## Overview

In March 2022, Uber announced that they will allow its New York City-based users to hire yellow taxis from its app. While we each all have our own preferences for how we get home on a late night, let's see if we can see some trends to back up why Uber and taxis have clashed for years.

Using hired-ride trip data from Uber and NYC Yellow cab from January 2009 through June 2015, and joining with local historical weather data, you will be providing a single Jupyter notebook and SQL queries to answer particular questions by downloading and cleaning data, storing the data in a SQLite database, and providing visualizations.

## Logistics

### Who

Your teams will be made up of 2 people of your choosing. You will have until end-of-day on April 8th to find group members. If you prefer a surprise, you can wait until the end of the week and you will be assigned to a group by me or a CA.

Groups will be managed via Canvas. Assign yourself to a group by navigating to the "People" section of this course in Canvas and joining a **Project Group**. Do not join or create a student group for this project. *Projects submitted within a "Student Group" as labeled on canvas will not be graded.*

### What

You will be turning in a URL to the repository where your project code can be found. Provide the HTTPS link, not the link to clone the URL.[1] We suggest using GitHub to host your repository, but you may also use BitBucket or GitLab.

See Requirements & Constraints for exactly what is needed and expected.

### When

Group selection is due **Friday, April 8th at 11:59pm ET**; otherwise we'll choose a group for you.

The project is due **Sunday, May 1st at 11:59pm ET**. Like homework assignments, you may turn it in up to 24 hours late with a 50% deduction in awarded points.

[1] The HTTP link looks like `https://github.com/YourUsername/YourProject`. The clone URL looks like `git@github.com/YourUsername/YourProject.git` or `https://github.com/YourUsername/YourProject.git`

# Datasets

- Uber rides: a sample of Uber rides from 01-2009 through 06-2015 can be downloaded from here.
- Historical weather data: the complete catalog of local climatological data for 2009 through 2015, plus the dataset's documentation, can be found in this folder.
- Yellow Taxi trip data: use this URL to download the CSV files needed for the project, as instructed in Part 1 below.
    - It may be helpful to read the "Trip Record User Guide" and "Yellow Trips Data Dictionary" under the "Data Dictionaries and MetaData" section towards the bottom.
    - We will *not* be using data for Green Taxi trips or FHV trips.
    - Each CSV file may take a few minutes or more to download, depending on your internet speed.

# Specifications

Create a Jupyter Notebook that is divided into four parts: Data Preprocessing, Storing Data, Understanding Data, and Visualizing Data (described further below). For each part, use Markdown cells to describe what is going on. Construct and write your notebook as if you could hand it to a (potentially Python-inclined) friend, and they could understand it completely without any questions.

Of course, you're able to experiment, understand, and play around with the data and your code's implementation. Be as messy as you need. Create multiple notebooks if you want. The final result should be clean, readable, and tell a logical story in a single notebook.

Write separate functions or specific tasks that help accomplish the goals of each part. This will make it easier to test different parts of the logic independently. Then in separate cells, execute the functions (so, don't just define functions; call them as well). I should be able to clear the output, then run all cells successfully to generate everything needed below.

## Part 1: Data Preprocessing

This project deals with a lot of data that can be too much for a personal computer to handle effectively. For instance, each month of yellow taxi data is about *2GB in size*. Therefore, it's imperative that you remove any unnecessary and invalid data, use appropriate data types when possible, and take samples.

For this part, you ~~will need~~ are required (update: 2022-04-21) to use the requests, BeautifulSoup, and pandas packages to help you programmatically download and clean every Yellow Taxi CSV file needed. Cleaning the data includes: removing unnecessary columns and invalid data points, normalizing column names, and removing trips that start and/or end outside of the following latitude/longitude coordinate box: (40.560445, -74.242330) and (40.908524, -73.717047).

While you will not need to programmatically download the Uber data, you will need to load it in from your computer, and clean the dataset as you did with the Yellow Taxi datasets.

Each month of Yellow Taxi data contains millions of trips. However, the provided Uber dataset is only a sampling of all data. Therefore, you will need to generate a sampling of Yellow Taxi data that's roughly equal to the sample size of the Uber dataset.

Also within this part, define a function that calculates the distance between two coordinates in kilometers that only uses the `math` module from the standard library. Write at least one unit test that tests this calculation function.

Using that function that calculates the distance in kilometers between two coordinates, add a column to each dataset that contains the distance between the pickup and dropoff location.

Finally, load in the weather datasets from your computer, and clean each dataset, including only the dates & columns needed to answer the questions in the other parts of the project.

Tips:

- Downloading Yellow Taxi data can take a while per file since each file is so large. Consider saving the sample data for each month to your computer in case you need to step away, and load it back in when you return.
- Relatedly, make use of your `.gitignore` file to avoid committing the Yellow Taxi sample dataset CSV files to your repo.
- Read ahead to figure out which columns are absolutely necessary for each dataset.
- Be mindful of the data types for each column, which will make it easier for yourself when storing and filtering data later on.
- Use the `re` module to help pull out the desired links for Yellow Taxi CSV files.
- *[added 2022-04-18]* You may find that the weather data you need later on does not exist at the frequency needed (daily vs hourly). You may calculate/generate samples from one to populate the other. Just document what you're doing so we can follow along.

## Part 2: Storing Data

Using SQLAlchemy, create a SQLite database with which you'll load in your preprocessed datasets.

Create and populate four tables: one for your sampled datasets of Yellow Taxi trips, one for Uber trips, one for hourly weather information, and one for daily weather information. Use appropriate data types for each column.

Create a `schema.sql` file that defines each table's schema. You can use SQLAlchemy within the notebook to help generate this file, (added 2022-04-21) or another programmatic approach, or create this schema file by hand.

Tips (added 2022-04-21):

- The first 48 lines of this gist is a good example of what makes up a schema file.
- I should be able to run this schema file to create the tables in a database via the SQLite CLI tool. That is, I should be able to run the following command in a Jupyter notebook cell to create a database with the four required tables (it is not expected that you do this yourself for the project, but this is a good sanity check for it to succeed without error):

```
!sqlite3 project.db < schema.sql
```

## Part 3: Understanding Data

For this part, define a SQL query for each of the following questions - one query per question. Save each query as a `.sql` file, naming it something illustrative of what the query is for, e.g. `top_10_hottest_days.sql`.

1. For 01-2009 through 06-2015, what hour of the day was the most popular to take a Yellow Taxi? The result should have 24 bins.
2. For the same time frame, what day of the week was the most popular to take an Uber? The result should have 7 bins.
3. What is the 95% percentile of distance traveled for *all* hired trips during July 2013?
4. What were the top 10 days with the highest number of hired rides for 2009, and what was the average distance for each day?
5. Which 10 days in 2014 were the windiest on average, and how many hired trips were made on those days?
6. During Hurricane Sandy in NYC (Oct 29-30, 2012), plus the week leading up and the week after, how many trips were taken each hour, and for each hour, how much precipitation did NYC receive and what was the sustained wind speed? There should be an entry for every single hour, even if no rides were taken, no precipitation was measured, or there was no wind.

For each query, be sure to execute it in the notebook so we can see your answers to the question.

Tips:

- You may wish to use SQLAlchemy within the notebook to help craft these queries and query files. You can also use pandas to help check the validity of your queries.
- You may want to familiarize yourself with COALESCE, WITH, and WITH RECURSIVE expressions for help in answering some of the questions.
- See appendix of lecture notes from module #10 for more tips/hints

## Part 4: Visualizing Data

For this final part, you will be creating a bunch of visualizations embedded in your notebook using matplotlib and/or other visualization libraries of your choice. Be sure to define a function for each visualization, then call these functions in separate cells to render each visualization.

This is where you can get creative with the look and feel of each visual. All that is required is that each visualization is immediately understandable without necessarily needing to read its associated function (i.e. labeled axes, titles, appropriate plot/graph/visual type, etc). You're welcome to use Markdown cells to introduce and/or explain each visualization.

You can use pandas to help parse data before generating a visualization, but you must read the data from your SQLite database (so, do not read directly from CSV files).

1. Create an appropriate visualization for the first query/question in part 3.
2. Create a visualization that shows the average distance traveled per month (regardless of year - so group by each month) for both taxis and Ubers combined. Include the 90% confidence interval around the mean in the visualization.
3. Define three lat/long coordinate boxes around the three major New York airports: LGA, JFK, and EWR (you can use bboxfinder to help). Create a visualization that compares what day of the week was most popular for drop offs for each airport.
4. Create a heatmap of all hired trips over a map of the area. Consider using KeplerGL or another library that helps generate geospatial visualizations.
5. Create a scatter plot that compares tip amount versus distance for Yellow Taxi rides. You may remove any outliers how you see fit.
6. Create another scatter plot that compares tip amount versus precipitation amount for Yellow Taxi rides. You may remove any outliers how you see fit.
7. Come up with 3 questions on your own that can be answered based on the data in the 4 tables. Create at least one visualization to answer each question. At least one visualization should require data from at least 3 tables.

# Requirements & Constraints

Your project must use Python 3.7 or higher. The project must be under Git version control. Your notebook must be able to execute completely and successfully in order. Any and all functions, classes, and class methods must have docstrings.

Your project will need to include the following:

- Source code
  - One (and only one) Jupyter Notebook. Leave your notebook executed (you don't need to clear the output). GitHub will therefore automatically render all generated visualizations for you.
  - Required SQL files (schema & queries).
- `README.md` file

    - A file describing the project, its use, and its behavior. See below for more detail on what's expected.
- `requirements.txt` file
    - This should include any Python package that you've had to `conda install` or `pip install`.
    - Be mindful that the Anaconda installation provides additional 3rd party libraries so you don't have to install them yourself. Your `requirements.txt` file must include anything that Anaconda already makes available.
- `.gitignore` file
    - A file to ignore files that should not be committed (e.g. pycache files, csv files, SQLite files)

Do *not* include:
- Any Yellow Taxi CSV files - there's no need to include data that will be downloaded by your code.
- Any database/SQLite files, other than your schema file and query files.
- Any images of your visualizations - they should be automatically embedded in your notebook when the appropriate cells are executed.
- You can certainly generate the above files as you are working, just do not include them in your repository
    - Tip: make use of the `.gitignore` file.

## Grading

The project will be out of 20 points. Everyone in the group will share the same grade.
- 10% - Clarity of code (including PEP8 + PEP257 style)
- 70% - Meets the above criteria (Specifications, Requirements & Constraints)
- 20% - Thorough use of Git
    - Your team has used Git to save incremental progress in your work. Each member has roughly the same number of commits, additions, and subtractions.

## Extra Credit

You have the opportunity to earn up to 10% in extra credit, for a maximum total of 22 points on the project. You may complete all extra credit opportunities, or pick and choose which you want to complete.
- 2.5% (0.5 points): Add two or more animations and/or Jupyter widgets for any of your visualizations.
- 2.5% (0.5 points): Define at least one unit test for every function and class method you've defined.
- 2.5% (0.5 points): Add type hints to all functions & class methods you've defined.
- 2.5% (0.5 points): Add a 4th table that contains daily sunset/sunrise data (which can be found in the original weather data). Write a question that considers this new table in

relation to one or more existing tables, and then create either a SQL query or a visualization that answers that question.

# Appendix

## What's a README?

A README file is a that sits at the root of the project, repo, or directory of an application and describes what the application does and how to use it. It is often in Markdown (`README.md`), reStructuredText (`README.rst`), or simple text (`README.txt`). Here are some good examples of READMEs.

- https://github.com/python/cpython/blob/master/README.rst
- https://github.com/pandas-dev/pandas/blob/master/README.md
- https://github.com/tensorflow/tensorflow/blob/master/README.md

Your README needs to include:

- A description of what has been implemented
  - This only needs to be a paragraph or less of a description. More is welcome but not needed.
- Your group name
  - Eg. Project Group 51
- A list containing the UNI for each member on the team
  - Of the form: "UNIs: [uni1, uni2, uni3]". Eg. "UNIs: [ab1234, cd7847, ef9873]"