id
```

	customer_id	changes	No changes
1	101	0	2
2	102	0	3
3	103	3	0
4	104	2	1
5	105	1	0

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

```
SELECT SUM(changes) AS [Orders with exclusions and extras] from(

SELECT

CASE WHEN exclusions IS NOT NULL AND extras IS NOT NULL THEN 1
ELSE 0 END AS [changes]

FROM pizza_runner.customer_orders C

LEFT JOIN pizza_runner.runner_orders AS R

ON R.order_id = C.order_id

WHERE cancellation IS NULL) as t
```

	Orders with exclusions and extras
1	1

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

```
SELECT DATEPART(HOUR,order_time) AS [Hourly Orders], COUNT(*) AS
[volume of pizzas]

FROM pizza_runner.customer_orders C

LEFT JOIN pizza_runner.runner_orders AS R

ON R.order_id = C.order_id

WHERE cancellation IS NULL

GROUP BY DATEPART(HOUR,order_time)
```

	Hourly Orders	volume of pizzas
1	13	3
2	18	3
3	19	1
4	21	2
5	23	3

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

```
SELECT DATENAME(weekday,order_time) AS [Week], COUNT(*) AS
[volume of pizzas]

FROM pizza_runner.customer_orders C

LEFT JOIN pizza_runner.runner_orders AS R
```

```
ON R.order_id = C.order_id

WHERE cancellation IS NULL

GROUP BY DATENAME(weekday,order_time)
```

	Week	volume of pizzas
1	Saturday	5
2	Thursday	3
3	Wednesday	4

B. Runner and Customer Experience

11. **How many runners signed up for each 1 week period? (i.e. week starts 2021-01-01)**
12. **What was the average time in minutes it took for each runner to arrive at the Pizza Runner HQ to pickup the order?**

```
SELECT RO.runner_id,
AVG(DATEDIFF(MINUTE,CO.order_time,RO.pickup_time)) AS [Average
Time Taken]

FROM pizza_runner.customer_orders AS CO

LEFT JOIN pizza_runner.runner_orders AS RO

ON CO.order_id = RO.order_id

GROUP BY RO.runner_id
```

	runner_id	Average Time Taken
1	1	15
2	2	24
3	3	10

13. **What was the average distance travelled for each customer?**

```
SELECT RO.runner_id, AVG(distance) AS [Average Distance]

FROM pizza_runner.customer_orders AS CO

LEFT JOIN pizza_runner.runner_orders AS RO
```

```
ON CO.order_id = RO.order_id

GROUP BY RO.runner_id
```

	runner_id	Average Distance
1	1	14.333333
2	2	23.400000
3	3	10.000000

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

```
SELECT  MAX(duration) AS [Longest time], min(duration) AS
[Shortest time]

FROM pizza_runner.customer_orders AS CO

LEFT JOIN pizza_runner.runner_orders AS RO

ON CO.order_id = RO.order_id
```

	Longest time	Shortest time
1	40	10

15. **What was the average speed for each runner for each delivery and do you notice any trend for these values?**

```
--NA
```

16. **What is the successful delivery percentage for each runner?**

```
Data was not sufficent has there is no reason to find the
percentage as there is no direct cancellation from rider.
Basically the percentage is --100% for each rider as per given
table.
```

C. Pricing and Ratings

17. **If a Meat Lovers pizza costs $12 and Vegetarian costs $10 and there were no charges for changes - how much money has Pizza Runner made so far if there are no delivery fees?**
18. **What if there was an additional $1 charge for any pizza extras?**

```
SELECT Pizza_name, concat('$','',SUM(CASE WHEN pizza_name =
'Meatlovers' THEN 12 ELSE 10 END) )AS Price,

concat('$','',SUM(CASE WHEN (pizza_name = 'Meatlovers' OR extras
IS NOT NULL) THEN 13 ELSE 11 END) )AS [Additional Charge]

FROM pizza_runner.customer_orders AS CO

LEFT JOIN pizza_runner.runner_orders AS RO ON

CO.order_id = RO.order_id

LEFT JOIN pizza_runner.pizza_names AS PN ON

PN.pizza_id = CO.pizza_id

WHERE cancellation IS NULL

GROUP BY Pizza_name;
```

	Pizza_name	Price	Additional Charge
1	Meatlovers	$108	$117
2	Vegetarian	$30	$35

19. The Pizza Runner team now wants to add an additional ratings system that allows customers to rate their runner, how would you design an additional table for this new dataset - generate a schema for this new table and insert your own data for ratings for each successful customer order between 1 to 5.

```
  CREATE TABLE runner_ratings (

    rating_id INT PRIMARY KEY IDENTITY(1,1),

    order_id INT,

    customer_id INT,

    runner_id INT,

    rating INT CHECK (rating BETWEEN 1 AND 5),

    FOREIGN KEY (order_id) REFERENCES customer_orders(order_id),

    FOREIGN KEY (runner_id) REFERENCES runner_orders(runner_id)

);

INSERT INTO runner_ratings (order_id, customer_id, runner_id,
rating) VALUES

    (1, 101, 1, 4),
```

```
(2, 101, 1, 5),
(3, 102, 1, 3),
(4, 103, 2, 4),
(5, 104, 3, 5),
(6, 101, 3, 2),
(7, 105, 2, 4),
(8, 102, 2, 5),
(9, 103, 1, 3),
(10, 104, 1, 4);
```

20. If a Meat Lovers pizza was $12 and Vegetarian $10 fixed prices with no cost for extras and each runner is paid $0.30 per kilometre traveled - how much money does Pizza Runner have left over after these deliveries?

```
SELECT runner_id,  SUM(distance * 0.30) AS [Runner Price]
FROM pizza_runner.customer_orders AS CO
LEFT JOIN
pizza_runner.runner_orders AS RO ON
CO.order_id = RO.order_id
LEFT JOIN pizza_runner.pizza_names AS PN ON
PN.pizza_id = CO.pizza_id
WHERE cancellation IS NULL
GROUP BY runner_id
```

	runner_id	Runner Price
1	1	25.8000
2	2	35.1000
3	3	3.0000

The End.

www.linkedin.com/in/viswanadh-gandimenu