

Case Study #2 - Pizza Runner

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!

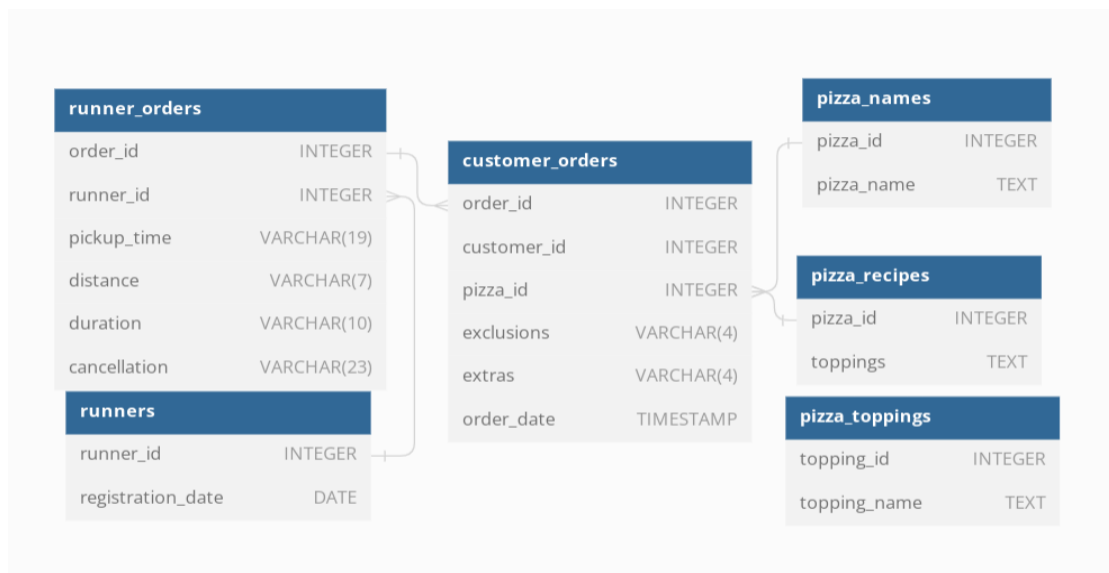
Danny started by recruiting “runners” to deliver fresh pizza from Pizza Runner Headquarters (otherwise known as Danny’s house) and also maxed out his credit card to pay freelance developers to build a mobile app to accept orders from customers.

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



Data Cleaning and Transformation

Looking at the customer_orders table below, we can see that there are

- In the exclusions column, there are missing/ blank spaces ' ' and null values.
- In the extras column, there are missing/ blank spaces ' ' and null values.

order_id	customer_id	pizza_id	exclusions	extras	order_time
1	101	1			2021-01-01 18:05:02.000
2	101	1			2021-01-01 19:00:52.000
3	102	1			2021-01-02 23:51:23.000
3	102	2		null	2021-01-02 23:51:23.000
4	103	1	4		2021-01-04 13:23:46.000
4	103	1	4		2021-01-04 13:23:46.000
4	103	2	4		2021-01-04 13:23:46.000
5	104	1	null	1	2021-01-08 21:00:29.000
6	101	2	null	null	2021-01-08 21:03:13.000
7	105	2	null	1	2021-01-08 21:20:29.000
8	102	1	null	null	2021-01-09 23:54:33.000
9	103	1	4	1, 5	2021-01-10 11:22:59.000
10	104	1	null	null	2021-01-11 18:34:49.000
10	104	1	2, 6	1, 4	2021-01-11 18:34:49.000

Our course of action to clean the table:

- Create a new table with all the columns.
- Remove null values in exclusions and extras columns and replace with blank space ' '.

```
insert into customer_order_new (order_id,customer_id,pizza_id,exclusions, extras,order_time )
select
  order_id,
  customer_id,
  pizza_id,
  case
    when exclusions like 'null' then ' '
    else exclusions
  end as exclusions,
  case
    when extras is null or extras like 'null' then ' '
    else extras
  end as extras,order_time
from customers_orders
```

This is how the clean customers_orders_new table looks like and we will use this table to run all our queries

order_id	customer_id	pizza_id	exclusions	extras	order_time
1	101	1			2021-01-01 18:05:02.000
2	101	1			2021-01-01 19:00:52.000
3	102	1			2021-01-02 23:51:23.000
3	102	2			2021-01-02 23:51:23.000
4	103	1	4		2021-01-04 13:23:46.000
4	103	1	4		2021-01-04 13:23:46.000
4	103	2	4		2021-01-04 13:23:46.000
5	104	1		1	2021-01-08 21:00:29.000
6	101	2			2021-01-08 21:03:13.000
7	105	2		1	2021-01-08 21:20:29.000
8	102	1			2021-01-09 23:54:33.000
9	103	1	4	1, 5	2021-01-10 11:22:59.000
10	104	1			2021-01-11 18:34:49.000
10	104	1	2, 6	1, 4	2021-01-11 18:34:49.000

Looking at the runner_orders table below, we can see that there are

- In the pickup_time column, there are null values.
- In the distance column, there are null values.
- In the duration column, there are null values.
- In the cancellation column, there are missing/ blank spaces ' ' and null values.

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	<i>null</i>
4	2	2021-01-04 13:53:03	23.4	40	<i>null</i>
5	3	2021-01-08 21:10:57	10	15	<i>null</i>
6	3	<i>null</i>	<i>null</i>	<i>null</i>	Restaurant Cancellation
7	2	2021-01-08 21:30:45	25km	25mins	<i>null</i>
8	2	2021-01-10 00:15:02	23.4 km	15 minute	<i>null</i>
9	2	<i>null</i>	<i>null</i>	<i>null</i>	Customer Cancellation
10	1	2021-01-11 18:50:20	10km	10minutes	<i>null</i>

Our course of action to clean the table:

- In pickup_time column, remove nulls and replace with blank space ''.
- In distance column, remove "km" and nulls and replace with blank space ''.
- In duration column, remove "minutes", "minute" and nulls and replace with blank space ''.
- In cancellation column, remove null and replace with blank space ''

```
insert into runner_order_new
select
  order_id,
  runner_id,
  case
    when pickup_time like 'null' then ' '
    else pickup_time
  end as pickup_time,
  case
    when distance like 'null' then ' '
    when distance like '%km' then trim('km' from distance)
    else distance
  end as distance ,
  case
    when duration like 'null' then ' '
    when duration like '%minutes' then trim ('minutes' from duration )
    when duration like '%mins' then trim ('mins' from duration )
    when duration like '%minute' then trim ('minute' from duration )
    else duration
  end as duration ,
  case
    when cancellation is null or cancellation like 'null' then ' '
    else cancellation
  end as cancellation
from runner_orders
```

This is how the clean runner_orders_new table looks like and we will use this table to run all our queries.

order_id	runner_id	pickup_time	distance	duration	cancellation
1	1	2021-01-01 18:15:34	20	32	
2	1	2021-01-01 19:10:54	20	27	
3	1	2021-01-03 00:12:37	13.4	20	
4	2	2021-01-04 13:53:03	23.4	40	
5	3	2021-01-08 21:10:57	10	15	
6	3				Restaurant Cancellation
7	2	2021-01-08 21:30:45	25	25	
8	2	2021-01-10 00:15:02	23.4	15	
9	2				Customer Cancellation
10	1	2021-01-11 18:50:20	10	10	

Case Study Questions

A. Pizza Metrics

1. How many pizzas were ordered?

```
select  
count(order_id) as pizza_order_count  
from customer_order_new
```

Result:

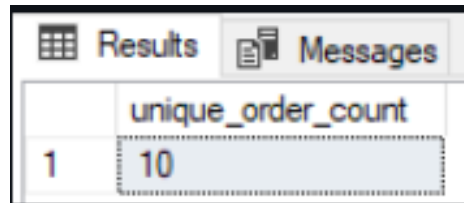
Results		Messages	
pizza_order_count			
1	14		

- Total of 14 pizzas were ordered

2. How many unique customer orders were made?

```
select  
count (distinct order_id) as unique_order_count  
from customer_order_new
```

Result:



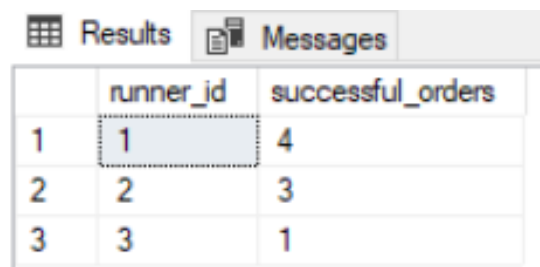
Results		Messages
	unique_order_count	
1	10	

- There are 10 unique customer orders.

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

```
select
  runner_id ,
  count (order_id)as successful_orders
from runner_order_new
where distance<>0
group by runner_id
```

Result:



Results		Messages
	runner_id	successful_orders
1	1	4
2	2	3
3	3	1

- Runner 1 has 4 successful delivered orders.
- Runner 2 has 3 successful delivered orders.
- Runner 3 has 1 successful delivered order

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

```
select
  p.pizza_name ,
  count(c.pizza_id) as delivered_pizza_count
from customer_order_new c
join runner_order_new r
on c.order_id =r.order_id
join pizza_names p
on c.pizza_id=p.pizza_id
where r.distance!=0
group by p.pizza_name
```

Result:

Results Messages		
	pizza_name	delivered_pizza_count
1	Meatlovers	9
2	Vegetarian	3

- There are 9 delivered Meatlovers pizzas and 3 Vegetarian pizzas

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


```

select
  c.customer_id,
  p.pizza_name,
  count (p.pizza_name) as order_count
from customer_order_new c
join pizza_names p
on c.pizza_id = p.pizza_id
group by c.customer_id, p.pizza_name

```

Result:

	customer_id	pizza_name	order_count
1	101	Meatlovers	2
2	101	Vegetarian	1
3	102	Meatlovers	2
4	102	Vegetarian	1
5	103	Meatlovers	3
6	103	Vegetarian	1
7	104	Meatlovers	3
8	105	Vegetarian	1

- Customer 101 ordered 2 Meatlovers pizzas and 1 Vegetarian pizza.
- Customer 102 ordered 2 Meatlovers pizzas and 2 Vegetarian pizzas.
- Customer 103 ordered 3 Meatlovers pizzas and 1 Vegetarian pizza.
- Customer 104 ordered 3 Meatlovers pizza.
- Customer 105 ordered 1 Vegetarian pizza

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

```
With CTE AS (  
  select  
    c.order_id ,  
    count (c.pizza_id) pizza_per_order  
  from customer_order_new c  
  join runner_order_new r  
  on c.order_id=r.order_id  
  where r.distance<>0  
  group by c.order_id)  
select max(pizza_per_order) pizza_count  
from CTE
```

Result:

	order_id	pizza_per_order
1	1	1
2	2	1
3	3	2
4	4	3
5	5	1
6	7	1
7	8	1
8	10	2

- Maximum number of pizza delivered in a single order is 3 pizzas.

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

```

select
c.customer_id ,
sum (case when c.exclusions <> ' ' or c.extras <> ' ' then 1 else 0 end )as at_least_1_change,
sum (case when c.exclusions = ' ' and c.extras = ' ' then 1 else 0 end) as no_change
from customer_order_new c
join runner_order_new r
on c.order_id =r.order_id
where r.distance <>0
group by c.customer_id
order by c.customer_id

```

Result:

	customer_id	at_least_1_change	no_change
1	101	0	2
2	102	0	3
3	103	3	0
4	104	2	1
5	105	1	0

- Customer 101 and 102 likes his/her pizzas per the original recipe.
- Customer 103, 104 and 105 have their own preference for pizza topping and requested at least 1 change (extra or exclusion topping) on their pizza.

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

```
select
count (c.order_id)orders_both_exclusions_extras
from customer_order_new c
join runner_order_new r
on c.order_id=r.order_id
where c.exclusions <> ' ' and c.extras <> ' ' and r.distance<>0
```

Result:

Results		Messages
orders_both_exclusions_extras		
1	2	

- Only 2 pizza delivered that had both extra and exclusion topping. That's one fussy customer!

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

```
select
datepart(hour,[order_time])as hours_per_day,
count(order_id) as count_pizza
from customer_order_new
group by datepart(hour,[order_time])
```

Result:

Results			Messages
	hour_of_day	pizza_count	
1	11	1	
2	12	2	
3	13	3	
4	18	3	
5	19	1	
6	21	3	
7	23	1	

- Highest volume of pizza ordered is at 13 (1:00 pm), 18 (6:00 pm) and 21 (9:00 pm).
- Lowest volume of pizza ordered is at 11 (11:00 am), 19 (7:00 pm) and 23 (11:00 pm).

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

```
select
format( dateadd (day,2,order_time),'dddd')as day_of_week ,
count(order_id) as total_pizzas_ordered
from customer_order_new
group by format( dateadd (day,2,order_time),'dddd')
```

Result:

Results		Messages
	day_of_week	total_pizzas_ordered
1	Friday	5
2	Monday	5
3	Saturday	3
4	Sunday	1

- There are 5 pizzas ordered on Friday and Monday.
- There are 3 pizzas ordered on Saturday.
- There is 1 pizza ordered on Sunday.

B. Runner and Customer Experience

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

```
select
datepart(week,registration_date)as registration_week,
count (runner_id)as runners_signup
from runners
group by datepart(week,registration_date)
```

Result:

	registration_week	runner_signup
1	1	2
2	2	1
3	3	1

- On Week 1 of Jan 2021, 2 new runners signed up.
- On Week 2 and 3 of Jan 2021, 1 new runner signed up

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?

```

with CTE as (
select
c.order_id,
c.order_time,
r.pickup_time ,
DATEDIFF (minute,c.order_id,r.pickup_time)as pickup_minutes
from customer_order_new c
join runner_order_new r
on c.order_id=r.order_id
where r.distance<> 0
group by c.order_id,c.order_time,r.pickup_time)
select avg (pickup_minutes) as avg_pickup_minutes
from cte ;

```

Result:

Results		Messages	
		avg_pickup_minutes	
1		15	

- The average time taken in minutes by runners to arrive at Pizza Runner HQ to pick up the order is 15 minutes.

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


```

with CTE as (
select
    c.order_id,
    count (c.order_id) as pizza_order ,
    c.order_time,r.pickup_time,
    datediff (minute,c.order_time,r.pickup_time) as prep_time_minutes
from customer_order_new c
join runner_order_new 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 CTE
group by pizza_order

```

Result:

	pizza_order	avg_prep_time_minutes
1	1	12
2	2	16
3	3	30

- On average, a single pizza order takes 12 minutes to prepare.
- An order with 3 pizzas takes 30 minutes at an average of 10 minutes per pizza.
- It takes 16 minutes to prepare an order with 2 pizzas which is 8 minutes per pizza — making 2 pizzas in a single order the ultimate efficiency rate.

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

```
select
  c.customer_id ,
  avg (r.distance ) avg_distance_travelled
from customer_order_new c
join runner_order_new r
on r.order_id= c.order_id
where r.distance<>0
group by c.customer_id
```

Result:

	customer_id	avg_distance_travelled
1	101	20
2	102	16.7333333333333
3	103	23.4
4	104	10
5	105	25

- Customer 104 stays the nearest to Pizza Runner HQ at average distance of 10km, whereas Customer 105 stays the furthest at 25km

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

```
select
  max(cast (duration as numeric)) -min(cast (duration as numeric))as delivery_time_diff
from runner_order_new
where duration not like ' ';
```

Result:

Results		Messages	
		delivery_time_diff	
1		30	

- The difference between longest (40 minutes) and shortest (10 minutes) delivery time for all orders is 30 minutes.

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

```
select
  r.runner_id ,
  c.customer_id ,
  c.order_id ,
  count(c.order_id)as pizza_count,
  r.distance,
  (r.duration /60)as duration_hour,ROUND (r.distance/r.duration *60,2)as avg_speed
from customer_order_new c
join runner_order_new r
on c.order_id=r.order_id
where r.distance<>0
group by r.runner_id ,c.customer_id ,r.distance,c.order_id,r.distance,r.duration
order by c.order_id
```

Result:

Results

Messages

	runner_id	customer_id	order_id	pizza_count	distance	duration_hr	avg_speed
1	1	101	1	1	20	0	37.5
2	1	101	2	1	20	0	44.44
3	1	102	3	2	13.4	0	40.2
4	2	103	4	3	23.4	0	35.1
5	3	104	5	1	10	0	40
6	2	105	7	1	25	0	60
7	2	102	8	1	23.4	0	93.6
8	1	104	10	2	10	0	60

- Runner 1's average speed runs from 37.5km/h to 60km/h.
- Runner 2's average speed runs from 35.1km/h to 93.6km/h. Danny should investigate Runner 2 as the average speed has a 300% fluctuation rate!
- Runner 3's average speed is 40km/h

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

```
select runner_id ,
ROUND (100* sum (case when distance =0 then 0 else 1 end) /count(*),0) as succesful_delivery
from runner_order_new
group by runner_id
```

Result:

Results		Messages
	runner_id	succesful_delivery
1	1	100
2	2	75
3	3	50

- Runner 1 has 100% successful delivery.
- Runner 2 has 75% successful delivery.
- Runner 3 has 50% successful delivery