Problem Statement:

From a database of IOT data for bicycles in Chicago, determine a business process for redistributing bicycles to stations to maintain low variance in bikes available and optimize for fewest number of bicycles transported.

Data Structure and Exploratory Data Analysis:

We have two major tables. The first is the bicycle log data (citibike), which is our fact table for customer rides. The second is the station table, our dimension table that supplies descriptive information about each bicycle station.

The citibike ride table is shown below in figure 1, and the station table in figure 2.

A screenshot of a computer

AI-generated content may be incorrect.

Fig 1. Structure of citibike table.

A screenshot of a computer

AI-generated content may be incorrect.

Fig 2. Structure of station table.

Here is a screenshot of the first few records in the citibike table.

A screenshot of a computer

AI-generated content may be incorrect.

And similarly for the station table:

A screenshot of a computer

AI-generated content may be incorrect.

Now, let us gather some information about the data. For a related case study, we may wish to analyze trends among members vs. casual riders. For the present study, we will just enumerate the rides taken by each, just to give us an idea of what type of customer ride is most common.

A screenshot of a computer program

AI-generated content may be incorrect.

So, the vast majority of rides are by members.

We also want to know the data collection period … what is the date range for all the rides? We can accomplish this by finding the earliest and latest records for the rides.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

So, the ride data encompasses one month near the end of 2024.

Finally, let’s find how many unique stations we have in this dataset.

A screenshot of a computer

AI-generated content may be incorrect.

It is useful to understand some basic metric like the above: how many records we are working with, what is the time span of the data, etc.

Now, we will dive into some deeper analysis. I want to understand how long these rides last, and how they are distributed. To do that, we first need to computer time duration of each ride. For this, we can use the epoch keyword.

A screenshot of a computer

AI-generated content may be incorrect.

You can verify that the first record in this relation (18 minutes) is equal to 1080 seconds (second column).

We update the ride table, filling in the duration column with values using the epoch. The result is shown below.

A screenshot of a computer

AI-generated content may be incorrect.

To visualize how rides are distributed by duration, I will create a histogram. An efficient way to to this when you want a large number of bins is to use a recursive query that creates the bins.

A screenshot of a computer

AI-generated content may be incorrect.

Here, we have 60 bins of size 60 seconds. The third record in this relation shows that 8,088 rides had a duration between 2 and 3 minutes.

We can encapsulate this query into a view for reuse as needed.

A screenshot of a computer code

AI-generated content may be incorrect.

Here is a visualization of this results set.

A screenshot of a graph

AI-generated content may be incorrect.

Most of the rides are only between 3 and 15 minutes. Finally, it might be useful to know, for any set of ride data, the duration that bounds 90% of the data. To do this, we can create a common curve in mathematics called a CDF (cumulative Density Function). This function tells us the percentage of data that falls below each value of interest (in our case, time).

We can create a CDF for a distribution by utilizing SQL window functions as follows.

A screenshot of a computer

AI-generated content may be incorrect.

We can compute the percentages as follows.

A screenshot of a computer code

AI-generated content may be incorrect.

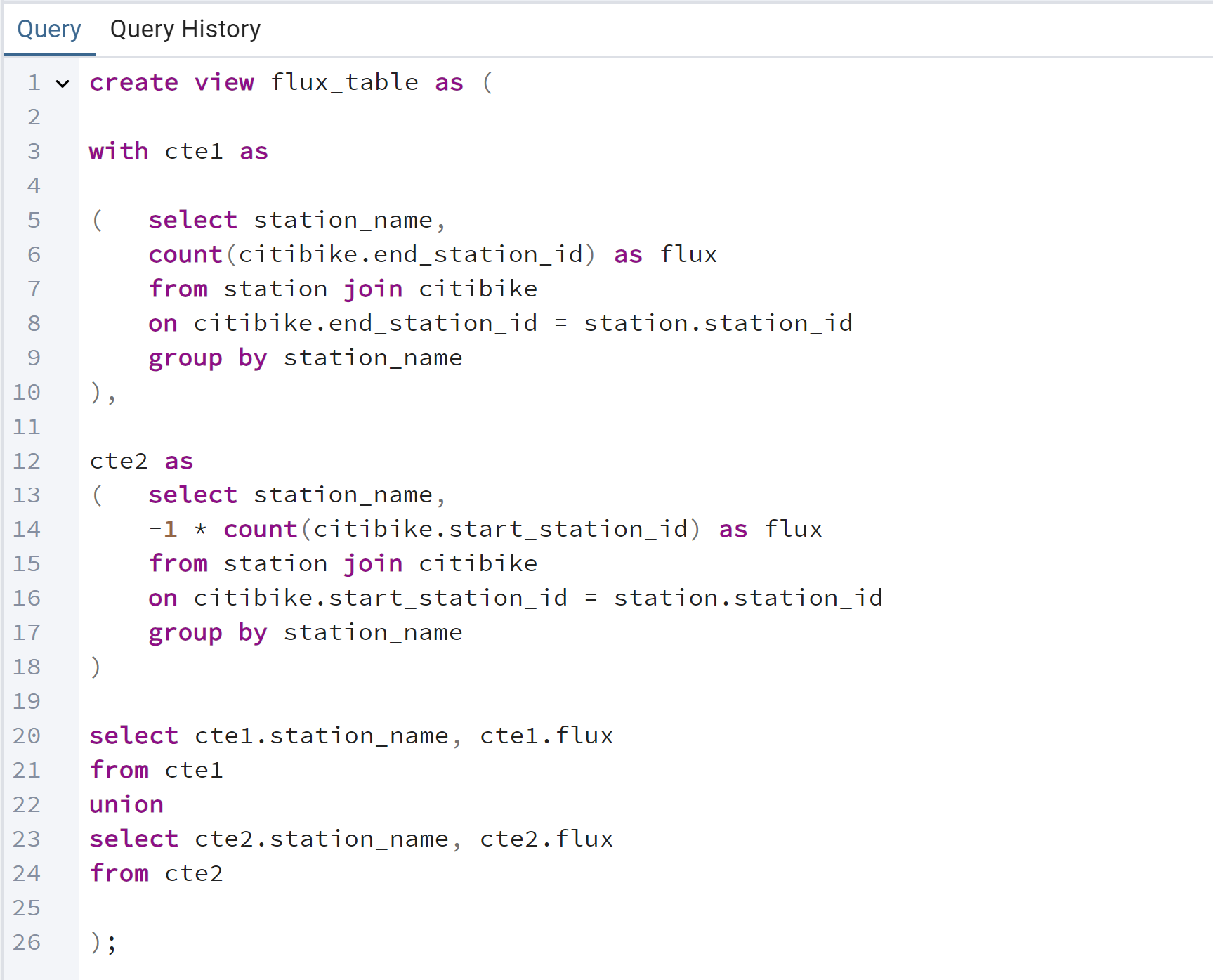
This gives us a nice CDF for the data.

A graph with a line graph

AI-generated content may be incorrect.

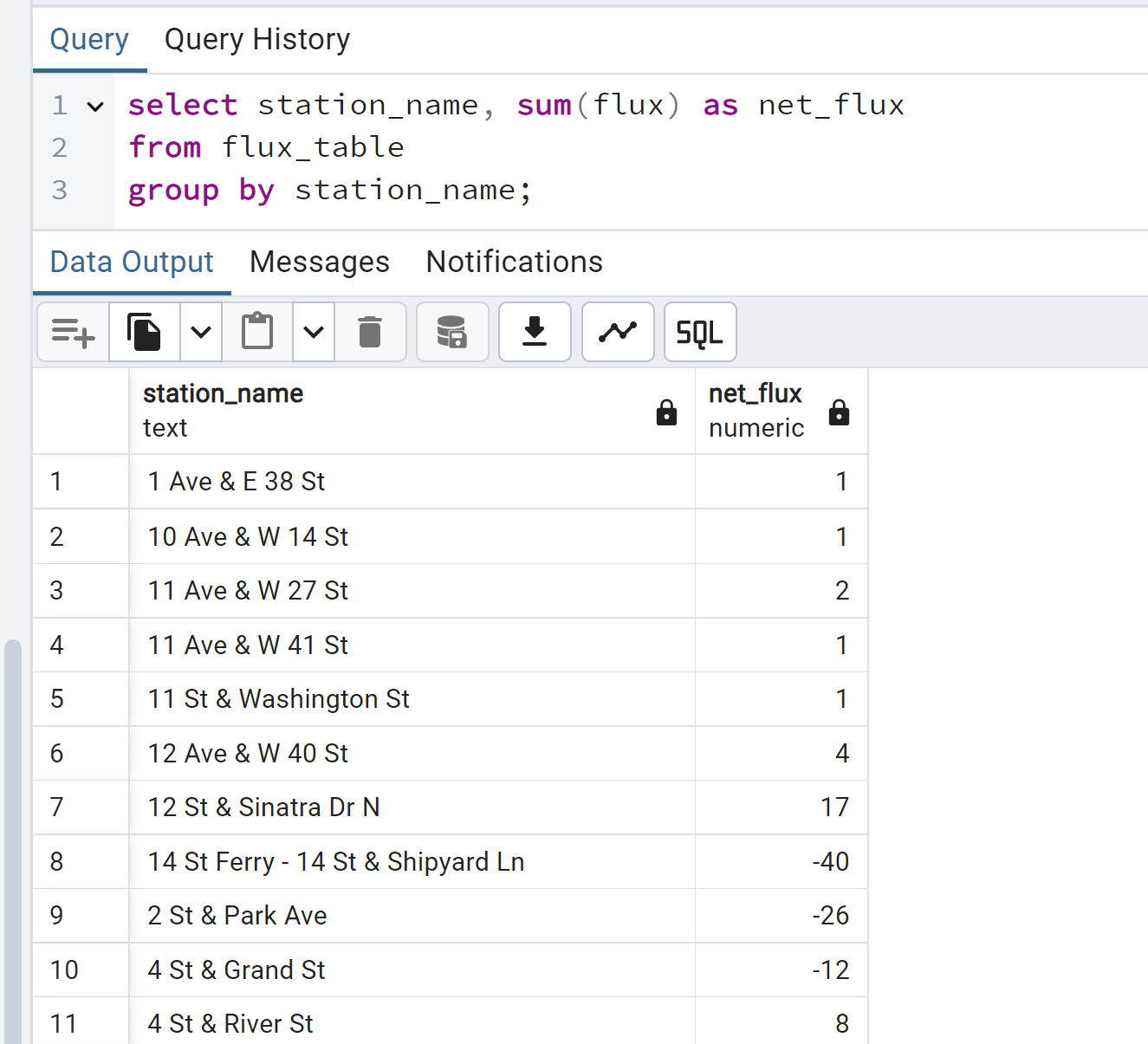
From the graph, 90% of the rides are less than 780 seconds in duration.

Now, for the main problem. We want to find how the stations accumulate or lose bikes over a period of time. In this case, 1 month. Denote this as the flux of a particular station. If the flux is negative, it means a bike was taken from the station, and if it is positive, the meaning is that a bike was left at the station. Below is a query for generating a table of flux for each ride.



Here, we are joining based on station id, which acts as a primary key for the station table, and the foreign key for the ride table.

Now, we need to find the net flux of each station by aggregating the flux of each ride by station as follows.



As we can see, some stations have almost no variance from the initial timestamp of the data, while others, like 2 St & Park Ave, have 26 fewer bikes.

These values are a bit large, because we are assuming no redistribution activity was done over the month, and the system was left to itself. For example, we don’t even know if there were 40 bikes to begin with at 14 St Ferry … The point of this study is to model a potential solution for redistribution, not to be completely realistic. In reality, redistributing the bikes might be done on a weekly, or even daily basis.

Now, for the redistribution, here are our objectives:

1. Redistribute bikes such that the variance at each station does not exceed 10.
2. Optimize for the total number of bikes that need to be transported, minimizing this quantity.

Mathematically, define to be the number of bikes moved from station i to station j, let denote the initial net flux at station i (the constants in the above table). The first condition is equivalent to

Where the sums are over all paris I,j.

Another set of conditions is on the values of the variables themselves. They should be integers, and nonnegative.

Finally, the objective is to minimize .

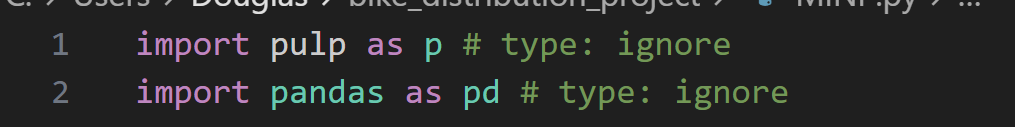
The total number of variables is equal to the square of the number of stations. There are 195 stations, and so this gives a need for 38,025 variables! Remarkably, this problem can by solved in less than 2 seconds on a laptop, using python’s PULP framework for linear programming.

We start by querying all the net flux data and saving this into a csv file.

A screenshot of a computer

AI-generated content may be incorrect.

In python, we start by importing necessary libraries. In this case, pulp and pandas.



Next, we define our problem as a minimization problem and define the variables (lines 8-12), objective (line 15) and import our data (lines 17-18).

A screen shot of a computer code

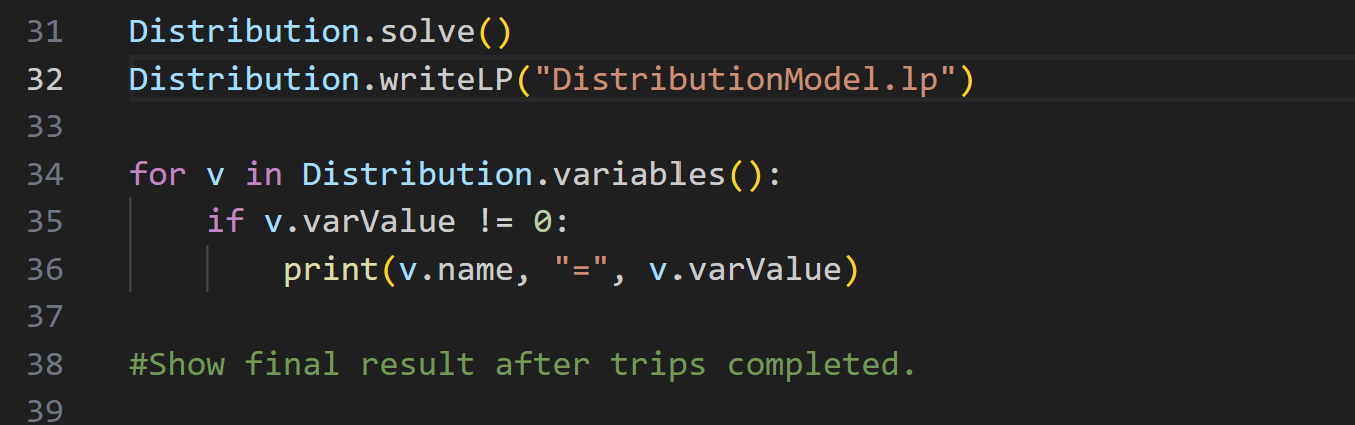
AI-generated content may be incorrect.

Next, define our constraints on final variance.

A screen shot of a computer code

AI-generated content may be incorrect.

The final step is to solve the model and print out the values.



Here is a screenshot of the first few lines of the output.

A screenshot of a computer program

AI-generated content may be incorrect.

For this case, we need to move 921 bikes, and the variables give us a directive for accomplishing this. The final step is to use the solution to tell us about the final variance at each of the ten stations. One way to do this is to transfer the data and results to Excel and recompute flux at each station using the SUMIF function.

A screenshot of a computer

AI-generated content may be incorrect.

The variance at each of the 195 stations is between -10 and 10, as we specified.

Why is this a useful process? One reason is that, counting the data output, we have only 57 non-zero X\_ij. So, to redistribute the bikes among 195 stations, we need to only make 57 trips, assuming we have the truck capacity to transport the number needed for each trip.

Final Remarks and Ideas for Further Work

In a real business setting, one idea for implementation would be to encapsulate this query process into a trigger function that runs automatically every week. Then, running the mathematical program provides a business process for employees to redistribute bicycles as needed to balance the available bikes. If the process is done more frequently, there will be less burden in terms of number of bikes we need to move.

The integer linear program could also be made more sophisticated by perhaps optimizing for fewest number of trips rather than number of bicycles.