# Documentation:

Get Data: The data files for each industry were downloaded from the TLC Trip Records page. To arrange the data files, create a local storage folder structure. studied industry-specific data dictionaries to understand the data kinds and columns.

## Organize:

Below tree represents downloaded files schema in my local directory

TLC\_Trip\_Record\_Data/

├── Yellow\_Taxi/

│ ├── yellow\_data\_1.parquet

│ ├── yellow\_data\_2.parquet

│ └── ...

├── Green\_taxi/

│ ├── green\_data\_1.parquet

│ ├── green\_data\_2.parquet

│ └── ...

├── For\_Hire\_vechicle/

│ ├── for\_hire\_data\_1.parquet

│ ├── for\_hire\_data\_2.parquet

│ └── ...

└── High\_Volume\_For\_Hire/

├── high\_volume\_data\_1.parquet

├── high\_volume\_data\_2.parquet

└── ...

Below screenshot represents the column names from all the four data files let’s check the possible relationship among them:

**Yellow Trips and Green trips:**

Common attributes between the two schemas include.

VendorID, PickupDateTime, DropoffDateTime, PassengerCount, TripDistance, PULocationID, DOLocationID, RateCodeID, StoreAndFwdFlag, PaymentTypeID, FareAmount, Extra, MTA\_Tax, ImprovementSurcharge, TipAmount, TollsAmount, and TotalAmount.

These two schemas appear to represent the same data from different sources. That is why a merger or consolidation of the data could represent a possible relationship between them.

A black background with white text

Description automatically generated with low confidence

**Yellow Trips and FHV Trip Records:**

Relationship: Common attributes between the two schemas include PickupDateTime, DropoffDateTime, PULocationID, DOLocationID.

The use of yellow taxis within the structure of FHV may be represented by a link between yellow taxi trip records and FHV trip records. We may learn more about common elements of the journeys, such well-liked pickup/dropoff sites or busy periods of travel, by merging the overlapping fields.

**Green Trip Records and FHV Trip Records:**

Relationship: Common attributes between the two schemas include VendorID, PickupDateTime, DropoffDateTime, PULocationID, DOLocationID, and other related fields.

The fact that both the data dictionaries "Yellow Taxi Trip Records" and "FHV Trip Records" record information on numerous New York City transportation industry subsectors is what unites them. By combining the overlapping values, we may be able to learn more about the common trip components, such as popular pickup/drop-off locations or busy travel times.

**Fact Tables:**

**Yellow Taxi Fact Table:**

**Columns**: VendorID, tpep\_pickup\_datetime, tpep\_dropoff\_datetime, Passenger\_count, Trip\_distance, PULocationID, DOLocationID, RateCodeID, Store\_and\_fwd\_flag, Payment\_type, Fare\_amount, Extra, MTA\_tax, Improvement\_surcharge, Tip\_amount, Tolls\_amount, Total\_amount, Congestion\_Surcharge, Airport\_fee

**Green Taxi Fact Table:**

**Columns**: VendorID, lpep\_pickup\_datetime, lpep\_dropoff\_datetime, Passenger\_count, Trip\_distance, PULocationID, DOLocationID, RateCodeID, Store\_and\_fwd\_flag, Payment\_type, Fare\_amount, Extra, MTA\_tax, Improvement\_surcharge, Tip\_amount, Tolls\_amount, Total\_amount, Trip\_type

**FHV Trip Records Fact Table:**

**Columns**: Dispatching\_base\_num, Pickup\_datetime, DropOff\_datetime, PULocationID, DOLocationID, SR\_Flag

**HVFHV Trip Records Fact Table:**

**Columns**: Hvfhs\_license\_num, Dispatching\_base\_num, Pickup\_datetime, DropOff\_datetime, PULocationID, DOLocationID, originating\_base\_num, request\_datetime, on\_scene\_datetime, trip\_miles, trip\_time, base\_passenger\_fare, tolls, bcf, sales\_tax, congestion\_surcharge, airport\_fee, tips, driver\_pay, shared\_request\_flag, shared\_match\_flag

**Dimension Tables:**

Common dimension tables for both yellow and green taxi

**Location Dimension Table**

Columns: LocationID, LocationName, Borough, Zone, Latitude, Longitude

**Vendor Dimension Table**

Columns: VendorID, VendorName

|  |
| --- |
| VendorID | VendorName |
| 1 | Vendor A |
| 2 | Vendor B |
| 3 | Vendor C |

**Rate code Dimension Table**

Columns: RateCodeID, RateCode

Here are the possible records for rate code table.

|  |  |
| --- | --- |
| **RateCodeID** | **RateCode** |
| 1 | Standard rate |
| 2 | JFK |
| 3 | Newark |
| 4 | Nassau or Westchester |
| 5 | Negotiated fare |
| 6 | Group ride |

**PaymentType Dimension table**

|  |  |
| --- | --- |
| PaymentType | PaymentType\_name |
| 1 | Credit card |
| 2 | Cash |
| 3 | No charge |
| 4 | Dispute |
| 5 | Unknown |
| 6 | Voided trip |

One extra dimension table for green taxi:

|  |  |
| --- | --- |
| **Trip\_type** | |
| Trip\_type\_id | Trip\_type |
| 1 | Street-hail |
| 2 | Dispatch |

For **HVFHV Trip data dictionary,** dimension table for licensees which is:

|  |  |
| --- | --- |
| **HVFHS licensees** | |
| Hvfhs\_license\_num | **HVFHS licensee** |
| HV0002 | Juno |
| HV0003 | Uber |
| HV0004 | Via |
| HV0005 | Lyft |

For FHV and HVFHS we can get two common dimension tables i.e Dispatching Base and Location table.

Fact table data are further contextualized and described by dimension tables. Here, dimension tables like the Location, Vendor, Rate Code, and Payment Type tables assist in categorizing and analyzing taxi trip data based on variables like location, vendor, price, and payment type.

## Ingest:

We may use a combination of cron tasks for Linux-based computers and cloud storage APIs to automate the script to run at 7 am on the 15th of every month and ingest the output into a AWS S3 cloud storage.

Because of its scalability, dependability, and accessibility, AWS S3 is chosen. It provides almost limitless storage space, excellent data availability, and exceptional data durability. S3 offers a straightforward interface for simple data management, is reasonably priced, and connects well with other AWS services. Due to these elements, AWS S3 is the best option for handling and storing the substantial amount of NYC cab trip records.

Setp 1: Set up a cron job:

Write a cron job configuration file let’s say corn\_config.txt with the following entry:

0 7 15 \* \* python3 /path/to/script.py

This cron job configuration specifies that the script should be executed at 7 am on the 15th of every month.

Save the cron\_config.txt file.

Step 2: Schedule the cron job:

Run the following command to load the cron job configuration:

crontab cron\_config.txt. This will schedule the script to run automatically at the specified time.

Step3: Set up an aws account and create a storage bucket integrate this code in python file.

import boto3

# Configure AWS credentials

aws\_access\_key = 'your-access-key'

aws\_secret\_key = 'your-secret-key'

bucket\_name = 'your-bucket-name'

# Create an S3 client

s3 = boto3.client('s3', aws\_access\_key\_id=aws\_access\_key, aws\_secret\_access\_key=aws\_secret\_key)

# Upload the processed Parquet files to S3

for file\_path in parquet\_files:

s3.upload\_file(file\_path, bucket\_name, file\_path.split("/")[-1])

After the validation and transformations are finished, we can set up the code mentioned above in a.py file, upload the data in a pandas data frame into a file, and then upload it to cloud AWS s3.

## Additional notes:

1. More information about the analysis requirement helps to get more insights from the data.
2. Are there any specific data transformations or cleansing steps required before ingesting the data into the cloud storage.
3. Are there any requirements to identify and handle duplicate records during ingestion?
4. For this type of validations and transformations we can use the azure data factory for configuring and scheduling purpose and, Apache Air flow provides scalable and efficient options for data processing and integration.