MICHIGAN ROSS

TO 640: Big Data Management Tools and Techniques

Final Project Report

Research Topic:

Weather Conditions and their Impact on the Airline Industry

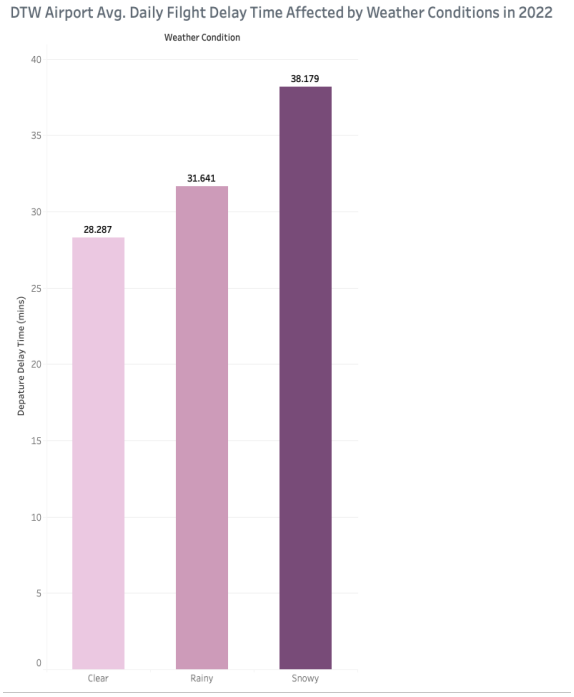
Group 27:
Jiaqi Liu
Yiqing Wang
Dong Yang
Ananya Srinivasa Desikan
Aadithyan Sai Subramanniann

25th February, 2025

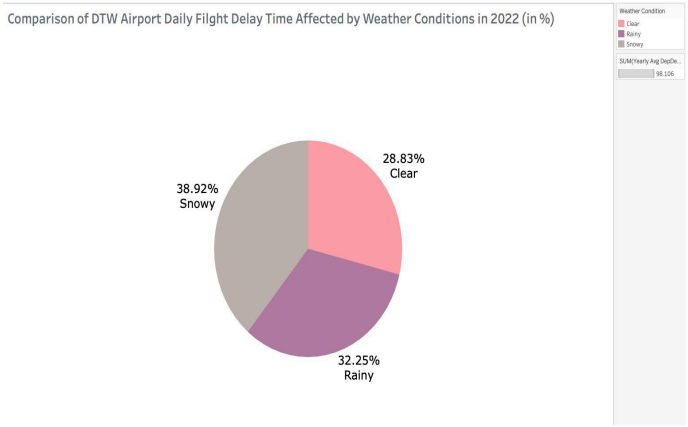
Executive Summary:

This report analyzes how different weather conditions affect the airline industry, focusing on flight delays, airport operational efficiency, snow-related disruptions across Michigan airports, seasonal delay differences, and airlines' ability to manage weather-related delays at DTW in 2022. The study provides insights into how force majeure factors like weather impact airport operations and coordination capabilities. Our flight data originated from the 'flights' SQL database, where we selected relevant fields and time frames, then removed any null or unnecessary entries. We also collected daily weather data from the Meteostat API, aligning both data sources by date and airport code. By matching each flight's scheduled date to its corresponding daily weather record, we could accurately compare delays across different weather conditions. This merged dataset enabled us to address the key business questions on how weather affects airline operations.

Question 1: Different weather conditions that affect flight delaying time at DTW airport in 2022



The bar chart indicates that the average daily flight departure delay time (in minutes) at Detroit Metropolitan Airport (STW) in 2022, there are three different weather conditions: clear, rainy, and snowy. The pie chart is a comparison of average delay time under these three weather conditions.

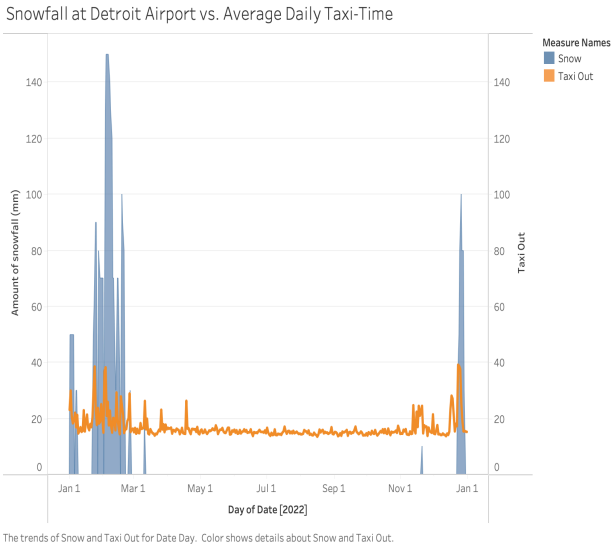


The key findings are firstly, **even clear weather** has an **average delay time as 28.287 min**, it may be because of operation inefficiencies, air traffic congestion, or technical issues. Secondly, **rainy weather** shows an avg. delay time of 31.641 min, **increasing by 3.35 mins** compared to clear days. It alludes that rainy weather could reduce visibility, wet runways and air traffic control could under slow operations in this case. Finally, **snowy weather** has the **highest delay time as 38.179 min**. In Detroit the winters are extremely cold, snowy weather conditions cause the most significant delays, since runways can be iced and ground operations could be disrupted.

However, delay times in rainy and snowy weather were expected to be significantly higher than clear weather. The reason from the analysis there are only a couple of minutes difference could be the research did not include canceled flight. Under extreme weather conditions, plenty of flights will be canceled that

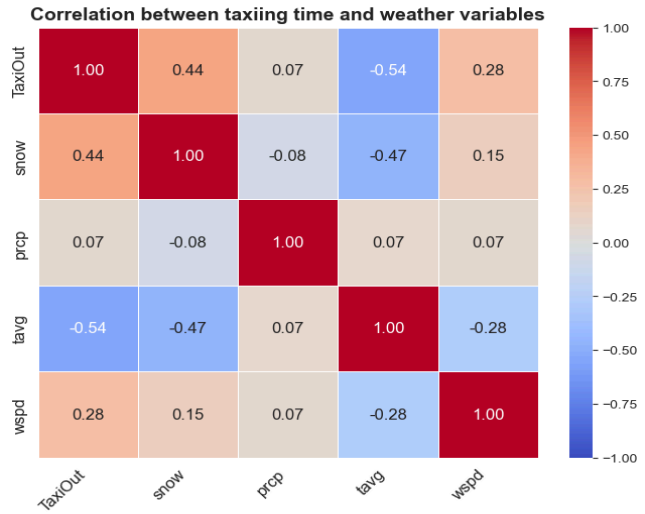
makes the total departure flights on that day decreased. If we count the canceled flights as another variable in the further study, we believe the impact will be more significant and accurate.

Question 2: Analysis of the impact of DTW airport operational efficiency under different weather conditions



Here, we analyze the impact of snowfall on taxi-out times.Through **Tableau visual analysis**, we observed that:On days with higher snowfall, taxi times increase significantly. This means that the taxi time of an aircraft on the runway is affected by snowfall, which may be due to:

- The need for additional snow removal operations (such as clearing runways and de-icing wings).
- Ground operations at the airport become more complicated, increasing the queuing time of the aircraft.
- Pilots may adopt a more conservative taxiing strategy in the event of low visibility and slippery runways.



We discovered from the correlation heat map:

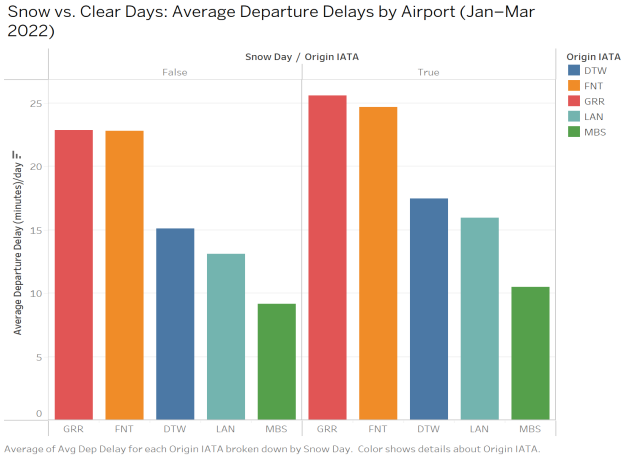
- The correlation coefficient between **snowfall and taxi time is 0.44**, indicating a middle positive correlation.
- **Precipitation** (prcp) has a relatively **small effect on taxi time** (**0.07**), probably because rain does not directly slow the ground speed.
- The average temperature (tavg) is negatively correlated with taxi time (-0.54), probably because snow combined with low temperatures creates ice, which requires additional de-icing operations at the airport.

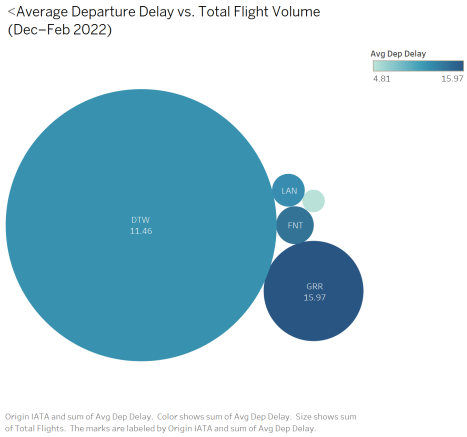
Question 3: Impact of Snowfall on Flight Delays Across Michigan Airports

This analysis examined the impact of snow on flight delays across major Michigan airports using historical weather and flight data. We used Meteosat API to collect daily snowfall data and SQL queries for flight delays at Michigan airports during winter months. We classified SnowDay = True if snowfall exceeded 0mm and ensured time consistency between weather and flight records.

We found that Lansing (LAN) and Grand Rapids (GRR) experience the largest increases in average departure delay on snowy days (+2.82 and +2.74 minutes, respectively), suggesting they may require stronger winter operations. Detroit Metro (DTW) shows a moderate rise of 2.38 minutes, indicating more robust snow management but still facing winter disruptions. Flint (FNT) and Saginaw (MBS) remain less

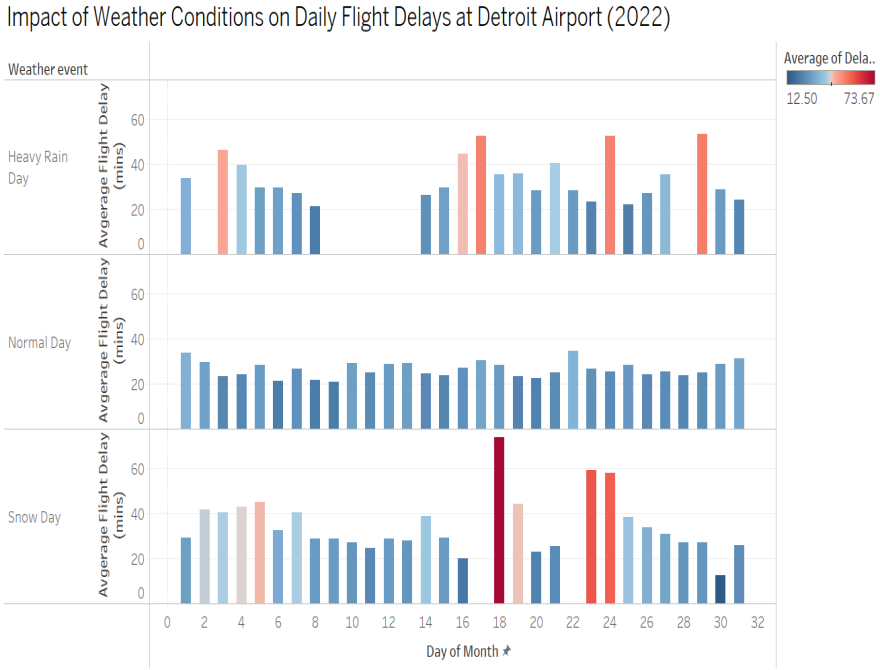
affected, at +1.88 and +1.33 minutes, potentially due to lower baseline delays or reduced traffic. To compare flight volume and average departure delay, I created a bubble chart where each airport's bubble size reflects its total flights, and its position indicates average delay. I found that Detroit Metro (DTW), despite having the largest flight volume, doesn't exhibit the highest average delay. This suggests that robust winter operations may help manage disruptions even under heavier traffic. Meanwhile, airports with smaller flight volumes can still see relatively higher delays, indicating that flight count alone doesn't determine performance—operational efficiency in snowy conditions also plays a key role.





<u>Question 4: To analyze if winter delays are worse than summer delays in DTW (2022)</u>

This analysis aims to analyze if winter delays are worse than summer delays in DTW airport. Based on the **python statistical t-test analysis** done, we got a **p-value of 0.0027 < 0.5**, which can tell us that winter delays are, in fact, worse than summer delays in the year 2022 in DTW airport. The obtained data is also visualized on Tableau as shown below. Two calculated fields–Weather event and Delay in DTW–are used for the visualization. Weather event field categorizes each day based on weather conditions, and determines the appropriate delay metric based on flight direction relative to DTW. If a flight originates from Detroit, it uses the departure delay (DepDelay), and vice versa for arrival delay (ArrDelay)**.**



- Snowy weather significantly impacts flight delays at Detroit Airport, causing the most severe delays and highest variability compared to heavy rain or normal days.

- While heavy rain also increases delays, the overall effect is consistently less severe than snow.

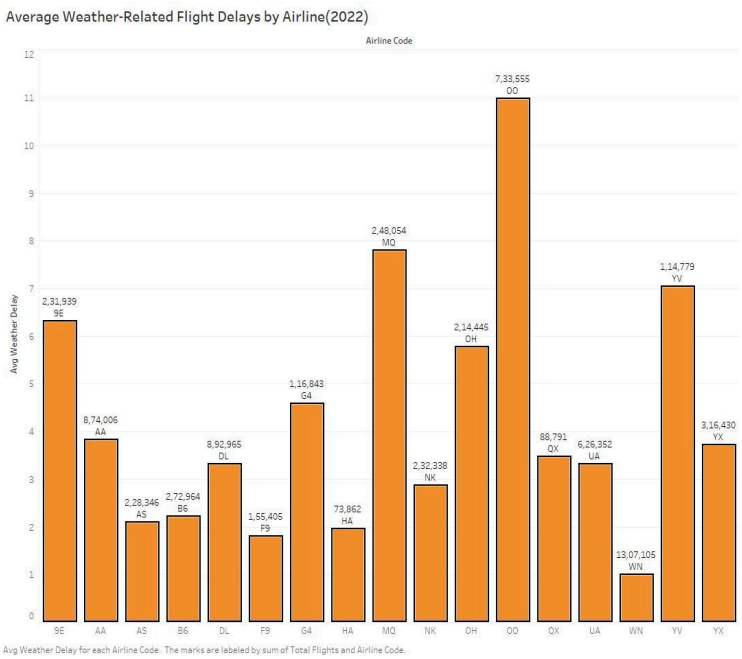
- Even normal weather days experience baseline delays, suggesting operational delays persist irrespective of weather conditions.
- Airlines and airport management should prioritize winter resource allocation and de-icing strategies, as snow clearly poses the greatest challenge to timely operations.

Question 5: Which airline handles weather-related delays more effectively in 2022

This analysis evaluates how different airlines managed weather-related flight delays in 2022, using historical flight and weather data. We merged flight delay records with daily weather conditions using a SQL-based join on "time" and "airport" to ensure accurate mapping between flight performance and weather disruptions. The Meteosat API was used to retrieve daily precipitation, snowfall, and wind speed data, which were then integrated into the dataset.

We calculated the average departure delay for each airline on days with measurable precipitation, snow, or high winds and visualized the results in Tableau. Airlines were ranked based on their average weather delay, with lower values indicating better handling of adverse conditions.

Key Findings & Insights:



- **Southwest Airlines (WN)** exhibited the **lowest average weather delay**, indicating strong operational resilience to adverse weather. This suggests effective scheduling, better contingency planning, and possibly more backup aircraft availability.
- Alaska Airlines (AS) and Hawaiian Airlines (HA) also showed lower-than-average delays, likely benefiting from operational experience in variable weather conditions.
- **SkyWest Airlines (OO)** had the **highest weather-related delay**, suggesting that regional airlines may struggle more with weather disruptions due to limited aircraft flexibility and reliance on smaller airports.
- Envoy Air (MQ) and Endeavor Air (9E) also had high delays, which aligns with trends observed for smaller regional carriers operating in weather-prone areas.
- Regional airlines (OO, MQ, 9E, YV) appear more affected by weather delays, potentially due to fewer backup planes and reliance on secondary airports.
- Major airlines like AA (American) and UA (United) show moderate delays, indicating better infrastructure and operational flexibility but still facing weather-related challenges.

APPENDIX

- **Detroit Weather Web Crawler: Meteostat API**

```python
import meteostat
from datetime import datetime

# Set the time range: January 1, 2016 - July 31, 2023
start = datetime(2016, 1, 1)
end = datetime(2023, 7, 31)

# Define the geographic coordinates (latitude, longitude, altitude) of Detroit
loc = meteostat.Point(42.348495, -83.060303, 70)

# Request the Meteostat API to get daily weather data
data = meteostat.Daily(loc, start, end)
data = data.fetch()

# display data
print(data.head())

# Select the required columns
measures = ["tavg", "tmin", "tmax", "prcp", "snow", "wdir", "wspd", "wpgt", "pres"]
weather_data = data[measures]

# Save data as CSV file
weather_data.to_csv("/Users/kevinyang/Desktop/Detroit_Weather_2016_2023.csv")
print("done!!!")


"""
Appendix: Use of AI in This Project
This project utilized AI assistance to:
    1. Understand the usage and logic of the Meteostat API.
    2. Fix the error
    3. After the CSV file is incorrectly written, help view the code and change to the desktop.
"""
```

This is the result of the 5C assignment, which we directly applied to the final project.

Question 1:

Step 1: Cleaning data from SQL

Flight-Flight: filter 2022 Detroit airport dep delay or arr delay flights.

```sql
SELECT id, AirlineCode, FlightNumber, OriginIATA, DestIATA, DepDelay, ArrDelay,
WeatherDelay, ArrivalTime
FROM flights.flight
WHERE (OriginIATA = 'DTW' OR DestIATA = 'DTW')
        AND (DepDelay > 0 OR ArrDelay > 0)
        AND YEAR(ArrivalTime) = 2022;
```

Step 2: Python- Identify weather types

```python
import pandas as pd

# read CSV file of weather
file_path = r"/Users/yiqing/Downloads/Detroit_Weather_2016_2023.csv"
df = pd.read_csv(file_path)

# identify weather typr
def classify_weather(row):
    if row['snow'] > 0:
        return "Snowy"  # snowy
    elif row['prcp'] > 0:
        return "Rainy"  # rainy
    elif row['wspd'] > 50:
        return "Windy"  # windy
    else:
        return "Clear"  # good

# create a new column of weather types
df['weather_condition'] = df.apply(classify_weather, axis=1)

# show prior lines
print(df.head())

# save back as CSV
df.to_csv(r"/Users/yiqing/Downloads/Detroit_Weather_Processed.csv", index=False)
```

Step 3: Filter 2022 weather data

```python
import pandas as pd

# read CSV file that just created, decode time
file_path = r"/Users/yiqing/Downloads/Detroit_Weather_Processed.csv"
df = pd.read_csv(file_path, parse_dates=['time'])

# ensure 'time' has been decode correctly
df['time'] = pd.to_datetime(df['time'], errors='coerce')

# check data types
print(df.dtypes)  # ensure time column is datetime64 type

# filter data in 2022
df_2022 = df[df['time'].dt.year == 2022]

# save as new csv file
output_path = r"/Users/yiqing/Downloads/Detroit_Weather_2022.csv"
df_2022.to_csv(output_path, index=False)
```

Step 4- merge weather condition to delay flight table

```python
import pandas as pd

# read delayed fight file
flight_path = r"/Users/yiqing/Downloads/Flight.csv"
df_flight = pd.read_csv(flight_path, parse_dates=['ArrivalTime'])

# read weather file
weather_path = r"/Users/yiqing/Downloads/Detroit_Weather_2022.csv"
df_weather = pd.read_csv(weather_path, parse_dates=['time'])

# ensure date format correct and read time
df_flight['Date'] = df_flight['ArrivalTime'].dt.date  # flight date
df_weather['Date'] = df_weather['time'].dt.date  # weather date

# **requres only weather conditions**
df_weather = df_weather[['Date', 'weather_condition']]

# **merge**
df_merged = df_flight.merge(df_weather, on='Date', how='left')

# **save as a new file**
output_path = r"/Users/yiqing/Downloads/Detroit_Flight_With_Weather.csv"
df_merged.to_csv(output_path, index=False)
```

Step 5 - Filter flight that only departed from DTW

```
#%%
import pandas as pd

# read file
file_path = "/Users/yiqing/Downloads/Detroit_Flight_With_Weather.csv"
df = pd.read_csv(file_path)

#ensure date
df["Date"] = pd.to_datetime(df["Date"])

# filter OriginIATA == "DTW"
df_dtw = df[df["OriginIATA"] == "DTW"]

# calculate delay time under diff. weather types (DepDelay)
avg_delay = df_dtw.groupby(["Date", "weather_condition"])["DepDelay"].mean().reset_index()

# name culomn
avg_delay.rename(columns={"DepDelay": "Avg_DepDelay"}, inplace=True)

# show prior rows
print(avg_delay.head())

# save as new CSV
output_path = "/Users/yiqing/Downloads/Avg_Delay_By_Weather.csv"
avg_delay.to_csv(output_path, index=False)
```

Step 6- calculated the daily average delay time for different weather types

```
import pandas as pd

#read the new file
file_path = "/Users/yiqing/Downloads/Avg_Delay_By_Weather.csv"
df = pd.read_csv(file_path)

# calculate yearly avg. delay time
yearly_avg_delay = df.groupby("weather_condition")["Avg_DepDelay"].mean().reset_index()

# rename culomn
yearly_avg_delay.rename(columns={"Avg_DepDelay": "Yearly_Avg_DepDelay"}, inplace=True)

# show prior rows
print(yearly_avg_delay)

# save new CSV
output_path = "/Users/yiqing/Downloads/Yearly_Avg_Delay_By_Weather.csv"
yearly_avg_delay.to_csv(output_path, index=False)
```

Question 2:

Step 1: cleaning data from SQL

```
SELECT FlightNumber, DestIATA AS Arrival_Airport, AirlineCode, TaxiOut, SchedDepart AS Scheduled_Departure,
ActualDepart AS Actual_Departure FROM flight WHERE OriginIATA = 'DTW' AND YEAR(SchedDepart) = 2022 AND TaxiOut IS
NOT NULL AND Cancelled = 0;
```

The purpose of this step is to clean up the flight table in SQL. Extract all flights from DTW in 2022. Clean up null values. And only look at delayed flights, not cancelled flights.

Step 2: Use Python to sort out the appropriate weather file

```python
import pandas as pd

# Read raw weather data
weather_data = pd.read_csv("/Users/kevinyang/Desktop/Detroit_Weather_2016_2023.csv")

# Make sure the time column is in datetime format.
weather_data['time'] = pd.to_datetime(weather_data['time'])

# Filtering out data from 2022
weather_2022 = weather_data[weather_data['time'].dt.year == 2022]

# Export new CSV
weather_2022.to_csv("/Users/kevinyang/Desktop/Detroit_Weather_2022.csv", index=False)

print("done!!")
```

Use python to extract the data for 2022 from the previous weather csv file. Because the previous weather csv file contains data from 2016 to 2023. Result looks like this:

Detroit_Weather_2022

time	tavg	tmin	tmax	prcp	snow	wdir	wspd	wpgt	pres
2022-01-01	3.4	-1.5	6.8	2.9	0	12	17		1008.2
2022-01-02	-3.3	-7.1	-2.1	4.5	50	356	19.6		1017.1
2022-01-03	-5.2	-9.3	-1.5	0	50	295	8.9		1027.9
2022-01-04	-0.8	-3.7	2.9	0	50	207	16.7		1020.5
2022-01-05	0.1	-8.2	4	0	50	230	28.7		1003.2
2022-01-06	-6.2	-8.2	-3.7	0	0	244	20.6		1018.7
2022-01-07	-7.4	-10.4	-4.3	0	30	299	14.2		1022.7
2022-01-08	-4.2	-11.5	1.8	0	30	181	15.8	115.2	1027
2022-01-09	0.7	-7.1	4	0.1	0	265	20.8		1019

Step 3: Use Python to organize the CSV file (flights) just exported from SQL

```python
import pandas as pd

taxi_data = pd.read_csv("/Users/kevinyang/Desktop/flight.csv")

# Make sure the time is listed in datetime format.
taxi_data['Scheduled_Departure'] = pd.to_datetime(taxi_data['Scheduled_Departure'])

# Retrieve the date and calculate the average TaxiOut time per day.
taxi_data['date'] = taxi_data['Scheduled_Departure'].dt.date
daily_taxiout = taxi_data.groupby('date')['TaxiOut'].mean().reset_index()

# Export CSV to desktop
daily_taxiout.to_csv("/Users/kevinyang/Desktop/Detroit_TaxiOut_Daily_2022.csv", index=False)

print("done!!!")
```

The goal is to obtain the average taxi time per day. For example, if 30 planes take off today, calculate the average taxi time for these 30 planes. Result looks like this:

Detroit_TaxiOut_Daily_2022

date	TaxiOut
2022-01-01	23.185929648241206
2022-01-02	30.00952380952381
2022-01-03	20.773700305810397
2022-01-04	18.379310344827587
2022-01-05	19.292207792207794

Step 4: Use Python to merge the two previous files

```python
import pandas as pd

# Read weather data
weather_data = pd.read_csv("/Users/kevinyang/Desktop/Detroit_Weather_2022.csv")

# Read taxi time data
taxiout_data = pd.read_csv("/Users/kevinyang/Desktop/Detroit_TaxiOut_Daily_2022.csv")

# Modify the time column of weather_data directly so that its name is the same as that of taxiout_data.
weather_data.rename(columns={'time': 'date'}, inplace=True)

# Merge data by date
merged_data = pd.merge(taxiout_data, weather_data, on="date", how="inner")

merged_data.to_csv("/Users/kevinyang/Desktop/Detroit_TaxiOut_Weather_2022.csv", index=False)

print("done!!!")
```

Change the names of the time columns in the two files to the same name to facilitate merging. Then merge to form a file that can finally be placed in tableau. Result looks like this:

Detroit_TaxiOut_Weather_2022

date	TaxiOut	tavg	tmin	tmax	prcp	snow	wdir	wspd	wpgt	pres
2022-01-01	23.185929648241206	3.4	-1.5	6.8	2.9	0	12	17		1008.2
2022-01-02	30.00952380952381	-3.3	-7.1	-2.1	4.5	50	356	19.6		1017.1
2022-01-03	20.7737003058104	-5.2	-9.3	-1.5	0	50	295	8.9		1027.9
2022-01-04	18.379310344827587	-0.8	-3.7	2.9	0	50	207	16.7		1020.5
2022-01-05	19.292207792207797	0.1	-8.2	4	0	50	230	28.7		1003.2
2022-01-06	22.11912225705329	-6.2	-8.2	-3.7	0	0	244	20.6		1018.7

Step 5: Try different variables in tableau and put them in rows and columns.

Step 6: Use Python to analyze the correlation between variables and taxi time

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from tkinter import Tk
from tkinter.filedialog import askopenfilename

# manually select a CSV file
Tk().withdraw()
file_path = askopenfilename(title="Select Detroit_TaxiOut_Weather_2022.csv")

# read data
data = pd.read_csv(file_path)

# Select relevant column
selected_columns = ["TaxiOut", "snow", "prcp", "tavg", "wspd"]
df = data[selected_columns]

# Calculate the correlation matrix
corr_matrix = df.corr()

# Create more attractive heat maps
plt.figure(figsize=(8, 6))
sns.set_style("whitegrid")

heatmap = sns.heatmap(
    corr_matrix,
    annot=True,
    cmap="coolwarm",
    center=0,
    linewidths=0.5,
    fmt=".2f",
    vmin=-1,
    vmax=1,
    annot_kws={"size": 12}
)

plt.title("Correlation between taxiing time and weather variables", fontsize=14, fontweight="bold")
plt.xticks(rotation=45, ha='right', fontsize=12)
plt.yticks(fontsize=12)

plt.show()
```

Regarding the use of AI in question 2:

AI was used to help analyze part of question 2 in the following ways:

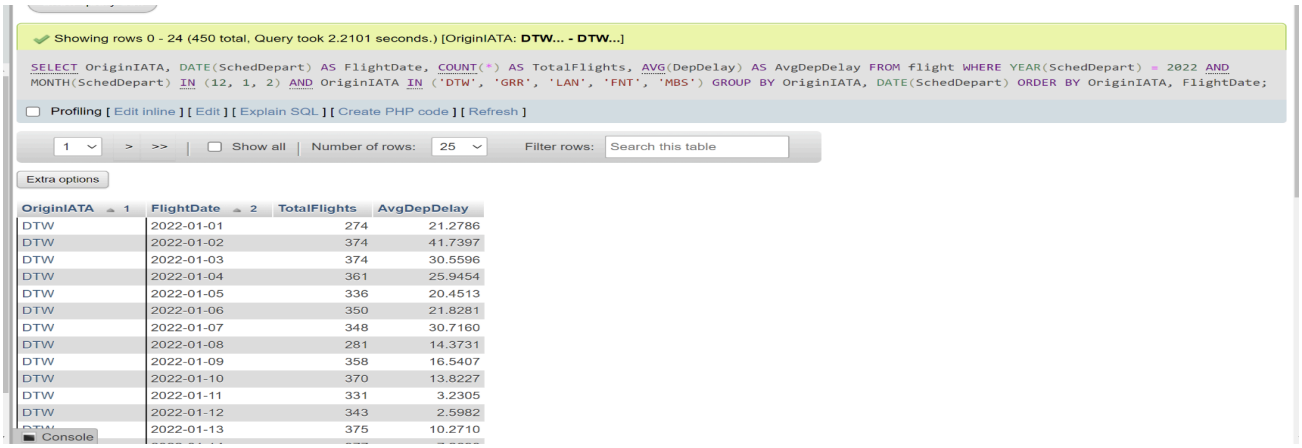
1. Show me how to use group by in python to calculate the average.

2. In the correlation analysis, I encountered problems letting Python read files. In the end, AI helped me write some code that allows me to manually select the file location and then read it.
3. Use of libraries not learned in class for correlation analysis.
4. Helps to beautify the visualization of correlation heat maps.

Question 3:

Step 1: cleaning data from SQL

SQL query in phpMyAdmin was used to filter flights from January through March 2022 at five Michigan airports (LAN, FNT, MBS, GRR, and DTW). The query grouped flights by day and airport, returning the total number of flights and the average departure delay. After verifying the results, the dataset was exported as a CSV file for further analysis in Python and Tableau.



Showing rows 0 - 24 (450 total, Query took 2.2101 seconds.) [OriginIATA: DTW... - DTW...]

```
SELECT OriginIATA, DATE(SchedDepart) AS FlightDate, COUNT(*) AS TotalFlights, AVG(DepDelay) AS AvgDepDelay FROM flight WHERE YEAR(SchedDepart) = 2022 AND
MONTH(SchedDepart) IN (12, 1, 2) AND OriginIATA IN ('DTW', 'GRR', 'LAN', 'FNT', 'MBS') GROUP BY OriginIATA, DATE(SchedDepart) ORDER BY OriginIATA, FlightDate;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

1 > >> | Show all | Number of rows: 25 Filter rows: Search this table

Extra options

OriginIATA 1	FlightDate 2	TotalFlights	AvgDepDelay
DTW	2022-01-01	274	21.2786
DTW	2022-01-02	374	41.7397
DTW	2022-01-03	374	30.5596
DTW	2022-01-04	361	25.9454
DTW	2022-01-05	336	20.4513
DTW	2022-01-06	350	21.8281
DTW	2022-01-07	348	30.7160
DTW	2022-01-08	281	14.3731
DTW	2022-01-09	358	16.5407
DTW	2022-01-10	370	13.8227
DTW	2022-01-11	331	3.2305
DTW	2022-01-12	343	2.5982
DTW	2022-01-13	375	10.2710

Console

Step 2:

Python script was used with the Meteosat library to fetch daily weather records from January through March 2022 for the same five Michigan airports (LAN, FNT, MBS, GRR, and DTW). The script converted the time index into a standard column, filled missing snowfall values with 0, and flagged any day with recorded snow as snow day. After verifying the data, this combined dataset was saved as a CSV file for subsequent analysis.

```python
# %% Import libraries
import pandas as pd
import meteostat
from datetime import datetime

# %% Step 1: Define time range and airports
start = datetime(2022, 1, 1)
end   = datetime(2022, 12, 31)

michigan_airports = {
    "DTW": meteostat.Point(42.2125, -83.3534, 195),
    "GRR": meteostat.Point(42.8803, -85.5229, 252),
    "LAN": meteostat.Point(42.7786, -84.5874, 266),
    "FNT": meteostat.Point(42.9655, -83.7435, 233),
    "MBS": meteostat.Point(43.5329, -84.0797, 204),
}

# %% Step 2: Fetch weather data and keep "time" column
weather_data = []

for code, location in michigan_airports.items():
    daily_data = meteostat.Daily(location, start, end).fetch()
    if not daily_data.empty:
        # Convert index to a normal column named "time"
        daily_data.reset_index(inplace=True)

        # Convert "time" to datetime (but keep the column named "time")
        if "time" in daily_data.columns:
            daily_data["time"] = pd.to_datetime(daily_data["time"])

        # Add the airport code
        daily_data["Airport"] = code
        weather_data.append(daily_data)
```

```python
# %% Step 3: Combine data
weather_df = pd.concat(weather_data, ignore_index=True)

# Fill missing "snow" with 0
weather_df["snow"] = weather_df.get("snow", 0).fillna(0)

# Mark SnowDay if snow > 0
weather_df["SnowDay"] = weather_df["snow"] > 0

# %% Step 4: Save final CSV
weather_df.to_csv("Processed_Weather_Data.csv", index=False)
print("Processed_Weather_Data.csv' created with 'time' column intact!")
```

Step 3: Compare Average Departure Delay vs. Total Flight Volume (Dec–Feb)

#Used ChatGPT to automate and optimize key parts of this analysis, including:

#Snow Classification: ChatGPT helped define a rule where SnowDay = True if snowfall > 0mm, and False otherwise. Missing values were automatically set to 0.

#Time Handling: ChatGPT provided solutions to convert Meteostat's "time" index into a proper datetime format and align it with flight data.

#By using ChatGPT, I was able to automate data processing, reduce errors

<u>Question 4:</u>

Step 1: Clean data using SQL
This SQL query retrieves 2022 flight data for Detroit (DTW), including departure/arrival delays, cancellations, and weather-related delays. It filters for flights departing from or arriving at DTW, ensures delays are non-negative, and selects key timing and airline details for analyzing weather impacts on delays and cancellations.

```sql
SELECT
    FlightNumber, AirlineCode, OriginIATA, DestIATA,
    SchedDepart, ActualDepart, SchedArrival, ArrivalTime,
    DepDelay, ArrDelay, Cancelled, CancellationReason,
    WeatherDelay
FROM flight
WHERE (OriginIATA = 'DTW' OR DestIATA = 'DTW')
AND YEAR(SchedDepart) = 2022
AND (DepDelay >= 0 OR ArrDelay >= 0);
```

Step-2: Obtain Weather Data from Meteostat API
This was done in Assignment 5C but has been altered to show data only for DTW airport in 2022. This code connects to Meteostat API, fetches daily weather data and stores it as a

DataFrame.

```python
from meteostat import Daily
from datetime import datetime
import pandas as pd

# Fetch and Prepare Weather Data
start = datetime(2022, 1, 1)
end = datetime(2022, 12, 31)

# Use Detroit Metro Airport Weather Station (72537)
detroit_weather = Daily('72537', start, end).fetch()


# Select relevant columns
weather_df = detroit_weather[['tavg', 'prcp', 'snow', 'wspd']]
weather_df.reset_index(inplace=True)
weather_df.rename(columns={"time": "flight_date"}, inplace=True)
```

```python
# Load and Prepare Flight Data
flights_df = pd.read_csv("C:/Users/adesikan/Downloads/flight.csv")

# Convert datetime columns
flights_df['SchedDepart'] = pd.to_datetime(flights_df['SchedDepart'])
flights_df['SchedArrival'] = pd.to_datetime(flights_df['SchedArrival'])

# Create 'flight_date' column from 'SchedDepart'
flights_df['flight_date'] = flights_df['SchedDepart'].dt.date
flights_df['flight_date'] = pd.to_datetime(flights_df['flight_date'])


# Merge Flight and Weather Data
merged_df = flights_df.merge(weather_df, on='flight_date', how='left')
```

This code then loads the cleaned flight data obtained from SQL and converts scheduled departure & arrival times to datetime format, to make it easier for merging with weather data. Flight data is then merged with weather data.

Step-3: Categorize flights by season and create weather event flag (in this case, snow and thunderstorm)

```python
# Define Season and Weather Flags
def assign_season(month):
    if month in [12, 1, 2]:
        return 'Winter'
    elif month in [6, 7, 8]:
        return 'Summer'
    else:
        return 'Other'

merged_df['season'] = merged_df['flight_date'].dt.month.apply(assign_season)

# Weather event flags
merged_df['snow_event'] = merged_df['snow'] > 0
# Thunderstorm event defined as days with significant rainfall (e.g., >5mm)
merged_df.loc[:, 'thunderstorm_event'] = merged_df['prcp'] > 5.0
```

This code uses month numbers to assign flights to Winter (Dec-Feb) or Summer (Jun-Aug). It then flags days with snowfall as snow>0.

Step 4: Statistical Analysis to answer, “Are winter delays worse than summer delays?”

```python
# Perform Statistical Analysis
from scipy.stats import ttest_ind

# Winter vs Summer delay comparison
winter_delays = merged_df[merged_df['season'] == 'Winter']['DepDelay'].dropna()
summer_delays = merged_df[merged_df['season'] == 'Summer']['DepDelay'].dropna()

t_stat, p_value = ttest_ind(winter_delays, summer_delays, equal_var=False)
print(f"\nWinter vs Summer Delays T-test:\nT-statistic: {t_stat}, P-value: {p_value}")

# Export Merged Data for Tableau
merged_df.to_csv("flights_weather_tableau.csv", index=False)

print("\nMerged dataset exported successfully!")
```

This code compares the average departure delay in Winter vs. Summer and conducts a t-test to determine statistical significance. If **p-value < 0.05**, winter delays are significantly worse than summer delays.

Question 5:

Step 1: Clean data using SQL

```sql
SELECT
    f.AirlineCode,
    COUNT(*) AS TotalFlights,
    SUM(CASE WHEN f.WeatherDelay > 0 THEN 1 ELSE 0 END) AS WeatherDelayedFlights,
    AVG(f.WeatherDelay) AS AvgWeatherDelay,
    SUM(CASE WHEN f.Cancelled = 1 THEN 1 ELSE 0 END) AS TotalCancellations,
    ROUND(100.0 * SUM(CASE WHEN f.WeatherDelay > 0 THEN 1 ELSE 0 END) / COUNT(*), 2) AS WeatherDelayRate,
    ROUND(100.0 * SUM(CASE WHEN f.Cancelled = 1 THEN 1 ELSE 0 END) / COUNT(*), 2) AS CancellationRate
FROM fligh f
WHERE YEAR(f.SchedDepart) = 2022
GROUP BY f.AirlineCode
ORDER BY AvgWeatherDelay ASC, CancellationRate ASC;
```

Step 2: Obtain Weather Data from Meteostat API

This code filters data only for the delays occurred in the year 2022.

```python
import pandas as pd
import meteostat
from datetime import datetime

# %% Step 1: Define time range and airports
start = datetime(2022, 1, 1)
end   = datetime(2022, 12, 31)

michigan_airports = {
    "DTW": meteostat.Point(42.2125, -83.3534, 195),
    "GRR": meteostat.Point(42.8803, -85.5229, 252),
    "LAN": meteostat.Point(42.7786, -84.5874, 266),
    "FNT": meteostat.Point(42.9655, -83.7435, 233),
    "MBS": meteostat.Point(43.5329, -84.0797, 204),
}

# %% Step 2: Fetch weather data and keep "time" column
weather_data = []

for code, location in michigan_airports.items():
    daily_data = meteostat.Daily(location, start, end).fetch()
    if not daily_data.empty:
        daily_data.reset_index(inplace=True)  # Convert index to column
        daily_data["time"] = pd.to_datetime(daily_data["time"])  # Ensure correct format
        daily_data["Airport"] = code  # Add airport code
        weather_data.append(daily_data)
```

Step 3: Obtain a merged .csv file containing the airline data and the weather delay data to visualize in Tableau

```python
# %% Step 4: Load Flight Data
flight_df = pd.read_csv("flight.csv")

# Ensure 'time' is in datetime format for both datasets
flight_df["time"] = pd.to_datetime(flight_df["time"])
weather_df["time"] = pd.to_datetime(weather_df["time"])

# %% Step 5: Merge Flight and Weather Data
merged_df = pd.merge(
    flight_df, weather_df, on=["time", "Airport"], how="left"
)

# %% Step 6: Save Final Merged Data
merged_df.to_csv("Merged_Flight_Weather.csv", index=False)
print("✅ 'Merged_Flight_Weather.csv' is ready for Tableau visualization!")
```