
PIZZA
RUNNER

INTRODUCTION

Did you know that over 115 million kilograms of pizza is consumed daily worldwide??? (Well according to Wikipedia anyway...)

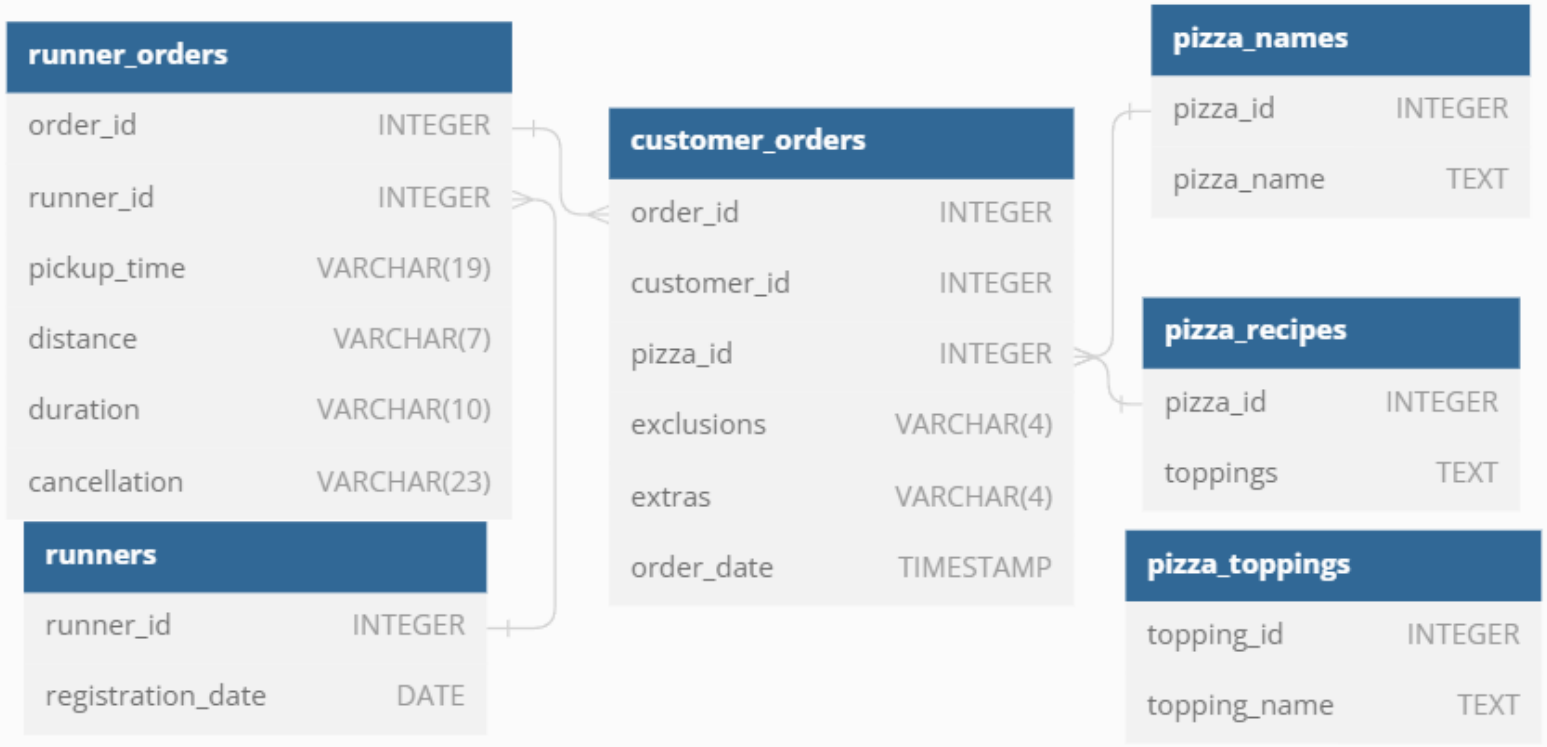
Danny was scrolling through his Instagram feed when something really caught his eye - "80s Retro Styling and Pizza Is The Future!" Danny was sold on the idea, but he knew that pizza alone was not going to help him get seed funding to expand his new Pizza Empire - so he had one more genius idea to combine 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.
All datasets exist within the pizza_runner database schema - be sure to include this reference within your SQL scripts as you start exploring the data and answering the case study questions.

ENTITY RELATIONSHIP DIAGRAM



DATASETS

Runners

runner_id	registration_date
1	2021-01-01
2	2021-01-03
3	2021-01-08
4	2021-01-15

Pizza Name

pizza_id	pizza_name
1	Meat Lovers
2	Vegetarian

Pizza Toppings

pizza_id	toppings
1	1, 2, 3, 4, 5, 6, 8, 10
2	4, 6, 7, 9, 11, 12

Customer Orders

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

Toppings Name

topping_id	topping_name
1	Bacon
2	BBQ Sauce
3	Beef
4	Cheese
5	Chicken
6	Mushrooms
7	Onions
8	Pepperoni
9	Peppers
10	Salami
11	Tomatoes
12	Tomato Sauce

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

Before starting any analysis on business requirement it is utmost important to clean the datasets, as it can give error and wrong values whilst performing any analysis.

Tables like Customer Orders and Runner Orders are need to be cleaned as it has many values that are not consistent and many hard null values.

customer_orders Data Cleaning

order_id	customer_id	pizza_id	exclusions	extras	order_time
1	101	1			2020-01-01 18:05:02
2	101	1			2020-01-01 19:00:52
3	102	1			2020-01-02 23:51:23
3	102	2		NULL	2020-01-02 23:51:23
4	103	1	4		2020-01-04 13:23:46
4	103	1	4		2020-01-04 13:23:46
4	103	2	4		2020-01-04 13:23:46
5	104	1	null	1	2020-01-08 21:00:29
6	101	2	null	null	2020-01-08 21:03:13
7	105	2	null	1	2020-01-08 21:20:29
8	102	1	null	null	2020-01-09 23:54:33
9	103	1	4	1, 5	2020-01-10 11:22:59
10	104	1	null	null	2020-01-11 18:34:49
10	104	1	2, 6	1, 4	2020-01-11 18:34:49

```sql
update customer_orders
set exclusions = null
where exclusions = 'null' or exclusions = '';

update customer_orders
set extras = null
where extras = 'null' or extras = '';
```

order_id	customer_id	pizza_id	exclusions	extras	order_time
1	101	1	NULL	NULL	2020-01-01 18:05:02
2	101	1	NULL	NULL	2020-01-01 19:00:52
3	102	1	NULL	NULL	2020-01-02 23:51:23
3	102	2	NULL	NULL	2020-01-02 23:51:23
4	103	1	4	NULL	2020-01-04 13:23:46
4	103	1	4	NULL	2020-01-04 13:23:46
4	103	2	4	NULL	2020-01-04 13:23:46
5	104	1	NULL	1	2020-01-08 21:00:29
6	101	2	NULL	NULL	2020-01-08 21:03:13
7	105	2	NULL	1	2020-01-08 21:20:29
8	102	1	NULL	NULL	2020-01-09 23:54:33
9	103	1	4	1, 5	2020-01-10 11:22:59
10	104	1	NULL	NULL	2020-01-11 18:34:49
10	104	1	2, 6	1, 4	2020-01-11 18:34:49

runner_orders Data Cleaning

order_id	runner_id	pickup_time	distance	duration	cancellation
1	1	2020-01-01 18:15:34	20km	32 minutes	
2	1	2020-01-01 19:10:54	20km	27 minutes	
3	1	2020-01-03 00:12:37	13.4km	20 mins	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 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

```sql
update runner_orders
set duration = replace(duration, 'minutes','');

update runner_orders
set duration = replace(duration, 'minute','');

update runner_orders
set duration = replace(duration, 'mins','');

update runner_orders
set distance = replace(distance, 'km','');

update runner_orders
set cancellation = null
where cancellation = 'null' or  cancellation = '';
```

order_id	runner_id	pickup_time	distance	duration	cancellation
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 C...
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 Ca...
10	1	2020-01-11 18:50:20	10	10	NULL

PIZZA METRICS

1. How many pizzas were ordered?

```
177 •   select count(*) as total_pizzas_ordered
178     from customer_orders;
```

Result Grid | Filter Rows: | Export: | Wrap Cel

total_pizzas_ordered
14

2. How many unique customer orders were made?

```
182 •   select count(distinct customer_id) as unique_customers
183     from customer_orders;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

unique_customers
5

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

```
186 •   select runner_id, count(order_id) as successful_orders
187     from runner_orders where cancellation is null group by runner_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

runner_id	successful_orders
1	4
2	3
3	1

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

```
192 •   select pn.pizza_name, count(ro.order_id) as no_of_orders
193     from runner_orders ro right join customer_orders co on ro.order_id = co.order_id
194     left join pizza_names pn on pn.pizza_id = co.pizza_id where ro.cancellation is null
195     group by pn.pizza_name;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

pizza_name	no_of_orders
Meatlovers	9
Vegetarian	3

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

```
199 •  select co.customer_id, pn.pizza_name, count(co.pizza_id) as pizza_ordered
200    from runner_orders ro right join customer_orders co on ro.order_id = co.order_id
201    left join pizza_names pn on pn.pizza_id = co.pizza_id where ro.cancellation is null
202    group by 1,2;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	pizza_name	pizza_ordered
101	Meatlovers	2
102	Meatlovers	2
102	Vegetarian	1
103	Meatlovers	2
103	Vegetarian	1
104	Meatlovers	3
105	Vegetarian	1

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

```
206 •  select max(no_of_pizza) as max_pizza_delivered from (select co.order_id, count(co.pizza_id) as no_of_pizza
207    from customer_orders co join runner_orders ro on ro.order_id = co.order_id
208    where cancellation is null
209    group by order_id) t;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

max_pizza_delivered
3

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

```
213 •  select customer_id,
214    sum(if(co.exclusions is null and co.extras is null, 1, 0)) as no_changes,
215    sum(if(co.exclusions is null and co.extras is null, 0, 1)) as atleast_one_change
216    from customer_orders co join runner_orders ro on ro.order_id = co.order_id
217    where cancellation is null
218    group by customer_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

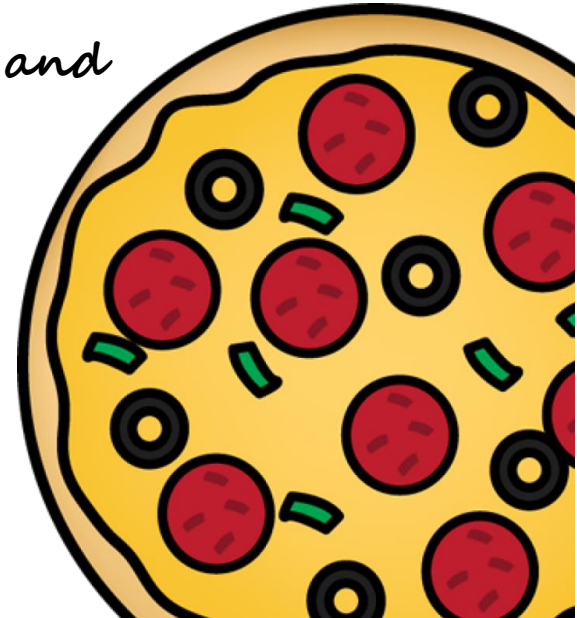
customer_id	no_changes	atleast_one_change
101	2	0
102	3	0
103	0	3
104	1	2
105	0	1

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

```
222 •  select
223    sum(if(exclusions is not null and extras is not null, 1, 0)) as pizza_delivered
224    from customer_orders co join runner_orders ro on ro.order_id = co.order_id
225    where cancellation is null;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

pizza_delivered
1



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

```
220 •   select hour(order_time) as hour_of_day, count(*) as no_of_pizza
221     from customer_orders
222     group by 1 order by 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

hour_of_day	no_of_pizza
11	1
13	3
18	3
19	1
21	3
23	3

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

```
226 •   select dayofweek(order_time) as day_num, dayname(order_time) as day_of_week,
227     count(*) as no_of_pizza
228     from customer_orders
229     group by 1,2 order by 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

day_num	day_of_week	no_of_pizza
4	Wednesday	5
5	Thursday	3
6	Friday	1
7	Saturday	5

INSIGHTS

1. Total Customers are 5 and total no of pizzas ordered was 14
2. Meatlovers Pizza was the top sold item.
3. Customer 101 and 102 does not prefer any change in there pizza.
4. Pizzas were mostly bought at night.
5. Wednesday is the day with highest selling pizza.

Runner and Customer Experience

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

```
235 •  select weekofyear(registration_date + interval 1 week) as week_num, count(*) as runners_signed
236    from runners group by 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

week_num	runners_signed
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 pickup the order?

```
240 •  select runner_id, round(avg(time_diff)) as avg_time from (
241        select distinct co.order_id, runner_id,
242        timestampdiff(minute, co.order_time, ro.pickup_time) as time_diff
243        from customer_orders co inner join runner_orders ro on ro.order_id = co.order_id
244        where cancellation is null
245    )t group by 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

runner_id	avg_time
1	14
2	20
3	10

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

```
249 • select pizza_count, round(avg(time_diff)) as avg_time from (
250       select distinct co.order_id, runner_id,
251       timestampdiff(minute, co.order_time, ro.pickup_time) as time_diff,
252       count(co.order_id) over(partition by co.order_id) as pizza_count
253       from customer_orders co inner join runner_orders ro on ro.order_id = co.order_id
254       where cancellation is null
255   )t group by 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

pizza_count	avg_time
1	12
2	18
3	29

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

```
259 •   select customer_id, round(avg(distance),1) as Avg_distance
260     from customer_orders co inner join runner_orders ro on ro.order_id = co.order_id
261     where cancellation is null group by 1
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	Avg_distance
101	20
102	16.7
103	23.4
104	10
105	25

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

```
265 •   select max(duration) as longest_delivery_time, min(duration) as shortest_delivery_time,
266     max(duration)-min(duration) as difference
267     from runner_orders where cancellation is null;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

longest_delivery_time	shortest_delivery_time	difference
40	10	30

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

```
271 • with speed as(
272     select order_id, runner_id, round(distance/(duration/60),2) as speed_kmph
273     from runner_orders where cancellation is null)
274     select *, round(avg(speed_kmph) over(partition by runner_id),2) as avg_speed
275     from speed;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	runner_id	speed_kmph	avg_speed
1	1	37.5	45.54
2	1	44.44	45.54
3	1	40.2	45.54
10	1	60	45.54
4	2	35.1	62.9
7	2	60	62.9
8	2	93.6	62.9
5	3	40	40

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

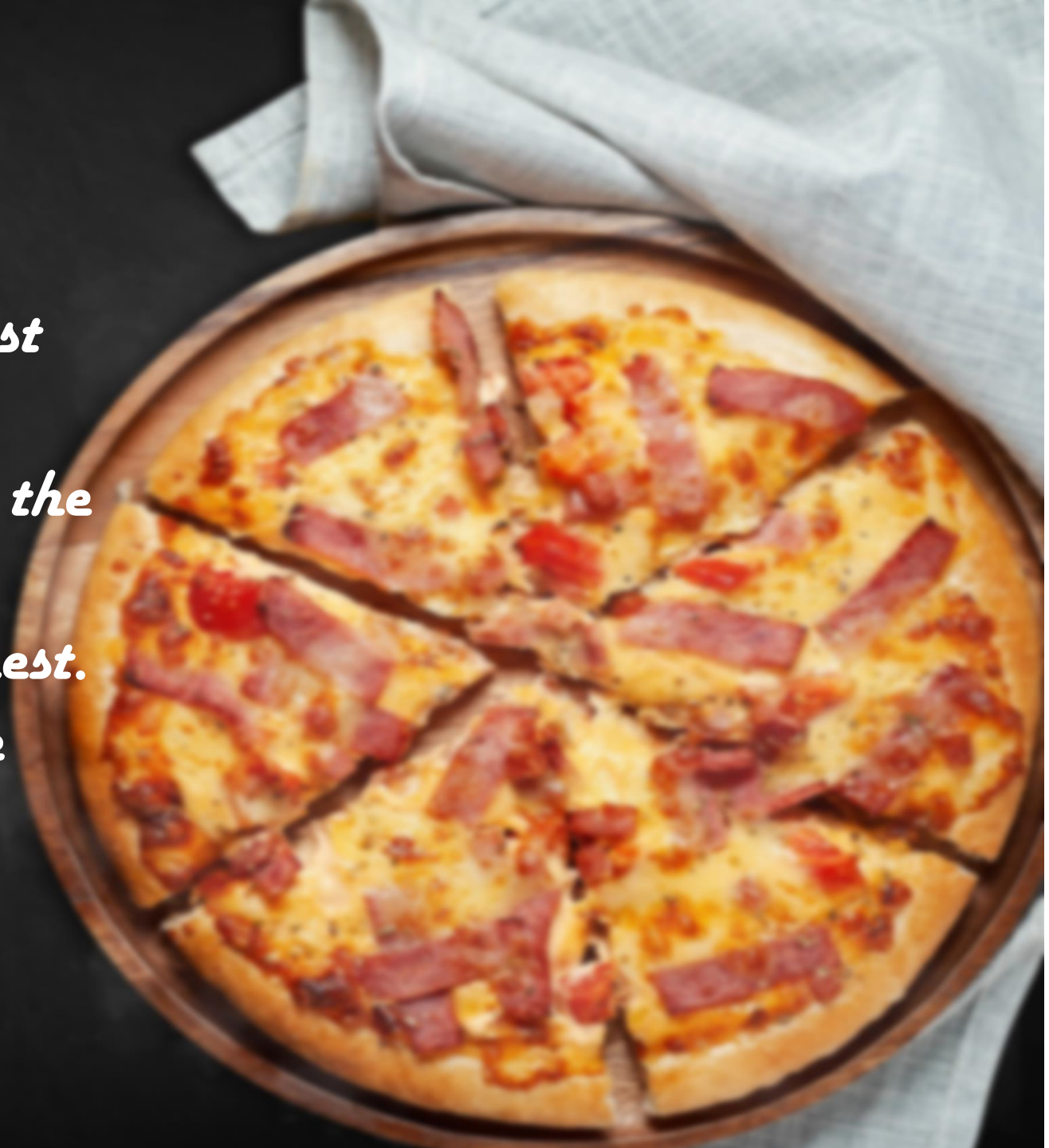
```
279 • with percentage_delivery as(
280     select runner_id,
281     sum(case when cancellation is null then 1 else 0 end) as successful_delivery,
282     count(*) as total_orders from runner_orders group by 1)
283     select runner_id, concat(round(successful_delivery * 100 / total_orders),' %') as success_prcnt
284     from percentage delivery;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

runner_id	success_prcnt
1	100 %
2	75 %
3	50 %

INSIGHTS GATHERED

1. In the first week, there was highest number of runners signed up.
2. It takes longer time to prepare as the quantity of pizza increases.
3. Customer 103 and 105 stays farthest.
4. Runner 2 has the highest average speed (62.9km/hr).
5. Runner 1 has the highest rate of successful delivery.



Ingredient Optimization

Transforming Pizza Recipe and Customer Order table

Result Grid | Filter Rows:

pizza_id	toppings
1	1, 2, 3, 4, 5, 6, 8, 10
2	4, 6, 7, 9, 11, 12

```
create view pizza_recipes_new as(
select t.pizza_id, trim(j.toppings) as topping
from pizza_recipes t
join json_table(
  replace(json_array(t.toppings), ',', '","'),
  '$[*]' columns (toppings varchar(50) path '$')
) j);
```

Result Grid | Filter

pizza_id	topping
1	1
1	2
1	3
1	4
1	5
1	6
1	8
1	10
2	4
2	6
2	7
2	9
2	11
2	12

order_id	customer_id	pizza_id	exclusions	extras	order_time
1	101	1	NULL	NULL	2020-01-01 18:05:02
2	101	1	NULL	NULL	2020-01-01 19:00:52
3	102	1	NULL	NULL	2020-01-02 23:51:23
3	102	2	NULL	NULL	2020-01-02 23:51:23
4	103	1	4	NULL	2020-01-04 13:23:46
4	103	1	4	NULL	2020-01-04 13:23:46
4	103	2	4	NULL	2020-01-04 13:23:46
5	104	1	NULL	1	2020-01-08 21:00:29
6	101	2	NULL	NULL	2020-01-08 21:03:13
7	105	2	NULL	1	2020-01-08 21:20:29
8	102	1	NULL	NULL	2020-01-09 23:54:33
9	103	1	4	1, 5	2020-01-10 11:22:59
10	104	1	NULL	NULL	2020-01-11 18:34:49
10	104	1	2, 6	1, 4	2020-01-11 18:34:49

```sql
create view customer_order_new as(
select t.order_id, t.customer_id, t.pizza_id, trim(j.exclusions) as exclusion,
trim(k.extras) as extra, t.order_time
from customer_orders t
join json_table(
  replace(json_array(t.exclusions), ',', '","'),
  '$[*]' columns (exclusions varchar(50) path '$')
) j
join json_table(
  replace(json_array(t.extras), ',', '","'),
  '$[*]' columns (extras varchar(50) path '$')
) k);
```

order_id	customer_id	pizza_id	exclusion	extra	order_time
1	101	1	NULL	NULL	2020-01-01 18:05:02
2	101	1	NULL	NULL	2020-01-01 19:00:52
3	102	1	NULL	NULL	2020-01-02 23:51:23
3	102	2	NULL	NULL	2020-01-02 23:51:23
4	103	1	4	NULL	2020-01-04 13:23:46
4	103	1	4	NULL	2020-01-04 13:23:46
4	103	2	4	NULL	2020-01-04 13:23:46
5	104	1	NULL	1	2020-01-08 21:00:29
6	101	2	NULL	NULL	2020-01-08 21:03:13
7	105	2	NULL	1	2020-01-08 21:20:29
8	102	1	NULL	NULL	2020-01-09 23:54:33
9	103	1	4	1	2020-01-10 11:22:59
9	103	1	4	5	2020-01-10 11:22:59
10	104	1	NULL	NULL	2020-01-11 18:34:49
10	104	1	2	1	2020-01-11 18:34:49
10	104	1	2	4	2020-01-11 18:34:49
10	104	1	6	1	2020-01-11 18:34:49
10	104	1	6	4	2020-01-11 18:34:49

1. What are the standard ingredients for each pizza?

```
317 •   select pr.pizza_id, group_concat(pt.topping_name) as ingredients
318     from pizza_recipes_new pr join pizza_toppings pt
319     on pr.topping = pt.topping_id group by 1;
320
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

pizza_id	ingredients
1	Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami
2	Cheese,Mushrooms,Onions,Peppers,Tomatoes,Tomato Sauce

2. what was the most commonly added extra?

```
323 •   select topping_name as most_common_extra from (
324     select pt.topping_name, count(*)
325     from customer_order_new con join pizza_toppings pt
326     on con.extra = pt.topping_id
327     where extra is not null group by 1 order by 2 desc limit 1) t;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

most_common_extra
Bacon

3. What was the most common exclusion?

```
330 •   select topping_name as most_common_exclusion from pizza_toppings
331     where topping_id = ( select exclusion from (
332     select exclusion, count(*) from customer_order_new
333     where exclusion is not null group by exclusion order by 2 desc limit 1)t);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

most_common_exclusion
Cheese

4. Generate an order item for each record in the customers_orders table in the format of one of the following:

- Meat Lovers
- Meat Lovers - Exclude Beef
- Meat Lovers - Extra Bacon
- Meat Lovers - Exclude Cheese, Bacon - Extra Mushroom, Peppers

```sql
with exclusion_extras as (
select con.order_id, con.customer_id, con.pizza_id, pn.pizza_name,
group_concat(distinct pt.topping_name) as exclusions, group_concat(distinct pt2.topping_name) as extras
from customer_order_new con join pizza_names pn
on con.pizza_id=pn.pizza_id left join pizza_toppings pt
on con.exclusion = pt.topping_id left join pizza_toppings pt2 on con.extra = pt2.topping_id
group by 1,2,3,4)
select order_id, customer_id,
concat(pizza_name, ifnull(concat(' -excludes ',exclusions),""), ifnull(concat(' -extras ',extras),'')) as order_item
from exclusion_extras;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	customer_id	order_item
1	101	Meatlovers
2	101	Meatlovers
3	102	Meatlovers
3	102	Vegetarian
4	103	Meatlovers -excludes Cheese
4	103	Vegetarian -excludes Cheese
5	104	Meatlovers -extras Bacon
6	101	Vegetarian
7	105	Vegetarian -extras Bacon
8	102	Meatlovers
9	103	Meatlovers -excludes Cheese -extras Bacon,Chicken
10	104	Meatlovers -excludes BBQ Sauce,Mushrooms -extras Bacon,Cheese

5. What is the total quantity of each ingredient used in all delivered pizzas sorted by most frequent first?

```
369  select *,count(toppings) over(partition by toppings) as frequency from (
370  select order_id, pizza_id, (ingredients_count-ifnull(total_exclusion,0)+ifnull(total_extras,0)) as toppings from (
371  select c.order_id, duration, c.pizza_id,
372  length(exclusions) - length(replace(exclusions,',','')) +1 as total_exclusion,
373  length(extras) - length(replace(extras,',','')) + 1 as total_extras,
374  ingredients_count
375  from runner_orders r join customer_orders c on r.order_id = c.order_id
376  join
377  (select pizza_id,count(*) as ingredients_count from pizza_recipes_new group by pizza_id) t on t.pizza_id=c.pizza_id
378  where cancellation is null) t2) t3
379  order by frequency desc;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	pizza_id	toppings	frequency
1	1	8	6
2	1	8	6
3	1	8	6
8	1	8	6
10	1	8	6
10	1	8	6
4	1	7	3
4	1	7	3
7	2	7	3
4	2	5	1
3	2	6	1
5	1	9	1

INSIGHTS GATHERED

1. Bacon is the most preferred topping.
2. Cheese was the least preferred topping.
3. The maximum no of topping was 9
4. Topping count 8 was most frequent.

Pricing and Ratings

1. 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?

```
385 • select sum(costs) as TotalRevenue from(
386   select c.order_id, c.pizza_id, p.pizza_name,
387   case
388       when pizza_name='Meatlovers' then 12
389       else 10
390   end as costs
391   from customer_orders c
392   join runner_orders r on c.order_id = r.order_id
393   join  pizza_names p on c.pizza_id = p.pizza_id
394   where cancellation is null) t;
```

Result Grid | Filter Rows: | Export: | Wrap Cell

TotalRevenue
138

2. What if there was an additional $1 charge for any pizza extras?

```
398 • select sum(total_cost) as Total_Revenue from(
399   select *, ifnull(total_extras,'')+costs as total_cost from(
400   select c.order_id, c.pizza_id, length(extras) - length(replace(extras,',','')) + 1 as total_extras,
401   if(pizza_id=1, 12, 10) as costs
402   from customer_orders c
403   join runner_orders r on c.order_id = r.order_id
404   where cancellation is null) t
405   )t2;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Total_Revenue
142

3. 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.

```
411 •   DROP TABLE IF EXISTS runner_ratings;
412 •   CREATE TABLE runner_ratings (
413       order_id INTEGER,
414       ratings INTEGER
415     );
416 •   INSERT INTO runner_ratings(order_id, ratings)
417     VALUES
418       ('1', '3'),
419       ('2', '5'),
420       ('3', '3'),
421       ('4', '2'),
422       ('5', '3'),
423       ('7', '2'),
424       ('8', '4'),
425       ('10', '4');
426 •   select * from runner_ratings;
427
```

Result Grid | Filter Rows: | Export: | Wrap

order_id	ratings
1	3
2	5
3	3
4	2
5	3
7	2
8	4
10	4

4. Using your newly generated table - can you join all of the information together to form a table which has the following information for successful deliveries?

- customer_id
- order_id
- runner_id
- rating
- order_time
- pickup_time
- Time between order and pickup
- Delivery duration
- Average speed
- Total number of pizzas

```
440 •   select  distinct r.order_id, c.customer_id, r.runner_id, rr.ratings, c.order_time, r.pickup_time,
441     timestampdiff(minute, c.order_time, r.pickup_time) as time_btw_ordr_pikup, r.duration,
442     Round(Avg((distance * 60)/duration),1) as avg_speed, count(pizza_id) as pizza_count
443     from customer_orders c
444     right join runner_orders r on r.order_id = c.order_id
445     join runner_ratings rr on r.order_id = rr.order_id
446     group by 1,2,3,4,5,6,7,8;
447
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

order_id	customer_id	runner_id	ratings	order_time	pickup_time	time_btw_ordr_pikup	duration	avg_speed	pizza_count
1	101	1	3	2020-01-01 18:05:02	2020-01-01 18:15:34	10	32	37.5	1
2	101	1	5	2020-01-01 19:00:52	2020-01-01 19:10:54	10	27	44.4	1
3	102	1	3	2020-01-02 23:51:23	2020-01-03 00:12:37	21	20	40.2	2
4	103	2	2	2020-01-04 13:23:46	2020-01-04 13:53:03	29	40	35.1	3
5	104	3	3	2020-01-08 21:00:29	2020-01-08 21:10:57	10	15	40	1
7	105	2	2	2020-01-08 21:20:29	2020-01-08 21:30:45	10	25	60	1
8	102	2	4	2020-01-09 23:54:33	2020-01-10 00:15:02	20	15	93.6	1
10	104	1	4	2020-01-11 18:34:49	2020-01-11 18:50:20	15	10	60	2

5. 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?

```
451 • select sum(total_amount-delivery_fee) as Total_Revenue from (
452   select r.order_id, distance, round(distance* 0.30) as delivery_fee, sum(costs) as total_amount from(
453   select c.order_id, c.pizza_id,
454   case
455       when c.pizza_id='1' then 12
456       else 10
457   end as costs
458   from customer_orders c
459   join runner_orders r on c.order_id = r.order_id
460   where cancellation is null) t join runner_orders r on t.order_id = r.order_id
461   group by 1,2,3) t2;
462
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Total_Revenue
94

INSIGHTS GATHERED

1. Total Revenue without any delivery charge and extra charge = $138
2. Revenue to be generated with charge on extras will be $142
3. Revenue generated after paying delivery charges to the runners = $94