8 Week SQL Challenge

Case Study #2 - Pizza Runner

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 Statement

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.

Danny has shared with you 6 key datasets for this case study:

1. runners
2. customers_orders
3. runner_orders
4. pizza_names
5. pizza_toppings
6. pizza_recipes

Entity Relationship Diagram

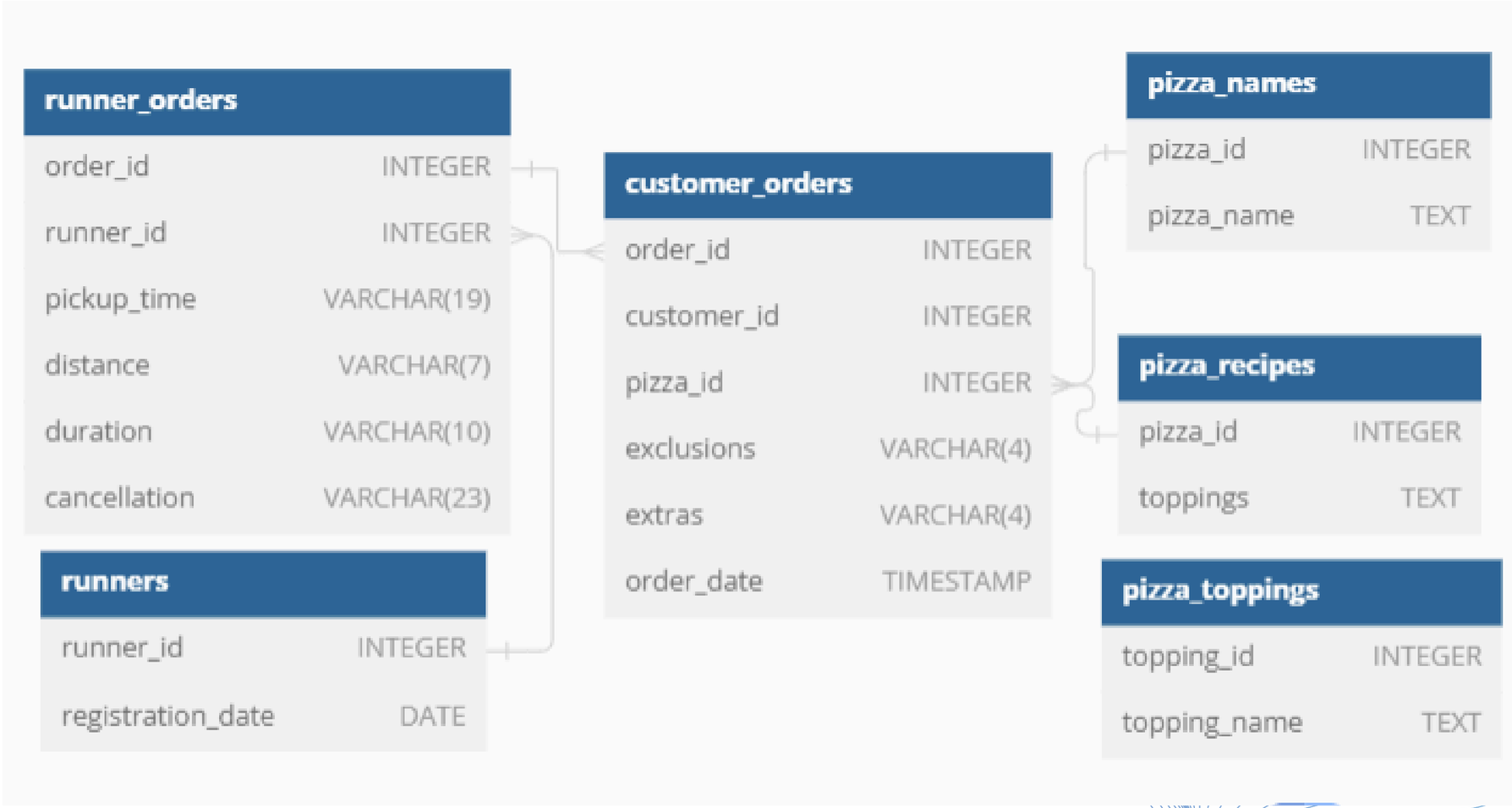

Table 1: runners

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.

Table - 2 customers_orders

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	1	2020-01-01 18:15:34	20km	32 minutes	
2	2	1	2020-01-01 19:10:54	20km	27 minutes	
3	3	1	2020-01-03 00:12:37	13.4km	20 mins	NULL
4	4	2	2020-01-04 13:53:03	23.4	40	NULL
5	5	3	2020-01-08 21:10:57	10	15	NULL
6	6	3	null	null	null	Restaurant Cancellation
7	7	2	2020-01-08 21:30:45	25km	25mins	null
8	8	2	2020-01-10 00:15:02	23.4 km	15 minute	null
9	9	2	null	null	null	Customer Cancellation
10	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 Optimisation
- Pricing and Ratings
- Bonus DML Challenges (DML = Data Manipulation Language)

Data Cleaning & Transformation

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 values and data types in the and tables!

Table: customer_orders

We can see that there are In the exclusions column, there are missing/ blank spaces ' ' and null values. In the extras column, there are missing/ blank spaces ' ' and null values.

Our course of action to clean the table: Create a temporary table with all the columns Remove null values in exlusions and extras columns and replace with blank space ' ' .

```
 Select order_id, customer_id, pizza_id,
  CASE WHEN exclusions IS null OR exclusions LIKE 'null' THEN ' ' ELSE
exclusions
      END AS exclusions,
  CASE WHEN extras IS NULL or extras LIKE 'null' THEN ' ' ELSE extras
      END AS extras, order_time
  into customer_orders_temp
  from customer_orders ;
```

This is how the clean customers_orders_temp table looks like and we will use this table to run all our queries.

	order_id	customer_id	pizza_id	exclusions	extras	order_time
1	1	101	1			2020-01-01 18:05:02.000
2	2	101	1			2020-01-01 19:00:52.000
3	3	102	1			2020-01-02 23:51:23.000
4	3	102	2			2020-01-02 23:51:23.000
5	4	103	1	4		2020-01-04 13:23:46.000
6	4	103	1	4		2020-01-04 13:23:46.000
7	4	103	2	4		2020-01-04 13:23:46.000
8	5	104	1		1	2020-01-08 21:00:29.000
9	6	101	2			2020-01-08 21:03:13.000
10	7	105	2		1	2020-01-08 21:20:29.000
11	8	102	1			2020-01-09 23:54:33.000
12	9	103	1	4	1, 5	2020-01-10 11:22:59.000
13	10	104	1			2020-01-11 18:34:49.000
14	10	104	1	2, 6	1, 4	2020-01-11 18:34:49.000

Table: runner_orders

Our course of action to clean the table:
In pickup_time column, remove nulls and replace with blank space ' '.
In distance column, remove "km" and nulls and replace with blank space ' '.
In duration column, remove "minutes", "minute" and nulls and replace with blank space ' '.
In cancellation column, remove NULL and null and and replace with blank space ' '.

```
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
  into runner_orders_temp
  FROM runner_orders ;
```

Then, we alter the pickup_time, distance and duration columns to the correct data type.

```
ALTER TABLE runner_orders_temp ALTER COLUMN distance FLOAT;
ALTER TABLE runner_orders_temp ALTER COLUMN duration INT;
ALTER TABLE runner_orders_temp ALTER COLUMN pickup_time DATETIME;
```

This is how the clean runner_orders_temp table looks like and we will use this table to run all our queries.

	order_id	runner_id	pickup_time	distance	duration	cancellation
1	1	1	2020-01-01 18:15:34.000	20	32	
2	2	1	2020-01-01 19:10:54.000	20	27	
3	3	1	2020-01-03 00:12:37.000	13.4	20	
4	4	2	2020-01-04 13:53:03.000	23.4	40	
5	5	3	2020-01-08 21:10:57.000	10	15	
6	6	3	1900-01-01 00:00:00.000	0	0	Restaurant Cancellation
7	7	2	2020-01-08 21:30:45.000	25	25	
8	8	2	2020-01-10 00:15:02.000	23.4	15	
9	9	2	1900-01-01 00:00:00.000	0	0	Customer Cancellation
10	10	1	2020-01-11 18:50:20.000	10	10	

Table : pizza_recipes (Optional)

Normalized the Pizza_recipes table such that each row has pizza_id and its corresponding one topping.
Having any form of a list within a cell is not a great way to proceed from a data or analytical standpoint.

This is how the pizza_recipes_temp table looks like and we will use this table to run all our queries.

	pizza_id	toppings
1	1	1
2	1	2
3	1	3
4	1	4
5	1	5
6	1	6
7	1	8
8	1	10
9	2	4
10	2	6
11	2	7
12	2	9
13	2	11
14	2	12

A. Pizza Metrics

Q1 How many pizzas were ordered?

```
select COUNT(pizza_id) as pizza_ordered from customer_orders
```

	pizza_ordered
1	14

A. Pizza Metrics

Q2 How many unique customer orders were made?

```sql
select count(distinct(order_id)) as unique_order_count from customer_orders
```

	unique_order_count
1	10

A. Pizza Metrics

Q3 How many successful orders were delivered by each runner?

```
select runner_id ,count(order_id) as [successful orders] from runner_orders_temp
where distance != 0
group by runner_id
```

	runner_id	successful orders
1	1	4
2	2	3
3	3	1

A. Pizza Metrics

Q4 How many of each type of pizza was delivered?

```
select c.pizza_id , COUNT(c.order_id) as successful_deliver from customer_orders c
inner join
runner_orders_temp r on r.order_id = c.order_id
where r.distance  != 0
group by c.pizza_id
```

	pizza_id	successful_deliver
1	1	9
2	2	3

A. Pizza Metrics

Q5 How many Vegetarian and Meatlovers were ordered by each customer?

```
Select CAST(customer_id AS NVARCHAR(100)) as customer_id , CAST(pizza_name AS NVARCHAR(100)) as pizza_name,
COUNT(cast (pizza_name as nvarchar(100))) as order_count  from customer_orders_temp c
inner join
pizza_names p on p.pizza_id = c.pizza_id
group by CAST(customer_id AS NVARCHAR(100)), CAST(pizza_name AS NVARCHAR(100))
order by customer_id
```

	customer_id	pizza_name	order_count
1	101	Meatlovers	2
2	101	Vegetarian	1
3	102	Meatlovers	2
4	102	Vegetarian	1
5	103	Meatlovers	3
6	103	Vegetarian	1
7	104	Meatlovers	3
8	105	Vegetarian	1

A. Pizza Metrics

Q6 What was the maximum number of pizzas delivered in a single order?

```
select Top 1 c.order_id , Count( pizza_id) as Max_pizza_deliver
from customer_orders_temp c
inner join
runner_orders_temp r on r.order_id = c.order_id
where distance != 0
group by c.order_id
order by Max_pizza_deliver desc
```

	order_id	Max_pizza_deliver
1	4	3

A. Pizza Metrics

Q7 For each customer, how many delivered pizzas had at least 1 change and how many had no changes?

```
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_temp c
inner join
runner_orders_temp r on r.order_id = c.order_id
where r.distance != 0
group by c.customer_id
ORDER BY c.customer_id;
```

	customer_id	at_least_1_change	no_change
1	101	0	2
2	102	0	3
3	103	3	0
4	104	2	1
5	105	1	0

A. Pizza Metrics

Q8 How many pizzas were delivered that had both exclusions and extras?

```
select COUNT(pizza_id) as pizza_count_w_exclusions_extras
from customer_orders_temp c
inner join
runner_orders_temp r on r.order_id = c.order_id
where distance != 0 and exclusions != ' ' and extras != ' '
```

	pizza_count_w_exclusions_extras
1	1

A. Pizza Metrics

Q9 What was the total volume of pizzas ordered for each hour of the day?

```
select DATEPART(HOUR , order_time) as Hour_of_day ,
COUNT(order_id) as pizza_count from customer_orders_temp
group by DATEPART(HOUR , order_time)
```

	Hour_of_day	pizza_count
1	11	1
2	13	3
3	18	3
4	19	1
5	21	3
6	23	3

A. Pizza Metrics

Q10 What was the volume of orders for each day of the week?

```
SELECT FORMAT(DATEADD(DAY, 2, order_time),'dddd') AS day_of_week,
COUNT(order_id) AS total_pizzas_ordered
FROM customer_orders_temp
GROUP BY FORMAT(DATEADD(DAY, 2, order_time),'dddd');
```

	day_of_week	total_pizzas_ordered
1	Friday	5
2	Monday	5
3	Saturday	3
4	Sunday	1

B. Runner and Customer Experience

Q1 How many runners signed up for each 1 week period? (i.e. week starts 2021-01-01)

```
select DATEDIFF(DAY, '2021-01-01',registration_date) / 7 + 1 AS week_number ,
count(runner_id) as Runners_signed
from runners
group by DATEDIFF(DAY, '2021-01-01',registration_date) / 7 + 1 ;
```

	week_number	Runners_signed
1	1	2
2	2	1
3	3	1

B. Runner and Customer Experience

Q2 What was the average time in minutes it took for each runner to arrive at the Pizza Runner HQ to pickup the order?

```
select runner_id ,AVG(DATEDIFF(MINUTE, order_time, pickup_time))
AS average_arrival_time_minutes
from customer_orders_temp c
inner join
runner_orders_temp r on r.order_id = c.order_id
where distance != 0
group by runner_id ;
```

	runner_id	average_arrival_time_minutes
1	1	15
2	2	24
3	3	10

B. Runner and Customer Experience

Q3 Is there any relationship between the number of pizzas and how long the order takes to prepare?

```
with cte as (select c.order_id,COUNT(pizza_id) as num_piza,DATEDIFF(MINUTE ,
order_time,pickup_time) as duration
from customer_orders_temp c
inner join
runner_orders_temp r on r.order_id = c.order_id
where distance != 0
group by c.order_id,DATEDIFF(MINUTE , order_time,pickup_time) )
select num_piza,AVG(duration) as avg_prepare_time from cte
group by num_piza ;
```

	num_piza	avg_prepare_time
1	1	12
2	2	18
3	3	30

B. Runner and Customer Experience

Q4 What was the average distance travelled for each customer?

```
select customer_id , AVG(distance) as avg_distance from customer_orders_temp c
inner join
runner_orders_temp r on r.order_id =c.order_id
where distance != 0
group by customer_id ;
```

	customer_id	avg_distance
1	101	20
2	102	16.7333333333333
3	103	23.4
4	104	10
5	105	25

B. Runner and Customer Experience

Q5 What was the difference between the longest and shortest delivery times for all orders?

```
with cte as (select order_id ,duration from runner_orders_temp
where duration != 0)
select (Max(duration) - Min(duration)) as diff_delivery from cte ;
```

	diff_delivery
1	30

B. Runner and Customer Experience

Q6 What was the average speed for each runner for each delivery and do you notice any trend for these values?

```
select runner_id ,c.order_id, Round(distance/duration * 60 ,1)
as Speed_kmh  from customer_orders_temp c
join
runner_orders_temp r on r.order_id=c.order_id
where distance != 0
group by runner_id ,c.order_id ,
Round(distance/duration * 60 ,1);
```

	runner_id	order_id	Speed_kmh
1	1	1	37.5
2	1	2	44.4
3	1	3	40.2
4	1	10	60
5	2	4	35.1
6	2	7	60
7	2	8	93.6
8	3	5	40

B. Runner and Customer Experience

Q7 What is the successful delivery percentage for each runner?

```
with cte as (select runner_id,COUNT(runner_id) as Total_recieve_del from runner_orders_temp
group by runner_id)
,cte1 as (select runner_id,COUNT(runner_id) as successful_del from runner_orders_temp
where distance != 0
group by runner_id)
select c.runner_id , Total_recieve_del ,successful_del ,
Cast((1.0* successful_del/Total_recieve_del)*100 as decimal (5,1))
as successful_pct from cte c
inner join
cte1 c1 on c.runner_id = c1.runner_id ;
```

	runner_id	Total_recieve_del	successful_del	successful_pct
1	1	4	4	100.0
2	2	4	3	75.0
3	3	2	1	50.0

C. Ingredient Optimisation

Q1 What are the standard ingredients for each pizza?

```
with cte as (select p.pizza_id ,Cast (topping_name as nvarchar(20)) as  topping_name
from pizza_recipes_temp p
inner join
pizza_toppings t on t.topping_id =  p.toppings)
select pizza_id , STRING_AGG(topping_name  , ' , ') as topping_name  from cte
group by pizza_id ;
```

	pizza_id	topping_name
1	1	Bacon , BBQ Sauce , Beef , Cheese , Chicken , Mus...
2	2	Cheese , Mushrooms , Onions , Peppers , Tomatoe...

C. Ingredient Optimisation

Q2 What was the most commonly added extra?

```
with cte as (select top 1 extras , count (extras)
as extras_counted from customer_orders_temp
where extras != ' '
group by extras
order by extras_counted desc)
select extras ,topping_name  from cte c
inner join
pizza_toppings p on p.topping_id = c.extras ;
```

	extras	topping_name
1	1	Bacon

C. Ingredient Optimisation

Q3 What was the most common exclusion?

```
with cte as (select Top 1  exclusions ,
count(exclusions) as Most_common from customer_orders_temp
where exclusions != ' '
group by exclusions
order by Most_common desc)
select exclusions,topping_name from cte c
join
pizza_toppings  p on p.topping_id =c.exclusions ;
```

	exclusions	topping_name
1	4	Cheese

D. Pricing and Ratings

Q1 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 pizza_id=1 then 12
when pizza_id=2 then 10 end) as Total_profit
from customer_orders_temp c
inner join
runner_orders_temp r on r.order_id=c.order_id
where distance != 0
```

	Total_profit
1	138

D. Pricing and Ratings

Q3 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 4 your own data for ratings for each successful customer order between 1 to 5.

```
create table ratings (
order_id integer,
rating integer);
insert into ratings
(order_id, rating)
values
(1,3),
(2,5),
(3,3),
(4,1),
(5,5),
(7,3),
 (8,4),
(10,3);
```

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

D. Pricing and Ratings

Q4 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

Query

```
select c.customer_id, c.order_id, r.runner_id, ratings.rating, c.order_time,
r.pickup_time, datediff(minute, order_time, pickup_time) as Time_bw_Order_Pickup,
r.duration, round(avg(r.distance*60/r.duration),1) as avgspeed, count(c.pizza_id) as PIzzaCount
from customer_orders_temp c
inner join runner_orders_temp r
on c.order_id = r.order_id
inner join ratings
on ratings.order_id = c.order_id
group by c.customer_id, c.order_id, r.runner_id, ratings.rating, c.order_time,
r.pickup_time, datediff(minute, order_time, pickup_time), r.duration
order by customer_id;
```

Solution :

Table

	customer_id	order_id	runner_id	rating	order_time	pickup_time	Time_bw_Order_Pickup	duration	avgspeed	PIzzaCount
1	101	1	1	3	2020-01-01 18:05:02.000	2020-01-01 18:15:34.000	10	32	37.5	1
2	101	2	1	5	2020-01-01 19:00:52.000	2020-01-01 19:10:54.000	10	27	44.4	1
3	102	3	1	3	2020-01-02 23:51:23.000	2020-01-03 00:12:37.000	21	20	40.2	2
4	102	8	2	4	2020-01-09 23:54:33.000	2020-01-10 00:15:02.000	21	15	93.6	1
5	103	4	2	1	2020-01-04 13:23:46.000	2020-01-04 13:53:03.000	30	40	35.1	3
6	104	5	3	5	2020-01-08 21:00:29.000	2020-01-08 21:10:57.000	10	15	40	1
7	104	10	1	3	2020-01-11 18:34:49.000	2020-01-11 18:50:20.000	16	10	60	2
8	105	7	2	3	2020-01-08 21:20:29.000	2020-01-08 21:30:45.000	10	25	60	1

D. Pricing and Ratings

Q5 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?

```
with cte as (select SUM(case when pizza_id=1 then 12
when pizza_id=2 then 10 end) as Total_profit
from customer_orders_temp c
inner join
runner_orders_temp r on r.order_id=c.order_id
where distance != 0 )
,cte1 as (select
SUM(distance * 0.3) as delivery_fee
from runner_orders_temp
where distance != 0 )

select Total_profit -(select * from cte1) as  Final_amount from cte
```

	Final_amount
1	94.44

Bonus Questions

If Danny wants to expand his range of pizzas - how would this impact the existing data design?

Because the pizza recipes table was modified to reflect foreign key designation for each topping linked
to the base pizza, the pizza_id will have multiple 3s and align with the standard toppings (individually) within the toppings column.
In addition, because the data type was casted to an int to take advantage of numerical functions,insertion of data would not
affect the existing data design, unlike the original dangerous approach of comma separated values in a singular row (list).