Case Study Questions

This case study has LOTS of questions - they are broken up by area of focus including:

- Pizza Metrics
- Runner and Customer Experience
- Ingredient Optimisation
- Pricing and Ratings
- Bonus DML Challenges (DML = Data Manipulation Language)

Each of the following case study questions can be answered using a single SQL statement.

Again, there are many questions in this case study - please feel free to pick and choose which ones you'd like to try!

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(order_id) as number_of_pizzas_ordered
From Customer_orders;
```

number_of_pizzas_ordered
14

2. How many unique customer orders were made?

```
Select count(distinct(order_id)) as number_of_unique_customer_orders
From Customer_orders;
```

number_of_unique_customer_orders
10

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

```
Select runner_id, count(runner_id) as number_of_successful_orders
From Runner_orders
Where distance != 0
Group by runner_id;
```

runner_id	number_of_successful_orders
1	4
2	3
3	1

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

```
Select p.pizza_name, count(c.pizza_id) as number_of_delivered_pizza
From customer_orders as c
Join runner_orders as r On c.order_id=r.order_id
Join pizza_names as p On c.pizza_id=p.pizza_id
Where r.distance != 0
Group by p.pizza_name;
```

	pizza_name	number_of_delivered_pizza
▶	Meatlovers	9
	Vegetarian	3

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

```
Select c.customer_id, p.pizza_name, count(p.pizza_name) as quatity
From customer_orders as c
Join pizza_names as p On c.pizza_id=p.pizza_id
Group by c.customer_id, p.pizza_name
Order by c.customer_id;
```

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

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

```
Create View count_of_pizza As
Select c.order_id, count(c.pizza_id) as quatity_of_pizza
From customer_orders as c
Join runner_orders as r On c.order_id=r.order_id
Where r.distance != 0
Group by c.order_id;
```

	order_id	quatity_of_pizza
▶	1	1
	2	1
	3	2
	4	3
	5	1
	7	1
	8	1
	10	2

```
Select max(quatity_of_pizza) as maximum_number_of_pizzas_delivered
From count_of_pizza;
```

	maximum_number_of_pizzas_delivered
▶	3

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

```sql
Select c. customer_id,
sum(case when c.exclusions != '' and c.exclusions != 'null' OR c.extras != ''and c.extras != 'null' Then 1 Else 0 END) as at_least_1_change,
sum(case when c.exclusions ='' or c.exclusions = 'null' AND c.extras = '' or c.extras = 'null' Then 1 Else 0 END) as no_changes
From customer_orders as c
Join runner_orders as r On c.order_id=r.order_id
Where r.distance != 0
Group by c.customer_id
Order by c.customer_id;
```

customer_id	at_least_1_change	no_changes
101	0	2
102	0	3
103	3	0
104	2	1
105	1	0

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

```sql
Select sum(case
when c.exclusions != '' and c.exclusions != 'null' AND c.extras != ''and c.extras != 'null'
Then 1 Else 0 END) as exclusions_and_extras
From customer_orders as c
Join runner_orders as r On c.order_id=r.order_id
Where r.distance != 0;
```

exclusions_and_extras
1

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

```sql
Select count(order_id) quatity_of_pizzas, hour(order_time) as hours_of_the_day
From customer_orders
Group by hours_of_the_day
Order by hours_of_the_day;
```

quatity_of_pizzas	hours_of_the_day
1	11
3	13
3	18
1	19
3	21
3	23

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

```sql
Select date_format(date_add(order_time, Interval 2 Day),'%W') as day_of_the_week, count(order_id) as total_of_pizzas
From customer_orders
Group by day_of_the_week
Order by total_of_pizzas DESC;
```

day_of_the_week	total_of_pizzas
Friday	5
Monday	5
Saturday	3
Sunday	1

B. Runner and Customer Experience

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

```
Select count(runner_id) as runner_sign_up, week(date_add(registration_date, Interval 2 Day)) as number_of_week
From runners
Group by number_of_week;
```

runner_sign_up	number_of_week
2	1
1	2
1	3

2. What was the average time in minutes it took for each runner to arrive at the Pizza Runner HQ to pickup the order?

```
Create View Delivery_Time_By_Runner As Select r.runner_id, r.pickup_time, c.order_time,
TIMESTAMPDIFF(minute, c.order_time, r.pickup_time) as delivery_time_in_minutes
From runner_orders as r
Join customer_orders as c On r.order_id = c.order_id
Where pickup_time != 0;
```

runner_id	pickup_time	order_time	delivery_time_in_minutes
1	2020-01-01 18:15:34	2020-01-01 18:05:02	10
1	2020-01-01 19:10:54	2020-01-01 19:00:52	10
1	2020-01-03 00:12:37	2020-01-02 23:51:23	21
1	2020-01-03 00:12:37	2020-01-02 23:51:23	21
2	2020-01-04 13:53:03	2020-01-04 13:23:46	29
2	2020-01-04 13:53:03	2020-01-04 13:23:46	29
2	2020-01-04 13:53:03	2020-01-04 13:23:46	29
3	2020-01-08 21:10:57	2020-01-08 21:00:29	10
2	2020-01-08 21:30:45	2020-01-08 21:20:29	10
2	2020-01-10 00:15:02	2020-01-09 23:54:33	20
1	2020-01-11 18:50:20	2020-01-11 18:34:49	15
1	2020-01-11 18:50:20	2020-01-11 18:34:49	15

```
Select runner_id, round(avg(delivery_time_in_minutes),0) as average_time
From delivery_time_by_runner
Group by runner_id;
```

runner_id	average_time
1	15
2	23
3	10

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

```
Create View Order_time_pizza As
Select c.order_id, count(c.order_id) as number_of_pizza, c.order_time, r.pickup_time,
TIMESTAMPDIFF(minute, c.order_time, r.pickup_time) as delivery_time_in_minutes
From customer_orders as c
Join runner_orders as r On c.order_id = r.order_id
Where pickup_time != 0
Group by c.order_id, c.order_time, r.pickup_time;
```

order_id	number_of_pizza	order_time	pickup_time	delivery_time_in_minutes
1	1	2020-01-01 18:05:02	2020-01-01 18:15:34	10
2	1	2020-01-01 19:00:52	2020-01-01 19:10:54	10
3	2	2020-01-02 23:51:23	2020-01-03 00:12:37	21
4	3	2020-01-04 13:23:46	2020-01-04 13:53:03	29
5	1	2020-01-08 21:00:29	2020-01-08 21:10:57	10
7	1	2020-01-08 21:20:29	2020-01-08 21:30:45	10
8	1	2020-01-09 23:54:33	2020-01-10 00:15:02	20
10	2	2020-01-11 18:34:49	2020-01-11 18:50:20	15

```
Select number_of_pizza, round(avg(delivery_time_in_minutes),0) as average_delivery_time
From Order_time_pizza
Group by number_of_pizza;
```

number_of_pizza	average_delivery_time
1	12
2	18
3	29

With more pizzas, the average delivery time increases.

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

```
Select c.customer_id, round(avg(r.distance),0) as average_distance
From customer_orders as c
Join runner_orders as r On c.order_id = r.order_id
Where r.duration != 0
Group by c.customer_id;
```

customer_id	average_distance
101	20
102	17
103	23
104	10
105	25

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

```sql
SELECT
  max(left(duration,2)) - min(left(duration,2)) as the_biggest_difference
FROM runner_orders
WHERE duration != 'null';
```

	the_biggest_difference
▶	30

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

```sql
Select r.runner_id, c.customer_id, c.order_id, count(c.order_id) as quality_of_pizzas,
 r.distance, round(avg(r.distance/(left(r.duration,2))*60),2) as average_speed
From runner_orders as r
Join customer_orders as c On r.order_id=c.order_id
Where r.duration != 'null'
Group by r.runner_id,  c.customer_id, c.order_id
Order by c. order_id;
```

	runner_id	customer_id	order_id	quality_of_pizzas	distance	average_speed
▶	1	101	1	1	20km	37.5
	1	101	2	1	20km	44.44
	1	102	3	2	13.4km	40.2
	2	103	4	3	23.4	35.1
	3	104	5	1	10	40
	2	105	7	1	25km	60
	2	102	8	1	23.4 km	93.6
	1	104	10	2	10km	60

Runner 1: Its average speeds are in the range of 37.5 to 60 and are achieved for a distance of 10-20 km with 1 or 2 pizzas. It reaches the highest speed over the shortest distance.

Runner 2: His average speeds are 35.1, 60, 93 over a similar distance. He moved the slowest with the most pizzas - 3.

Runner 3: Only once did he deliver pizzas - his speed was 10, for a distance of 40

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

```sql
Select runner_id,
round( 100* sum(case when distance = 'null' Then 0 Else 1 END) / count(distance),0) as successful_delivery_percentage
From runner_orders
Group by runner_id;
```

	runner_id	successful_delivery_percentage
▶	1	100
	2	75
	3	50

C. Ingredient Optimisation

1 What are the standard ingredients for each pizza?

```
Create table pizza_recipos_2
(pizza_id int,
toppinings int);

Insert into pizza_recipos_2(pizza_id, toppinings)
values
(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);

Select *
From pizza_recipos_2;
```

pizza_id	toppinings
1	1
1	2
1	3
1	4
1	5
1	6
1	8
1	10
2	4
2	6
2	7

```
Create View Ingredients_pizza As
Select pn.pizza_name, pr2.pizza_id, pt.topping_name
From pizza_recipos_2 as pr2
Join pizza_toppings as pt On pr2.toppinings = pt.topping_id
Join pizza_names as pn On pr2.pizza_id=pn.pizza_id
Order by pr2.pizza_id;
```

	pizza_name	pizza_id	topping_name
▶	Meatlovers	1	Bacon
	Meatlovers	1	BBQ Sauce
	Meatlovers	1	Beef
	Meatlovers	1	Cheese
	Meatlovers	1	Chicken
	Meatlovers	1	Mushrooms
	Meatlovers	1	Pepperoni
	Meatlovers	1	Salami
	Vegetarian	2	Cheese
	Vegetarian	2	Mushrooms
	Vegetarian	2	Onions

Solution:

```
Select pizza_name, group_concat(topping_name) as all_ingredients
From Ingredients_pizza
Group by pizza_name;
```

	pizza_name	all_ingredients
▶	Meatlovers	Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami
	Vegetarian	Cheese,Mushrooms,Onions,Peppers,Tomatoes,Tomato Sauce

2. What was the most commonly added extra?

First let's see for which pizza_id extras are given.

```
Select pizza_id, extras
From customer_orders
Where extras != '' and extras != 'null';
```

	pizza_id	extras
▶	1	1
	2	1
	1	1, 5
	1	1, 4

We see that we need to create a normalized table.

```
Create table extras_pizza
(pizza_id int,
extras int);

Insert into extras_pizza(pizza_id, extras)
values
(1,1), (2,1), (1,1), (1,5), (1,1), (1,4);

Select *
From extras_pizza;
```

	pizza_id	extras
▶	1	1
	2	1
	1	1
	1	5
	1	1
	1	4

Finally we can present the final table.

```
Select ep.extras, pt.topping_name, count(ep.extras) as number_of_repetitions
From extras_pizza as ep
Join pizza_toppings as pt On ep.extras = pt.topping_id
Group by ep.extras;
```

	extras	topping_name	number_of_repetitions
▶	1	Bacon	4
	4	Cheese	1
	5	Chicken	1

3. What was the most common exclusion?

First let's see for which pizza_id exclusions are given.

```
Select pizza_id, exclusions
From customer_orders
Where exclusions != '' and exclusions != 'null';
```

	pizza_id	exclusions
▶	1	4
	1	4
	2	4
	1	4
	1	2, 6

We see that we need to create a normalized table.

```
Create table exclusions_pizza
(pizza_id int,
exclusions int);

Insert into exclusions_pizza(pizza_id, exclusions)
values
(1,4), (1,4), (2,4), (1,4), (1,2), (1,6);

Select *
From exclusions_pizza;
```

	pizza_id	exclusions
▶	1	4
	1	4
	2	4
	1	4
	1	2
	1	6

Finally we can present the final table.

```
Select xp.exclusions, pt.topping_name, count(xp.exclusions) as number_of_repetitions
From exclusions_pizza as xp
Join pizza_toppings as pt On xp.exclusions = pt.topping_id
Group by xp.exclusions;
```

	exclusions	topping_name	number_of_repetitions
▶	2	BBQ Sauce	1
	4	Cheese	4
	6	Mushrooms	1

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

a. *Meat Lovers*
b. *Meat Lovers - Exclude Beef*
c. *Meat Lovers - Extra Bacon*
d. *Meat Lovers - Exclude Cheese, Bacon - Extra Mushroom, Peppers*

```sql
Select c.order_id, c.customer_id, c.pizza_id, c.exclusions, c.extras, c.order_time, case
when c.pizza_id = 1 and (exclusions = '' or exclusions = 'null') and (extras = 'null' or extras = '') then 'Meat Lovers'
when c.pizza_id = 1 and (exclusions like '%3' or exclusions = 3) and (extras ='null' or extras ='') then 'Meat Lovers - Exclude Beef'
when c.pizza_id = 1 and (extras like '%1' or extras = 1) and (exclusions ='null' or exclusions ='') then 'Meat Lovers - Extra Bacon'
when c.pizza_id =1 and (exclusions like '%1' or exclusions = 1) and (exclusions like '%4' or exclusions =4)
and (extras like '%6' or extras =6) and (extras like '%9' or extras = 9) then 'Meat Lovers - Exclude Cheese, Bacon - Extras Mushroom, Peppers'
when c.pizza_id = 1 and (exclusions like '%4' or exclusions = 4) and (extras ='null' or extras ='') then 'Meat Lovers - Exclude Cheese'
when c.pizza_id =1 and (exclusions like '%4' or exclusions = 4) and (extras like '%1' or extras =1) and (extras like '%5' or extras = 5)
then 'Meat Lovers - Exclude Chese - Extras Bacon, Chicken'
when c.pizza_id =1 and (exclusions like '%2' or exclusions = 2) and (exclusions like '%6' or exclusions =6)
and (extras like '%1' or extras =1) and (extras like '%4' or extras = 4)
then 'Meat Lovers - Exclude BBQ Sauce, Mushrooms - Extras Bacon, Cheese'
when c.pizza_id = 2 and (exclusions = '' or exclusions = 'null') and (extras = 'null' or extras = '' or extras is null) then 'Veg Lovers'
when c.pizza_id = 2 and (exclusions like '%4' or exclusions = 4)
and (extras = 'null' or extras = '' or extras is null) then 'Veg Lovers - Exclude Cheese'
when c.pizza_id = 2 and (extras like '%1' or extras = 1) and (exclusions ='null' or exclusions ='') then 'Veg Lovers - Extra Bacon'
End as Order_Item
From customer_orders as c
Join pizza_names as p On p.pizza_id=c.pizza_id;
```

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

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

a. For example: "Meat Lovers: 2xBacon, Beef, ... , Salami"

We create the auxiliary table

```sql
Create view ingredient_names as
Select pr.pizza_id, group_concat(topping_name) as ingredients
From pizza_recipos_2 as pr
Join pizza_toppings as pt On pr.toppinings = pt.topping_id
Group by pr.pizza_id;

Select *
From ingredient_names;
```

pizza_id	ingredients
1	Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushr...
2	Cheese,Mushrooms,Onions,Peppers,Tomatoes,...

And one more auxiliary table

```sql
Create view table_of_extras as
Select c.order_id, case
when (c.extras like '%1' or c.extras = 1) Then 'Bacon'
Else ''
End as Extras_Ingredients_1, case
when (c.extras like '%4' or c.extras = 4) Then 'Cheese'
when (c.extras like '%5' or c.extras = 5) Then 'Chicken'
Else ''
End as Extras_Ingredients_2, pn.pizza_name, i.ingredients
From customer_orders as c
Join pizza_names as pn On c.pizza_id = pn.pizza_id
Join ingredient_names as i On pn.pizza_id= i.pizza_id;

Select *
From table_of_extras;
```

order_id	Extras_Ingredients_1	Extras_Ingredients_2	pizza_name	ingredients
1			Meatlovers	Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami
2			Meatlovers	Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami
3			Meatlovers	Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami
3			Vegetarian	Cheese,Mushrooms,Onions,Peppers,Tomatoes,Tomato Sauce
4			Meatlovers	Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami
4			Meatlovers	Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami
4			Vegetarian	Cheese,Mushrooms,Onions,Peppers,Tomatoes,Tomato Sauce
5	Bacon		Meatlovers	Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami
6			Vegetarian	Cheese,Mushrooms,Onions,Peppers,Tomatoes,Tomato Sauce
7	Bacon		Vegetarian	Cheese,Mushrooms,Onions,Peppers,Tomatoes,Tomato Sauce
8			Meatlovers	Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami
9	Bacon	Chicken	Meatlovers	Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami
10			Meatlovers	Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami
10	Bacon	Cheese	Meatlovers	Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami

Final table

```sql
SELECT order_id, CONCAT(pizza_name, ' : ', case
when Extras_Ingredients_1 != ''
Then REPLACE(REPLACE(ingredients, Extras_Ingredients_2,
CONCAT('2x',Extras_Ingredients_2)), Extras_Ingredients_1, CONCAT('2x',Extras_Ingredients_1))
Else ingredients
End)
as pizza_ingredients
FROM table_of_extras;
```

order_id	pizza_ingredients
1	Meatlovers : Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami
2	Meatlovers : Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami
3	Meatlovers : Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami
3	Vegetarian : Cheese,Mushrooms,Onions,Peppers,Tomatoes,Tomato Sauce
4	Meatlovers : Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami
4	Meatlovers : Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami
4	Vegetarian : Cheese,Mushrooms,Onions,Peppers,Tomatoes,Tomato Sauce
5	Meatlovers : 2xBacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami
6	Vegetarian : Cheese,Mushrooms,Onions,Peppers,Tomatoes,Tomato Sauce
7	Vegetarian : Cheese,Mushrooms,Onions,Peppers,Tomatoes,Tomato Sauce
8	Meatlovers : Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami
9	Meatlovers : 2xBacon,BBQ Sauce,Beef,Cheese,2xChicken,Mushrooms,Pepperoni,Salami
10	Meatlovers : Bacon,BBQ Sauce,Beef,Cheese,Chicken,Mushrooms,Pepperoni,Salami
10	Meatlovers : 2xBacon,BBQ Sauce,Beef,2xCheese,Chicken,Mushrooms,Pepperoni,Salami

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

```sql
Select pt.topping_name, count(c.pizza_id) as number_of_used
From customer_orders as c
Join pizza_recipos_2 as pr On pr.pizza_id = c.pizza_id
Join pizza_toppings as pt On pt.topping_id = pr.toppinings
Join runner_orders as r On r.order_id = c.order_id
Where r.distance != 0
Group by pr.toppinings
Order by number_of_used DESC;
```

topping_name	number_of_used
Cheese	12
Mushrooms	12
Bacon	9
BBQ Sauce	9
Beef	9
Chicken	9
Pepperoni	9
Salami	9
Onions	3
Peppers	3
Tomatoes	3
Tomato Sauce	3

Unfortunately, my solution lacks data about extras and exclusions:

extras	topping_name	number_of_repetitions
1	Bacon	4
4	Cheese	1
5	Chicken	1

exclusions	topping_name	number_of_repetitions
2	BBQ Sauce	1
4	Cheese	4
6	Mushrooms	1

So the change should be at position: Cheese (12->9), Mushrooms (12->11), Bacon (9-13), BBQ Sause (9->8) and Chicken (9->10).

D. 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?

```
Select sum(case
when c.pizza_id =1 Then 12
Else 10 End) as money_earned
From runner_orders as r
Join customer_orders as c On r.order_id=c.order_id
Where r.distance != 0;
```

money_earned
138

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

a. *Add cheese is $1 extra*

```
Create view total_money as
Select (case
when c.pizza_id =1 Then 12
Else 10 End) as money_earned, c.extras, c.exclusions
From runner_orders as r
Join customer_orders as c On r.order_id=c.order_id
Where r.distance != 0;
```

money_earned	extras	exclusions
12		
12		
12		
10	NULL	
12		4
12		4
10		4
12	1	null
10	1	null
12	null	null
12	null	null
12	1, 4	2, 6

```
Select sum(case
when (extras ='' or extras ='null' or extras is null) Then money_earned
when length(extras) = 1 Then money_earned +1
Else money_earned + 2
End) as total_cost
From total_money;
```

total_cost
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. & **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?

a. *customer_id*
b. *order_id*
c. *runner_id*
d. *rating*
e. *order_time*
f. *pickup_time*
g. *Time between order and pickup*
h. *Delivery duration*
i. *Average speed*
j. *Total number of pizzas*

First I create table based on which I create a rating.

```
Create View All_information As
Select c.customer_id, c.order_id, r.runner_id, c.order_time, r.pickup_time,
timestampdiff( minute, c.order_time, r.pickup_time) as time_between_order_and_pickup_time, r.duration as delivery_duration,
round(avg(r.distance/(left(r.duration,2))*60),2) as average_speed, count(c.order_id) as total_number_of_pizzas
From customer_orders as c
Join runner_orders as r On c.order_id=r.order_id
Group by c.customer_id, c.order_id, r.runner_id, c.order_time, r.pickup_time,time_between_order_and_pickup_time, r.duration
Order by c.customer_id;
```

customer_id	order_id	runner_id	order_time	pickup_time	time_between_order_and_pickup_time	delivery_duration	average_speed	total_number_of_pizzas
101	1	1	2020-01-01 18:05:02	2020-01-01 18:15:34	10	32 minutes	37.5	1
101	2	1	2020-01-01 19:00:52	2020-01-01 19:10:54	10	27 minutes	44.44	1
101	6	3	2020-01-08 21:03:13	null	NULL	null	NULL	1
102	3	1	2020-01-02 23:51:23	2020-01-03 00:12:37	21	20 mins	40.2	2
102	8	2	2020-01-09 23:54:33	2020-01-10 00:15:02	20	15 minute	93.6	1
103	4	2	2020-01-04 13:23:46	2020-01-04 13:53:03	29	40	35.1	3
103	9	2	2020-01-10 11:22:59	null	NULL	null	NULL	1
104	5	3	2020-01-08 21:00:29	2020-01-08 21:10:57	10	15	40	1
104	10	1	2020-01-11 18:34:49	2020-01-11 18:50:20	15	10minutes	60	2
105	7	2	2020-01-08 21:20:29	2020-01-08 21:30:45	10	25mins	60	1

I made my rating system dependent on the delivery time of the pizza.

Delivery time – null Rating 1

Delivery time >25 Rating 2

Delivery time >= 20 and <25 Rating 3

Delivery time >10 and <20 Rating 4

Delivery time <= 10 Rating 5

```sql
Create View All_information_with_rating as
Select *, (case
when time_between_order_and_pickup_time is null Then 1
when time_between_order_and_pickup_time > 25 Then 2
when time_between_order_and_pickup_time >= 20 and time_between_order_and_pickup_time <25 Then 3
when time_between_order_and_pickup_time >10 and time_between_order_and_pickup_time < 20  Then 4
Else 5 End) as rating
From All_information;
```

customer_id	order_id	runner_id	order_time	pickup_time	time_between_order_and_pickup_time	delivery_duration	average_speed	total_number_of_pizzas	ratin
101	1	1	2020-01-01 18:05:02	2020-01-01 18:15:34	10	32 minutes	37.5	1	5
101	2	1	2020-01-01 19:00:52	2020-01-01 19:10:54	10	27 minutes	44.44	1	5
101	6	3	2020-01-08 21:03:13	null	NULL	null	NULL	1	1
102	3	1	2020-01-02 23:51:23	2020-01-03 00:12:37	21	20 mins	40.2	2	3
102	8	2	2020-01-09 23:54:33	2020-01-10 00:15:02	20	15 minute	93.6	1	3
103	4	2	2020-01-04 13:23:46	2020-01-04 13:53:03	29	40	35.1	3	2
103	9	2	2020-01-10 11:22:59	null	NULL	null	NULL	1	1
104	5	3	2020-01-08 21:00:29	2020-01-08 21:10:57	10	15	40	1	5
104	10	1	2020-01-11 18:34:49	2020-01-11 18:50:20	15	10minutes	60	2	4
105	7	2	2020-01-08 21:20:29	2020-01-08 21:30:45	10	25mins	60	1	5

```sql
Select runner_id, round(avg(rating),2) as average_rating
From All_information_with_rating
Group by runner_id
Order by runner_id;
```

runner_id	average_rating
1	4.25
2	2.75
3	3.00

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?

```sql
SELECT round(sum(CASE
WHEN c.pizza_id=1 THEN 12
WHEN c.pizza_id = 2 THEN 10 END)  - SUM((r.distance+0) * 0.3),2) AS pizza_cost
FROM runner_orders r
JOIN customer_orders c ON c.order_id = r.order_id
WHERE r.distance != 0;
```

pizza_cost
73.38