Made by: *Mai Huy Khang*

Note: *all results use for practicing SQL (MySQL) only*

INTRODUCTION

❖ Context

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 wants to use the data to answer a few simple questions about his business, especially about their order, revenue, best seller product, etc. Having this meaningful result will help him to plan and develop his business in the future.

CHAPTER 1: CREATE DATASET AS REQUIRED

❖ **Coding part**

```sql
CREATE SCHEMA pizza_runner;
SET search_path = pizza_runner;

DROP TABLE IF EXISTS runners;
CREATE TABLE runners (
  runner_id INTEGER,
  registration_date DATE
);
INSERT INTO runners
  (runner_id, registration_date)
VALUES
  (1, '2021-01-01'),
  (2, '2021-01-03'),
  (3, '2021-01-08'),
  (4, '2021-01-15');


DROP TABLE IF EXISTS customer_orders;
CREATE TABLE customer_orders (
  order_id INTEGER,
  customer_id INTEGER,
  pizza_id INTEGER,
  exclusions VARCHAR(4),
  extras VARCHAR(4),
  order_time TIMESTAMP
);

INSERT INTO customer_orders
  (order_id, customer_id, pizza_id, exclusions, extras, order_time)
VALUES
  ('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', '', NULL, '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', 'null', '1', '2020-01-08 21:00:29'),
  ('6', '101', '2', 'null', 'null', '2020-01-08 21:03:13'),
  ('7', '105', '2', 'null', '1', '2020-01-08 21:20:29'),
  ('8', '102', '1', 'null', 'null', '2020-01-09 23:54:33'),
  ('9', '103', '1', '4', '1, 5', '2020-01-10 11:22:59'),
  ('10', '104', '1', 'null', 'null', '2020-01-11 18:34:49'),
  ('10', '104', '1', '2, 6', '1, 4', '2020-01-11 18:34:49');


DROP TABLE IF EXISTS runner_orders;
CREATE TABLE runner_orders (
  order_id INTEGER,
  runner_id INTEGER,
  pickup_time VARCHAR(19),
  distance VARCHAR(7),
  duration VARCHAR(10),
  cancellation VARCHAR(23)
);

INSERT INTO runner_orders
  (order_id, runner_id, pickup_time, distance, duration, cancellation)
VALUES
  ('1', '1', '2020-01-01 18:15:34', '20km', '32 minutes', ''),
  ('2', '1', '2020-01-01 19:10:54', '20km', '27 minutes', ''),
  ('3', '1', '2020-01-03 00:12:37', '13.4km', '20 mins', NULL),
  ('4', '2', '2020-01-04 13:53:03', '23.4', '40', NULL),
  ('5', '3', '2020-01-08 21:10:57', '10', '15', NULL),
  ('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');


DROP TABLE IF EXISTS pizza_names;
CREATE TABLE pizza_names (
  pizza_id INTEGER,
  pizza_name TEXT
);
INSERT INTO pizza_names
  (pizza_id, pizza_name)
VALUES
  (1, 'Meatlovers'),
  (2, 'Vegetarian');


DROP TABLE IF EXISTS pizza_recipes;
CREATE TABLE pizza_recipes (
  pizza_id INTEGER,
  toppings TEXT
);
INSERT INTO pizza_recipes
  (pizza_id, toppings)
VALUES
  (1, '1, 2, 3, 4, 5, 6, 8, 10'),
  (2, '4, 6, 7, 9, 11, 12');


DROP TABLE IF EXISTS pizza_toppings;
CREATE TABLE pizza_toppings (
  topping_id INTEGER,
  topping_name TEXT
);
INSERT INTO pizza_toppings
  (topping_id, topping_name)
VALUES
  (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');
```

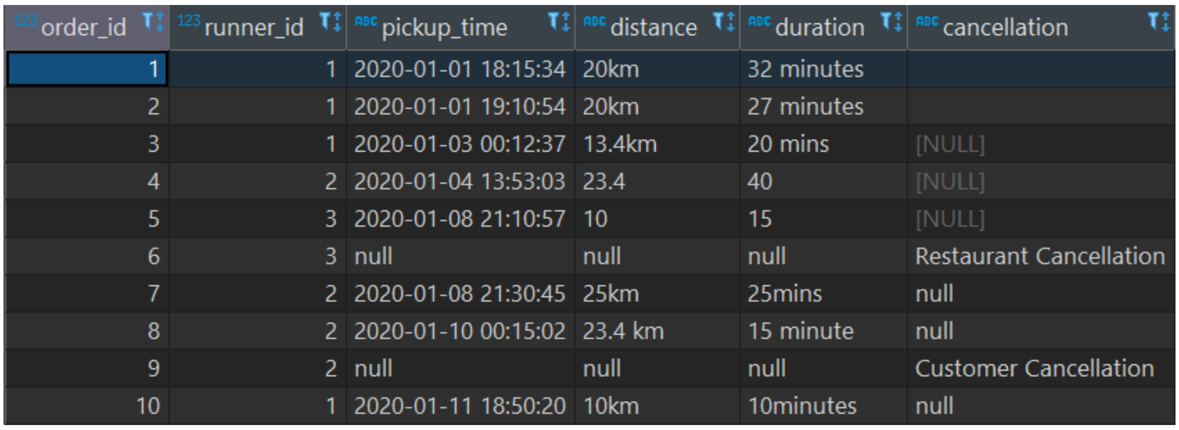
❖ PREPROCESSING DATA

Runner_order table

```
CREATE TABLE runner_orders_temp AS
(
SELECT r.order_id,
           r.runner_id,
           CASE WHEN r.pickup_time = 'null' THEN NULL
                ELSE r.pickup_time
           END as pickup_time,
           CASE WHEN r.distance = 'null' THEN NULL
                 ELSE ROUND(CAST(REGEXP_SUBSTR(r.distance,'[0-
9]+[/./]?[0-9]+') AS FLOAT),1)
           END as distance,
           CASE WHEN r.duration = 'null' THEN NULL
                 ELSE CAST(REGEXP_SUBSTR(r.duration,'[0-9]*') AS
UNSIGNED)
```

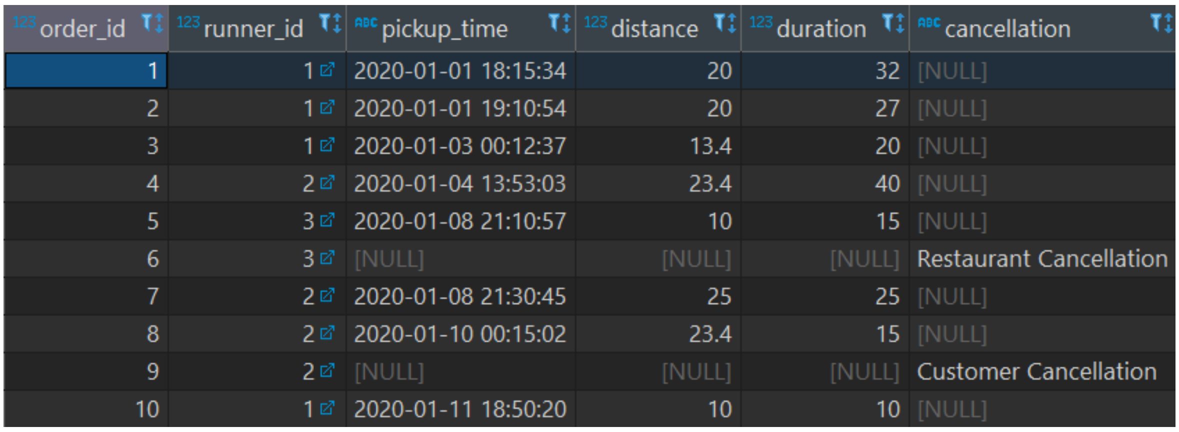
```
           END as duration,
           CASE WHEN r.cancellation IN ('','null') THEN NULL
                 ELSE r.cancellation
           END as cancellation
FROM runner_orders r
)

RENAME TABLE runner_orders to runner_orders_old, runner_orders_temp to
runner_orders

DROP TABLE runner_orders_old
```

Before



order_id	runner_id	pickup_time	distance	duration	cancellation
1	1	2020-01-01 18:15:34	20km	32 minutes	
2	1	2020-01-01 19:10:54	20km	27 minutes	
3	1	2020-01-03 00:12:37	13.4km	20 mins	[NULL]
4	2	2020-01-04 13:53:03	23.4	40	[NULL]
5	3	2020-01-08 21:10:57	10	15	[NULL]
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

Result

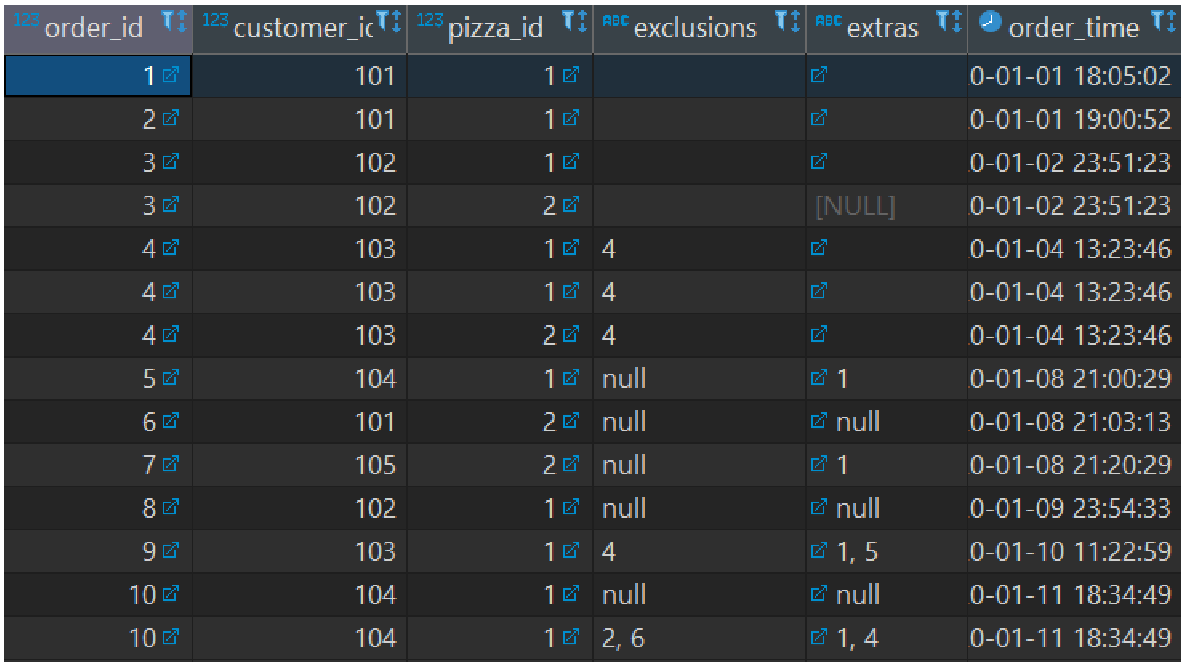


order_id	runner_id	pickup_time	distance	duration	cancellation
1	1	2020-01-01 18:15:34	20	32	[NULL]
2	1	2020-01-01 19:10:54	20	27	[NULL]
3	1	2020-01-03 00:12:37	13.4	20	[NULL]
4	2	2020-01-04 13:53:03	23.4	40	[NULL]
5	3	2020-01-08 21:10:57	10	15	[NULL]
6	3	[NULL]	[NULL]	[NULL]	Restaurant Cancellation
7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

Customer_orders table:

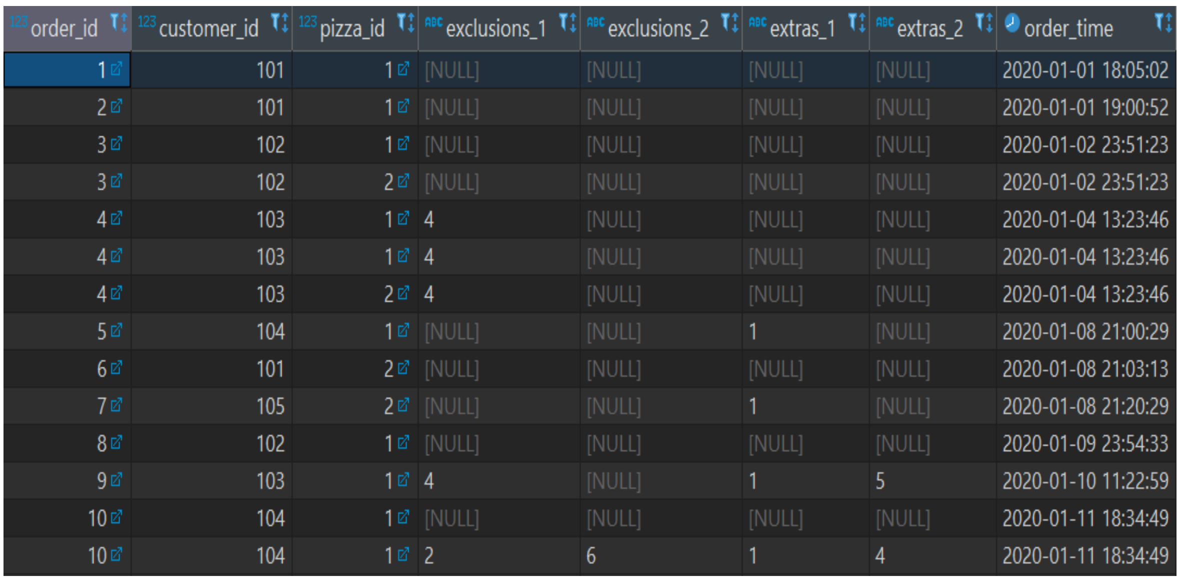
```
CREATE TABLE customer_orders_temp as
(
SELECT     c.order_id ,
           c.customer_id ,
           c.pizza_id ,
           CASE WHEN c.exclusions in ('','null') THEN NULL
                 WHEN substr(c.exclusions,2,1) = ',' THEN
substr(c.exclusions,1,1)
                 ELSE c.exclusions
           END as exclusions_1,
           CASE WHEN c.exclusions in ('','null') or
length(c.exclusions) = 1 THEN NULL
                 WHEN substr(c.exclusions,2,1) = ',' THEN
substr(c.exclusions,4,1)
                 ELSE c.exclusions
           END as exclusions_2,
           CASE WHEN c.extras in('','null') THEN NULL
                 WHEN substr(c.extras ,2,1) = ',' THEN
substr(c.extras ,1,1)
                 ELSE c.extras
           END as extras_1,
           CASE WHEN c.extras in('','null') or length(c.extras) =
1 THEN NULL
                 WHEN substr(c.extras ,2,1) = ',' THEN
substr(c.extras ,4,1)
                 ELSE c.extras
           END as extras_2,
           c.order_time
FROM customer_orders_old c
)
```

Before



order_id	customer_ic	pizza_id	exclusions	extras	order_time
1	101	1			0-01-01 18:05:02
2	101	1			0-01-01 19:00:52
3	102	1			0-01-02 23:51:23
3	102	2		[NULL]	0-01-02 23:51:23
4	103	1	4		0-01-04 13:23:46
4	103	1	4		0-01-04 13:23:46
4	103	2	4		0-01-04 13:23:46
5	104	1	null	1	0-01-08 21:00:29
6	101	2	null	null	0-01-08 21:03:13
7	105	2	null	1	0-01-08 21:20:29
8	102	1	null	null	0-01-09 23:54:33
9	103	1	4	1, 5	0-01-10 11:22:59
10	104	1	null	null	0-01-11 18:34:49
10	104	1	2, 6	1, 4	0-01-11 18:34:49

After

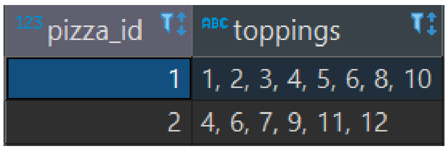


order_id	customer_id	pizza_id	exclusions_1	exclusions_2	extras_1	extras_2	order_time
1	101	1	[NULL]	[NULL]	[NULL]	[NULL]	2020-01-01 18:05:02
2	101	1	[NULL]	[NULL]	[NULL]	[NULL]	2020-01-01 19:00:52
3	102	1	[NULL]	[NULL]	[NULL]	[NULL]	2020-01-02 23:51:23
3	102	2	[NULL]	[NULL]	[NULL]	[NULL]	2020-01-02 23:51:23
4	103	1	4	[NULL]	[NULL]	[NULL]	2020-01-04 13:23:46
4	103	1	4	[NULL]	[NULL]	[NULL]	2020-01-04 13:23:46
4	103	2	4	[NULL]	[NULL]	[NULL]	2020-01-04 13:23:46
5	104	1	[NULL]	[NULL]	1	[NULL]	2020-01-08 21:00:29
6	101	2	[NULL]	[NULL]	[NULL]	[NULL]	2020-01-08 21:03:13
7	105	2	[NULL]	[NULL]	1	[NULL]	2020-01-08 21:20:29
8	102	1	[NULL]	[NULL]	[NULL]	[NULL]	2020-01-09 23:54:33
9	103	1	4	[NULL]	1	5	2020-01-10 11:22:59
10	104	1	[NULL]	[NULL]	[NULL]	[NULL]	2020-01-11 18:34:49
10	104	1	2	6	1	4	2020-01-11 18:34:49

Pizza_recipes table

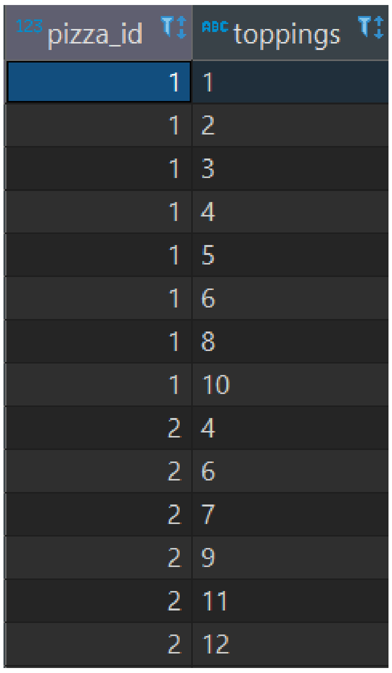
```
CREATE TABLE pizza_recipes_temp as
(
SELECT t.pizza_id,
       trim(j.toppings) AS toppings
FROM pizza_recipes as t
INNER JOIN json_table(trim(replace(json_array(t.toppings), ',', '","')),
'$[*]' columns (toppings varchar(50) PATH '$')) j
)

RENAME TABLE pizza_recipes to pizza_recipes_old, pizza_recipes_temp to
pizza_recipes

DROP TABLE pizza_recipes_old
```

Before



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

After



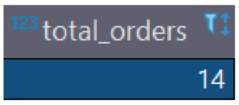
pizza_id	toppings
1	1
1	2
1	3
1	4
1	5
1	6
1	8
1	10
2	4
2	6
2	7
2	9
2	11
2	12

CHAPTER 2: SOLVE CASE STUDY QUESTION

I. PIZZA METRICS

1) How many pizzas were ordered?

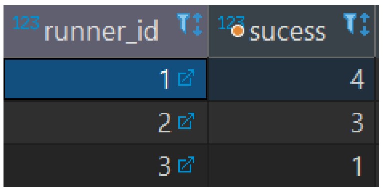
```
SELECT count(pizza_id) as total_orders
FROM customer_orders co
```



total_orders
14

2) How many successful orders were delivered by each runner?

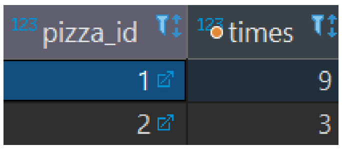
```
SELECT runner_id,
       count(pickup_time) as sucess
FROM runner_orders ro
WHERE pickup_time is not null
GROUP BY runner_id
```



runner_id	success
1	4
2	3
3	1

3) How many of each type of pizza was delivered?

```
SELECT co.pizza_id,
     count(ro.pickup_time) as times
FROM customer_orders as co
     LEFT JOIN runner_orders as ro
     USING(order_id)
WHERE ro.pickup_time is not null
GROUP BY co.pizza_id
```

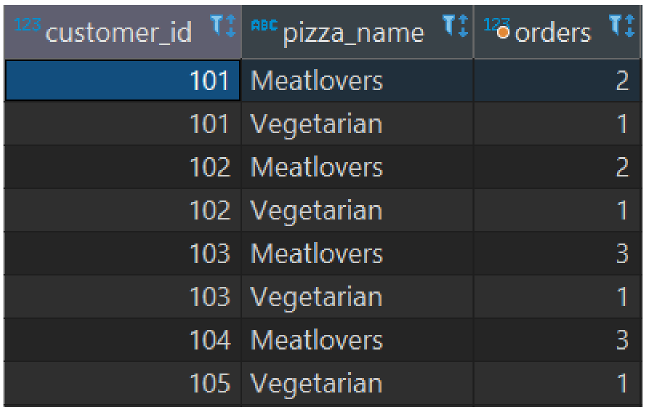


pizza_id	times
1	9
2	3

4) How many Vegetarian and Meatlovers were ordered by each customer?

```
SELECT co.customer_id,
          pn.pizza_name,
          count(co.pizza_id) as orders
FROM customer_orders co
     LEFT JOIN pizza_names pn
```

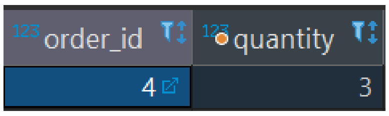
```
      USING(pizza_id)
GROUP BY 1,2
ORDER BY 1
```



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

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

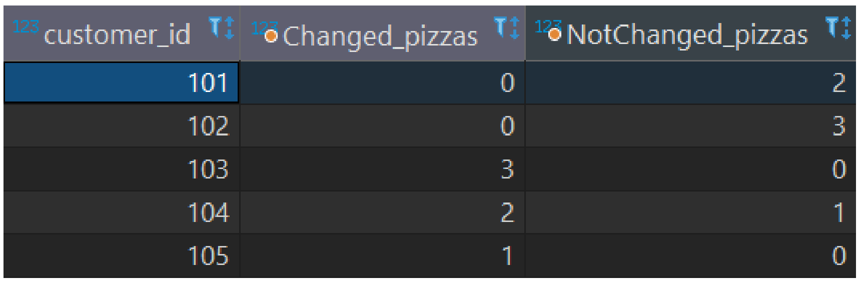
```
SELECT co.order_id,
           count(co.order_id) as quantity
FROM customer_orders as co
      LEFT JOIN runner_orders as ro
      USING(order_id)
WHERE ro.pickup_time is not null
GROUP BY co.order_id
ORDER BY quantity DESC limit 1
```



order_id	quantity
4	3

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

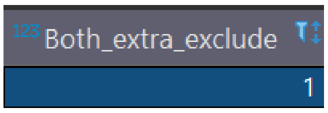
```
WITH temp as
(
SELECT co.customer_id,
           CASE WHEN co.exclusions_1 >= 1 or co.extras_1 >=1 THEN
1
                  ELSE 0
           END as Changed_pizzas,
           CASE WHEN co.exclusions_1 is null and co.extras_1 is
null THEN 1
                  ELSE 0
           END as NotChanged_pizzas,
           co.order_id
FROM customer_orders co
)
SELECT t.customer_id,
           sum(t.Changed_pizzas) as Changed_pizzas,
           sum(t.NotChanged_pizzas) as NotChanged_pizzas
FROM temp as t
LEFT JOIN runner_orders ro
USING(order_id)
WHERE ro.pickup_time is not null
GROUP BY 1
```



customer_id	Changed_pizzas	NotChanged_pizzas
101	0	2
102	0	3
103	3	0
104	2	1
105	1	0

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

```
WITH temp as
(
SELECT co.customer_id,
           co.order_id,
           CASE WHEN co.exclusions_1 >= 1 and co.extras_1 >=1 THEN
1
                  ELSE 0
           END as boths
FROM customer_orders co
)
SELECT sum(t.boths) as Both_extra_exclude
FROM temp as t
LEFT JOIN runner_orders ro
USING(order_id)
WHERE ro.pickup_time is not null
```

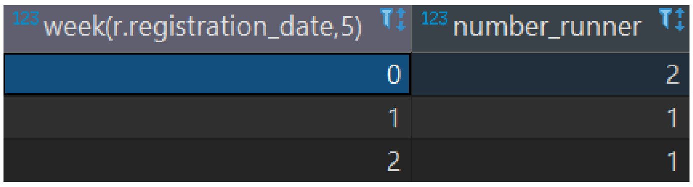


Both_extra_exclude
1

II. Runner and Customer Experience

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

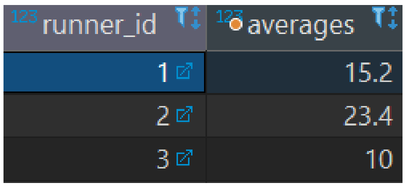
```
SELECT week(r.registration_date,5),
       count(r.runner_id) as number_runner
FROM runners as r
GROUP BY week(r.registration_date,5)
```



week(r.registration_date,5)	number_runner
0	2
1	1
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?

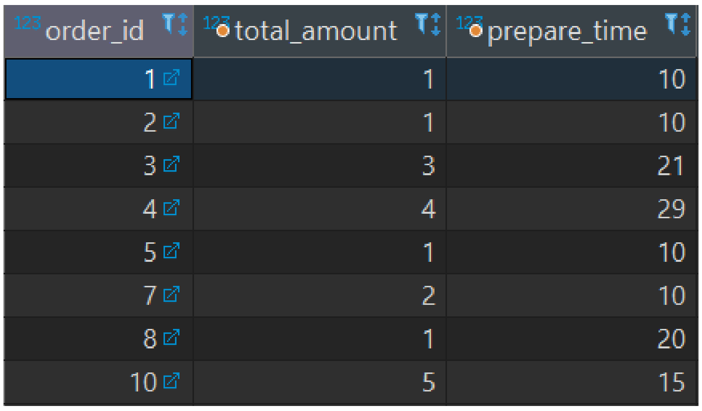
```
SELECT ro.runner_id ,

     ROUND(AVG(timestampdiff(minute,co.order_time,ro.pickup_time)),1) as
averages
FROM customer_orders co
LEFT JOIN runner_orders ro
USING(order_id)
GROUP BY 1
```



runner_id	averages
1	15.2
2	23.4
3	10

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

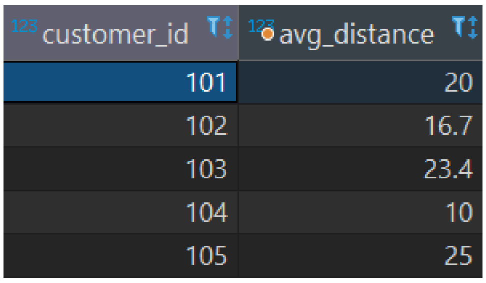
```
SELECT co.order_id ,
      SUM(co.pizza_id) as total_amount,
     ROUND(AVG(timestampdiff(minute,co.order_time,ro.pickup_time)),1) as
     prepare_time
FROM customer_orders co
     LEFT JOIN runner_orders ro
     USING(order_id)
WHERE ro.pickup_time is not null
GROUP BY 1
```



order_id	total_amount	prepare_time
1	1	10
2	1	10
3	3	21
4	4	29
5	1	10
7	2	10
8	1	20
10	5	15

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

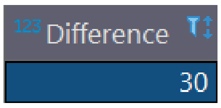
```
SELECT co.customer_id,
        ROUND(AVG(ro.distance),1) as avg_distance
FROM customer_orders as co
      LEFT JOIN runner_orders ro
      USING(order_id)
WHERE ro.pickup_time is not null
GROUP BY co.customer_id
```



customer_id	avg_distance
101	20
102	16.7
103	23.4
104	10
105	25

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

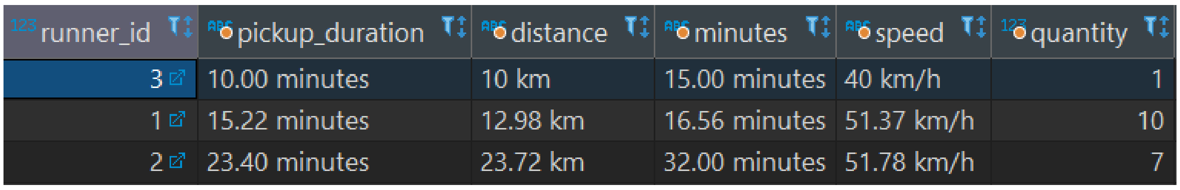
```
SELECT (max(ro.duration)-min(ro.duration)) Difference
FROM runner_orders ro
```



Difference
30

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

```
SELECT     ro.runner_id,
     concat(ROUND(AVG(timestampdiff(minute,co.order_time,ro.pickup
_time)),2),' minutes') as pickup_duration,
           concat(ROUND(AVG(ro.distance),2),' km') as distance ,
           concat(ROUND(AVG(ro.duration),2),' minutes') as minutes
,
           concat(ROUND(AVG(ro.distance/(ro.duration/60)),2),'
km/h') as speed,
           sum(co.pizza_id) as quantity
FROM runner_orders ro
     LEFT JOIN customer_orders co
     USING(order_id)
WHERE ro.pickup_time is not null
GROUP BY 1
ORDER BY 5 ASC
```

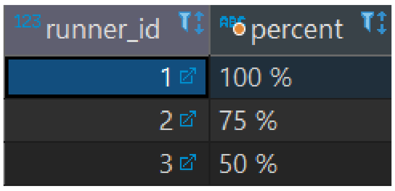


runner_id	pickup_duration	distance	minutes	speed	quantity
3	10.00 minutes	10 km	15.00 minutes	40 km/h	1
1	15.22 minutes	12.98 km	16.56 minutes	51.37 km/h	10
2	23.40 minutes	23.72 km	32.00 minutes	51.78 km/h	7

One of the insights I can tell is that the more average speed the runners have the more duration they need to pickup the orders from the store, that’s strange.

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

```
WITH temp as
(
SELECT ro.runner_id ,
           CASE WHEN ro.pickup_time is not null THEN 1
                 ELSE 0
           END as success,
           CASE WHEN ro.pickup_time is null THEN 1
                 ELSE 0
           END as fail
FROM runner_orders ro
)
SELECT t.runner_id,

     CONCAT(ROUND((sum(t.success)/(sum(success)+sum(t.fail)))*100)
,' %') as percent
FROM temp as t
GROUP BY 1
```

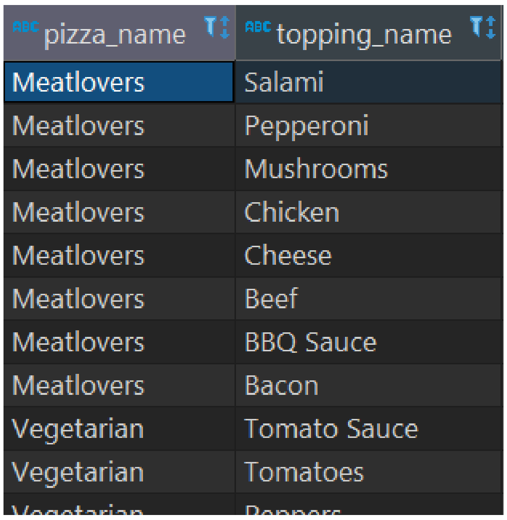


runner_id	percent
1	100 %
2	75 %
3	50 %

III. Ingredient Optimisation

1) What are the standard ingredients for each pizza?

```
ALTER TABLE pizza_recipes RENAME COLUMN toppings TO topping_id

SELECT pn.pizza_name ,
        pt.topping_name
FROM pizza_names pn
      LEFT JOIN pizza_recipes pr
      USING(pizza_id)
            LEFT JOIN pizza_toppings pt
            USING(topping_id)
```

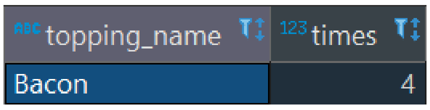


pizza_name	topping_name
Meatlovers	Salami
Meatlovers	Pepperoni
Meatlovers	Mushrooms
Meatlovers	Chicken
Meatlovers	Cheese
Meatlovers	Beef
Meatlovers	BBQ Sauce
Meatlovers	Bacon
Vegetarian	Tomato Sauce
Vegetarian	Tomatoes
Vegetarian	Peppers

2) What was the most commonly added extra?

```
WITH temp as
(
SELECT pt.topping_name
FROM customer_orders co
      LEFT JOIN pizza_toppings pt
      ON pt.topping_id = co.extras_1
WHERE co.extras_1 is not null
UNION ALL
SELECT pt.topping_name
FROM customer_orders co
      LEFT JOIN pizza_toppings pt
      ON pt.topping_id = co.extras_2
WHERE co.extras_2 is not null
)
SELECT t.topping_name,
            count(t.topping_name) as times
```

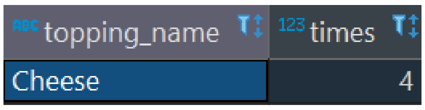
```
FROM temp as t
GROUP BY 1
ORDER BY 2 DESC limit 1
```



topping_name	times
Bacon	4

3) What was the most common exclusion?

```
WITH temp as
(
SELECT pt.topping_name
FROM customer_orders co
     LEFT JOIN pizza_toppings pt
     ON pt.topping_id = co.exclusions_1
WHERE co.exclusions_1 is not null
UNION ALL
SELECT pt.topping_name
FROM customer_orders co
     LEFT JOIN pizza_toppings pt
     ON pt.topping_id = co.exclusions_2
WHERE co.exclusions_2 is not null
)
SELECT t.topping_name,
           count(t.topping_name) as times
FROM temp as t
GROUP BY 1
ORDER BY 2 DESC limit 1
```



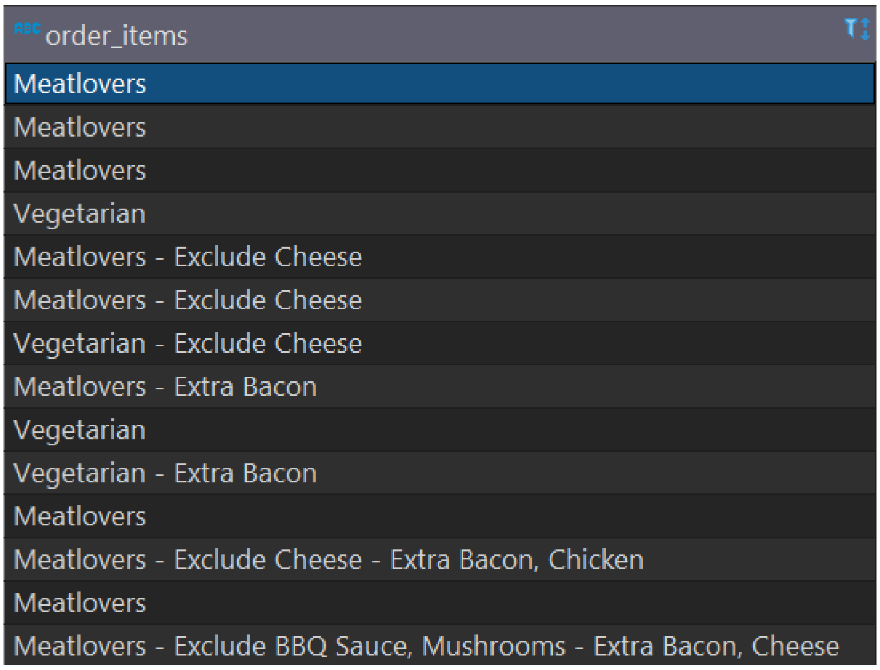
topping_name	times
Cheese	4

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`

```
WITH temp as
(
SELECT co.order_id ,
           pn.pizza_name ,
           CASE WHEN pt.topping_name is null THEN ''
```

```
                ELSE CONCAT(' - Exclude ',pt.topping_name)
           END as exclusions_1,
           CASE WHEN pt2.topping_name is null THEN ''
                WHEN pt2.topping_name is not null and
pt.topping_name is not null THEN concat(', ',pt2.topping_name)
                ELSE pt2.topping_name
           END as exclusions_2,
           CASE WHEN pt3.topping_name is null THEN ''
                ELSE CONCAT(' - Extra ',pt3.topping_name)
           END as extras_1,
           CASE WHEN pt4.topping_name is null THEN ''
                WHEN pt4.topping_name is not null and
pt3.topping_name is not null THEN concat(', ',pt4.topping_name)
                ELSE pt4.topping_name
           END as extras_2
FROM customer_orders co
      LEFT JOIN pizza_names pn
      USING(pizza_id)
           LEFT JOIN pizza_toppings pt
           ON pt.topping_id = co.exclusions_1
                LEFT JOIN pizza_toppings pt2
                ON pt2.topping_id = co.exclusions_2
                     LEFT JOIN pizza_toppings pt3
                     ON pt3.topping_id = co.extras_1
                          LEFT JOIN pizza_toppings pt4
                          ON pt4.topping_id = co.extras_2
)
SELECT
concat(t.pizza_name,t.exclusions_1,t.exclusions_2,t.extras_1,t.ext
ras_2) as order_items
FROM temp as t
```



order_items
Meatlovers
Meatlovers
Meatlovers
Vegetarian
Meatlovers - Exclude Cheese
Meatlovers - Exclude Cheese
Vegetarian - Exclude Cheese
Meatlovers - Extra Bacon
Vegetarian
Vegetarian - Extra Bacon
Meatlovers
Meatlovers - Exclude Cheese - Extra Bacon, Chicken
Meatlovers
Meatlovers - Exclude BBQ Sauce, Mushrooms - Extra Bacon, Cheese

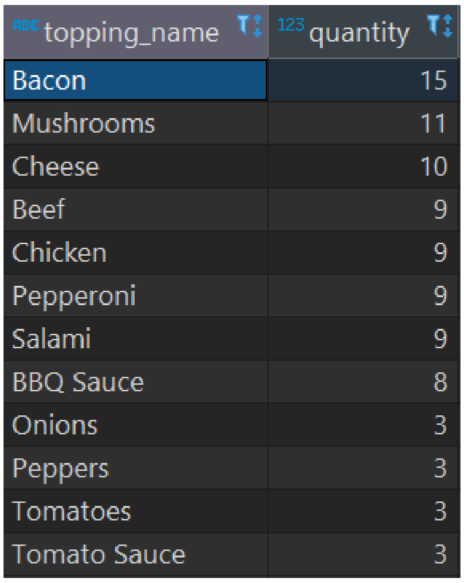
5) Generate an alphabetically ordered comma separated ingredient list for each pizza order from the `customer_orders` table and add a `2x` in front of any relevant ingredients

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

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

```
WITH toppings as
(
(SELECT pt.topping_name ,
           pt.topping_id,
           pr.pizza_id
FROM pizza_toppings pt
     LEFT JOIN pizza_recipes pr
     USING(topping_id))
UNION ALL
(SELECT pt2.topping_name ,
           pt2.topping_id ,
           co3.pizza_id
FROM customer_orders co3
LEFT JOIN pizza_toppings pt2
     ON co3.extras_1 = pt2.topping_id
WHERE co3.pizza_id = 2 and co3.extras_1 = 1)
)
, amounts as
(
```

```sql
SELECT co.pizza_id ,
           count(co.pizza_id) as amount
FROM customer_orders co
     LEFT JOIN runner_orders ro
     USING(order_id)
WHERE ro.pickup_time is not null
GROUP BY 1
)
, exclusive as
(
(SELECT co2.exclusions_1 as exclusions,
           co2.pizza_id,
           count(co2.exclusions_1) as counts
FROM customer_orders co2
LEFT JOIN runner_orders ro2
USING(order_id)
WHERE co2.exclusions_1 is not null and ro2.pickup_time is not null
GROUP BY 1,2)
UNION
(SELECT co2.exclusions_2 as exclusions,
           co2.pizza_id,
           count(co2.exclusions_2) as counts
FROM customer_orders co2
LEFT JOIN runner_orders ro2
USING(order_id)
WHERE co2.exclusions_2 is not null and ro2.pickup_time is not null
GROUP BY 1,2)
)
, extra as
(
(SELECT co2.extras_1 as extras,
           co2.pizza_id,
           count(co2.extras_1) as counts
FROM customer_orders co2
LEFT JOIN runner_orders ro2
USING(order_id)
WHERE co2.extras_1 is not null and ro2.pickup_time is not null
GROUP BY 1,2)
UNION
(SELECT co2.extras_2 as extras,
           co2.pizza_id,
           count(co2.extras_2) as counts
FROM customer_orders co2
LEFT JOIN runner_orders ro2
USING(order_id)
WHERE co2.extras_2 is not null and ro2.pickup_time is not null
GROUP BY 1,2)
)
, temp as
(
SELECT t.topping_name,
           CASE WHEN a.amount is null THEN 0
                 ELSE a.amount
           END as amounts,
           CASE WHEn e.counts is null THEN 0
                 ELSE e.counts
           END as exclusions,
           CASE WHEN e2.counts is null THEN 0
                 ELSE e2.counts
           END as extras
FROM toppings as t
      LEFT JOIN amounts as a
      USING(pizza_id)
           LEFT JOIN exclusive as e
           ON e.exclusions = t.topping_id and e.pizza_id =
t.pizza_id
                LEFT JOIN extra as e2
                ON e2.extras = t.topping_id and e2.pizza_id =
t.pizza_id
ORDER BY 1
)
SELECT t. topping_name,
           (sum(t.amounts)-sum(t.exclusions)+sum(extras)) as
quantity
FROM temp as t
GROUP BY 1
ORDER BY 2 DESC
```

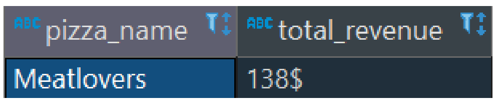


topping_name	quantity
Bacon	15
Mushrooms	11
Cheese	10
Beef	9
Chicken	9
Pepperoni	9
Salami	9
BBQ Sauce	8
Onions	3
Peppers	3
Tomatoes	3
Tomato Sauce	3

IV. 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?

```
WITH temp as
(
SELECT pn.pizza_name ,
           CASE WHEN pn.pizza_id = 1 THEN 12
                 ELSE 10
           END as price
FROM customer_orders co
      LEFT JOIN pizza_names pn
      USING(pizza_id)
           LEFT JOIN runner_orders ro
           USING(order_id)
WHERE ro.pickup_time is not null
)
SELECT t.pizza_name,
           CONCAT(sum(price),'$') as total_revenue
FROM temp as t
```

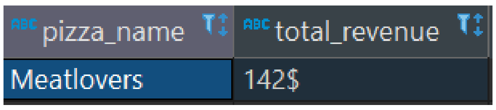


```
pizza_name   total_revenue
Meatlovers   138$
```

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

- Add cheese is $1 extra

```
WITH temp as
(
SELECT co.order_id ,
           pn.pizza_name ,
           CASE WHEN pn.pizza_id = 1 THEN 12
                 ELSE 10
           END as price,
           CASE WHEN co.extras_1 is null and co.extras_2 is null
THEN 0
                 WHEN co.extras_1 is not null and co.extras_2 is
not null THEN 2
                 ELSE 1
           END as extra_price
FROM customer_orders co
      LEFT JOIN pizza_names pn
      USING(pizza_id)
            LEFT JOIN runner_orders ro
            USING(order_id)
WHERE ro.pickup_time is not null
)
SELECT t.pizza_name,
            CONCAT(sum(price)+sum(extra_price),'$') as
total_revenue
FROM temp as t
```



pizza_name	total_revenue
Meatlovers	142$

3) The Pizza Runner team now wants to add an additional ratings system that allows customers to rate their runner, how would you design an additional table for this new dataset - generate a schema for this new table and insert your own data for ratings for each successful customer order between 1 to 5.

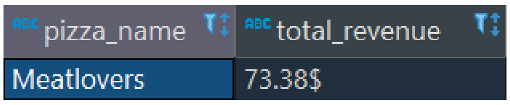
4) Using your newly generated table - can you join all of the information together to form a table which has the following information for successful deliveries?

- `customer_id`
- `order_id`
- `runner_id`
- `rating`
- `order_time`
- `pickup_time`
- Time between order and pickup
- Delivery duration
- Average speed
- Total number of pizzas

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?

```
WITH temp as
(
SELECT co.order_id ,
            pn.pizza_name ,
            CASE WHEN pn.pizza_id = 1 THEN 12
                   ELSE 10
            END as price,
            (ro.distance*0.3) as shipfees
FROM customer_orders co
      LEFT JOIN pizza_names pn
```

```
     USING(pizza_id)
          LEFT JOIN runner_orders ro
          USING(order_id)
WHERE ro.pickup_time is not null
)
SELECT t.pizza_name,
          CONCAT(ROUND(sum(price)-sum(shipfees),2),'$') as
total_revenue
FROM temp as t
```



pizza_name	total_revenue
Meatlovers	73.38$

V. E. Bonus Questions

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?