

Uber Data Analytics Project

Github link: <https://github.com/AtharvaBOT7/Uber-Data-Analysis>

Pre-requisites:

- Python
- SQL
- Jupyter Notebook

Google Cloud Platform (GCP) Services to be used:

1. Cloud Storage
2. Computer Engine
3. BigQuery
4. Looker Studio

Google Cloud Storage is an online file storage service which is provided by Google, it allows us to store and retrieve data from and in the cloud, which makes the data accessible anywhere in the world with a good internet connection.

Google Compute Engine is a service that provides virtual machines for running applications and services. It allows us to create, configure and manage virtual machines with different operating systems and hardware configurations.

BigQuery is a SQL-like tool which helps us to store, analyze and query the data which is present in the cloud. It is a serverless and highly scalable solution that can process and analyze large datasets in real time.

(Serverless means that the cloud provider will handle all the server infrastructure in the back end and the users do not need to do all the configuration themselves.)

Finally, we will use Looker Studio, this is a tool which is used for web-based data visualization and reporting. It helps us to create visually interactive dashboards combined from a variety of data sources.

We are going to use MAGE as our open source pipelining tool for transforming and integrating data.

More information about Fact table and Dimension table:


Fact Table:

The fact table is a centralised table in the data warehouse that stores quantitative data and measurable facts about business processes. Example:

 **Example:** `sales_fact`

| <code>sale_id</code> | <code>date_key</code> | <code>product_key</code> | <code>customer_key</code> | <code>amount</code> | <code>quantity</code> |
|----------------------|-----------------------|--------------------------|---------------------------|---------------------|-----------------------|
| 1001 | 20230701 | 501 | 301 | 150.00 | 3 |

Dimension table stores descriptive data attributes about the data in fact table. Example:

 **Example:** `product_dimension`

| <code>product_key</code> | <code>product_name</code> | <code>category</code> | <code>brand</code> |
|--------------------------|---------------------------|-----------------------|--------------------|
| 501 | iPhone 14 | Smartphone | Apple |

Now that we have information about the fact and dimension tables, we will start cleaning the data in our local machine before uploading the data onto Google Cloud Platform.

The first and foremost step is to convert this csv data into a Dataframe for better access. After converting to dataframe, we will assign the column 'trip_id' as the index for the whole table.

Then we will check for any null values in the dataframe. As this data belongs to the NYC government, it is extremely refined therefore it does not contain any null or Nan values.

Then when we run the .info() method on this data, we find out that a few attributes are object type attributes and we cannot perform analysis on object type attributes nor we can make any modifications to that attribute therefore we convert it to a datetime variable which can be modified.

The next step is to remove any duplicates present in 'tpep_pickup_datetime', 'tpep_dropoff_datetime', 'passenger_count', 'trip_distance', 'RatecodeID', 'pickup_longitude', 'pickup_latitude', 'dropoff_longitude', 'dropoff_latitude', 'payment_type'.

After this we set the index to the nearest index count for each of the individual columns.

The next step is to merge multiple dimension tables with raw data using the 'trip_id' column as the primary key.

This concludes our dimension modelling and data cleaning processes.

Then we login to our Google Cloud Console, it is free for first time users and they provide \$300 free credits for beginners to get a hang of the platform.

Then we will create a bucket and upload our Uber data to it. Here, it is important to make the data publicly available, therefore we change the permissions and give a fine-grained access to all the users. This will make the data publicly available (read only access) to anyone with a link.

Data link - https://storage.googleapis.com/uber-data-engineering-bucket-yt/uber_data.csv

The above is then uploaded to a newly created bucket.

*** Remember, the instance name for each and every bucket must be unique worldwide, it cannot be the same for two users in any case.**

After creating the bucket and uploading the data, we will need to make an instance of the compute engine. Computer engine means a computer which will help us access the data which is present on the Google Cloud.

For our case, we will choose the E2 version with 4 CPUs and 16Gb RAM.

Now we will connect our instance with our data bucket using a SSH link in the browser itself. Google offers hassle free browser connection of our virtual machine with the data uploaded on the cloud. Unlike AWS S3, Google Cloud is much more easy to use and beginner friendly.

We have installed pandas in our virtual environment.

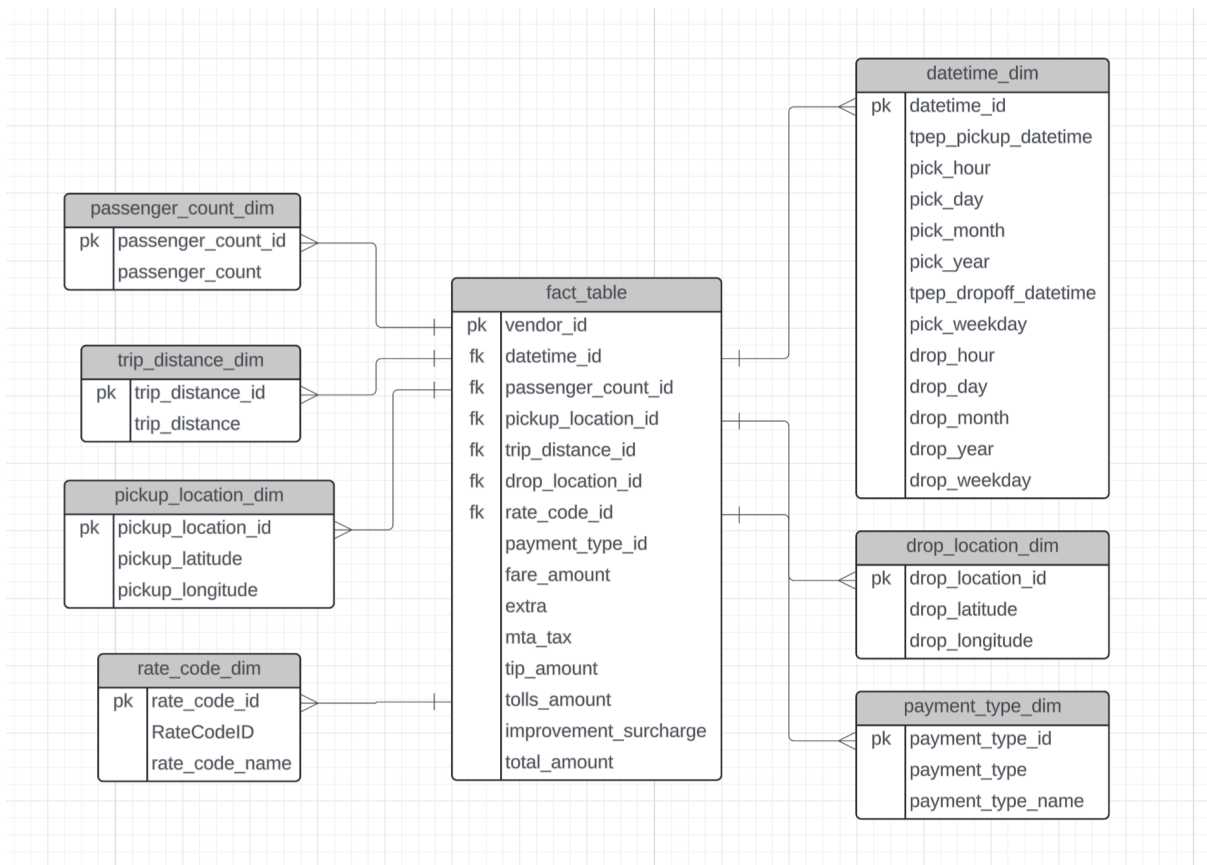
Then we installed mage-ai.

The next step is to start a new project named uberdeproject on the local host, running on port 6789.

We will then navigate to VM instances → our project → then scroll down to network interfaces → go to the available interface → go to the firewalls sections and here we have to add our port 6789 to access the Mage web UI using an external IP address.

Now you will create a new firewall policy giving access to all the instances on the network and all the IPs as well use 0.0.0.0/0 in the IP section and select the protocol to be TCP, and then enter the configuration port as 6789.

This will help us access the mage web UI using the `http:<your_external_IP>:6789`.



In the Mage web UI, we created a Data Loader file using Python → API:

```
PY DATA LOADER uber_data_loader ← Edit parents

import io
import pandas as pd
import requests
if 'data_loader' not in globals():
    from mage_ai.data_preparation.decorators import data_loader
if 'test' not in globals():
    from mage_ai.data_preparation.decorators import test

@data_loader
def load_data_from_api(*args, **kwargs):
    """
    Template for loading data from API
    """
    url = 'https://storage.googleapis.com/uber-data-engineering-bucket-yt/uber_data.csv'
    response = requests.get(url)

    return pd.read_csv(io.StringIO(response.text), sep=',')

@test
def test_output(output, *args) → None:
    """
    Template code for testing the output of the block.
    """
    assert output is not None, 'The output is undefined'
```

We then inserted the link to our data.

The next step is to create a transformer, this will transform our data, i.e. this will clean our data just like we first did trial cleaning of our data in the python file earlier, we will write the same code here to clean the data.

```
PY TRANSFORMER uber_data_transformation ← 1 parent
Positional arguments for decorated function:
@transformer
def transform(data):
    data → uber_data_loader

import pandas as pd

if 'transformer' not in globals():
    from mage_ai.data_preparation.decorators import transformer
if 'test' not in globals():
    from mage_ai.data_preparation.decorators import test

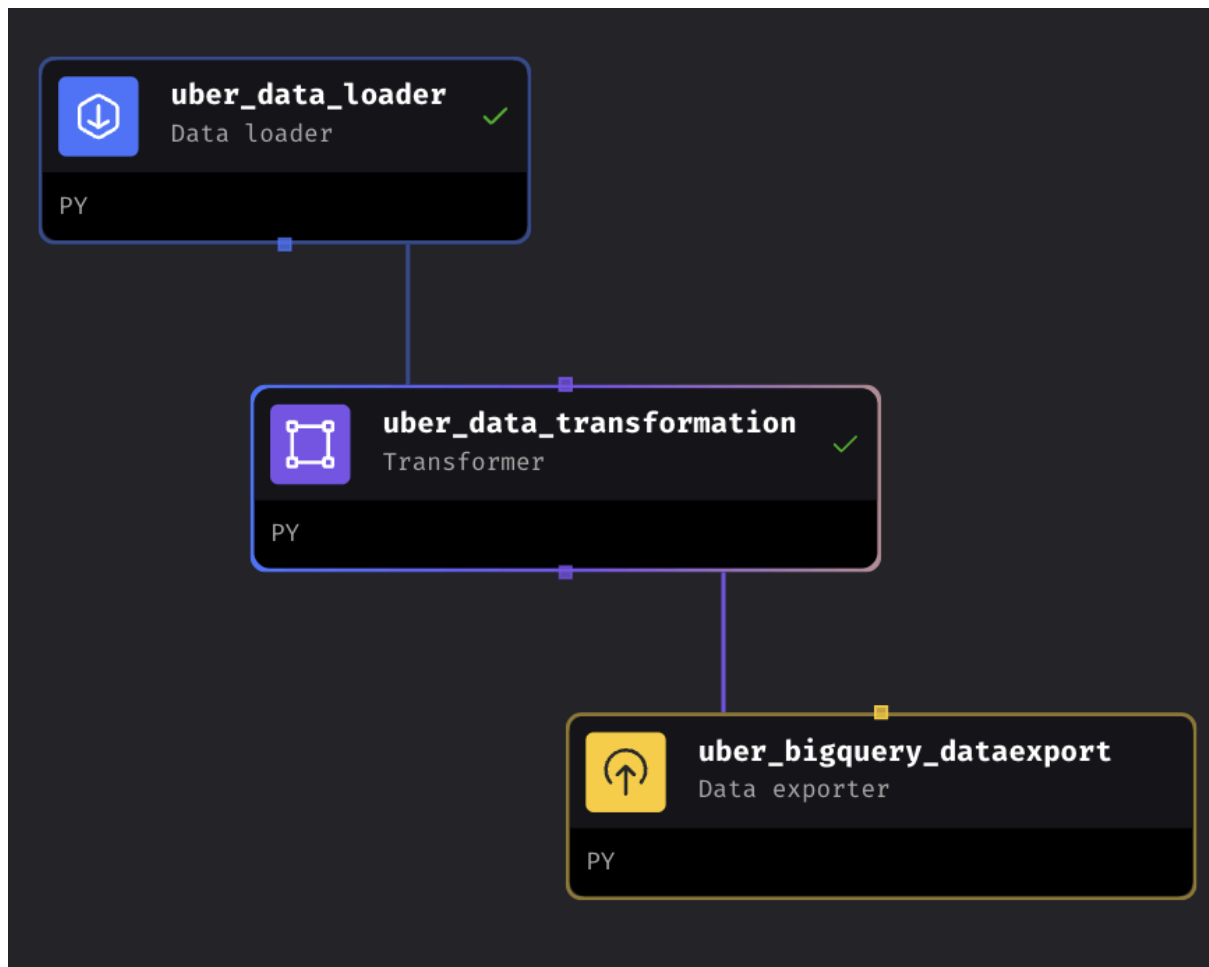
@transformer
def transform(data, *args, **kwargs):
    """
    Transformer block to create dimension and fact tables from raw Uber trip data.
    """
    df = data.copy()
    df['trip_id'] = df.index
    df['tpep_pickup_datetime'] = pd.to_datetime(df['tpep_pickup_datetime'])
    df['tpep_dropoff_datetime'] = pd.to_datetime(df['tpep_dropoff_datetime'])

    # Datetime dimension
    datetime_dim = df[['tpep_pickup_datetime', 'tpep_dropoff_datetime']].drop_duplicates().reset_index()
    datetime_dim['pick_hour'] = datetime_dim['tpep_pickup_datetime'].dt.hour
    datetime_dim['pick_day'] = datetime_dim['tpep_pickup_datetime'].dt.day
    datetime_dim['pick_month'] = datetime_dim['tpep_pickup_datetime'].dt.month
    datetime_dim['pick_year'] = datetime_dim['tpep_pickup_datetime'].dt.year
    datetime_dim['pick_weekday'] = datetime_dim['tpep_pickup_datetime'].dt.weekday
    datetime_dim['drop_hour'] = datetime_dim['tpep_dropoff_datetime'].dt.hour
    datetime_dim['drop_day'] = datetime_dim['tpep_dropoff_datetime'].dt.day
    datetime_dim['drop_month'] = datetime_dim['tpep_dropoff_datetime'].dt.month
```

The next step is to create a data exporter, for this we will have to make modifications to the `io_config.yaml` file present in the mage UI by default.

First we will need to visit APIs and Services in our Google Console, create a new credential and then inside that create a new key and give BigQueryAdmin access to the key so that we are able to create and update tables. After creating the key, Google will ask you to download a credentials file, we will need this file to configure our BigQuery with our Mage UI. We will go to the `io_config.yaml` file and then paste the credentials as per the Google downloaded file.

This is our pipeline till now:



Code for each of the blocks is present on the github link present at the start.

This is our first query and this is the result it will return if the data is successfully uploaded to the Google Cloud bucket.

Query:

```
1 select * from `data-with-atharva.uber_dataset_project.fact_table` limit 10;
```

Result:

✔ Query completed

Query results [Save results](#) [Open in](#)

Job information **Results** Chart JSON Execution details Execution graph

| Row | trip_id | VendorID | datetime_id | passenger_count... | trip_distance_id | rate_code_id | store_and_fwd_flag | pickup_locati |
|-----|---------|----------|-------------|--------------------|------------------|--------------|--------------------|---------------|
| 4 | 9032 | 2 | 9011 | 0 | 880 | 0 | N | |
| 5 | 2231 | 2 | 2225 | 0 | 375 | 0 | N | |
| 6 | 37906 | 2 | 37849 | 0 | 392 | 0 | N | : |
| 7 | 88843 | 2 | 88711 | 1 | 481 | 0 | N | : |
| 8 | 29814 | 2 | 29770 | 4 | 79 | 0 | N | : |
| 9 | 71008 | 2 | 70906 | 1 | 169 | 0 | N | : |
| 10 | 407 | 2 | 403 | 0 | 283 | 0 | N | : |

Few Example Queries and their outputs:

```

1 select VendorID, AVG(total_amount) as Average_Amount from `data-with-atharva.uber_dataset_project.fact_table`
2 group by VendorID;

```

✔ Query completed

Query results [Save results](#)

Job information **Results** Chart JSON Execution details Execution graph

| Row | VendorID | Average_Amount |
|-----|----------|--------------------|
| 1 | 2 | 16.271339114879986 |
| 2 | 1 | 17.294056369399502 |

Untitled query

Run

Save

Download

Share

Schedule

Open in

More

```
1 select b.payment_type_name, avg(a.tip_amount) from `data-with-atharva.uber_dataset_project.fact_table` a
2 join `data-with-atharva.uber_dataset_project.payment_type_dim` b
3 on a.payment_type_id = b.payment_type_id
4 group by b.payment_type_name;
```

Query completed

Query results

Save results

Job information

Results

Chart

JSON

Execution details

Execution graph

| Row | payment_type_name | f0_ |
|-----|-------------------|--------------------|
| 1 | No charge | 0.038265895953... |
| 2 | Dispute | -0.017368421052... |
| 3 | Cash | 6.505436255759... |
| 4 | Credit card | 2.813692973492... |

These were a few example queries to check if our table is properly uploaded or not and if Google BigQuery is working or not, now for analysis we will create a new table with only the parameters (columns) which might be of use to use during an analysis, then we will use Google Looker Studio to make a visually appealing dashboard.


Now we will create a new table for analysis, this is the query which will be used to create a new table:

```

1 CREATE OR REPLACE TABLE `data-with-atharva.uber_dataset_project.table_analytics` AS (
2 SELECT
3     f.vendor_id,
4     d.tpep_pickup_datetime,
5     d.tpep_dropoff_datetime,
6     p.passenger_count,
7     t.trip_distance,
8     r.rate_code_name,
9     pick.pickup_latitude,
10    pick.pickup_longitude,
11    dropoff.dropoff_latitude,
12    dropoff.dropoff_longitude,
13    pay.payment_type_name,
14    f.fare_amount,
15    f.extra,
16    f.mta_tax,
17    f.tip_amount,
18    f.tolls_amount,
19    f.improvement_surcharge,
20    f.total_amount
21 FROM
22     `data-with-atharva.uber_dataset_project.fact_table` f
23 JOIN
24     `data-with-atharva.uber_dataset_project.datetime_dim` d
25     ON f.datetime_id = d.datetime_id
26 JOIN
27     `data-with-atharva.uber_dataset_project.passenger_count_dim` p
28     ON f.passenger_count_id = p.passenger_count_id
29 JOIN
30     `data-with-atharva.uber_dataset_project.trip_distance_dim` t
31     ON f.trip_distance_id = t.trip_distance_id
32 JOIN
33     `data-with-atharva.uber_dataset_project.rate_code_dim` r

```

This is how you will connect the BigQuery data source after selecting a blank project:



BigQuery

By Google

BigQuery is Google's fully managed, petabyte scale, low-cost analytics data warehouse. BigQuery charges for querying/processing data. Those queries are charged to the credit card of the billing project.

[LEARN MORE](#)
[REPORT AN ISSUE](#)

RECENT PROJECTS

MY PROJECTS

SHARED PROJECTS

CUSTOM QUERY

PUBLIC DATA-SETS

Project

data-with-Atharva

My First Project

My First Project

Data set

uber_dataset_Project

Table

datetime_dim

dropoff_location_dim


fact_table

passenger_count_dim

payment_type_dim

pickup_location_dim

rate_code_dim

table_analytics 

trip_distance_dim

Cancel

Add