PART-1

Hot PIZZA RUNNER

@picnicrautaray

8_week_SQL_challenge

INTRODUCTION

According to this source, around 115 million kilograms of pizza are consumed daily worldwide. (Well according to Wikipedia anyway...)

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.

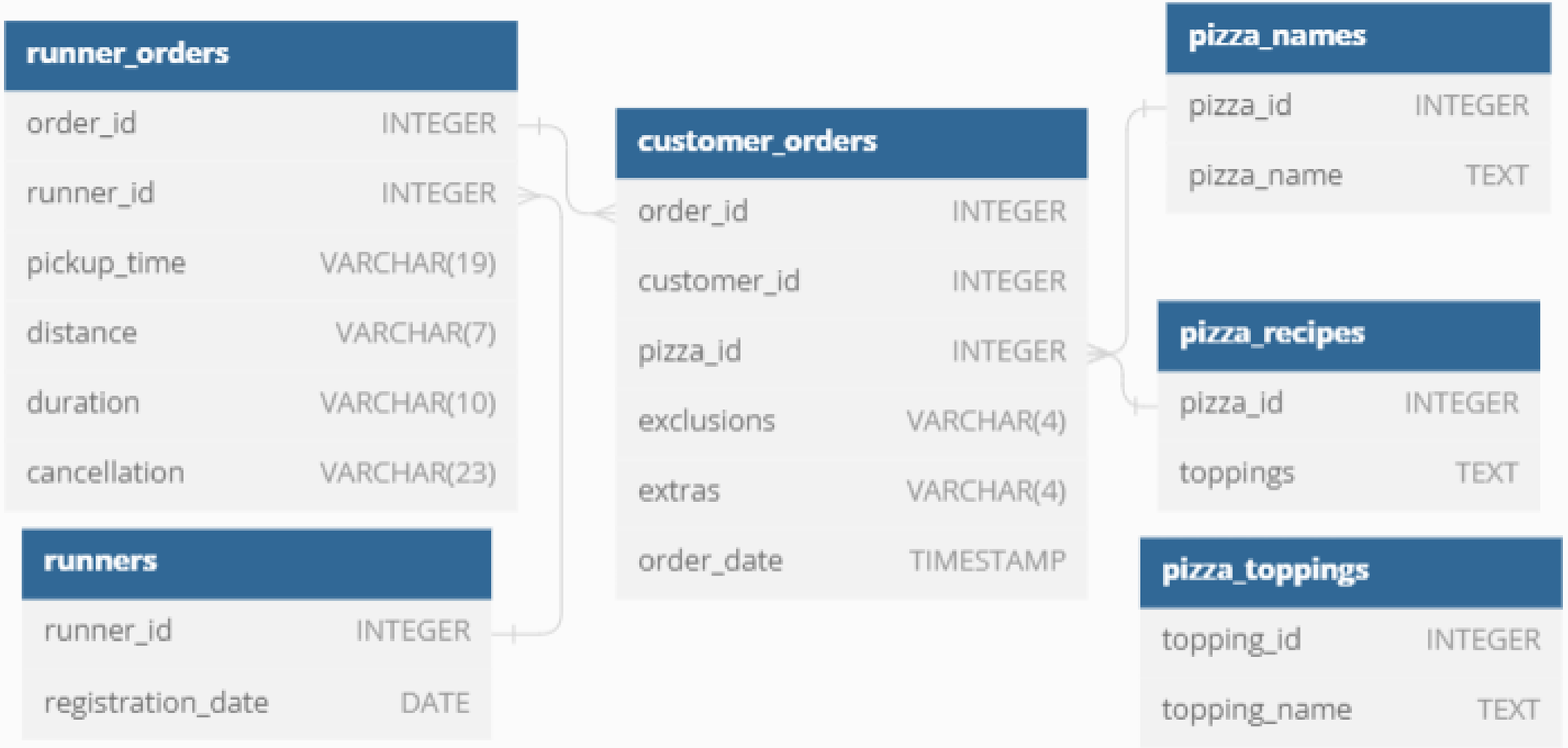
PROBLEM STATEMENT

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 optimise Pizza Runner's operations.

SCHEMA

Entity Relationship Diagram



DATA CLEANING PROCESS

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.

DATA CLEANING PROCESS

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 2 : Customer_orders

DATA CLEANING NEEDED

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.

There are some known data issues with this table so be careful when using this in your queries - make sure to check the data types for each column in the schema SQL!

DATA CLEANING NEEDED

Table 3: runner_orders

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 C
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 Ca
10	1	2020-01-11 18:50:20	10km	10minutes	null

DATA CLEANING PROCESS

Table 3: runner_orders

```
select *,
        case when exclusions = 'null' then '' else exclusions END as update_exclusions,
        case when extras = 'null' or extras is  NULL THEN '' else extras END as update_extras
        from customer_orders;
update customer_orders
set exclusions = case when exclusions = 'null' then '' else exclusions END,
    extras = case when extras = 'null' or extras is  NULL THEN '' else extras END
select * from customer_orders
```

	order_id integer	customer_id integer	pizza_id integer	exclusions character varying (4)	extras character varying (4)	order_time timestamp without time zone
1	1	101	1			2020-01-01 18:05:02
2	2	101	1			2020-01-01 19:00:52
3	3	102	1			2020-01-02 23:51:23
4	3	102	2			2020-01-02 23:51:23
5	4	103	1	4		2020-01-04 13:23:46
6	4	103	1	4		2020-01-04 13:23:46
7	4	103	2	4		2020-01-04 13:23:46
8	5	104	1		1	2020-01-08 21:00:29
9	6	101	2			2020-01-08 21:03:13
10	7	105	2		1	2020-01-08 21:20:29
11	8	102	1			2020-01-09 23:54:33
12	9	103	1	4	1, 5	2020-01-10 11:22:59
13	10	104	1			2020-01-11 18:34:49

DATA CLEANING PROCESS

Table 3: runner_orders

```
CREATE TABLE runner_orders_new AS
SELECT order_id,
       runner_id,
       CASE WHEN pickup_time = 'null' THEN NULL ELSE CAST(pickup_time AS TIMESTAMP) END AS pickup_time_adjusted,
       CAST(REGEXP_SUBSTR(distance, '[0-9]+(\.[0-9]+)?') AS FLOAT) AS dist_adjusted,
       CAST(REGEXP_SUBSTR(duration, '^[0-9]+') AS INT) AS dur_adjusted,
       CASE WHEN cancellation IN ('null', '') THEN NULL ELSE cancellation END AS cancellation_adjusted
FROM runner_orders;
select * from runner_orders_new
```

order_id integer	runner_id integer	pickup_time_adjusted timestamp without time zone	dist_adjusted double precision	dur_adjusted integer	cancellation_adjusted character varying
1	1	2020-01-01 18:15:34	20	32	[null]
2	1	2020-01-01 19:10:54	20	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	15	[null]
6	3	[null]	[null]	[null]	Restaurant Cancellation
7	2	2020-01-08 21:30:45	25	25	[null]
8	2	2020-01-10 00:15:02	23.4	15	[null]
9	2	[null]	[null]	[null]	Customer Cancellation

PIZZA METRICS

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

```
select count(*) as total_order
from customer_orders
```

	total_order bigint
1	14

PIZZA METRICS

2. How many unique customer orders were made?

```
select count(distinct order_id)
from customer_orders
```

	count bigint
1	10

PIZZA METRICS

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

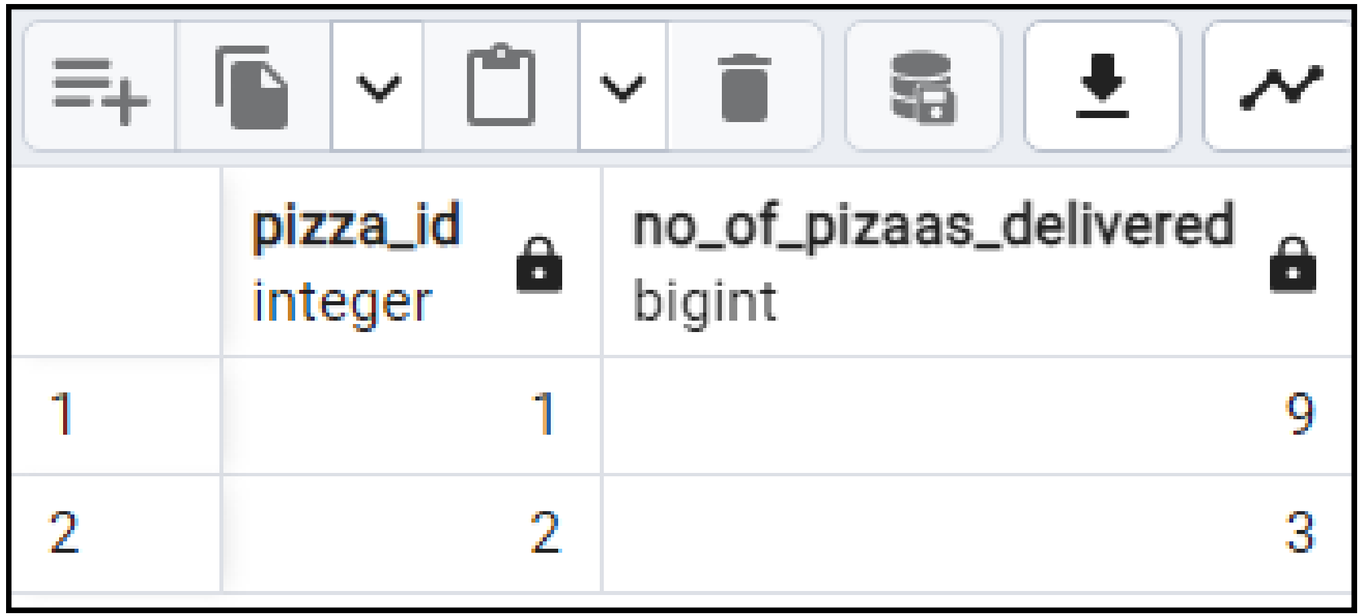
```sql
select runner_id,count(order_id) as successful_orders
from runner_orders_new
where cancellation_adjusted IS NULL
group by runner_id
```

	runner_id integer	successful_orders bigint
1	1	4
2	2	3
3	3	1

PIZZA METRICS

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

```
select distinct co.pizza_id,count(co.pizza_id) as no_of_pizaas_delivered
from customer_orders co
left join runner_orders_new ro
on co.order_id = ro.order_id
where ro.cancellation_adjusted is NULL
group by co.pizza_id;
```



	pizza_id integer	no_of_pizaas_delivered bigint
1	1	9
2	2	3

PIZZA METRICS

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

```
select co.customer_id,np.pizza_name,count(*) as pizza_count
from pizza_names as np
join customer_orders as co
on co.pizza_id = np.pizza_id
group by co.customer_id,np.pizza_name
```

	customer_id integer	pizza_name text	pizza_count bigint
1	105	Vegetarian	2
2	102	Meatlovers	4
3	101	Vegetarian	2
4	104	Meatlovers	6
5	103	Vegetarian	2
6	101	Meatlovers	4
7	103	Meatlovers	6
8	102	Vegetarian	2

PIZZA METRICS

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

```
with max_pizza as
(
select co.order_id,count(co.pizza_id) as pizza_order
from customer_orders as co
join runner_orders_new rn
on co.order_id = rn.order_id
group by co.customer_id,co.order_id
)
select max(pizza_order)
from max_pizza
```

	max bigint
1	3

PIZZA METRICS

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

```sql
select co.customer_id,
       count(case when co.exclusions<> '' or  co.extras <> '' then 1 END) as changes,
       count(case when exclusions = '' and co.extras = '' then 1 end) as no_changes
from customer_orders as co
join runner_orders_new as ro
on co.order_id= ro.order_id
where cancellation_adjusted is null
group by co.customer_id;
```

	customer_id integer	changes bigint	no_changes bigint
1	101	0	2
2	102	0	3
3	105	1	0
4	104	2	1
5	103	3	0

PIZZA METRICS

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

```
select count(*) as special_pizzas
from customer_orders as co
join runner_orders_new rn
on co.order_id = rn.order_id
where co.exclusions <> ''
      and co.extras <> ''
      and cancellation_adjusted is NULL;
```

	special_pizzas bigint
1	1

PIZZA METRICS

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

```
select count(pizza_id) as pizzas_count,extract(HOUR FROM order_time) as hour_of_the_day
  from customer_orders
  group by hour_of_the_day
  order by hour_of_the_day ASC;
```

	pizzas_count bigint	hour_of_the_day numeric
1	1	11
2	3	13
3	3	18
4	1	19
5	3	21
6	3	23

PIZZA METRICS

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

```
select  TO_CHAR(order_time,'DAY') AS day_of_the_week,
         count(pizza_id) as pizzas_count
from customer_orders
group by day_of_the_week
order by day_of_the_week asc;
```

	day_of_the_week text	pizzas_count bigint
1	FRIDAY	1
2	SATURDAY	5
3	THURSDAY	3
4	WEDNESDAY	5

PIZZA METRICS - INSIGHTS

- Out of **14 pizzas** that were ordered, **10 were unique customer** orders.
- Out of the **8 successful orders, runner_1** had the highest number of successful orders.
- **9 meat lover** pizzas and **3 vegetarian** pizzas were sold.
- The **maximum number of pizzas** delivered in a single order is **3 pizzas.**
- There was **1 pizza** that was delivered with **both exclusions and extras.**

8-WEEK_SQL_CHALLENGE

THANK YOU

8_week_SQL_challenge