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!

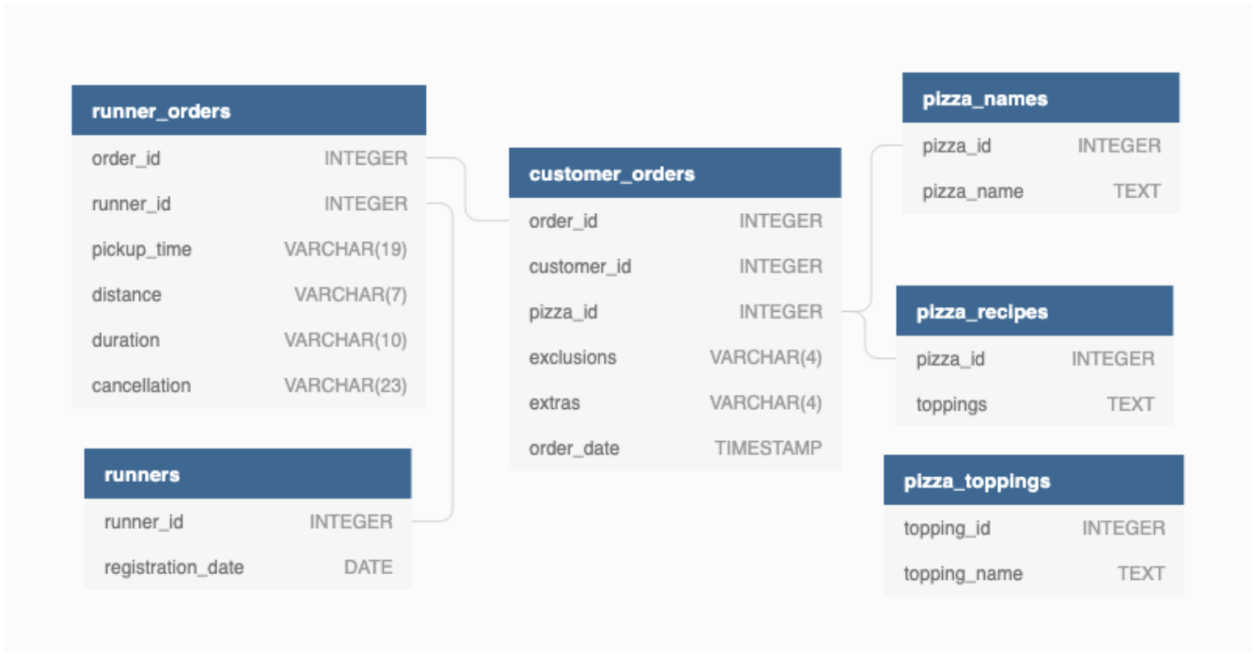
Danny started by recruiting “runners” to deliver fresh pizza from Pizza Runner Headquarters (otherwise known as Danny’s house) and also maxed out his credit card to pay freelance developers to build a mobile app to accept orders from customers.

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

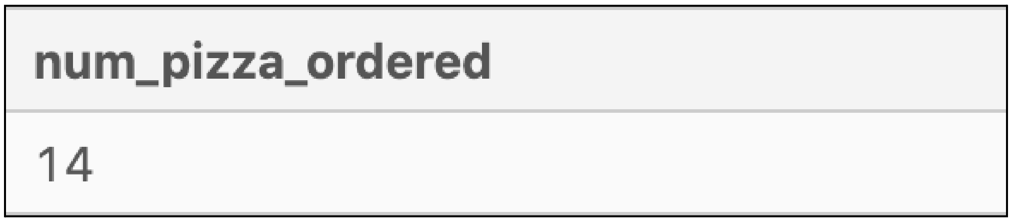


Case Study Questions:

A. Pizza Metrics

1. How many pizzas were ordered?

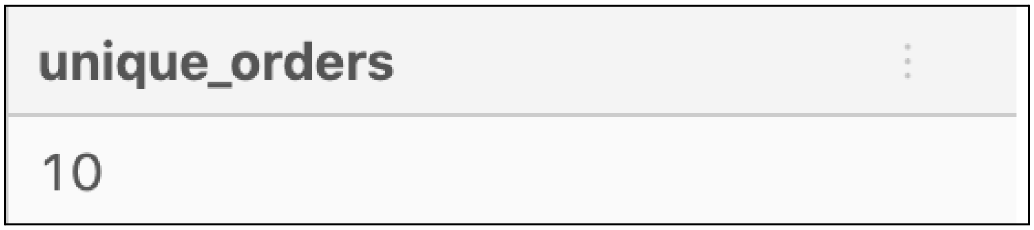
```
select count(*)  as num_pizza_ordered
from pizza_runner.customer_orders;
```



num_pizza_ordered
14

2. How many unique customer orders were made?

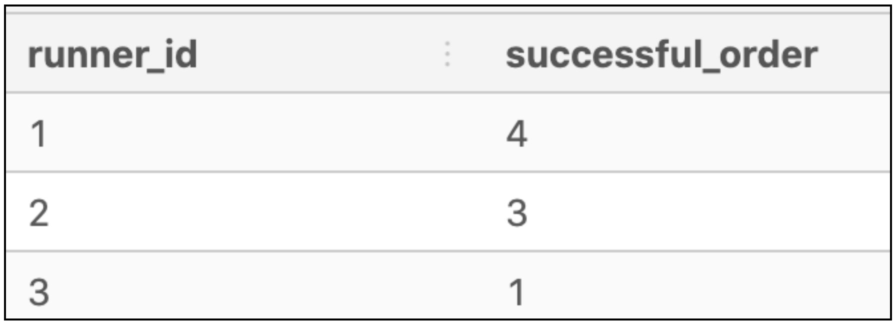
```
select count(distinct order_id) as unique_orders
from pizza_runner.customer_orders;
```



unique_orders
10

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

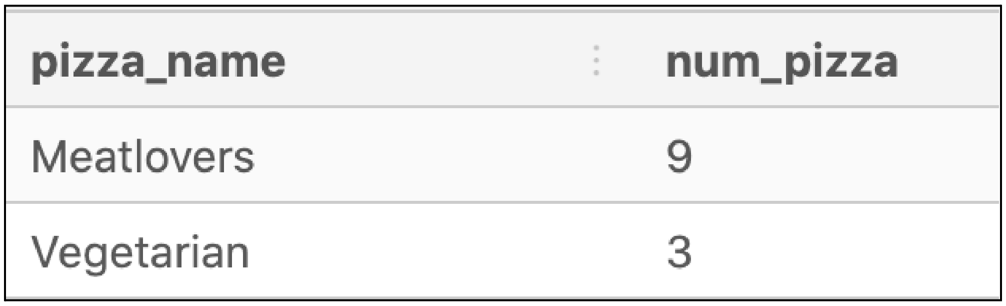
```
select runner_id, count(order_id) as successful_order
from pizza_runner.runner_orders
where pickup_time!='null'
group by runner_id
order by runner_id;
```



runner_id	successful_order
1	4
2	3
3	1

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

```
select pizza_name, count(*) as num_pizza
from pizza_runner.customer_orders o
inner join pizza_runner.runner_orders ro on o.order_id=ro.order_id
inner join pizza_runner.pizza_names pn on pn.pizza_id=o.pizza_id
where pickup_time !='null'
group by pizza_name;
```

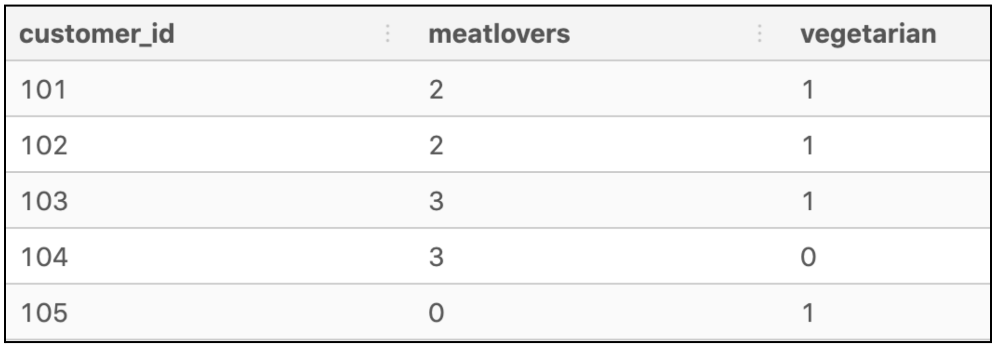


pizza_name	num_pizza
Meatlovers	9
Vegetarian	3

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

```
with cte_meatlovers as (
        select o.customer_id as customer_id,
              count(n.pizza_name) as meatlovers
        from pizza_runner.customer_orders o
        join pizza_runner.pizza_names n on o.pizza_id=n.pizza_id
        where n.pizza_name='Meatlovers'
        group by o.customer_id),
cte_vegetarian as (
        select o.customer_id as customer_id,
              count(n.pizza_name) as vegetarian
        from pizza_runner.customer_orders o
        join pizza_runner.pizza_names n on o.pizza_id=n.pizza_id
        where n.pizza_name='Vegetarian'
```

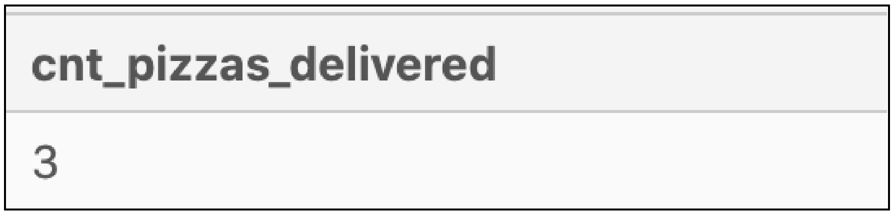
```
            group by o.customer_id)
select m.customer_id,
      coalesce(meatlovers,0) as meatlovers,
      coalesce(vegetarian,0) as vegetarian
from cte_meatlovers m
left join cte_vegetarian  v on m.customer_id=v.customer_id
union
select v.customer_id,
      coalesce(meatlovers,0) as meatlovers ,
      coalesce(vegetarian,0) as vegetarian
from cte_meatlovers m
right join cte_vegetarian  v on m.customer_id=v.customer_id
order by customer_id;
```



customer_id	meatlovers	vegetarian
101	2	1
102	2	1
103	3	1
104	3	0
105	0	1

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

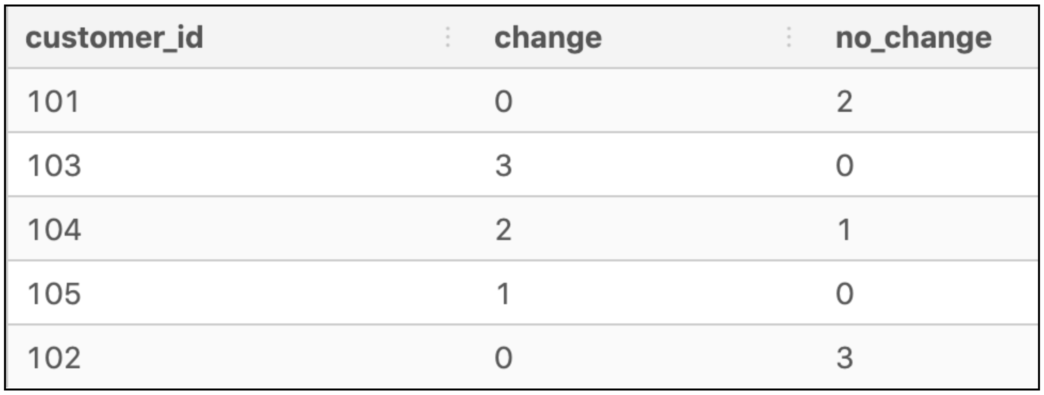
```
with cte_rank_delivered_pizzas as (
            select order_id,
                  cnt_pizzas_delivered,
                              rank() over(order by cnt_pizzas_delivered desc) as rnk
           from (
                    select o.order_id,count(pizza_id) as cnt_pizzas_delivered
                    from pizza_runner.customer_orders o
                    inner join pizza_runner.runner_orders ro on ro.order_id=o.order_id
                    where pickup_time !='null'
                    group by o.order_id)x)
select distinct cnt_pizzas_delivered
from cte_rank_delivered_pizzas where rnk=1;
```



cnt_pizzas_delivered
3

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

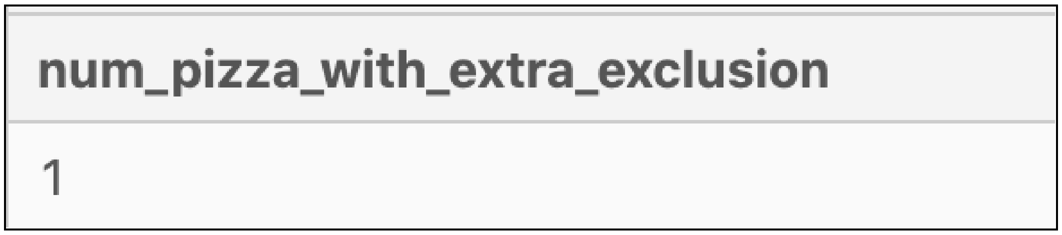
```
with cte as (
        select o.order_id,
            customer_id,
            pizza_id,
            case when exclusions = 'null' or exclusions is null or exclusions = ''
                then '0' else exclusions end as exclusions,
            case when extras = 'null' or extras is null or extras = ''
                then '0' else extras end as extras
        from pizza_runner.customer_orders o
        join pizza_runner.runner_orders ro  on o.order_id=ro.order_id
        where pickup_time != 'null')
select customer_id,
    sum(change) as change ,
    sum(no_change) as no_change
from (
       select customer_id,
            exclusions,
            extras,
            case when exclusions !='0' or extras !='0'
                then 1 else 0 end as change,
            case when exclusions ='0' and extras ='0'
                then 1 else 0 end as no_change
        from cte) x
group by customer_id;
```



customer_id	change	no_change
101	0	2
103	3	0
104	2	1
105	1	0
102	0	3

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

```
with cte_deliver as (
        select o.order_id,
            customer_id,
            pizza_id,
            case when exclusions = 'null' or exclusions is null or exclusions =''
                then '0' else exclusions end as exclusions,
            case when extras = 'null' or extras is null or extras = ''
                then '0' else extras end as extras
        from pizza_runner.customer_orders o
        join pizza_runner.runner_orders ro on o.order_id=ro.order_id
        where pickup_time != 'null')
select count(*) as num_pizza_with_extra_exclusion
from cte_deliver
where exclusions !='0' and extras !='0';
```

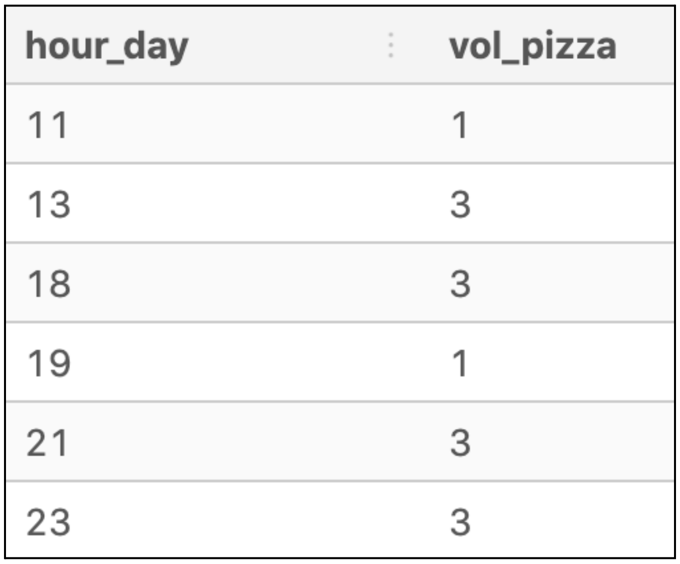


num_pizza_with_extra_exclusion
1

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

```
select hour_day,
    count(pizza_id) as vol_pizza
from (
        select order_id,
            pizza_id,
            extract(hour from order_time) as hour_day
        from pizza_runner.customer_orders o
```

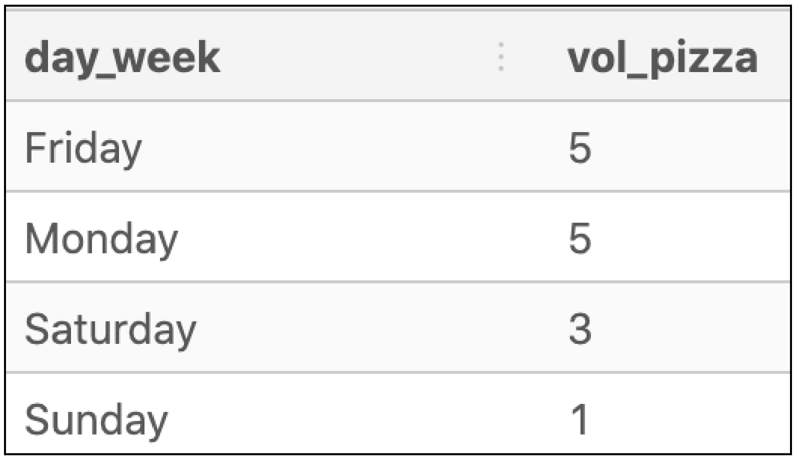
```
            order by order_id,pizza_id)x
group by hour_day
order by hour_day;
```



hour_day	vol_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?

```
select day_week,
       count(pizza_id) as vol_pizza
from (
            select order_id,
                   pizza_id,
                   to_char(order_time,'Day') as day_week
            from pizza_runner.customer_orders)x
group by day_week
order by day_week;
```

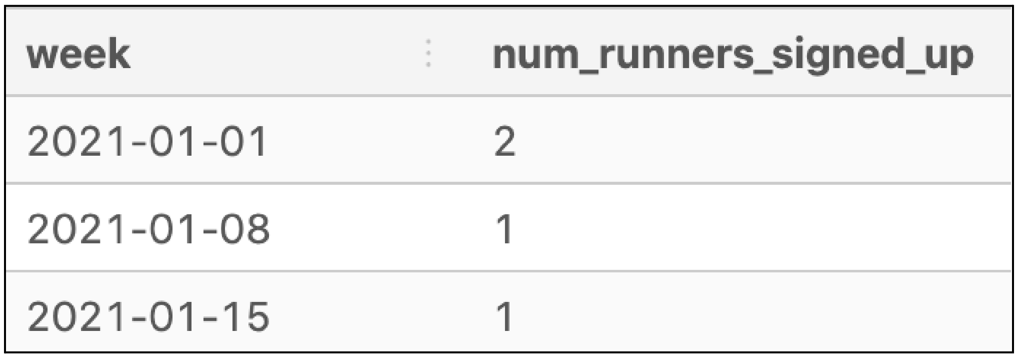


day_week	vol_pizza
Friday	5
Monday	5
Saturday	3
Sunday	1

B. <u>Runner and Customer Experience</u>

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

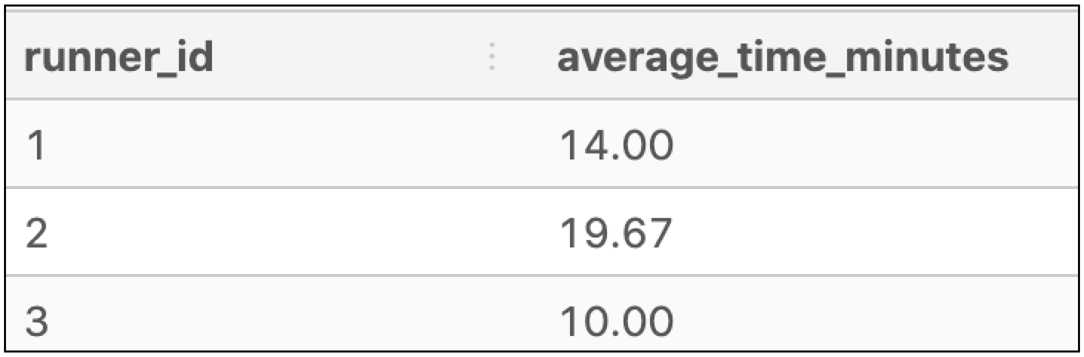
```
select cast('2021-01-01' as date)+week * interval '7days' as Week ,
    count(runner_id) as num_runners_signed_up
from (
        select runner_id,
            (registration_date::date-cast('2021-01-01' as date))::integer/7 as
        week
        from pizza_runner.runners)x
group by cast('2021-01-01' as date)+week * interval '7days'
order by week;
```



week	num_runners_signed_up
2021-01-01	2
2021-01-08	1
2021-01-15	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?

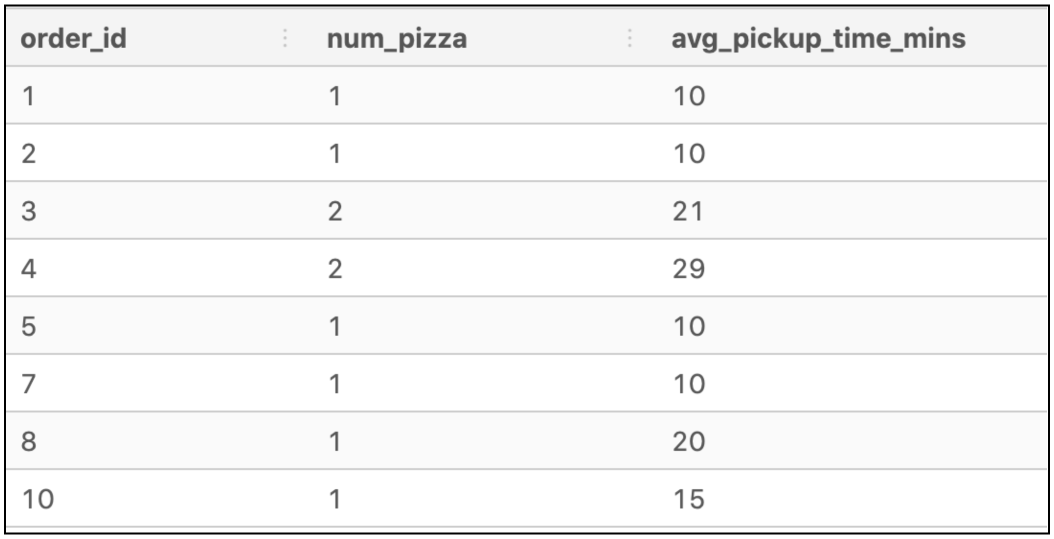
```
with cte as (
         select distinct runner_id,ro.order_id,pickup_time,order_time
         from pizza_runner.runner_orders ro
         join pizza_runner.customer_orders co on ro.order_id=co.order_id
         where pickup_time != 'null')
select runner_id,
        round(avg(EXTRACT(MINUTE FROM (cast(pickup_time as timestamp) -
        cast(order_time as timestamp))))::decimal,2) as average_time_minutes
from cte
group by runner_id;
```



runner_id	average_time_minutes
1	14.00
2	19.67
3	10.00

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

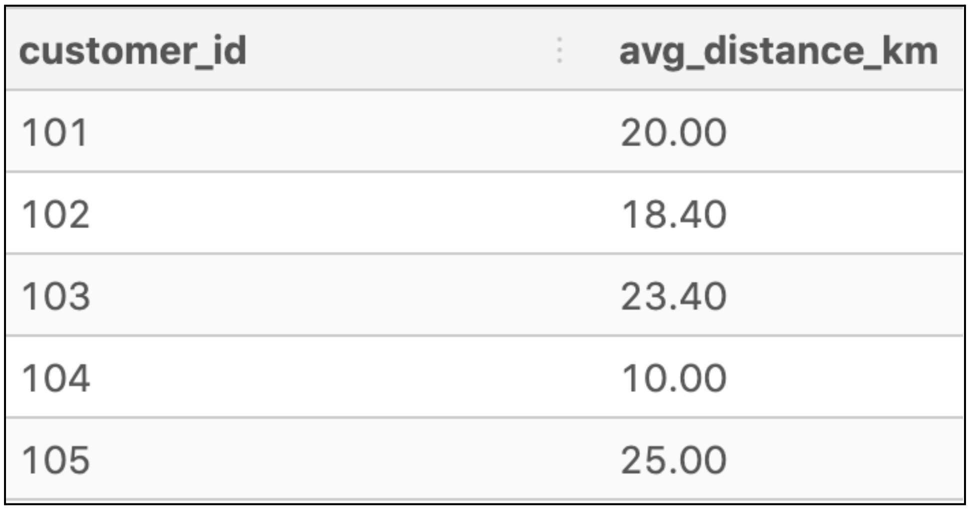
```
with cte as (
            select distinct runner_id,
                  ro.order_id,
                  pickup_time,
                  order_time,
                  pizza_id,
                  DATE_PART('minutes',
          AGE(pickup_time::TIMESTAMP,order_time))::INTEGER as time_taken
            from pizza_runner.runner_orders ro
            join pizza_runner.customer_orders co on ro.order_id=co.order_id
            where pickup_time != 'null')
select order_id,
             count(pizza_id) as num_pizza,
             avg(time_taken) as avg_pickup_time_mins
from cte
group by order_id
order by order_id,avg_pickup_time_mins;
```



order_id	num_pizza	avg_pickup_time_mins
1	1	10
2	1	10
3	2	21
4	2	29
5	1	10
7	1	10
8	1	20
10	1	15

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

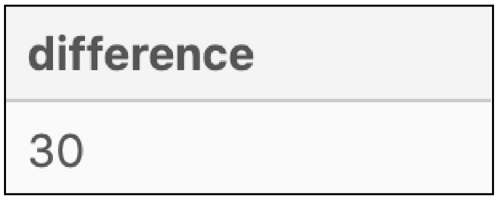
```
with cte as (
        select co.order_id,
            customer_id,trim('km' from distance)::float as distance
        from pizza_runner.customer_orders co
        inner join pizza_runner.runner_orders ro on ro.order_id=co.order_id
        where duration  != 'null'
        group by co.order_id,customer_id, distance)
select customer_id,
    round(avg(distance)::numeric,2) as avg_distance
from cte
group by customer_id
order by customer_id;
```



customer_id	avg_distance_km
101	20.00
102	18.40
103	23.40
104	10.00
105	25.00

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

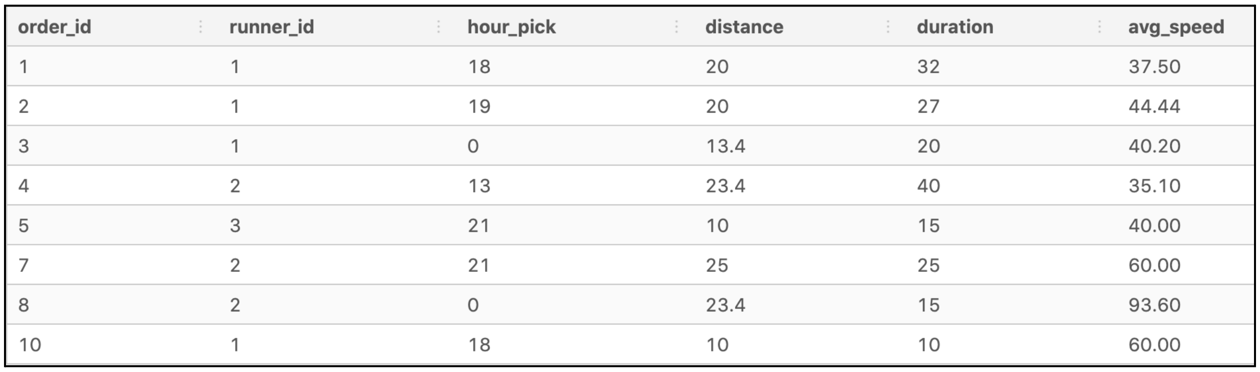
```
select max(duration)-min(duration) as difference
from (
        select trim('minutes' from duration)::numeric as duration
        from pizza_runner.runner_orders
        where duration  != 'null')x
```



difference
30

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

```
with cte as (
         select co.order_id,
              runner_id,
              extract('hour' from pickup_time::timestamp) as hour_pick,
              trim('km' from distance)::numeric as distance,
              trim('minutes' from duration)::numeric as duration
         from pizza_runner.customer_orders co
         inner join pizza_runner.runner_orders ro on co.order_id=ro.order_id
         where  pickup_time  != 'null'
         group by 1,2,3,4,5)
select *, round(distance*60/duration,2) as avg_speed
from cte
order by order_id;
```

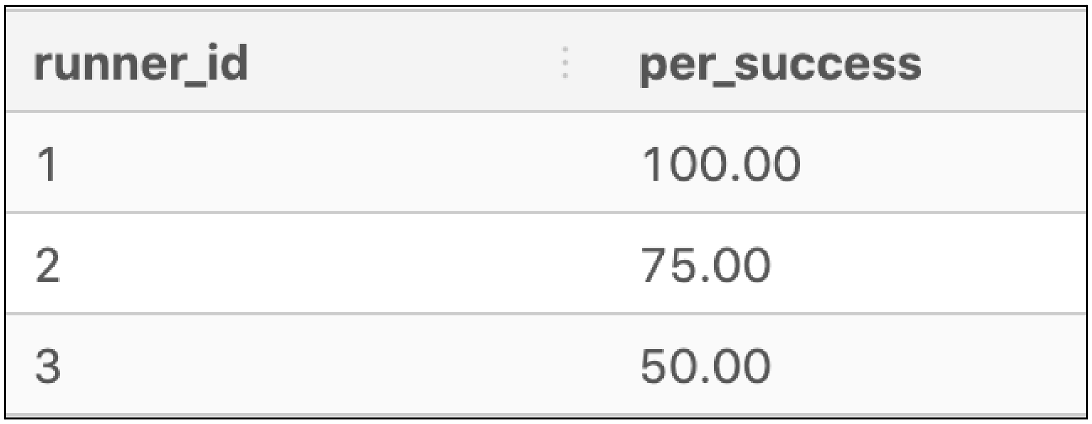


order_id	runner_id	hour_pick	distance	duration	avg_speed
1	1	18	20	32	37.50
2	1	19	20	27	44.44
3	1	0	13.4	20	40.20
4	2	13	23.4	40	35.10
5	3	21	10	15	40.00
7	2	21	25	25	60.00
8	2	0	23.4	15	93.60
10	1	18	10	10	60.00

**** As the hour of pick-up moves towards midnight, the average speed increases**

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

```
with cte as (
        select runner_id,
            sum(case when cancellation in ('Restaurant Cancellation',
              'Customer Cancellation') then 0 else 1
            end) as successful_deliveries,
            count(*) as all_deliveries
        from pizza_runner.runner_orders
        group by runner_id)
select runner_id,
    round(1.0 *successful_deliveries/all_deliveries*100,2) as per_success
from cte;
```



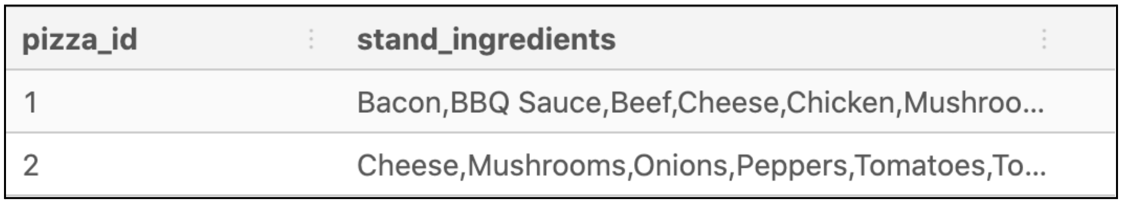
runner_id	per_success
1	100.00
2	75.00
3	50.00

C. <u>**Ingredient Optimisation**</u>

1. What are the standard ingredients for each pizza?

```
with cte as (
        select pizza_id,
            topping_name from pizza_runner.pizza_toppings t
        inner join (
              select pizza_id,
                    unnest(string_to_array(toppings, ', '))::integer as
                    toppings_list
              from pizza_runner.pizza_recipes )x on
       x.toppings_list=t.topping_id)
select pizza_id, string_agg(distinct topping_name,',') as stand_ingredients
from cte
```

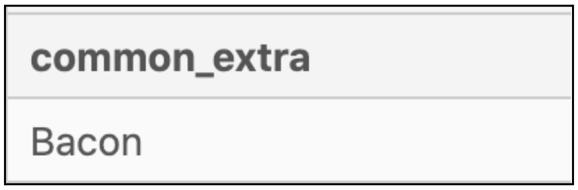
```
group by pizza_id;
```



pizza_id	stand_ingredients
1	Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushroo...
2	Cheese,Mushrooms,Onions,Peppers,Tomatoes,To...

2. **What was the most commonly added extra?**

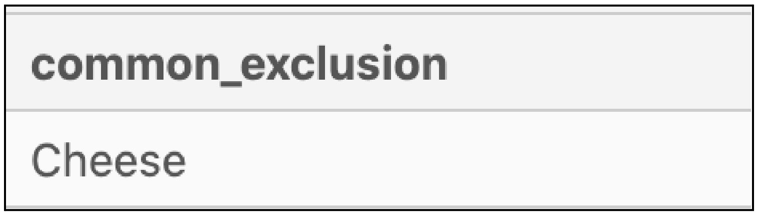
```
with cte as (
        select extras
        from (
                select order_id,unnest(string_to_array(extras, ', ')) as extras
                from (
                        select order_id,
                            case when extras = 'null' or extras is null
                                  then '' else extras end as extras
                        from pizza_runner.customer_orders o
                        order by customer_id,pizza_id)x
                where extras is not null )y
         group by extras
         order by count(*) desc)
select topping_name as common_extra
from cte
join pizza_runner.pizza_toppings t
on cte.extras::integer=t.topping_id
limit 1;
```



common_extra
Bacon

3. **What was the most common exclusion?**

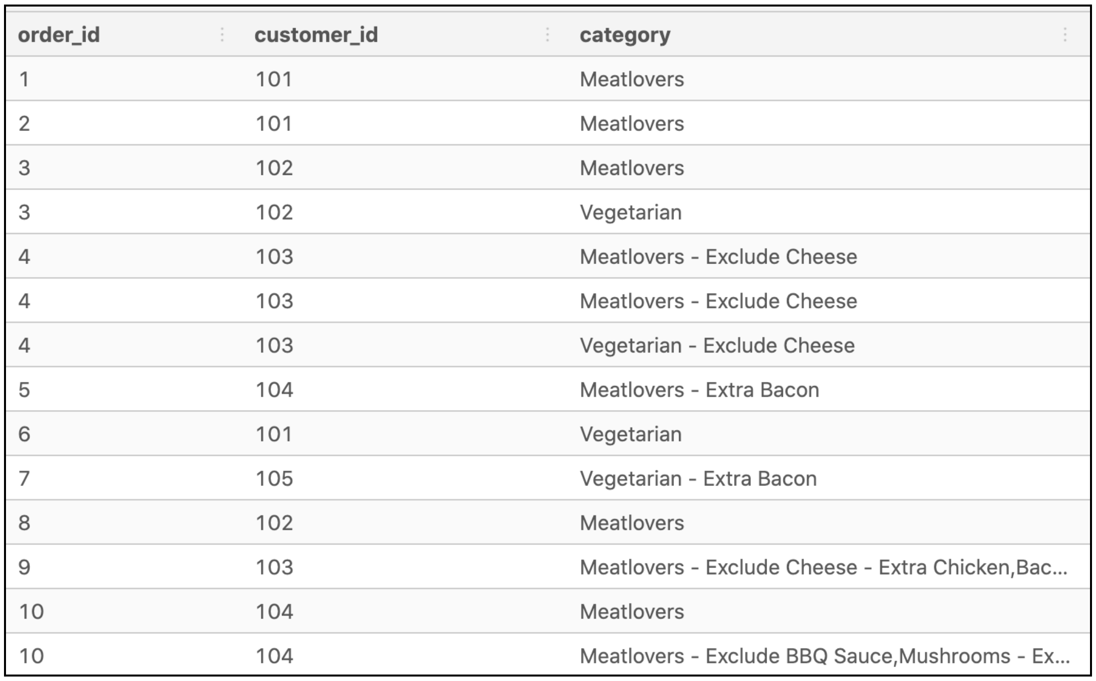
```
with cte as (
        select exclusions,
            count(*) as cnt
        from (
                select order_id,
                    unnest(string_to_array(exclusions, ', ')) as exclusions
                from (
                                select order_id,
                                       case when exclusions='null' or exclusions is null
                                                then '' else exclusions end as exclusions
                                from pizza_runner.customer_orders o
                                order by customer_id,pizza_id)x
                        where exclusions is not null )y
                group by exclusions
                order by count(*) desc)
select topping_name  as common_exclusion
from cte
join pizza_runner.pizza_toppings t on cte.exclusions::integer=t.topping_id
order by cnt desc
limit 1;
```



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

```
with cte_main as (
        select rownum,
               order_id,
               customer_id,
               pizza_id,
               exclusions,
               unnest(string_to_array(extras_1, ', ')) as extras
        from(
                select rownum,
                       order_id,
                       customer_id,
                       pizza_id,
                       unnest(string_to_array(exclusions_1, ', ')) as
                   exclusions,extras_1
                from (
                        select *,
                                              row_number() over(order by
                                             order_id,customer_id,pizza_id ) as rownum,
                                              case when exclusions='null' or exclusions is null or
                                             exclusions ='' then '0' else exclusions end as
                                             exclusions_1,
                                              case when extras = 'null' or extras is null  or
                                              extras = '' then '0' else extras end as extras_1
                                        from pizza_runner.customer_orders o
                                        order by customer_id,pizza_id)x)y ),
cte_exclusion as (
            select rownum,
                  order_id,
                  customer_id,
                  pizza_id,
                  exclusions,
                  extras,
                  topping_name as exclusions_name
            from pizza_runner.pizza_toppings t
            right join cte_main z on z.exclusions::integer=t.topping_id),
cte_extras as (
            select rownum,
                  order_id,
                  customer_id,
                  pizza_id,
                  exclusions,
                  extras,
                  exclusions_name,
                  topping_name as extras_name
            from pizza_runner.pizza_toppings t
            right join cte_exclusion z on z.extras::integer=t.topping_id),
cte_pizza as (
            select rownum,
                  order_id,
                  customer_id,
                  e.pizza_id,
                  exclusions_name,
                  extras_name,
                  pizza_name
            from cte_extras e join pizza_runner.pizza_names pzn on
          e.pizza_id=pzn.pizza_id),
cte_agg_extras as (
            select rownum,
                  order_id,
                  customer_id,
            pizza_name,
            exclusions_name,
            string_agg(extras_name,',') as extras_name
        from cte_pizza
        group by rownum,order_id,customer_id,pizza_name,exclusions_name),
cte_agg_exclusions as (
        select rownum,
            order_id,
            customer_id,
            pizza_name,
            extras_name,
            string_agg(exclusions_name,',') as exclusions_name
        from cte_agg_extras
        group by rownum,order_id,customer_id,pizza_name,extras_name)
select order_id,
    customer_id,
    case when extras_name is not null and exclusions_name is not null
    then concat(pizza_name,' ','-',' ','Exclude',' ',exclusions_name,' ','-',' ','Extra','
',extras_name)
    when exclusions_name is not null then concat(pizza_name,' ','-',' ','Exclude','
',exclusions_name)
    when extras_name is not null then concat(pizza_name,' ','-',' ','Extra','
',extras_name)
    else pizza_name end as category
from cte_agg_exclusions;
```



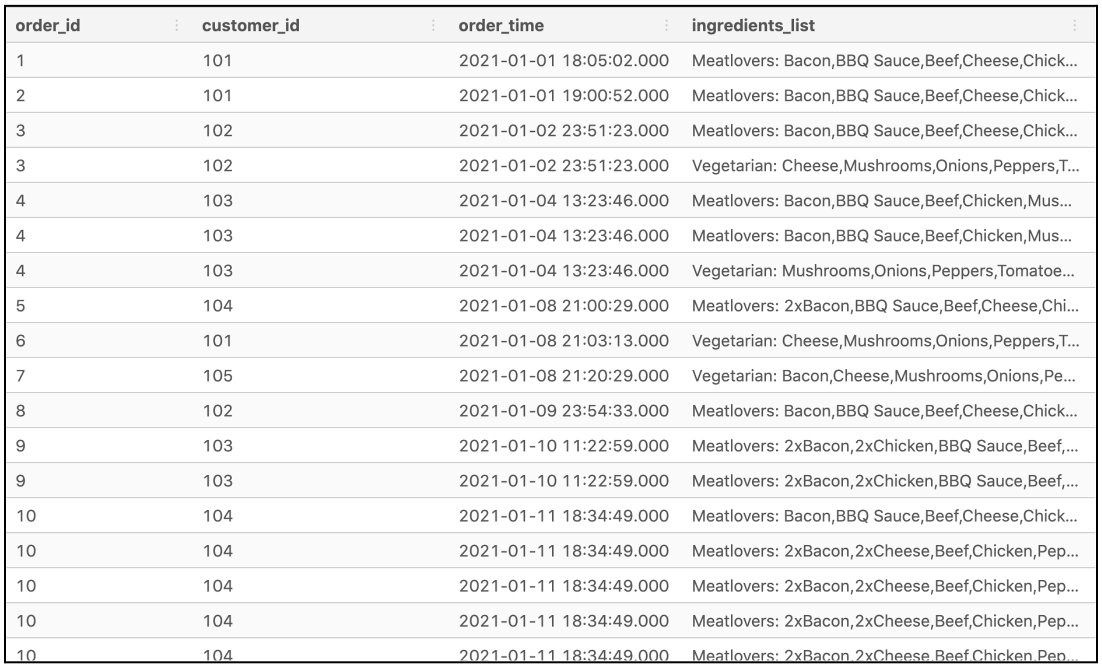
order_id	customer_id	category
1	101	Meatlovers
2	101	Meatlovers
3	102	Meatlovers
3	102	Vegetarian
4	103	Meatlovers - Exclude Cheese
4	103	Meatlovers - Exclude Cheese
4	103	Vegetarian - Exclude Cheese
5	104	Meatlovers - Extra Bacon
6	101	Vegetarian
7	105	Vegetarian - Extra Bacon
8	102	Meatlovers
9	103	Meatlovers - Exclude Cheese - Extra Chicken,Bac...
10	104	Meatlovers
10	104	Meatlovers - Exclude BBQ Sauce,Mushrooms - Ex...

5. Generate an alphabetically ordered comma-separated ingredient list for each pizza order from the customer_orders table and add a 2x in front of any relevant ingredients
-For example: "Meat Lovers: 2xBacon, Beef, ... , Salami"

```
with cte_main as (
       select
         rownum,
         order_time,
         order_id,
         customer_id,
         pizza_id,
         exclusions,
         unnest(
           string_to_array(extras_1, ', ')
         ) as extras
       from
         (
           select
             rownum,
             order_time,
             order_id,
             customer_id,
             pizza_id,
             unnest(
               string_to_array(exclusions_1, ', ')
             ) as exclusions,
             extras_1
           from
             (
               select
                 *,
                 row_number() over(
                   order by
                     order_id,
                     customer_id,
                     pizza_id
                 ) as rownum,
                 case when exclusions = 'null'
                 or exclusions is null
                 or exclusions = '' then '0' else exclusions end as exclusions_1,
                 case when extras = 'null'
                 or extras is null
```

```
            or extras = '' then '0' else extras end as extras_1
          from
            pizza_runner.customer_orders o
          order by
            customer_id,
            pizza_id
        ) x
    ) y
),
cte_exclusion as (
  select
    rownum,
    order_id,
    customer_id,
    pizza_id,
    exclusions,
    extras,
    topping_name as exclusions_name
  from
    pizza_runner.pizza_toppings t
    right join cte_main z on z.exclusions :: integer = t.topping_id
),
cte_extras as (
  select
    rownum,
    order_id,
    customer_id,
    pizza_id,
    exclusions,
    extras,
    exclusions_name,
    topping_name as extras_name
  from
    pizza_runner.pizza_toppings t
    right join cte_exclusion z on z.extras :: integer = t.topping_id
),
cte_pizza as (
  select
    rownum,
    order_id,
    customer_id,
    e.pizza_id,
    exclusions_name,
    extras_name,
    pizza_name
  from
    cte_extras e
    join pizza_runner.pizza_names pzn on e.pizza_id = pzn.pizza_id
),
cte_pizza_recipes as (
  select
    rownum,
    order_id,
    customer_id,
    pizza_name,
    exclusions_name,
    extras_name,
    unnest(
      string_to_array(pr.toppings, ',')
    ) as toppings
  from
    cte_pizza p
    join pizza_runner.pizza_recipes pr on pr.pizza_id = p.pizza_id
),
cte_pizza_recipes_toppings as (
  select
    rownum,
    order_id,
    customer_id,
    pizza_name,
    exclusions_name,
    extras_name,
    t.topping_name
  from
    cte_pizza_recipes r
    join pizza_runner.pizza_toppings t on t.topping_id = r.toppings :: integer
),
cte_pizza_excl as (
  select
    rownum,
    order_id,
    customer_id,
    exclusions_name,
    pizza_name
  from
    cte_pizza
  where
    exclusions_name is not null
  group by
    rownum,
    order_id,
    customer_id,
    exclusions_name,
    pizza_name
),
cte_pizza_wo_excl_extras as (
  select
    rownum,
    order_id,
    customer_id,
    pizza_name,
    topping_name
  from
    cte_pizza_recipes_toppings x1
  where
    topping_name not in (
      select
        exclusions_name
      from
        cte_pizza_excl x2
      where
        x1.rownum = x2.rownum
        and x1.order_id = x2.order_id
        and x1.customer_id = x2.customer_id
        and x1.pizza_name = x2.pizza_name
    )
  order by
    order_id,
    rownum
),
cte_pizza_extra as (
  select
    rownum,
    order_id,
    customer_id,
    pizza_name,
    extras_name as topping_name
  from
    cte_pizza
  where
    extras_name is not null
  group by
```

```
    rownum,
    order_id,
    customer_id,
    extras_name,
    pizza_name
),
cte_pizza_toppings as (
  select
    rownum,
    order_id,
    customer_id,
    pizza_name,
    topping_name
  from
    cte_pizza_wo_excl_extras
  group by
    rownum,
    order_id,
    customer_id,
    pizza_name,
    topping_name
),
cte_relevant_toppings as (
  select
    rownum,
    order_id,
    customer_id,
    pizza_name,
    topping_name
  from
    cte_pizza_extra
  union all
  select
    rownum,
    order_id,
    customer_id,
    pizza_name,
    topping_name
  from
    cte_pizza_toppings
  order by
    rownum,
    order_id
),
cte_topping_count as (
  select
    rownum,
    order_id,
    customer_id,
    pizza_name,
    case when count(*)> 1 then concat(
      count(*),
      'x',
      topping_name
    ) else topping_name end as topping_name
  from
    cte_relevant_toppings
  group by
    rownum,
    order_id,
    customer_id,
    pizza_name,
    topping_name
  order by
    rownum,
    order_id,
    customer_id,
    pizza_name,
    topping_name
),
cte_all_toppings as (
  select
    rownum,
    order_id,
    customer_id,
    pizza_name,
    string_agg(topping_name, ',') as topping_name
  from
    cte_topping_count
  group by
    rownum,
    order_id,
    customer_id,
    pizza_name
)
select
  x1.order_id,
  x1.customer_id,
  order_time,
  concat(
    pizza_name, ':', ' ', topping_name
  ) as ingredients_list
from
  cte_all_toppings x1
  join cte_main x2 on x1.rownum = x2.rownum;
```

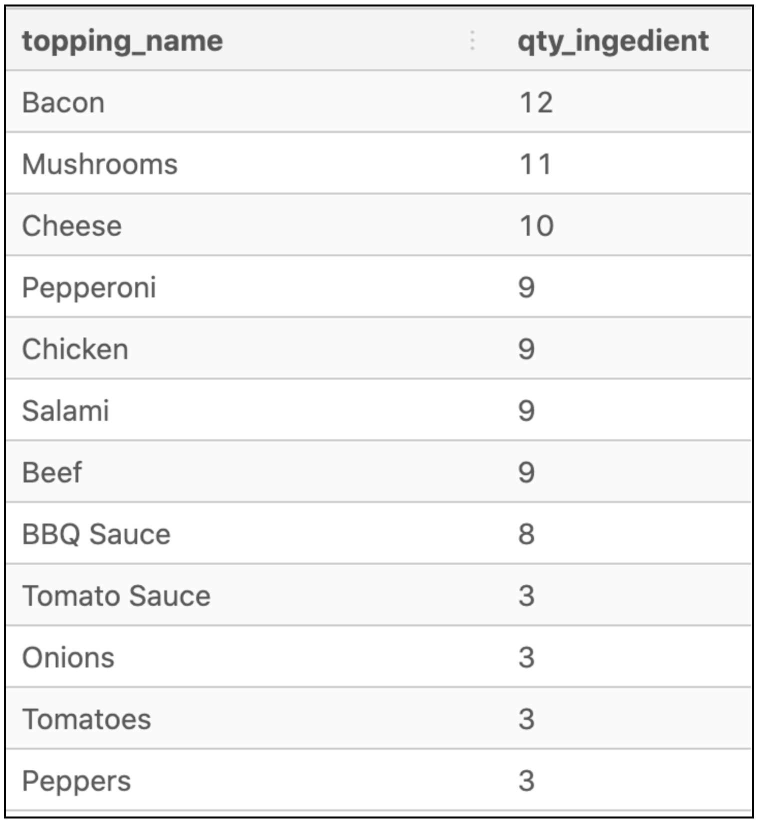


order_id	customer_id	order_time	ingredients_list
1	101	2021-01-01 18:05:02.000	Meatlovers: Bacon,BBQ Sauce,Beef,Cheese,Chick...
2	101	2021-01-01 19:00:52.000	Meatlovers: Bacon,BBQ Sauce,Beef,Cheese,Chick...
3	102	2021-01-02 23:51:23.000	Meatlovers: Bacon,BBQ Sauce,Beef,Cheese,Chick...
3	102	2021-01-02 23:51:23.000	Vegetarian: Cheese,Mushrooms,Onions,Peppers,T...
4	103	2021-01-04 13:23:46.000	Meatlovers: Bacon,BBQ Sauce,Beef,Chicken,Mus...
4	103	2021-01-04 13:23:46.000	Meatlovers: Bacon,BBQ Sauce,Beef,Chicken,Mus...
4	103	2021-01-04 13:23:46.000	Vegetarian: Mushrooms,Onions,Peppers,Tomatoe...
5	104	2021-01-08 21:00:29.000	Meatlovers: 2xBacon,BBQ Sauce,Beef,Cheese,Chi...
6	101	2021-01-08 21:03:13.000	Vegetarian: Cheese,Mushrooms,Onions,Peppers,T...
7	105	2021-01-08 21:20:29.000	Vegetarian: Bacon,Cheese,Mushrooms,Onions,Pe...
8	102	2021-01-09 23:54:33.000	Meatlovers: Bacon,BBQ Sauce,Beef,Cheese,Chick...
9	103	2021-01-10 11:22:59.000	Meatlovers: 2xBacon,2xChicken,BBQ Sauce,Beef,...
9	103	2021-01-10 11:22:59.000	Meatlovers: 2xBacon,2xChicken,BBQ Sauce,Beef,...
10	104	2021-01-11 18:34:49.000	Meatlovers: Bacon,BBQ Sauce,Beef,Cheese,Chick...
10	104	2021-01-11 18:34:49.000	Meatlovers: 2xBacon,2xCheese,Beef,Chicken,Pep...
10	104	2021-01-11 18:34:49.000	Meatlovers: 2xBacon,2xCheese,Beef,Chicken,Pep...
10	104	2021-01-11 18:34:49.000	Meatlovers: 2xBacon,2xCheese,Beef,Chicken,Pep...
10	104	2021-01-11 18:34:49.000	Meatlovers: 2xBacon,2xCheese,Beef,Chicken,Pep...

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

```
with cte_main as (
  select
    rownum,
    order_id,
    pizza_id,
    exclusions,
    unnest(
      string_to_array(extras_1, ', ')
    ) as extras
  from
    (
      select
        rownum,
        order_id,
        pizza_id,
        unnest(
          string_to_array(exclusions_1, ', ')
        ) as exclusions,
        extras_1
      from
        (
          select
            o.order_id,
            pizza_id,
            row_number() over(
              order by
                o.order_id,
                customer_id,
                pizza_id
            ) as rownum,
            case when exclusions = 'null'
            or exclusions is null
            or exclusions = '' then '0' else exclusions end as exclusions_1,
            case when extras = 'null'
            or extras is null
            or extras = '' then '0' else extras end as extras_1
          from
            pizza_runner.customer_orders o
            join pizza_runner.runner_orders ro on o.order_id = ro.order_id
          where
            pickup_time != 'null'
          order by
            customer_id,
            pizza_id
        ) x
    ) y
),
cte_exclusion as (
  select
    rownum,
    order_id,
    pizza_id,
    exclusions
  from
    cte_main
  group by
    1,2,3,4
),
cte_extras as (
  select
    rownum,
    order_id,
    pizza_id,
    extras
  from
    cte_main
  group by
    1,2,3,4
),
cte_toppings as (
  select
    rownum,
    order_id,
    m.pizza_id,
    unnest(
      string_to_array(r.toppings, ', ')
    ) as toppings
  from
    cte_main m
    inner join pizza_runner.pizza_recipes r on r.pizza_id = m.pizza_id
  group by
    1,2,3,4
),
cte_toppings_wo_exclusion as (
  select
    rownum,
    order_id,
    pizza_id,
    toppings
  from
    cte_toppings t
  where
    not exists (
      select
        1
      from
        cte_exclusion e
      where
        t.rownum = e.rownum
        and t.order_id = e.order_id
        and t.pizza_id = e.pizza_id
        and t.toppings = e.exclusions
    )
```

```
),
cte_all_toppings as (
  select
    rownum,
    order_id,
    pizza_id,
    toppings
  from
    cte_toppings_wo_exclusion
  union all
  select
    rownum,
    order_id,
    pizza_id,
    extras as toppings
  from
    cte_extras
  where
    extras != '0'
)
select
  topping_name,
  count(*) as qty_ingedient
from
  cte_all_toppings ct
  join pizza_runner.pizza_toppings pt on pt.topping_id = ct.toppings ::
integer
group by
  topping_name
order by
  count(*) desc;
```



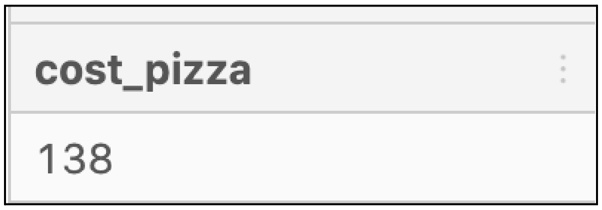
topping_name	qty_ingedient
Bacon	12
Mushrooms	11
Cheese	10
Pepperoni	9
Chicken	9
Salami	9
Beef	9
BBQ Sauce	8
Tomato Sauce	3
Onions	3
Tomatoes	3
Peppers	3

D. <u>Pricing and Ratings</u>

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?

```
with cte_main as (
        select order_id,
            customer_id,
            pizza_name,
            row_number() over(order by order_id,customer_id,o.pizza_id) as
       rnum
        from pizza_runner.customer_orders o
        inner join pizza_runner.pizza_names pz on o.pizza_id=pz.pizza_id),
       cte_delivered_orders as (
        select order_id,
            customer_id,
            pizza_name,
            rnum
        from cte_main where exists(
         select 1 from pizza_runner.runner_orders
```

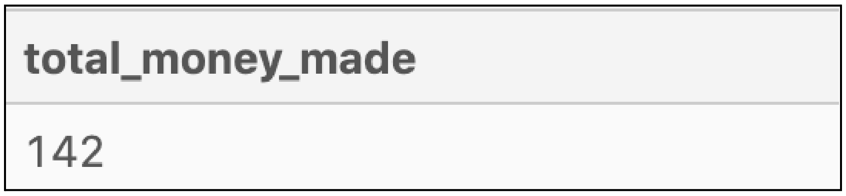
```
    where cte_main.order_id = runner_orders.order_id
    AND runner_orders.pickup_time != 'null'))
select sum(case when pizza_name='Meatlovers' then 12
when pizza_name='Vegetarian' then 10 end)as cost
from cte_delivered_orders;
```



cost_pizza
138

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

```
with cte_main as (
  select
    order_id,
    customer_id,
    extras,
    pizza_name,
    row_number() over(
      order by
        order_id,
        customer_id,
        o.pizza_id
    ) as rnum
  from
    pizza_runner.customer_orders o
    inner join pizza_runner.pizza_names pz on o.pizza_id = pz.pizza_id
),
cte_extras as (
  select
    order_id,
    customer_id,
    pizza_name,
    rnum,
    case when extras = 'null'
    or extras is null
    or extras = '' then '0' else extras end as extras_1
  from
    cte_main
  WHERE
    EXISTS (
      SELECT
        1
      FROM
        pizza_runner.runner_orders
      WHERE
        cte_main.order_id = runner_orders.order_id
        AND runner_orders.pickup_time != 'null'
    )
),
cte_unnest_extras as (
  select
    order_id,
    customer_id,
    pizza_name,
    rnum,
    unnest(
      string_to_array(extras_1, ',')
    ) as extras
  from
    cte_extras
),
cte_cnt_extras as (
  select
    order_id,
    customer_id,
    pizza_name,
    rnum,
    case when extras :: integer = 0 then 0 else 1 end as extras_1
  from
    cte_unnest_extras
  order by
    rnum,
    order_id
),
cte_total_extras as (
  select
    order_id,
    customer_id,
    pizza_name,
    rnum,
    sum(extras_1) as total_extras
  from
    cte_cnt_extras
  group by
    order_id,
    customer_id,
    pizza_name,
    rnum
  order by
    rnum,
    order_id
)
select
  sum(total_cost) as total_money_made
from
  (
    select
      order_id,
      customer_id,
      pizza_name,
      total_extras,
      case when pizza_name = 'Meatlovers' then 12 + total_extras when
pizza_name = 'Vegetarian' then 10 + total_extras end as total_cost
    from
      cte_total_extras
  ) x;
```



total_money_made
142

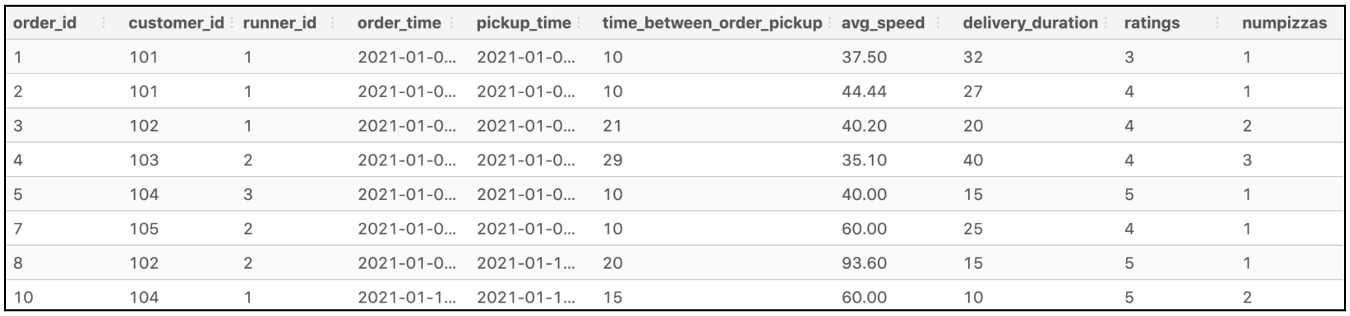
3. The Pizza Runner team now wants to add an additional rating system that allows customers to rate their runner. How would you design an additional table for this new dataset, generate a schema for this new table, and insert your own data for ratings for each successful customer order between 1 to 5.

```
create table pizza_runner.runner_rating (order_id integer, ratings integer)
desc pizza_runner.runner_orders;
insert into pizza_runner.runner_rating
        values (1, 3), (2, 4), (3, 4), (4, 4), (5, 5),  (7, 4),  (8, 5),  (10, 5);
```

4. Using your newly generated table - can you join all of the information together to form a table that 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

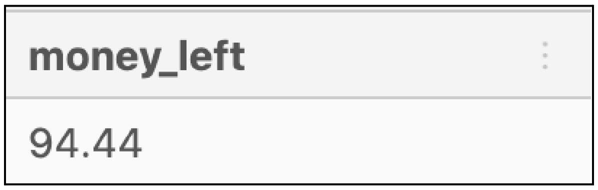
```
select
  ro.order_id,
  customer_id,
  runner_id,
  order_time,
  pickup_time,
  extract(
    'minutes'
    from
      pickup_time :: timestamp - order_time
  ) as time_between_order_pickup,
  round(
    60.0 * unnest(
      regexp_matches(distance, '^[0-9.]+')
    ):: numeric / unnest(
      regexp_matches(duration, '^[0-9]+')
    ):: numeric,
    2
  ) as avg_speed,
  unnest(
    regexp_matches(duration, '^[0-9]+')
  ):: numeric as delivery_duration,
  ratings,
  count(*) as numPizzas
from
  pizza_runner.customer_orders o
  join pizza_runner.runner_orders ro on ro.order_id = o.order_id
  join pizza_runner.runner_rating r on r.order_id = ro.order_id
where
  pickup_time != 'null'
group by
  1,2,3,4,5,6,7,8,9
order by
  order_id;
```



order_id	customer_id	runner_id	order_time	pickup_time	time_between_order_pickup	avg_speed	delivery_duration	ratings	numpizzas
1	101	1	2021-01-0...	2021-01-0...	10	37.50	32	3	1
2	101	1	2021-01-0...	2021-01-0...	10	44.44	27	4	1
3	102	1	2021-01-0...	2021-01-0...	21	40.20	20	4	2
4	103	2	2021-01-0...	2021-01-0...	29	35.10	40	4	3
5	104	3	2021-01-0...	2021-01-0...	10	40.00	15	5	1
7	105	2	2021-01-0...	2021-01-0...	10	60.00	25	4	1
8	102	2	2021-01-0...	2021-01-1...	20	93.60	15	5	1
10	104	1	2021-01-1...	2021-01-1...	15	60.00	10	5	2

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

```
select
       sum(revenue - cost) as money_left
      from
       (
        select
         order_id,
         distance_travelled * 0.3 as cost,
         sum(revenue) as revenue
        from
         (
          select
           ro.order_id,
           unnest(
             regexp_matches(ro.distance, '^[0-9.]+')
           ):: numeric as distance_travelled,
           case when pizza_name = 'Meatlovers' then 12 when pizza_name =
      'Vegetarian' then 10 end as revenue
          from
           pizza_runner.customer_orders o
           join pizza_runner.runner_orders ro on ro.order_id = o.order_id
           join pizza_runner.pizza_names n on o.pizza_id = n.pizza_id
          where
           pickup_time != 'null'
          order by
           order_id
         ) x
        group by
         order_id,
         distance_travelled
        order by
         order_id
       ) x;
```



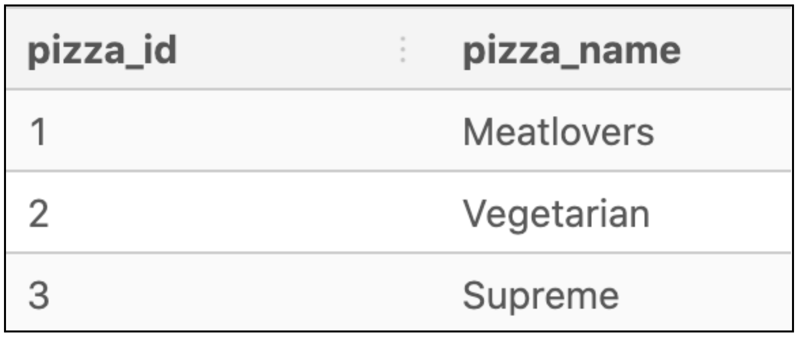
money_left
94.44

E. Bonus Questions

If Danny wants to expand his range of pizzas - how would this impact the existing data design? Write an INSERT statement to demonstrate what would happen if a new Supreme pizza with all the toppings was added to the Pizza Runner menu.

```
create table pizza_runner.pizza_names_temp
(pizza_id integer, pizza_name text);

-- Inserting data from the existing table
insert into pizza_runner.pizza_names_temp (
        select
          pizza_id,
          pizza_name
        from
          pizza_runner.pizza_names);

-- Inserting 'Supreme' pizza data
insert into pizza_runner.pizza_names_temp
              values (3, 'Supreme');

create table pizza_runner.pizza_recipes_temp
              (pizza_id integer, toppings text);

-- Inserting data from the existing table
  insert into pizza_runner.pizza_recipes_temp (
    select
      pizza_id,
      toppings
    from
      pizza_runner.pizza_recipes);
```

```
-- Inserting 'Supreme' pizza data
  insert into pizza_runner.pizza_recipes_temp
           values (3, (select string_agg(topping_id :: text, ',')
                       from pizza_runner.pizza_toppings));
```



pizza_id	pizza_name
1	Meatlovers
2	Vegetarian
3	Supreme