Northeastern University

P4: Working with the Database

Boston Blue Bikes Analysis

Project Group 15

Anjali Kabra

Jhalak Surve

Krishna Kapadia

Shubham Shah

Brief description of the data maintenance process:

- Since we have used GBFS live data for our project (as mentioned in P3) and this data keeps on getting updated from time to time, we have created a python script to fetch the latest data from the API and load it into our database.

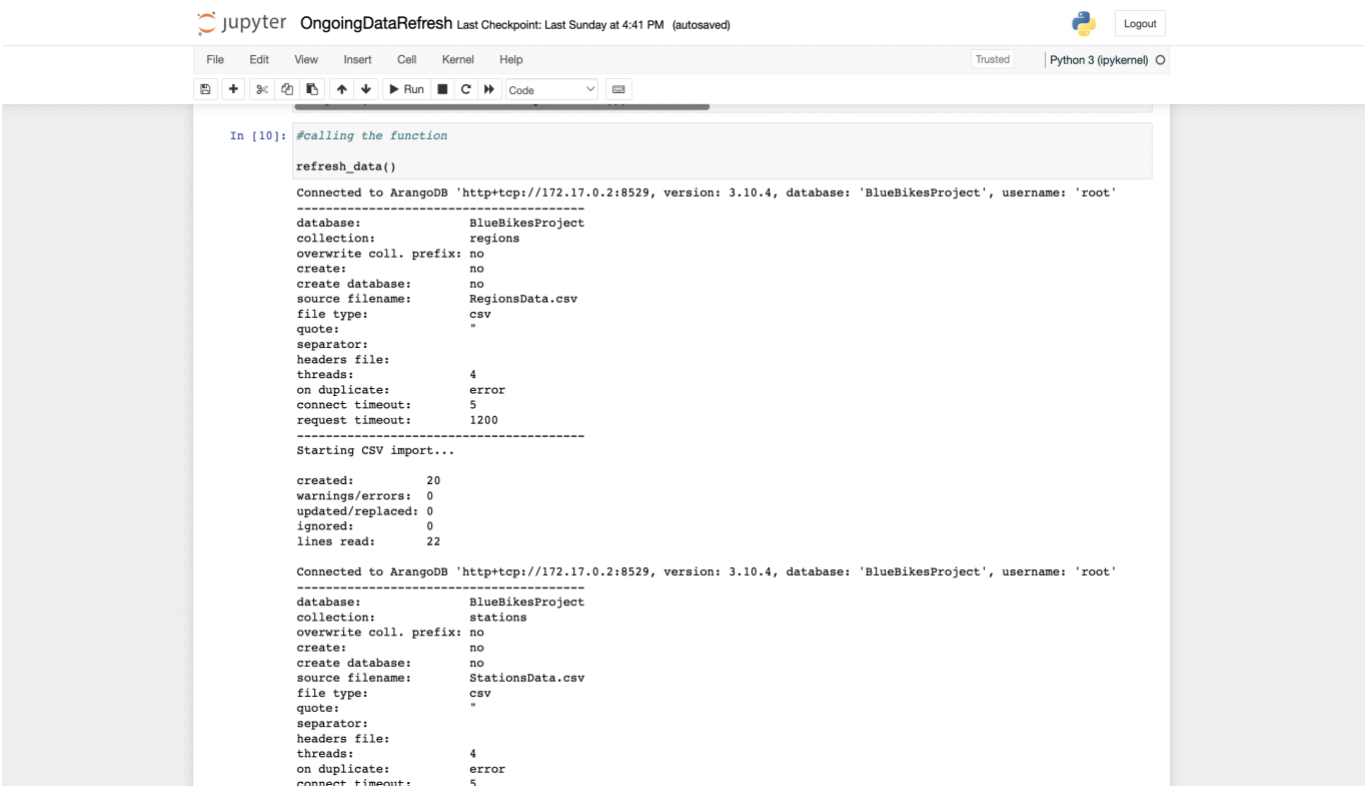
- For this purpose, we have created a function in python **refresh_data()** which fetches the latest data from the GBFS website, cleans it and loads into our arango db collections.

jupyter OngoingDataRefresh Last Checkpoint: Last Sunday at 4:41 PM (autosaved)

File Edit View Insert Cell Kernel Help

```
In [8]: import requests
        import pandas as pd
        from arango import ArangoClient
        import subprocess
        import docker
        import os
        import tarfile

In [9]: def refresh_data():

            client = ArangoClient(hosts='http://localhost:8529')
            db = client.db('BlueBikesProject',username='root', password='root')

            # Drop the collection if it exists
            if db.has_collection('regions'):
                db.delete_collection('regions')

            if db.has_collection('stations'):
                db.delete_collection('stations')

            if db.has_collection('station_status'):
                db.delete_collection('station_status')

            # Defining the collections
            region_col = db.create_collection('regions')
            station_col = db.create_collection('stations')
            status_col = db.create_collection('station_status')

            # Downloading the data from GBFS

            gbfs_url = 'https://gbfs.bluebikes.com/gbfs/gbfs.json'
            gbfs_data = requests.get(gbfs_url).json()

            #fetching the URLs

            region_url = None
            station_url = None
            stationstatus_url = None
            for feed in gbfs_data['data']['en']['feeds']:
                if feed['name'] == 'system_regions':
                    region_url = feed['url']
                elif feed['name'] == 'station_information':
```

- Then, we're calling this function to execute the data refreshment process.

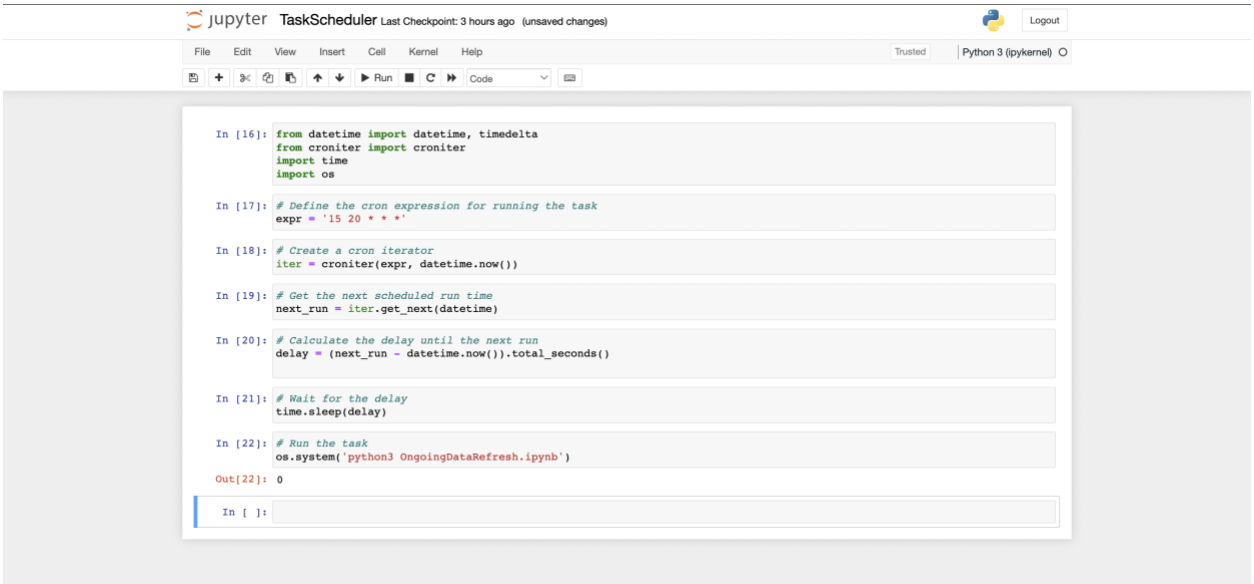


- We have also attached the jupyter notebook file **OngoingDataRefresh.ipynb** with our submission.

Scheduling the task to run the python script:

- Now, to schedule this python script to run daily, we have used **croniter,** which is a python library for working with cron expressions. Cron expressions are used in MacOS for task scheduling.

- We have created another python script TaskScheduler.ipynb, to schedule this script to run daily at 8:15 PM. (Also attached TaskScheduler.ipynb with the submission)

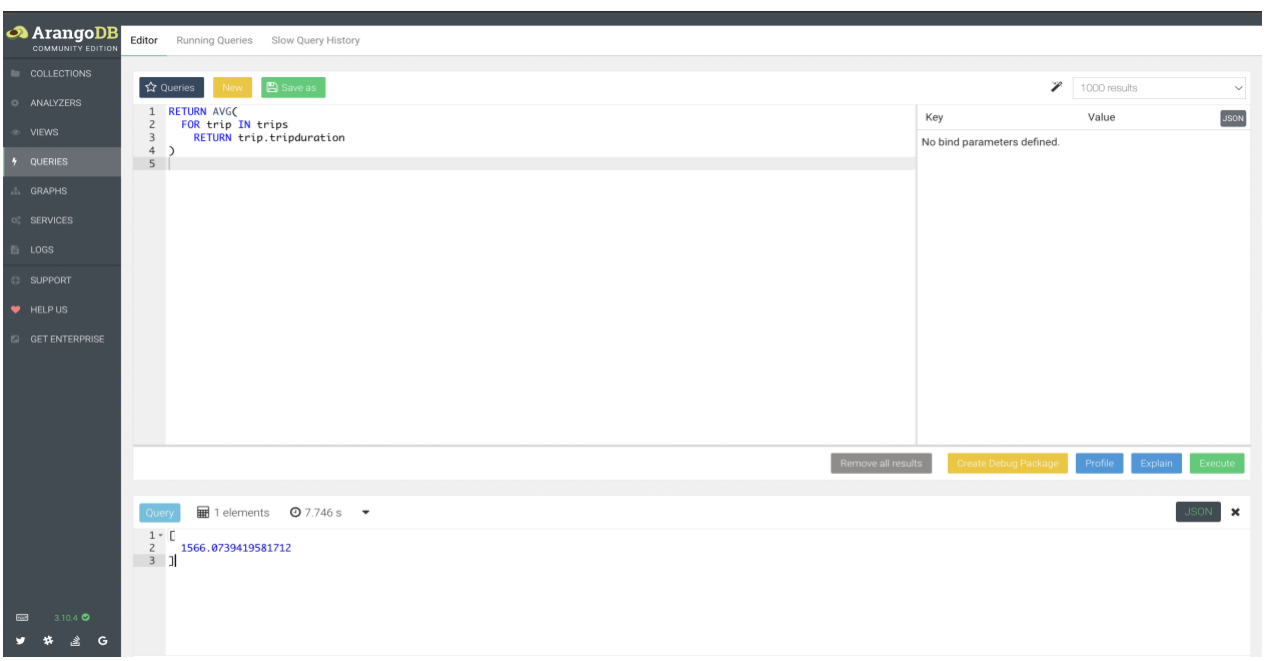


AQL Queries to work with the database

We have also written some AQL queries to work with our database (attached a .sql file with the submission for this):

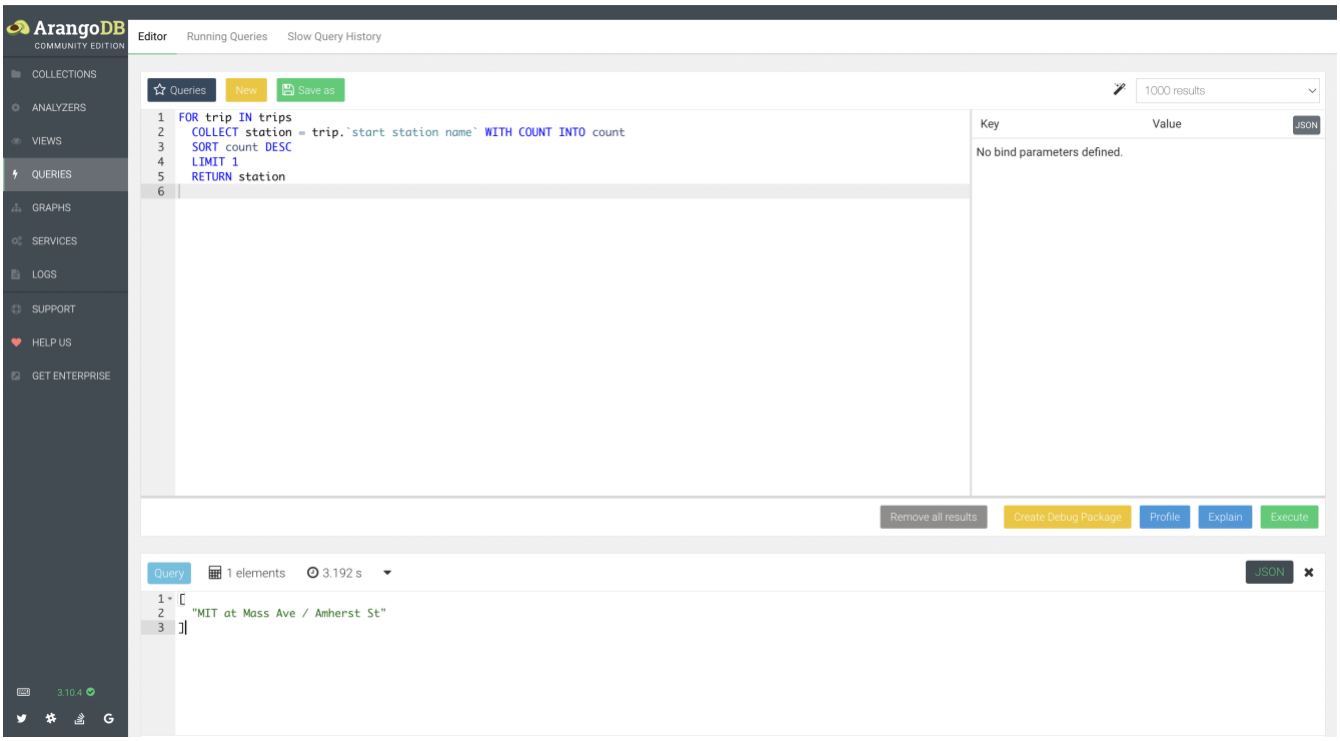
1. AQL query to find out the average duration for which a rider rents a bike

```
RETURN AVG(
  FOR trip IN trips
    RETURN trip.tripduration
)
```



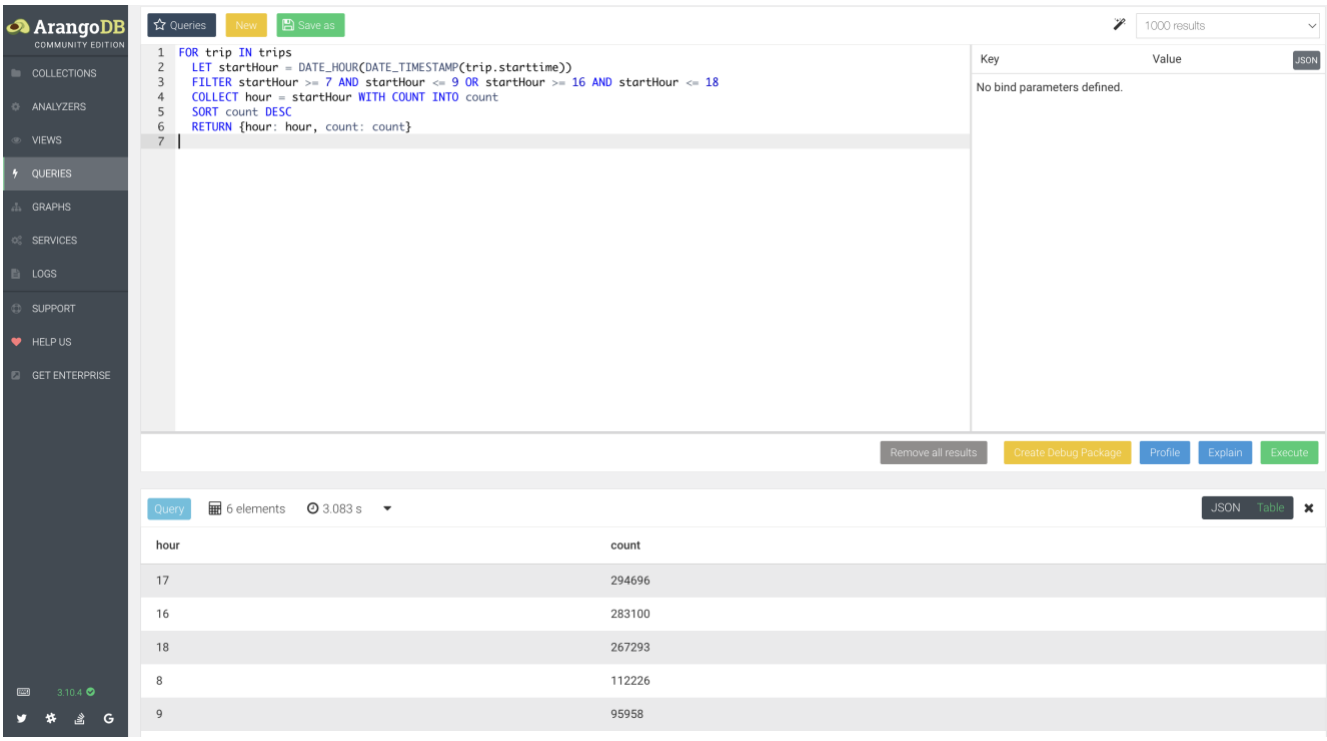
2. AQL query for to find out the most frequently accessed station

```
FOR trip IN trips
  COLLECT station = trip.`start station name` WITH COUNT INTO count
  SORT count DESC
  LIMIT 1
  RETURN station
```



3. AQL query for to find out the rush hours

```
FOR trip IN trips
  LET startHour = DATE_HOUR(DATE_TIMESTAMP(trip.starttime))
  FILTER startHour >= 7 AND startHour <= 9 OR startHour >= 16 AND startHour <= 18
  COLLECT hour = startHour WITH COUNT INTO count
  SORT count DESC
  RETURN {hour: hour, count: count}
```



4. AQL query for Bike usage by user type

```
FOR trip IN trips
    COLLECT userType = trip.usertype WITH COUNT INTO count
    RETURN { userType: userType, tripCount: count }
```