Introduction to Database Systems

Final Project Proposal

10811020 陳光悅

10801128 陳俊鴻

0816091 荊姿芸

1. Main idea

○ The purpose of your application

Traffic is a daily issue in our life, especially for New Yorker!!! There's even no time people not stuck in New York City. Based on the desperate situation, here comes a new Marvel hero, the Taxi Prophet, who always know where and when New Yorkers can get a ride on taxi in NYC. Thanks to the charity of Taxi Prophet, people know some interesting truth about taxi in NYC but nothing change their life. However, we still cannot ignore the Taxi Prophet's effort and stop worshipping his/her (whatever pronoun it wants, coz it's too messy in this era and we still want to pretend our political correction) charm.

2. Data

- Data Source:

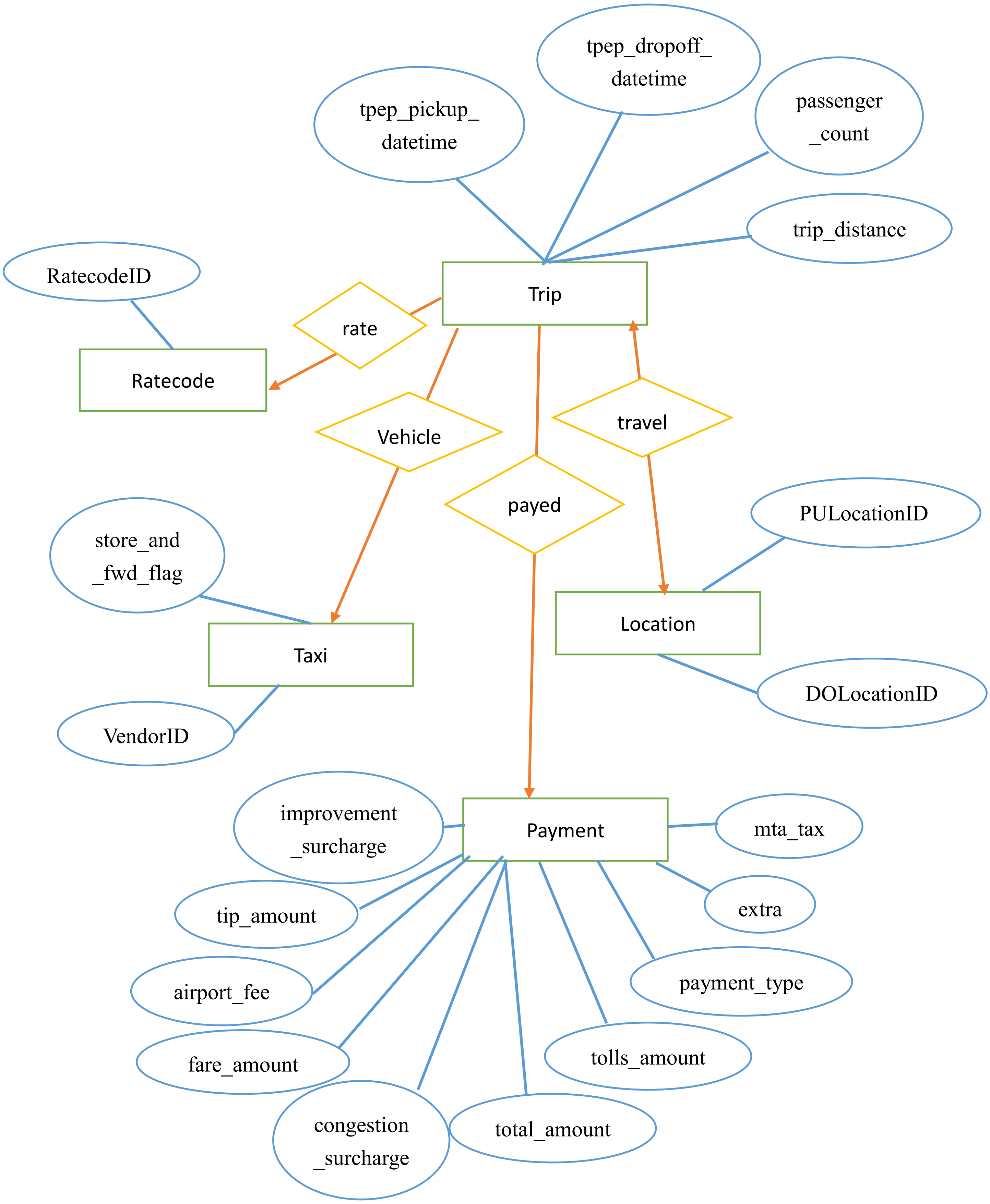
  NYC Taxi & Limousine Commission:
  https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page

- Data Description:

  VendorID:
  A code indicating the TPEP provider that provided the record.
  tpep_pickup_datetime:
  The date and time when the meter was engaged.
  tpep_dropoff_datetime:
  The date and time when the meter was disengaged.
  Passenger_count:
  The number of passengers in the vehicle. This is a driver-entered value.
  Trip_distance:
  The elapsed trip distance in miles reported by the taximeter.
  PULocationID:
  TLC Taxi Zone in which the taximeter was engaged
  DOLocationID:

TLC Taxi Zone in which the taximeter was disengaged
RateCodeID:
The final rate code in effect at the end of the trip.
1= Standard rate 2=JFK 3=Newark 4=Nassau or Westchester 5=Negotiated fare 6=Group ride
Store_and_fwd_flag:
This flag indicates whether the trip record was held in vehicle memory before sending to the vendor, aka "store and forward," because the vehicle did not have a connection to the server.
Y= store and forward trip N= not a store and forward trip
Payment_type:
A numeric code signifying how the passenger paid for the trip.
1= Credit card 2= Cash 3= No charge 4= Dispute 5= Unknown 6= Voided trip
Fare_amount:
The time-and-distance fare calculated by the meter.
Extra:
Miscellaneous extras and surcharges. Currently, this only includes the $0.50 and $1 rush hour and overnight charges.
MTA_tax:
$0.50 MTA tax that is automatically triggered based on the metered rate in use.
Improvement_surcharge:
$0.30 improvement surcharge assessed trips at the flag drop.
Tip_amount:
Tip amount – This field is automatically populated for credit card tips. Cash tips are not included.
Tolls_amount:
Total amount of all tolls paid in trip.
Total_amount:
The total amount charged to passengers. Does not include cash tips.
Congestion_Surcharge:
Total amount collected in trip for NYS congestion surcharge.
Airport_fee:
$1.25 for pick up only at LaGuardia and John F. Kennedy Airports

- ER model


tpep_dropoff_ datetime
tpep_pickup_ datetime
passenger _count
trip_distance
RatecodeID
Trip
rate
Ratecode
Vehicle
travel
payed
store_and _fwd_flag
PULocationID
Location
Taxi
DOLocationID
VendorID
improvement _surcharge
Payment
mta_tax
tip_amount
extra
airport_fee
payment_type
fare_amount
tolls_amount
congestion _surcharge
total_amount

3. Database

  ○ What database do you use

    Pyspark.sql

  ○ How do you maintain your database (update data, add new data etc.)

    The databases we use are collections of real-world data, collected and provided to the NYC Taxi and Limousine Commission (TLC) by technology providers authorized under the Taxicab & Livery Passenger Enhancement Programs (TPEP/LPEP).

    Except some driver-entered value like passenger_count, the trip log data were automatically uploaded from devices such as taximeter, meter on the taxi. Each trip data will be recorded in one row and the dataset published monthly with two months delay.

    If you want to add new data locally, convert the data to a pyspark dataframe first and make sure the order of the values corresponds to the columns of the data. Then **union** the new data to the dataset and the data will be added at the bottom of the dataset.

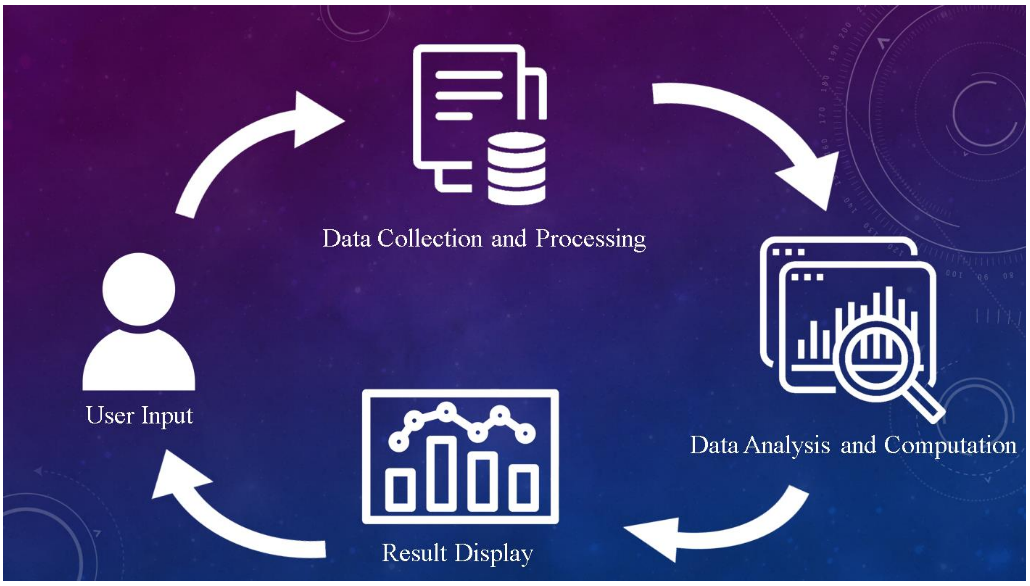
    Use **withColumn**("col_name", col_value), to add a new column to the dataset.

    Use **drop**("col_name") to delete a column from the dataset.

    Instead of delete a specific row, **filter**(condition) is a better way to get rid of useless rows. Filter works similar as select … where… in MYSQL.

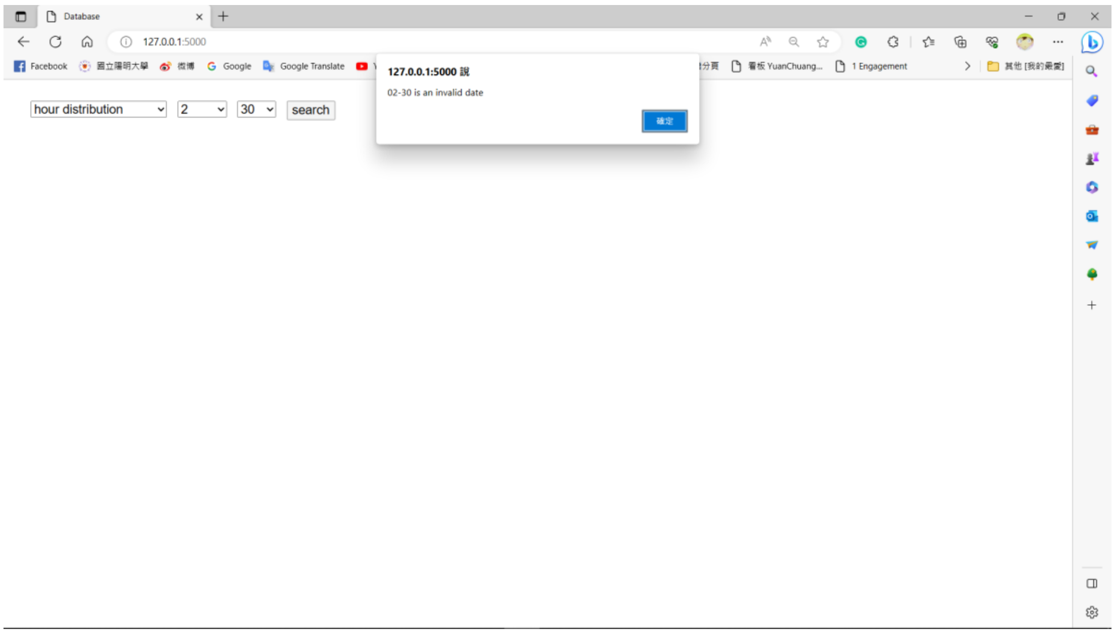
  ○ Describe how you connect your database to your application as detail as possible

Our application is an interactive website; hence, the query would be generated as a token in the frontend codes, and we then deliver the token to the database for some specific functions. We designed a semaphore to make the frontend codes wait for the return of the functions. After processing these functions, the frontend codes capture the path to desired pictures and demonstrate them to the users. Due to the plot of pictures, that would race with our GUI codes as a sub-thread, we merged it to the main thread to prevent the crisis. The query results were passed back to the main code to fulfill the workflow. Additionally, to save more time, the database would load all the data once at the beginning of the activation of whole codes.



exception handling:

Return error on application if the date selected by user is invalid.



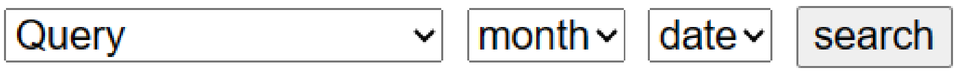
127.0.0.1:5000 說

02-30 is an invalid date

4. Application

○ Describe the interface of your application

The interface we design has 3 selectors and 1 button.



- Query selector：User can use query selector to select the function they want to access. There are 5 functions in the selector: Hour distribution, Average of hour distribution, Average of month distribution, Pickup map, Dropoff map.
- Month/Date selector：For Hour distribution, Pickup map, and Dropoff

map, user can use month/date selector to select a specific month or date to access the data of the date selected.

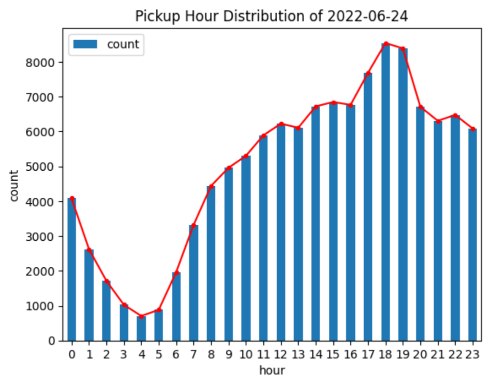
- ◆ search button：Press search button to return the result of the function chosen. Once the button is pressed, the interface will call backend code and generate the result figure.

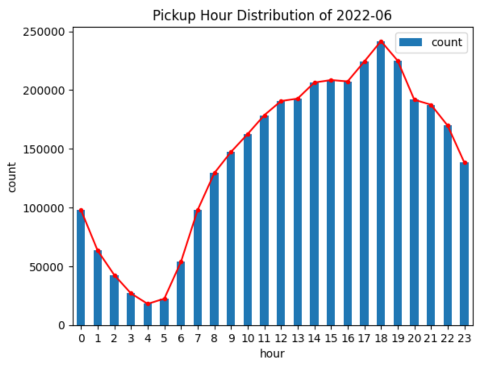
The result will display in png file so that the user can easily check the analysis results. The png file is either bar chart or heat map.

In order to improve data security and prevent unauthorized copying, we designed the interface to disable the right-clicks in the application. This prevents users from selecting options such as "Copy" or "Save Image" through the right-click menu, thereby preventing them from copying the results directly from the application. Avoids unauthorized use and data leakage, ensuring that the png file is only available in the application.

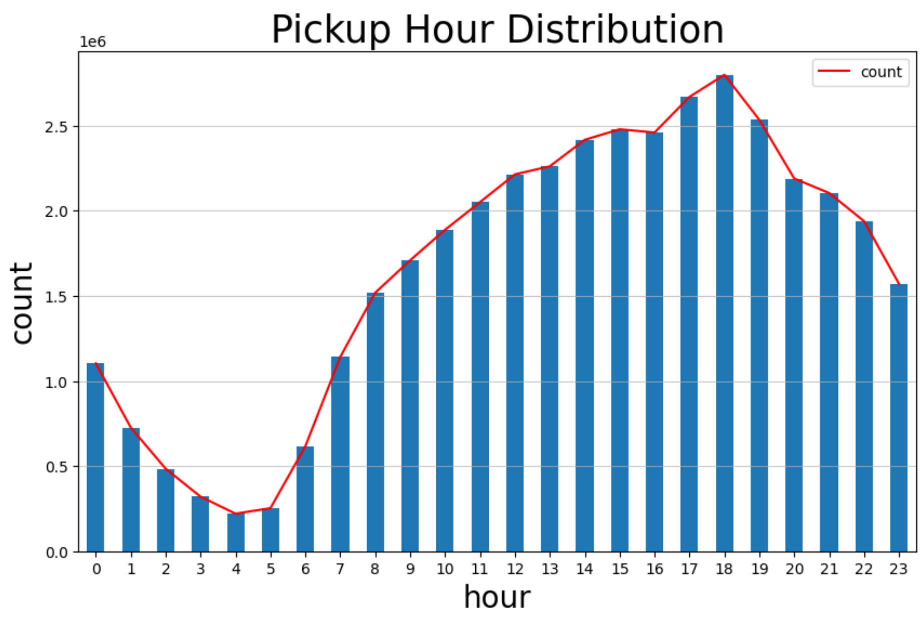
○Functions of application

- ◆ Hour distribution:
  Displays the bar chart of pickup hour distribution of a specific date or month, determined by the user input selection.
  x-axis is the 24 hours of a day and y-axis is the count of the pickups within certain hour.

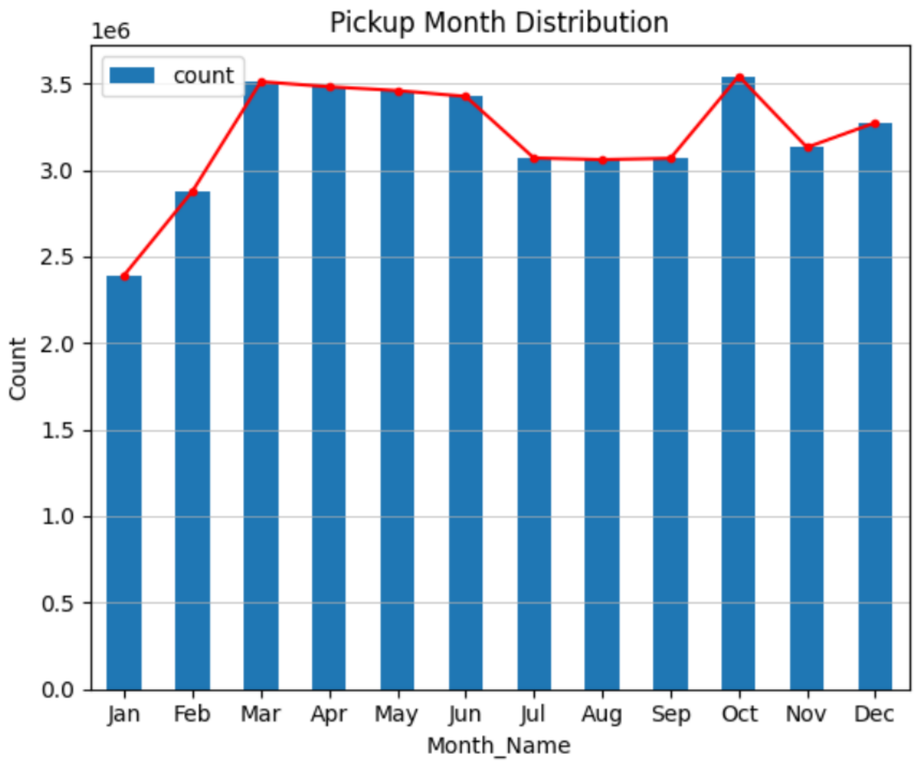




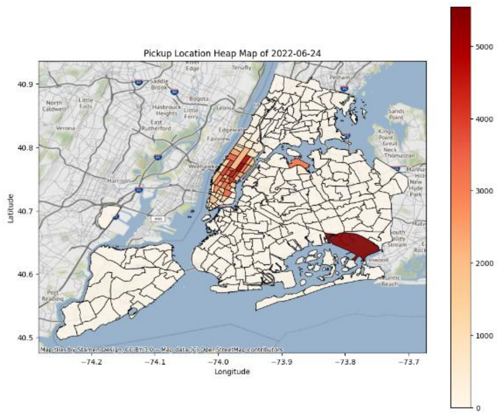
- ◆ Average of hour distribution:
  Displays the bar chart of hour counts of each hour in 2022 whole year.
  x-axis is the 24 hours of a day and y-axis is the taxi counts.

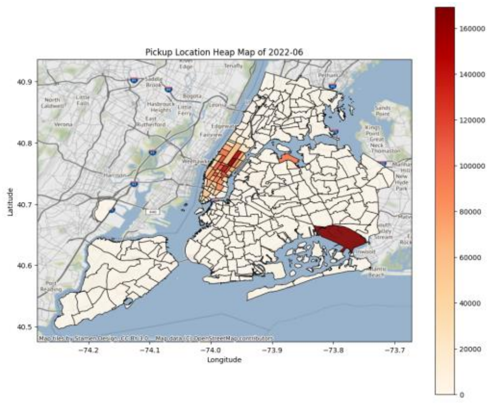


- Average of month distribution:
  Displays the bar chart of taxi counts of each month in 2022 whole year. x-axis is the 12 months of a year and y-axis is the taxi counts.

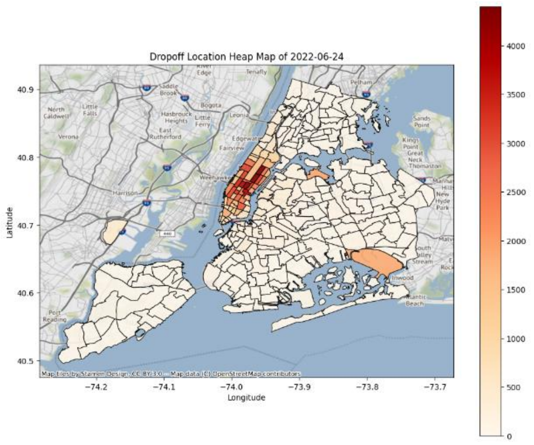


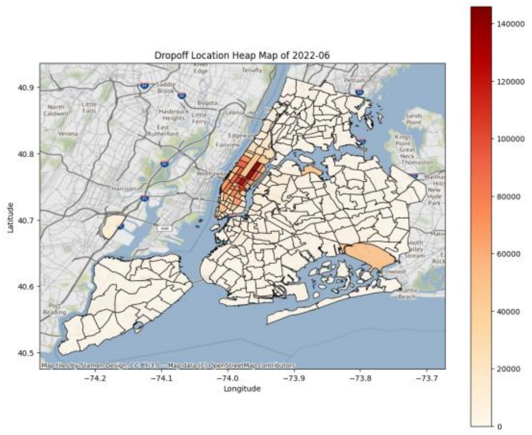
- Pickup map:
  Displays the heat map of pickup counts of a specific date or month in each region in New York City.
  Date is determined by the user input selection and region is determined by taxi_zones_shapefile associate with a taxi_zone lookup table.





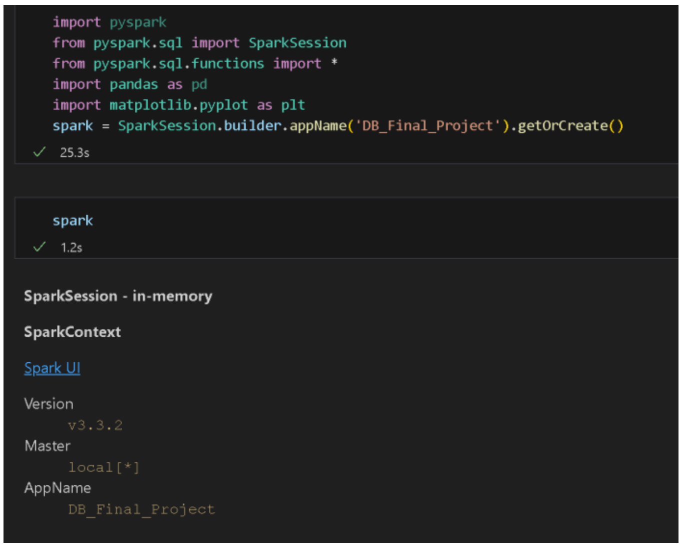
- Dropoff map:
  Displays the heat map of dropoff counts of a specific date or month in each region in New York City.
  Date is determined by the user input selection and region is determined by taxi_zones_shapefile associate with a taxi_zone lookup table.





○ For each function of your application describe how you make it possible in detail

- Fundamental:
  Create a spark session for pyspark



```
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
import pandas as pd
import matplotlib.pyplot as plt
spark = SparkSession.builder.appName('DB_Final_Project').getOrCreate()
```

✓ 25.3s

```
spark
```

✓ 1.2s

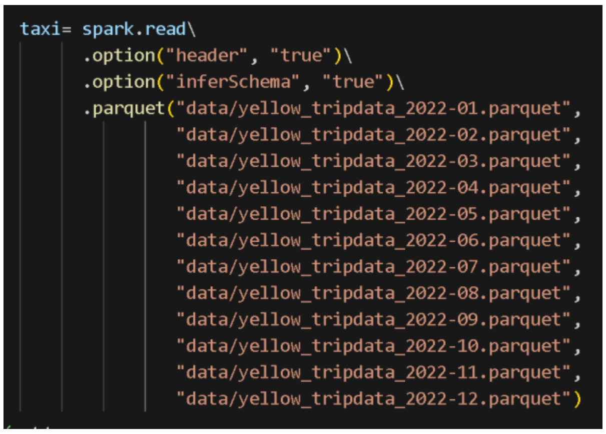
```
SparkSession - in-memory

SparkContext

Spark UI

Version
    v3.3.2
Master
    local[*]
AppName
    DB_Final_Project
```

Read in the datasets



```
taxi= spark.read\
        .option("header", "true")\
        .option("inferSchema", "true")\
        .parquet("data/yellow_tripdata_2022-01.parquet",
                 "data/yellow_tripdata_2022-02.parquet",
                 "data/yellow_tripdata_2022-03.parquet",
                 "data/yellow_tripdata_2022-04.parquet",
                 "data/yellow_tripdata_2022-05.parquet",
                 "data/yellow_tripdata_2022-06.parquet",
                 "data/yellow_tripdata_2022-07.parquet",
                 "data/yellow_tripdata_2022-08.parquet",
                 "data/yellow_tripdata_2022-09.parquet",
                 "data/yellow_tripdata_2022-10.parquet",
                 "data/yellow_tripdata_2022-11.parquet",
                 "data/yellow_tripdata_2022-12.parquet")
```

Remove (1)null value rows, (2)outliers, and (3)duplicate rows

```
taxi=taxi.na.drop(how="any")
taxi=taxi.filter(year(col("tpep_pickup_datetime"))=="2022")
taxi=taxi.distinct()
```

For hour distribution and pickup/dropoff map, date is the date selected from the application, returning in the format 2022-MM-DD or 2022-MM for specific date or month respectively.

The data selection for hour distribution and pickup/dropoff map work as follow.

Select the rows which starts with the date selected.

```
pickup_by_day=taxi.filter(col("tpep_pickup_datetime").startswith(date))
```

Pyspark command related to MYSQL command:

filter->select * where condition

select->select columns

alias->name as …

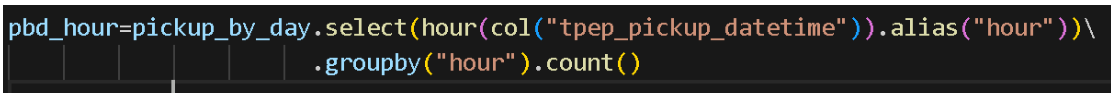
groupby-> group by

count->count

merge->join

- Hour distribution

  From the rows selected, we only need the hour value of tpep_pickup_datetime. Project the hour column and group by hour to get the pickup counts of each hour.

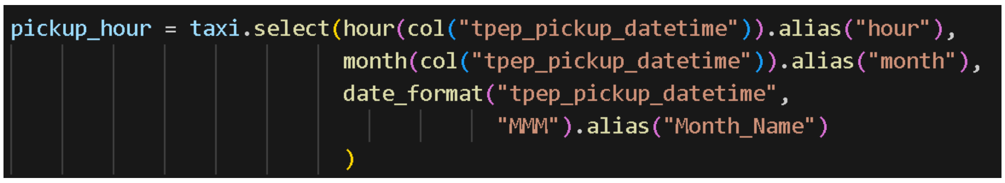


```
pbd_hour=pickup_by_day.select(hour(col("tpep_pickup_datetime")).alias("hour"))\
                                .groupby("hour").count()
```

Pbd_hour is now contains 2 columns: hour and counts (pickup_counts group by hour). Then the result can be plotted using matplotlib.

- Average of hour/month distribution

  Similar to Hour distribution, average of hour/month distribution is the hour/month distribution of the whole year instead of a specific date. Therefore filter is not necessary. Just select the hour/month value from tpep_pickup_datetime.



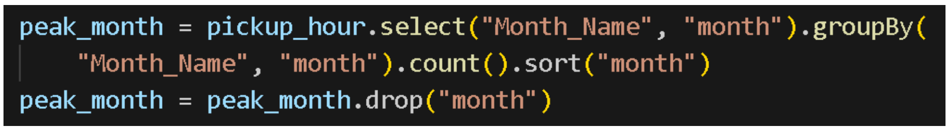
```
pickup_hour = taxi.select(hour(col("tpep_pickup_datetime")).alias("hour"),
                          month(col("tpep_pickup_datetime")).alias("month"),
                          date_format("tpep_pickup_datetime",
                                      "MMM").alias("Month_Name")
                          )
```

Average of hour distribution:

Same as Hour distribution above, project the hour column and group by hour to get the pickup counts of each hour. Then the result can be plotted using matplotlib.

Average of month distribution:

Similar to Average of hour distribution, but project month and month_name instead of hour column. Group by month and month_name to get the pickup counts of each month. Then the result can be plotted using matplotlib.



```
peak_month = pickup_hour.select("Month_Name", "month").groupBy(
    "Month_Name", "month").count().sort("month")
peak_month = peak_month.drop("month")
```

- Pickup map/Dropoff map

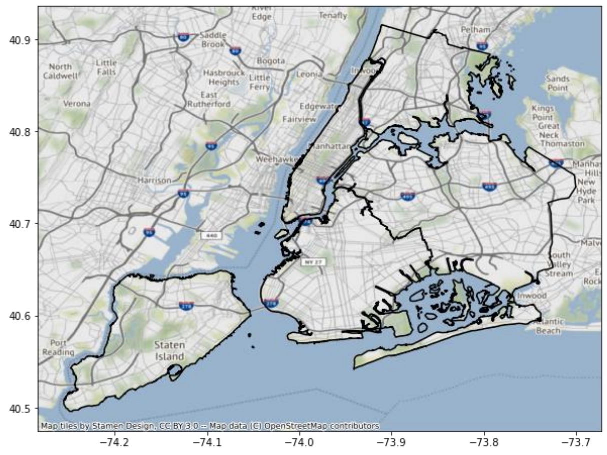
  To display a heat map of NYC, the map data must be loaded before. There are 2 part of map data: taxi_zone to distinguish the pickup/dropoff regions and the base map of NYC.

  taxi_zone: NYC is separated into a total of 264 regions. The details can be checked in taxi zone lookup table.

```python
#read location map data
import geopandas as gpd
shapefile_path = "yellow_tripdata_2022/taxi_zones.zip"
gdf = gpd.read_file(shapefile_path)
```

NYC basemap:

```python
cx.add_basemap(ax,crs=gdf_dolocation.crs)
```


40.9
40.8
40.7
40.6
40.5
−74.2
−74.1
−74.0
−73.9
−73.8
−73.7
Map tiles by Stamen Design, CC BY 3.0 -- Map data (C) OpenStreetMap contributors
Staten Island
Manhattan

From the rows selected, project the PULocationID/DOLocationID and name as LocationID then group by Location ID and get the pickup/dropoff counts of each taxi_zone.

```python
pbd_location=pickup_by_day.select(col("PULocationID").alias("LocationID"))\
                                  .groupBy("LocationID").count()
```

Since gdf is a pandas dataframe, change pbd_location from pyspark dataframe to pandas dataframe.

```python
pbd_location_pd=pbd_location.toPandas()
```

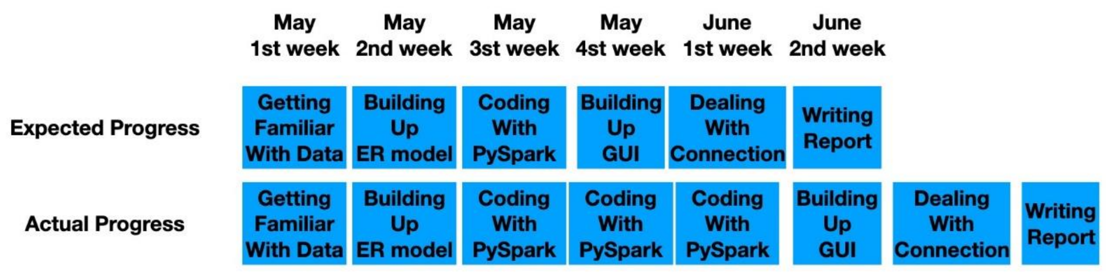
Left join gdf(taxi zone map data) with pbd_(do)location_pd(location counts)

If the region has none location counts, fill in with 0 instead of NA.

```python
#left join gdf and pbd_location_pd
gdf_location = gdf.merge(pbd_location_pd,
                         on='LocationID',
                         suffixes=('', '_count'),
                         how="left")
gdf_location['count'] = gdf_location['count'].fillna(0)
```

Then the result(heat map) can be plotted.

5. Others
   - ○ Draw the progress of your project



	May 1st week	May 2nd week	May 3st week	May 4st week	June 1st week	June 2nd week		
Expected Progress	Getting Familiar With Data	Building Up ER model	Coding With PySpark	Building Up GUI	Dealing With Connection	Writing Report		
Actual Progress	Getting Familiar With Data	Building Up ER model	Coding With PySpark	Coding With PySpark	Coding With PySpark	Building Up GUI	Dealing With Connection	Writing Report

○ What were the problems you met in the project, how did you solve them?

We wrote the Python program with two threads. One displays a GUI and the other gets input from a scanner, process data in an online database and return a matplotlib plot. Then we add "matplotlib.pyplot.switch_backend('Agg')", so that our server does not try to create (and then destroy) GUI windows that will never be seen.

○ List the contribution of each team member in the project clearly

荊姿芸	backend、report
陳光悅	Connect front/backend、report、demo
陳俊鴻	frontend、PPT、presentation

○ Provide a link to your repository

https://github.com/ginnyching/2023DB_Final_Project

○ Provide a link to your discussion channel

We use LINE to discuss, the chat history will be updated on github.