8 Week SQL Challenge
Pizza Runner Case study

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

Because Danny had a few years of experience as a data scientist - he was very aware that data collection would be critical for his business’ growth.

He has prepared for us an entity relationship diagram of his database design. Still, he requires further assistance to clean his data and apply some basic calculations to better direct his runners and optimize 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.

Datasets

Table 1: runners

The runners table shows the registration_date for each new runner

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

Table 2: customer_orders

Customer pizza orders are captured in the customer_orders table with 1 row for each individual pizza that is part of the order.

The pizza_id relates to the type of pizza which was ordered whilst the exclusions are the ingredient_id values which should be removed from the pizza and the extras are the ingredient_id values that need to be added to the pizza.

Note that customers can order multiple pizzas in a single order with varying exclusions and extras values even if the pizza is the same type!

The exclusions and extra columns will need to be cleaned up before using them in your queries.

order_id	customer_id	pizza_id	exclusions	extras	order_time
1	101	1			2021-01-01 18:05:02
2	101	1			2021-01-01 19:00:52
3	102	1			2021-01-02 23:51:23
3	102	2		NaN	2021-01-02 23:51:23
4	103	1	4		2021-01-04 13:23:46
4	103	1	4		2021-01-04 13:23:46
4	103	2	4		2021-01-04 13:23:46
5	104	1	null	1	2021-01-08 21:00:29
6	101	2	null	null	2021-01-08 21:03:13
7	105	2	null	1	2021-01-08 21:20:29
8	102	1	null	null	2021-01-09 23:54:33
9	103	1	4	1, 5	2021-01-10 11:22:59
10	104	1	null	null	2021-01-11 18:34:49
10	104	1	2, 6	1, 4	2021-01-11 18:34:49

Table 3: runner_orders

After each order are received through the system - they are assigned to a runner - however not all orders are fully completed and can be canceled by the restaurant or the customer.

The pickup_time is the timestamp at which the runner arrives at the Pizza Runner headquarters to pick up the freshly cooked pizzas. The distance and duration fields are related to how far and long the runner had to travel to deliver the order to the respective customer.

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

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

Table 4: pizza_names

At the moment - Pizza Runner only has 2 pizzas available the Meat Lovers or Vegetarian!

pizza_id	pizza_name
1	Meat Lovers
2	Vegetarian

Table 5: pizza_recipes

Each pizza_id has a standard set of toppings which are used as part of the pizza recipe.

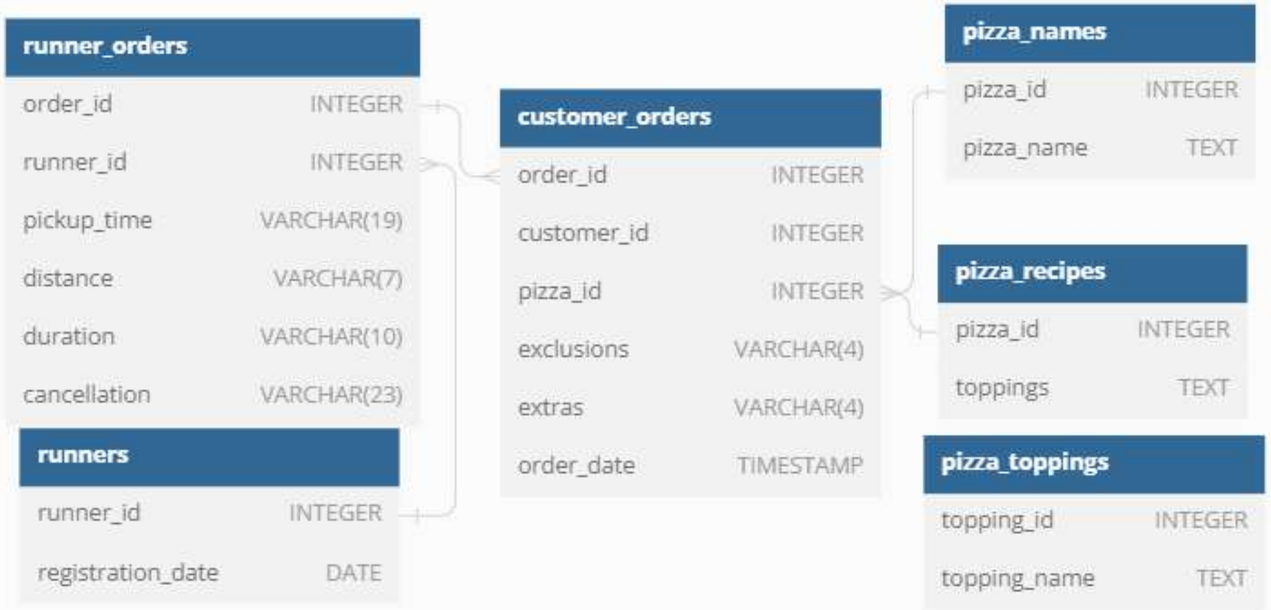
pizza_id	toppings
1	1, 2, 3, 4, 5, 6, 8, 10
2	4, 6, 7, 9, 11, 12

Table 6: pizza_toppings

This table contains all of the topping_name values with their corresponding topping_id value

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

Entity Relationship Diagram



Data Cleaning

Before starting the analysis the table needs to be cleaned.

1. Customer_order Table

The Extras and Exclusion column are having null and NaN values which i will replace with blank values

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

After replacing all the null and NaN values with blank values the table is now ready to be used in our queries

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			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		1	2020-01-08 21:00:29
6	101	2			2020-01-08 21:03:13
7	105	2		1	2020-01-08 21:20:29
8	102	1			2020-01-09 23:54:33
9	103	1	4	1, 5	2020-01-10 11:22:59
10	104	1			2020-01-11 18:34:49
10	104	1	2, 6	1, 4	2020-01-11 18:34:49

2. Runner_orders Table

This table is also having null and NaN values which need to be replaced with blank values.
As you can also see that Distance column Is having data km but we need to remove the units to make some calculations.
We will also remove minutes from the duration column.

```
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 '%mins' THEN TRIM('mins' from duration)
    WHEN duration LIKE '%minute' THEN TRIM('minute' from duration)
    WHEN duration LIKE '%minutes' THEN TRIM('minutes' from duration)
    ELSE duration END AS duration,
  CASE
    WHEN cancellation IS NULL or cancellation LIKE 'null' THEN ''
    ELSE cancellation END AS cancellation
```

After replacing all the null and NaN values with blank values the table is now ready to be used in our queries

order_id	runner_id	pickup_time	distance	duration	cancellation
1	1	2020-01-01 18:15:34	20	32	
2	1	2020-01-01 19:10:54	20	27	
3	1	2020-01-03 00:12:37	13.4	20	
4	2	2020-01-04 13:53:03	23.4	40	
5	3	2020-01-08 21:10:57	10	15	
6	3				Restaurant Cancellation
7	2	2020-01-08 21:30:45	25	25	
8	2	2020-01-10 00:15:02	23.4	15	
9	2				Customer Cancellation
10	1	2020-01-11 18:50:20	10	10	

Pizza Metrics

1. How many pizzas were ordered?

```
SELECT
    COUNT(1) AS Total_orders
FROM
    customer_orders;
```

A total of 14 Pizza orders were made for pizza runner restaurant

	Total_orders
▶	14

2. How many unique customer orders were made?

```
SELECT
    COUNT(DISTINCT customer_id) AS total_customers
FROM
    customer_orders;
```

There were 5 Unique customers

	total_customers
▶	5

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

```
SELECT
    runner_id, COUNT(1) AS total_deliveries
FROM
    runner_orders
WHERE
    distance != ''
GROUP BY runner_id;
```

runner_id	total_deliveries
1	4
2	3
3	1

Runner 1 has delivered the highest number of pizza whereas runner 3 has delivered the least

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

```
SELECT
    pizza_name, COUNT(1) AS total_orders_deliverd
FROM
    customer_orders AS c
        JOIN
    runner_orders AS r ON r.order_id = c.order_id
        JOIN
    pizza_names AS p ON p.pizza_id = c.pizza_id
WHERE
    r.distance != ''
GROUP BY pizza_name;
```

pizza_name	total_orders_deliverd
Meatlovers	9
Vegetarian	3

Meatlovers Pizza was delivered 9 times and vegetarian pizza was delivered 3 times

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

```sql
SELECT
    c.customer_id, p.pizza_name, COUNT(1) AS total_orders
FROM
    customer_orders AS c
        JOIN
    pizza_names AS p ON p.pizza_id = c.pizza_id
GROUP BY c.customer_id , p.pizza_name
ORDER BY c.customer_id;
```

customer_id	pizza_name	total_orders
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?

Most Pizza delivered in a single order was 3

```sql
SELECT
    r.order_id, COUNT(1) AS total_orders_deliverd
FROM
    customer_orders AS c
        JOIN
    runner_orders AS r ON r.order_id = c.order_id
WHERE
    r.distance != ''
GROUP BY r.order_id
ORDER BY total_orders_deliverd DESC
LIMIT 1;
```

order_id	total_orders_deliverd
4	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 != '' 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_orders AS c
JOIN runner_orders AS r
  ON c.order_id = r.order_id
WHERE r.duration != ''
GROUP BY c.customer_id
ORDER BY c.customer_id;
```

customer-id 101, and 102 had no changes made whereas customer 103 has most changes

customer_id	at_least_1_change	no_change
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
    c.customer_id,
    SUM(CASE
        WHEN c.exclusions != '' AND c.extras != '' THEN 1
        ELSE 0
    END) AS at_least_1_change
FROM
    customer_orders AS c
        JOIN
    runner_orders AS r ON c.order_id = r.order_id
WHERE
    r.duration != ''
GROUP BY c.customer_id
ORDER BY c.customer_id;
```

Only customer 104 had both exclusion and added extra ingredients to the pizza

customer_id	at_least_1_change
101	0
102	0
103	0
104	1
105	0

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

```sql
SELECT
    DATE(order_time) AS Dt,
    HOUR(order_time) AS Hr,
    COUNT(1) AS total_orders
FROM
    customer_orders
GROUP BY Dt , Hr;
```

The highest number of pizzas were ordered in the afternoon at 1 pm and 9 pm in night

	Hr	total_orders
1	18	1
1	19	1
2	23	2
4	13	3
8	21	3
9	23	1
0	11	1
1	18	2

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

```sql
WITH orders_by_day AS (
SELECT
    COUNT(order_id) AS order_count,
    WEEKDAY(order_time) AS day
FROM customer_orders
GROUP BY day
ORDER BY day
)

SELECT
    order_count,
    CASE
    WHEN day = 0 THEN 'Monday'
        WHEN day = 1 THEN 'Tuesday'
    WHEN day = 2 THEN 'Wednesday'
    WHEN day = 3 THEN 'Thursday'
    WHEN day = 4 THEN 'Friday'
    WHEN day = 5 THEN 'Saturday'
    WHEN day = 6 THEN 'Sunday'
   END AS day
FROM orders_by_day;
```

The highest number of pizza were ordered on Wednesday and Saturday and the least of Friday's

order_count	day
5	Wednesday
3	Thursday
1	Friday
5	Saturday

Runner and Customer Experience

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

```sql
SELECT
    EXTRACT(WEEK FROM registration_date + 3) AS week_of_year,
    COUNT(1) AS total_signups
FROM
    runners
GROUP BY week_of_year;
```

2 Runner Signed up on the 1st Week of the year in the following weeks other 2 runner's joined

week_of_year	total_signups
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 pick up the order?

```sql
SELECT
    r.runner_id,
    AVG(MINUTE(TIMEDIFF(r.pickup_time, c.order_time))) AS time_mins
FROM
    customer_orders AS c
        LEFT JOIN
    runner_orders AS r ON c.order_id = r.order_id
GROUP BY r.runner_id;
```

Runner 2 took 23 mins on avg to pick up the order and runners 1 & 3 took 10 mins on avg

runner_id	time_mins
1	15.3333
2	23.4000
3	10.0000

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

```sql
SELECT
    c.order_id,
    MINUTE(TIMEDIFF(r.pickup_time, c.order_time)) AS prep_time,
    COUNT(1) AS tot_pizzas
FROM
    customer_orders AS c
        JOIN
    runner_orders AS r ON r.order_id = c.order_id
WHERE
    r.distance <> ''
GROUP BY c.order_id , prep_time;
```

As the number of pizza orders increases the preparation time also increases

order_id	prep_time	tot_pizzas
1	10	1
2	10	1
3	21	2
4	29	3
5	10	1
7	10	1
8	20	1
10	15	2

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

```sql
SELECT
    c.customer_id, ROUND(AVG(distance), 2) AS avg_distance
FROM
    customer_orders AS c
        JOIN
    runner_orders AS r ON r.order_id = c.order_id
WHERE
    distance <> ''
GROUP BY c.customer_id;
```

The maximum distance covered for delivery was 25 km for customer 105 and the minimum distance covered was 10kms for customer 104

customer_id	avg_distance
101	20
102	16.73
103	23.4
104	10
105	25

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

```sql
SELECT
    MAX(duration) - MIN(duration) AS time_diff
FROM
    runner_orders
WHERE
    distance <> '';
```

The time difference between the longest and shortest delivery is 10 mins

time_diff
30

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

```sql
SELECT
    order_id,
    runner_id,
    ROUND(AVG(distance / (duration / 60)), 2) AS speed
FROM
    runner_orders
WHERE
    distance <> ''
GROUP BY order_id , runner_id;
```

Runner 1 has delivered the maximum number of orders with an avg speed of 40 km/hr and Runner 2 has given only 2 orders. it suggests that higher the avg speed lower will be the numberof deliveries.

order_id	runner_id	speed
1	1	37.5
2	1	44.44
3	1	40.2
4	2	35.1
5	3	40
7	2	60
8	2	93.6
10	1	60

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

Runner 1 is having highest delivery rate at 100% followed by runner 2 at 75% and runner 3 at 50 %

```sql
SELECT
    runner_id,
    ROUND(100 * SUM(CASE
                WHEN cancellation = '' THEN 1
                ELSE 0
            END) / COUNT(1),
            2) AS delivery_percent
FROM
    runner_orders
GROUP BY runner_id;
```

runner_id	delivery_percent
1	100.00
2	75.00
3	50.00

Before moving to the next part we need to rearrange the Pizza_recipes table

pizza_id	toppings
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

Ingredient Optimisation

What are the standard ingredients for each pizza?

```
SELECT
    p.pizza_name, GROUP_CONCAT(t.topping_name) AS ingredients
FROM
    pizza_names AS p
        JOIN
    pizza_recipes_temp AS r ON r.pizza_id = p.pizza_id
        JOIN
    pizza_toppings AS t ON t.topping_id = r.toppings
GROUP BY p.pizza_name;
```

These are the standard ingredients for each pizza

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

What was the most commonly added extra?

```
with ct as
(
select t.topping_name as most_common_extras, count(1)  as count_extras
from customer_orders as c
join pizza_toppings as t
on t.topping_id= c.extras
where c.extras <> ''
group by t.topping_name
order by count_extras desc
limit 1)
select most_common_extras from ct;
```

Bacon was most commonly added as extras in pizzas

most_common_extras
Bacon

What was the most common exclusion?

```sql
with ct as
(
select t.topping_name as most_common_exclusions, count(1)  as count_exclusions
from customer_orders as c
join pizza_toppings as t
on t.topping_id= c.exclusions
where c.exclusions <> ''
group by t.topping_name
order by count_exclusions desc
limit 1)

select most_common_exclusions from ct;
```

Cheese was most commonly excluded from the pizza

most_common_exclusions
Cheese

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 ct as(
select c.order_id, c.customer_id, p.pizza_name,
c.exclusions,c.extras,t.topping_name as minus,
b.topping_name as plus from customer_orders as c
join pizza_names as p
on p.pizza_id= c.pizza_id
left join pizza_toppings as t
on c.exclusions = t.topping_id
left join pizza_toppings as b
on c.extras=b.topping_id)
select order_id, case when pizza_name is not null and minus is null and plus is null then pizza_name
when pizza_name is not null and minus is not null and plus is not null then concat(pizza_name, " - ", "Exclude ", minus, " - ", "Extra "
when pizza_name is not null and minus is not null and plus is  null then concat(pizza_name, " - ", "Exclude ", minus)
when pizza_name is not null and minus is null and plus is not null then concat(pizza_name, " - ", "Extra ", plus)
end  as order_item
from ct;
```

order_id	order_item
1	Meatlovers
2	Meatlovers
3	Meatlovers
3	Vegetarian
4	Meatlovers - Exclude Cheese
4	Meatlovers - Exclude Cheese
4	Vegetarian - Exclude Cheese
5	Meatlovers - Extra Bacon
6	Vegetarian
7	Vegetarian - Extra Bacon
8	Meatlovers
9	Meatlovers - Exclude Cheese - Extra Bacon
10	Meatlovers
10	Meatlovers - Exclude BBQ Sauce - Extra Bacon

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

```sql
SELECT
    t.topping_name, COUNT(1) AS qty
FROM
    pizza_recipes_temp AS r
        JOIN
    pizza_toppings AS t ON t.topping_id = r.toppings
        JOIN
    customer_orders AS c ON c.pizza_id = r.pizza_id
        JOIN
    runner_orders AS ro ON ro.order_id = c.order_id
WHERE
    distance <> ''
GROUP BY t.topping_name
ORDER BY qty DESC;
```

topping_name	qty
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

Pricing and Ratings

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 ct as
(
select p.pizza_name,
 sum(case
when p.pizza_name = 'Meatlovers' then 12
else 10 end )  as price
from customer_orders as c
join runner_orders as r
on r.order_id = c.order_id
join pizza_names as p
on p.pizza_id = c.pizza_id
where r.distance <> ''
group by p.pizza_name)
select sum(price) from ct;
```

Pizza Runner has made total revenue of 138$ to date

	revenue
▶	138

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

- **Add cheese is $1 extra**

```
with ct as
(
select p.pizza_name,
 sum(case
when p.pizza_name = 'Meatlovers' then 12
when pizza_name = 'Vegetarian' then 10 end )  as price,
sum(case
when c.extras != ''  then 1
else 0 end )as e_price
from customer_orders as c
join runner_orders as r
on r.order_id = c.order_id
join pizza_names as p
on p.pizza_id = c.pizza_id
where r.distance <> ''
group by p.pizza_name),
tot as(
select e_price+price as total  from ct)
select sum(total) as total_revenue from tot;
```

If any extras are added to the order will be charged at 1$ then the revenue will $141

total_revenue
141

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 customer_ratings;

insert into customer_ratings
values (1,5),(2,4),(3,3),(4,3),(5,3),(6,1),(7,5),(8,4),(9,3),(10,5);

select   * from customer_ratings;
```

These are the ratings by different customers

order_id	rating
1	5
2	4
3	3
4	3
5	3
6	1
7	5
8	4
9	3
10	5

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

```sql
SELECT
    c.customer_id,
    c.order_id,
    r.runner_id,
    rt.rating,
    c.order_time,
    r.pickup_time,
    MINUTE(TIMEDIFF(c.order_time, r.pickup_time)) AS order_pickup_time,
    r.duration,
    ROUND(AVG(60 * r.distance / r.duration), 1) AS avg_speed,
    COUNT(1) AS pizza_count
FROM
    customer_orders AS c
        JOIN
    runner_orders AS r ON r.order_id = c.order_id
        JOIN
    customer_ratings AS rt ON rt.order_id = c.order_id
WHERE
    r.distance <> ''
GROUP BY c.customer_id , c.order_id , r.runner_id , rt.rating , c.order_time , r.pickup_time , order_pickup_time , r.duration
ORDER BY c.order_id;
```

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

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 kilometer traveled - how much money does Pizza Runner have left over after these deliveries?

After Deducting the delivery costs the total revenue will be $94.4

```
SET @total_price_pizza = 138;
select @total_price_pizza - round((sum(distance) * 0.3),2) as final_price
from runner_orders;
```

final_price
94.44