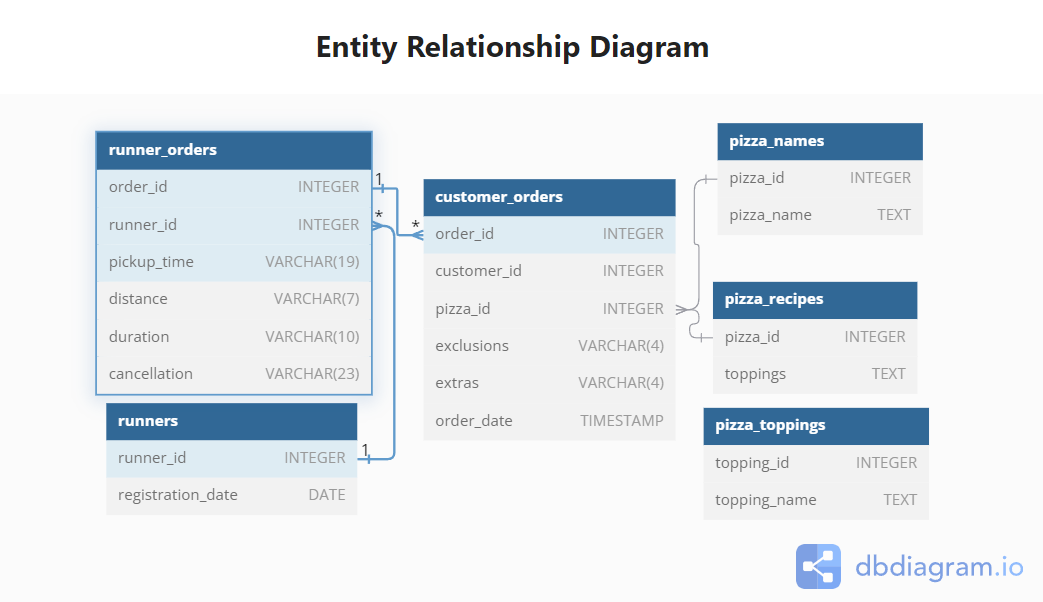
Pizza Runner SQL Challenge - PySpark Solution Documentation

<https://8weeksqlchallenge.com/case-study-2/>

**Case Study II - Pizza Runner (with a twist—I solved it using PySpark instead of SQL! 💻🍕)**





**Table 1: runners**

The runners table shows the registration\_date for each new runner

| **runner\_id** | **registration\_date** |
| --- | --- |
| 1 | 2021-01-01 |
| 2 | 2021-01-03 |
| 3 | 2021-01-08 |
| 4 | 2021-01-15 |

**Table 2: customer\_orders**

Customer pizza orders are captured in the customer\_orders table with 1 row for each individual pizza that is part of the order.

The pizza\_id relates to the type of pizza which was ordered whilst the exclusions are the ingredient\_id values which should be removed from the pizza and the extras are the ingredient\_id values which need to be added to the pizza.

Note that customers can order multiple pizzas in a single order with varying exclusions and extras values even if the pizza is the same type!

The exclusions and extras columns will need to be cleaned up before using them in your queries.

| **order\_id** | **customer\_id** | **pizza\_id** | **exclusions** | **extras** | **order\_time** |
| --- | --- | --- | --- | --- | --- |
| 1 | 101 | 1 |  |  | 2021-01-01 18:05:02 |
| 2 | 101 | 1 |  |  | 2021-01-01 19:00:52 |
| 3 | 102 | 1 |  |  | 2021-01-02 23:51:23 |
| 3 | 102 | 2 |  | NaN | 2021-01-02 23:51:23 |
| 4 | 103 | 1 | 4 |  | 2021-01-04 13:23:46 |
| 4 | 103 | 1 | 4 |  | 2021-01-04 13:23:46 |
| 4 | 103 | 2 | 4 |  | 2021-01-04 13:23:46 |
| 5 | 104 | 1 | null | 1 | 2021-01-08 21:00:29 |
| 6 | 101 | 2 | null | null | 2021-01-08 21:03:13 |
| 7 | 105 | 2 | null | 1 | 2021-01-08 21:20:29 |
| 8 | 102 | 1 | null | null | 2021-01-09 23:54:33 |
| 9 | 103 | 1 | 4 | 1, 5 | 2021-01-10 11:22:59 |
| 10 | 104 | 1 | null | null | 2021-01-11 18:34:49 |
| 10 | 104 | 1 | 2, 6 | 1, 4 | 2021-01-11 18:34:49 |

**Table 3: runner\_orders**

After each orders are received through the system - they are assigned to a runner - however not all orders are fully completed and can be cancelled by the restaurant or the customer.

The pickup\_time is the timestamp at which the runner arrives at the Pizza Runner headquarters to pick up the freshly cooked pizzas. The distance and duration fields are related to how far and long the runner had to travel to deliver the order to the respective customer.

There are some known data issues with this table so be careful when using this in your queries - make sure to check the data types for each column in the schema SQL!

| **order\_id** | **runner\_id** | **pickup\_time** | **distance** | **duration** | **cancellation** |
| --- | --- | --- | --- | --- | --- |
| 1 | 1 | 2021-01-01 18:15:34 | 20km | 32 minutes |  |
| 2 | 1 | 2021-01-01 19:10:54 | 20km | 27 minutes |  |
| 3 | 1 | 2021-01-03 00:12:37 | 13.4km | 20 mins | NaN |
| 4 | 2 | 2021-01-04 13:53:03 | 23.4 | 40 | NaN |
| 5 | 3 | 2021-01-08 21:10:57 | 10 | 15 | NaN |
| 6 | 3 | null | null | null | Restaurant Cancellation |
| 7 | 2 | 2020-01-08 21:30:45 | 25km | 25mins | null |
| 8 | 2 | 2020-01-10 00:15:02 | 23.4 km | 15 minute | null |
| 9 | 2 | null | null | null | Customer Cancellation |
| 10 | 1 | 2020-01-11 18:50:20 | 10km | 10minutes | null |

**Table 4: pizza\_names**

At the moment - Pizza Runner only has 2 pizzas available the Meat Lovers or Vegetarian!

| **pizza\_id** | **pizza\_name** |
| --- | --- |
| 1 | Meat Lovers |
| 2 | Vegetarian |

**Table 5: pizza\_recipes**

Each pizza\_id has a standard set of toppings which are used as part of the pizza recipe.

| **pizza\_id** | **toppings** |
| --- | --- |
| 1 | 1, 2, 3, 4, 5, 6, 8, 10 |
| 2 | 4, 6, 7, 9, 11, 12 |

**Table 6: pizza\_toppings**

This table contains all of the topping\_name values with their corresponding topping\_id value

| **topping\_id** | **topping\_name** |
| --- | --- |
| 1 | Bacon |
| 2 | BBQ Sauce |
| 3 | Beef |
| 4 | Cheese |
| 5 | Chicken |
| 6 | Mushrooms |
| 7 | Onions |
| 8 | Pepperoni |
| 9 | Peppers |
| 10 | Salami |
| 11 | Tomatoes |
| 12 | Tomato Sauce |

**Solution:**

# Importing Libraries:

from pyspark.sql.functions import \*

from pyspark.sql.types import \*

from pyspark.sql.window import \*

import datetime

# Creating Schemas and DataFrames

##Runner Table

schema = StructType(

    [

        StructField("Runner\_ID", IntegerType(), True),

        StructField("Reg\_date", DateType(), True)

    ]

)

#Runners\_Data

data= [ (1, datetime.date(2021,1,1)),

  (2, datetime.date(2021,1,3)),

  (3, datetime.date(2021,1,8)),

  (4, datetime.date(2021,1,15))]

df\_run= spark.createDataFrame(data,schema)

#Customer Order

schema1= StructType (

    [

        StructField("Order\_Id", IntegerType(), True),

        StructField("Customer\_ID", IntegerType(), True),

        StructField("Pizza\_ID", IntegerType(), True),

        StructField("Exclusions", StringType(), True),

        StructField("Extras", StringType(), True),

        StructField("Order\_time", TimestampType(), True)

    ]

)

#Customer Orders\_Data

data1= [

      (1, 101, 1, None, None, datetime.datetime(2020,1,1,18,5,2)),

  (2, 101,1, None,None, datetime.datetime(2020,1,1,19,0,52)),

  (3, 102, 1, None,None, datetime.datetime(2020,1,2,23,51,23)),

  (3, 102, 2, None, None, datetime.datetime(2020,1,2,23,51,23)),

  (4, 103, 1, 4, None, datetime.datetime(2020,1,4,13,23,46)),

  (4, 103, 1, 4, None, datetime.datetime(2020,1,4,13,23,46)),

  (4, 103, 2, 4, None, datetime.datetime(2020,1,4,13,23,46)),

  (5, 104, 1, None, 1, datetime.datetime(2020,1,8,21,0,29)),

  (6, 101, 2, None, None, datetime.datetime(2020,1,8,21,3,13)),

  (7, 105, 2, None, 1, datetime.datetime(2020,1,8,21,20,29)),

  (8, 102, 1, None, None, datetime.datetime(2020,1,9,23,54,33)),

  (9, 103, 1, 4, '1, 5', datetime.datetime(2020,1,10, 11,22,59)),

  (10, 104, 1, None, None, datetime.datetime(2020,1,11,18,34,49)),

  (10, 104, 1, '2, 6', '1, 4', datetime.datetime(2020,1,11,18,34,49))

]

df\_cust\_ord= spark.createDataFrame(data1,schema1)

#Runner Order Table

schema2= StructType(

    [

        StructField("Order\_Id", IntegerType(), True),

        StructField("Runner\_ID", IntegerType(), True),

        StructField("Pickup\_time", TimestampType(), True),

        StructField("Distance", StringType(), True),

        StructField("Duration", StringType(), True),

        StructField("Cancellation", StringType(), True),

    ]

)

data2=[

    (1, 1, datetime.datetime(2020,1,1,18,15,34), '20km', '32 minutes', None),

  (2, 1, datetime.datetime(2020,1,1,19,10,54), '20km', '27 minutes', None),

  (3, 1, datetime.datetime(2020,1,3,0,12,37), '13.4km', '20 mins', None),

  (4, 2, datetime.datetime(2020,1,4,13,53,3), '23.4', '40', None),

  (5, 3, datetime.datetime(2020,1,8,21,10,57), '10', '15', None),

  (6, 3, None, None, None, 'Restaurant Cancellation'),

  (7, 2, datetime.datetime(2020,1,8,21,30,45), '25km', '25mins', None),

  (8, 2, datetime.datetime(2020,1,10,0,15,2), '23.4 km', '15 minute', None),

  (9, 2, None, None, None, 'Customer Cancellation'),

  (10, 1, datetime.datetime(2020,1,11,18,50,20), '10km', '10minutes', None)

]

df\_run\_ord= spark.createDataFrame(data2,schema2)

#Pizza\_names

schema3= StructType(

    [

        StructField("Pizza\_ID",IntegerType(), True),

        StructField("Pizza\_Name", StringType(), True)

    ]

)

data3 = [

     (1, 'Meatlovers'),

     (2, 'Vegetarian')

]

df\_pizza = spark.createDataFrame(data3,schema3)

#Pizza Recipes

schema4= StructType(

    [

        StructField("Pizza\_ID", IntegerType(), True),

        StructField("Toppings", StringType(), True)

    ]

)

data4=[

    (1, '1, 2, 3, 4, 5, 6, 8, 10'),

    (2, '4, 6, 7, 9, 11, 12')

]

df\_rec = spark.createDataFrame(data4,schema4)

#Pizza Toppings

schema5= StructType(

    [

        StructField("Topping\_ID", IntegerType(), True),

        StructField("Topping\_Name", StringType(), True)

    ]

)

data5=[

    (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')

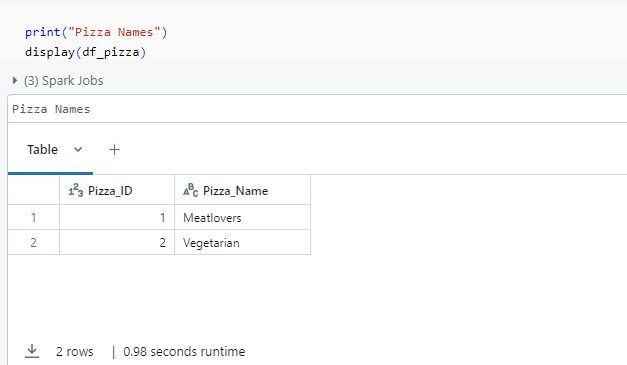
]

df\_top = spark.createDataFrame(data5,schema5)

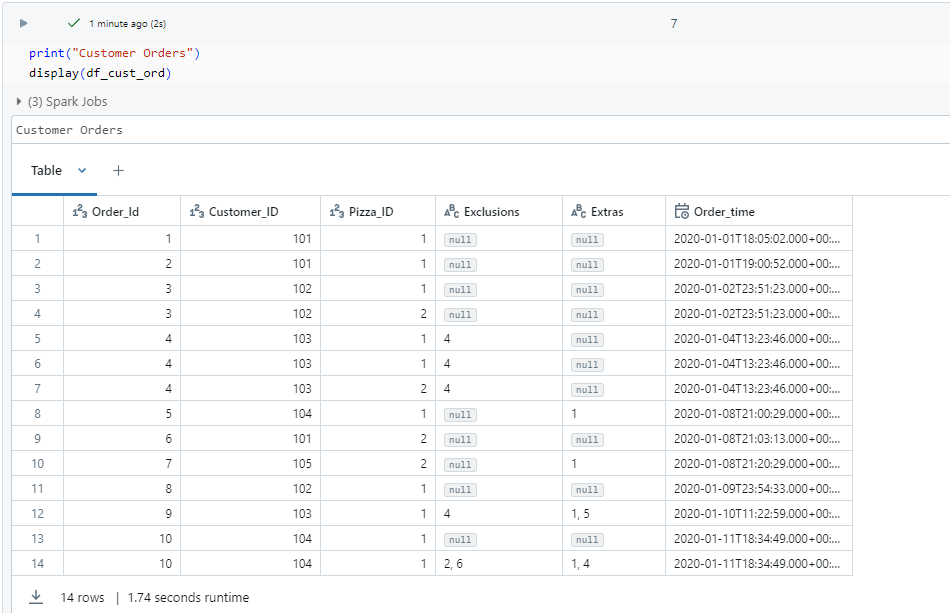
# Displaying the DataFrames:



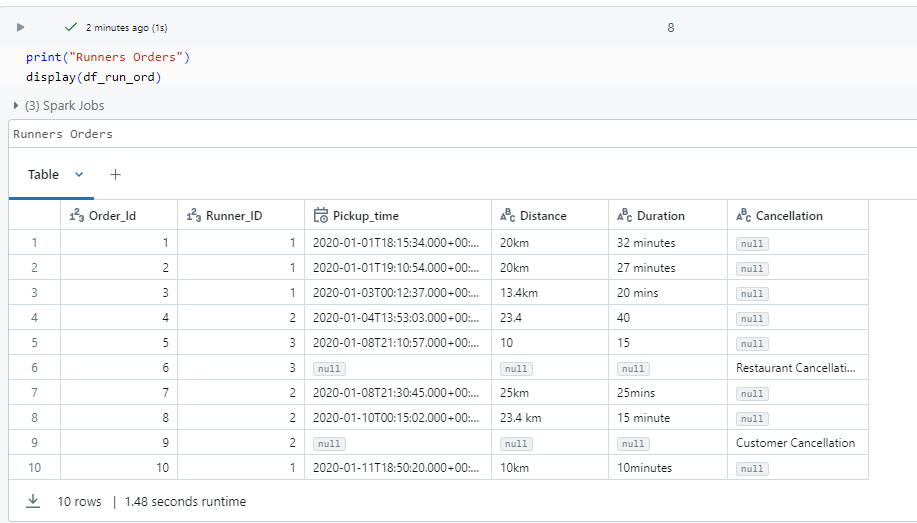
Pizza\_Names

****

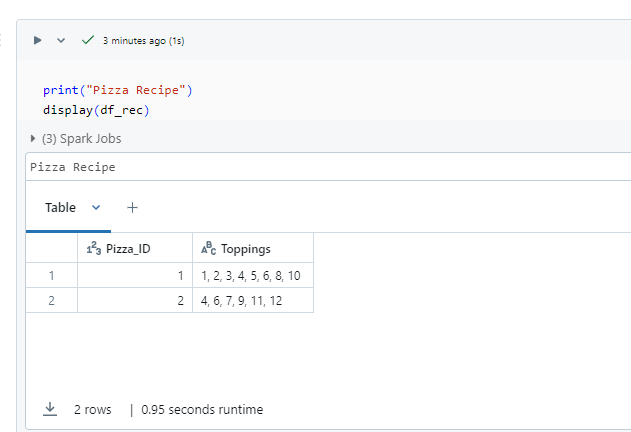
Customer\_Orders



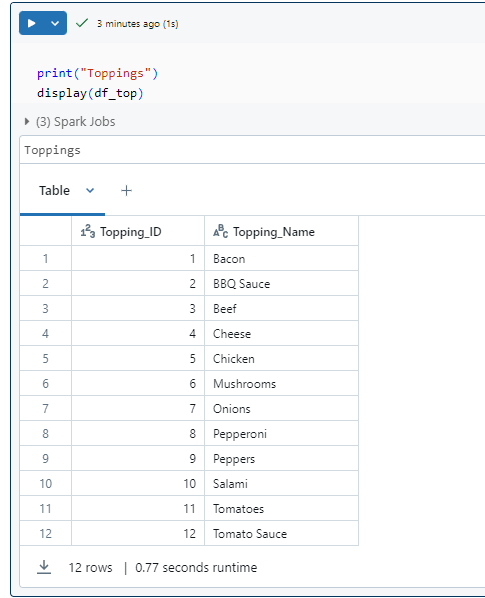
Runners\_Orders



**Pizza Recipe:**

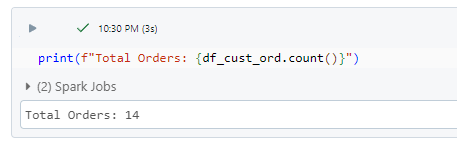
****

**Pizza\_Toppings**

****

**Pizza Metrics:**

1. How many pizzas were ordered?



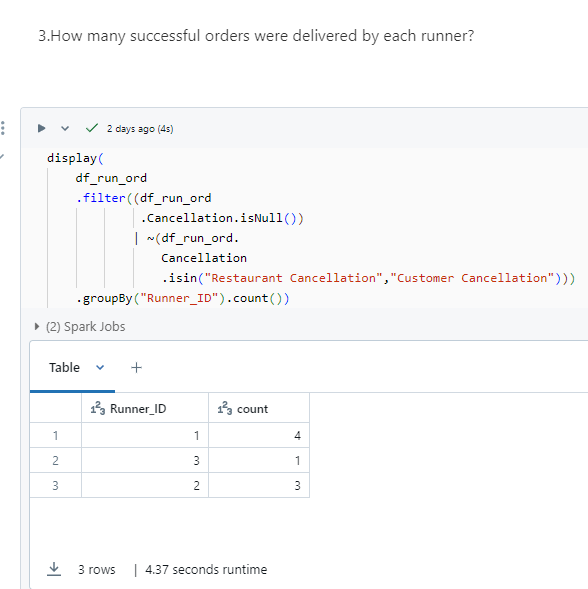
* Total of 14 pizzas were ordered.

1. How many unique customer orders were made?

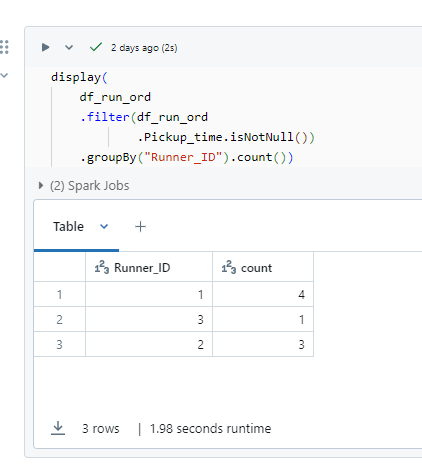


* There are 10 unique customer orders.

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

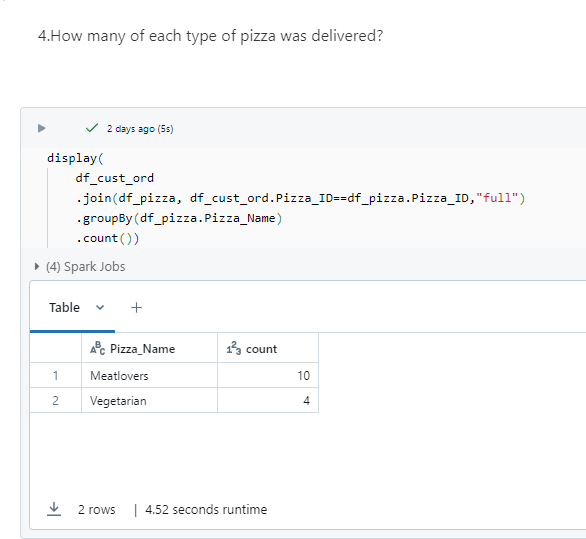


OR



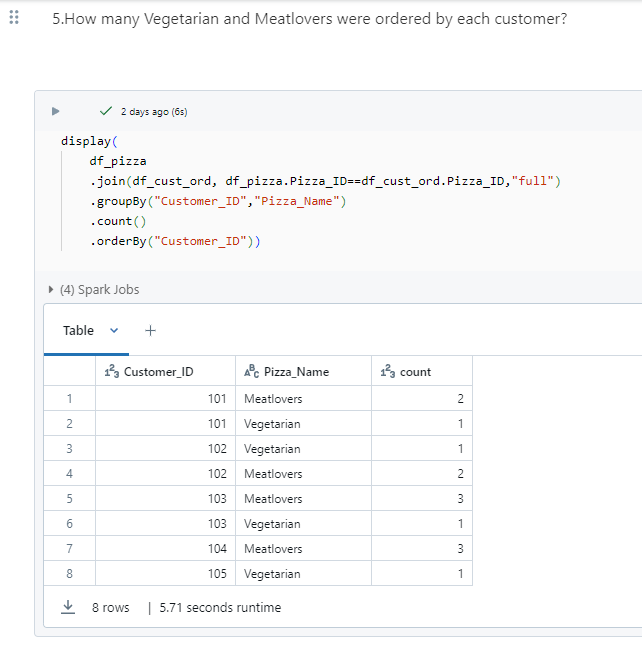
* Runner 1 has 4 successful delivered orders.
* Runner 2 has 3 successful delivered orders.
* Runner 3 has 1 successful delivered order.

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



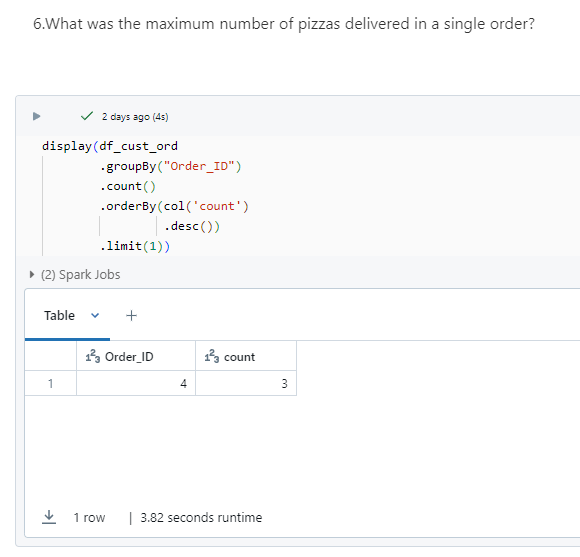
* There are 10 delivered Meatlovers pizzas and 4 Vegetarian pizzas.

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



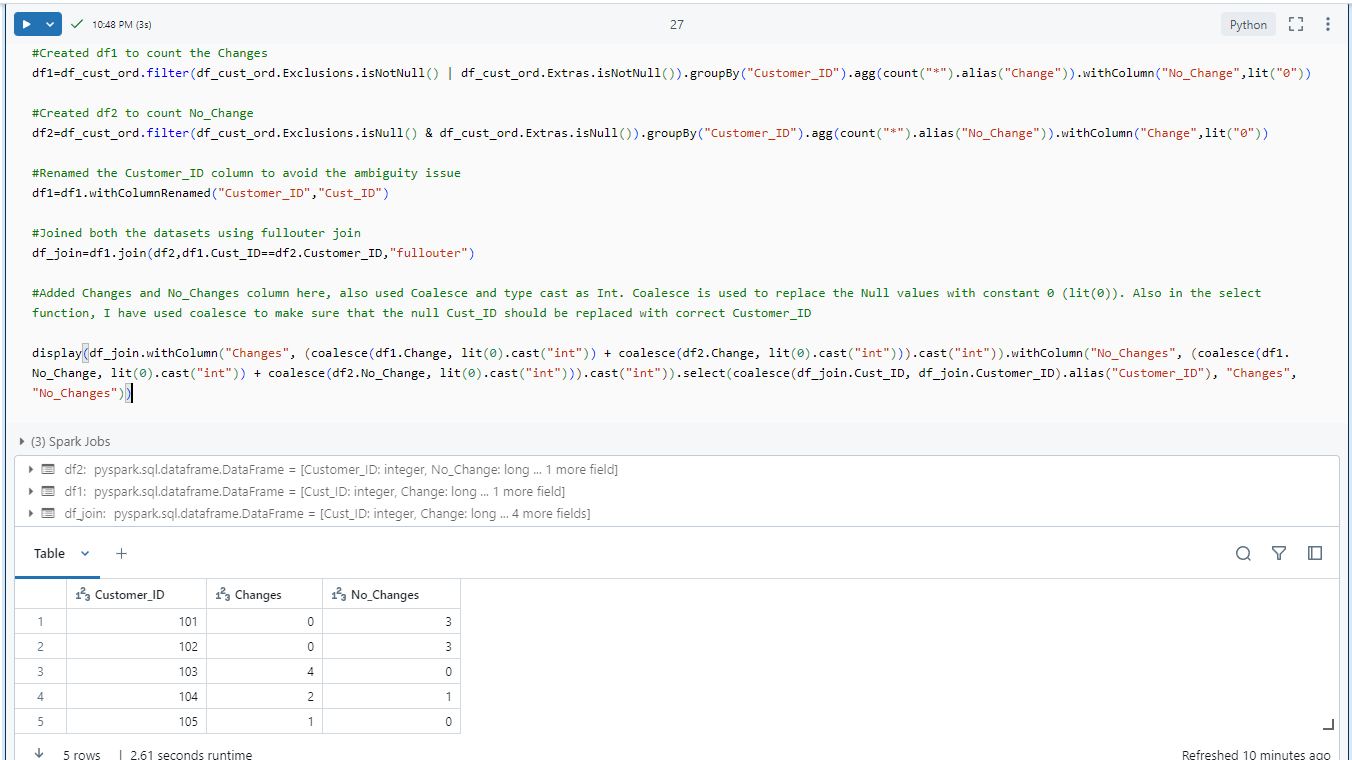
* Customer 101 ordered 2 Meatlovers pizzas and 1 Vegetarian pizza.
* Customer 102 ordered 2 Meatlovers pizzas and 1 Vegetarian pizzas.
* Customer 103 ordered 3 Meatlovers pizzas and 1 Vegetarian pizza.
* Customer 104 ordered 1 Meatlovers pizza.
* Customer 105 ordered 1 Vegetarian pizza.

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



* Maximum number of pizza delivered in a single order is 3 pizzas.

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



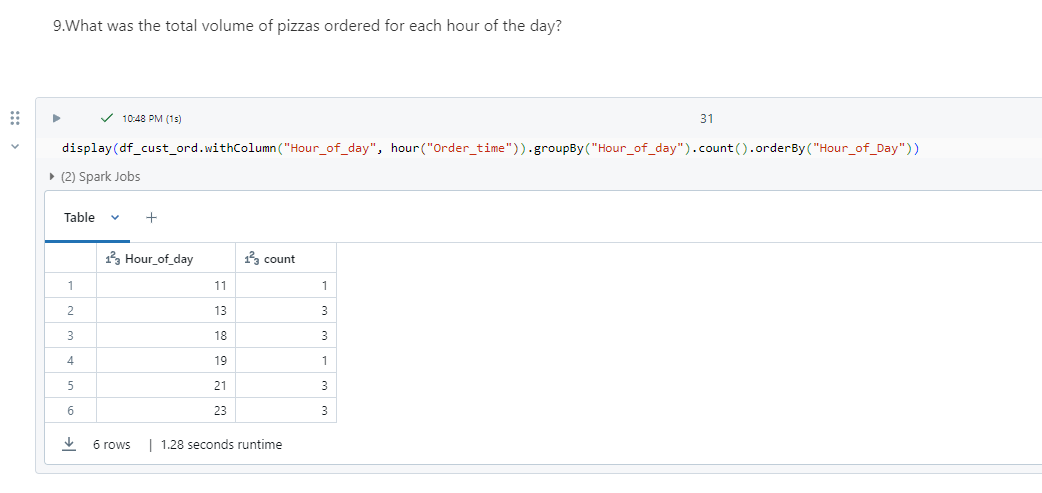
* Customer 101 and 102 likes his/her pizzas per the original recipe.
* Customer 103, 104 and 105 have their own preference for pizza topping and requested at least 1 change (extra or exclusion topping) on their pizza.

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



* 2 Pizzas delivered that had both extra and exclusion toppings (Customer 103 and 104)

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

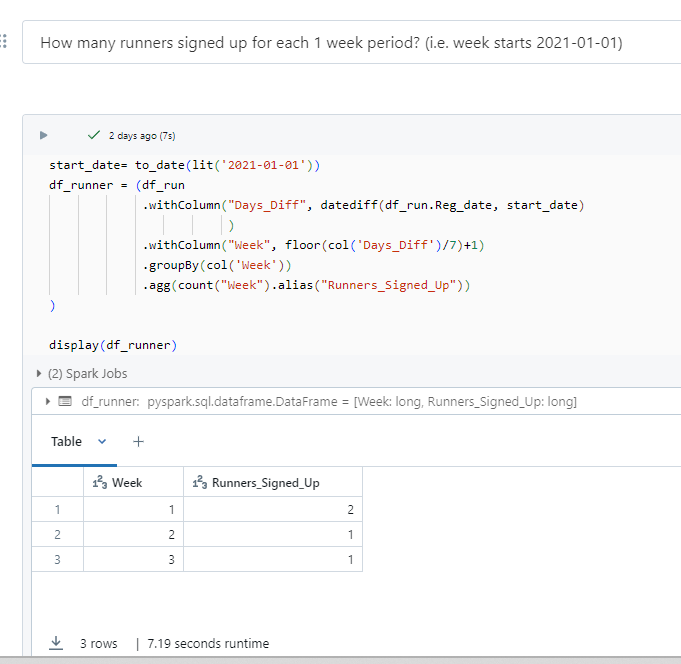


1. What was the volume of orders for each day of the week?

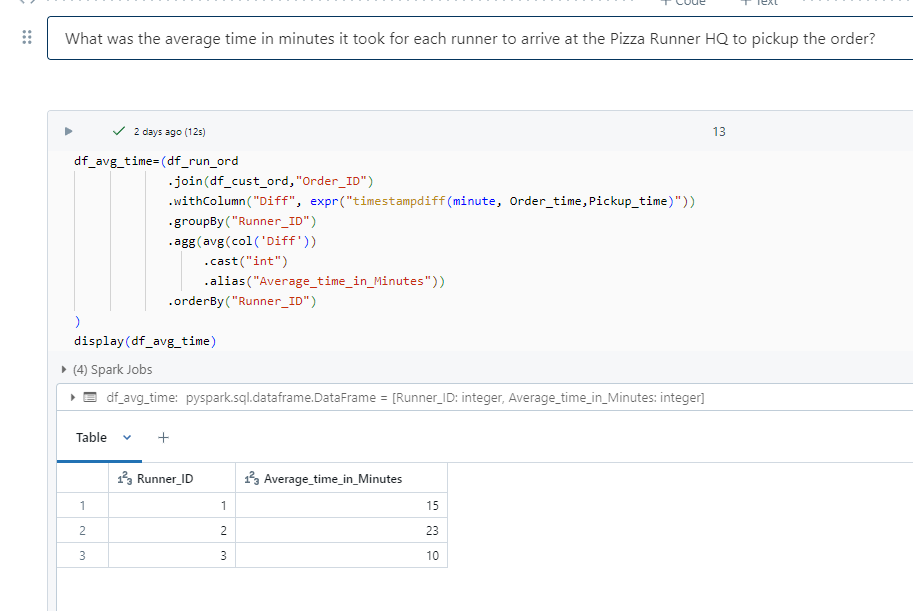


**Part II - Runner And Customer Experience**

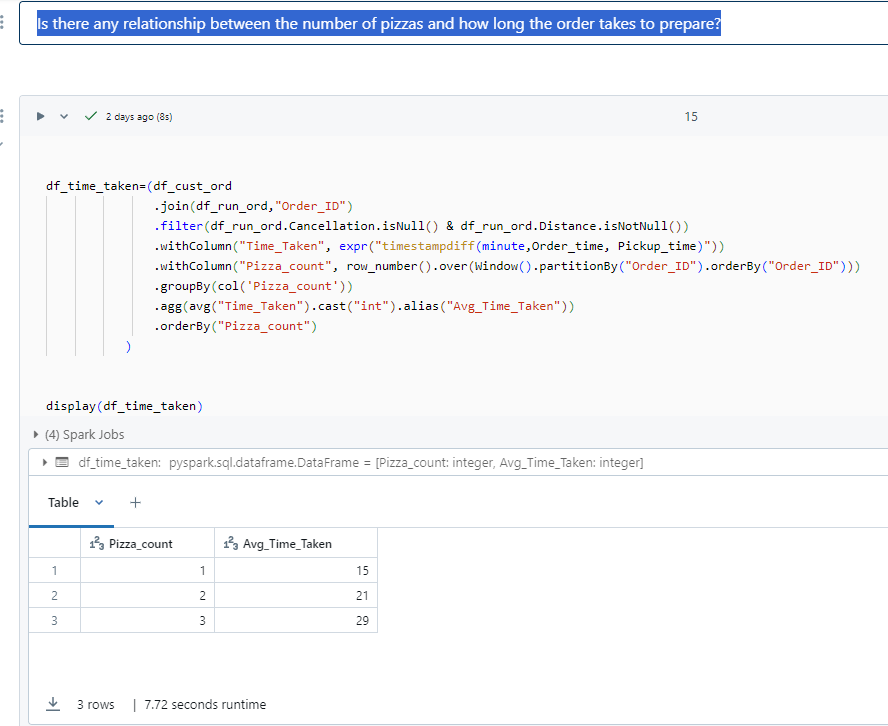
* 1. How many runners signed up for each 1 week period? (i.e. week starts 2021-01-01)



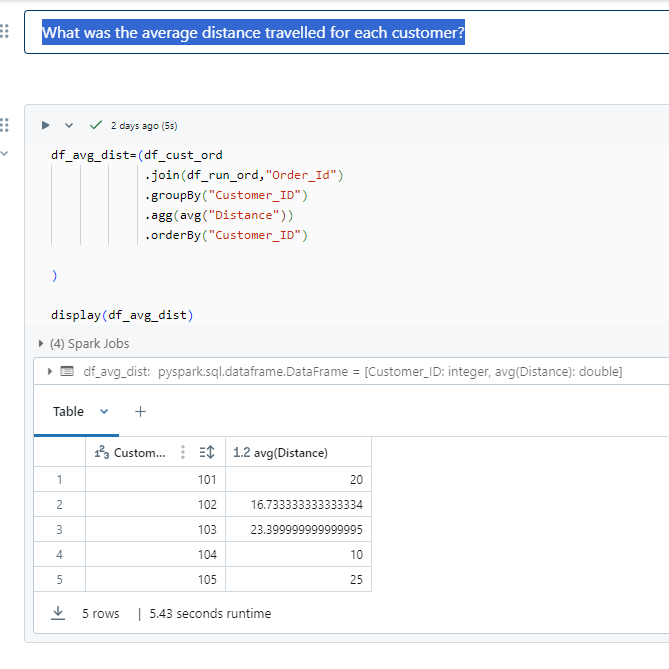
* 1. What was the average time in minutes it took for each runner to arrive at the Pizza Runner HQ to pickup the order?



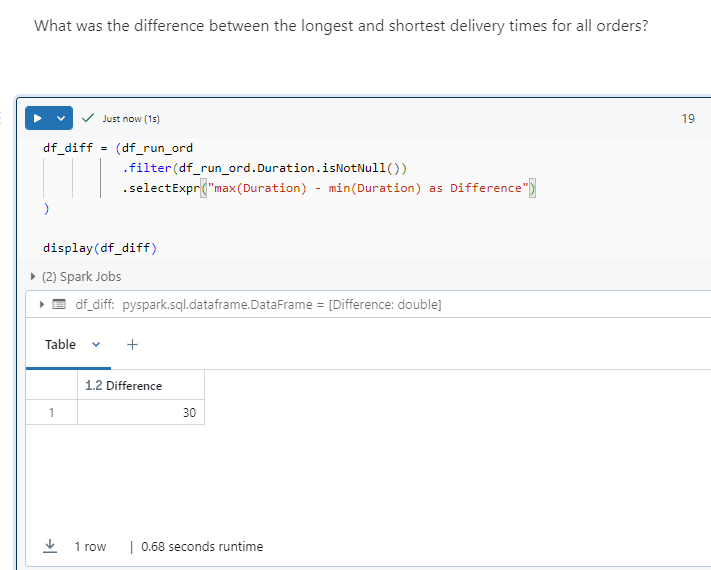
* 1. Is there any relationship between the number of pizzas and how long the order takes to prepare?



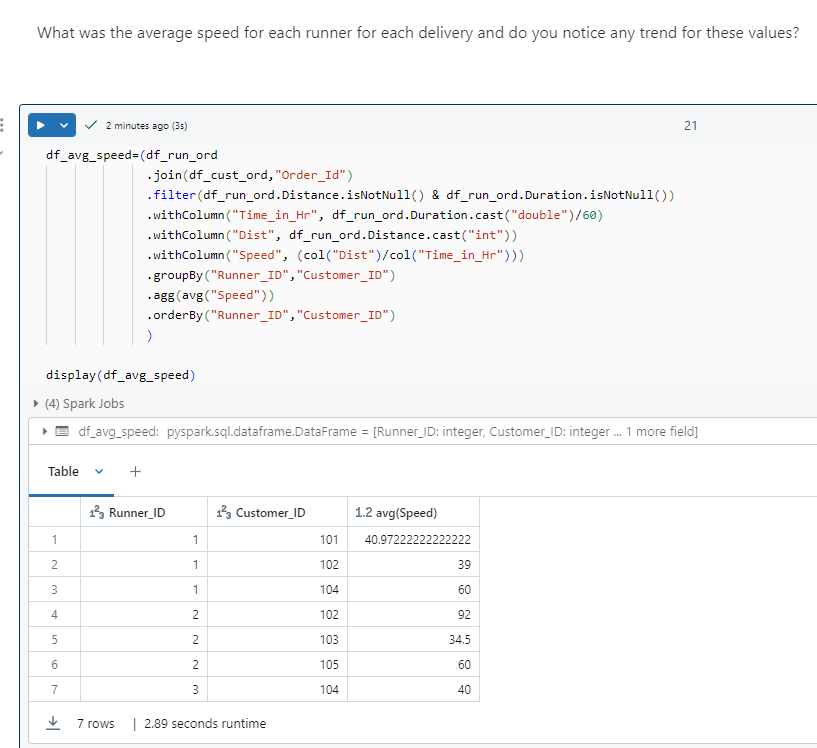
* 1. What was the average distance travelled for each customer?



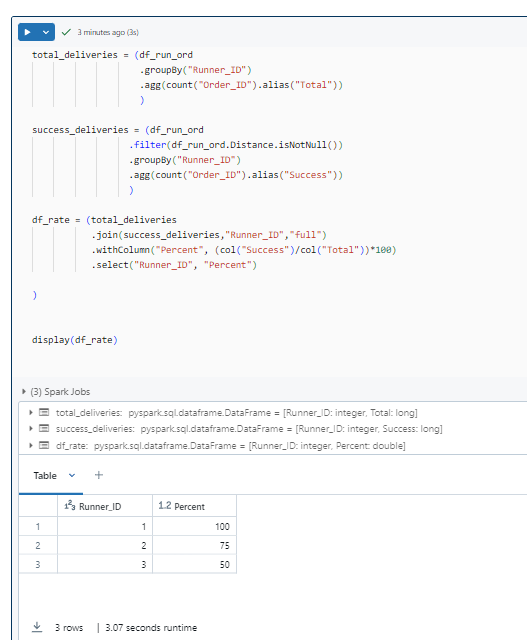
* 1. What was the difference between the longest and shortest delivery times for all orders?



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

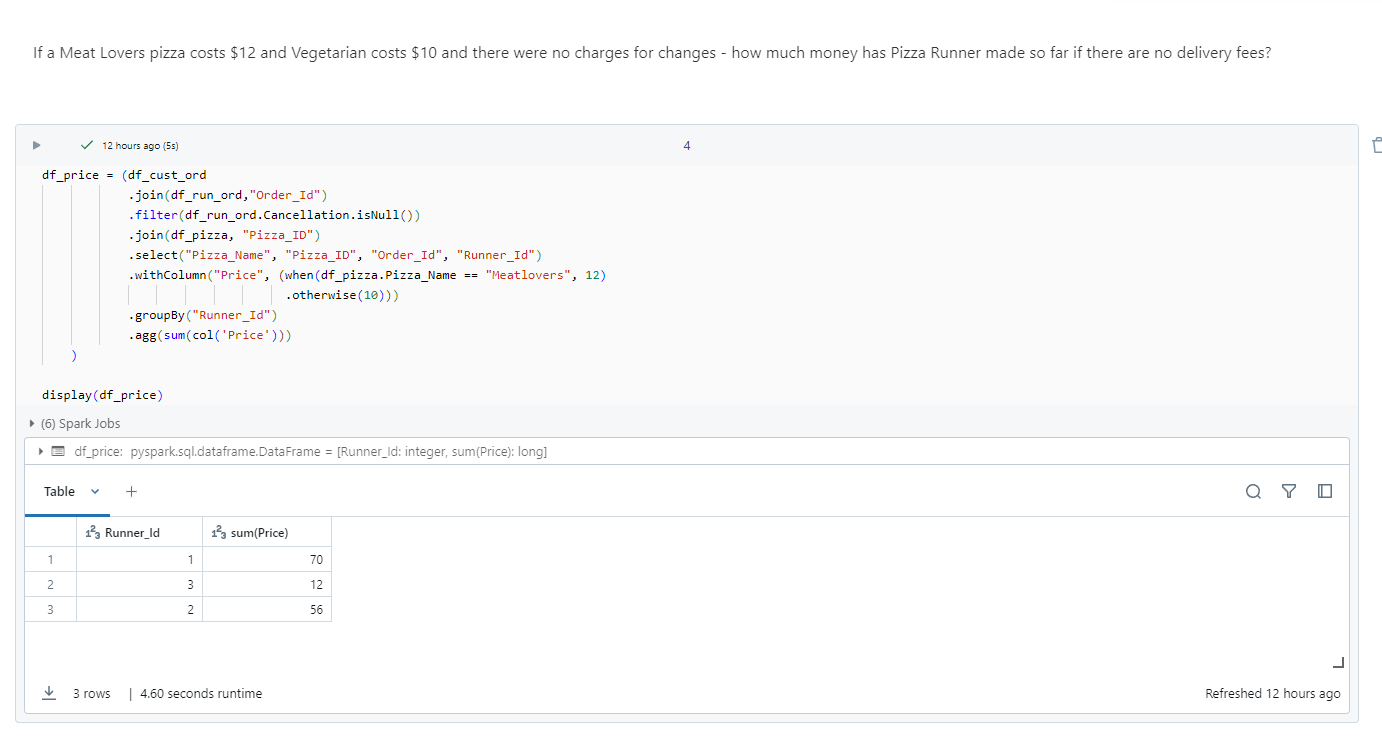


* 1. What is the successful delivery percentage for each runner?

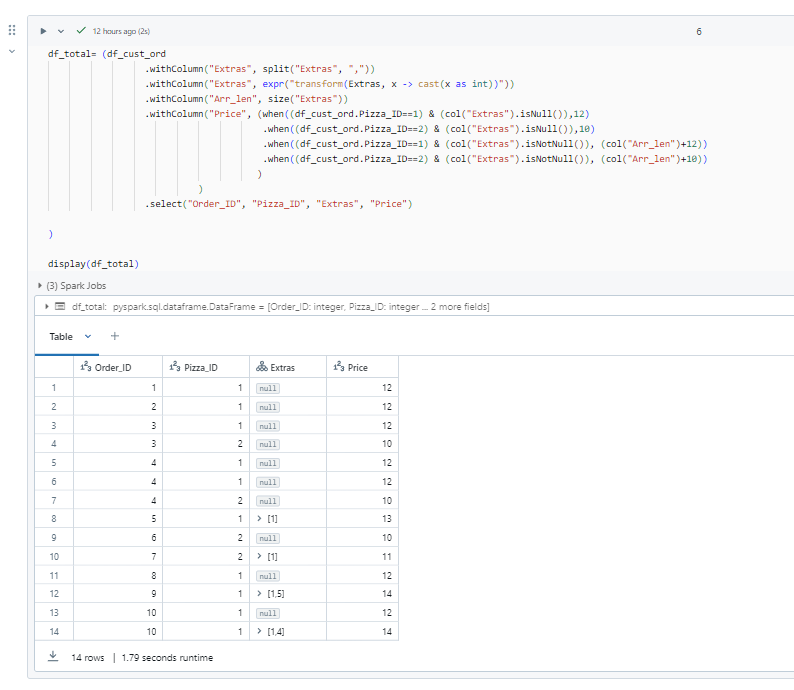


**Part III - Pricing and Ratings**

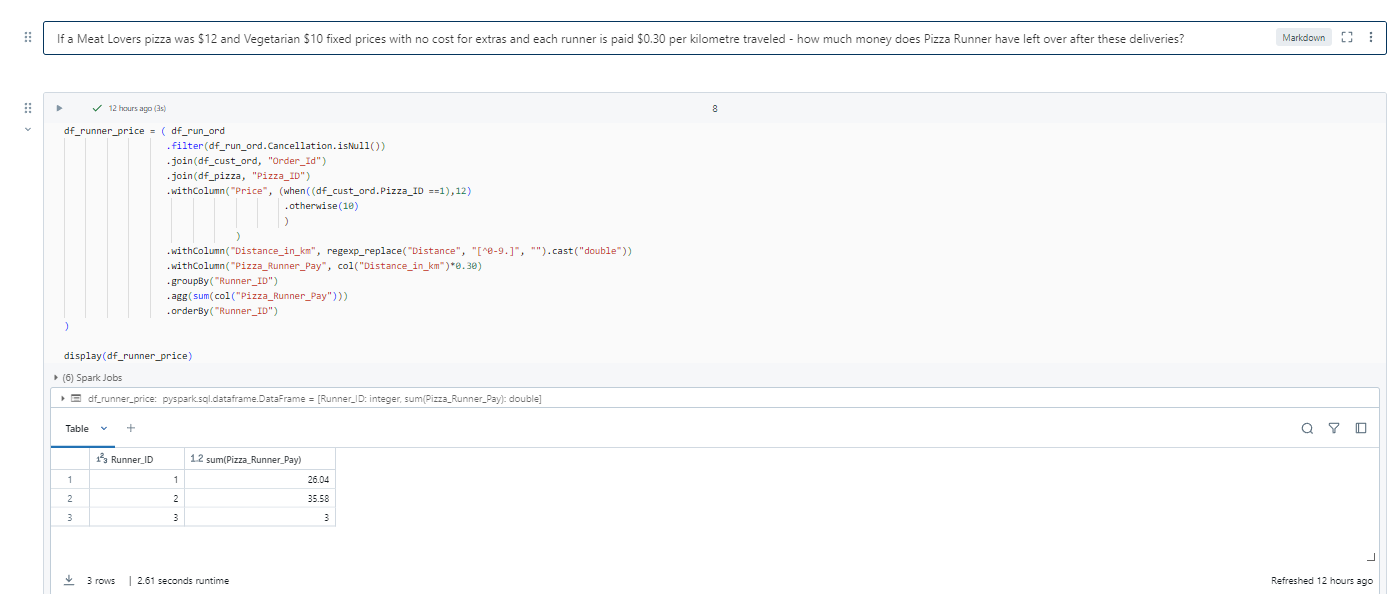
* 1. If a Meat Lovers pizza costs $12 and Vegetarian costs $10 and there were no charges for changes - how much money has Pizza Runner made so far if there are no delivery fees?



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

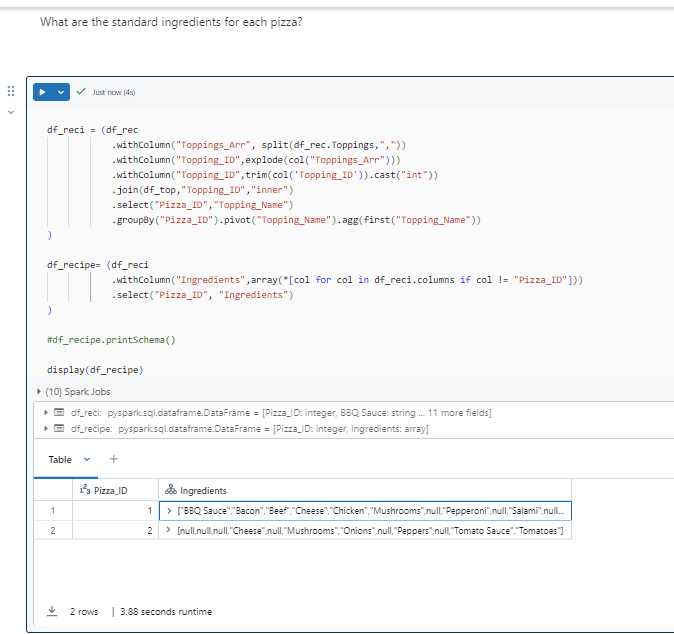


* 1. If a Meat Lovers pizza was $12 and Vegetarian $10 fixed prices with no cost for extras and each runner is paid $0.30 per kilometre travelled - how much money does Pizza Runner have left over after these deliveries?

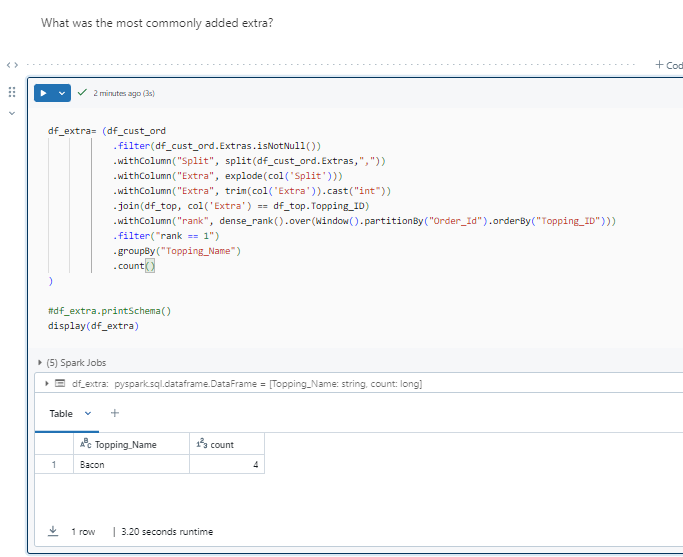


**Part IV - Ingredient Optimization**

* 1. What are the standard ingredients for each pizza?



* 1. What was the most commonly added extra?



* 1. What was the most common exclusion?

