# Overview of the Assignment:

Let’s focus on SCD maintenance of the load process and analysis and presentation.

**Part 9:** Provide **updated** code and screenshots of loading a delta into the one of the dimensions which is either SCD2 or SCD3. Your code should show how SCD2 or SCD3 is maintained when changes or a brand-new record is detected.

DimAirbnb\_Host table: SCD Type 3

1. --- Add a new column to store the effective date of the change
2. ALTER TABLE DimAirbnb\_Host
3. ADD Effective\_date DATETIME NULL;
4. --- Modify the existing Host\_name column to be nullable
5. ALTER TABLE DimAirbnb\_Host
6. ALTER COLUMN Host\_name VARCHAR(255) NULL;

-- Create a trigger to update the Host\_name\_prev and Effective\_date columns when Host\_name is updated

CREATE TRIGGER tr\_DimAirbnb\_Host\_Update

ON DimAirbnb\_Host

AFTER UPDATE

AS

BEGIN

IF UPDATE(Host\_name)

BEGIN

UPDATE DimAirbnb\_Host

SET Host\_name\_prev = CASE

WHEN DimAirbnb\_Host.Host\_name IS NULL THEN NULL

ELSE DELETED.Host\_name

END,

Effective\_date = GETDATE()

FROM DimAirbnb\_Host

INNER JOIN DELETED ON DimAirbnb\_Host.Listing\_host\_id = DELETED.Listing\_host\_id

END

END

-- Update an existing record

UPDATE DimAirbnb\_Host

SET Host\_name = 'John Wick'

WHERE Host\_id = 1;

-- Insert a new record

INSERT INTO DimAirbnb\_Host (Listing\_host\_id, Host\_name, Host\_name\_prev, airbnb\_id, Effective\_date)

VALUES (2571, 'Teedo Ruan', NULL, 1101224, GETDATE());

A screenshot of a computer screen

Description automatically generated with medium confidence

A screenshot of a computer screen

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

DimTaxi table: SCD Type 2

1. --- SCD Type 2
2. CREATE TABLE DimTaxi\_New (
3. Taxi\_id INT,
4. Id VARCHAR(50),
5. Passenger\_count INT,
6. Trip\_distance FLOAT,
7. Fare\_amount FLOAT,
8. Trip\_duration INT,
9. Effective\_date DATETIME,
10. End\_date DATETIME,
11. Is\_current BIT
12. );
13. --- Insert value
14. INSERT INTO DimTaxi\_New (Id, Passenger\_count, Trip\_duration, Effective\_date, End\_date, Is\_current)
15. VALUES
16. ('id0000001', 2, 600, '2022-04-25', NULL, 1), --- maintain
17. ('id5000003', 4, 1200, '2022-04-23', NULL, 1), --- new
18. ('id5000005', 5, 900, '2022-04-22', NULL, 1), --- new
19. ('id0000008', 1, 450, '2022-04-24', NULL, 1), --- maintain
20. ('id5000008', 1, 1500, '2022-04-26', NULL, 1); --- new
21. --- Update measures
22. UPDATE DimTaxi\_New
23. SET
24. Trip\_distance = CASE
25. WHEN t.Trip\_duration IS NOT NULL THEN t.Trip\_duration \* 0.2
26. ELSE 0
27. END,
28. Fare\_amount = CASE
29. WHEN t.Trip\_duration IS NOT NULL THEN t.trip\_duration \* 0.05 + 10
30. ELSE 0
31. END
32. FROM DimTaxi\_New t

-- Update records that already exist in the table

UPDATE DimTaxi

SET Is\_current = 0, End\_date = GETDATE()

WHERE Id IN ('id0000001', 'id0000008') AND Is\_current = 1;

-- Insert records into the table

INSERT INTO DimTaxi (Id, Passenger\_count, Trip\_distance, Fare\_amount, Trip\_duration, Effective\_date, End\_date, Is\_current)

VALUES

('id5000003', 4, 240, 70, 1200, '2022-04-23', NULL, 1),

('id5000005', 5, 180, 55, 900, '2022-04-22', NULL, 1),

('id5000008', 1, 300, 85, 1500, '2022-04-26', NULL, 1),

('id0000001', 2, 120, 40, 600, '2022-04-25', NULL, 1), --- update

('id0000008', 1, 90, 32.5, 450, '2022-04-24', NULL, 1); --- update

-- Update the new records to be current

UPDATE DimTaxi

SET Is\_current = 1, End\_date = NULL

1. WHERE Id IN ('id5000003', 'id5000005', 'id5000008');

DROP TABLE DimTaxi\_New;

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

DimLocation: SCD Type 1

Change it directly.

**Part 10:** Outline one of the business questions you outlined in part 2 and answer it with an analytical query – it will be important to show complexity within the query (i.e. think about pivots, window functions, inline views, aggregates etc).

What is the most common pickup/ drop-off locations in NYC (Top 3) ?

WITH pickup\_counts AS (

SELECT

l.Longitude,

l.Latitude,

COUNT(DISTINCT ft.Taxi\_id) as pickup\_count

FROM

FactTrip ft

JOIN DimLocation l ON ft.Location\_id = l.Location\_id

WHERE

l.Location\_type = 'Taxi Pickup'

GROUP BY

l.Longitude,

l.Latitude

),

dropoff\_counts AS (

SELECT

l.Longitude,

l.Latitude,

COUNT(DISTINCT ft.Taxi\_id) as dropoff\_count

FROM

FactTrip ft

JOIN DimLocation l ON ft.Location\_id = l.Location\_id

WHERE

l.Location\_type = 'Taxi Dropoff'

GROUP BY

l.Longitude,

l.Latitude

),

pickup\_ranked AS (

SELECT

\*,

ROW\_NUMBER() OVER (ORDER BY pickup\_count DESC) AS rank

FROM

pickup\_counts

),

dropoff\_ranked AS (

SELECT

\*,

ROW\_NUMBER() OVER (ORDER BY dropoff\_count DESC) AS rank

FROM

dropoff\_counts

)

-- Top 3 most common pickup locations

SELECT

Longitude,

Latitude,

pickup\_count,

'Pickup' as type

FROM

pickup\_ranked

WHERE

rank <= 3

UNION ALL

-- Top 3 most common drop-off locations

SELECT

Longitude,

Latitude,

dropoff\_count,

'Dropoff' as type

FROM

dropoff\_ranked

WHERE

rank <= 3;

A screenshot of a computer

Description automatically generated

# Question 4

# Transform the latitude and Longitude into Borough and Neighborhood format

import pandas as pd

from opencage.geocoder import OpenCageGeocode

# Your API key from OpenCage

api\_key = '0827312559814ae1bd7b952dad51edf8'

geocoder = OpenCageGeocode(api\_key)

# Replace this with the DataFrame containing your coordinates

data = pd.DataFrame({'Longitude': [-73.976936,-73.972221,-73.993942,-73.989388,-73.996239,-73.991371], 'Latitude': [40.750000,40.750000,40.750000,40.750000,40.750000,40.750000]})

def reverse\_geocode(row):

result = geocoder.reverse\_geocode(row['Latitude'], row['Longitude'])

if result and result[0]:

components = result[0]['components']

borough = components.get('borough')

neighborhood = components.get('neighbourhood')

return pd.Series([borough, neighborhood], index=['Borough', 'Neighborhood'])

else:

return pd.Series([None, None], index=['Borough', 'Neighborhood'])

data[['Borough', 'Neighborhood']] = data.apply(reverse\_geocode, axis=1)

print(data)

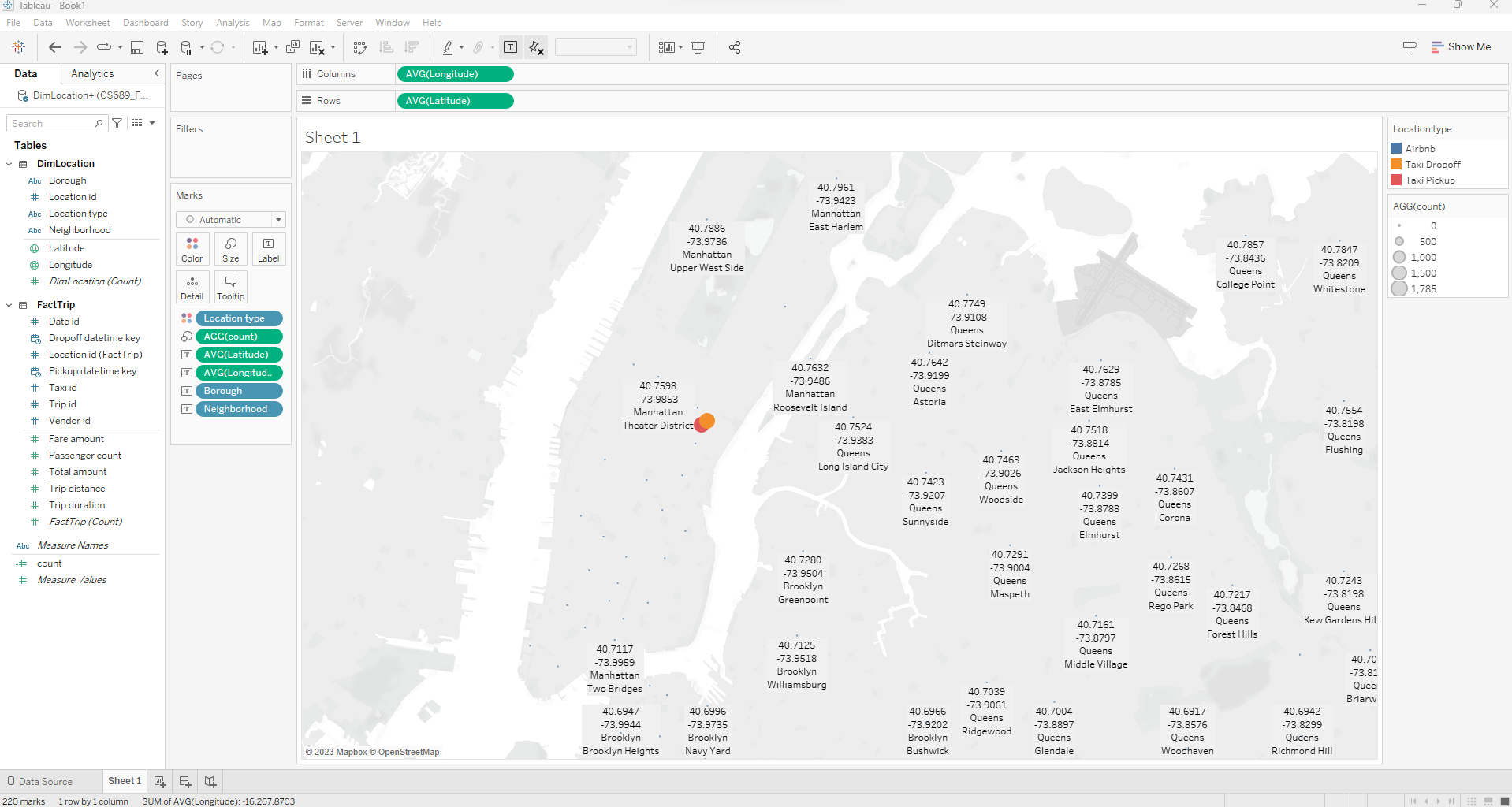
A screenshot of a computer code

Description automatically generated with medium confidence

**Part 11:** Outline one of the business questions you outlined in part 2 (can be same as part 10) and answer it with a Tableau or PowerBI visualization. In a single sentence explain why you selected this particular visualization to answer this question.

**A screenshot of a computer

Description automatically generated**

****

**For this question, using the geo chart would be better understand the physical location of pickup and drop-off location.**

If you update any of the previous parts and would like feedback for a specific part, please make a note:

Updates in PART 4, PART 5, and PART 6, PART 7, PART 8.

**Part 1**: Are you working on your own or with a partner? If with a partner provide their name. If on your own, just state that this is the case.

Individual work.

**Part 2**: Determine the project scope

* In a short paragraph, describe the topic you wish to explore – an update if any
* Update the five business questions that your data warehouse will answer.

The topic is about building a full-stack data warehouse with ETL pipelines, and visualization plots or queries based on the questions we want to answer.

In this project, I plan to use the datasets of NYC Taxi Trip Duration and NYC Airbnb Open Date. These two data are deeply relative to New York City’s tourism activity. The goal is to explore the relationship between taxi passengers, trip duration and Airbnb rentals in NYC and provide insights by using the data warehouse.

Updated business questions:

1. Does people who rent in high average rental rate region also pay high fare for their taxi rides?
2. What is the relation between Airbnb density and the taxi taking rate?
3. How does the average trip duration for taxi rides vary depending on the number of passengers and the pickup/ drop-off location in relation to popular Airbnb rental neighborhoods?
4. What is the most common pickup/ drop-off locations in NYC (Top 3) ?
5. How does the average fare and trip duration for taxi rides vary based on the distance between Airbnb rental locations and popular tourist attractions in the city?

**Part 3:** Data Sources

* Provide two data sources you will be using, for each data source list the number or columns and rows that are in each data source. Provide a header and first 5 rows from each source in separate file.

1. NYC Taxi Trip Duration

Train.csv: (columns: 11, rows: 1458644)

Background pattern

Description automatically generated

Table

Description automatically generated

Test.csv: (columns: 9, rows: 625134)

Table

Description automatically generated

1. New York City Airbnb Open Data

AB\_NYC\_2019.csv (columns: 16, rows: 48895)

A picture containing website

Description automatically generated

Table

Description automatically generated

* What is the URL or location of the data?

1. NYC Taxi Trip Duration: [**https://www.kaggle.com/c/nyc-taxi-trip-duration/data**](https://www.kaggle.com/c/nyc-taxi-trip-duration/data)
2. NYC Airbnb Open Data: [**https://www.kaggle.com/dgomonov/new-york-city-airbnb-open-data**](https://www.kaggle.com/dgomonov/new-york-city-airbnb-open-data)

* What information does this data provide that will help answer one or more of the above questions?

The latitude, longitude, and neighborhood in the NYC Airbnb data can let us know the location of the Airbnb. Drop-off longitude, latitude, and drop-off datetime in the NYC Taxi trip Duration can also let us know where the most common destination of the passenger is, and when do they arrive. According to the location, we might know whether there is relation between Airbnb and Taxi taking rate. We might also get to predict the level of convenience in the neighborhood.

* Do you see any issues in the data that will require transformation.

The length (amount) of rows in these two datasets has a huge difference. Therefore, we may need to sample the taxi dataset for better combination. Luckily, both datasets have latitude and longitude columns, which make the analysis more easily.

Also, in taxi dataset, train and test datasets have the different columns, which will need to be integrated.

**Part 4**: Dimensions - Review the data and the business questions from part 2.

* What fields (attributes) are in the data that will be used for the dimensions.

Based on the questions above, we can identify the attributes that can be used for the dimensions from both datasets. In the preliminary stage of selecting attributes, we will focus on

NYC Taxi Trip Duration:

* + pickup\_datetime
  + passenger\_count
  + pickup\_longitude
  + dropoff\_longitude
  + trip\_duration
  + fare\_amount
  + vendor\_id
  + store\_and\_fwd\_flag

NYC Airbnb Open Data:

* + longitude
  + latitude
  + host\_id
  + neighbourhood
  + neighbourhood\_group
  + room\_type
  + price
  + host\_name
  + calculated\_host\_listings\_count
  + name
  + last\_review
  + review\_per\_month
  + availability\_365
* Determine the dimension tables. There should be at least two non-date dimensions and one date dimension for each fact table.

Dimension table:

DimDate:

Date\_id (PK)

Hour

Day

Week

Month

Year

DateHour

DimLocation:

Location\_id (PK)

Location\_type

Neighborhood

Borough

Latitude

Longitude

DimTaxi:

Taxi\_id (PK)

Id (taxi\_id in taxi\_staging)

Passenger\_count

Trip\_distance

Trip\_duration

Fare\_amount

Effective\_date

End\_date

Is\_current

DimAirbnb\_Property:

Airbnb\_id (PK)

Leasing\_id (original Airbnb\_id)

Room\_name

Room\_type

Price

Neighborhood

Neighborhood\_group

Latitude

Longitude

Availability\_365

Last\_review

Price\_prev

Effective\_date

DimAirbnb\_Host:

Host\_id (PK)

Listing\_host\_id (original Host\_id)

Airbnb\_id (FK)

Host\_name

Host\_name\_prev

Effective\_date

Fact table:

FactTrip (Cumulative)

Trip\_id (PK)

Taxi\_id (FK)

Location\_id (FK)

Date\_id (FK)

Pickup\_datetime\_key (FK to DimDate)

Dropoff\_datetime\_key (FK to DimDate)

Passenger\_count

Trip\_distance

Trip\_duration

Fare\_amount

Total\_amount

Vendor\_id

FactAirbnb (Snapshot)

Id (PK)

Airbnb\_id (FK)

Date\_id (FK)

Host\_id (FK)

Location\_id (FK)

Room\_name

Room\_type

Neighborhood

Borough

Availability

Price

Revenue

Available\_rooms

Booked\_rooms

Occupancy\_rate

Average\_daily\_rate

FactTaxiMonthlySummary (Cumulative)

Sum\_id (PK)

Date\_id (FK)

Taxi\_id (FK)

Total\_passengers

Total\_trip\_duration

Total\_revenue

Monthly

* At least one (non-date) dimension in your design should have a hierarchy.

For the hierarchy, we design a parent-child relationship between boroughs and neighborhoods in DimLocation. The boroughs are the higher-level grouping and neighborhoods are the lower-level grouping.

(e.g., {Boroughs: Manhattan}, {Neighborhoods: Battery Park City})

Also, in DimDate, we have a hierarchy of time from Year, month, day to time.

* What are the attributes that will be tracked via slowly changing dimensions?

For DimAirbnb\_Property table, the price could be tracked by using SCD Type 3 with new column for the price to capture any changes over time.

Also, in the DimAirbnb\_Host table, we can use SCD Type 3 to capture the host name changes.

For DimTaxi table, we added effective date and end date as SCD Type 2 to record any changes in vendor id.

DimLocation using SCD Type 1 would be enough.

* What attributes within the dimensions will need transformation before they are loaded into the dimension, for example it could be to build consistency or any other issues? This is where for example you might build case statements in your code to handle various scenarios. Two to three examples showing some sample data and what you think the transformation will be during your ETL would be helpful here.

1. Date parsing: In the DimDate table, the pickup\_datetime or dropoff\_datetime attribute may need to be parsed into separate columns to aggregate data by different period.
2. Dealing with missing values in some tables can be beneficial to our analysis.

Graphical user interface, table

Description automatically generated Table

Description automatically generated with medium confidence

1. Name standardization: In the DimTaxi table, the vendor’s name attribute may need to be standardized to ensure consistency.
2. Address normalization: In the DimAirbnb table, the address attribute may need to be normalized to make sure the address format is consistency. Also, normalized address can help avoiding duplication. For this problem, I would like to try deriving a new column called pickup\_neighborhood based on the latitude and longitude.
3. In the DimAirbnb table, I found some values in the neighborhood column have the different format.
4. Date parsing: In the DimDate table, the pickup\_datetime or dropoff\_datetime attribute may need to be parsed into separate columns to aggregate data by different period.
5. Dealing with missing values in some tables can be beneficial to our analysis.

Graphical user interface, table

Description automatically generated Table

Description automatically generated with medium confidence

1. Name standardization: In the DimTaxi table, the vendor’s name attribute may need to be standardized to ensure consistency.
2. Address normalization: In the DimAirbnb table, the address attribute may need to be normalized to make sure the address format is consistency. Also, normalized address can help avoiding duplication. For this problem, I would like to try deriving a new column called pickup\_neighborhood based on the latitude and longitude.
3. In the DimAirbnb table, I found some values in the neighborhood column have the different format.

Table

Description automatically generated

**Part 5**: Facts – Review the data and the business questions from step 1.

* What measurements are in the data that will be used for the fact tables?

Trip\_duration

Trip\_distance

Total\_passengers

Total\_trip\_duration

Total\_revenue

Available\_rooms

Booked\_rooms

Average\_daily\_rate

Occupancy\_rate

* What measures will you be calculating (i.e. using an aggregate function, or some other transformation – recall as an example some of the aggregation you did in assignment 1A)

1. Average fare per pickup location

Average rental rate per pickup location

Correlation coefficient between average rental rate and average fare per pickup location

Code example:

SELECT

DimTaxi.vendor\_id,

AVG(FactTaxi.fare\_amount) AS avg\_fare,

AVG(FactTaxi.trip\_duration) AS avg\_duration

FROM

DimTaxi

INNER JOIN FactTaxi ON DimTaxi.taxi\_id = FactTaxi.taxi\_id

GROUP BY

DimTaxi.vendor\_id

1. Total number of Airbnb rentals per pickup location

Total number of taxi rides per pickup location

Code example:

SELECT

DimAirbnb.neighborhood,

COUNT(FactAirbnb.listing\_id) AS total\_listings,

AVG(FactAirbnb.price) AS avg\_rental\_rate

FROM

DimAirbnb

INNER JOIN FactAirbnb ON DimAirbnb.listing\_id = FactAirbnb.listing\_id

GROUP BY

DimAirbnb.neighborhood

1. Average trip duration per number of passengers

Average trip duration per pickup location and Airbnb rental neighborhood

1. Percentage of Airbnb rentals that occur at these pickup/ drop-off locations.
2. Average fare per distance between Airbnb location and popular tourist spots.

Average trip duration per distance between Airbnb location and popular tourist spots.

**Part 6**: Design – Create a Draw.io, Visio or Lucidchart diagram of your constellation data warehouse design.

A picture containing text, handwriting, post-it note, diagram

Description automatically generated

Let’s focus on the ETL Process.

Fist iteration of ETL: Take a part of your design and implement ETL. The scope at this point is limited and should include just two of your dimensions and one fact table. Using Python, SQL document and walk through the initial process of loading your data into staging (SQL staging or a data frame) doing some transformation and loading into your dimensions and a fact table.

**Part 7**: Provide code and screenshots of loading your data into staging/data frame\

*#!/usr/bin/env python*

*# coding: utf-8*

*# In[32]:*

**import** pandas **as** pd

**import** numpy **as** np

**import** pyodbc **as** odbc

*# In[69]:*

Airbnb\_df **=** pd.read\_csv('AB\_NYC\_2019.csv')

Airbnb\_df.head()

*# In[3]:*

Taxi\_df **=** pd.read\_csv('train.csv')

Taxi\_df.head()

*# In[4]:*

Taxi\_df2 **=** pd.read\_csv('test.csv')

Taxi\_df2.head()

*# In[6]:*

*# Combine the train and test df as Taxi\_df*

*# Add a new column to identify the source*

Taxi\_df['source'] **=** 'train'

Taxi\_df2['source'] **=** 'test'

*# Reset Index*

Taxi\_df2 **=** Taxi\_df2.reindex(columns**=**Taxi\_df.columns)

*# Concatenate train and test dataframe*

Taxi\_df **=** pd.concat([Taxi\_df, Taxi\_df2], ignore\_index**=True**)

*# Fill any missing value with NaN*

Taxi\_df['trip\_duration'] **=** Taxi\_df['trip\_duration'].fillna(**-**1)

*# Save the new df as CSV file*

Taxi\_df.to\_csv('NYC\_Taxi.csv', index**=False**)

*# In[7]:*

*# Check the new df*

Taxi\_df.head()

*# In[8]:*

*# Total rows in our target dfs*

Taxi\_total\_rows **=** Taxi\_df.shape[0]

Airbnb\_total\_rows **=** Airbnb\_df.shape[0]

print('Taxi total rows: ', Taxi\_total\_rows, '\nAirbnb total rows: ', Airbnb\_total\_rows)

*# In[11]:*

*# Count the total columns*

Taxi\_total\_cols **=** Taxi\_df.shape[1]

Airbnb\_total\_cols **=** Airbnb\_df.shape[1]

print('Taxi total columns: ', Taxi\_total\_cols, '\nAirbnb total columns: ', Airbnb\_total\_cols)

*# In[14]:*

*# Dtypes*

Airbnb\_df.dtypes

*# In[15]:*

Taxi\_df.dtypes

*# In[19]:*

*# PART 2 - Creating SCD Type 2 and 3 Dim and Key Maintenance*

*# dfs Cleaning*

Airbnb\_df.isnull().sum()

*# Duplication check*

duplicate\_counts **=** Airbnb\_df.groupby(['id','host\_id','host\_name']).size().reset\_index(name**=**'Count')

*# Filter the rows where the count is greater than 1 to find the combinations*

Airbnb\_duplicate **=** duplicate\_counts[duplicate\_counts['Count'] **>** 1]

print("Duplicate combinations of Columns in Airbnb:\n")

print(Airbnb\_duplicate['Count'].sum())

print('\n',Airbnb\_duplicate)

*# In[43]:*

*# Data normalization*

Airbnb\_df['host\_name'] **=** Airbnb\_df['host\_name'].str.slice(stop**=**255)

Airbnb\_df['host\_name'] **=** Airbnb\_df['host\_name'].str.encode('ascii', errors**=**'replace').str.decode('ascii')

Airbnb\_df['name'] **=** Airbnb\_df['name'].str.slice(stop**=**255)

Airbnb\_df['name'] **=** Airbnb\_df['name'].str.encode('ascii', errors**=**'replace').str.decode('ascii')

Airbnb\_df['last\_review'] **=** pd.to\_datetime(Airbnb\_df['last\_review'])

Airbnb\_df['reviews\_per\_month'] **=** Airbnb\_df['reviews\_per\_month'].fillna(0)

*# In[45]:*

*# Filter and load data*

Airbnb\_filtered **=** Airbnb\_df[['id','name','host\_id','host\_name','neighbourhood\_group','neighbourhood',

                            'room\_type','longitude','latitude','price','calculated\_host\_listings\_count',

                             'availability\_365','last\_review','reviews\_per\_month']]

Airbnb\_filtered.head()

*#Airbnb\_filtered.dtypes*

*# In[66]:*

*# Estiblish the connection*

conn **=** odbc.connect(Trusted\_Connection **=** 'YES',

                    Driver **=** '{ODBC Driver 17 for SQL Server}',

                    Server **=** 'DESKTOP-IAN\JLM\_SQLSERVER',

                    Database **=** 'CS689\_fINALpROJ')

print(conn)

*# In[67]:*

*# Create table*

cursor **=** conn.cursor()

cursor.execute("""

CREATE TABLE Airbnb\_staging (

    airbnb\_id INT PRIMARY KEY,

    name NVARCHAR(255),

    host\_id INT,

    host\_name NVARCHAR(255),

    neighborhood\_group NVARCHAR(255),

    neighborhood NVARCHAR(255),

    room\_type NVARCHAR(255),

    longitude FLOAT,

    latitude FLOAT,

    price INT,

    calculated\_host\_listings\_count INT,

    availability\_365 INT,

    last\_review DATETIME,

    reviews\_per\_month FLOAT

)

""")

conn.commit()

conn.close()

*# In[68]:*

*# Load data*

conn **=** odbc.connect(Trusted\_Connection **=** 'YES',

                    Driver **=** '{ODBC Driver 17 for SQL Server}',

                    Server **=** 'DESKTOP-IAN\JLM\_SQLSERVER',

                    Database **=** 'CS689\_fINALpROJ')

cursor **=** conn.cursor()

**for** row **in** Airbnb\_filtered.itertuples():

**try**:

        cursor.execute("""

            INSERT INTO Airbnb\_staging (

                airbnb\_id,

                name,

                host\_id,

                host\_name,

                neighborhood\_group,

                neighborhood,

                room\_type,

                longitude,

                latitude,

                price,

                calculated\_host\_listings\_count,

                availability\_365,

                last\_review,

                reviews\_per\_month

            )

            VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)

            """,

            row.id,

            row.name **if** **not** pd.isna(row.name) **else** 'Unknown',

            row.host\_id,

            row.host\_name **if** **not** pd.isna(row.host\_name) **else** 'Unknown',

            row.neighbourhood\_group,

            row.neighbourhood,

            row.room\_type,

            float(row.longitude),

            float(row.latitude),

            row.price,

            row.calculated\_host\_listings\_count,

            row.availability\_365,

            row.last\_review **if** **not** pd.isna(row.last\_review) **else** **None**,

            row.reviews\_per\_month **if** **not** pd.isna(row.reviews\_per\_month) **else** **None**

        )

**except** odbc.ProgrammingError **as** e:

        print(**f**"Error occurred for row: {row}")

        print(**f**"Error message: {e}")

**break**

conn.commit()

conn.close()

# load Taxi\_df to Taxi\_staging

# dfs Cleaning

Taxi\_df.isnull().sum()

Taxi\_df\_clean = Taxi\_df[~Taxi\_df.isnull().any(axis=1)]

Taxi\_df\_clean.isnull().sum()

duplicate\_counts = Taxi\_df\_clean.groupby(['id','pickup\_longitude','pickup\_latitude','dropoff\_longitude','dropoff\_latitude']).size().reset\_index(name='Count')

Taxi\_duplicate = duplicate\_counts[duplicate\_counts['Count'] > 1]

print("Duplicate combinations of Columns in Taxi:\n")

print(Taxi\_duplicate['Count'].sum())

print('\n',Taxi\_duplicate)

# Data normalization

Taxi\_df\_clean = Taxi\_df\_clean.copy()

# Transform into datetime data type

Taxi\_df\_clean.loc[:, 'pickup\_datetime'] = pd.to\_datetime(Taxi\_df\_clean['pickup\_datetime'])

Taxi\_df\_clean.loc[:, 'dropoff\_datetime'] = pd.to\_datetime(Taxi\_df\_clean['dropoff\_datetime'])

Taxi\_df\_clean.dtypes

# Filter data

Taxi\_filtered = Taxi\_df\_clean[['id','vendor\_id','pickup\_datetime','dropoff\_datetime',

'passenger\_count','pickup\_longitude','pickup\_latitude',

'dropoff\_longitude','dropoff\_latitude','store\_and\_fwd\_flag',

'trip\_duration']]

Taxi\_filtered.head()

# Transform the latitude and Longitude into Borough and Neighborhood format

!pip install geopy

from geopy.geocoders import Nominatim

from geopy.exc import GeocoderTimedOut

import time

def get\_borough\_neighborhood(latitude, longitude):

geolocator = Nominatim()

location = geolocator.reverse(f"{latitude}, {longitude}")

address = location.raw['address']

borough = address.get('county')

neighborhood = address.get('neighbourhood')

return borough, neighborhood

# set start index and retry limit

start\_index = 0

retry\_limit = 3

# loop through all records

for i in range(start\_index, len(Taxi\_filtered)):

row = Taxi\_filtered.iloc[i]

# retry up to retry\_limit times if timed out

for j in range(retry\_limit):

try:

# get pickup and dropoff borough and neighborhood

pickup\_borough, pickup\_neighborhood = get\_borough\_neighborhood(row['pickup\_latitude'], row['pickup\_longitude'])

dropoff\_borough, dropoff\_neighborhood = get\_borough\_neighborhood(row['dropoff\_latitude'], row['dropoff\_longitude'])

# assign values to dataframe

Taxi\_filtered.at[i, 'pickup\_borough'] = pickup\_borough

Taxi\_filtered.at[i, 'pickup\_neighborhood'] = pickup\_neighborhood

Taxi\_filtered.at[i, 'dropoff\_borough'] = dropoff\_borough

Taxi\_filtered.at[i, 'dropoff\_neighborhood'] = dropoff\_neighborhood

# print progress every 1000 records

if i % 1000 == 0:

print(f"Processed {i} records")

# break out of retry loop if successful

break

# handle timed out exception

except geopy.exc.GeocoderTimedOut:

print(f"Timed out, retrying {j+1}/{retry\_limit}")

time.sleep(1)

# save last processed index to a file or a database

with open('last\_processed\_index.txt', 'w') as f:

f.write(str(i))

# Estiblish the connection

conn = odbc.connect(Trusted\_Connection = 'YES',

Driver = '{ODBC Driver 17 for SQL Server}',

Server = 'DESKTOP-IAN\JLM\_SQLSERVER',

Database = 'CS689\_FinalProj')

print(conn)

# Create Taxi\_staging table

cursor = conn.cursor()

cursor.execute("""

CREATE TABLE Taxi\_staging (

taxi\_id NVARCHAR(255) PRIMARY KEY,

vendor\_id INT,

pickup\_datetime DATETIME,

dropoff\_datetime DATETIME,

passenger\_count FLOAT,

pickup\_longitude FLOAT,

pickup\_latitude FLOAT,

dropoff\_longitude FLOAT,

dropoff\_latitude FLOAT,

store\_and\_fwd\_flag NVARCHAR(255),

trip\_duration FLOAT,

)

""")

conn.commit()

conn.close()

# load data

conn = odbc.connect(Trusted\_Connection = 'YES',

Driver = '{ODBC Driver 17 for SQL Server}',

Server = 'DESKTOP-IAN\JLM\_SQLSERVER',

Database = 'CS689\_FinalProj')

cursor = conn.cursor()

for row in Taxi\_filtered.itertuples():

try:

cursor.execute("""

INSERT INTO Taxi\_staging (taxi\_id, vendor\_id, pickup\_datetime, dropoff\_datetime,

passenger\_count, pickup\_longitude, pickup\_latitude,

dropoff\_longitude, dropoff\_latitude, store\_and\_fwd\_flag, trip\_duration

)

VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)

""",

row.id,

row.vendor\_id,

row.pickup\_datetime,

row.dropoff\_datetime,

row.passenger\_count,

row.pickup\_longitude,

row.pickup\_latitude,

row.dropoff\_longitude,

row.dropoff\_latitude,

row.store\_and\_fwd\_flag,

row.trip\_duration,

)

except odbc.ProgrammingError as e:

print(f"Error occurred for row: {row}")

print(f"Error message: {e}")

break

conn.commit()

conn.close()

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated with medium confidence

A screenshot of a computer program

Description automatically generated with medium confidence

SELECT \* FROM Airbnb\_staging;

A screenshot of a computer

Description automatically generated

**Part 8:** Provide code and screenshots of transforming the data. Perhaps you are adjusting for consistency of data or calculating aggregates.

In the Python part, we already have filtered some of the data in Airbnb\_df, such as NA value check and columns’ normalization, such as transform the data type of last\_review into dattime format and so on.

A screenshot of a computer code

Description automatically generated with low confidence

A screen shot of a computer program

Description automatically generated with medium confidence

DELETE FROM Airbnb\_staging

WHERE last\_review IS NULL;

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

Since last\_review will become an important parameter for us to create the measures between DimDate and FactAirbnb, so we dropped all the NULL values in last\_review.

Before dropped: 48895 rows.

After dropped: 38843 rows.

**Part 9:** Provide code and screenshots of loading the data into the two dimensions and the fact. At this time, you do not need to worry about maintenance of slowly changing dimensions, the focus is on the initial data load. If you are loading into SCD2 or SCD3, make sure to show the SCD maintenance attributes populated.

-- Create DimAirbnb\_Property

CREATE TABLE DimAirbnb\_Property (

Airbnb\_id INT IDENTITY(1,1) PRIMARY KEY,

Leasing\_id INT,

Room\_name NVARCHAR(255),

Room\_type NVARCHAR(50),

Price INT,

Neighborhood NVARCHAR(255),

Neighborhood\_group NVARCHAR(255),

Latitude DECIMAL(9,6),

Longitude DECIMAL(9,6),

Availability\_365 INT,

Price\_prev INT

);

A screenshot of a computer

Description automatically generated

-- Insert data into DimAirbnb\_Property

INSERT INTO DimAirbnb\_Property (Airbnb\_id, Room\_name, Room\_type, Price, Neighborhood, Neighborhood\_group, Latitude, Longitude, Availability\_365)

SELECT airbnb\_id, name, room\_type, price, neighborhood, neighborhood\_group, latitude, longitude, availability\_365

FROM Airbnb\_staging;

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

-- Create DimAirbnb\_Host

CREATE TABLE DimAirbnb\_Host (

Host\_id INT IDENTITY(1,1) PRIMARY KEY,

Listing\_host\_id INT,

Host\_name NVARCHAR(255),

Host\_name\_prev NVARCHAR(255)

);

A screenshot of a computer screen

Description automatically generated with medium confidence

-- Insert data into DimAirbnb\_Host

INSERT INTO DimAirbnb\_Host (Listing\_host\_id, Host\_name)

SELECT DISTINCT host\_id, host\_name

FROM Airbnb\_staging;

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

CREATE TABLE DimDate (

Date\_id INT IDENTITY(1,1) PRIMARY KEY,

DateHour DATETIME NOT NULL,

Hour INT NOT NULL,

Day INT NOT NULL,

Month INT NOT NULL,

Year INT NOT NULL,

Week INT NOT NULL

);

A screenshot of a computer

Description automatically generated with medium confidence

WITH DateRange (DateHour) AS (

SELECT CAST('2010-01-01 00:00:00' AS DATETIME)

UNION ALL

SELECT DATEADD(hour, 1, DateHour)

FROM DateRange

WHERE DateHour < '2023-12-31 23:00:00'

)

INSERT INTO DimDate (DateHour, Hour, Day, Month, Year, Week)

SELECT

DateHour,

DATEPART(hour, DateHour) AS Hour,

DAY(DateHour) AS Day,

MONTH(DateHour) AS Month,

YEAR(DateHour) AS Year,

DATEPART(wk, DateHour) AS Week

FROM DateRange

OPTION (MAXRECURSION 0);

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

-- Create FactAirbnb

CREATE TABLE FactAirbnb (

Id INT IDENTITY(1,1) PRIMARY KEY NOT NULL,

Airbnb\_id INT NOT NULL FOREIGN KEY REFERENCES DimAirbnb\_Property(Airbnb\_id),

Date\_id INT NOT NULL FOREIGN KEY REFERENCES DimDate(Date\_id),

Host\_id INT NOT NULL FOREIGN KEY REFERENCES DimAirbnb\_Host(Host\_id),

Room\_name VARCHAR(300),

Room\_type VARCHAR(50),

Neighborhood VARCHAR(100),

Borough VARCHAR(50),

Availability INT,

Price FLOAT,

Revenue FLOAT,

Available\_rooms INT,

Booked\_rooms INT,

Occupancy\_rate FLOAT,

Average\_daily\_rate FLOAT

);

);

-- Insert data into FactAirbnb

INSERT INTO FactAirbnb (Airbnb\_id, Date\_id, Host\_id, Room\_name, Room\_type, Neighborhood, Borough, Availability, Price, Revenue, Occupancy\_rate, Average\_daily\_rate)

SELECT

p.Airbnb\_id,

d.Date\_id,

h.Host\_id,

p.Room\_name,

p.Room\_type,

p.Neighborhood,

p.Neighborhood\_group,

p.Availability\_365,

p.Price,

p.Price \* p.Availability\_365 AS Revenue,

NULL AS Occupancy\_rate,

NULL AS Average\_daily\_rate

FROM

DimAirbnb\_Property p

INNER JOIN DimDate d ON CAST(p.last\_review AS DATE) = CAST(d.DateHour AS DATE)

INNER JOIN DimAirbnb\_Host h ON p.Leasing\_id = h.airbnb\_id;

UPDATE FactAirbnb

SET

available\_rooms = Availability / 365,

booked\_rooms = (365 - Availability) / 365

UPDATE FactAirbnb

SET Occupancy\_rate = CASE WHEN Availability > 0 THEN (365-Availability)/365 ELSE 0 END,

Average\_daily\_rate = CASE WHEN Availability > 0 THEN Revenue/Availability ELSE 0 END;

ALTER TABLE FactAirbnb

ADD Location\_id INT;

UPDATE FactAirbnb

SET location\_id = (

SELECT TOP 1 location\_id

FROM DimLocation

WHERE Location\_type = 'Airbnb'

AND Neighborhood = FactAirbnb.Neighborhood

AND Borough = FactAirbnb.Borough

);

DimLocation:

CREATE TABLE DimLocation (

Location\_id INT IDENTITY(1,1) PRIMARY KEY,

Location\_type VARCHAR(50),

Neighborhood VARCHAR(255),

Borough VARCHAR(255),

Longitude DECIMAL(9, 6),

Latitude DECIMAL(9, 6)

);

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

INSERT INTO DimLocation (Latitude, Longitude, Location\_type)

SELECT DISTINCT

dropoff\_latitude,

dropoff\_longitude,

'Taxi Dropoff'

FROM Taxi\_staging;

A screenshot of a computer

Description automatically generated with medium confidence

UPDATE FactAirbnb

SET

Availability = p.availability\_365 / 365.0,

Revenue = p.availability\_365 \* p.price,

Available\_rooms = p.availability\_365,

Booked\_rooms = 365 - p.availability\_365,

Occupancy\_rate = (365 - p.availability\_365) / 365.0,

Average\_daily\_rate = (

SELECT AVG(price)

FROM DimAirbnb\_Property

WHERE Airbnb\_id = FactAirbnb.Id

)

FROM DimAirbnb\_Property p

WHERE p.Airbnb\_id = FactAirbnb.Id;

A screenshot of a computer

Description automatically generated with medium confidence

CREATE TABLE DimTaxi (

Taxi\_id INT IDENTITY(1,1) NOT NULL,

Id NVARCHAR(255),

Pickup\_datetime DATETIME,

Dropoff\_datetime DATETIME,

Passenger\_count INT,

Trip\_distance FLOAT,

pickup\_latitude FLOAT,

pickup\_longitude FLOAT,

dropoff\_latitude FLOAT,

dropoff\_longitude FLOAT,

Fare\_amount FLOAT,

Effective\_date DATE,

End\_date DATE,

Is\_current BIT

);

A screenshot of a computer

Description automatically generated with medium confidence

INSERT INTO DimTaxi (

Id,

Pickup\_datetime,

Dropoff\_datetime,

Passenger\_count,

Effective\_date,

End\_date,

Is\_current,

pickup\_latitude,

pickup\_longitude,

dropoff\_latitude,

dropoff\_longitude

)

SELECT

taxi\_id,

Pickup\_datetime,

Dropoff\_datetime,

Passenger\_count,

CAST(GETDATE() AS DATE),

NULL,

1,

pickup\_latitude,

pickup\_longitude,

dropoff\_latitude,

dropoff\_longitude

FROM

Taxi\_staging;

A screenshot of a computer

Description automatically generated

CREATE TABLE FactTaxiMonthlySummary (

Sum\_id INT IDENTITY(1,1) PRIMARY KEY NOT NULL,

Date\_id INT NOT NULL,

Taxi\_id INT NOT NULL,

Total\_passengers INT,

Total\_trip\_duration FLOAT,

Total\_revenue FLOAT,

Monthly INT,

Vendor\_id INT

);

A screenshot of a computer

Description automatically generated with medium confidence

CREATE TABLE FactTrip (

Trip\_id INT IDENTITY(1,1) PRIMARY KEY NOT NULL,

Taxi\_id INT NOT NULL,

Location\_id INT NOT NULL,

Date\_id INT NOT NULL,

Pickup\_datetime\_key INT,

Dropoff\_datetime\_key INT,

Pickup\_location INT,

Dropoff\_location INT,

Passenger\_count INT,

Trip\_distance FLOAT,

Trip\_duration INT,

Fare\_amount FLOAT,

Total\_amount FLOAT

);

UPDATE DimTaxi

SET Trip\_duration =

CASE

WHEN Taxi\_staging.trip\_duration IS NOT NULL THEN Taxi\_staging.Trip\_duration

ELSE DATEDIFF(second, pickup\_datetime, dropoff\_datetime)

END

FROM DimTaxi

JOIN Taxi\_staging ON DimTaxi.Id = Taxi\_staging.taxi\_id;

A screenshot of a computer

Description automatically generated with medium confidence

UPDATE DimTaxi

SET

trip\_distance = CASE

WHEN ts.Trip\_duration IS NOT NULL THEN ts.trip\_duration \* 0.2

ELSE 0

END,

fare\_amount = CASE

WHEN ts.Trip\_duration IS NOT NULL THEN ts.trip\_duration \* 0.05 + 10

ELSE 0

END

FROM DimTaxi t

JOIN taxi\_staging ts ON t.Id = ts.taxi\_id;

A screenshot of a computer

Description automatically generated

ALTER TABLE Taxi\_staging ADD taxi\_datetime DATETIME;

UPDATE Taxi\_staging

SET taxi\_datetime = DATEADD(HOUR, DATEDIFF(HOUR, 0, dropoff\_datetime), 0);

A screenshot of a computer

Description automatically generated

SELECT

d.Date\_id,

tx.Taxi\_id,

SUM(tx.Passenger\_count) AS Total\_passengers,

SUM(tx.Trip\_duration) AS Total\_trip\_duration,

SUM(tx.Fare\_amount) AS Total\_revenue,

FORMAT(d.DateHour, 'yyyy-MM') AS Monthly

FROM

DimTaxi tx

JOIN Taxi\_staging t ON t.taxi\_id = tx.Id

JOIN DimDate d ON CAST(t.taxi\_datetime AS DATE) = CAST(d.DateHour AS DATE)

GROUP BY

d.Date\_id,

tx.Taxi\_id,

FORMAT(d.DateHour, 'yyyy-MM');

ALTER TABLE FactTaxiMonthlySummary

ADD Monthly NVARCHAR(255);

A screenshot of a computer

Description automatically generated with medium confidence

INSERT INTO FactTaxiMonthlySummary (Date\_id, Taxi\_id, Total\_passengers, Total\_trip\_duration, Total\_revenue, Monthly)

SELECT

d.Date\_id,

tx.Taxi\_id,

SUM(tx.Passenger\_count) AS Total\_passengers,

SUM(tx.Trip\_duration) AS Total\_trip\_duration,

SUM(tx.Fare\_amount) AS Total\_revenue,

FORMAT(d.DateHour, 'yyyy-MM') AS Monthly

FROM

DimTaxi tx

JOIN Taxi\_staging t ON t.taxi\_id = tx.Id

JOIN DimDate d ON CAST(t.taxi\_datetime AS DATE) = CAST(d.DateHour AS DATE)

GROUP BY

d.Date\_id,

tx.Taxi\_id,

FORMAT(d.DateHour, 'yyyy-MM');

ALTER TABLE FactTrip

ALTER COLUMN Pickup\_datetime\_key DATETIME;

ALTER TABLE FactTrip

ALTER COLUMN Dropoff\_datetime\_key DATETIME;

A screenshot of a computer

Description automatically generated

INSERT INTO FactTrip (Taxi\_id, Location\_id, Date\_id, Pickup\_datetime\_key, Dropoff\_datetime\_key, Passenger\_count, Trip\_distance, Trip\_duration, Fare\_amount, Vendor\_id)

SELECT DISTINCT

tx.Taxi\_id,

l.Location\_id,

d.Date\_id,

ts.pickup\_datetime,

ts.dropoff\_datetime,

tx.Passenger\_count,

tx.Trip\_distance,

tx.Trip\_duration,

tx.Fare\_amount,

ts.vendor\_id

FROM

DimTaxi tx

JOIN Taxi\_staging ts ON ts.taxi\_id = tx.Id

JOIN (

SELECT l1.Location\_id, l1.latitude

FROM DimLocation l1

JOIN Taxi\_staging ts1 ON ts1.dropoff\_latitude = l1.latitude AND l1.Location\_type = 'Taxi Dropoff'

UNION

SELECT l2.Location\_id, l2.latitude

FROM DimLocation l2

JOIN Taxi\_staging ts2 ON ts2.pickup\_latitude = l2.latitude AND l2.Location\_type = 'Taxi Pickup'

) AS l ON l.latitude = ts.dropoff\_latitude OR l.latitude = ts.pickup\_latitude

JOIN DimDate d ON CAST(ts.taxi\_datetime AS DATE) = CAST(d.DateHour AS DATE)

UNION

SELECT DISTINCT

tx.Taxi\_id,

l.Location\_id,

d.Date\_id,

ts.pickup\_datetime,

ts.dropoff\_datetime,

tx.Passenger\_count,

tx.Trip\_distance,

tx.Trip\_duration,

tx.Fare\_amount,

ts.vendor\_id

FROM

DimTaxi tx

JOIN Taxi\_staging ts ON ts.taxi\_id = tx.Id

JOIN (

SELECT l1.Location\_id, l1.longitude

FROM DimLocation l1

JOIN Taxi\_staging ts1 ON ts1.dropoff\_longitude = l1.longitude AND l1.Location\_type = 'Taxi Dropoff'

UNION

SELECT l2.Location\_id, l2.longitude

FROM DimLocation l2

JOIN Taxi\_staging ts2 ON ts2.pickup\_longitude = l2.longitude AND l2.Location\_type = 'Taxi Pickup'

) AS l ON l.longitude = ts.dropoff\_longitude OR l.longitude = ts.pickup\_longitude

JOIN DimDate d ON CAST(ts.taxi\_datetime AS DATE) = CAST(d.DateHour AS DATE)

**A screenshot of a computer

Description automatically generated with medium confidence**

UPDATE FactTrip

SET Total\_amount = subquery.Total\_amount

FROM (

SELECT Taxi\_id, SUM(Fare\_amount) as Total\_amount

FROM FactTrip

GROUP BY Taxi\_id

) AS subquery

WHERE FactTrip.Taxi\_id = subquery.Taxi\_id;

Use the **Ask the Teaching Team Discussion Forum** if you have any questions regarding the how to approach this assignment.

Save your assignment as ***lastnameFirstname\_ProjectUpdate2.docx*** and submit it in the *Assignments* section of the course.

For help uploading files please refer to the *Technical Support* page in the syllabus.

Project scoping is graded based on the following:

1 – On track, 0-Off track, .5 – partially on track