IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Fei J
02/11/2023

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Executive Summary

- **Summary of methodologies**

  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction

- **Summary of all results**

  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

Introduction

- **Project background and context**

SpaceX offers Falcon 9 rocket launches at a cost of $62 million, significantly lower than other providers, primarily due to their ability to reuse the first stage. Accurately predicting whether the first stage will successfully land is crucial for cost estimation in rocket launches. This information can be valuable for competitors looking to bid against SpaceX for rocket launch contracts. The project's objective is to develop a machine learning pipeline for predicting the successful landing of the first stage, thus enabling more accurate cost projections for launch bids.

- Problems you want to find answers

1. What are the determinants of a successful rocket landing?

- Understanding the interplay of various elements that contribute to the likelihood of a successful landing.

2. What operational prerequisites must be met to guarantee a successful rocket landing program?

- Identifying the specific conditions and requirements necessary for the success of a rocket landing program.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:

Data was collected by SpaceX API and web scraping from Wikipedia.

- Perform data wrangling

One-hot encoding was applied to categorical features

- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- Describe how data sets were collected.

Step 1: SpaceX API Data Collection

GET request to SpaceX API & API response decoded as JSON

Step 2: Transforming Data into Pandas DataFrame

Transform JSON into pandas dataframe & .json_normalize()

Step 3: Data Cleaning and Missing Value Handling

Data cleaning & Check for missing values & Fill in missing values

Step 4: Web Scraping from Wikipedia

Web scraping from Wikipedia & BeautifulSoup

Step 5: Parsing and Converting Web Scraped Data

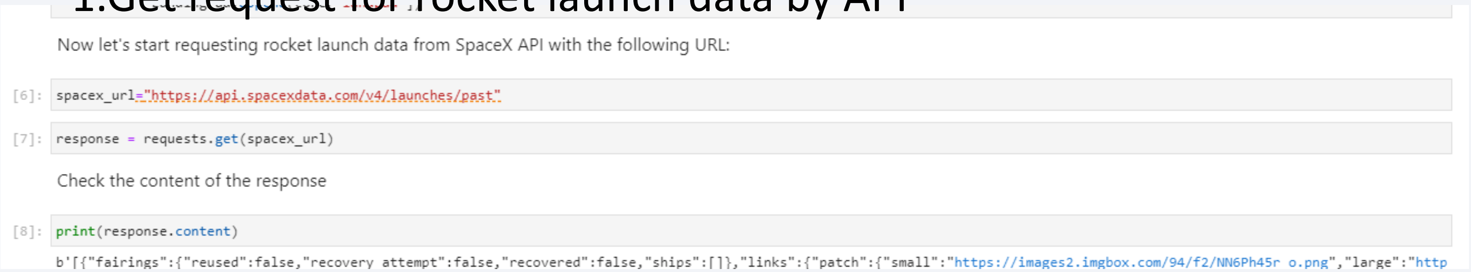
Extract launch records as HTML table & Parse HTML table & Convert to pandas dataframe

Data Collection – SpaceX API

- Present your data collection with SpaceX REST calls using key phrases and flowcharts

- https://github.com/GoRushB/Winning-Space-Race-with-Data-Science/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

1.Get request for rocket launch data by API



Now let's start requesting rocket launch data from SpaceX API with the following URL:

```
[6]: spacex_url="https://api.spacexdata.com/v4/launches/past"
```

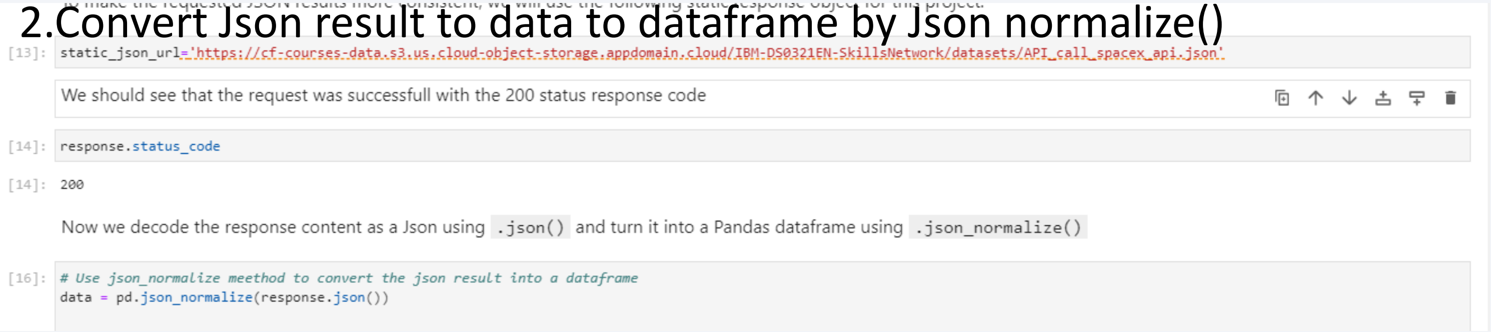
```
[7]: response = requests.get(spacex_url)
```

Check the content of the response

```
[8]: print(response.content)
```

```
b'[{"fairings":{"reused":false,"recovery attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://images2.imgbox.com/94/f2/NN6Ph45r o.png","large":"http
```

2.Convert Json result to data to dataframe by Json normalize()



```
[13]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

We should see that the request was successfull with the 200 status response code

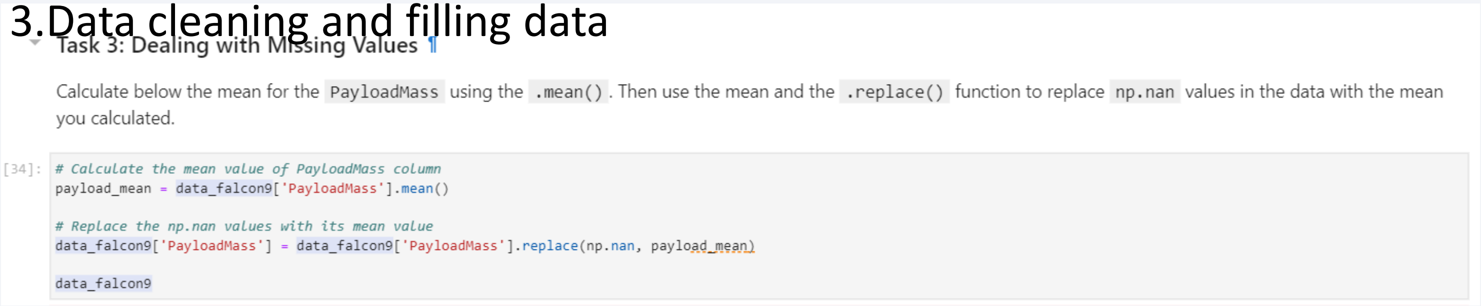
```
[14]: response.status_code
```

```
[14]: 200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
[16]: # Use json_normalize meethod to convert the json result into a dataframe
      data = pd.json_normalize(response.json())
```

3.Data cleaning and filling data



Task 3: Dealing with Missing Values

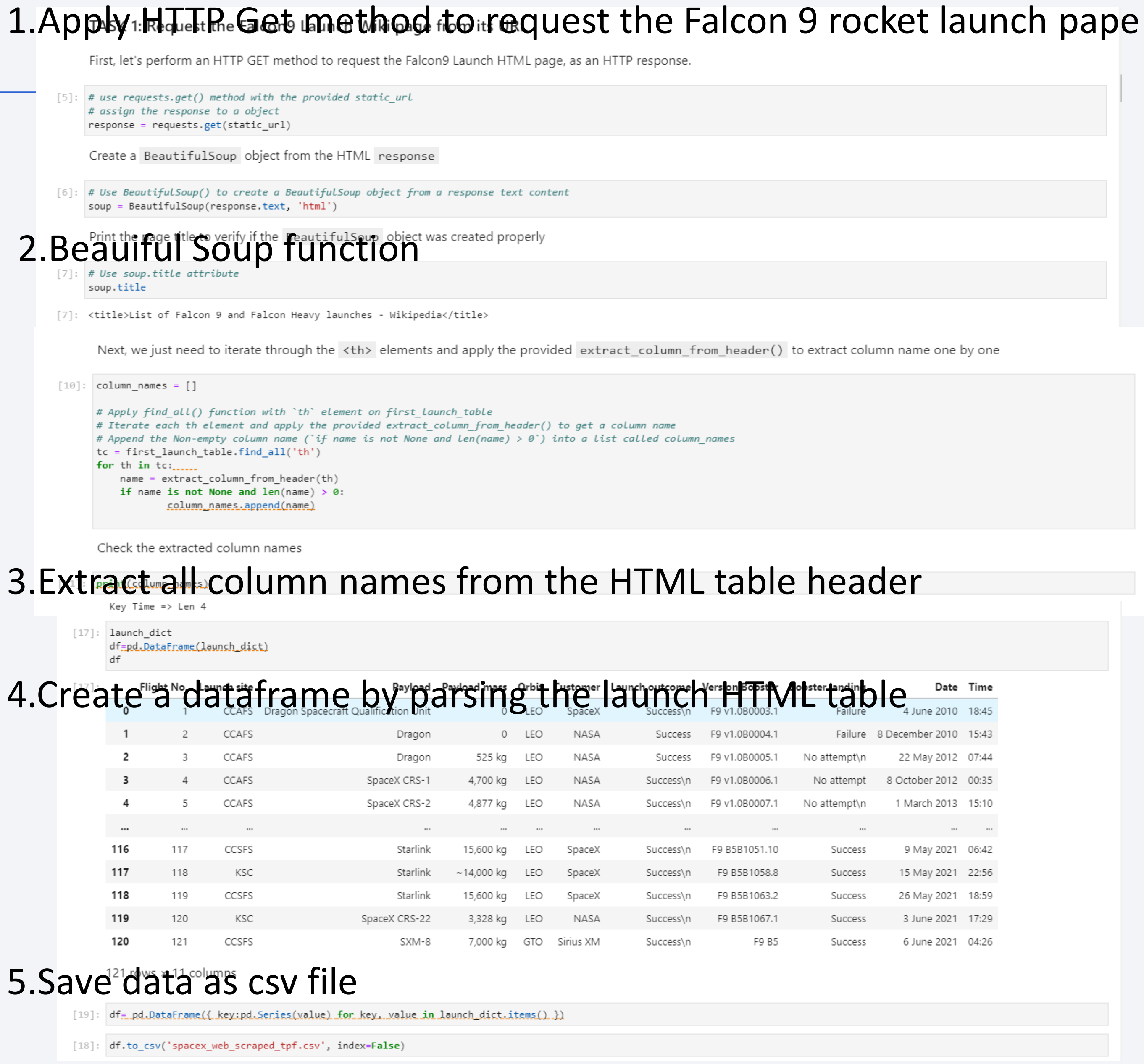
Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

```
[34]: # Calculate the mean value of PayloadMass column
      payload_mean = data_falcon9['PayloadMass'].mean()

      # Replace the np.nan values with its mean value
      data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, payload_mean)

      data_falcon9
```

Data Collection - Scraping

- Present your web scraping process using key phrases and flowcharts

- https://github.com/GoRushB/Winning-Space-Race-with-Data-Science/blob/main/jupyter-labs-webscraping.ipynb



1.Apply HTTP Get method to request the Falcon 9 rocket launch pape

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
[5]: # use requests.get() method with the provided static_url
     # assign the response to a object
     response = requests.get(static_url)
```

Create a `BeautifulSoup` object from the HTML `response`

```
[6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
     soup = BeautifulSoup(response.text, 'html')
```

2.Beauiful Soup function

Print the page title to verify if the `BeautifulSoup` object was created properly

```
[7]: # Use soup.title attribute
     soup.title

[7]: <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

Next, we just need to iterate through the `<th>` elements and apply the provided `extract_column_from_header()` to extract column name one by one

```
[10]: column_names = []

      # Apply find_all() function with `th` element on first_launch_table
      # Iterate each th element and apply the provided extract_column_from_header() to get a column name
      # Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
      tc = first_launch_table.find_all('th')
      for th in tc:
          name = extract_column_from_header(th)
          if name is not None and len(name) > 0:
                column_names.append(name)
```

Check the extracted column names

3.Extract all column names from the HTML table header

```
      print(column_names)

      Key Time => Len 4

[17]: launch_dict
      df=pd.DataFrame(launch_dict)
      df
```

4.Create a dataframe by parsing the launch HTML table

[17]:

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.0B0003.1	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10
...	...	...	...	...	...	...	...	...	...	...	...
116	117	CCSFS	Starlink	15,600 kg	LEO	SpaceX	Success\n	F9 B5B1051.10	Success	9 May 2021	06:42
117	118	KSC	Starlink	~14,000 kg	LEO	SpaceX	Success\n	F9 B5B1058.8	Success	15 May 2021	22:56
118	119	CCSFS	Starlink	15,600 kg	LEO	SpaceX	Success\n	F9 B5B1063.2	Success	26 May 2021	18:59
119	120	KSC	SpaceX CRS-22	3,328 kg	LEO	NASA	Success\n	F9 B5B1067.1	Success	3 June 2021	17:29
120	121	CCSFS	SXM-8	7,000 kg	GTO	Sirius XM	Success\n	F9 B5	Success	6 June 2021	04:26

121 rows × 11 columns

5.Save data as csv file

```
[19]: df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })

[18]: df.to_csv('spacex_web_scraped_tpf.csv', index=False)
```

Data Wrangling

Step 1: Exploratory Data Analysis (EDA)

Key Phrases:

"Performed EDA"

"Explored dataset"

"Data visualization“

Step 2: Determining Training Labels

Key Phrases:

"Determined training labels“

Step 3: Calculating Launch Site and Orbit Statistics

Key Phrases:

"Calculated launch site statistics"

"Analyzed orbit data“

Link:

https://github.com/GoRushB/Winning-Space-Race-with-Data-Science/blob/main/Data%20Wrangling.ipynb

Step 4: Creating Landing Outcome Labels

Key Phrases:

"Created landing outcome label"

"Derived labels from outcome column“

Step 5: Exporting Results to CSV

Key Phrases:

"Exported results to CSV"

EDA with Data Visualization

Certainly, here's a concise representation of the data exploration tasks:

1. Visualized Flight Number and Launch Site Relationship:

- Explored the relationship between flight number and launch site.

2. Visualized Payload and Launch Site Relationship:

- Examined the connection between payload and launch site.

3. Analyzed Success Rate of Each Orbit Type:

- Investigated the success rate of different orbit types.

4. Explored Flight Number and Orbit Type Relationship

- Studied the relationship between flight number and orbit type.

5. Examined Yearly Launch Success Trends

- Explored trends in launch success on a yearly basis.

Link: https://github.com/GoRushB/Winning-Space-Race-with-Data-Science/blob/main/EDA%20with%20Data%20Visualization.ipynb

EDA with SQL

Certainly, here's a more concise representation of the data loading and SQL-based exploratory data analysis tasks:

1. Loaded SpaceX Dataset into PostgreSQL Database

- Imported the SpaceX dataset into a PostgreSQL database seamlessly within Jupyter Notebook.

2. Applied EDA with SQL

- Utilized SQL queries to gain insights from the data, including:

1) Obtaining unique launch site names.

2) Calculating the total payload mass carried by NASA (CRS) boosters.

3) Calculating the average payload mass carried by booster version F9 v1.1.

4) Counting successful and failed mission outcomes.

5) Identifying failed landing outcomes on drone ships, along with their associated booster versions and launch site names.

Link: https://github.com/GoRushB/Winning-Space-Race-with-Data-Science/blob/main/EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

1. **Mapped Launch Sites with Folium**

- Marked launch sites on a map.
- Added markers, circles, and lines to represent launch success or failure.

2. Assigned Launch Outcomes to Classes

- Categorized launch outcomes as 0 (failure) and 1 (success).

3. Identified High Success Rate Sites

- Used color-labeled markers to identify launch sites with high success rates.

4. Calculated Distances and Answered Questions

- Measured distances between launch sites and nearby features, including railways, highways, coastlines, and cities.
- Investigated if launch sites maintain specific distances from cities.

Link: https://github.com/GoRushB/Winning-Space-Race-with-Data-Science/blob/main/Interactive%20Visual%20Analytics%20with%20Folium%20lab.ipynb

Build a Dashboard with Plotly Dash

1. Created Interactive Dashboard with Plotly Dash

- Developed an interactive dashboard using Plotly Dash.

2. Generated Pie Charts for Total Launches by Sites

- Created pie charts to visualize the total number of launches at specific sites.

3. Produced Scatter Graphs to Analyze Outcome vs. Payload Mass

- Used scatter graphs to analyze the relationship between launch outcomes and payload mass (in Kg) for different booster versions.

Link: https://github.com/GoRushB/Winning-Space-Race-with-Data-Science/blob/main/Plotly%20Dash.py

Predictive Analysis (Classification)

1. Loaded and Transformed Data

Loaded data using NumPy and Pandas & Transformed the data.

2. Data Splitting

Split the data into training and testing sets.

3. Machine Learning Model Building

Developed various machine learning models & Tuned hyperparameters using GridSearchCV.

4. Model Evaluation

Utilized accuracy as the metric for model evaluation & Improved the model through feature engineering and algorithm tuning.

5. Identified Best Performing Model

Determined the best-performing classification model.

Link: https://github.com/GoRushB/Winning-Space-Race-with-Data-Science/blob/main/Machine%20Learning%20Prediction.ipynb

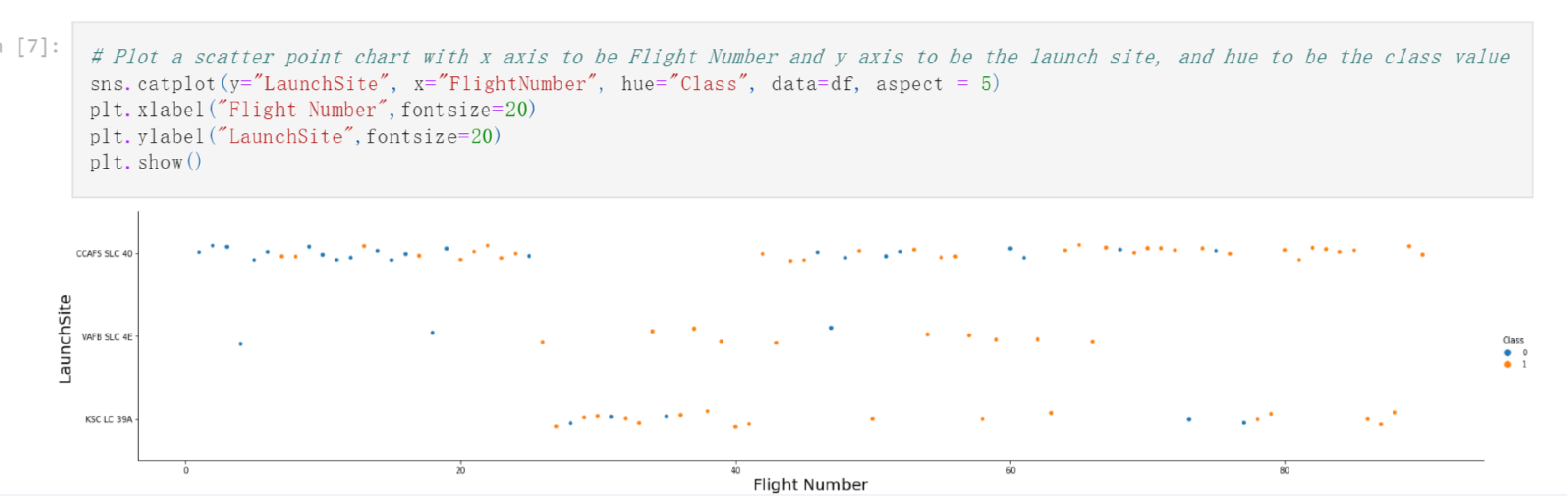
Results

- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

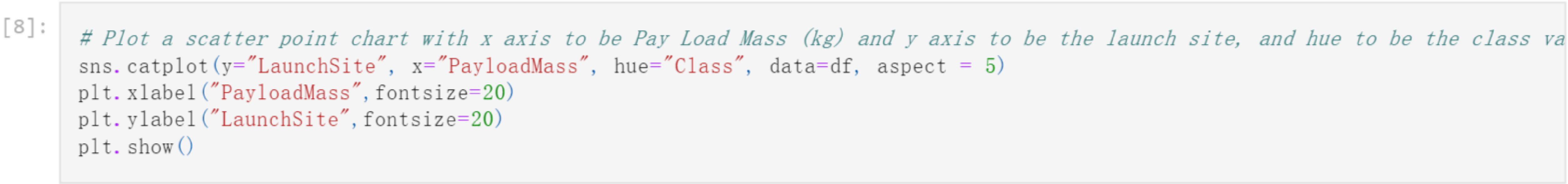


[7]:

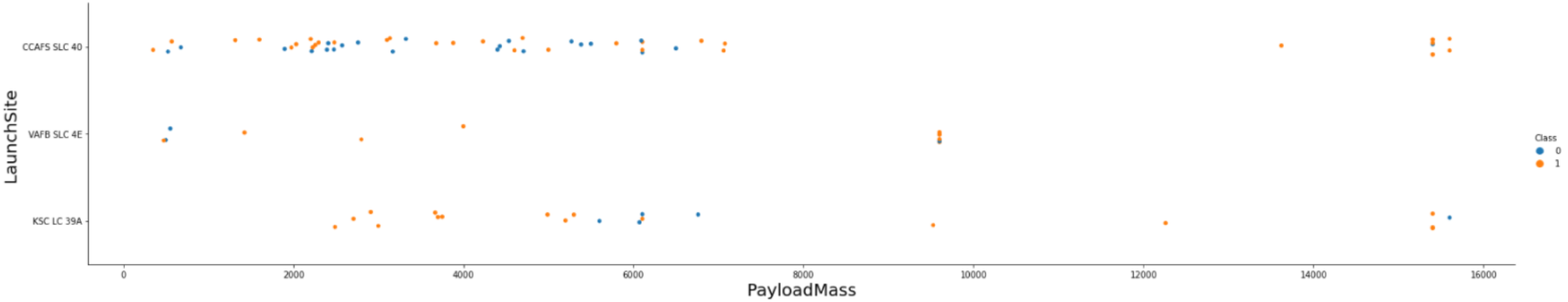
```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```

The larger the flight amount at a launch site, the greater the success rate at a launch site

Payload vs. Launch Site

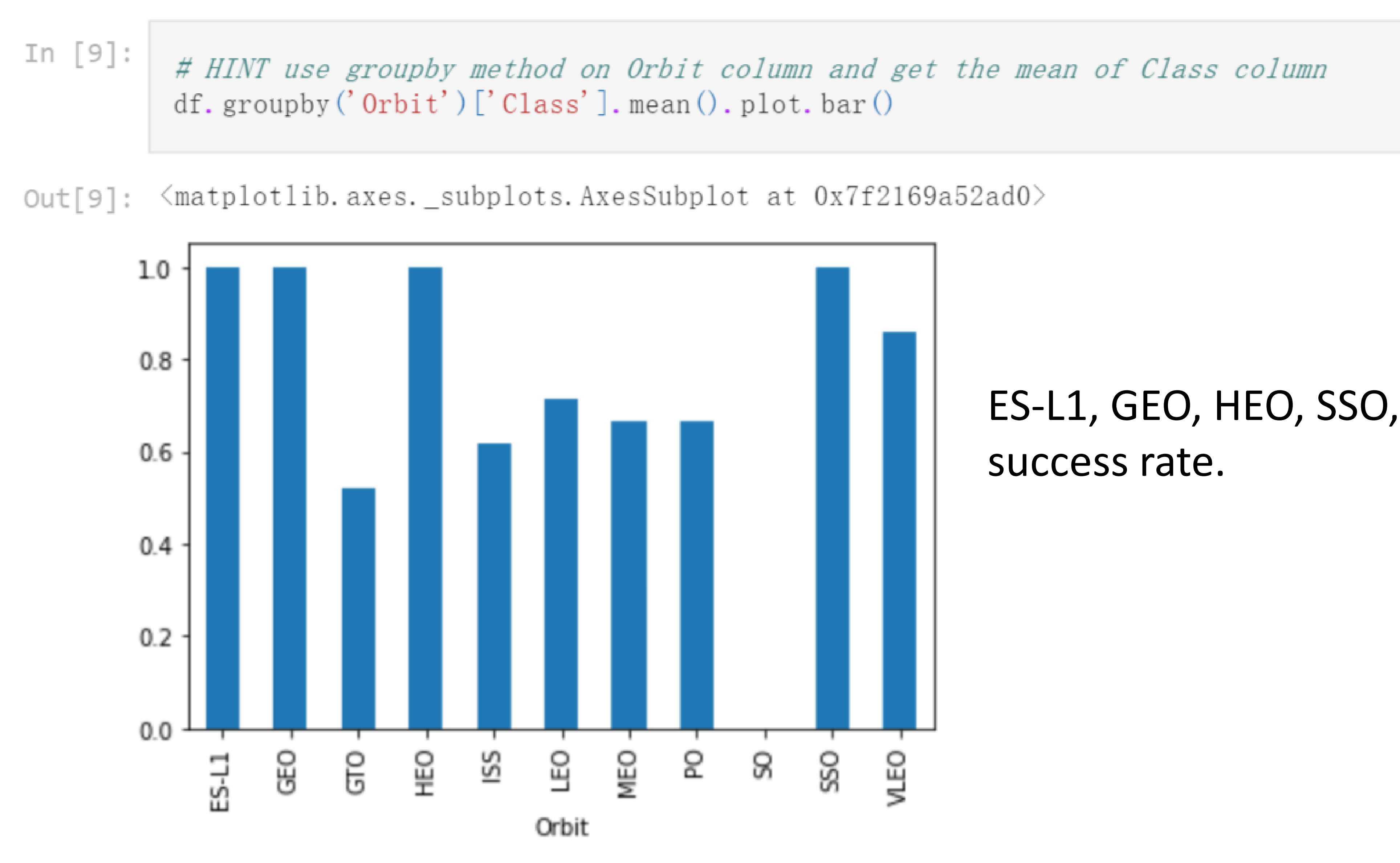


```
[8]:  # Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class va
      sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
      plt.xlabel("PayloadMass",fontsize=20)
      plt.ylabel("LaunchSite",fontsize=20)
      plt.show()
```



The greater the payload mass for launch site CCAFS SLC 40 the higher the success rate for the rocket.

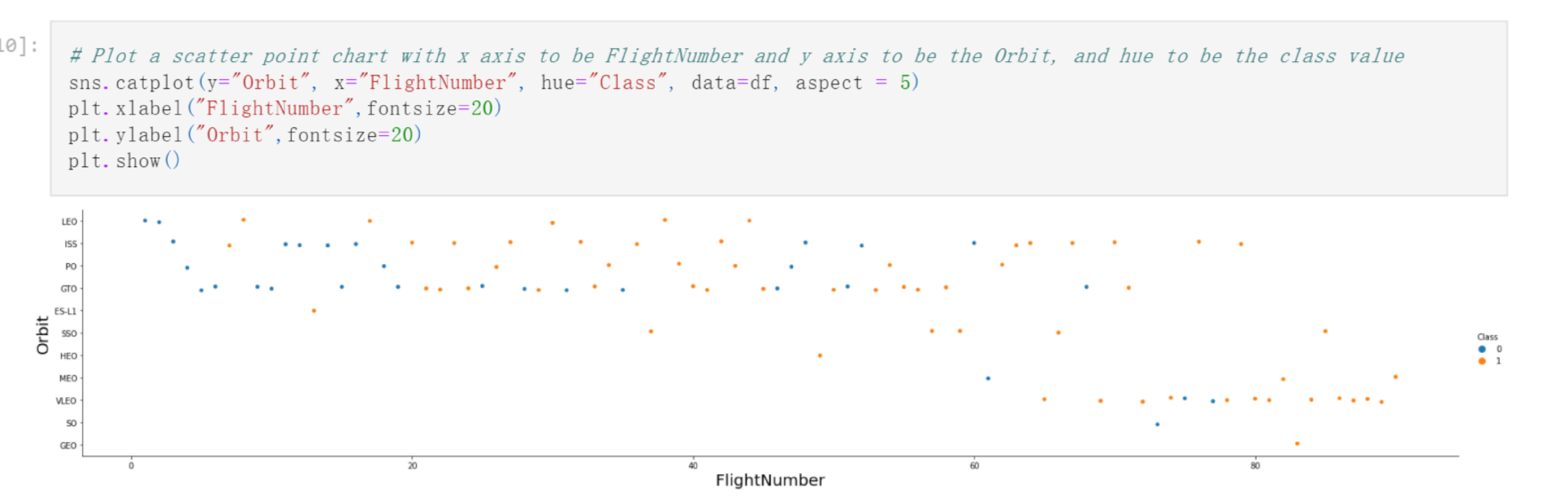
Success Rate vs. Orbit Type



```
In [9]:  # HINT use groupby method on Orbit column and get the mean of Class column
         df.groupby('Orbit')['Class'].mean().plot.bar()

Out[9]:  <matplotlib.axes._subplots.AxesSubplot at 0x7f2169a52ad0>
```

ES-L1, GEO, HEO, SSO, VLEO had the most success rate.

Flight Number vs. Orbit Type

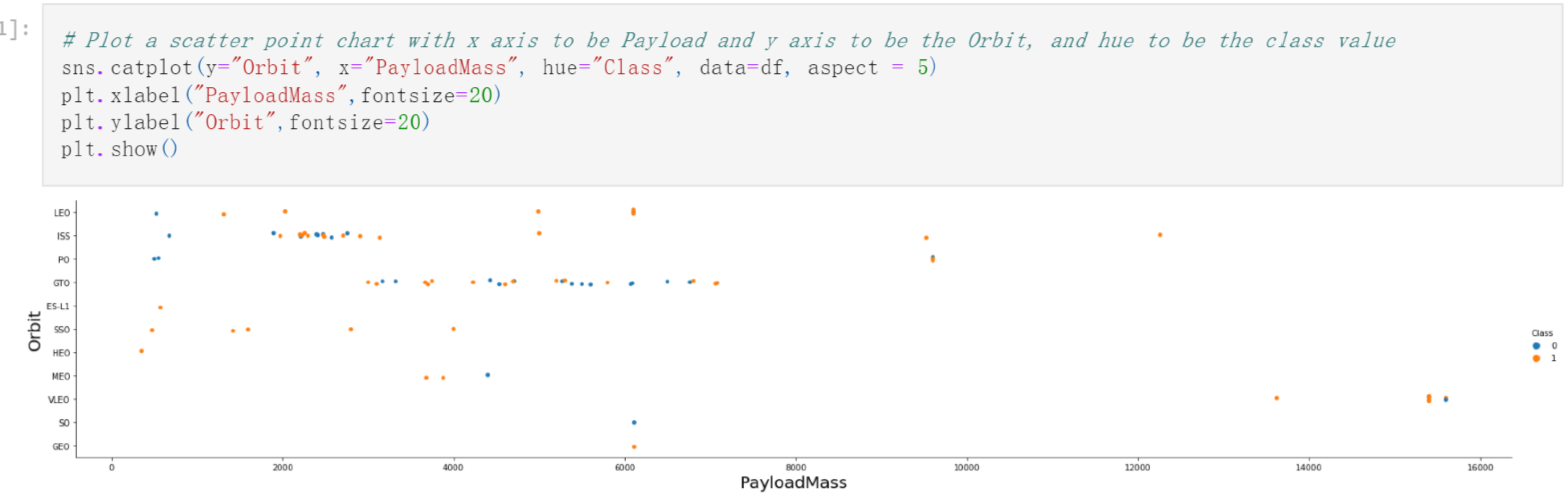


10]:

```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("FlightNumber",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```

In the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.

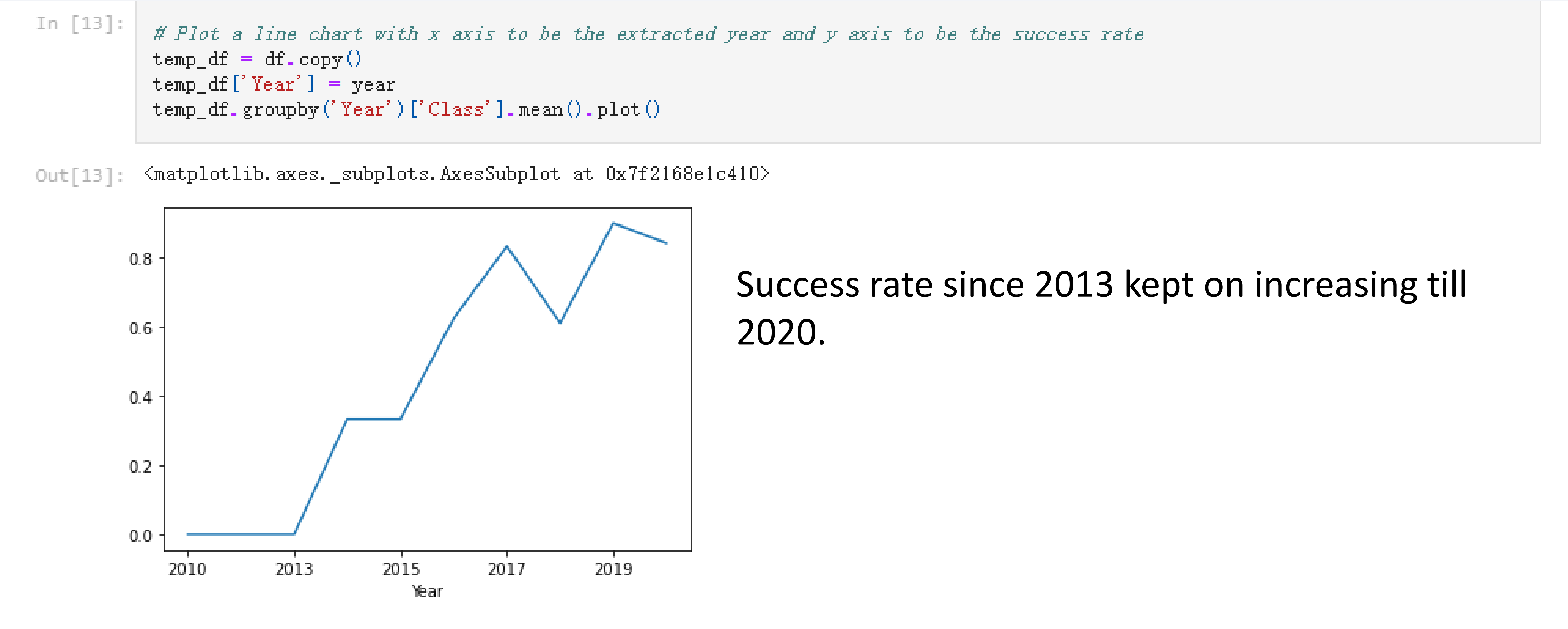
Payload vs. Orbit Type



```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("PayloadMass",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```

With heavy payloads, the successful landing are more for PO, LEO and ISS orbits.

Launch Success Yearly Trend



In [13]:

```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
temp_df = df.copy()
temp_df['Year'] = year
temp_df.groupby('Year')['Class'].mean().plot()
```

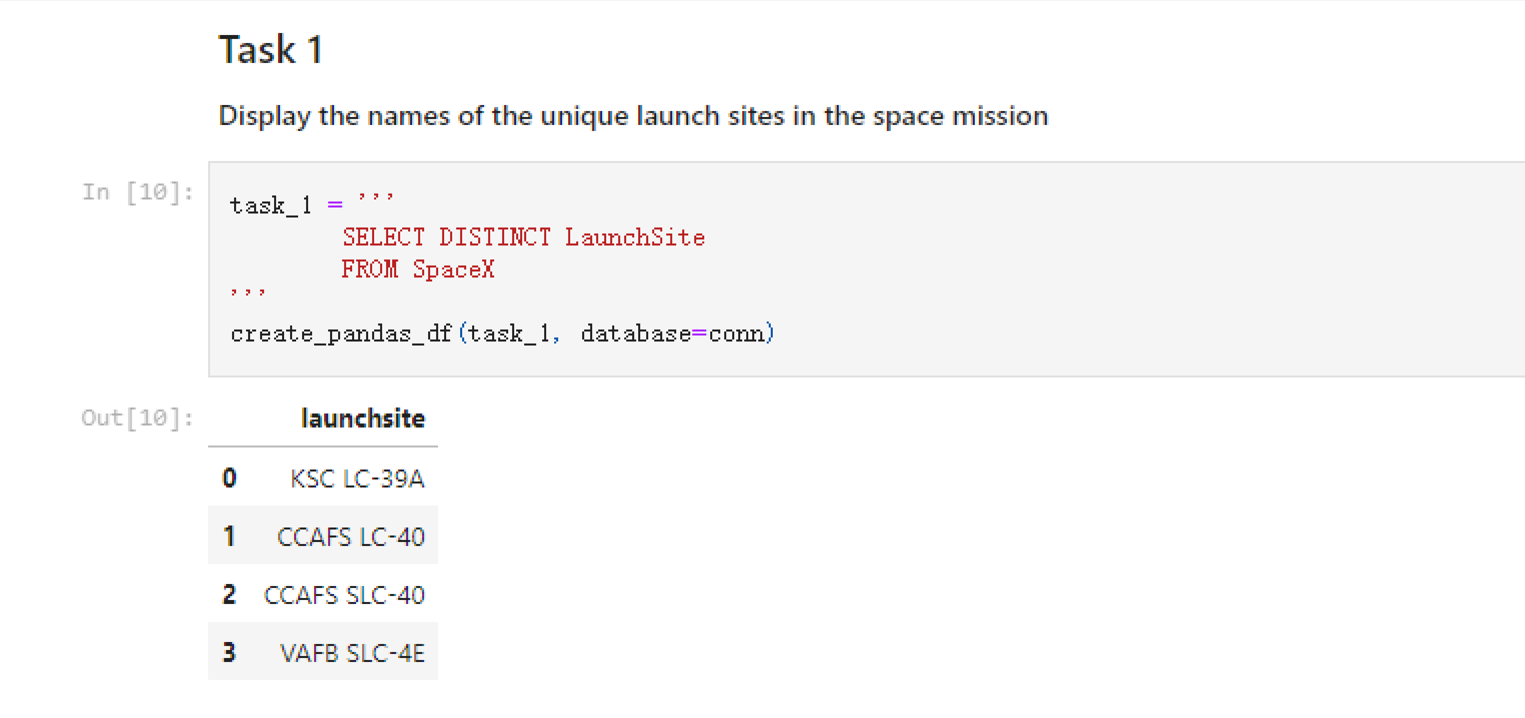
Out[13]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2168e1c410>
```

Success rate since 2013 kept on increasing till 2020.

All Launch Site Names

Used the key word **DISTINCT** to show only unique launch sites from the SpaceX data.



Task 1

Display the names of the unique launch sites in the space mission

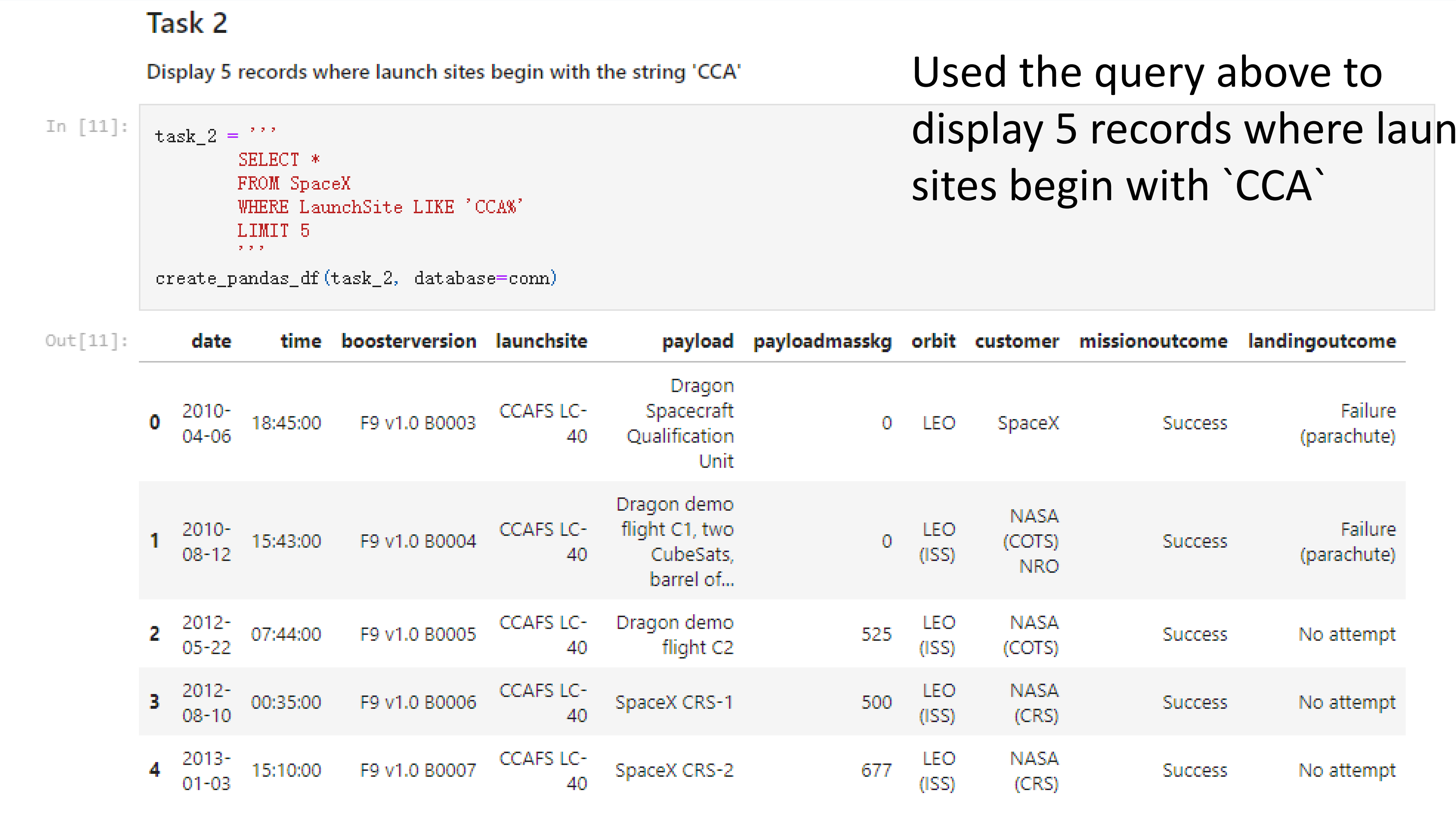
In [10]:

```
task_1 = '''
        SELECT DISTINCT LaunchSite
        FROM SpaceX
'''
create_pandas_df(task_1, database=conn)
```

Out[10]:

	launchsite
0	KSC LC-39A
1	CCAFS LC-40
2	CCAFS SLC-40
3	VAFB SLC-4E

Launch Site Names Begin with 'CCA'



Task 2

Display 5 records where launch sites begin with the string 'CCA'

In [11]:

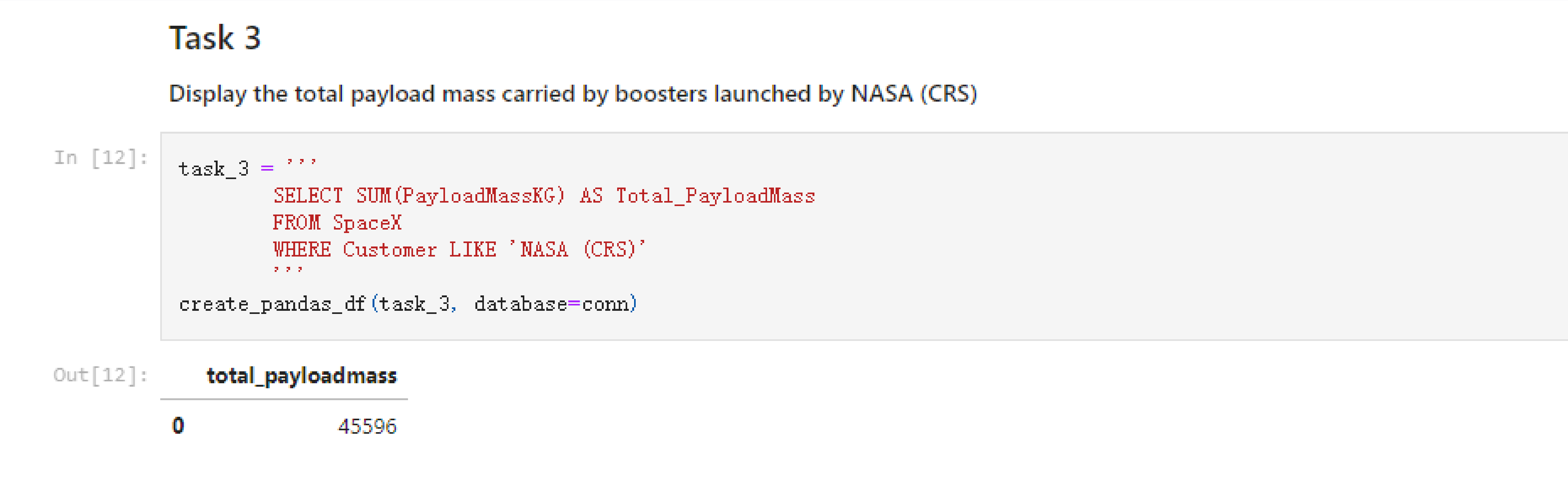
```
task_2 = '''
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'CCA%'
        LIMIT 5
        '''
create_pandas_df(task_2, database=conn)
```

Out[11]:

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Used the query above to display 5 records where launch sites begin with `CCA`

Total Payload Mass



Task 3

Display the total payload mass carried by boosters launched by NASA (CRS)

In [12]:

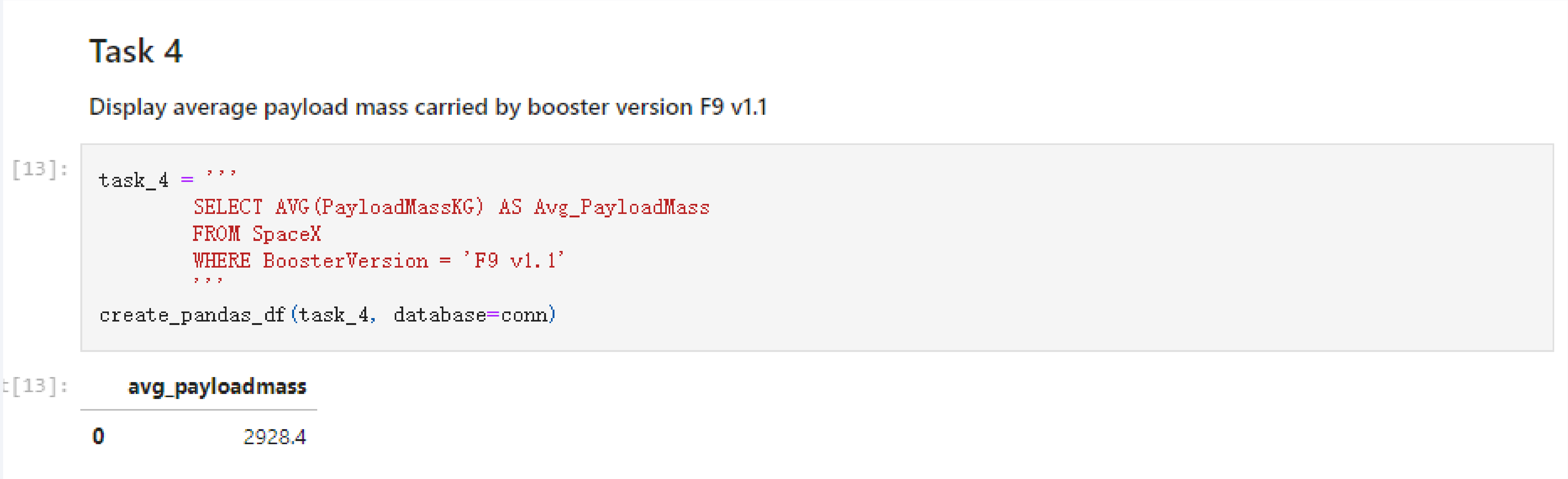
```
task_3 = '''
        SELECT SUM(PayloadMassKG) AS Total_PayloadMass
        FROM SpaceX
        WHERE Customer LIKE 'NASA (CRS)'
        '''
create_pandas_df(task_3, database=conn)
```

Out[12]:

	total_payloadmass
0	45596

The total payload carried by boosters from NASA as 45596 using the query below

Average Payload Mass by F9 v1.1



Task 4

Display average payload mass carried by booster version F9 v1.1

[13]:

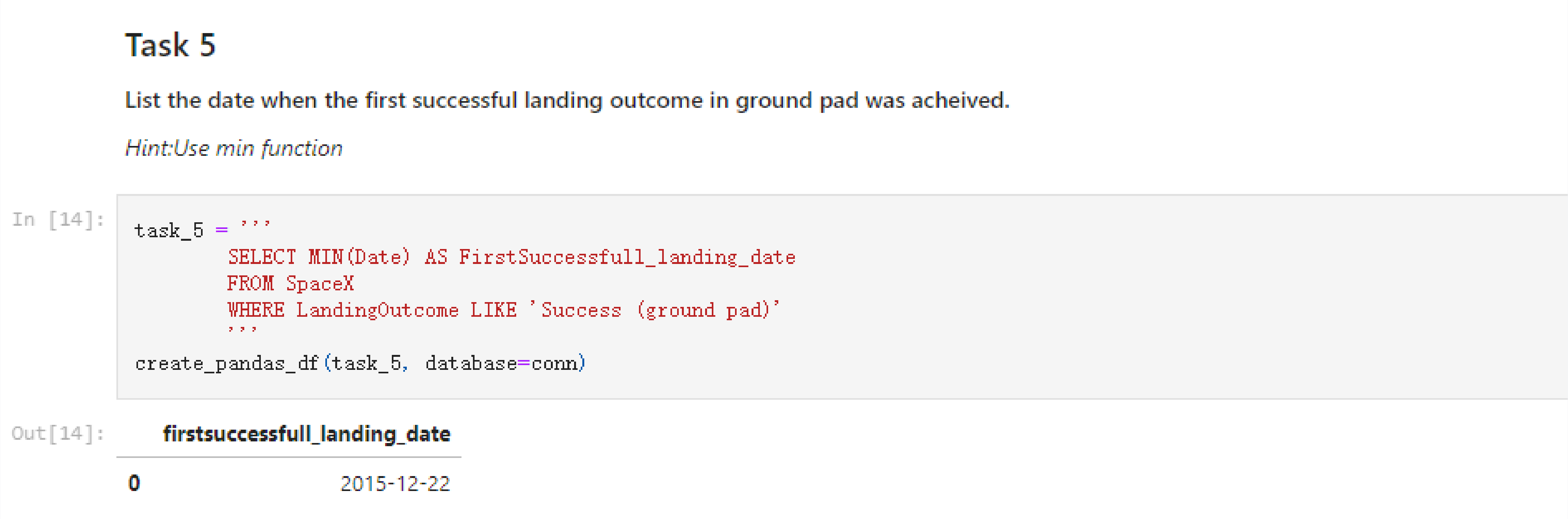
```
task_4 = '''
        SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
        FROM SpaceX
        WHERE BoosterVersion = 'F9 v1.1'
        '''
create_pandas_df(task_4, database=conn)
```

t[13]:

	avg_payloadmass
0	2928.4

Average payload mass is 2928.4

First Successful Ground Landing Date



Task 5

List the date when the first successful landing outcome in ground pad was acheived.

Hint:Use min function

In [14]:

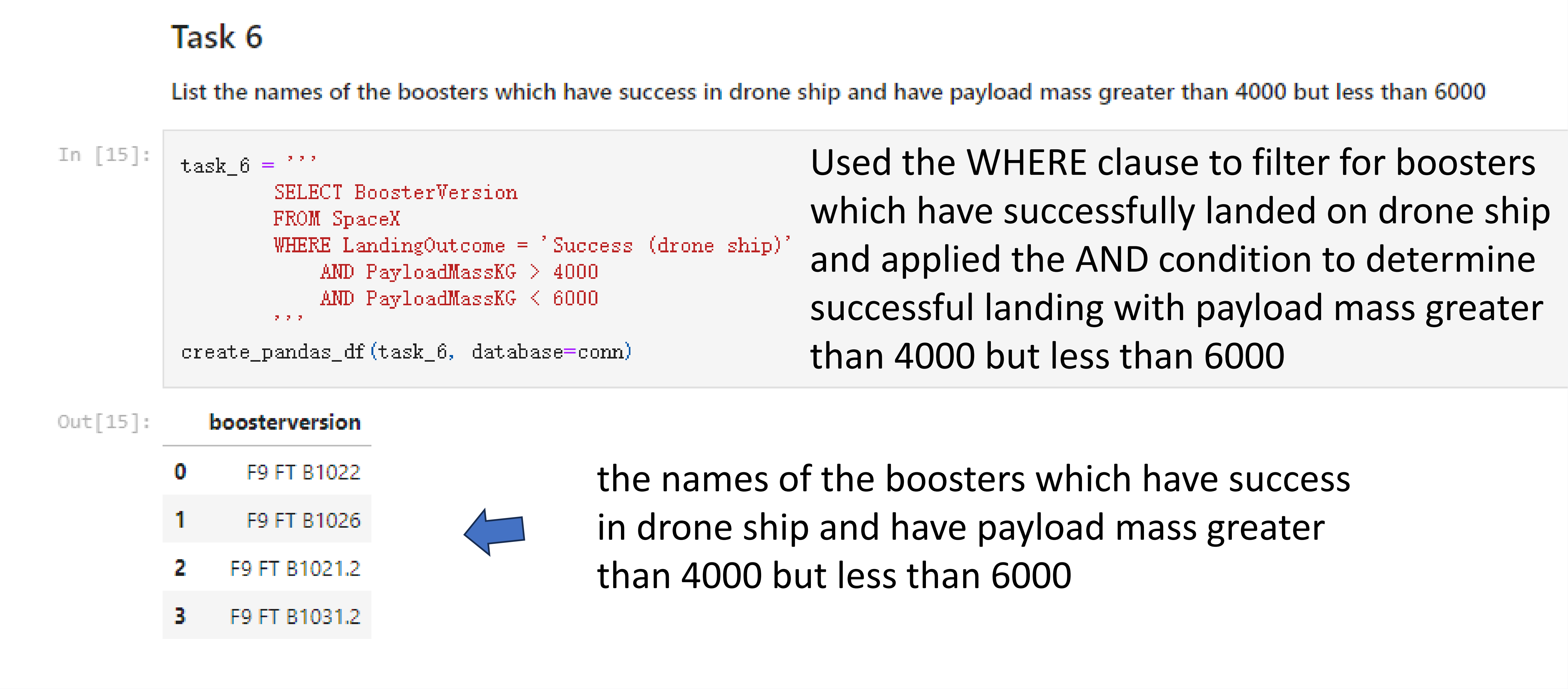
```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''
create_pandas_df(task_5, database=conn)
```

Out[14]:

```
   firstsuccessfull_landing_date
0                     2015-12-22
```

First successful ground landing data is 2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000



Task 6

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
In [15]:  task_6 = '''
                  SELECT BoosterVersion
                  FROM SpaceX
                  WHERE LandingOutcome = 'Success (drone ship)'
                      AND PayloadMassKG > 4000
                      AND PayloadMassKG < 6000
                  '''
          create_pandas_df(task_6, database=conn)
```

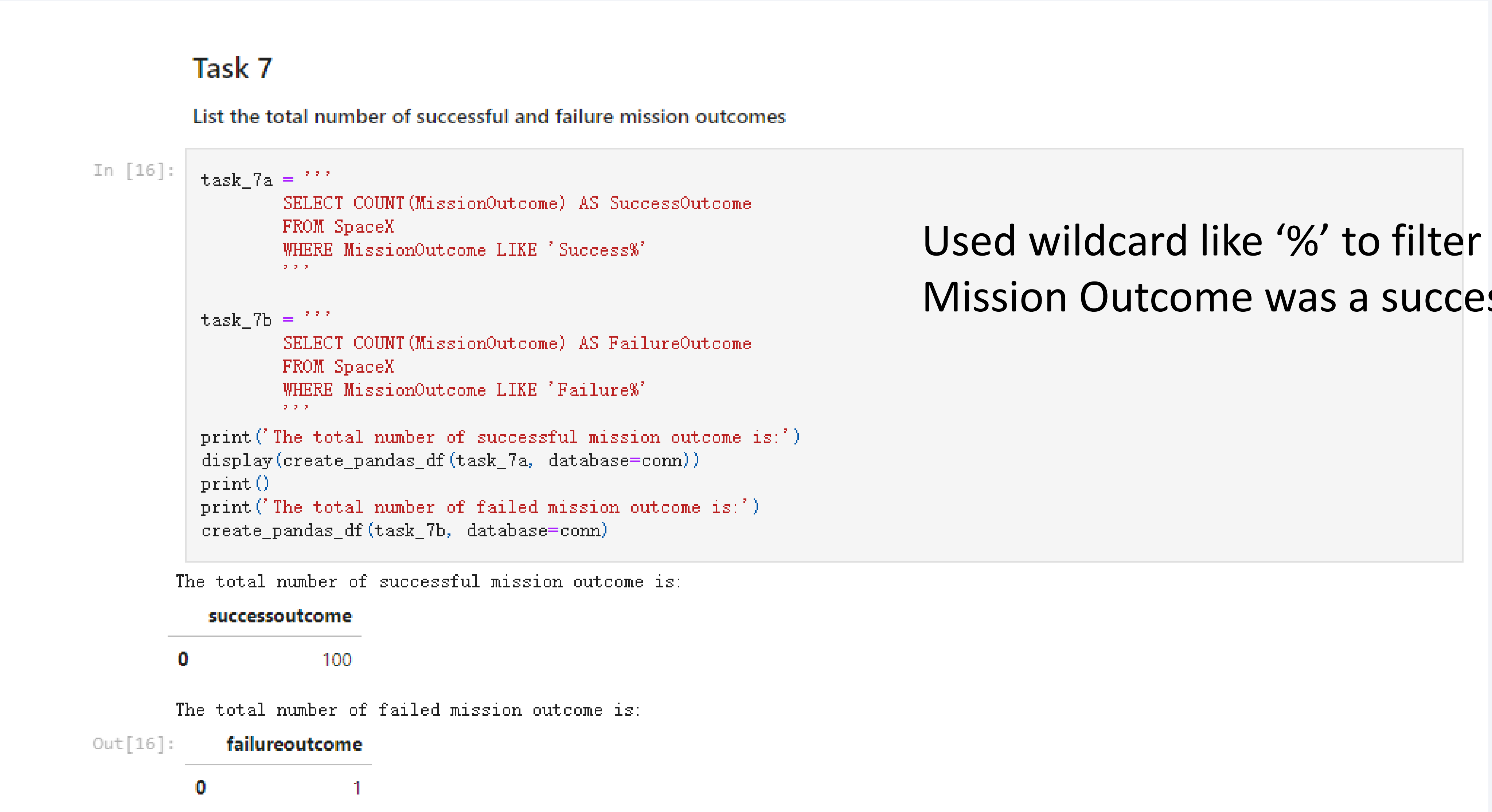
Out[15]:

	boosterversion
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

Used the WHERE clause to filter for boosters which have successfully landed on drone ship and applied the AND condition to determine successful landing with payload mass greater than 4000 but less than 6000

the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

Total Number of Successful and Failure Mission Outcomes



Task 7

List the total number of successful and failure mission outcomes

```
In [16]:
task_7a = '''
        SELECT COUNT(MissionOutcome) AS SuccessOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Success%'
        '''

task_7b = '''
        SELECT COUNT(MissionOutcome) AS FailureOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Failure%'
        '''
print('The total number of successful mission outcome is:')
display(create_pandas_df(task_7a, database=conn))
print()
print('The total number of failed mission outcome is:')
create_pandas_df(task_7b, database=conn)
```

```
The total number of successful mission outcome is:
```

	successoutcome
0	100

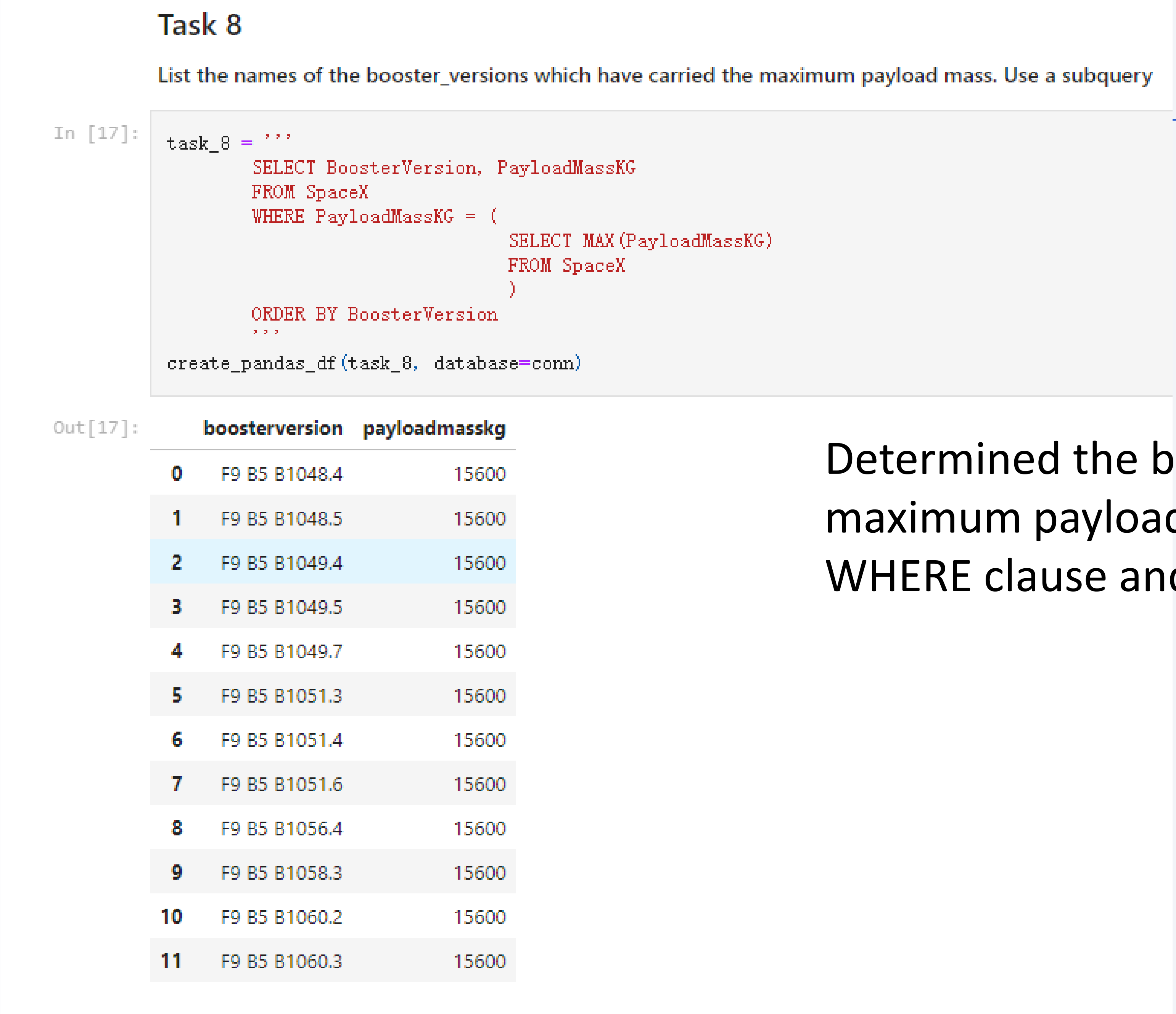
```
The total number of failed mission outcome is:
```

Out[16]:

	failureoutcome
0	1

Used wildcard like '%' to filter for WHERE Mission Outcome was a success or a failure.

Boosters Carried Maximum Payload



Task 8

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

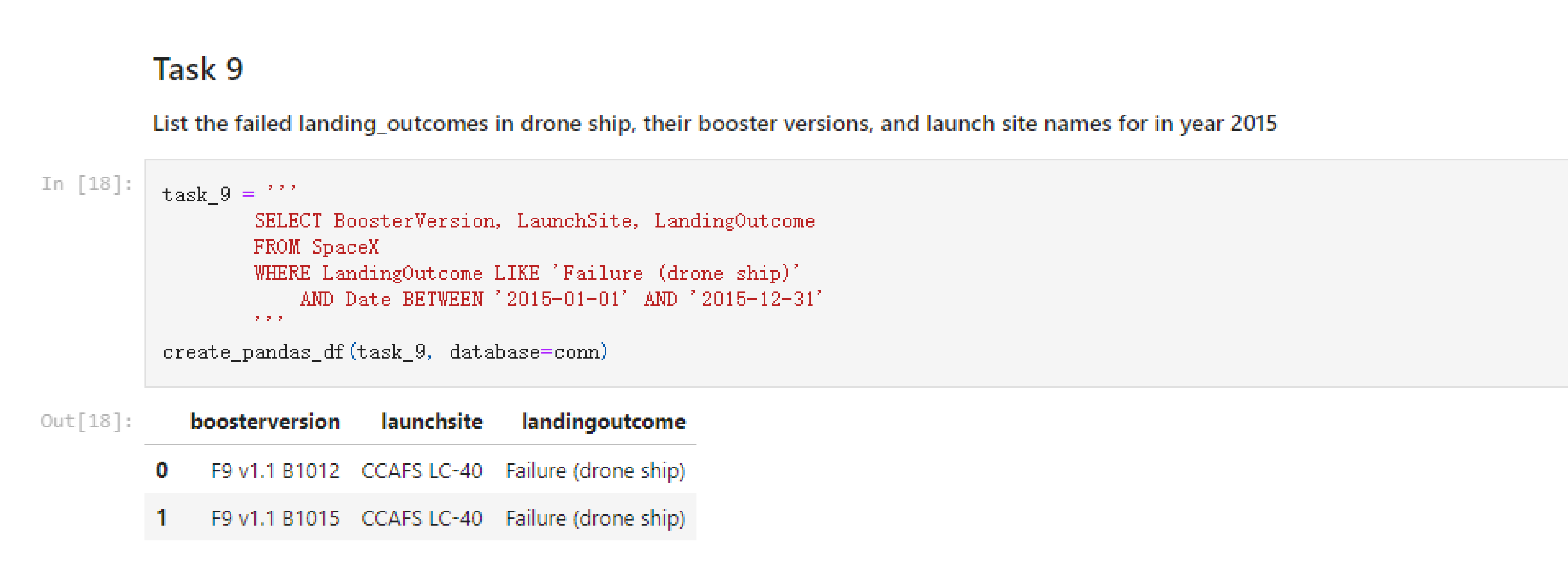
```
In [17]:
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, database=conn)
```

Out[17]:

	boosterversion	payloadmasskg
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600
5	F9 B5 B1051.3	15600
6	F9 B5 B1051.4	15600
7	F9 B5 B1051.6	15600
8	F9 B5 B1056.4	15600
9	F9 B5 B1058.3	15600
10	F9 B5 B1060.2	15600
11	F9 B5 B1060.3	15600

Determined the booster that have carried the maximum payload using a subquery in the WHERE clause and the MAX() function.

2015 Launch Records



Task 9

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

In [18]:

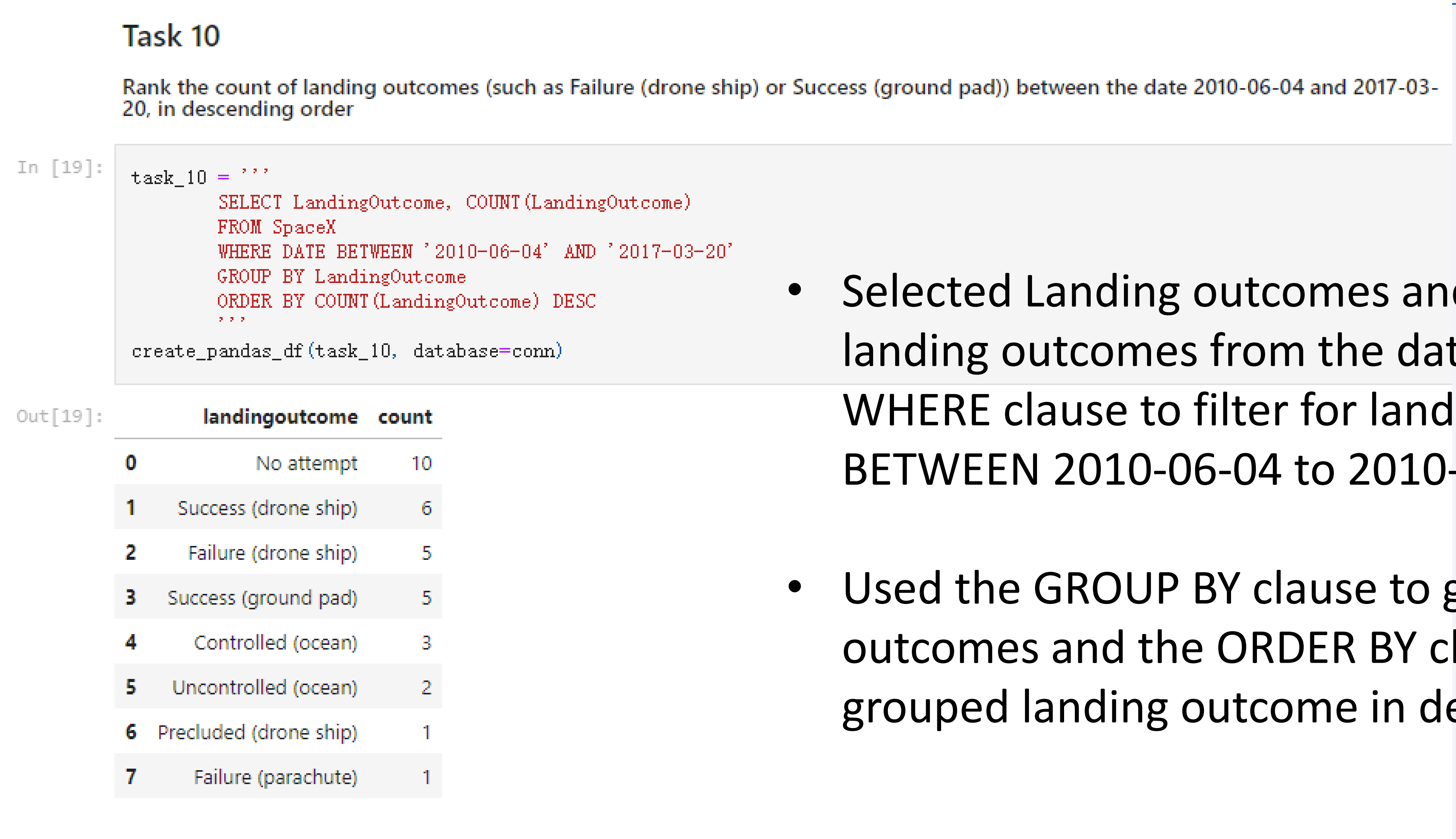
```
task_9 = '''
        SELECT BoosterVersion, LaunchSite, LandingOutcome
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'
        '''
create_pandas_df(task_9, database=conn)
```

Out[18]:

	boosterversion	launchsite	landingoutcome
0	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
1	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

Combine the WHERE clause, LIKE, AND, and BETWEEN conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

Rank Landing Outcomes Between 2010-06-04 and 2017-03-20



Task 10

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

In [19]:

```
task_10 = '''
        SELECT LandingOutcome, COUNT(LandingOutcome)
        FROM SpaceX
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY LandingOutcome
        ORDER BY COUNT(LandingOutcome) DESC
        '''
create_pandas_df(task_10, database=conn)
```

Out[19]:

	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

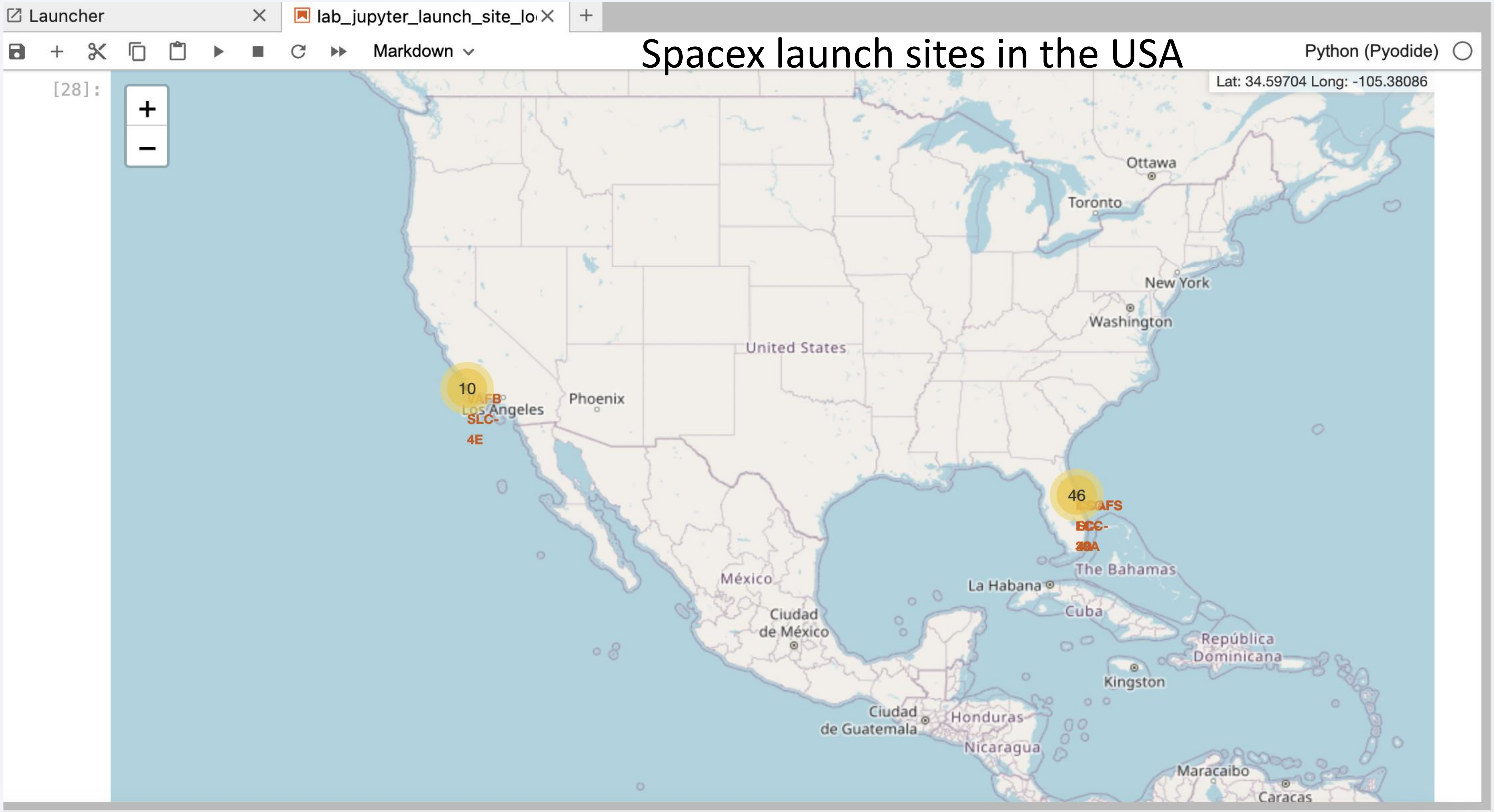
- Selected Landing outcomes and the COUNT of landing outcomes from the data and used the WHERE clause to filter for landing outcomes BETWEEN 2010-06-04 to 2010-03-20.

- Used the GROUP BY clause to group the landing outcomes and the ORDER BY clause to order the grouped landing outcome in descending order.

Section 3

Launch Sites Proximities Analysis

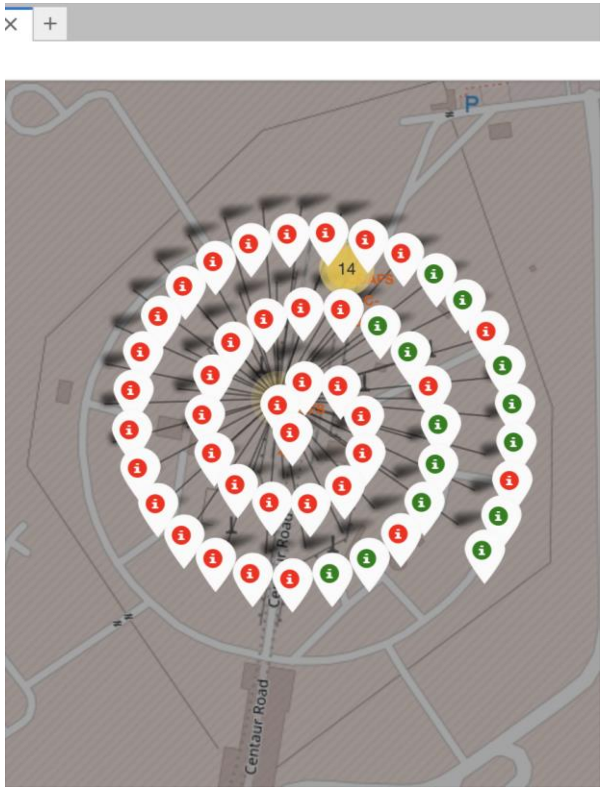
<Folium Map Screenshot 1>

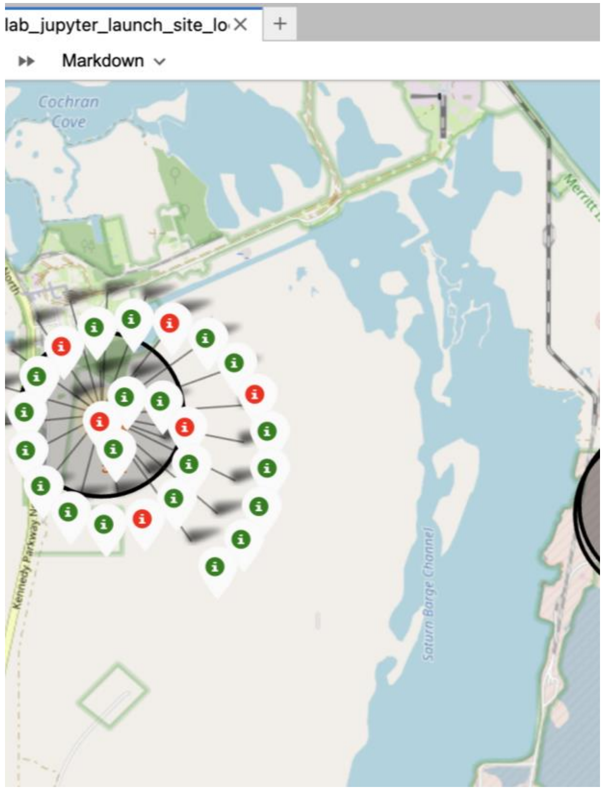


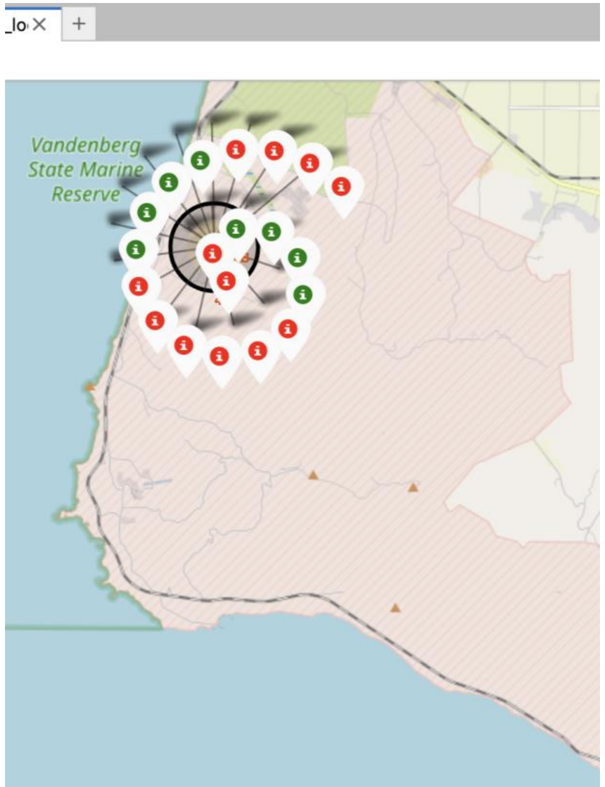
Spacex launch sites in the USA

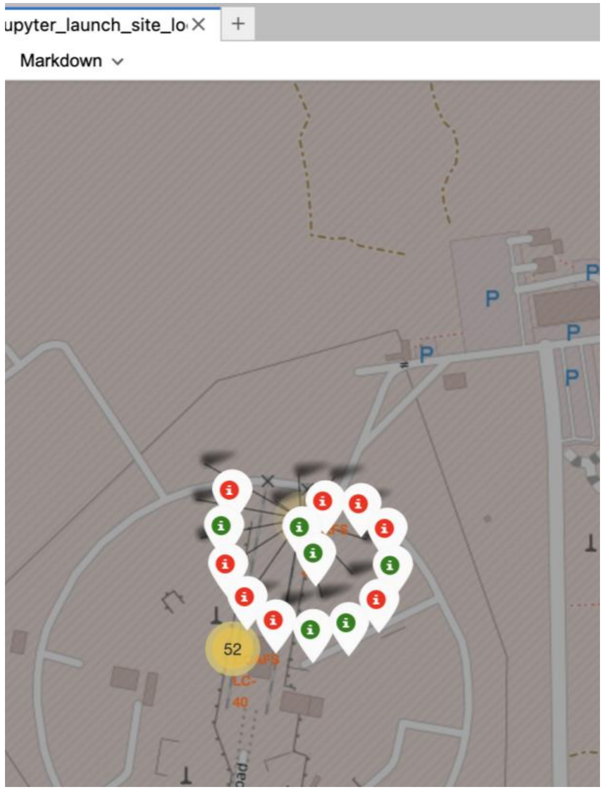
<Folium Map Screenshot 2>

Green marker: successfully launch
Red marker: unsuccessfully launch

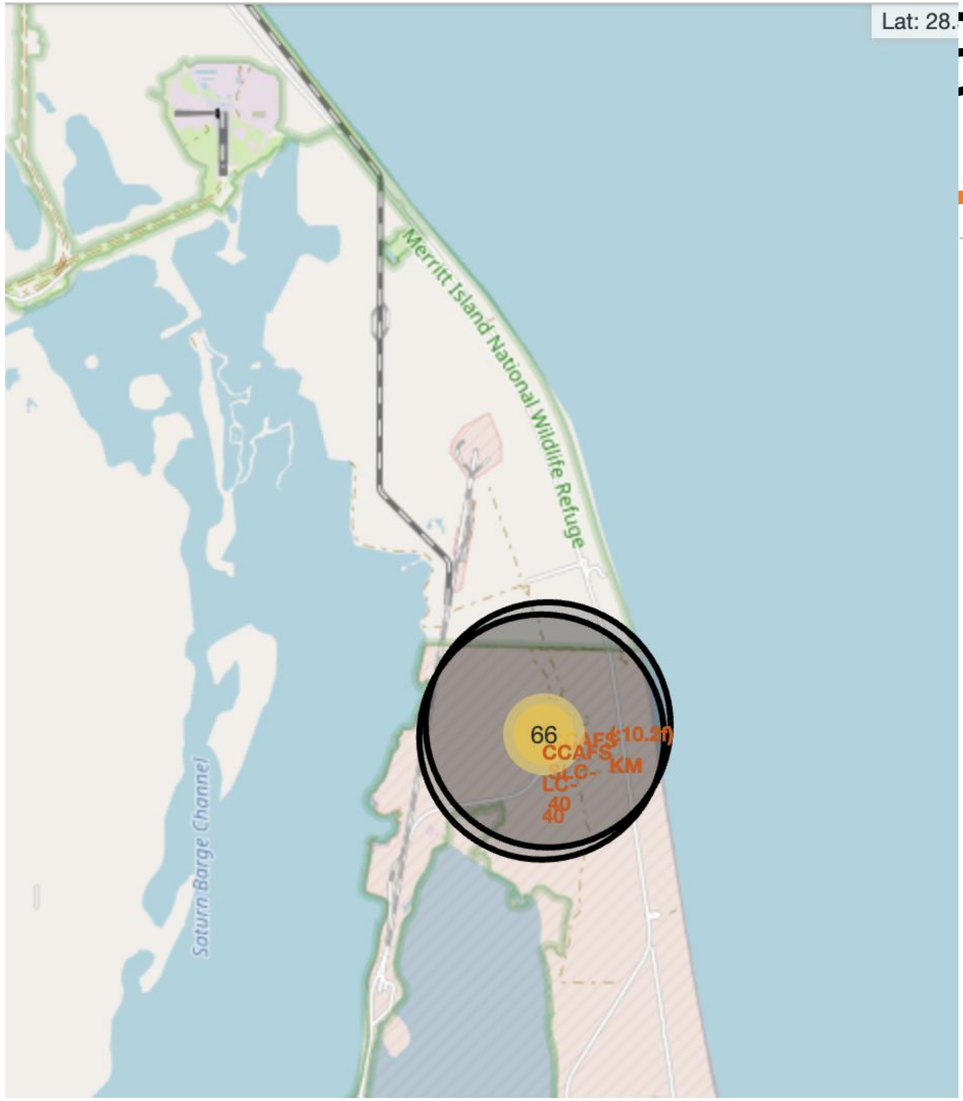


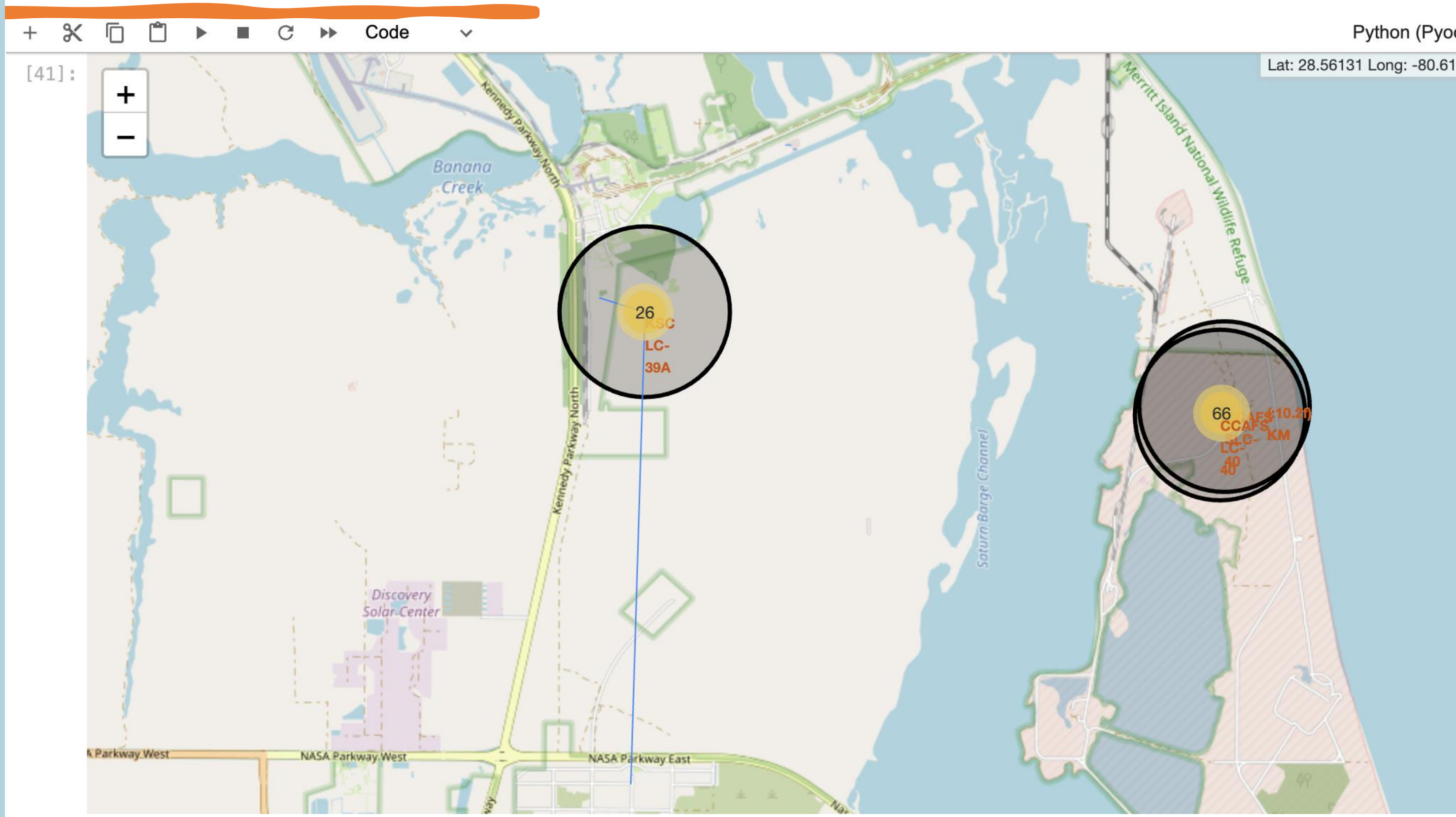


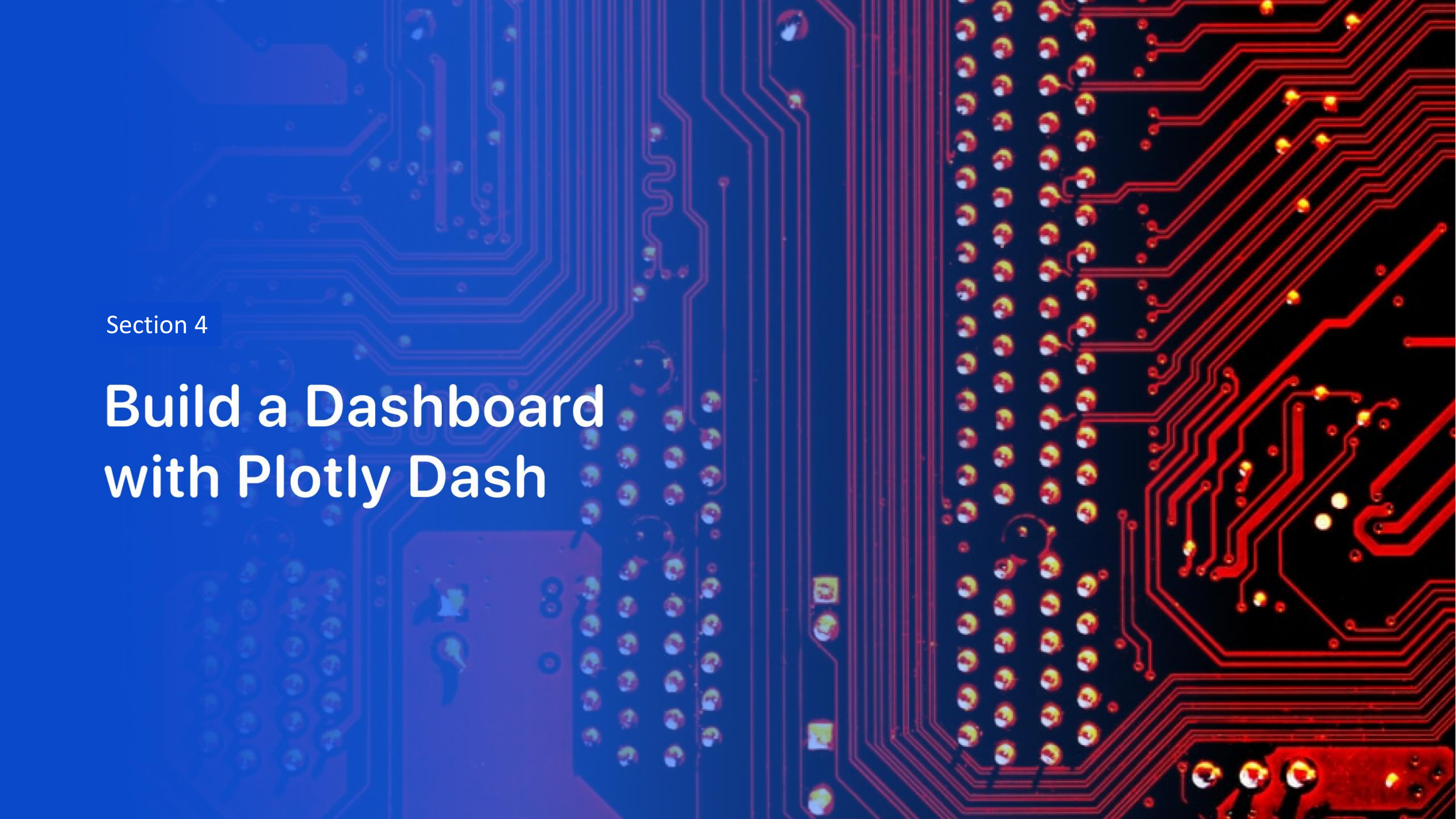




ap Screenshot 3>



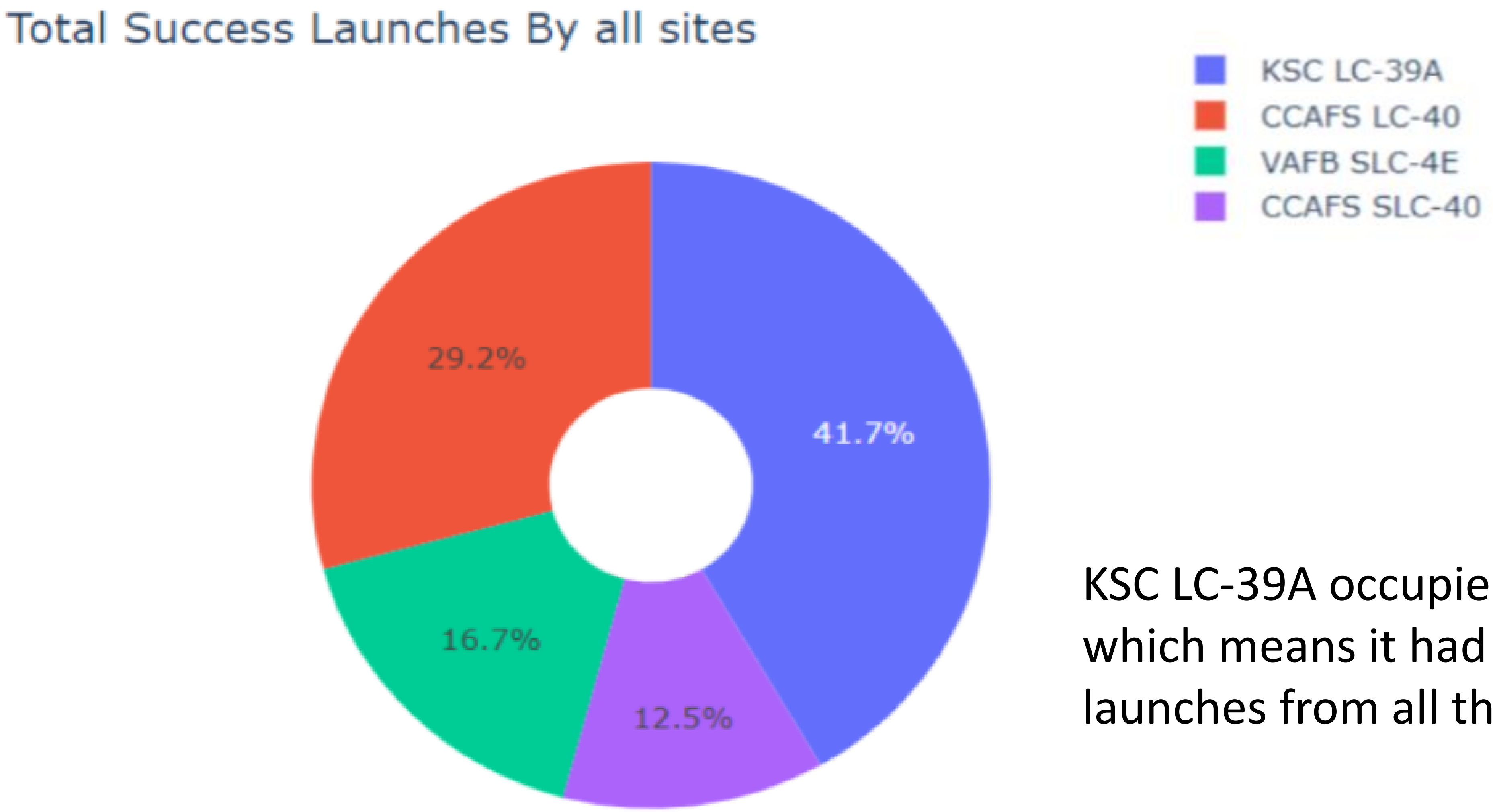




Section 4

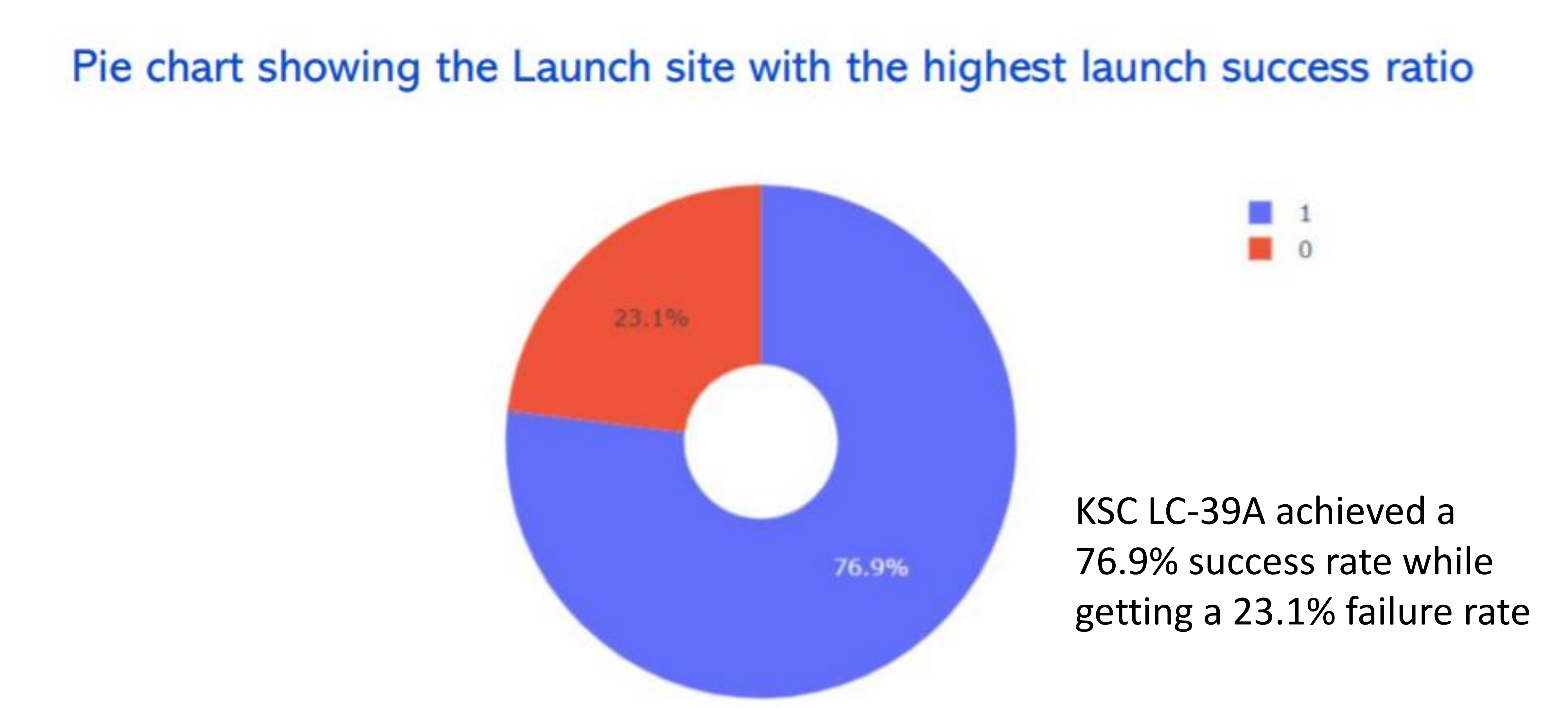
Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>



KSC LC-39A occupied the biggest part, which means it had the most successful launches from all the sites

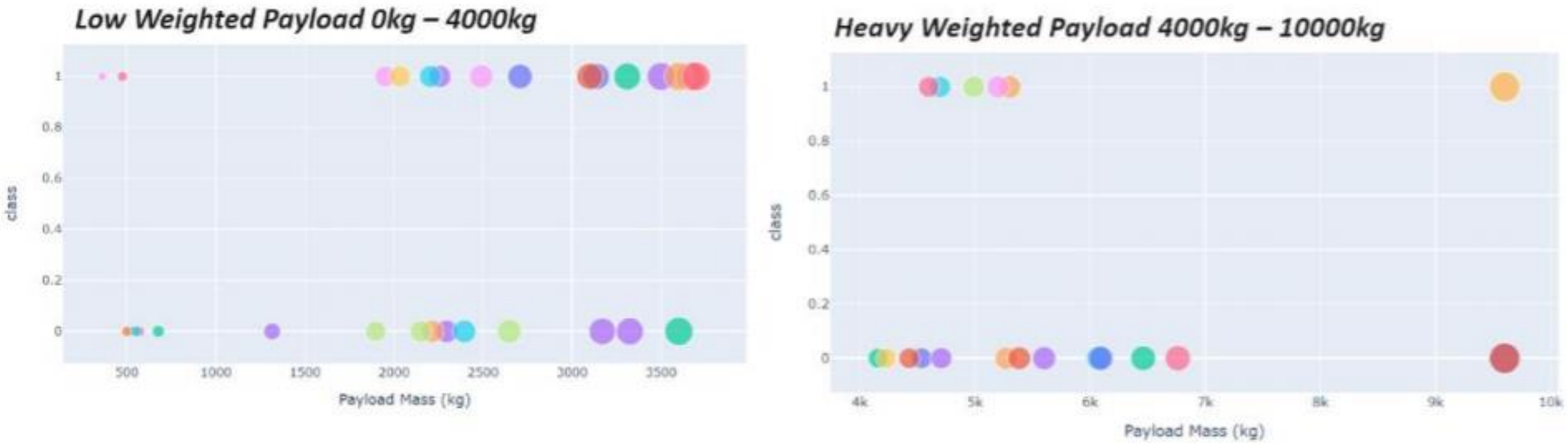
<Dashboard Screenshot 2>



Pie chart showing the Launch site with the highest launch success ratio

KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

<Dashboard Screenshot 3>



Low Weighted Payload 0kg – 4000kg

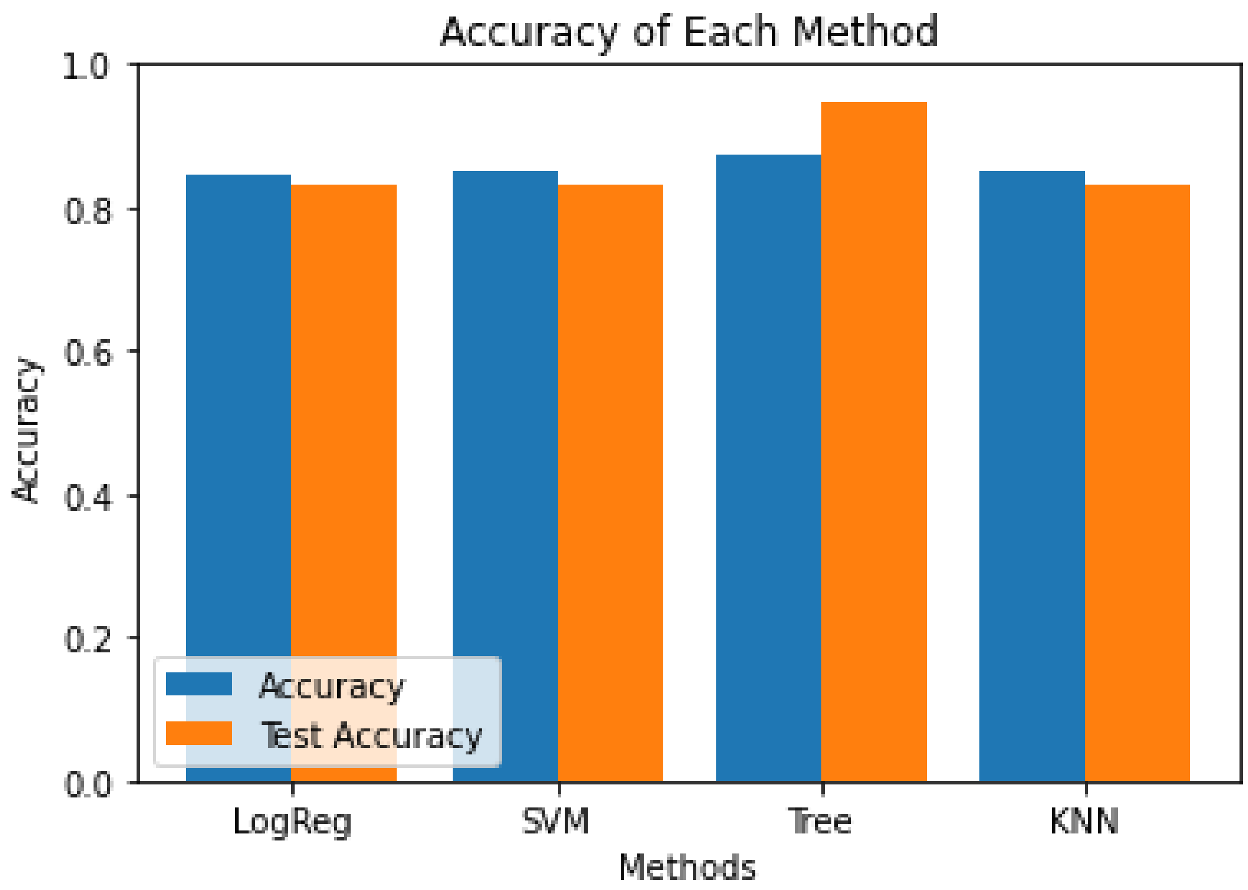
Heavy Weighted Payload 4000kg – 10000kg

The success rate of low weighted payload is higher than heavy weighted payload.

Section 5

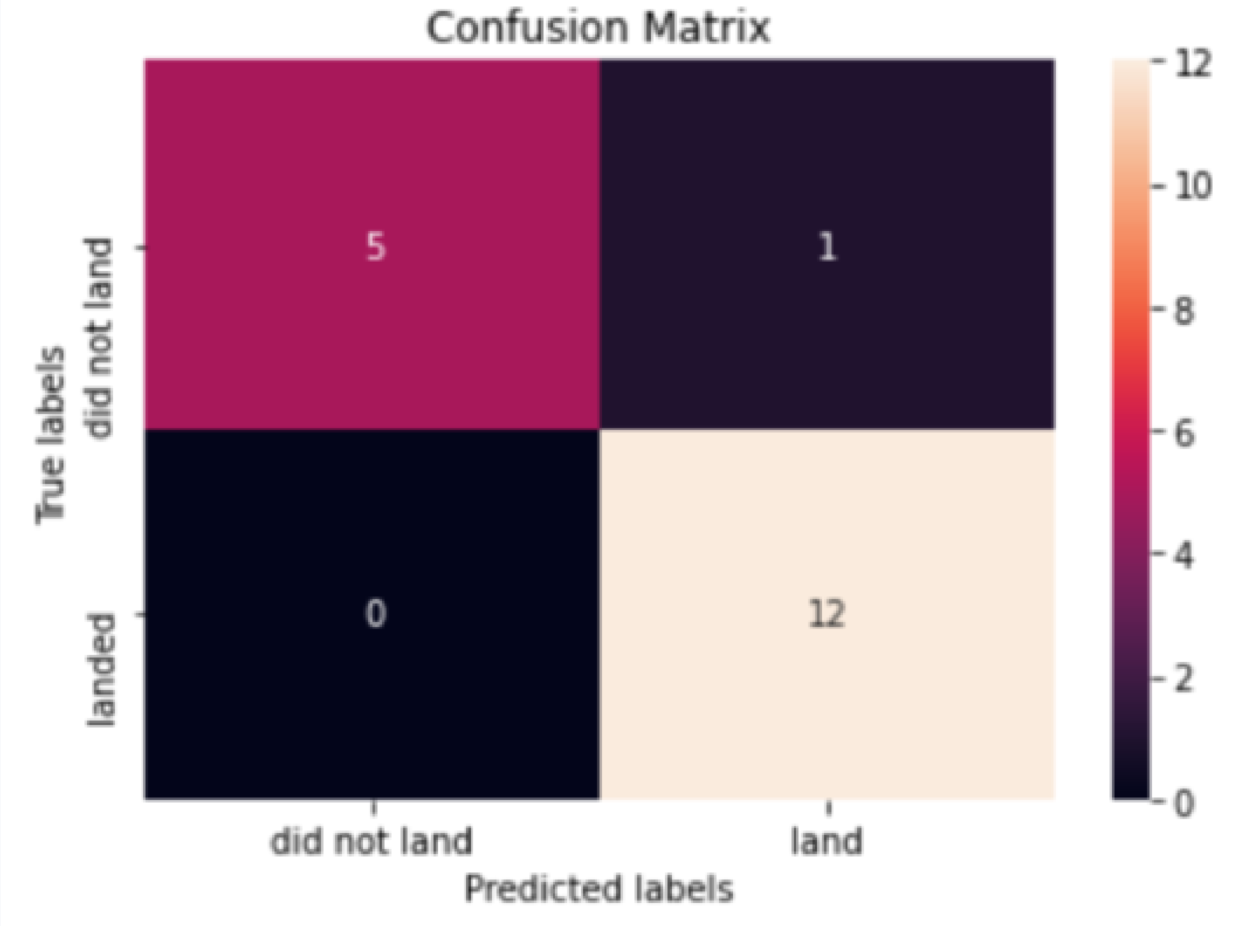
Predictive Analysis (Classification)

Classification Accuracy



Tree method has highest accuracy

Confusion Matrix – Tree classifier



In the confusion matrix for the decision tree classifier, it is evident that the classifier is capable of distinguishing between different classes. However, a notable issue is the occurrence of false positives. These false positives indicate instances where the classifier incorrectly labels a landing as successful when it was, in fact, unsuccessful. This suggests a need for further improvement in correctly identifying unsuccessful landings to reduce such false positives.

Conclusions

Point 1: More flights at a launch site are associated with a higher success rate.

Point 2: The launch success rate has been consistently increasing from 2013 to 2020.

Point 3: Orbits ES-L1, GEO, HEO, SSO, and VLEO have the highest success rates.

Point 4: KSC LC-39A is the launch site with the most successful launches.

Point 5: The Decision Tree classifier is the best machine learning algorithm for this task.

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!