

**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.

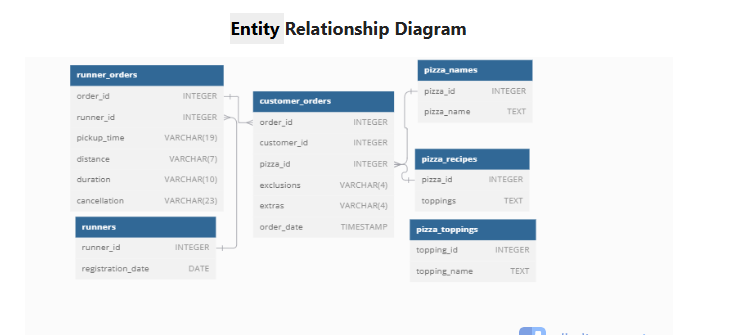
**Available Data**

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**



You can use the embedded DB Fiddle below to easily access these example datasets - this interactive session has everything you need to start solving these questions using SQL.

You can click on the Edit on DB Fiddle link on the top right hand corner of the embedded session below and it will take you to a fully functional SQL editor where you can write your own queries to analyse the data.

You can feel free to choose any SQL dialect you’d like to use, the existing Fiddle is using PostgreSQL 13 as default.

Serious SQL students can copy and paste the Schema SQL code below directly into their SQLPad GUI to create permanent tables or they can add a TEMP within the table creation steps like we’ve done throughout the entire course to keep our schemas clean and tidy!

**Schema**

**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');

**----------- -- -------Answers-------------------------------**

## 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)

Each of the following case study questions can be answered using a single SQL statement.

Again, there are many questions in this case study - please feel free to pick and choose which ones you’d like to try!

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 null values and data types in the customer\_orders and runner\_orders tables!

### A. Pizza Metrics

1. How many pizzas were ordered?
2. How many unique customer orders were made?
3. How many successful orders were delivered by each runner?
4. How many of each type of pizza was delivered?
5. How many Vegetarian and Meatlovers were ordered by each customer?
6. What was the maximum number of pizzas delivered in a single order?
7. For each customer, how many delivered pizzas had at least 1 change and how many had no changes?
8. How many pizzas were delivered that had both exclusions and extras?
9. What was the total volume of pizzas ordered for each hour of the day?
10. What was the volume of orders for each day of the week?

### B. Runner and Customer Experience

1. How many runners signed up for each 1 week period? (i.e. week starts 2021-01-01)
2. What was the average time in minutes it took for each runner to arrive at the Pizza Runner HQ to pickup the order?
3. Is there any relationship between the number of pizzas and how long the order takes to prepare?
4. What was the average distance travelled for each customer?
5. What was the difference between the longest and shortest delivery times for all orders?
6. What was the average speed for each runner for each delivery and do you notice any trend for these values?
7. What is the successful delivery percentage for each runner?

### C. Ingredient Optimisation

1. What are the standard ingredients for each pizza?
2. What was the most commonly added extra?
3. What was the most common exclusion?
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
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?

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

**Queries**

**----------------Pizza Metrics---------**

**select name from sys.tables**

**---1.How many pizzas were ordered?**

**select count(order\_id) as Pizza\_Count from customer\_orders**

**---2.How many unique customer orders were made?**

**select count(distinct order\_id) as Unique\_customer\_orders from customer\_orders**

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

**select runner\_id,count(order\_id) as Successful\_orders from runner\_orders where cancellation is not null group by runner\_id**

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

**select count(pizza\_id) as NoofPizzadelivered,pizzaname**

**from (select A.pizza\_id,CAST(B.pizza\_name as nvarchar(50)) as pizzaname from customer\_orders A inner join pizza\_names B on A.pizza\_id = B.pizza\_id ) as A**

**group by pizzaname**

**----How many Vegetarian and Meatlovers were ordered by each customer?**

**select customer\_id,count(customer\_id) as Customer,pizzaname From (select A.customer\_id,CAST(B.pizza\_name as nvarchar(50)) as pizzaname from customer\_orders A inner join pizza\_names B on A.pizza\_id = B.pizza\_id ) as A**

**group by customer\_id,pizzaname order by pizzaname**

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

**select Max(order\_count) as Order\_count from (select order\_id,count(customer\_id) as order\_count from customer\_orders group by order\_id) as A**

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

**SELECT customer\_id,**

**COUNT(CASE WHEN (exclusions IS NOT NULL AND exclusions <> '')**

**OR (extras IS NOT NULL AND extras <> '')**

**THEN 1 END) AS Noofchanges,**

**COUNT(CASE WHEN (exclusions IS NULL OR exclusions = '')**

**AND (extras IS NULL OR extras = '')**

**THEN 1 END) AS Noofnochanges**

**FROM customer\_orders**

**WHERE order\_id IN (SELECT order\_id**

**FROM runner\_orders**

**WHERE cancellation IS NULL OR LEN(cancellation) = 0)**

**GROUP BY customer\_id;**

**----------**

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

**select count(pizza\_id) as NOOFpizzadelivered from customer\_orders where len(exclusions)>=1 and len(extras)>=1**

**-------**

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

**select count(pizza\_id) as noofpizzas,Datepart(HOUR,order\_time) as Hour from customer\_orders group by Datepart(HOUR,order\_time)**

**order by Datepart(HOUR,order\_time)**

**--What was the volume of orders for each day of the week?**

**select count(pizza\_id) as noofpizzas,Datepart(WEEK,order\_time) as Week from customer\_orders group by Datepart(WEEK,order\_time)**

**order by Datepart(WEEK,order\_time)**

**-----. Runner and Customer Experience------------------**

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

**select**

**Dateadd(Week,Datediff(Week,'2021-01-01',registration\_date),'2021-01-01') as Week1,**

**count(runner\_id) as signups**

**from runners**

**group by Dateadd(Week,Datediff(Week,'2021-01-01',registration\_date),'2021-01-01')**

**--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(cast(SUBSTRING(duration,1,2) as INT)) as AvgMinutes from**

**runner\_orders where cancellation= '' or cancellation is null**

**group by runner\_id**

**--Is there any relationship between the**

**--number of pizzas and how long the order takes to prepare?**

**select \* from runner\_orders where order\_id = 4**

**select order\_id,count(order\_id) as Noofpizzas,**

**Avg(timetooktopickupinmin) as AVGtimetooktopickupinmin from (select A.order\_id,B.order\_id as orderidb,A.order\_time,B.pickup\_time,cancellation,**

**cast (DATEDIFF(MINUTE,order\_time,pickup\_time) as Int) as timetooktopickupinmin**

**from customer\_orders A left join runner\_orders B**

**on A.order\_id = B.order\_id) As A where cancellation = '' or cancellation is null**

**group by order\_id**

**--What was the average distance travelled for each customer?**

**select customer\_id,round(AVG(DistanceinKM),2) as DistanceTravelled from (select A.order\_id,B.customer\_id,distance,**

**cast(REPLACE(distance,'km','') as float) as DistanceinKM**

**from runner\_orders A left join customer\_orders B on**

**A.order\_id = B.order\_id where cancellation = '' or cancellation is null) as A**

**group by customer\_id**

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

**select Max(minutes) as longestdeliverytime,MIN(minutes) shortesteliverytime,**

**Max(minutes) - MIN(minutes) as difference from (select B.order\_id,B.order\_time,A.pickup\_time,**

**DATEDIFF(MINUTE,order\_time,pickup\_time) as minutes from runner\_orders A**

**left join customer\_orders B on A.order\_id = B.order\_id**

**where cancellation = '' or cancellation is null) As A**

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

**select order\_id,runner\_id,round (distance/Minutes,2) as speed from (select order\_id,runner\_id,cast(replace(distance,'km','') as float) as distance,cast(replace(replace(REPLACE(duration,'minutes',''),'mins',''),'minute','') as Int) as Minutes from runner\_orders**

**where cancellation = '' or cancellation is null) as A**

**---What is the successful delivery percentage for each runner?**

**select runner\_id,sum(case when cancellation = '' or cancellation is null then 1 else 0 end)\*100/count(order\_id) as successfuldeliverypercentage from runner\_orders group by runner\_id**

**----------C. Ingredient Optimisation-----------**

**--What are the standard ingredients for each pizza?**

**with cte as(**

**select A.pizza\_id,A.pizza\_name,trim(split\_toppings.value) as standard from pizza\_names A**

**inner join pizza\_recipes B on A.pizza\_id = B.pizza\_id**

**cross apply**

**string\_split(cast(B.toppings as varchar(50)),',') as split\_toppings)**

**select pizza\_id,pizza\_name,standard,topping\_name from cte**

**inner join pizza\_toppings C on cte.standard = C.topping\_id**

**------------**

**--What was the most commonly added extra?**

**select topping\_name,extra,most\_used from (select top 1 toppings.value as extra,count(trim(toppings.value)) as most\_used from customer\_orders A**

**cross apply**

**string\_split(A.extras,',') as toppings**

**where len(extras)>=1**

**group by toppings.value**

**order by count(trim(toppings.value)) desc) as A**

**inner join pizza\_toppings B on A.extra = B.topping\_id**

**----**

**--What was the most common exclusion?**

**select topping\_name,exclusion,most\_used from (select top 1 toppings.value as exclusion,count(trim(toppings.value)) as most\_used from customer\_orders A**

**cross apply**

**string\_split(A.exclusions,',') as toppings**

**where len(exclusions)>=1**

**group by toppings.value**

**order by count(trim(toppings.value)) desc) as A**

**inner join pizza\_toppings B on A.exclusion = B.topping\_id**

**------------**

**--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**

**select pizza\_name +' '+ case when exclusions = ''then '' else 'Exclude-' end +exclusions + case when Extras = '' then '' else ', Extra ' end + Extras as ordernames**

**from (select order\_id,pizza\_name,STRING\_AGG(Exclusions,',') AS exclusions,**

**STRING\_AGG(Extras,',') AS Extras**

**From**

**(**

**select order\_id,cast(pizza\_name as varchar(50)) as pizza\_name,cast(Isnull(C.topping\_name,'') as varchar(50)) as Exclusions,**

**cast(ISNULL(D.topping\_name,'') as varchar(50)) as Extras**

**from (select order\_id,A.pizza\_id,B.pizza\_name,A.exclusions,A.extras,**

**exclude\_toppings.value as exclusion\_topping,trim(extra\_toppings.value) as extra\_toppings**

**From customer\_orders A**

**inner join pizza\_names B on A.pizza\_id = B.pizza\_id**

**cross apply**

**string\_split(A.exclusions,',') as exclude\_toppings**

**cross apply**

**string\_split(A.extras,',') as extra\_toppings**

**) as data**

**left join pizza\_toppings C on data.exclusion\_topping = c.topping\_id**

**left join pizza\_toppings D on data.extra\_toppings = D.topping\_id) as A**

**group by order\_id,pizza\_name) as Orders**

**-------------**

**--D. Pricing and Ratings--**

**--If a Meat Lovers pizza costs $12 and Vegetarian costs $10 and there were no charges**

**--for changes - how much money has Pizza Runner made so far if there are no delivery fees?**

**select \* from customer\_orders**

**select \* from runner\_orders**

**select**

**runner\_id,**

**sum(case when pizza\_id =1 then 12 else 10 End) as Money\_madeindollars**

**from customer\_orders A**

**inner join runner\_orders B on A.order\_id = B.order\_id**

**where B.cancellation ='' or cancellation is null**

**group by runner\_id**

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

**--Add cheese is $1 extra**

**select topping\_id from pizza\_toppings where cast(topping\_name as varchar(50)) = 'Cheese'**

**select \*,**

**case**

**when pizza\_id=1 then 12**

**when pizza\_id =1 and CHARINDEX((select cast(topping\_id as varchar(10)) from pizza\_toppings where cast(topping\_name as varchar(50)) = 'Cheese'),extras)>=1 then 13**

**when pizza\_id=2 then 10**

**when pizza\_id =2 and CHARINDEX((select cast(topping\_id as varchar(10)) from pizza\_toppings where cast(topping\_name as varchar(50)) = 'Cheese'),extras)>=1 then 11**

**end as TotalPizzacost**

**from customer\_orders**

**---**

**--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.**

**create table customerratings**

**(**

**rating\_id Int Identity(1,1) Primary Key,**

**customer\_id varchar(50),**

**order\_id varchar(30),**

**rating int**

**)**

**insert into customerratings (customer\_id,order\_id,rating)**

**values**

**('101',1,3),**

**('101',2,4),**

**('102',3,2),**

**('103',4,5),**

**('104',5,4),**

**('105',7,5),**

**('102',8,4),**

**('104',10,5)**

**---------**

**--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**

**select customer\_id,order\_id,runner\_id,rating,order\_time,pickup\_time,**

**InMinutes as TimebetweenorderandpickupinMInutes,**

**DurationinMIns as [Delivery duration],DistanceinKM,**

**Avg(round(DistanceinKM \* (60/DurationinMIns),2)) as AVGSpeed,**

**NoofPizzas as [Total number of pizzas]**

**from(select A.customer\_id,A.order\_id,runner\_id,c.rating,order\_time,pickup\_time,**

**cast(DATEDIFF(MINUTE,order\_time,pickup\_time) as INT) as InMinutes,**

**cast(replace(replace(replace(duration,'minutes',''),'mins',''),'minute','') as float) as DurationinMIns,**

**cast(REPLACE(distance,'km','') as float) as DistanceinKM,**

**COUNT(pizza\_id) as NoofPizzas**

**from customer\_orders A**

**left join**

**runner\_orders B on A.order\_id = B.order\_id**

**left join customerratings C on A.order\_id = C.order\_id**

**where rating is not null**

**group by A.customer\_id,A.order\_id,runner\_id,rating,order\_time,pickup\_time,duration,distance**

**)As Data**

**group by customer\_id,order\_id,runner\_id,rating,order\_time,pickup\_time,**

**InMinutes,**

**DurationinMIns,DistanceinKM,NoofPizzas**

**--**

**--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?**

**select runner\_id,sum(cost+KMprice) as TotalPrice from (select**

**A.order\_id,**

**runner\_id,**

**case when pizza\_id = 1 then 12 else 10 End as cost,**

**cast(REPLACE(distance,'km','') as float) as DistanceinKM,**

**cast(REPLACE(distance,'km','') as float) \* 0.30 as KMprice**

**from customer\_orders A**

**inner join runner\_orders B on A.order\_id = B.order\_id**

**where B.cancellation = '' or B.cancellation is null**

**) as Data**

**----------**