Business Questions

1. Does people who rent in high average rental rate region also pay high fare for their taxi rides?

Failed method (SSMS):

(SELECT TOP 5

dL.neighborhood,

COUNT(aB.Airbnb\_id) AS Airbnb\_listings\_count

FROM

FactAirbnb aB

JOIN DimLocation dL ON aB.Location\_id = dL.Location\_id

GROUP BY

dL.neighborhood

ORDER BY

Airbnb\_listings\_count DESC;

---USE [CS689\_FinalProj];

CREATE OR ALTER FUNCTION is\_within\_neighborhood\_bounds(

@pickup\_lat FLOAT,

@pickup\_lon FLOAT,

@neighborhood\_lat FLOAT,

@neighborhood\_lon FLOAT,

@threshold FLOAT

)

RETURNS BIT

AS

BEGIN

DECLARE @result BIT = 0;

IF (ABS(@pickup\_lat - @neighborhood\_lat) <= @threshold) AND

(ABS(@pickup\_lon - @neighborhood\_lon) <= @threshold)

BEGIN

SET @result = 1;

END

RETURN @result;

END;

WITH TopNeighborhoods AS (

SELECT TOP 5

dL.neighborhood,

COUNT(aB.Airbnb\_id) AS Airbnb\_listings\_count,

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

FROM

FactAirbnb aB

JOIN DimLocation dL ON aB.Location\_id = dL.Location\_id

WHERE

dL.location\_type = 'Airbnb'

GROUP BY

dL.neighborhood

ORDER BY

Airbnb\_listings\_count DESC

),

TaxiRidesInNeighborhood AS (

SELECT

t.Trip\_id,

t.Fare\_amount,

l.neighborhood

FROM

FactTrip t

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

WHERE

l.location\_type = 'Taxi Pickup'

AND EXISTS (

SELECT 1

FROM TopNeighborhoods tn

WHERE

dbo.is\_within\_neighborhood\_bounds(l.latitude, l.longitude, tn.latitude, tn.longitude, 1.11) = 1

AND tn.neighborhood = l.neighborhood

)

)

SELECT

n.neighborhood,

n.Airbnb\_listings\_count,

n.avg\_rental\_rate,

AVG(t.Fare\_amount) AS avg\_taxi\_fare

FROM

TopNeighborhoods n

JOIN TaxiRidesInNeighborhood t ON n.neighborhood = t.neighborhood

GROUP BY

n.neighborhood,

n.Airbnb\_listings\_count,

n.avg\_rental\_rate

ORDER BY

n.Airbnb\_listings\_count DESC;)

Success method (Python):

import pandas as pd

import pyodbc as odbc

from haversine import haversine

# Connect to the database and run the SQL query

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

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

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

Database = 'CS689\_FinalProj')

query = """

SELECT TOP 5

dL.neighborhood,

COUNT(aB.Airbnb\_id) AS Airbnb\_listings\_count,

AVG(dAP.Latitude) AS avg\_latitude,

AVG(dAP.Longitude) AS avg\_longitude

FROM

FactAirbnb aB

JOIN DimLocation dL ON aB.Location\_id = dL.Location\_id

JOIN DimAirbnb\_Property dAP ON aB.Airbnb\_id = dAP.Airbnb\_id

GROUP BY

dL.neighborhood

ORDER BY

Airbnb\_listings\_count DESC;

"""

top\_neighborhoods = pd.read\_sql(query, conn)

# Get taxi ride data

taxi\_query = """

SELECT

t.Trip\_id,

t.Fare\_amount,

l.neighborhood,

l.latitude,

l.longitude

FROM

FactTrip t

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

WHERE

l.location\_type = 'Taxi Pickup';

"""

taxi\_data = pd.read\_sql(taxi\_query, conn)

conn.close()

# Define the is\_within\_neighborhood\_bounds function

def is\_within\_neighborhood\_bounds(pickup\_lat, pickup\_lon, neighborhood\_lat, neighborhood\_lon, threshold\_km):

distance = haversine((pickup\_lat, pickup\_lon), (neighborhood\_lat, neighborhood\_lon), unit='km')

return distance <= threshold\_km

# Calculate the average taxi fare for each neighborhood

results = []

for index, row in top\_neighborhoods.iterrows():

neighborhood = row['neighborhood']

avg\_latitude = row['avg\_latitude']

avg\_longitude = row['avg\_longitude']

# Filter taxi rides based on the is\_within\_neighborhood\_bounds function

taxi\_neighborhood = taxi\_data[taxi\_data.apply(lambda x: is\_within\_neighborhood\_bounds(x['latitude'], x['longitude'],

avg\_latitude, avg\_longitude, 2), axis=1)]

# Calculate the average taxi fare for the neighborhood

avg\_taxi\_fare = taxi\_neighborhood['Fare\_amount'].mean()

results.append({

'neighborhood': neighborhood,

'avg\_rental\_rate': row['Airbnb\_listings\_count'],

'avg\_taxi\_fare': avg\_taxi\_fare

})

# Convert the results to a DataFrame and display it

results\_df2 = pd.DataFrame(results)

print(results\_df2)

A screenshot of a computer

Description automatically generated

A picture containing text, screenshot, font, line

Description automatically generated

A screenshot of a graph

Description automatically generated with medium confidence

According to the graphs above, we can observe that except for Williamsburg with the highest Airbnb rental rate and not a cheap fare for taxi. But, even in some neighborhoods without high rental rate, the taxi fare could also be very expensive, such as Bushwick.

One possible explanation for this pattern is that the more affordable areas are located farther from NYC's downtown, resulting in higher taxi fares due to the increased distance.

1. What types of Airbnb rooms generate the highest revenue in the top 5 neighborhoods with the highest average rental rates?

The SQL code first calculates the average rental rate for each neighborhood and selects the top 3 neighborhoods with the highest average rental rates.

It then joins the resulting top neighborhoods with the FactAirbnb table to select the Airbnb listings in those neighborhoods.

Finally, it aggregates the data by room type and calculates the average revenue generated by each room type in the selected neighborhoods.

WITH avg\_rental\_rate AS (

SELECT

dL.neighborhood,

AVG(aB.Price) AS avg\_rental\_rate

FROM

FactAirbnb aB

JOIN DimLocation dL ON aB.Location\_id = dL.Location\_id

GROUP BY

dL.neighborhood

),

top\_rental\_neighborhoods AS (

SELECT TOP 3

neighborhood

FROM

avg\_rental\_rate

ORDER BY

avg\_rental\_rate DESC

),

top\_rental\_rooms AS (

SELECT

dL.neighborhood,

aB.Room\_type,

aB.Revenue

FROM

FactAirbnb aB

JOIN DimLocation dL ON aB.Location\_id = dL.Location\_id

JOIN top\_rental\_neighborhoods tRN ON dL.neighborhood = tRN.neighborhood

)

SELECT

neighborhood,

Room\_type,

AVG(Revenue) AS avg\_revenue

FROM

top\_rental\_rooms

GROUP BY

neighborhood,

Room\_type

ORDER BY

neighborhood,

avg\_revenue DESC;

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Entire home/apt has the majority of the Airbnb rental room type, especially in Flatiron District and Tribeca.

1. What is the most common pickup/ drop-off locations for taxis in NYC ?

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

# 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

**A screenshot of a computer

Description automatically generated**

**A screenshot of a map

Description automatically generated with medium confidence**

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

1. What is the relationship between Airbnb density and the taxi taking rate?

# Connect to the database and run the SQL query

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

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

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

Database = 'CS689\_FinalProj')

query2 = """

SELECT

dL.neighborhood,

COUNT(aB.Airbnb\_id) AS Airbnb\_listings\_count,

AVG(dAP.Latitude) AS avg\_latitude,

AVG(dAP.Longitude) AS avg\_longitude

FROM

FactAirbnb aB

JOIN DimLocation dL ON aB.Location\_id = dL.Location\_id

JOIN DimAirbnb\_Property dAP ON aB.Airbnb\_id = dAP.Airbnb\_id

GROUP BY

dL.neighborhood;

"""

neighborhoods = pd.read\_sql(query, conn)

# Get taxi ride data

taxi\_query2 = """

SELECT

t.Trip\_id,

l.neighborhood,

l.latitude,

l.longitude

FROM

FactTrip t

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

WHERE

l.location\_type = 'Taxi Pickup';

"""

taxi\_data2 = pd.read\_sql(taxi\_query2, conn)

conn.close()

# Calculate Airbnb density and taxi taking rate for each neighborhood

results = []

for index, row in neighborhoods.iterrows():

neighborhood = row['neighborhood']

avg\_latitude = row['avg\_latitude']

avg\_longitude = row['avg\_longitude']

# Filter Airbnb listings and taxi rides based on the is\_within\_neighborhood\_bounds function

airbnb\_neighborhood = neighborhoods[neighborhoods.apply(lambda x: is\_within\_neighborhood\_bounds(x['avg\_latitude'], x['avg\_longitude'], avg\_latitude, avg\_longitude, 2), axis=1)]

taxi\_neighborhood = taxi\_data[taxi\_data.apply(lambda x: is\_within\_neighborhood\_bounds(x['latitude'], x['longitude'], avg\_latitude, avg\_longitude, 2), axis=1)]

# Calculate Airbnb density and taxi taking rate

area = 3.14159 \* (2\*\*2) # Area of the neighborhood (circle with radius 2 km)

airbnb\_density = len(airbnb\_neighborhood) / area

taxi\_taking\_rate = len(taxi\_neighborhood) / area

results.append({

'neighborhood': neighborhood,

'airbnb\_density': airbnb\_density,

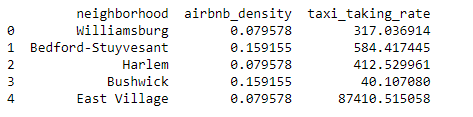
'taxi\_taking\_rate': taxi\_taking\_rate

})

# Convert the results to a DataFrame and display it

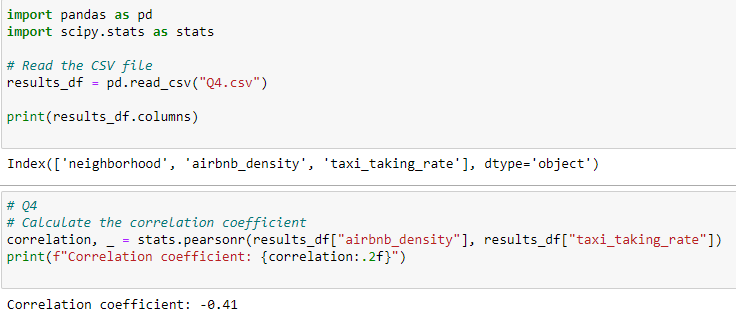
results\_df3 = pd.DataFrame(results)

print(results\_df3)



A screenshot of a graph

Description automatically generated with low confidence



Since we have a negative correlation between Airbnb density and taxi taking rate, -0.41, we can conclude that these two variables are not have a significant relationship.

1. What are the top months with the highest month-over-month revenue growth rate for the NYC taxi industry compared to the last year?

WITH monthly\_revenue\_duration AS (

SELECT

FORMAT(DateHour, 'yyyy') AS year,

FORMAT(DateHour, 'MM') AS month,

SUM(Total\_revenue) AS revenue,

SUM(Total\_trip\_duration) / 3600.0 AS duration\_hours

FROM

FactTaxiMonthlySummary ft

JOIN DimDate d ON ft.Date\_id = d.Date\_id

GROUP BY

FORMAT(DateHour, 'yyyy'),

FORMAT(DateHour, 'MM')

),

revenue\_growth AS (

SELECT

month,

year,

revenue,

duration\_hours,

LAG(revenue) OVER (ORDER BY year, month) AS prev\_revenue,

LAG(duration\_hours) OVER (ORDER BY year, month) AS prev\_duration\_hours,

(revenue - LAG(revenue) OVER (ORDER BY year, month)) / LAG(revenue) OVER (ORDER BY year, month) AS growth\_rate

FROM

monthly\_revenue\_duration

)

SELECT

month,

year,

revenue,

prev\_revenue,

duration\_hours,

prev\_duration\_hours,

growth\_rate

FROM

revenue\_growth

ORDER BY

growth\_rate DESC;

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a graph

Description automatically generated with medium confidence

Compared to last year's growth rate, the overall rate showing a decreasing order from January to July.