

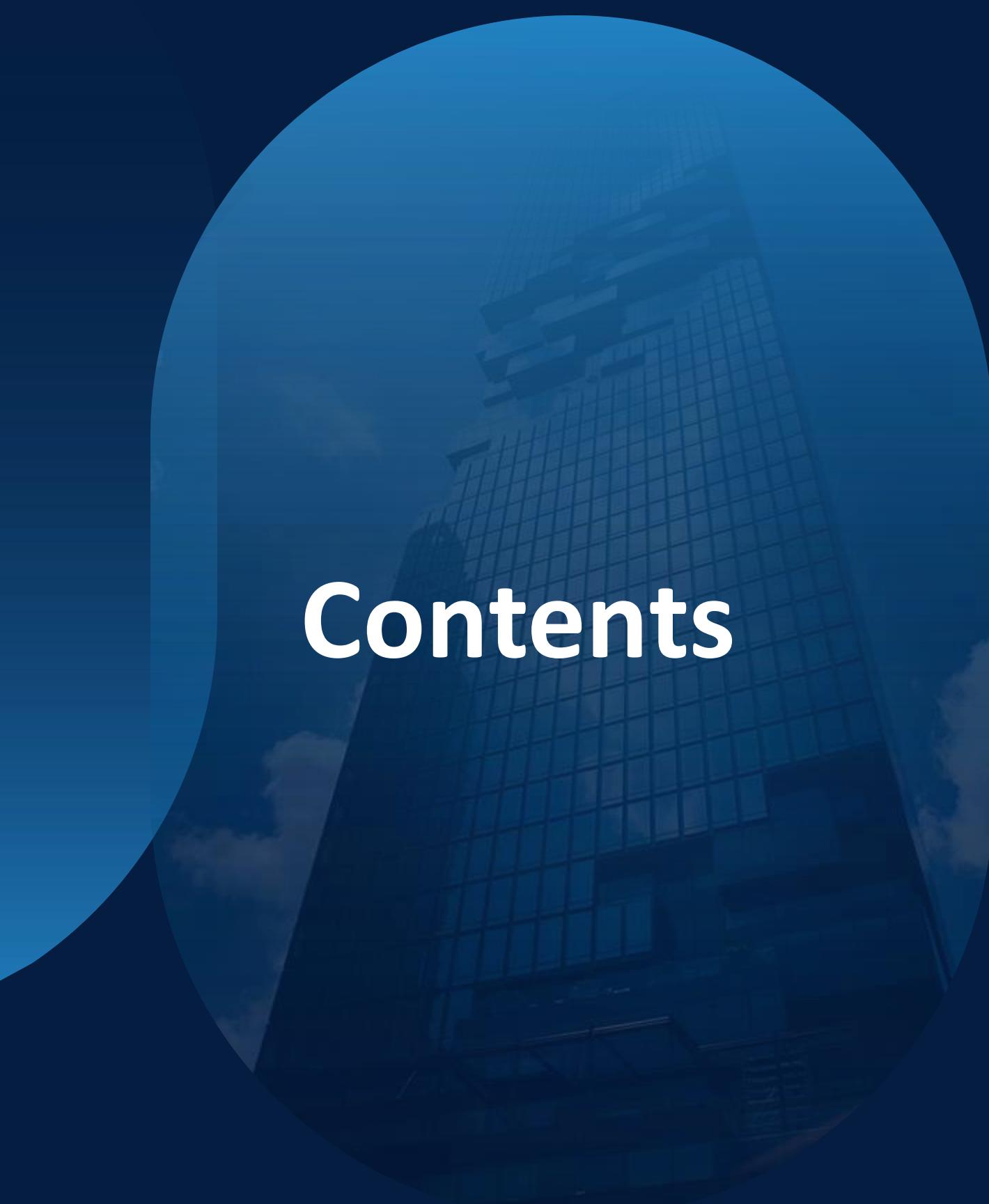
ETL Project

NASA Mars Rover

Data Analysis

Course: Business Intelligence
Presented by: Victoria Chueh

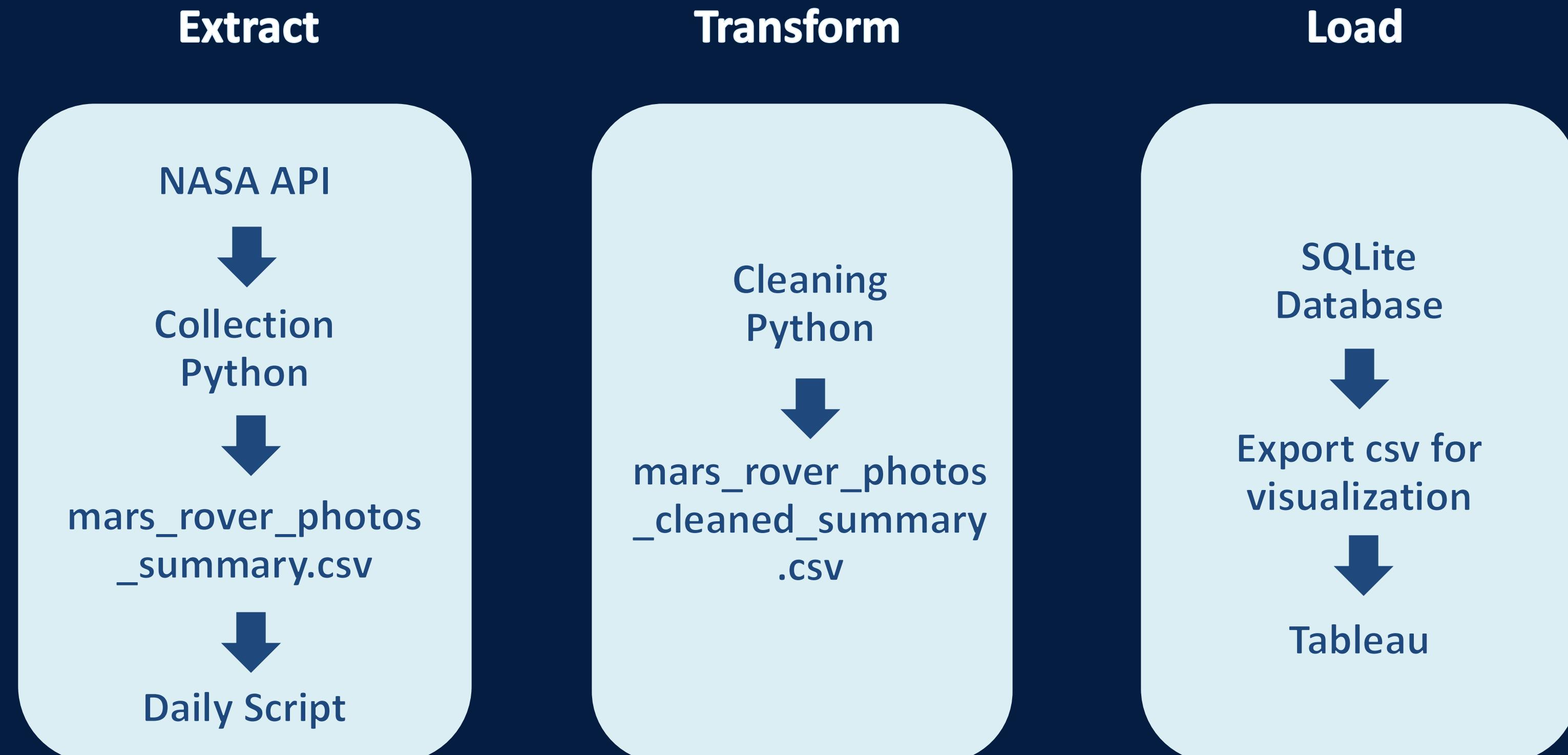




Contents

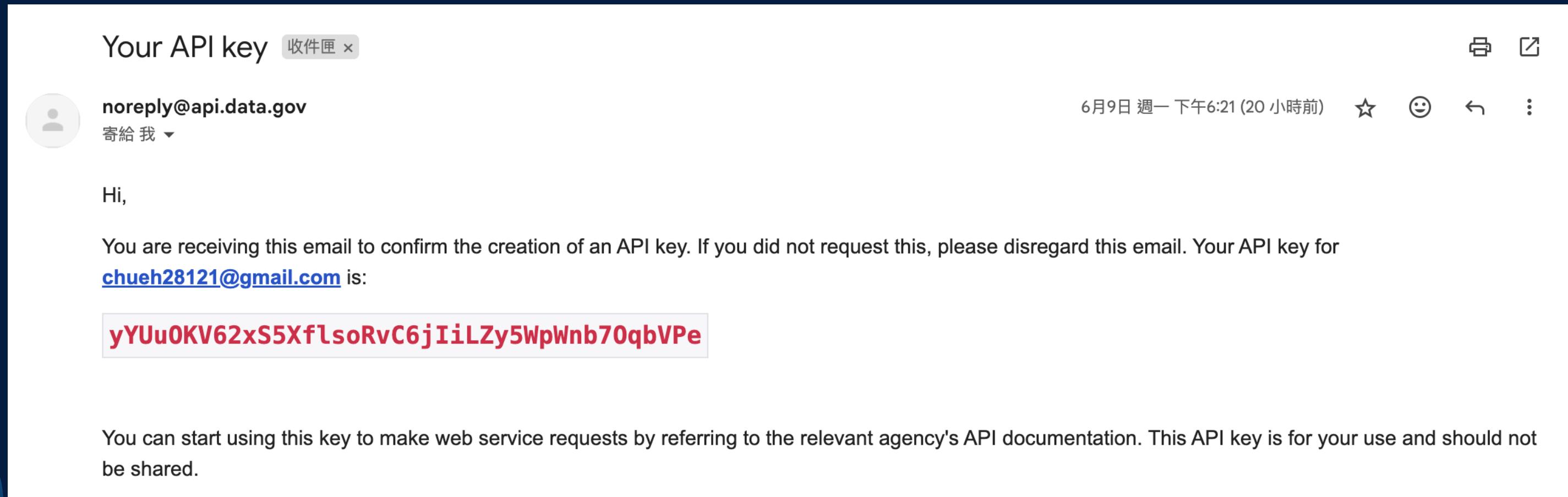
- 01 ETL Flow
- 02 Data Collection
- 03 Data Cleaning & Preparation
- 04 Data Storage
- 05 Workflow Orchestration
- 06 Data Analysis & Visualization

01 ETL Flow



02 Data Collection

NASA APIs – Imagery, astronomy, satellites, and more
<https://api.nasa.gov/>



The screenshot shows an email titled "Your API key" from "noreply@api.data.gov". The recipient is "chueh28121@gmail.com". The email body contains a greeting "Hi," and a message confirming the creation of an API key. It states: "You are receiving this email to confirm the creation of an API key. If you did not request this, please disregard this email. Your API key for [chueh28121@gmail.com](#) is:" followed by a red box containing the API key: "yYUuOKV62xS5Xf1soRvC6jIiLZy5WpWnb70qbVPe". A note at the bottom says: "You can start using this key to make web service requests by referring to the relevant agency's API documentation. This API key is for your use and should not be shared."

Your API key 收件匣 ×

 noreply@api.data.gov 6月9日 週一下午6:21 (20 小時前) ☆ 😊 ✉ ⋮

寄給我 ▾

Hi,

You are receiving this email to confirm the creation of an API key. If you did not request this, please disregard this email. Your API key for [chueh28121@gmail.com](#) is:

yYUuOKV62xS5Xf1soRvC6jIiLZy5WpWnb70qbVPe

You can start using this key to make web service requests by referring to the relevant agency's API documentation. This API key is for your use and should not be shared.

```

1 import requests
2 import pandas as pd
3 from collections import defaultdict
4 import time
5
6 API_KEY = "yYUuOKV62xS5XflsoRvC6jIiLZy5WpWnb70qbVPe" # My NASA API Key
7 ROVERS = ["curiosity", "perseverance"]
8 START_DATE = "2023-01-01"
9 END_DATE = "2023-01-31" # Collect data of January 2023
10
11 def daterange(start_date, end_date):
12     from datetime import datetime, timedelta
13     start = datetime.strptime(start_date, "%Y-%m-%d")
14     end = datetime.strptime(end_date, "%Y-%m-%d")
15     delta = timedelta(days=1)
16     current = start
17     while current <= end:
18         yield current.strftime("%Y-%m-%d")
19         current += delta
20
21 def fetch_photos(rover, date):
22     url = f"https://api.nasa.gov/mars-photos/api/v1/rovers/{rover}/photos"
23     params = {
24         "earth_date": date,
25         "api_key": API_KEY
26     }
27     response = requests.get(url, params=params)
28     if response.status_code == 200:
29         photos = response.json().get("photos", [])
30         print(f"{rover} {date} 照片數量: {len(photos)}")
31         return photos
32     else:
33         print(f"API錯誤[{response.status_code}] on {rover} {date}")
34     return []
35

```

```

36 def main():
37     records = []
38     for rover in ROVERS:
39         for date in daterange(START_DATE, END_DATE):
40             photos = fetch_photos(rover, date)
41             camera_counts = defaultdict(int)
42             for photo in photos:
43                 camera_counts[photo["camera"]["name"]] += 1
44
45             total_photos = len(photos)
46             if total_photos > 0:
47                 for camera, count in camera_counts.items():
48                     records.append({
49                         "rover": rover,
50                         "earth_date": date,
51                         "camera": camera,
52                         "photo_count": count,
53                         "total_photos": total_photos
54                     })
55             else:
56                 records.append({
57                     "rover": rover,
58                     "earth_date": date,
59                     "camera": None,
60                     "photo_count": 0,
61                     "total_photos": 0
62                 })
63             time.sleep(1)
64
65     df = pd.DataFrame(records)
66     df.to_csv("/Users/vc/Downloads/mars_rover_photos_summary.csv", index=False)
67     print("資料已儲存 mars_rover_photos_summary.csv")
68
69     if __name__ == "__main__":
70         main()

```



mars_rover_photos_summary.csv

CSV 文件

03 Data Cleaning & Preparation

```
1 import pandas as pd
2 import numpy as np
3
4 # Load the CSV downloaded from API
5 df = pd.read_csv("mars_rover_photos_summary.csv")
6
7 # 1. Check data structure and missing values
8 print(df.info())
9 print(df.head())
10 print(df.isnull().sum()) # Which columns have missing values?
11
12 # 2. Fill or handle missing values
13 # The "camera" column has None (null) values, so we can fill them with the string "Unknown" for easier analysis later.
14 df['camera'] = df['camera'].fillna('Unknown')
15
16 # 3. Format adjustment
17 # Ensure the date is in datetime format for easier time-based analysis.
18 df['earth_date'] = pd.to_datetime(df['earth_date'])
19
20 # 4. Filter or transform columns
21 # For example, only look at data where photos were taken (total_photos > 0).
22 df_nonzero = df[df['total_photos'] > 0].copy()
23
24 # 5. Add calculated columns (Optional)
25 # Calculate the photo ratio (photo_count / total_photos) for all cameras on the same day for the same rover.
26 df_nonzero['photo_ratio'] = df_nonzero['photo_count'] / df_nonzero['total_photos']
27
28 # 6. Confirm data status after cleaning
29 print(df_nonzero.head())
30
31 # 7. Export the cleaned data for use in presentations or dashboards.
32 df_nonzero.to_csv("mars_rover_photos_summary_cleaned.csv", index=False)
33 print("Cleaned data saved to mars_rover_photos_summary_cleaned.csv")
```



mars_rover_photos_summary_cleaned.csv

CSV 文件

03 Data Cleaning & Preparation

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
/usr/local/bin/python3 "/Users/vc/Downloads/# 將 SQLite 中的資料輸出為 CSV.py"
● vc@VC-MacBook-Air ~ % /usr/local/bin/python3 "/Users/vc/Downloads/# 將 SQLite 中的資料輸出為 CSV.py"
● vc@VC-MacBook-Air ~ % /usr/local/bin/python3 "/Users/vc/Downloads/# 將 SQLite 中的資料輸出為 CSV.py"
    ✓ 匯出完成 : mars_photos_for_tableau.csv
○ vc@VC-MacBook-Air ~ % >....
   2   camera      339 non-null    object
   3   photo_count  341 non-null    int64
   4   total_photos 341 non-null    int64
dtypes: int64(2), object(3)
memory usage: 13.4+ KB
None
      rover  earth_date   camera  photo_count  total_photos
0  curiosity  2023-01-01    FHAZ          5        357
1  curiosity  2023-01-01    RHAZ          2        357
2  curiosity  2023-01-01    MAST         243        357
3  curiosity  2023-01-01  CHEMCAM         28        357
4  curiosity  2023-01-01   MAHLI         68        357
rover
earth_date
camera
photo_count
total_photos
dtype: int64
      rover  earth_date   camera  photo_count  total_photos  photo_ratio
0  curiosity  2023-01-01    FHAZ          5        357  0.014006
1  curiosity  2023-01-01    RHAZ          2        357  0.005602
2  curiosity  2023-01-01    MAST         243        357  0.680672
3  curiosity  2023-01-01  CHEMCAM         28        357  0.078431
4  curiosity  2023-01-01   MAHLI         68        357  0.190476
清理後資料已儲存 mars_rover_photos_summary_cleaned.csv
vc@VC-MacBook-Air ~ % /usr/local/bin/python3 "/Users/vc/Downloads/import pandas as pd.py"
zsh: parse error near `\'n'
○ vc@VC-MacBook-Air ~ %
```

Filled the 2 missing values in the camera column using fillna()

The earth_date column has been correctly converted to datetime format.

Calculated the photo ratio, which represents the proportion of photos taken by each camera relative to the total photos

Saved cleaned data to mars_rover_photos_summary_cleaned.csv.

04 Data Storage

```
1 import sqlite3
2 import pandas as pd
3 import os
4
5 # 1. Read the cleaned data
6 input_path = os.path.expanduser("~/data/processed/mars_rover_photos_summary_cleaned.csv")
7 df_cleaned = pd.read_csv(input_path)
8
9 # 2. Establish SQLite database connection (creates the database file if it doesn't exist)
10 conn = sqlite3.connect("mars_rover_photos.db")
11
12 # 3. Write data to the table: photos_summary
13 df_cleaned.to_sql("photos_summary", conn, if_exists="replace", index=False)
14
15 print("✅ Data successfully written to the 'photos_summary' table in 'mars_rover_photos.db'")
16
17 # 4. Query: Number of records for each rover
18 query = "SELECT rover, COUNT(*) as count FROM photos_summary GROUP BY rover"
19 result = pd.read_sql_query(query, conn)
20
21 print("\n📊 Number of records per rover:")
22 print(result)
23
24 # 5. Close the connection
25 conn.close()
```

04 Data Storage

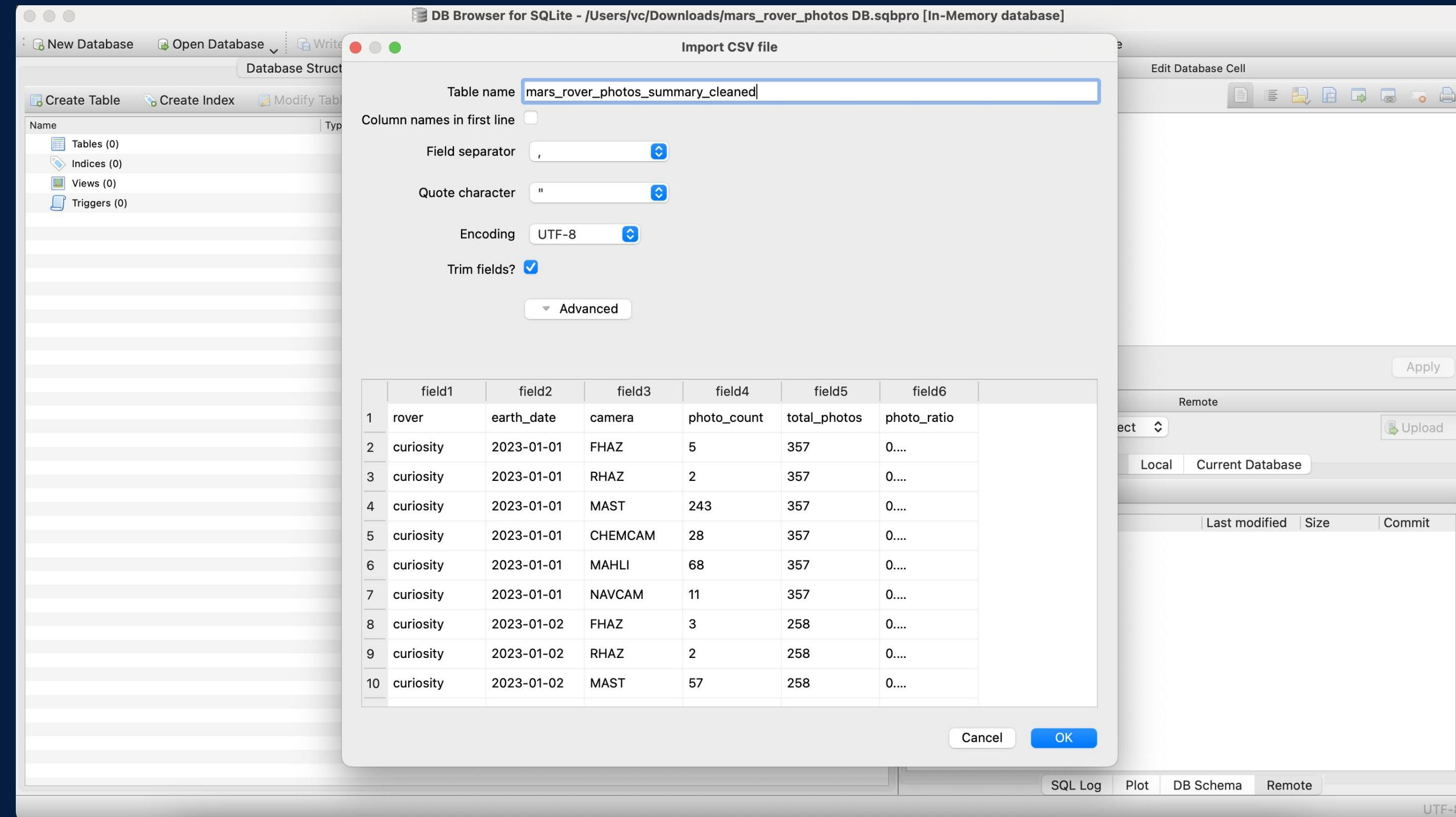
```
vc@VC-MacBook-Air ~ % /usr/local/bin/python3 /Users/vc/Downloads/save_to_sqlite.py
✓ 資料已成功寫入資料庫 mars_rover_photos.db 的 photos_summary 資料表

📊 每台探測車的紀錄數量：
    rover  count
0  Perseverance  192
1      curiosity  147
```

 mars_rover_photos.db

文件

04 Data Storage



04 Data Storage

The screenshot shows the DB Browser for SQLite interface. The title bar indicates it's connected to an in-memory database at `/Users/vc/Downloads/mars_rover_photos DB.sqbpro`. The main window has tabs for Database Structure, Browse Data, Edit Pragmas, and Execute SQL. The Execute SQL tab is active, displaying the following SQL query and its results:

```
1 SELECT * FROM mars_rover_photos_summary_cleaned
```

	field1	field2	field3	field4	field5	field6
1	rover	earth_date	camera	photo_count	total_photos	photo_ratio
2	curiosity	2023-01-01	FHAZ	5	357	0.014005602240896359
3	curiosity	2023-01-01	RHAZ	2	357	0.0056022408963585435
4	curiosity	2023-01-01	MAST	243	357	0.680672268907563
5	curiosity	2023-01-01	CHEMCAM	28	357	0.0784313725490196
6	curiosity	2023-01-01	MAHLI	68	357	0.19047619047619047
7	curiosity	2023-01-01	NAVCAM	11	357	0.03081232492997199
8	curiosity	2023-01-02	FHAZ	3	258	0.011627906976744186
9	curiosity	2023-01-02	RHAZ	2	258	0.007751937984496124
10	curiosity	2023-01-02	MAST	57	258	0.22093023255813954
11	curiosity	2023-01-02	CHEMCAM	3	258	0.011627906976744186
12	curiosity	2023-01-02	MAHLI	26	258	0.10077519379844961
13	curiosity	2023-01-02	MARDI	2	258	0.007751937984496124
14	curiosity	2023-01-02	NAVCAM	165	258	0.6395348837209303
15	curiosity	2023-01-03	MAST	28	50	0.56
16	curiosity	2023-01-03	CHEMCAM	4	50	0.08
17	curiosity	2023-01-03	NAVCAM	18	50	0.36
18	curiosity	2023-01-04	FHAZ	6	389	0.015424164524421594

Execution finished without errors.
Result: 340 rows returned in 26ms
At line 1:
SELECT * FROM mars_rover_photos_summary_cleaned

The right panel shows the database schema, listing one table: `mars_rover_photos_summary_cleaned` with the following CREATE TABLE statement:

```
CREATE TABLE "mars_rover_photos_summary_cleaned"
```

05 Workflow Orchestration

```
[vc@VC-MacBook-Air ~ % crontab -e  
  
UW PICO 5.09                                         File: /tmp/crontab.H3piky707K  
  
0 9 * * * /usr/local/bin/python3 /Users/vc/Downloads/download_data.py >> /Users/vc/Downloads/cron_log.txt 2>&1
```

```
crontab: installing new crontab  
vc@VC-MacBook-Air ~ % crontab -l  
0 9 * * * /usr/local/bin/python3 /Users/vc/Downloads/download_data.py >> /Users/vc/Downloads/cron_log.txt 2>&1
```

06 Data Analysis & Visualization



export_sqlite_to_csv.py

Python Script

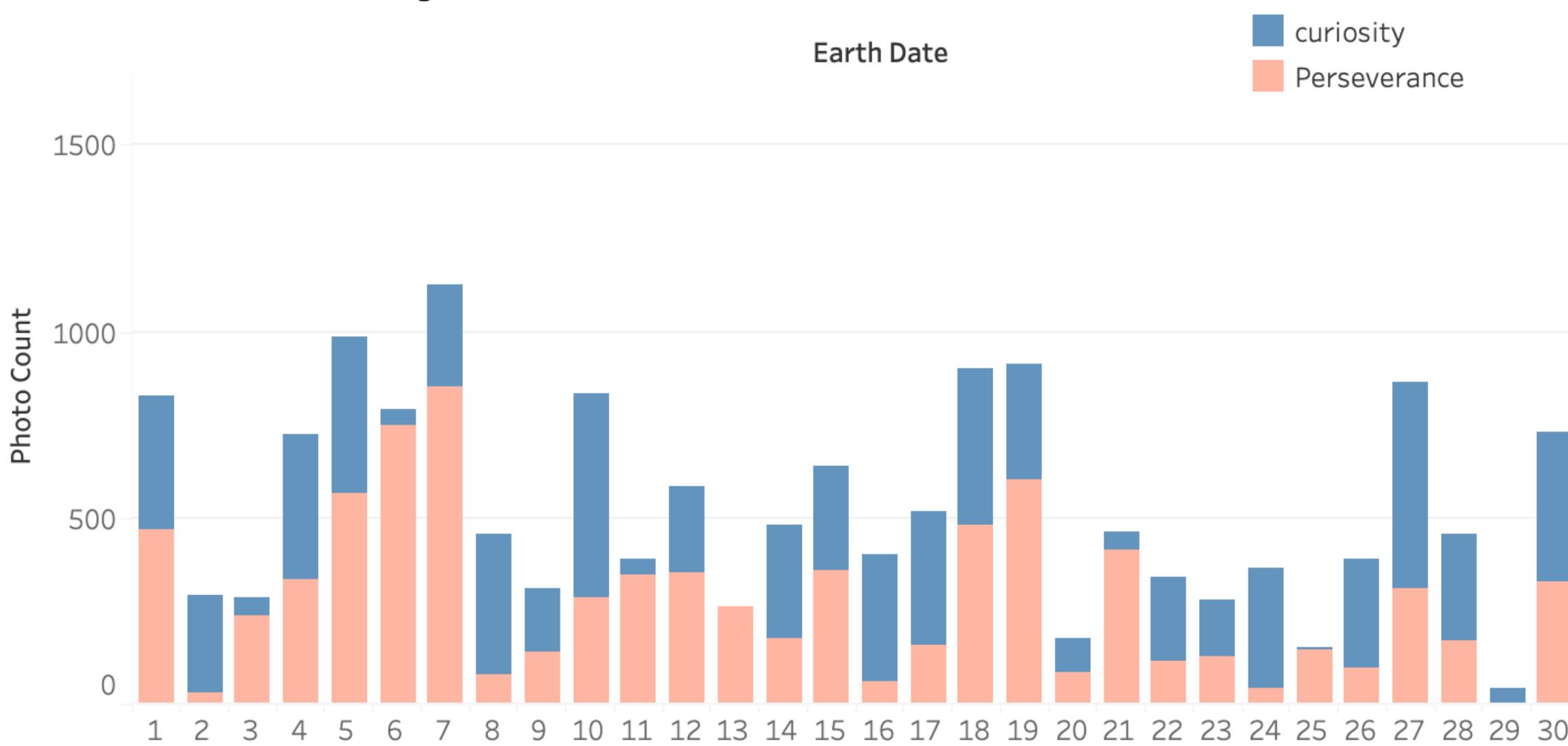
```
vc@VC-MacBook-Air ~ % /usr/local/bin/python3 /Users/vc/Downloads/export_sqlite_to_csv.py
```



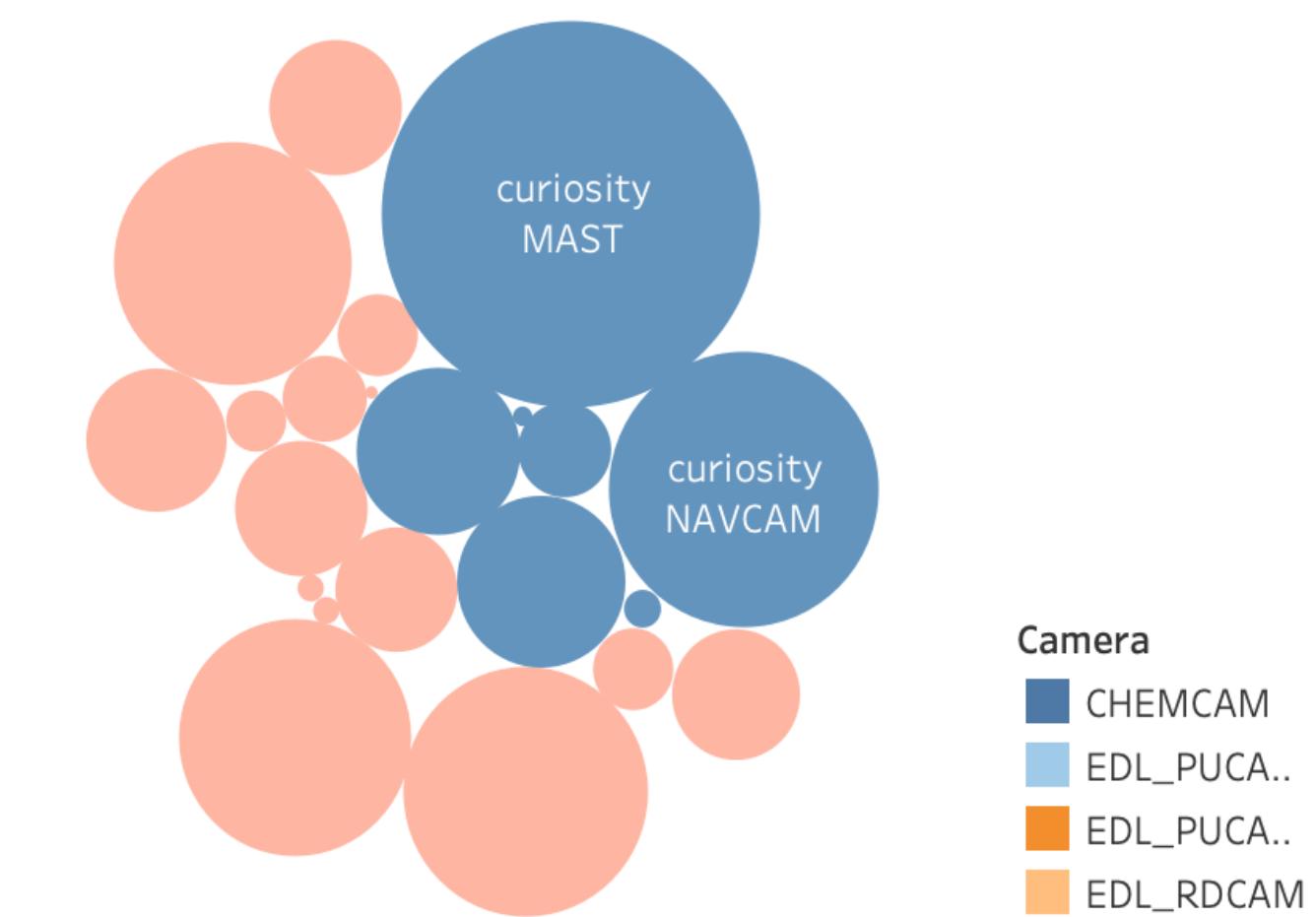
mars_photos_for_tableau.csv

CSV 文件

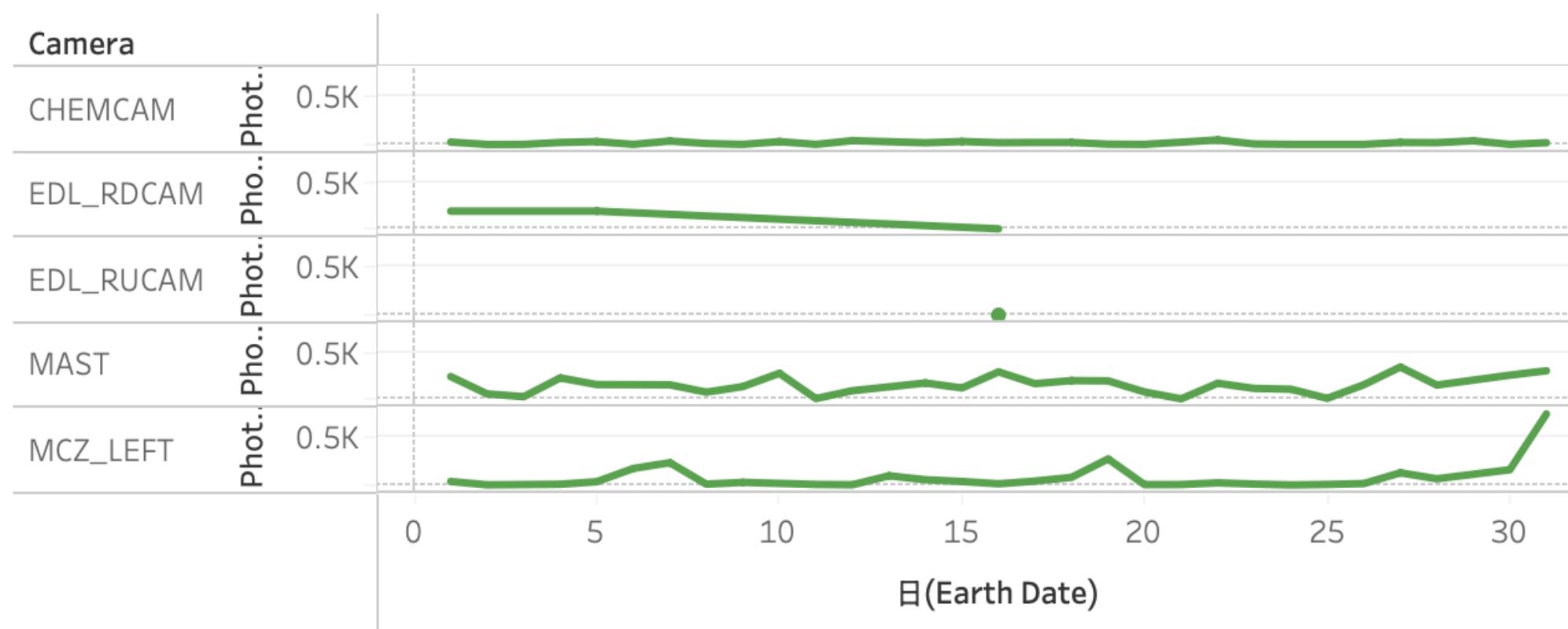
Number of Photos by Earth Date and Rover



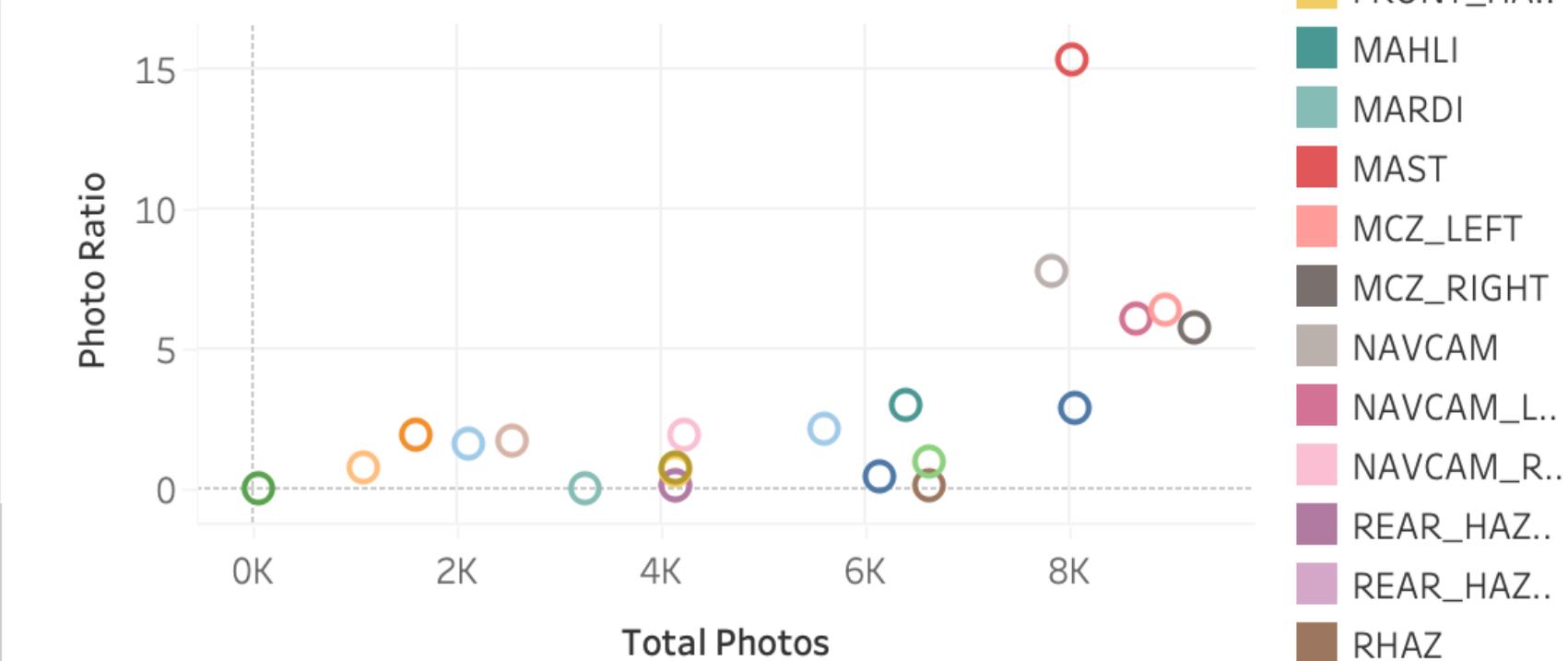
Proportion of Photos Taken by Each Camera



Camera Performance Over Time



Daily Photo Count vs. Camera Ratio Correlation



THANK YOU

GitHub Link:

https://github.com/VictoriaChueh/Business-Intelligence-Final_ETL-Project.git

