NYC Green Taxi data - Challenge

Part 1) Use the publicly available BigQuery dataset named `nyc-tlc.green.trips_2015`, provide SQL queries to answer the following questions:

Tasks

1. What is the total amount and passenger counts for the months of February, March and April?
2. What has been the average hourly passenger count throughout the year?
3. What has been the change/delta in total amount billed over days? What we would like see is how much (positive or negative) difference we have seen, day over day, in terms of `total_amount`.
4. What hour of the day has seen the longest rides in April?

Part 2) Let's say you needed to connect to an API and pull down data that was into a CSV file and into a database. Assume it's a large amount of data, 500MB a day. Please share a few lines of Java or Python code that shows how you would connect and describe, in your ideal world, where the code would be physically running, where the downloaded data would be saved and which database you would want to store it into.

Solutions

PART 1

1. What is the total amount and passenger counts for the months of February, March and April?

By individual month for February, March, April
Query

```
SELECT extract(month from pickup_datetime) as month,
sum(passenger_count) as total_passengers,
sum(total_amount) as total_amount
FROM `nyc-tlc.green.trips_2015`
GROUP BY month
having month between 2 and 4
order by month;
```

Result Preview

Row	month	total_passengers	total_amount
1	2	2170664	2.2819330450027745E7
2	3	2371877	2.510962085003214E7
3	4	2293040	2.465736017003168E7

Combined totals for February through April

Query

```
with task1 as (
SELECT extract(month from pickup_datetime) as month,
sum(passenger_count) as total_passengers,
sum(total_amount) as total_amount
FROM `nyc-tlc.green.trips_2015`
GROUP BY month
having month between 2 and 4
order by month
)
select sum(total_passengers) as total_passengers_FebtoApr,
sum(total_amount) as total_amount_FebtoApr
from task1
```

Row	total_passengers_FebtoApr	total_amount_FebtoApr
1	6835581	7.258631147009157E7

Results Preview

2. What has been the average hourly passenger count throughout the year?

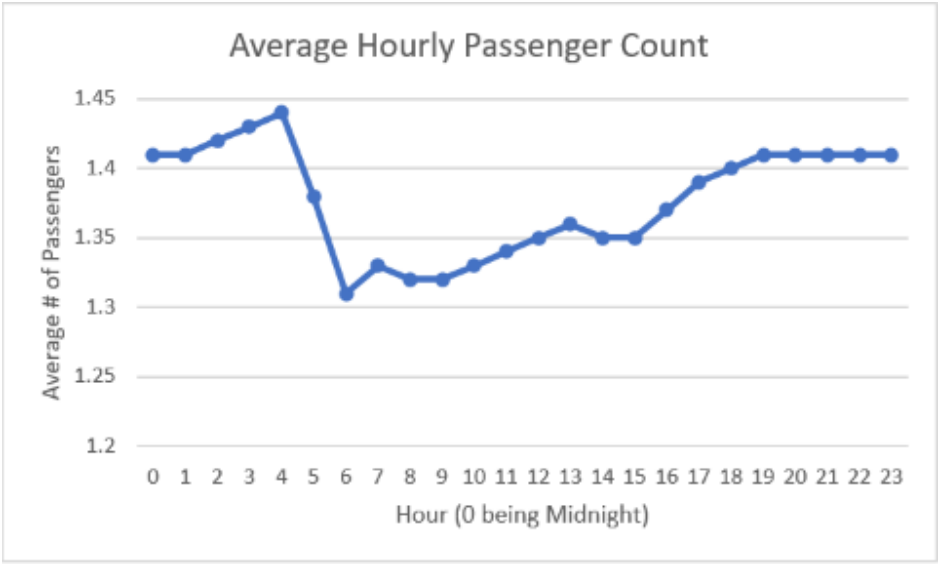
Average by Hour

Query

```
SELECT extract(hour from pickup_datetime) as hour,
ROUND(avg((passenger_count)), 2) as avg_passenger_count
FROM `nyc-tlc.green.trips_2015`
group by hour
order by hour
```

Results Preview and Visualization

Row	hour	avg_passenger_count
1	0	1.41
2	1	1.41
3	2	1.42
4	3	1.43
5	4	1.44
6	5	1.38
7	6	1.31
8	7	1.33
9	8	1.32



Single aggregate average passengers per hour

Query

```
With avg_pass_byhour as (SELECT extract(hour from pickup_datetime) as hour,
ROUND(avg((passenger_count)), 2) as avg_passenger_count
FROM `nyc-tlc.green.trips_2015`
group by hour
order by hour)

select round(avg(avg_passenger_count),2) as avg_passengers_perhour
from avg_pass_byhour
```

Results Preview

Row	avg_passengers_perhour
1	1.38

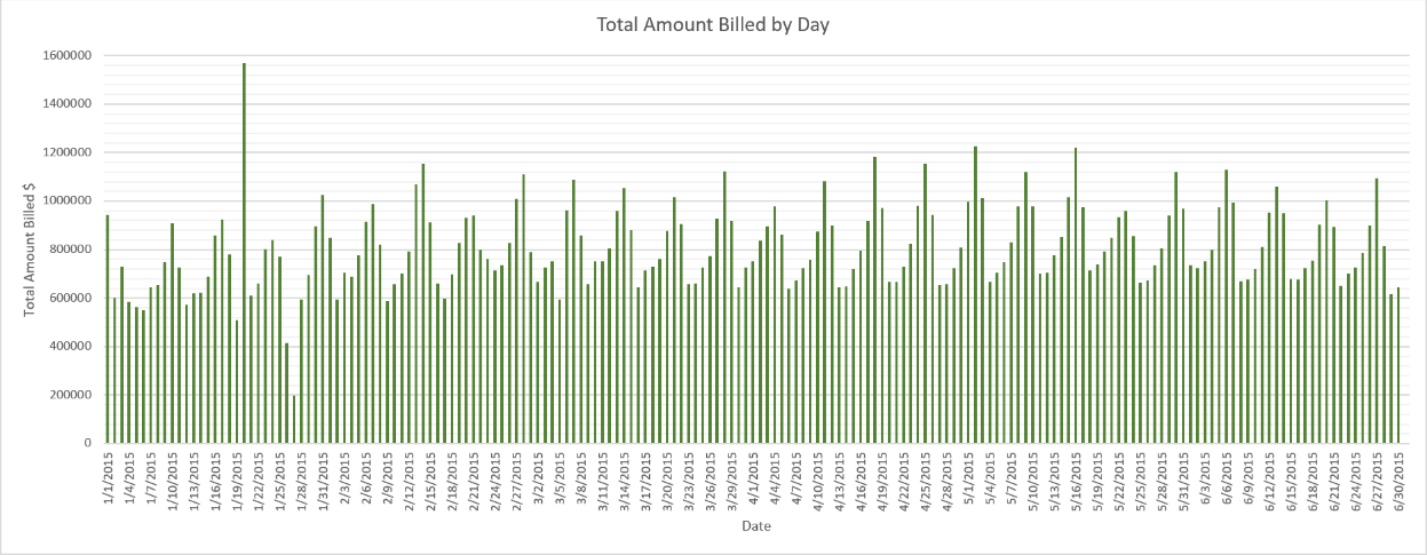
3. What has been the change/delta in total amount billed over days?

Query

```
WITH total_amount_byday as
(SELECT extract(date from pickup_datetime) as date,
ROUND(sum(total_amount), 2) as total_amount
FROM `nyc-tlc.green.trips_2015`
group by date
order by date)
select date, total_amount,
ROUND(total_amount - LAG(total_amount)
OVER (order by date), 2) total_amount_change
from total_amount_byday
order by date
```

Results Preview and Visualization

Row	date	total_amount	total_amount_change
1	2015-01-01	941985.49	*null*
2	2015-01-02	599885.13	-342100.36
3	2015-01-03	728199.31	128314.18
4	2015-01-04	586520.42	-141678.89
5	2015-01-05	564458.23	-22062.19
6	2015-01-06	549165.09	-15293.14
7	2015-01-07	644796.69	95631.6
8	2015-01-08	653953.39	9156.7
9	2015-01-09	749604.77	95651.38



4. What hour of the day has seen the longest rides in April?

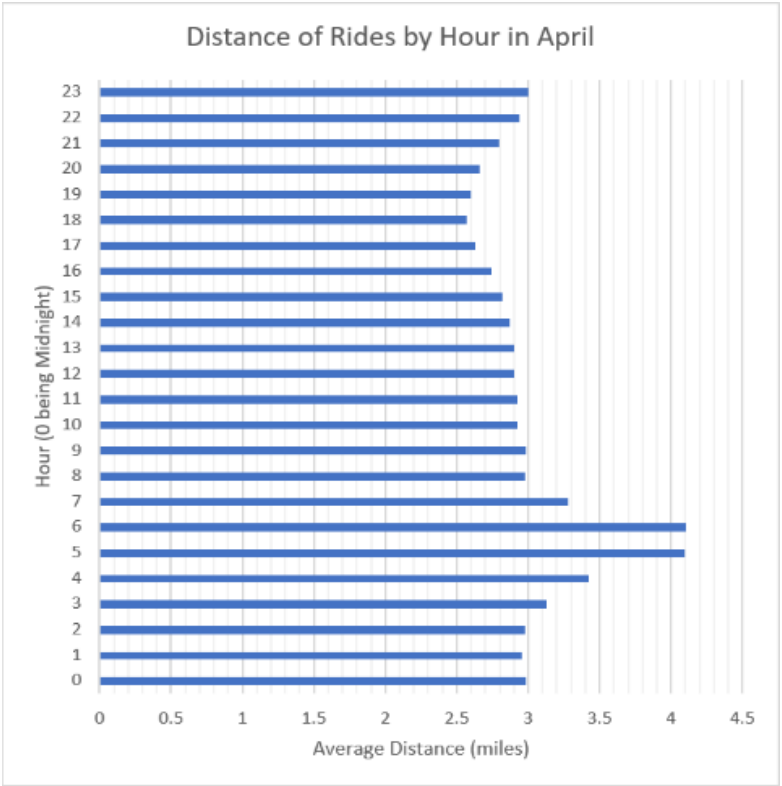
Distances by Hour

Query

```
select extract(hour from pickup_datetime) as hour,
avg(trip_distance) as avg_distance
FROM `nyc-tlc.green.trips_2015`
where extract(month from pickup_datetime) = 4
group by hour
order by avg_distance desc
```

Results Preview and Visualization

Row	hour	avg_distance
1	6	4.102159090909088
2	5	4.098513747689466
3	4	3.4271303127361725
4	7	3.2781622117701046
5	3	3.12637079770411
6	23	2.997424506704458
7	9	2.9885405909979297
8	0	2.9843876001742746
9	2	2.9778413650092403



The hour of 6am (6:00 - 6:59) has the greatest average distance, with the hour of 5am being a very close second.

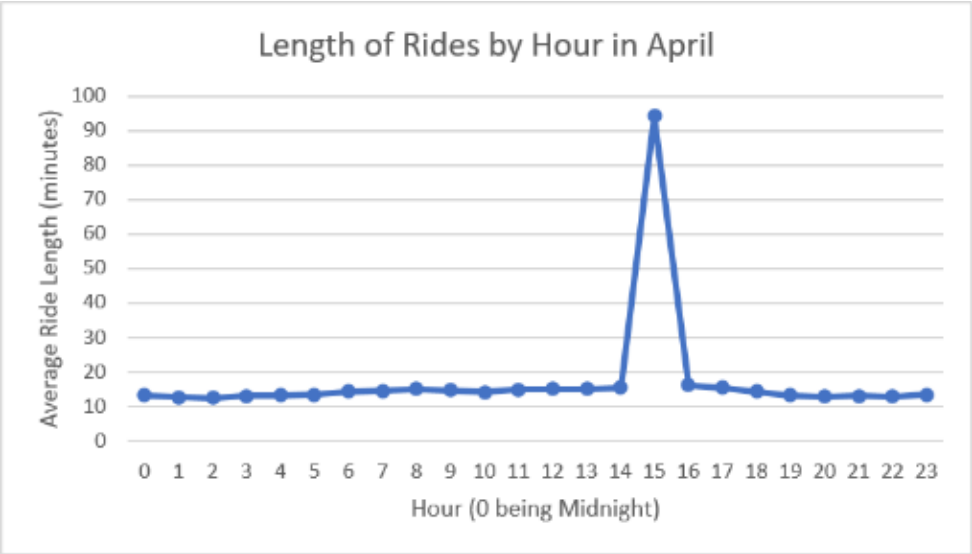
Length of rides by time

Query

```
select extract(hour from pickup_datetime) as hour,
Round(avg(DATETIME_DIFF(dropoff_datetime,
pickup_datetime, minute)), 2) as avg_length_mins
FROM `nyc-tlc.green.trips_2015`
where extract(month from pickup_datetime) = 4
group by hour
order by avg_length_mins desc
```

Results Preview and Visualization

Row	hour	avg_length_mins
1	15	94.18
2	16	16.21
3	14	15.53
4	17	15.49
5	8	15.24
6	13	15.15
7	12	15.11
8	11	15.09
9	9	14.8



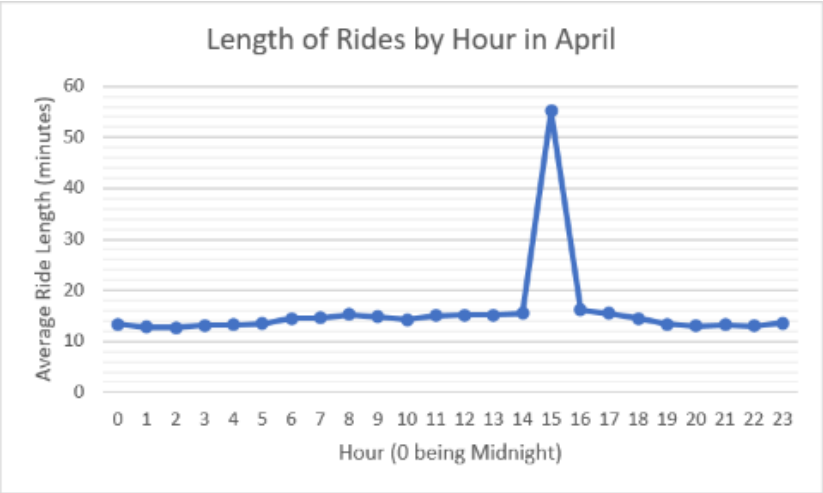
With outlier/anomaly removed (see appendix note 2 for more info)

Query

```
select extract(hour from pickup_datetime) as hour,
Round(avg(DATETIME_DIFF(dropoff_datetime, pickup_datetime, minute)),
2) as avg_length_mins
FROM `nyc-tlc.green.trips_2015`
WHERE extract(month from pickup_datetime) = 4
and DATETIME_DIFF(dropoff_datetime, pickup_datetime, minute) <>
(SELECT max(DATETIME_DIFF(dropoff_datetime, pickup_datetime,
minute)) from `nyc-tlc.green.trips_2015`)
group by hour
order by avg_length_mins desc
```

Results Preview and Visualization

Row	hour	avg_length_mins
1	15	55.2
2	16	16.21
3	14	15.53
4	17	15.49
5	8	15.24
6	13	15.15
7	12	15.11
8	11	15.09
9	9	14.8



With or without the anomalous outlier, hour 15 (3:00 – 3:59pm) had the longest rides on average in April by a significant amount.

PART 2

Let's say you needed to connect to an API and pull down data that was into a CSV file and into a database. Assume it's a large amount of data, 500MB a day. Please share a few lines of Java or Python code that shows how you would connect and describe, in your ideal world, where the code would be physically running, where the downloaded data would be saved and which database you would want to store it into.

I would have this code run on a Linux or Windows server that has the required network access to communicate with the desired API (internal or public).
After pulling the CSV using a web request, I would store the data from the CSV in either a variable defined Pandas DataFrame, or locally save as a file on my server. Then I would have my Python script store the data as a table in a SQL DataBase.
Below is a Python script that I wrote and tested on my home lab server with success on a ~500MB file (credentials altered).

```
import requests
import sqlalchemy
import mysql.connector
import pandas as pd
import io

#getting the file and saving its contents to url_content
r = requests.get('http://192.168.1.40:81/latest.csv') #if API needs username + pw, add argument
auth=('username', 'password')
url_content = r.content

##don't need this if we're not saving it locally
#csv_file = open('latest.csv', 'wb')
#csv_file.write(url_content)
#csv_file.close()
#latest_df = pd.read_csv('latest.csv')
##print(latest_df.head())

#if not saving the csv locally, create a DF from CSV content from API:
latest_df = pd.read_csv(io.StringIO(url_content.decode('utf-8')))
##print(latest_df.head())

#interacting with database
#log in to DB 'mysql+mysqlconnector://<username>:<password>@<host name or IP>/<database name>'
database_connection
=sqlalchemy.create_engine('mysql+mysqlconnector://usrname:psswrd@192.168.1.40/taxi_data).connect(
)

#Using the open database connection, put dataframe as table into database
latest_df.to_sql(con=database_connection, name='latest_taxidata_pull', if_exists='replace')
database_connection.close()
```

To schedule this script to run daily, I would use Crontab for a Linux server (which is what I used when testing) or the Task Scheduler for a Windows server and choose an appropriate daily runtime.

APPENDIX

1. While working on Task 1 of Part 1, I was curious about the total_amount category, since in the BigQuery preview of the data, they were all showing as 0.0, and yet the sum was coming out large. So I looked for the top 5 values of the total_amount category:

```
SELECT total_amount
FROM `nyc-tlc.green.trips_2015`
order by 1 desc
limit 5
```

And was surprised to see that the highest total_amount paid for a ride was nearly a million dollars (definitely an outlier!) but the next highest was around 8k.

Row	total_amount
1	989970.39
2	8011.3
3	5660.1
4	4035.46
5	3789.3

Looking for more info on that ride:

```
select * from `nyc-tlc.green.trips_2015`
where total_amount = (SELECT max(total_amount) from `nyc-tlc.green.trips_2015`)
```

(Results displayed as JSON to increase readability)


```
  {
    "pickup_datetime": "2015-01-20 00:47:29 UTC",
    "dropoff_datetime": "2015-01-20 00:50:01 UTC",
    "store_and_fwd_flag": "N",
    "rate_code": "1",
    "pickup_longitude": "-73.94461822509766",
    "pickup_latitude": "40.8341064453125",
    "dropoff_longitude": "-73.94622802734375",
    "dropoff_latitude": "40.83613204956055",
    "passenger_count": "1",
    "trip_distance": "0.18",
    "fare_amount": "3.5",
    "extra": "0.5",
    "mta_tax": "0.5",
    "tip_amount": "0.0",
    "tolls_amount": "0.0",
    "ehail_fee": null,
    "total_amount": "989970.39",
    "payment_type": "2",
    "distance_between_service": "0.0",
    "time_between_service": "7526",
    "trip_type": "1"
  }
]
```


Why did the total amount come out so high when the trip_distance was only 0.18, fare_amount was 3.5, and tip_amount was 0.0? Some kind of error? Possible money laundering? If this were a real client's data, it may be worth reporting and/or looking into. But it happened in January, so it does not affect the totals for the requested data.

2. While working on Task 4 of Part 1, I was surprised by how much larger the average ride length (in minutes) was for hour 15 compared to the others, so I decided to check for an outlier by checking the maximum ride length:

```
select *, DATETIME_DIFF(dropoff_datetime, pickup_datetime, minute) from `nyc-tlc.green.trips_2015`
where DATETIME_DIFF(dropoff_datetime, pickup_datetime, minute) = (SELECT
max(DATETIME_DIFF(dropoff_datetime, pickup_datetime, minute)) from `nyc-tlc.green.trips_2015`)
```

(Results displayed as JSON to increase readability) The last entry is the calculated minutes difference between pickup time and dropoff time.


```
[
  {
    "pickup_datetime": "2015-04-04 15:16:02 UTC",
    "dropoff_datetime": "2021-04-03 21:13:32 UTC",
    "store_and_fwd_flag": "Y",
    "rate_code": "1",
    "pickup_longitude": "-73.95892333984375",
    "pickup_latitude": "40.729000091552734",
    "dropoff_longitude": "-73.95892333984375",
    "dropoff_latitude": "40.72900390625",
    "passenger_count": "1",
    "trip_distance": "0.0",
    "fare_amount": "2.5",
    "extra": "0.5",
    "mta_tax": "0.5",
    "tip_amount": "0.0",
    "tolls_amount": "0.0",
    "ehail_fee": null,
    "total_amount": "3.8",
    "payment_type": "4",
    "distance_between_service": "0.0",
    "time_between_service": "1302",
    "trip_type": "1",
    "f0_": "3155397"
  }
]
```


This trip was 3155397 minutes or over ***5 years long***, and yet the trip distance was 0.0 and the fare amount was 2.5. Something must be wrong here. I would probably remove this observation from the data set and redo my aggregations, since there must have been an input error or something.

I was unable to actually delete the entry since I do not have permissions to alter this public data set, but I was able to exclude it from the aggregates with the extended WHERE clause shown in the last response to the task. After this, the data looked more normalized (though hour 15 is still an outlier among the averages in April).