SQL PROJECT

PIZZA RUNNER BUSINESS ANALYSIS

PRESENTED BY
DENIS M

INTRODUCTION

- Danny launched Pizza Runner after noticing an increasing demand for pizzas.
- Danny wants to expand his new Pizza Empire.
- So he started recruiting “runners” to deliver fresh pizza from Pizza Runner Headquarters to customers.
- The Case Study is going to provide insights on Pizzas, Runner and customer experiences, ingredients, and pricing strategies to Danny.

AVAILABLE DATA

Danny had a few years of experience as a Data Scientist, he prepared an entity relationship diagram of his database design but requires further assistance to clean his data and apply some calculations so he could optimize Pizza Runner's operations.

Entity Relationship Diagram

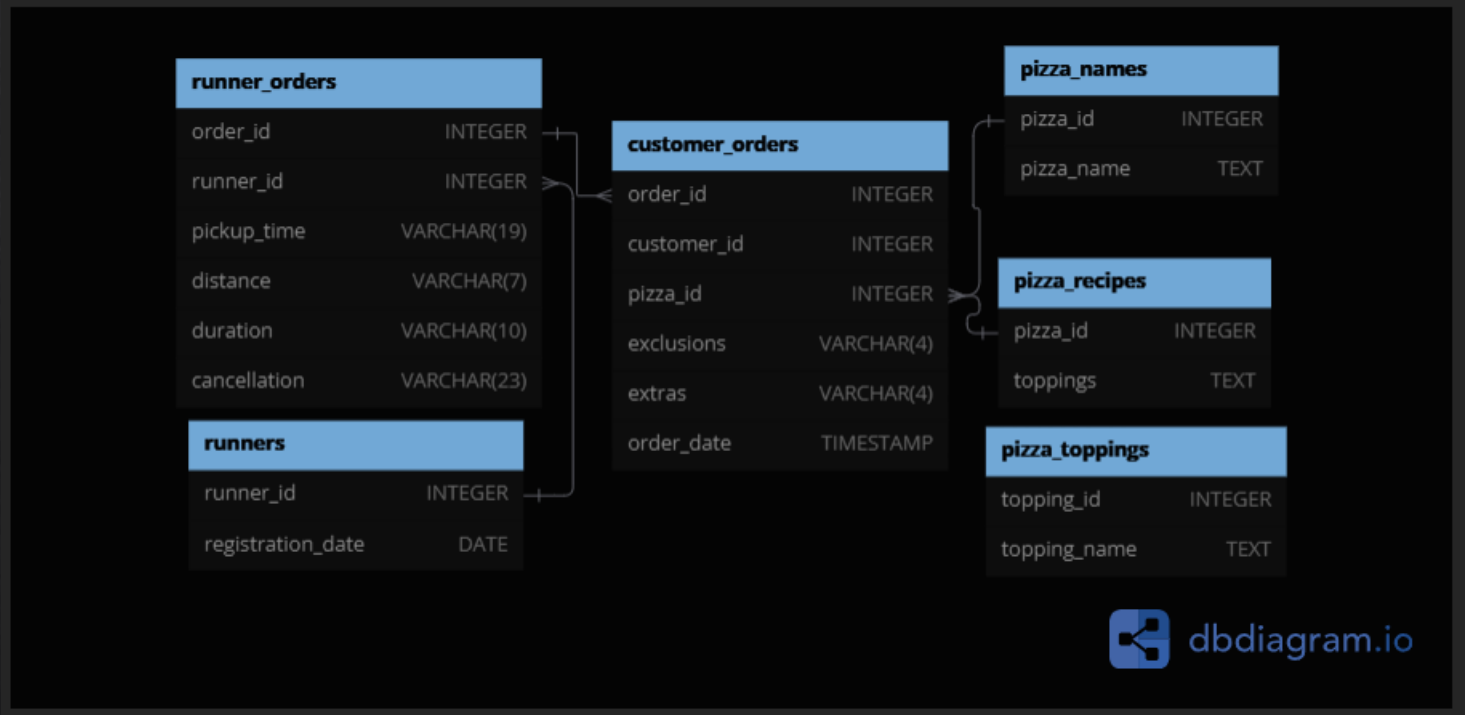


TABLE 1: RUNNERS

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 table with 1 row for each individual pizza that is part customer orders 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 C
7	2	2020-01-08 21:30:45	25km	25mins	null
8	2	2020-01-10 00:15:02	23.4 km	15 minute	null
9	2	null	null	null	Customer Ca
10	1	2020-01-11 18:50:20	10km	10minutes	null

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

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 Optimization
- ❑Pricing and Ratings

CASE STUDY QUESTIONS : PIZZA METRICS

1.How many pizzas were ordered?

```sql
SELECT COUNT(pizza_id) AS Total_pizzas
FROM customer_orders;
```

	Total_pizzas
▶	14

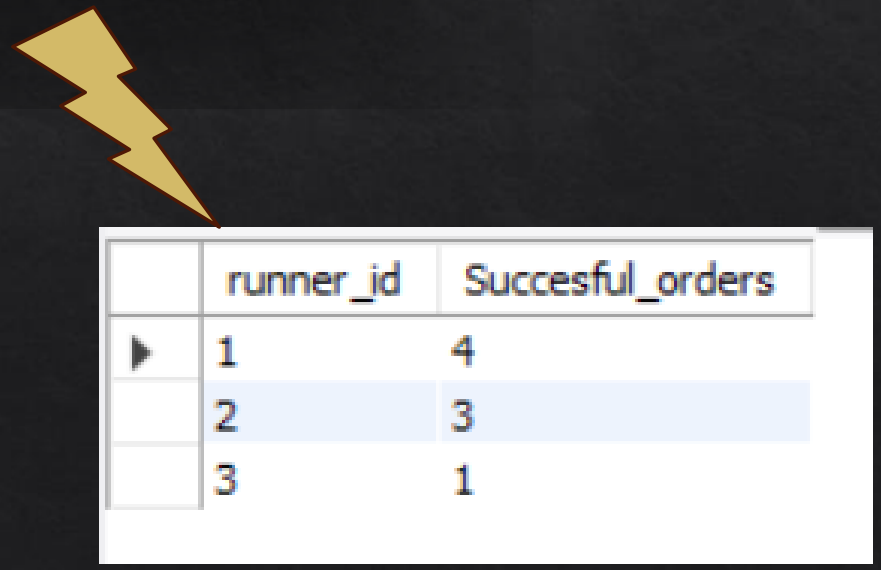
2.How many unique customer orders were made?

```sql
SELECT customer_id, COUNT(DISTINCT order_id) AS unique_orders
FROM customer_orders
GROUP BY customer_id;
```

	unique_orders
▶	10

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

```sql
SELECT ro.runner_id,COUNT(DISTINCT ro.order_id) AS Succesful_orders
FROM runner_orders ro
JOIN runners r
ON ro.runner_id=r.runner_id
WHERE cancellation IS NULL
GROUP BY ro.runner_id;
```



runner_id	Succesful_orders
1	4
2	3
3	1

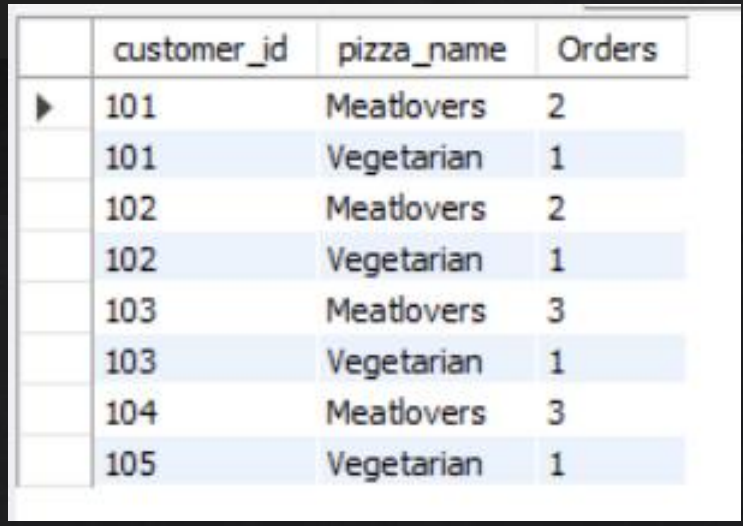
4.How many of each type of pizza was delivered?

```sql
SELECT p.pizza_name, COUNT(*) AS Pizza_delivered
FROM customer_orders co
JOIN runner_orders ro
ON co.order_id=ro.order_id
JOIN pizza_names p
ON co.pizza_id=p.pizza_id
WHERE ro.cancellation IS NULL
GROUP BY p.pizza_name;
```

pizza_name	Pizza_delivered
Meatlovers	9
Vegetarian	3

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

```sql
SELECT co.customer_id,p.pizza_name, COUNT(*) AS Orders
FROM customer_orders co
JOIN pizza_names p
ON co.pizza_id=p.pizza_id
GROUP BY co.customer_id,p.pizza_name
ORDER BY co.customer_id,p.pizza_name;
```



customer_id	pizza_name	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?

```sql
SELECT co.order_id, COUNT(*) AS Orders
FROM customer_orders co
JOIN runner_orders ro
ON co.order_id=ro.order_id
WHERE ro.cancellation IS NULL
GROUP BY co.order_id
ORDER BY Orders DESC
LIMIT 1;
```

	order_id	Orders
▶	4	3

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

```sql
SELECT
    SUM(CASE
            WHEN (co.exclusions IS NOT NULL OR co.extras IS NOT NULL) THEN 1
            ELSE 0
        END) AS Pizzas_with_1_change,
    SUM(CASE
            WHEN (co.exclusions IS NULL AND co.extras IS NULL) THEN 1
            ELSE 0
        END) AS Pizzas_with_no_change
FROM customer_orders co
JOIN runner_orders ro
    ON co.order_id = ro.order_id
WHERE ro.cancellation IS NULL;
```

	Pizzas_with_1_change	Pizzas_with_no_change
▶	6	6

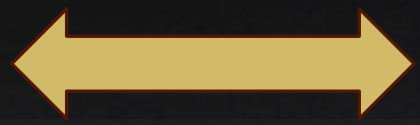
8.How many pizzas were delivered that had both exclusions and extras?

```sql
SELECT COUNT(*) AS Orders
FROM runner_orders ro
JOIN customer_orders co
ON ro.order_id=co.order_id
WHERE co.exclusions AND co.extras IS NOT NULL AND ro.cancellation IS NULL;
```

	Orders
▶	1

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

```sql
WITH CTE AS (
SELECT *,HOUR(order_time) AS `Hour`
FROM customer_orders)
SELECT `Hour`,COUNT(*) AS Orders
FROM CTE
GROUP BY `Hour`
ORDER BY `Hour`;
```



	Hour	Orders
▶	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
WITH CTE AS (
SELECT *,DAYOFWEEK(order_time) AS Day_of_week
FROM customer_orders)
SELECT Day_of_week,COUNT(*) AS Orders
FROM CTE
GROUP BY Day_of_week;
```

	Day_of_week	Orders
▶	4	5
	5	3
	7	5
	6	1

INSIGHTS : Pizza Metrics

- ✓ 14 pizzas were ordered in 10 orders.
- ✓ Runner-1 has made the most successful deliveries followed by Runner-2.
- ✓ 9 Meat-lovers and 3 Vegetarian-type pizzas are successfully delivered.
- ✓ Order-4 has ordered the most Meat-lovers pizzas and order-5 ordered only Vegetarian type.
- ✓ Maximum number of pizzas delivered is 3 for order-4.
- ✓ 6 Pizza orders made changes and 6 with no change.
- ✓ Only 1 pizza order had both exclusions and extras
- ✓ Higher orders are placed during hours 13,18,21 and 23.
- ✓ During weekdays most orders are placed on day 4 and day 7.

CASE STUDY QUESTIONS : RUNNER AND CUSTOMER EXPERIENCE

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

```sql
SELECT COUNT(*) Runners_signed
FROM runners
WHERE registration_date <= DATE_ADD("2021-01-01", INTERVAL 7 DAY);
```

	Runners_signed
▶	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?

```sql
SELECT ro.runner_id,
ROUND(AVG(TIMESTAMPDIFF(MINUTE,co.order_time, ro.pickup_time))) AS avg_time_in_minutes
FROM runner_orders ro
JOIN customer_orders co
ON ro.order_id=co.order_id
WHERE ro.cancellation IS NULL
GROUP BY ro.runner_id;
```

runner_id	avg_time_in_minutes
1	15
2	23
3	10

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

```sql
WITH CTE AS (
SELECT ro.order_id,ro.runner_id,
TIMESTAMPDIFF(MINUTE,co.order_time, ro.pickup_time) AS time_taken
FROM customer_orders co
JOIN runner_orders ro
ON co.order_id=ro.order_id
WHERE ro.cancellation IS NULL)
SELECT time_taken AS Order_preparation_time, COUNT(*) AS Orders
FROM CTE
GROUP BY time_taken
ORDER  BY time_taken;
```

Order_preparation_time	Orders
10	4
15	2
20	1
21	2
29	3

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

```sql
SELECT co.customer_id, ROUND(AVG(ro.distance)) AS Avg_distance_in_KM
FROM customer_orders co
JOIN runner_orders ro
ON co.order_id=ro.order_id
GROUP BY co.customer_id;
```

customer_id	Avg_distance_in_KM
101	20
102	16
103	23
104	10
105	25

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

```sql
WITH CTE AS (
SELECT co.order_id,ro.runner_id,ro.duration
FROM runner_orders ro
JOIN customer_orders co
ON ro.order_id=co.order_id
WHERE ro.cancellation IS NULL)
SELECT order_id,
MAX(duration) AS longest_delivery_time_in_minutes,
MIN(duration) AS shortest_delivery_time_in_minutes
FROM CTE
GROUP BY order_id;
```

order_id	longest_delivery_time_in_minutes	shortest_delivery_time_in_minutes
1	32	32
2	27	27
3	20	20
4	40	40
5	15	15
7	25	25
8	15	15
10	10	10

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

```sql
WITH CTE AS (
SELECT ro.*,(ro.distance/ro.duration) AS Speed_KM_per_minute
FROM runner_orders ro
JOIN customer_orders co
ON ro.order_id=co.order_id
WHERE ro.cancellation IS NULL)
SELECT order_id, ROUND(AVG(Speed_KM_per_minute),3) AS Avg_Speed_KM_per_minute
FROM CTE
GROUP BY order_id;
```

order_id	Avg_Speed_KM_per_minute
1	0.625
2	0.741
3	0.650
4	0.575
5	0.667
7	1.000
8	1.533
10	1.000

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

```sql
WITH CTE1 AS (
SELECT co.order_id,co.customer_id,r.runner_id,ro.pickup_time,ro.cancellation
FROM  customer_orders co
JOIN  runner_orders ro
ON co.order_id=ro.order_id
RIGHT JOIN runners r
ON ro.runner_id=r.runner_id),
CTE2 AS (
SELECT runner_id , COUNT(order_id) AS Total_orders
FROM CTE1
GROUP BY runner_id),
CTE3 AS
(
SELECT runner_id , COUNT(order_id) AS Total_successfull_orders
FROM CTE1
WHERE cancellation IS NULL
GROUP BY runner_id
)
SELECT t2.runner_id,
ROUND((Total_successfull_orders/Total_orders)*100) AS successful_delivery_percentage
FROM CTE2 t2
JOIN CTE3 t3
ON t2.runner_id=t3.runner_id;
```

runner_id	successful_delivery_percentage
1	100
2	83
3	50
4	NULL

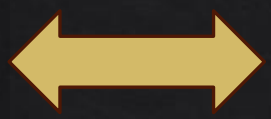
INSIGHTS : Runner and Customer Experience

- ✓ 3 Runners signed up for the first week.
- ✓ An average runner-3 takes 10 minutes to pick up the pizza, runner-1 takes 15 minutes to pick up the pizza, and runner-2 takes 15 minutes to pick up the pizza
- ✓ Per order on average pizza runners have to travel 18.8 KM for order delivery.

CASE STUDY QUESTIONS : INGREDIENT OPTIMIZATION

```sql
SELECT topping_name
FROM pizza_toppings
WHERE topping_id IN (
WITH RECURSIVE SplitToppings AS (
  SELECT
    pizza_id,
    TRIM(SUBSTRING_INDEX(toppings, ',', 1)) AS topping_id,
    CASE
      WHEN LOCATE(',', toppings) > 0 THEN TRIM(SUBSTRING(toppings, LOCATE(',', toppings) + 1))
      ELSE NULL
    END AS rest
  FROM pizza_recipes
  UNION ALL
  SELECT
    pizza_id,
    TRIM(SUBSTRING_INDEX(rest, ',', 1)) AS topping_id,
    CASE
      WHEN LOCATE(',', rest) > 0 THEN TRIM(SUBSTRING(rest, LOCATE(',', rest) + 1))
      ELSE NULL
    END AS rest
  FROM SplitToppings
  WHERE rest IS NOT NULL AND rest != ''
)
SELECT
  topping_id
FROM
  SplitToppings
GROUP BY
  topping_id
HAVING
  COUNT(DISTINCT pizza_id) = 2);
```

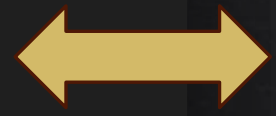
1.How many runners signed up for each 1 week period? (i.e. week starts 2021-01-01)



	topping_name
▶	Cheese
	Mushrooms

2.What was the most commonly added extra?

```sql
SELECT topping_name
FROM pizza_toppings
WHERE topping_id IN (
WITH RECURSIVE Splitextras AS (
  SELECT
    pizza_id,
    TRIM(SUBSTRING_INDEX(extras, ',', 1)) AS topping_id,
    CASE
      WHEN LOCATE(',', extras) > 0 THEN TRIM(SUBSTRING(extras, LOCATE(',', extras) + 1))
      ELSE NULL
    END AS rest
  FROM customer_orders
  UNION ALL
  SELECT
    pizza_id,
    TRIM(SUBSTRING_INDEX(rest, ',', 1)) AS topping_id,
    CASE
      WHEN LOCATE(',', rest) > 0 THEN TRIM(SUBSTRING(rest, LOCATE(',', rest) + 1))
      ELSE NULL
    END AS rest
  FROM Splitextras
  WHERE rest IS NOT NULL AND rest != ''
)
SELECT topping_id
FROM
  Splitextras
WHERE topping_id IS NOT NULL
GROUP BY topping_id
HAVING COUNT(DISTINCT pizza_id)>1);
```



	topping_name
▶	Bacon

```sql
SELECT topping_name
FROM pizza_toppings
WHERE topping_id IN (
WITH RECURSIVE Splitexclusions  AS (
  SELECT
    pizza_id,
    TRIM(SUBSTRING_INDEX(exclusions, ',', 1)) AS topping_id,
    CASE
      WHEN LOCATE(',', exclusions) > 0 THEN TRIM(SUBSTRING(exclusions, LOCATE(',', exclusions) +
1))
      ELSE NULL
    END AS rest
  FROM customer_orders
  UNION ALL
  SELECT
    pizza_id,
    TRIM(SUBSTRING_INDEX(rest, ',', 1)) AS topping_id,
    CASE
      WHEN LOCATE(',', rest) > 0 THEN TRIM(SUBSTRING(rest, LOCATE(',', rest) + 1))
      ELSE NULL
    END AS rest
  FROM Splitexclusions
  WHERE rest IS NOT NULL AND rest != ''
)
SELECT topping_id
FROM Splitexclusions
WHERE topping_id IS NOT NULL
GROUP BY topping_id
HAVING COUNT(pizza_id)>1);
```

3. What was the most common exclusion?

	topping_name
▶	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 Bacon Mushrooms, Extra Bacon Cheese

```sql
SELECT co.order_id,co.customer_id,
CASE
  WHEN co.pizza_id=1 AND co.exclusions IS NULL AND co.extras IS NULL THEN "Meat Lovers"
    WHEN co.pizza_id=1 AND co.exclusions=3 AND  co.extras IS NOT NULL THEN "Meat Lovers - Exclude Beef"
    WHEN co.pizza_id=1 AND co.exclusions IS NULL AND co.extras=1 THEN "Meat Lovers - Extra Bacon"
    WHEN co.pizza_id=1 AND co.exclusions=4   AND co.extras IS NULL THEN "Meat Lovers - Exclude Cheese"
    WHEN co.pizza_id=1 AND co.exclusions=4   AND co.extras=1 THEN "Meat Lovers - Exclude Cheese,Extra Bacon"
    WHEN co.pizza_id=1 AND co.exclusions=2   AND co.extras=1 THEN "Meat Lovers - Exclude Bacon Mushrooms,Extra Bacon Cheese"
    WHEN co.pizza_id=2 AND co.exclusions IS NULL AND co.extras IS NULL THEN "Vegetable Lovers"
    WHEN co.pizza_id=2 AND co.exclusions=4   AND co.extras IS NULL THEN "Vegetable Lovers - Exclude Cheese"
    WHEN co.pizza_id=2 AND co.exclusions IS NULL  AND co.extras=1 THEN "Vegetable Lovers - Extra Cheese"
    ELSE null
END AS Customer_classification
FROM customer_orders co
JOIN pizza_names p
ON co.pizza_id=p.pizza_id;
```

order_id	customer_id	Customer_classification
1	101	Meat Lovers
2	101	Meat Lovers
3	102	Meat Lovers
3	102	Vegetable Lovers
4	103	Meat Lovers - Exclude Cheese
4	103	Meat Lovers - Exclude Cheese
4	103	Vegetable Lovers - Exclude Cheese
5	104	Meat Lovers - Extra Bacon
6	101	Vegetable Lovers
7	105	Vegetable Lovers - Extra Cheese
8	102	Meat Lovers
9	103	Meat Lovers - Exclude Cheese,Extra Bacon
10	104	Meat Lovers
10	104	Meat Lovers - Exclude Bacon Mushrooms,Extra Bacon Cheese

5.Generate an alphabetically ordered comma separated ingredient list for each pizza order from the customer orders table

```sql
WITH CTE_incrediants as (
WITH SplitToppings AS (
  SELECT
    pr.pizza_id,
    TRIM(SUBSTRING_INDEX(SUBSTRING_INDEX(pr.toppings, ',', numbers.n), ',', -1)) AS topping_id
  FROM
    (SELECT 1 AS n UNION SELECT 2 UNION SELECT 3 UNION SELECT 4 UNION SELECT 5 UNION SELECT 6 UNION
SELECT 7 UNION SELECT 8 UNION SELECT 9 UNION SELECT 10 UNION SELECT 11 UNION SELECT 12) AS numbers
  JOIN pizza_recipes pr
    ON CHAR_LENGTH(pr.toppings) - CHAR_LENGTH(REPLACE(pr.toppings, ',', '')) >= numbers.n - 1
)
SELECT
  pr.pizza_id,
  GROUP_CONCAT(pt.topping_name ORDER BY pt.topping_name ASC) AS ingredients
FROM
  SplitToppings st
JOIN pizza_toppings pt
  ON st.topping_id = pt.topping_id
JOIN pizza_recipes pr
  ON st.pizza_id = pr.pizza_id
GROUP BY
  pr.pizza_id)
  SELECT co.order_id,co.customer_id,c.ingredients
  FROM customer_orders co
  JOIN CTE_incrediants c
  ON co.pizza_id = c.pizza_id;
```

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

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

```sql
WITH SplitToppings AS (
  SELECT
    TRIM(SUBSTRING_INDEX(SUBSTRING_INDEX(pr.toppings, ',', numbers.n), ',', -1)) AS topping_id
  FROM
    (SELECT 1 AS n UNION SELECT 2 UNION SELECT 3 UNION SELECT 4 UNION SELECT 5 UNION SELECT 6 UNION
SELECT 7 UNION SELECT 8 UNION SELECT 9 UNION SELECT 10 UNION SELECT 11 UNION SELECT 12) AS numbers
  JOIN pizza_recipes pr
    ON CHAR_LENGTH(pr.toppings) - CHAR_LENGTH(REPLACE(pr.toppings, ',', '')) >= numbers.n - 1
  JOIN customer_orders co
  ON pr.pizza_id=co.pizza_id)
SELECT
  pt.topping_name,
  COUNT(st.topping_id) AS quantity_used
FROM
  SplitToppings st
JOIN pizza_toppings pt
  ON st.topping_id = pt.topping_id
GROUP BY
  pt.topping_name
ORDER BY
  quantity_used DESC;
```

topping_name	quantity_used
Cheese	14
Mushrooms	14
Bacon	10
BBQ Sauce	10
Beef	10
Chicken	10
Pepperoni	10
Salami	10
Onions	4
Peppers	4
Tomatoes	4
Tomato Sauce	4

INSIGHTS : Ingredient Optimisation

- ✓ Cheese and Mushrooms are the standard ingredients for both pizzas
- ✓ The commonly added extra’s to the pizzas is Bacon
- ✓ The most common exclusion is Cheese
- ✓ Cheese and Mushrooms are the highly used ingredients in terms of quantity.

CASE STUDY QUESTIONS : 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
WITH CTE_costs AS (
SELECT co.order_id,co.pizza_id,ro.runner_id,
CASE
  WHEN co.pizza_id=1 THEN 12
  ELSE 10
END AS Cost
FROM customer_orders co
JOIN runner_orders ro
ON co.order_id=ro.order_id
WHERE ro.cancellation IS NULL
)
SELECT runner_id,SUM(Cost) AS Sales
FROM CTE_costs
GROUP BY runner_id;
```

runner_id	Sales
1	70
2	56
3	12

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

```sql
WITH CTE_additional_costs AS (
SELECT co.order_id,co.pizza_id,ro.runner_id,
CASE
  WHEN co.pizza_id=1 AND co.extras IS NULL THEN 12
  WHEN co.pizza_id=2 AND co.extras IS NULL THEN 10
  WHEN co.pizza_id=1 AND co.extras=1 AND LENGTH(co.extras)=1  THEN 12+1
  WHEN co.pizza_id=2 AND co.extras=1 AND LENGTH(co.extras)=1 THEN 12+1
  WHEN co.pizza_id=1 AND LENGTH(co.extras)>1 THEN 12+1+1
  WHEN co.pizza_id=2 AND LENGTH(co.extras)>1 THEN 12+1+1
END AS Cost
FROM customer_orders co
JOIN runner_orders ro
ON co.order_id=ro.order_id
WHERE ro.cancellation IS NULL
ORDER BY co.pizza_id
)
SELECT runner_id,SUM(Cost) AS Sales
FROM CTE_additional_costs
GROUP BY runner_id;
```

runner_id	Sales
1	72
2	59
3	13

```sql
CREATE TABLE Ratings
(
`order_id` INTEGER,
`customer_id` INTEGER,
`pizza_id` INTEGER,
`order_time` TIMESTAMP,
`runner_id` INTEGER,
`pickup_time` VARCHAR(19),
`distance` VARCHAR(7),
`duration` VARCHAR(10),
`cancellation` VARCHAR(23),
`Ratings` FLOAT
);

INSERT INTO ratings
SELECT *
FROM (
SELECT co.order_id,co.customer_id,co.pizza_id,co.order_time,ro.runner_id,
ro.pickup_time,ro.distance,ro.duration,ro.cancellation,
CASE
  WHEN ro.duration<=10  THEN 5
  WHEN ro.duration<=ro.distance  THEN 5
  WHEN ro.duration>=10 AND ro.distance<=15  THEN 4
    WHEN ro.duration=20 AND ro.distance<=15   THEN 4
    WHEN ro.duration>20 AND ro.distance<=30  THEN 3.5
  WHEN ro.duration>=30 AND ro.distance<=25   THEN 3
END AS Ratings
FROM customer_orders co
LEFT JOIN runner_orders ro
ON co.order_id=ro.order_id
WHERE ro.cancellation IS NULL
) AS subquery;

SELECT * FROM ratings;
```

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.

order_id	customer_id	pizza_id	order_time	runner_id	pickup_time	distance	duration	cancellation	Ratings
1	101	1	2020-01-01 18:05:02	1	2020-01-01 18:15:34	20	32	NULL	3.5
2	101	1	2020-01-01 19:00:52	1	2020-01-01 19:10:54	20	27	NULL	3.5
3	102	1	2020-01-02 23:51:23	1	2020-01-03 00:12:37	13	20	NULL	4
3	102	2	2020-01-02 23:51:23	1	2020-01-03 00:12:37	13	20	NULL	4
4	103	1	2020-01-04 13:23:46	2	2020-01-04 13:53:03	23	40	NULL	3.5
4	103	1	2020-01-04 13:23:46	2	2020-01-04 13:53:03	23	40	NULL	3.5
4	103	2	2020-01-04 13:23:46	2	2020-01-04 13:53:03	23	40	NULL	3.5
5	104	1	2020-01-08 21:00:29	3	2020-01-08 21:10:57	10	15	NULL	4
7	105	2	2020-01-08 21:20:29	2	2020-01-08 21:30:45	25	25	NULL	5
8	102	1	2020-01-09 23:54:33	2	2020-01-10 00:15:02	23	15	NULL	5
10	104	1	2020-01-11 18:34:49	1	2020-01-11 18:50:20	10	10	NULL	5
10	104	1	2020-01-11 18:34:49	1	2020-01-11 18:50:20	10	10	NULL	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? 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 customer_id,order_id,runner_id,ratings,order_time,pickup_time,duration,distance,
ROUND(TIMESTAMPDIFF(MINUTE,order_time,pickup_time)) AS Time_diferrence,
ROUND(AVG(distance/duration),2) AS Average_speed,
COUNT(pizza_id) AS Total_pizzas
FROM ratings
GROUP BY customer_id,order_id,runner_id,ratings,order_time,pickup_time,duration,distance;
```

customer_id	order_id	runner_id	ratings	order_time	pickup_time	duration	distance	Time_diferrence	Average_speed	Total_pizzas
101	1	1	3.5	2020-01-01 18:05:02	2020-01-01 18:15:34	32	20	10	0.62	1
101	2	1	3.5	2020-01-01 19:00:52	2020-01-01 19:10:54	27	20	10	0.74	1
102	3	1	4	2020-01-02 23:51:23	2020-01-03 00:12:37	20	13	21	0.65	2
103	4	2	3.5	2020-01-04 13:23:46	2020-01-04 13:53:03	40	23	29	0.57	3
104	5	3	4	2020-01-08 21:00:29	2020-01-08 21:10:57	15	10	10	0.67	1
105	7	2	5	2020-01-08 21:20:29	2020-01-08 21:30:45	25	25	10	1	1
102	8	2	5	2020-01-09 23:54:33	2020-01-10 00:15:02	15	23	20	1.53	1
104	10	1	5	2020-01-11 18:34:49	2020-01-11 18:50:20	10	10	15	1	2

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

```sql
WITH CTE_Gross_profit AS (
SELECT co.order_id,co.pizza_id,ro.runner_id,
CASE
  WHEN co.pizza_id=1 THEN 12
  ELSE 10
END AS Cost,
(ro.distance * 0.30) AS Delivery_fees
FROM customer_orders co
JOIN runner_orders ro
ON co.order_id=ro.order_id
WHERE ro.cancellation IS NULL
)
SELECT runner_id,SUM(Cost) AS Cost,SUM(Delivery_fees) AS Delivery_fees,
(SUM(Cost)-SUM(Delivery_fees)) AS Gross_profit
FROM CTE_Gross_profit
GROUP BY runner_id;
```

runner_id	Cost	Delivery_fees	Gross_profit
1	70	25.80	44.20
2	56	35.10	20.90
3	12	3.00	9.00

INSIGHTS : Ingredient Optimisation

- ✓ If a Meat Lovers pizza costs $12 and a Vegetarian costs $10 and there were no charges for changes runner-1 makes $70,runner-2 makes $56, and runner-3 makes $12.
- ✓ If there was an additional $1 charge for any pizza extras runner-1 makes $72,runner-2 makes $59, and runner-3 makes $13.
- ✓ On average each runner had spent 41.36$ for the delivery charge. The Gross Profit made by each runners is runner-1 earned $ 44.20,runner-2 made $20.90, and runner-3 made $9.00.


Thank you