
DATAWITHDANNY.COM
START YOUR SQL ENGINES
PIZZA
RUNNER
CASE STUDY #2
8 WEEK SQL
CHALLENGE
8WEEKSQLCHALLENGE.COM

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.

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 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

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 which 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 extras 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 orders are received through the system - they are assigned to a runner - however not all orders are fully completed and can be cancelled 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

Solutions

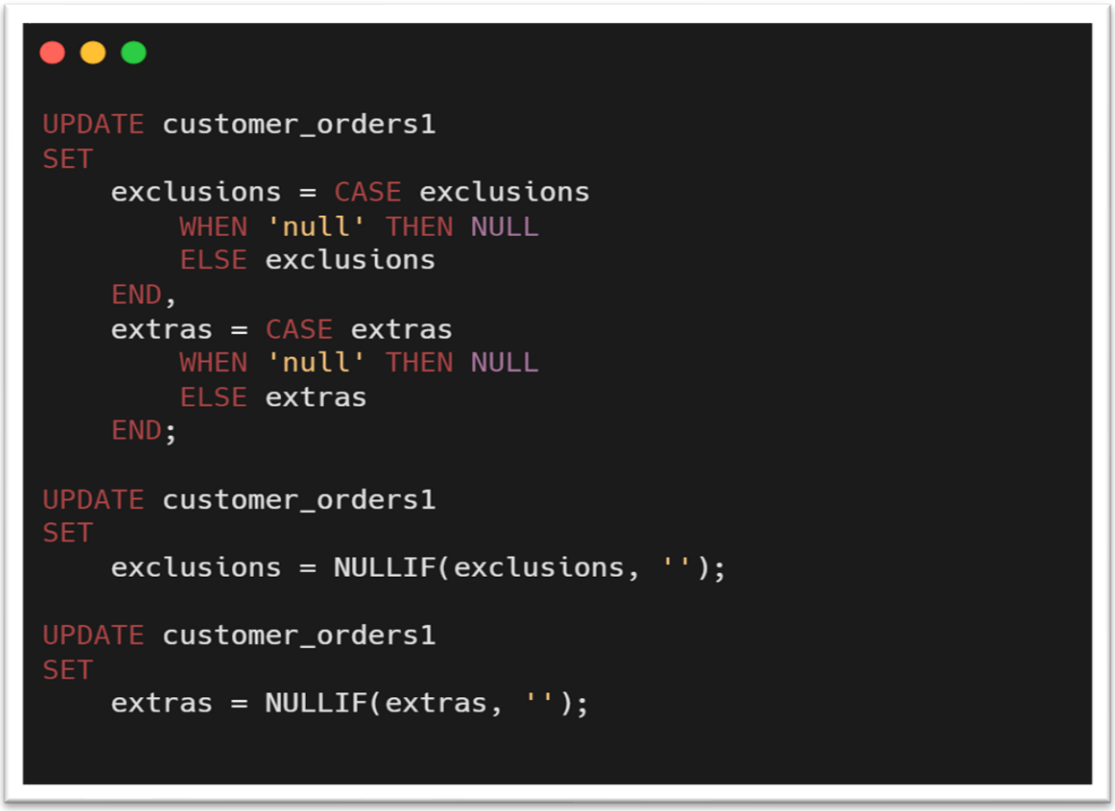
I used MySQL Workbench and these are the particular functions I employed:

- Aggregate functions (SUM, COUNT, AVG)
- Table Joins
- Date Functions (TIME_DIFF)
- Common Table Expressions (CTE)
- CONCAT Function

Data Cleaning

1. Cleaning Customer_orders1

- Creating a new table customer_orders1 to avoid any data loss.



```
UPDATE customer_orders1
SET
    exclusions = CASE exclusions
        WHEN 'null' THEN NULL
        ELSE exclusions
    END,
    extras = CASE extras
        WHEN 'null' THEN NULL
        ELSE extras
    END;

UPDATE customer_orders1
SET
    exclusions = NULLIF(exclusions, '');

UPDATE customer_orders1
SET
    extras = NULLIF(extras, '');
```

2. Cleaning Runner_orders

- Creating a new table runner_orders1 to avoid any data loss.

```
CREATE TABLE runner_orders1 SELECT * FROM runner_orders;
```

```sql
UPDATE runner_orders1
SET distance = REPLACE(distance, 'km', '');

UPDATE runner_orders1
SET distance = NULLIF(distance, 'null');

UPDATE runner_orders1
SET duration = REPLACE(duration, 'minutes', '');

UPDATE runner_orders1
SET duration = REPLACE(duration, 'mins', '');

UPDATE runner_orders1
SET duration = REPLACE(duration, 'minute', '');

UPDATE runner_orders1
SET duration = NULLIF(duration, 'null');

UPDATE runner_orders1
SET pickup_time = NULLIF(pickup_time, 'null');

UPDATE runner_orders1
SET cancellation = NULLIF(cancellation, 'null');

UPDATE runner_orders1
SET cancellation = NULLIF(cancellation, '');
```

- Updating Datatype of runner_orders1

```sql
ALTER TABLE runner_orders1

 MODIFY COLUMN pickup_time datetime NULL,
 MODIFY COLUMN distance decimal(5,1) NULL,
 MODIFY COLUMN duration int NULL;
```

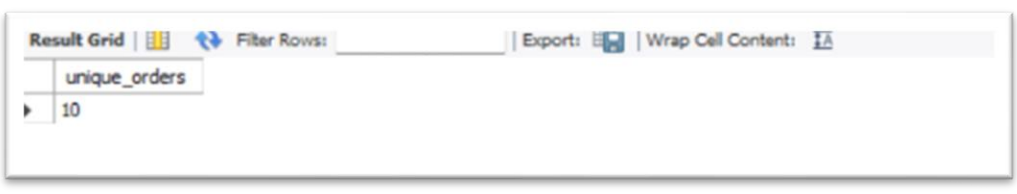
A. Pizza Metrics

1. How many pizzas were ordered?

```sql
SELECT
    COUNT(order_id) AS total_num_pizza
FROM
    customer_orders1;
```

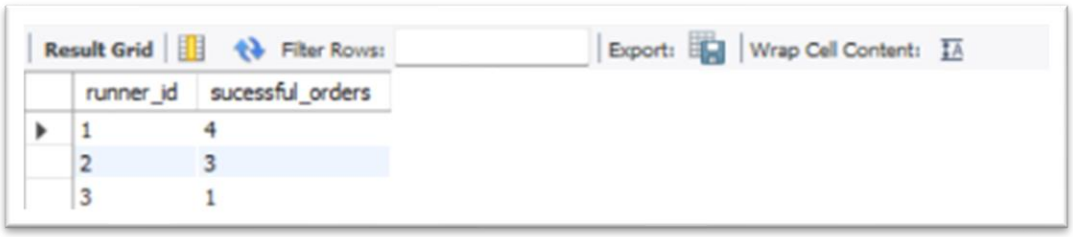
2. How many unique customer orders were made?

```sql
SELECT
    COUNT(DISTINCT (order_id)) AS unique_orders
FROM
    customer_orders1;
```



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

```sql
SELECT
    runner_id, COUNT(order_id) AS sucessful_orders
FROM
    runner_orders1
WHERE
    distance IS NOT NULL
GROUP BY runner_id;
```



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

runner_id	sucessful_orders
1	4
2	3
3	1

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

```sql
SELECT
    pn.pizza_name, COUNT(co1.order_id) AS num_of_pizza
FROM
    customer_orders1 co1
        JOIN
    pizza_names pn ON co1.pizza_id = pn.pizza_id
        JOIN
    runner_orders1 ro1 ON co1.order_id = ro1.order_id
WHERE
    distance IS NOT NULL
GROUP BY pn.pizza_name;
```

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

pizza_name	num_of_pizza
Meatlovers	9
Vegetarian	3

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

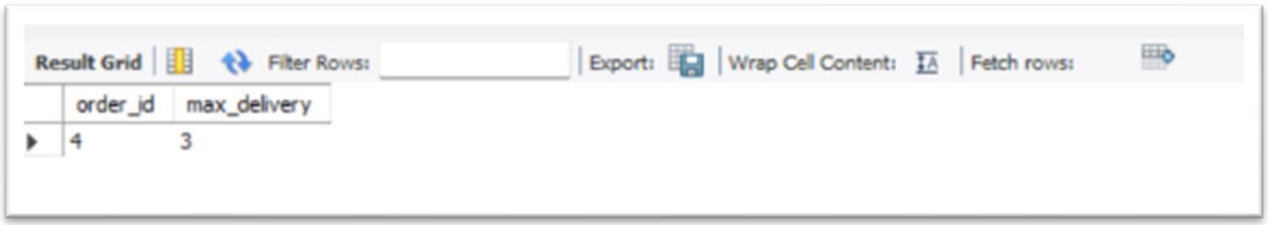
```sql
SELECT
    co1.customer_id,
    pn.pizza_name,
    COUNT(co1.order_id) num_of_pizza
FROM
    customer_orders1 co1
        JOIN
    pizza_names pn ON co1.pizza_id = pn.pizza_id
        JOIN
    runner_orders1 ro1 ON co1.order_id = ro1.order_id
GROUP BY pn.pizza_name , co1.customer_id
ORDER BY co1.customer_id;
```

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

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

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

```sql
SELECT
    co1.order_id, COUNT(co1.customer_id) AS max_delivery
FROM
    customer_orders1 co1
        JOIN
    runner_orders1 ro1 ON co1.order_id = ro1.order_id
WHERE
    distance IS NOT NULL
GROUP BY co1.order_id
ORDER BY max_delivery DESC
LIMIT 1;
```



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

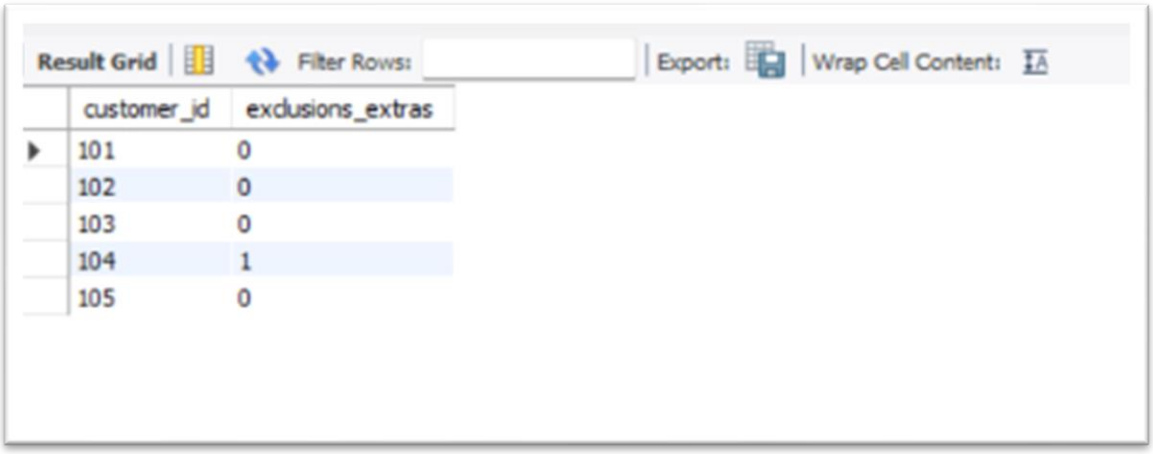
```sql
CREATE TEMPORARY TABLE changed_pizza (SELECT co1.order_id,
co1.customer_id, co1.exclusions, co1.extras FROM
customer_orders1 co1
JOIN runner_orders1 ro1
ON co1.order_id = ro1.order_id
WHERE cancellation IS NULL);

SELECT
    customer_id,
    exclusions,
    extras,
    SUM(CASE
        WHEN exclusions IS NOT NULL OR extras IS NOT NULL
        THEN 1
        ELSE 0
    END) AS atleast_one_change,
    SUM(CASE
        WHEN exclusions IS NULL AND extras IS NULL
        THEN 1
        ELSE 0
    END) AS no_change
FROM changed_pizza
GROUP BY customer_id;
```

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

customer_id	exclusions	extras	atleast_one_change	no_change
101	NULL	NULL	0	2
102	NULL	NULL	0	3
103	4	NULL	3	0
104	NULL	1	2	1
105	NULL	1	1	0

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

```sql
SELECT
    customer_id,
    SUM(CASE
        WHEN exclusions IS NOT NULL
                AND extras IS NOT NULL
        THEN 1
        ELSE 0
    END) AS exclusions_extras
FROM changed_pizza
GROUP BY customer_id;
```

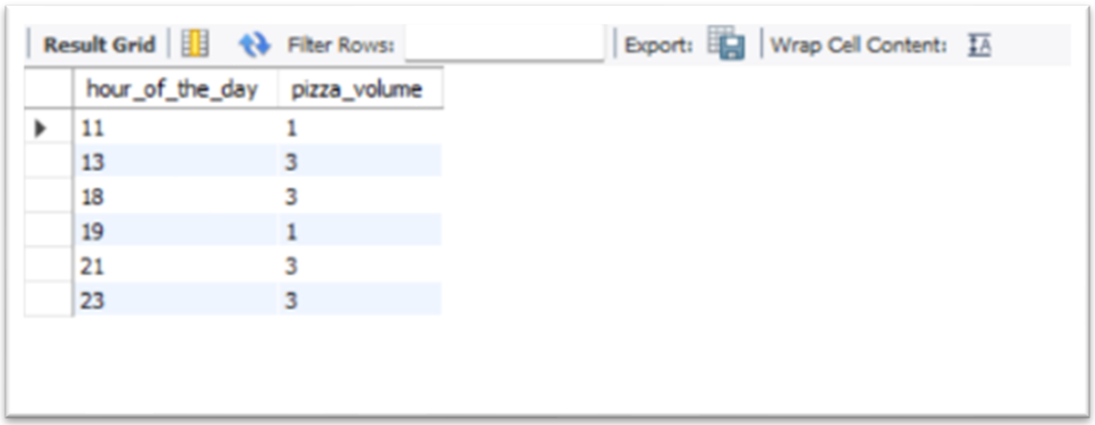


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	exclusions_extras
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
    HOUR(order_time) AS hour_of_the_day,
    COUNT(order_id) AS pizza_volume
FROM customer_orders1
GROUP BY hour_of_the_day
ORDER BY hour_of_the_day;
```

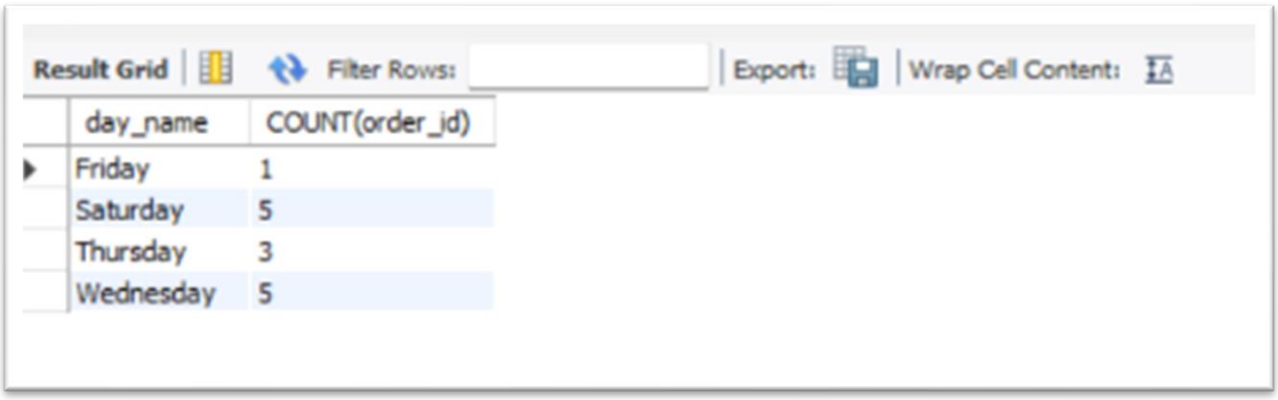


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

hour_of_the_day	pizza_volume
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?

```sql
SELECT
    DAYNAME(order_time) AS day_name, COUNT(order_id)
FROM
    customer_orders1
GROUP BY day_name
ORDER BY day_name;
```



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

day_name	COUNT(order_id)
Friday	1
Saturday	5
Thursday	3
Wednesday	5

B. 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) AS week_num,
    COUNT(runner_id) AS runner_signed_up
FROM
    runners
GROUP BY week_num;
```

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

week_num	runner_signed_up
0	1
1	2
2	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 ?

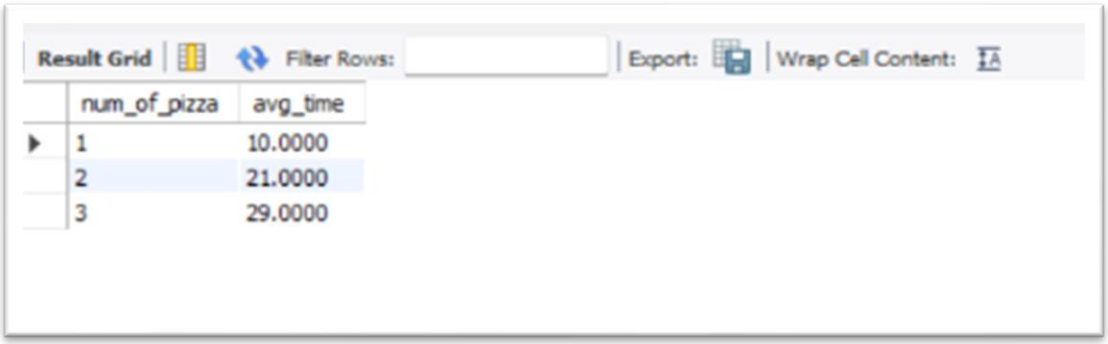
```sql
SELECT
    runner_id,
    AVG(EXTRACT(MINUTE FROM TIMEDIFF(pickup_time,
order_time))) AS avg_time
FROM
    runner_orders1 ro1
        JOIN
    customer_orders1 co1 ON ro1.order_id = co1.order_id
GROUP BY runner_id;
```

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

runner_id	avg_time
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
    num_of_pizza, avg_time
FROM
    (SELECT
        COUNT(co1.order_id) AS num_of_pizza,
            AVG(EXTRACT(MINUTE FROM TIMEDIFF(pickup_time,
order_time))) AS avg_time
    FROM
        customer_orders1 co1
    JOIN runner_orders1 ro1 ON co1.order_id = ro1.order_id
    GROUP BY co1.order_id) temp
GROUP BY num_of_pizza;
```

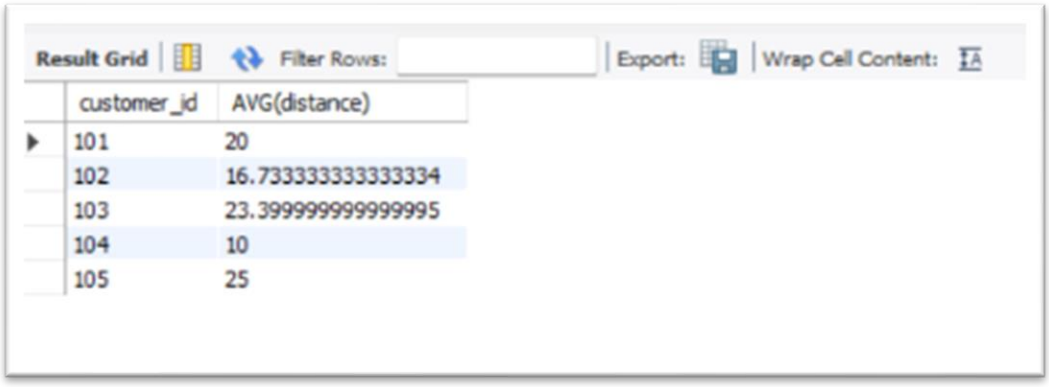


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

num_of_pizza	avg_time
1	10.0000
2	21.0000
3	29.0000

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

```sql
SELECT
    customer_id, AVG(distance)
FROM
    runner_orders1 ro1
        JOIN
    customer_orders1 co1 ON ro1.order_id = co1.order_id
GROUP BY customer_id;
```

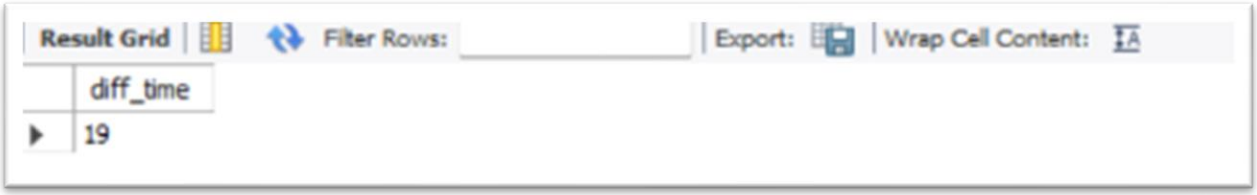


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	AVG(distance)
101	20
102	16.733333333333334
103	23.399999999999995
104	10
105	25

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

```sql
SELECT
    MAX(time_taken) - MIN(time_taken) AS diff_time
FROM
    (SELECT
        EXTRACT(MINUTE FROM TIMEDIFF(order_time,
pickup_time)) AS time_taken
    FROM
        runner_orders1 ro1
    JOIN customer_orders1 co1 ON ro1.order_id =
co1.order_id) temp;
```



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

diff_time
19

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

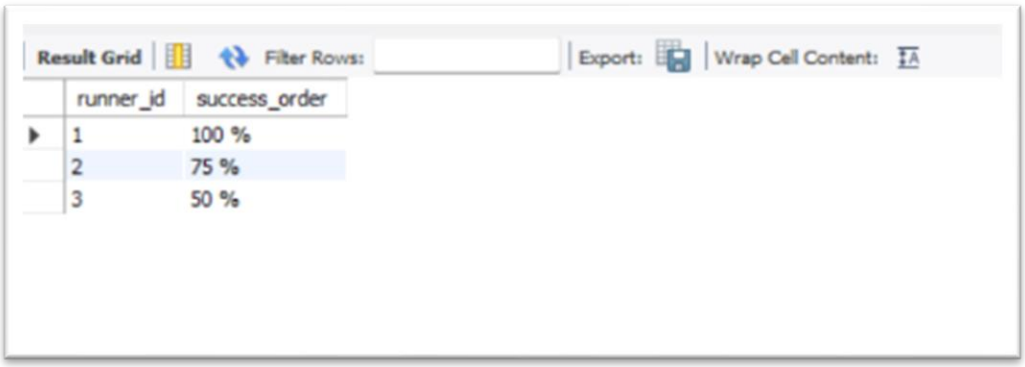
```sql
SELECT
    order_id,
    runner_id,
    CONCAT(ROUND((distance / duration * 60)),
            ' km/hr') AS speed
FROM
    runner_orders1
WHERE
    distance IS NOT NULL
ORDER BY runner_id;
```

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

order_id	runner_id	speed
1	1	38 km/hr
2	1	44 km/hr
3	1	40 km/hr
10	1	60 km/hr
4	2	35 km/hr
7	2	60 km/hr
8	2	94 km/hr
5	3	40 km/hr

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

```sql
SELECT
    runner_id,
    CONCAT(ROUND(100 * SUM(IF(cancellation IS NULL, 1, 0)) /
COUNT(*)),
           ' %') AS success_order
FROM
    runner_orders1
GROUP BY runner_id;
```



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

runner_id	success_order
1	100 %
2	75 %
3	50 %

C. Ingredient Optimisation

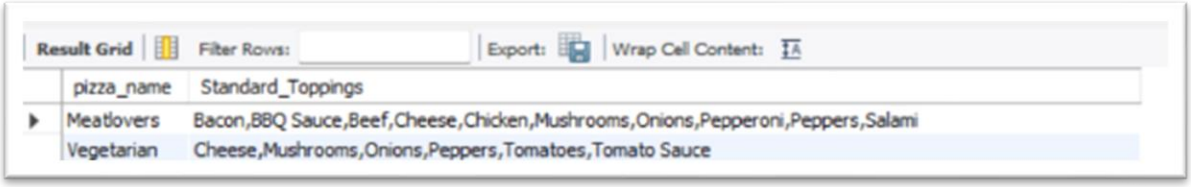
```sql
CREATE TABLE pizza_recipes_new (
    pizza_id INTEGER,
    toppings INTEGER
);
INSERT INTO pizza_recipes_new (pizza_id, toppings)
  VALUES (1, 1),
  (1, 2),
  (1, 3),
  (1, 4),
  (1, 5),
  (1, 6),
  (1, 7),
  (1, 8),
  (1, 9),
  (1, 10),
  (2, 4),
  (2, 6),
  (2, 7),
  (2, 9),
  (2, 11),
  (2, 12);
```

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

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

1. What are the standard ingredients for each pizza?

```sql
WITH cte AS ( SELECT pn.pizza_name, prn.pizza_id,
pt.topping_name
FROM pizza_recipes_new prn
JOIN pizza_toppings pt
ON prn.toppings = pt.topping_id
JOIN pizza_names pn
ON pn.pizza_id = prn.pizza_id
ORDER BY pizza_name, prn.pizza_id)
SELECT pizza_name, GROUP_CONCAT(topping_name) AS
Standard_Toppings
FROM cte
GROUP BY pizza_name;
```

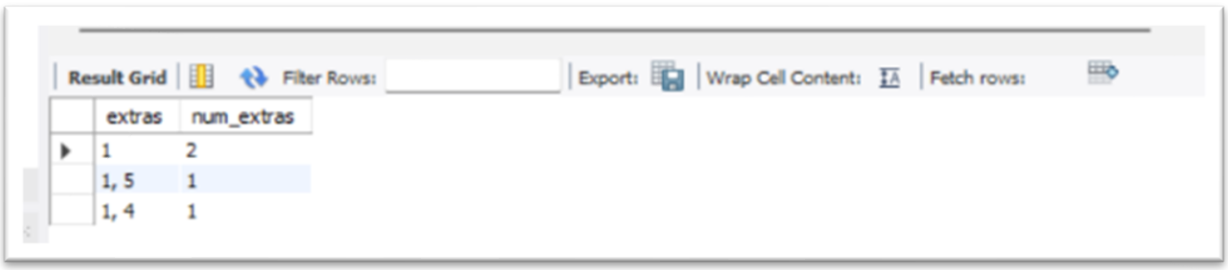


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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

2. What was the most commonly added extra?

```sql
SELECT
  extras,
  COUNT(*) as num_extras
FROM customer_orders1
GROUP BY extras
limit 4 offset 1;
```

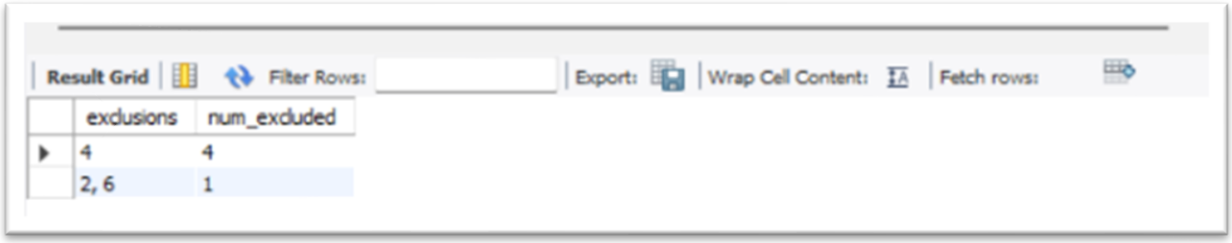


Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

extras	num_extras
1	2
1, 5	1
1, 4	1

3. What was the most common exclusion?

```sql
SELECT
    exclusions, COUNT(*) AS num_excluded
FROM
    customer_orders1
GROUP BY exclusions
LIMIT 3 OFFSET 1;
```



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

exclusions	num_excluded
4	4
2, 6	1

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

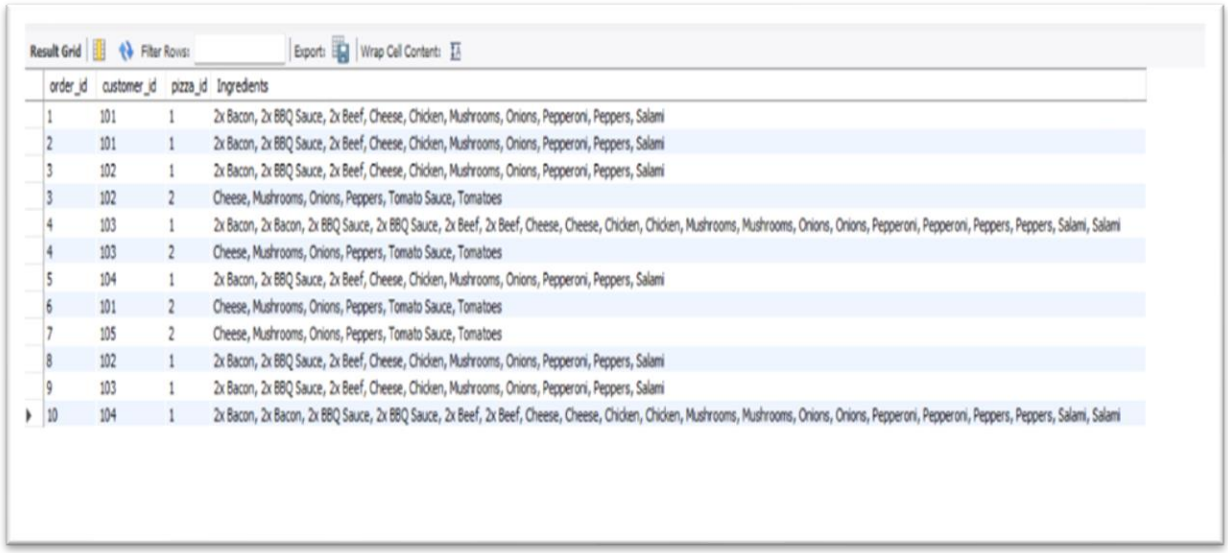
```sql
SELECT
    customer_orders1.order_id,
    customer_orders1.pizza_id,
    pizza_names.pizza_name,
    customer_orders1.exclusions,
    customer_orders1.extras,
    (CASE WHEN
            customer_orders1.pizza_id = 1
        THEN (CASE WHEN
                    customer_orders1.exclusions = 4
                        AND (customer_orders1.extras IS NULL
                        OR customer_orders1.extras = 0)
         THEN 'Meat Lovers - Exclude Cheese'
                WHEN
                    (customer_orders1.exclusions LIKE '%3%'
                        OR customer_orders1.exclusions = 3)
                        AND (customer_orders1.extras IS NULL
                        OR customer_orders1.extras = 0)
          THEN 'Meat Lovers - Exclude Beef'
                WHEN
                    customer_orders1.extras = 1
                    AND (customer_orders1.exclusions IS NULL
                    OR customer_orders1.exclusions = 0)
         THEN 'Meat Lovers - Extra Bacon'
                WHEN
                   customer_orders1.exclusions = '1, 4'
                        AND customer_orders1.extras = '6, 9'
          THEN 'Meat Lovers - Exclude Cheese, Bacon - Extra
Mushroom, Peppers'
                WHEN
                    customer_orders1.exclusions = '2, 6'
                        AND customer_orders1.extras = '1, 4'
          THEN 'Meat Lovers - Exclude BBQ Sauce, Mushroom -
Extra Bacon, Cheese'
                WHEN
                    customer_orders1.exclusions = 4
                        AND customer_orders1.extras = '1, 5'
          THEN 'Meat Lovers - Exclude Cheese - Extra Bacon,
Chicken'
              ELSE 'Meat Lovers'  END)
        WHEN customer_orders1.pizza_id = 2
        THEN CASE WHEN
                    customer_orders1.exclusions = 4
                        AND (customer_orders1.extras IS NULL
                        OR customer_orders1.extras = 0)
                THEN  'Veg Lovers - Exclude Cheese'
                ELSE 'Veg Lovers' END
    END) OrderItem
FROM  customer_orders1
INNER JOIN pizza_names
ON pizza_names.pizza_id = customer_orders1.pizza_id;
```

order_id	pizza_id	pizza_name	exclusions	extras	OrderItem
1	1	Meatlovers	NULL	NULL	Meat Lovers
2	1	Meatlovers	NULL	NULL	Meat Lovers
3	1	Meatlovers	NULL	NULL	Meat Lovers
3	2	Vegetarian	NULL	NULL	Veg Lovers
4	1	Meatlovers	4	NULL	Meat Lovers - Exclude Cheese
4	1	Meatlovers	4	NULL	Meat Lovers - Exclude Cheese
4	2	Vegetarian	4	NULL	Veg Lovers - Exclude Cheese
5	1	Meatlovers	NULL	1	Meat Lovers - Extra Bacon
6	2	Vegetarian	NULL	NULL	Veg Lovers
7	2	Vegetarian	NULL	1	Veg Lovers
8	1	Meatlovers	NULL	NULL	Meat Lovers
9	1	Meatlovers	4	1, 5	Meat Lovers - Exclude Cheese - Extra Bacon, Chicken
10	1	Meatlovers	NULL	NULL	Meat Lovers
10	1	Meatlovers	2, 6	1, 4	Meat Lovers - Exclude BBQ Sauce, Mushroom - Extra Bacon, Cheese

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"

```sql
SELECT
    cust.order_id,
    cust.customer_id,
    cust.pizza_id,
    GROUP_CONCAT(CASE
            WHEN recipe.toppings IN (1 , 2, 3) THEN
CONCAT('2x ', topping.topping_name)
            ELSE topping.topping_name
        END
        ORDER BY topping.topping_name
        SEPARATOR ', ') AS Ingredients
FROM
    customer_orders1 cust
        INNER JOIN
    pizza_recipes_new recipe ON cust.pizza_id =
recipe.pizza_id
        INNER JOIN
    pizza_toppings topping ON topping.topping_id =
recipe.toppings
GROUP BY cust.order_id , cust.customer_id , cust.pizza_id;
```

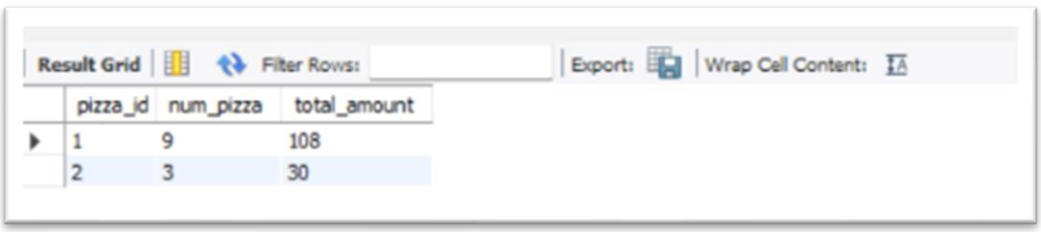


order_id	customer_id	pizza_id	Ingredients
1	101	1	2x Bacon, 2x BBQ Sauce, 2x Beef, Cheese, Chicken, Mushrooms, Onions, Pepperoni, Peppers, Salami
2	101	1	2x Bacon, 2x BBQ Sauce, 2x Beef, Cheese, Chicken, Mushrooms, Onions, Pepperoni, Peppers, Salami
3	102	1	2x Bacon, 2x BBQ Sauce, 2x Beef, Cheese, Chicken, Mushrooms, Onions, Pepperoni, Peppers, Salami
3	102	2	Cheese, Mushrooms, Onions, Peppers, Tomato Sauce, Tomatoes
4	103	1	2x Bacon, 2x Bacon, 2x BBQ Sauce, 2x BBQ Sauce, 2x Beef, 2x Beef, Cheese, Cheese, Chicken, Chicken, Mushrooms, Mushrooms, Onions, Onions, Pepperoni, Pepperoni, Peppers, Peppers, Salami, Salami
4	103	2	Cheese, Mushrooms, Onions, Peppers, Tomato Sauce, Tomatoes
5	104	1	2x Bacon, 2x BBQ Sauce, 2x Beef, Cheese, Chicken, Mushrooms, Onions, Pepperoni, Peppers, Salami
6	101	2	Cheese, Mushrooms, Onions, Peppers, Tomato Sauce, Tomatoes
7	105	2	Cheese, Mushrooms, Onions, Peppers, Tomato Sauce, Tomatoes
8	102	1	2x Bacon, 2x BBQ Sauce, 2x Beef, Cheese, Chicken, Mushrooms, Onions, Pepperoni, Peppers, Salami
9	103	1	2x Bacon, 2x BBQ Sauce, 2x Beef, Cheese, Chicken, Mushrooms, Onions, Pepperoni, Peppers, Salami
10	104	1	2x Bacon, 2x Bacon, 2x BBQ Sauce, 2x BBQ Sauce, 2x Beef, 2x Beef, Cheese, Cheese, Chicken, Chicken, Mushrooms, Mushrooms, Onions, Onions, Pepperoni, Pepperoni, Peppers, Peppers, Salami, Salami

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?

```sql
SELECT
    pizza_id,
    COUNT(pizza_id) AS num_pizza,
    CASE
        WHEN pizza_id = 1 THEN COUNT(pizza_id) * 12
        ELSE COUNT(pizza_id) * 10
    END AS total_amount
FROM
    customer_orders1 co1
        JOIN
    runner_orders1 ro1 ON co1.order_id = ro1.order_id
WHERE
    distance IS NOT NULL
GROUP BY pizza_id;
```

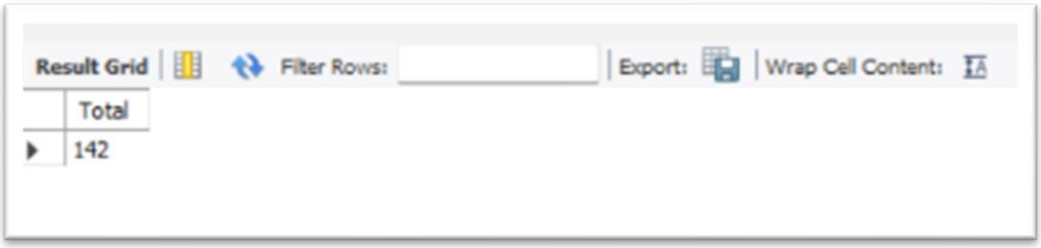


pizza_id	num_pizza	total_amount
1	9	108
2	3	30

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

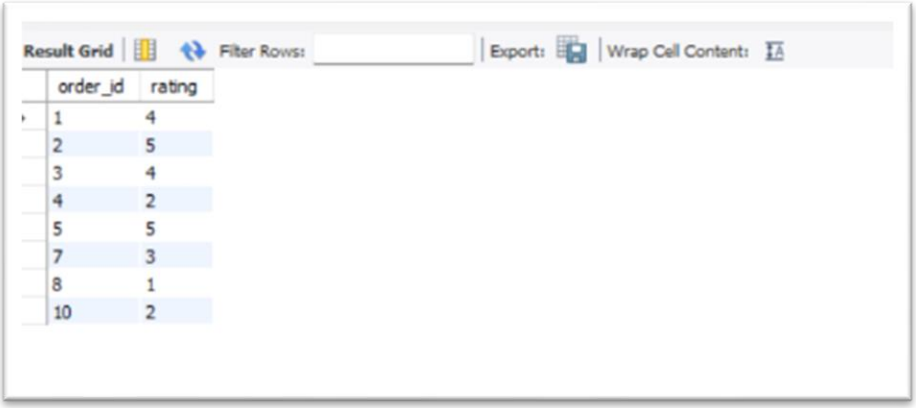
Add cheese is $1 extra

```
set @basecost = 138;
SELECT
    (LENGTH(GROUP_CONCAT(extras)) -
LENGTH(REPLACE(GROUP_CONCAT(extras), ',', '')) + 1) +
@basecost AS Total
FROM
    customer_orders1
        INNER JOIN
    runner_orders1 ON customer_orders1.order_id =
runner_orders1.order_id
WHERE
    extras IS NOT NULL
        AND distance IS NOT NULL;
```



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.

```sql
CREATE TABLE ratings (
    order_id INTEGER,
    rating INTEGER
);
insert into ratings
(order_id, rating)
values
(1,4),
(2,5),
(3,4),
(4,2),
(5,5),
(7,3),
(8,1),
(10,2);

SELECT
    *
FROM
    ratings;
```



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

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

```sql
SELECT
    cO1.customer_id,
    co1.order_id,
    ro1.runner_id,
    ratings.rating,
    co1.order_time,
    ro1.pickup_time,
    TIMESTAMPDIFF(MINUTE,
        order_time,
        pickup_time) AS Order_pickup_time,
    ro1.duration,
    ROUND(AVG(ro1.distance * 60 / ro1.duration), 1) AS
avg_speed,
    COUNT(co1.pizza_id) AS pizza_count
FROM
    customer_orders1 co1
        JOIN
    runner_orders1 ro1 ON co1.order_id = ro1.order_id
        JOIN
    ratings ON ratings.order_id = co1.order_id
GROUP BY co1.customer_id , co1.order_id , ro1.runner_id ,
ratings.rating , co1.order_time , ro1.pickup_time ,
Order_pickup_time , ro1.duration
ORDER BY customer_id;
```

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

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

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?

```
set @pizzaamountearned = 138;
select @pizzaamountearned - (sum(distance))*0.3 as
amount_left
from runner_orders1;
```

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

amount_left
94.44

E. Bonus Questions

1. 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?

```sql
INSERT INTO pizza_names VALUES(3, 'Supreme');


SELECT * FROM pizza_names;


INSERT INTO pizza_recipes
VALUES(3, (SELECT GROUP_CONCAT(topping_id SEPARATOR ', ')
FROM pizza_toppings));


SELECT * FROM pizza_recipes;
```

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

pizza_id	pizza_name
1	Meatlovers
2	Vegetarian
3	Supreme

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

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