IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Baris Turgay

2nd December 2021

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

  Space X offers Falcon 9 rocket flights on their website for 62 million dollars; other suppliers charge up to 165 million dollars per launch; much of the savings are due to Space X's ability to reuse the first stage. As a result, determining whether the first stage will land allows us to estimate the cost of a launch. This information can be utilized if another firm wishes to compete with Space X for a rocket launch. The project's purpose is to develop a machine learning pipeline for predicting whether the first stage will land successfully.

  Problems you want to find answers

  - What elements affect if the rocket lands successfully?
  - The interplay of several elements determines the likelihood of a successful landing.
  - What operating conditions are required for a successful landing program?

Section 1

Methodology

Methodology

Executive Summary

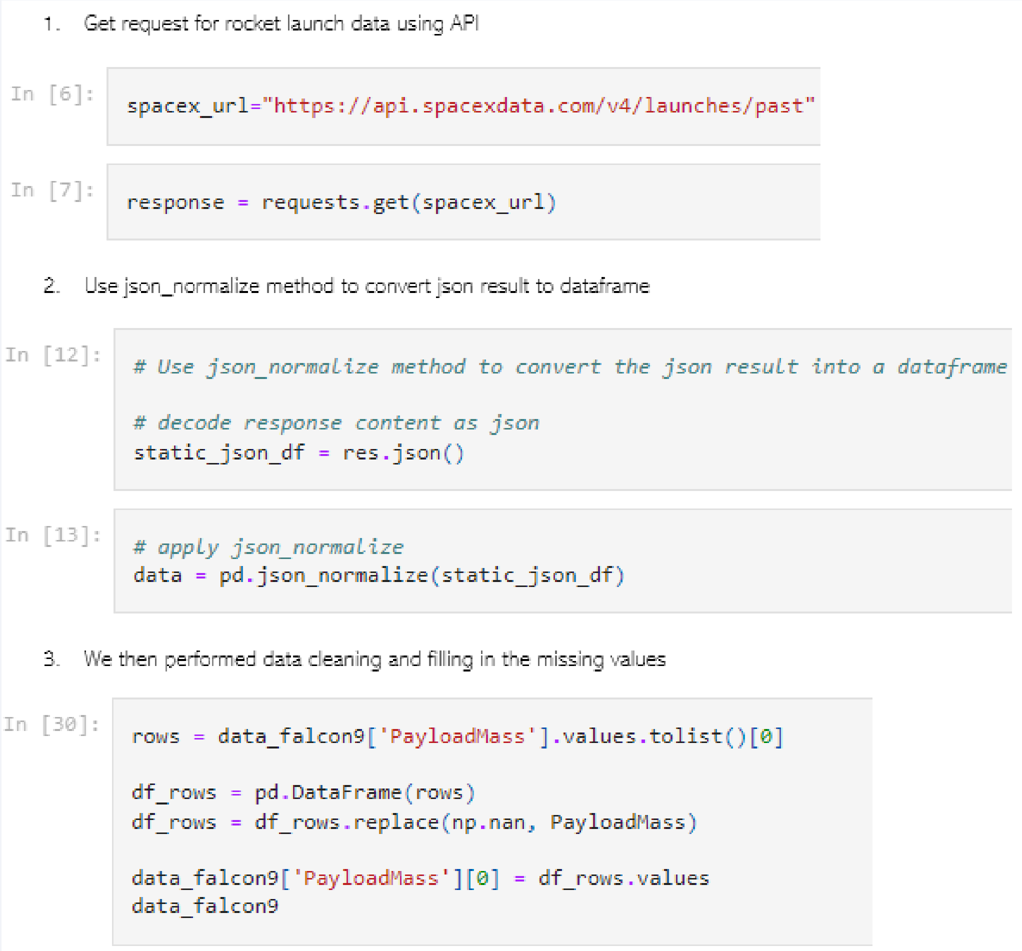
- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
  - One-hot encoding was applied to categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- The data was collected using various methods
  - Data collection was done using get request to the SpaceX API.
  - Next, we decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
  - We then cleaned the data, checked for missing values and fill in missing values where necessary.
  - In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
  - The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

- We utilized the get request to the SpaceX API to collect data, sanitize it, and do some basic data wrangling and formatting.
- The link to the notebook is https://github.com/MrBoris1/SpaceX/blob/main/jupyter-labs-spacex-data-collection-api.ipynb



1. Get request for rocket launch data using API

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

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

2. Use json_normalize method to convert json result to dataframe

```
In [12]: # Use json_normalize method to convert the json result into a dataframe

         # decode response content as json
         static_json_df = res.json()
```

```
In [13]: # apply json_normalize
         data = pd.json_normalize(static_json_df)
```

3. We then performed data cleaning and filling in the missing values

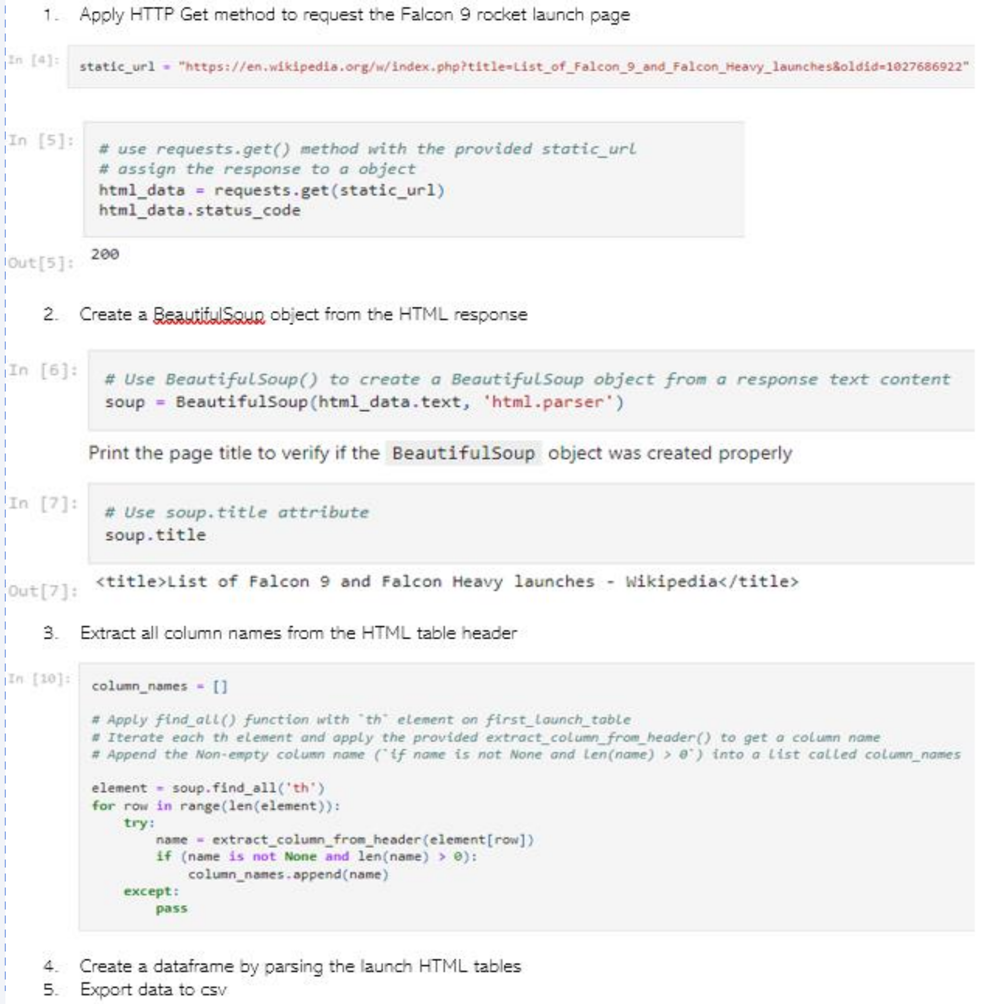
```
In [30]: rows = data_falcon9['PayloadMass'].values.tolist()[0]

         df_rows = pd.DataFrame(rows)
         df_rows = df_rows.replace(np.nan, PayloadMass)

         data_falcon9['PayloadMass'][0] = df_rows.values
         data_falcon9
```

Data Collection - Scraping

- We used BeautifulSoup to web scrape Falcon 9 launch records.
- We processed the table and transformed it to a Pandas dataframe.
- The link to the notebook is https://github.com/MrBoris1/SpaceX/blob/main/jupyter-labs-webscraping.ipynb



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

```
In [4]:  static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

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

Out[5]:  200
```

2. Create a BeautifulSoup object from the HTML response

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

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

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

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

3. Extract all column names from the HTML table header

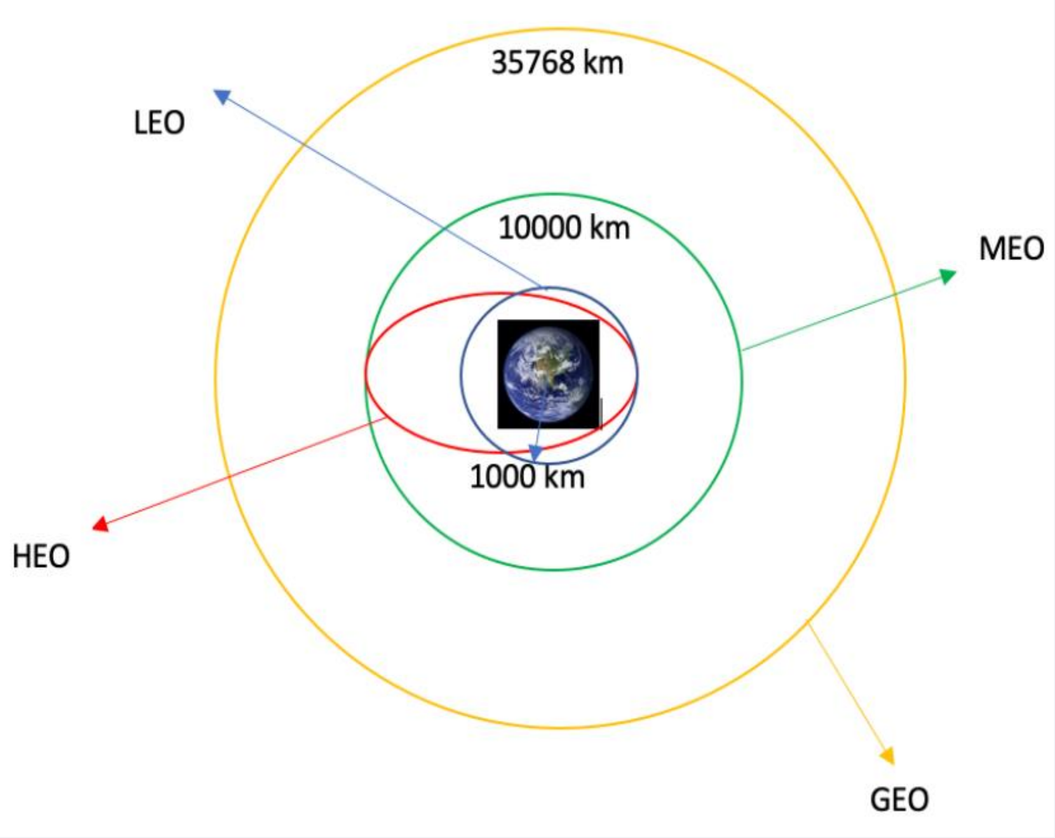
```
In [10]: column_names = []

         # 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

         element = soup.find_all('th')
         for row in range(len(element)):
             try:
                 name = extract_column_from_header(element[row])
                 if (name is not None and len(name) > 0):
                     column_names.append(name)
             except:
                 pass
```

4. Create a dataframe by parsing the launch HTML tables
5. Export data to csv

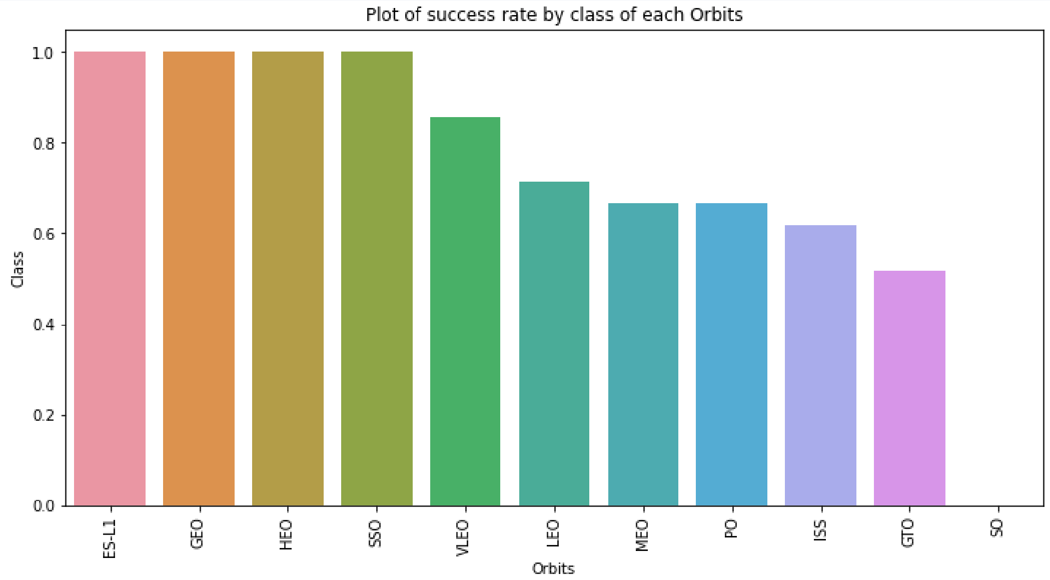
Data Wrangling

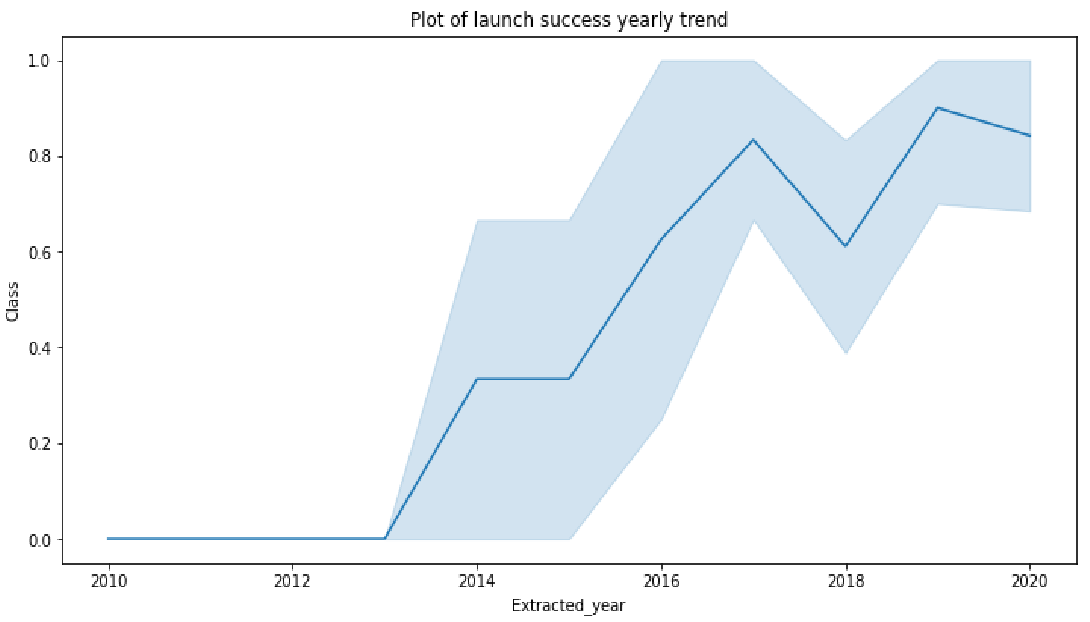


- We did exploratory data analysis to find the training labels.
- We computed the number of launches at each location, as well as the number and frequency of each orbit.
- We generated a landing outcome label from the outcome column and exported the results to CSV.
- The link to the notebook is https://github.com/MrBoris1/SpaceX/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- We analyzed the data by displaying the association between flight number and launch site, payload and launch site, success rate of each orbit type, flight number and orbit type, and yearly launch success trends.





- The link to the notebook is https://github.com/MrBoris1/SpaceX/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

- We loaded the SpaceX dataset into a PostgreSQL database without exiting the Jupyter notebook.
- We used EDA and SQL to gain insight from the data. We created queries to find out, for example:
  - The names of distinct launch locations of the space project.
  - The total payload mass of boosters launched by NASA (CRS).
  - The average payload mass of rocket type F9 v1.1.The overall number of successful and unsuccessful mission results.
  - The failure landing results in drone ships, their booster versions, and launch location names.
- The link to the notebook is https://github.com/MrBoris1/SpaceX/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- We highlighted all launch locations and added map items such as markers, circles, and lines to indicate the success or failure of launches at each site on the folium map.
- We classified the feature launch results (failure or success) as class 0 and 1.i.e., zero for failure and one for achievement.
- Using color-labeled marker clusters, we determined which launch sites have a reasonably high success rate.
- We estimated the distances between a launch location and its surroundings. We answered some questions, for example:
  - Are launch locations near trains, highways, and coastlines?
  - Do launch sites preserve a particular distance from cities?

Build a Dashboard with Plotly Dash

- We created an interactive dashboard using Plotly Dash.
- We produced pie charts displaying the total launches by a particular location.
- We created a scatter graph demonstrating the association between Outcome and Payload Mass (Kg) for each booster version.
- The link to the notebook is https://github.com/MrBoris1/SpaceX/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- We imported the data with numpy and pandas, converted it, then partitioned it into training and testing.
- We used GridSearchCV to create several machine learning models and modify hyperparameters.
- We utilized accuracy as the measure for our model and enhanced it through feature engineering and algorithm tweaking.
- We discovered the best-performing categorization model.
- The link to the notebook is https://github.com/MrBoris1/SpaceX/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

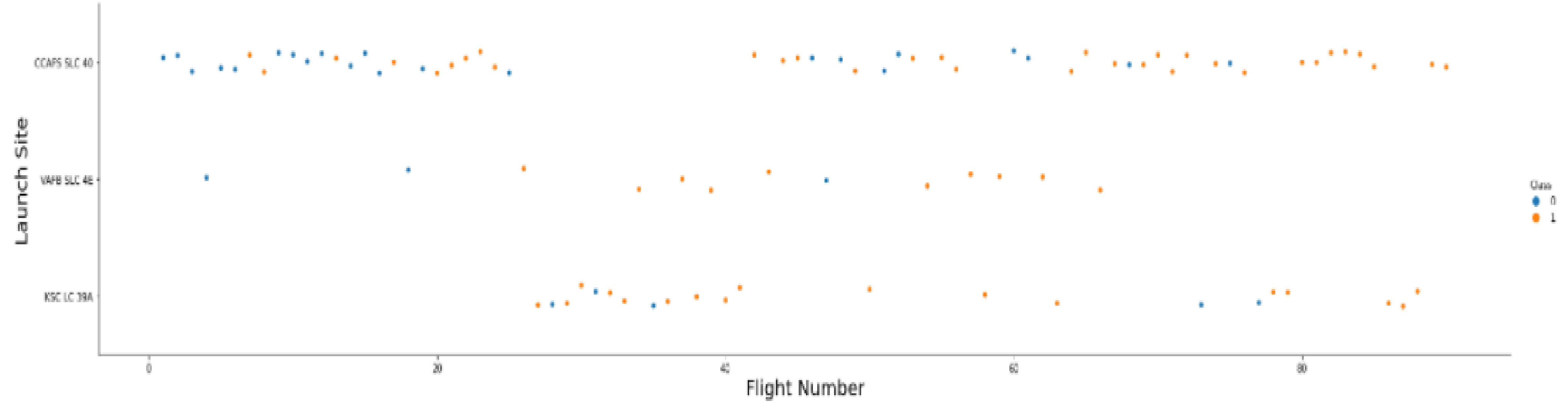
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

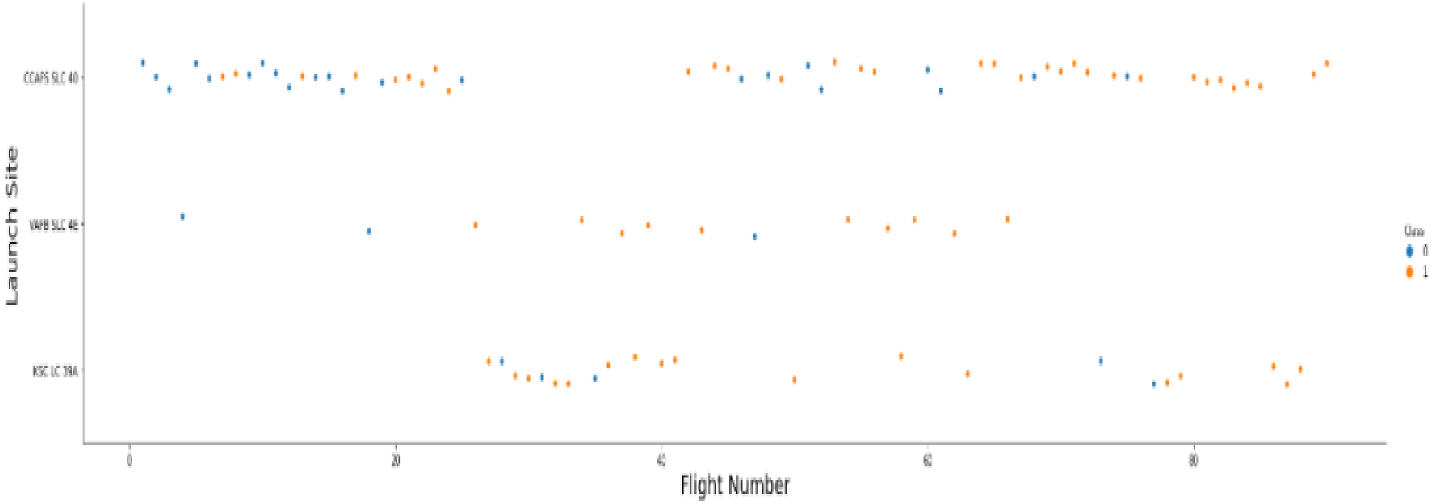
Flight Number vs. Launch Site

- According to the plot, the more the number of flights at a launch point, the higher the success rate.



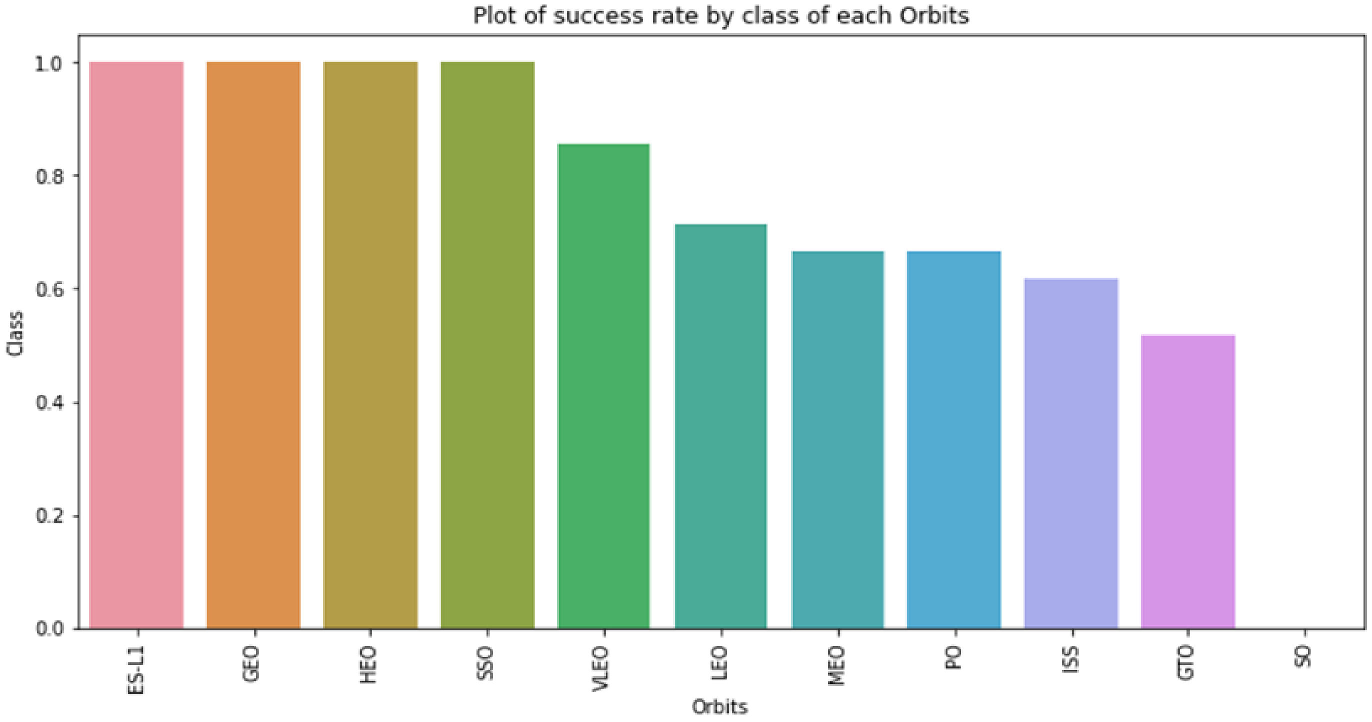
Payload vs. Launch Site

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



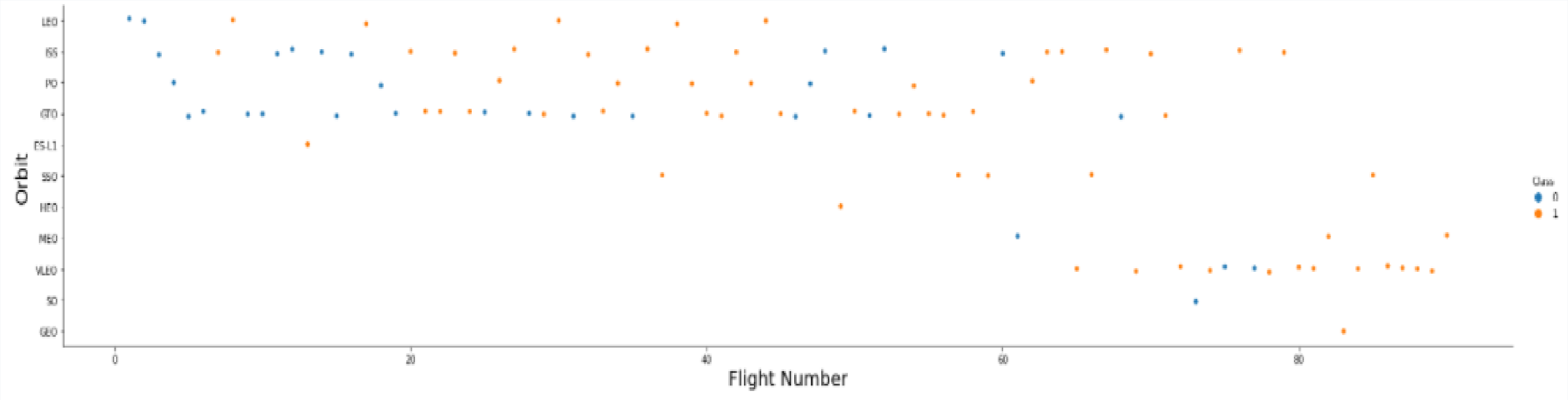
Success Rate vs. Orbit Type

- The figure shows that ES-L1, GEO, HEO, SSO, and VLEO had the highest success rate.



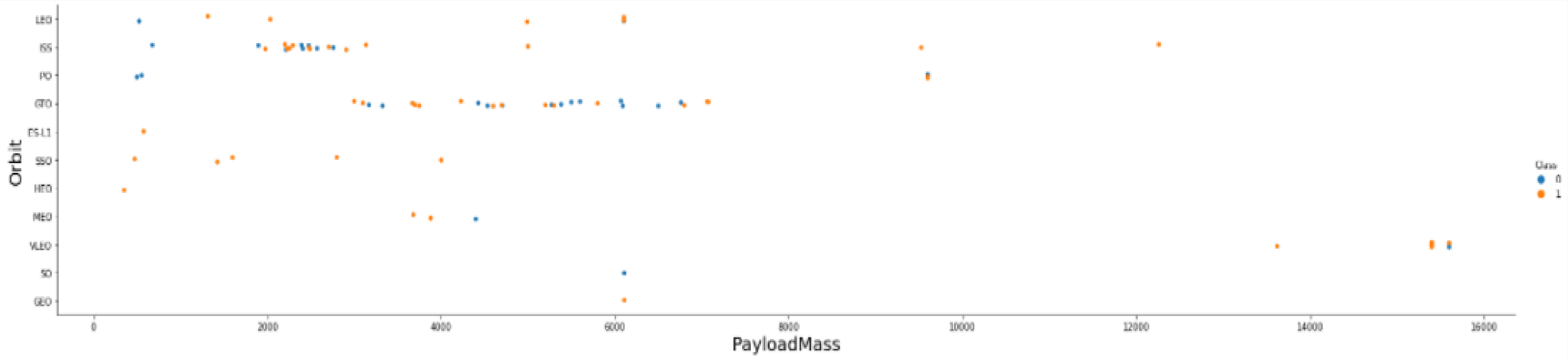
Flight Number vs. Orbit Type

- The graphic below depicts the Flight Number vs. Orbit Type. We see that success in the LEO orbit is proportional to the number of flights, however in the GTO orbit, there is no link between flight number and orbit.



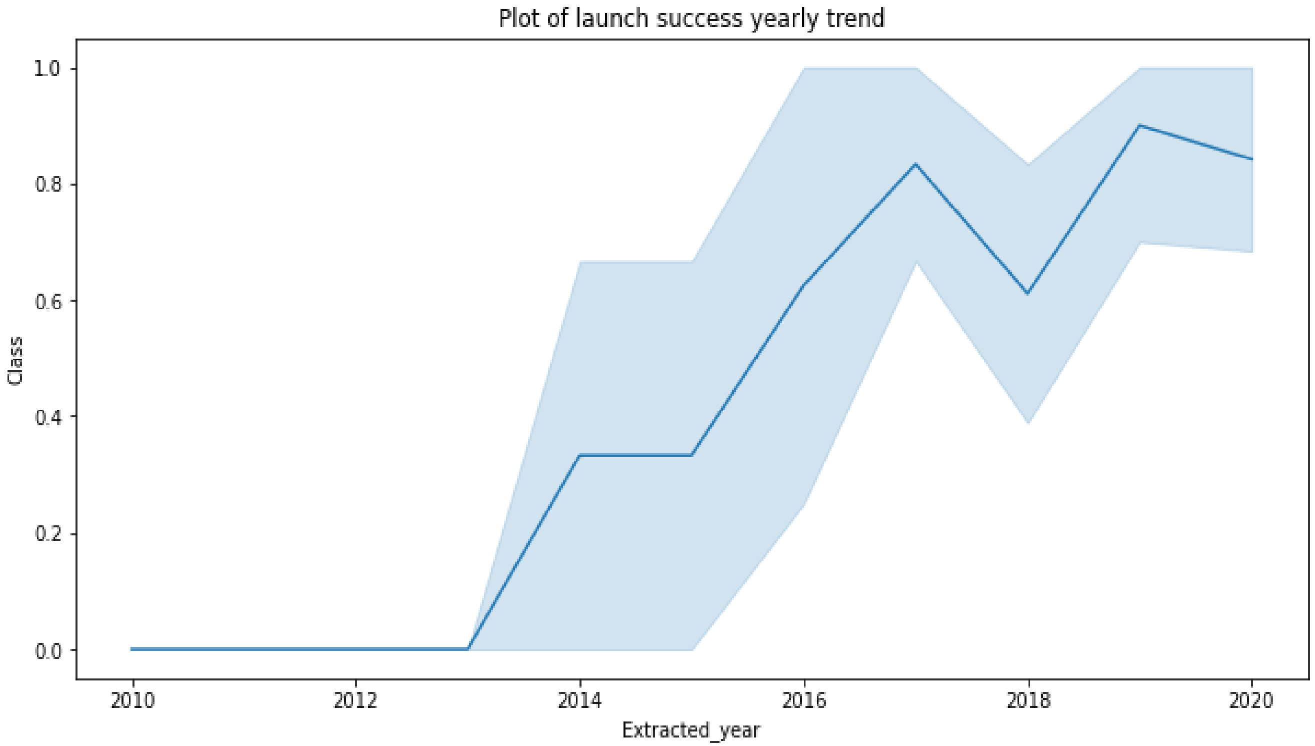
Payload vs. Orbit Type

- We can see that large payloads result in more successful landings in PO, LEO, and ISS orbits.



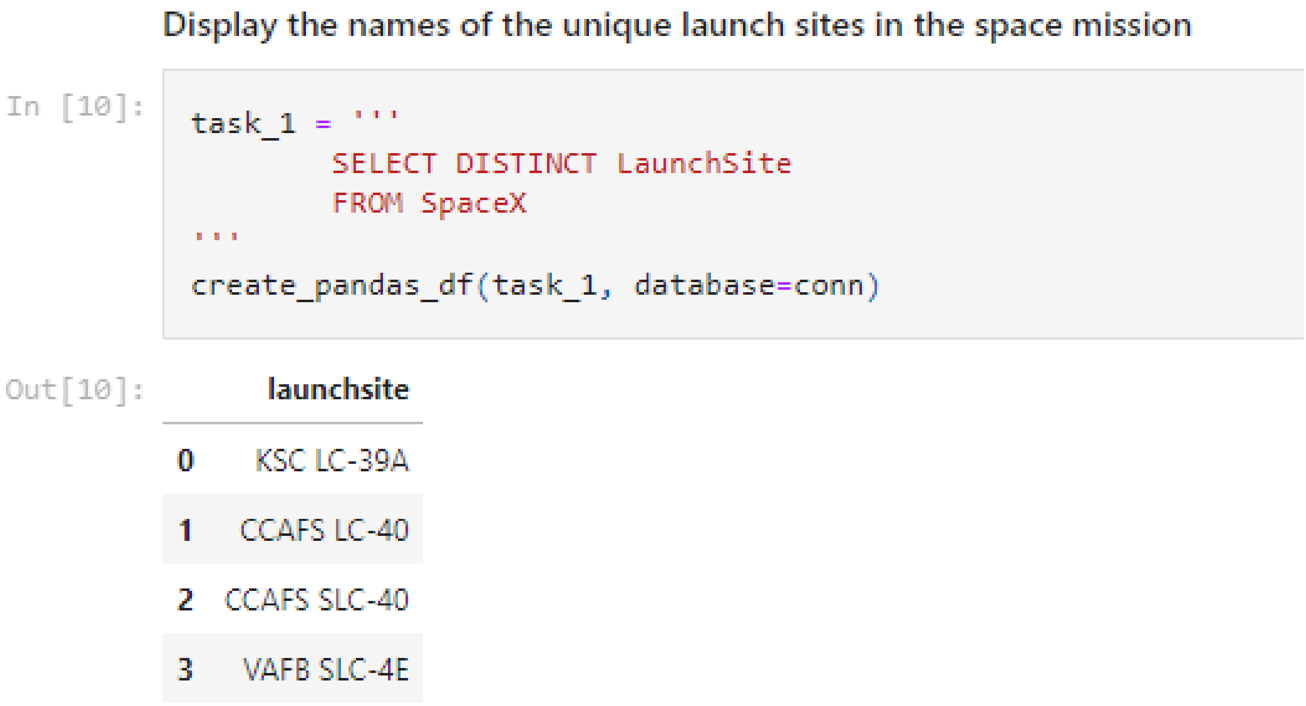
Launch Success Yearly Trend

- The figure shows that the success rate has increased steadily since 2013 and will continue to do so until 2020.



All Launch Site Names

- We utilized the keyword DISTINCT to display only unique launch locations from SpaceX data.

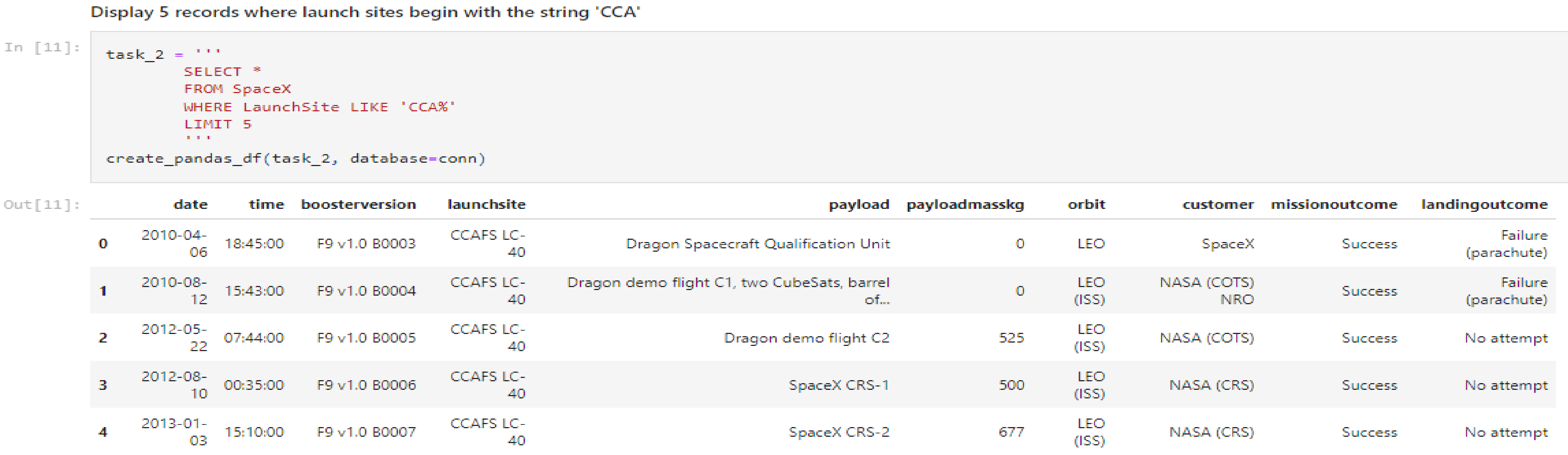


Display the names of the unique launch sites in the space mission

```
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'



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)
```

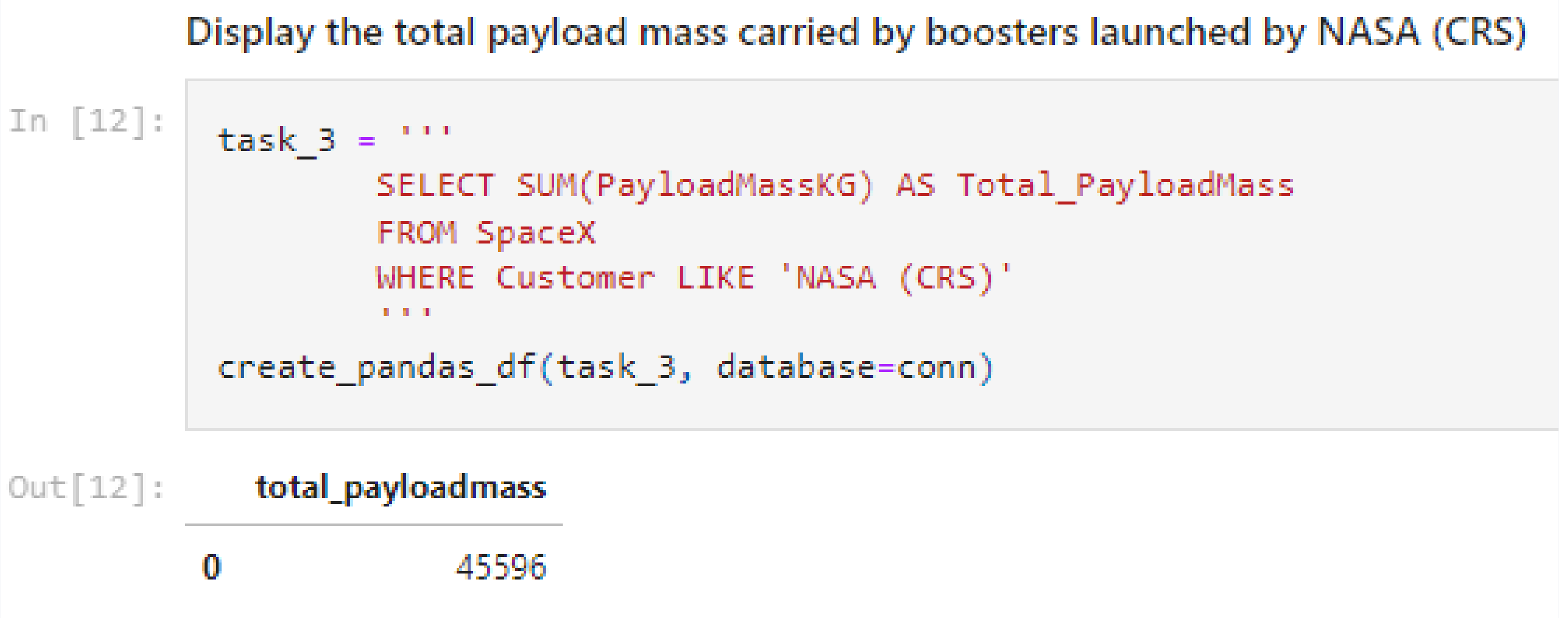
Out[11]:

	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

- We performed the above query to display five results whose launch sites begin with 'CCA'.

Total Payload Mass

- We computed the total payload carried by NASA boosters as 45596 using the query below.



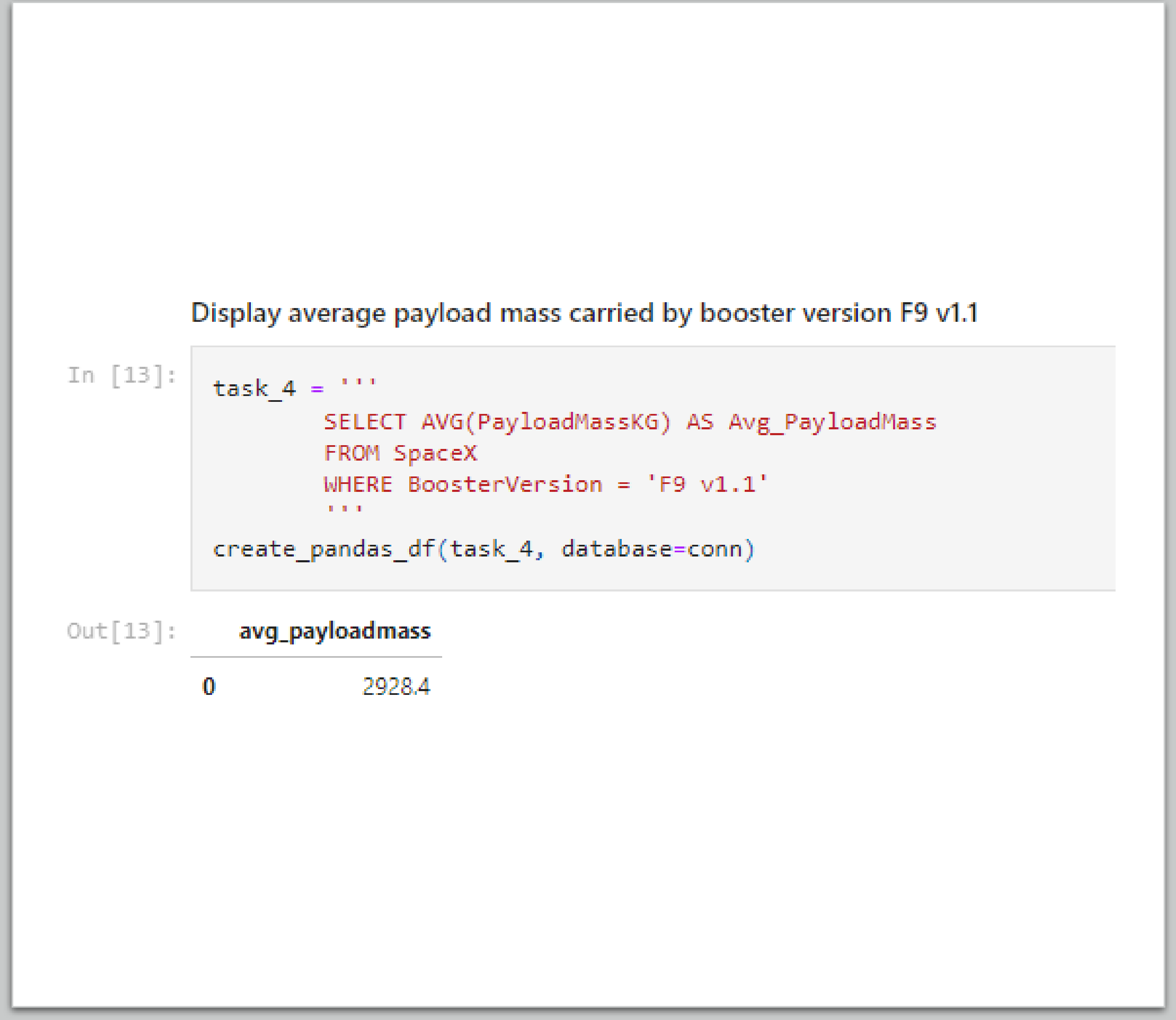
Display the total payload mass carried by boosters launched by NASA (CRS)

```
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
```

Average Payload Mass by F9 v1.1

- We determined that the average payload mass carried by booster version F9 v1.1 is 2928.4.



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

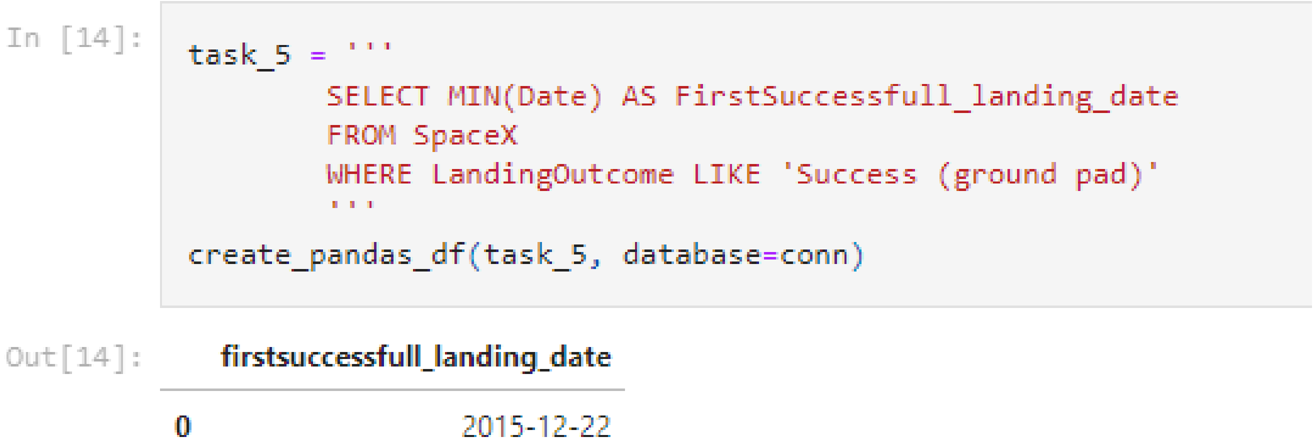
```
In [13]:  task_4 = '''
                  SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
                  FROM SpaceX
                  WHERE BoosterVersion = 'F9 v1.1'
                  '''
          create_pandas_df(task_4, database=conn)
```

Out[13]:

	avg_payloadmass
0	2928.4

First Successful Ground Landing Date

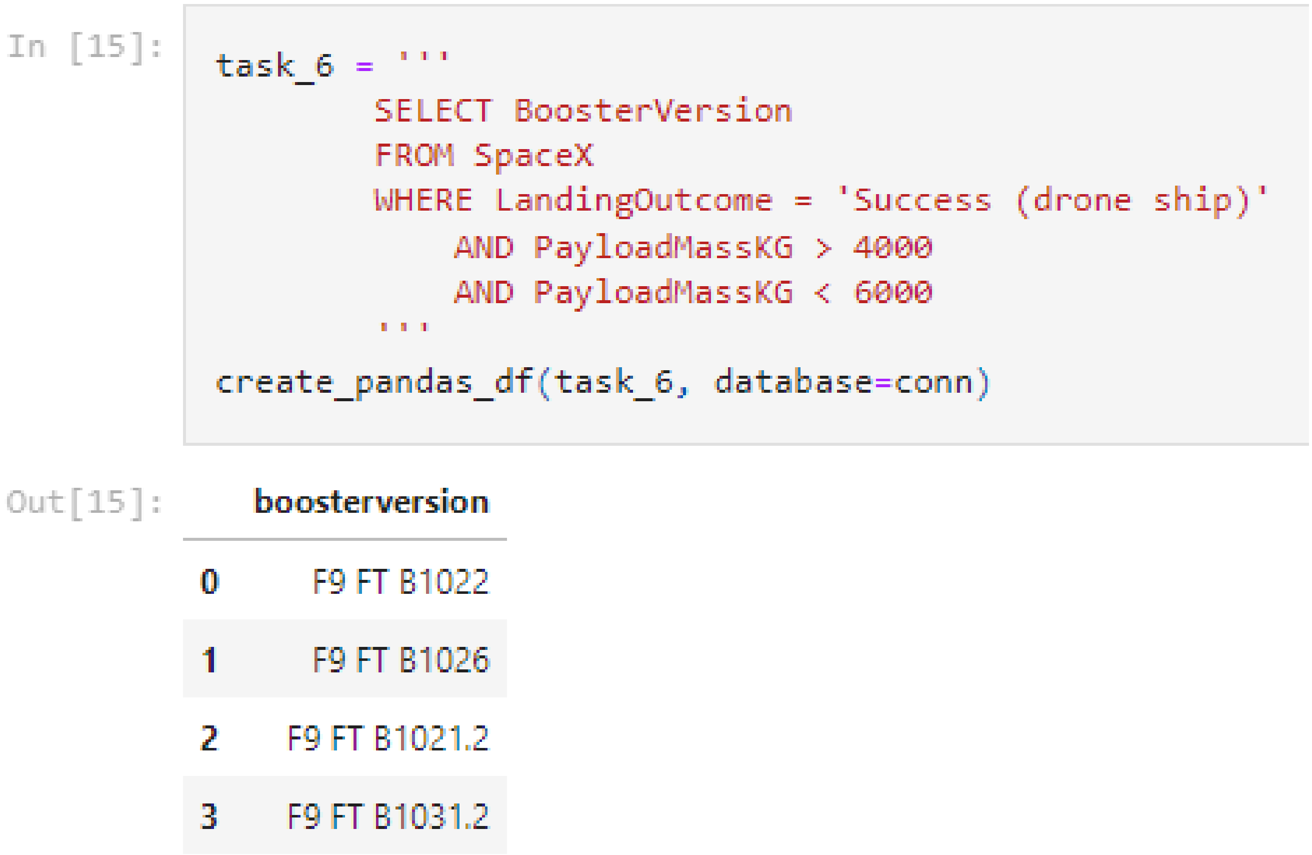
- We noted that the dates of the first successful landing outcome on the ground pad were December 22, 2015.



```
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
```

Successful Drone Ship Landing with Payload between 4000 and 6000



In [15]:

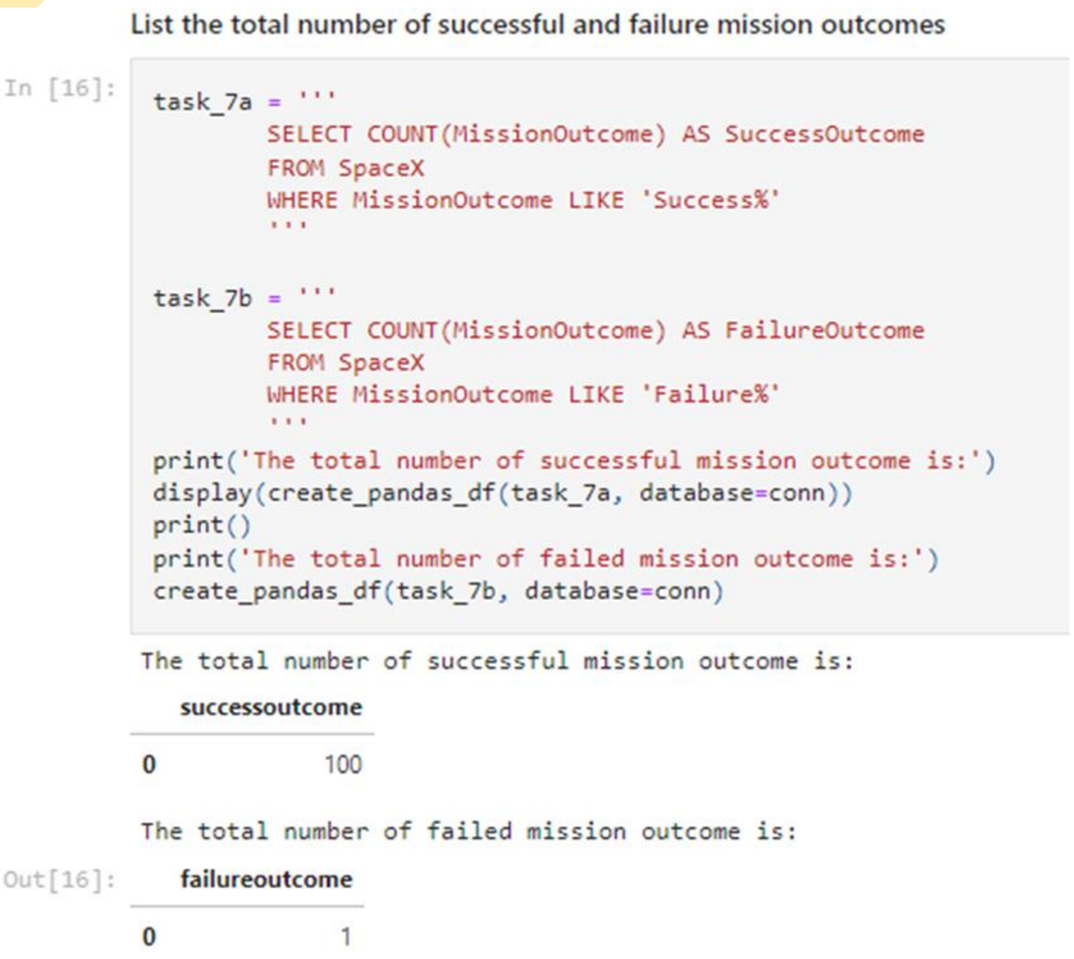
```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

Out[15]:

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

- We 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

Total Number of Successful and Failure Mission Outcomes



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:
```

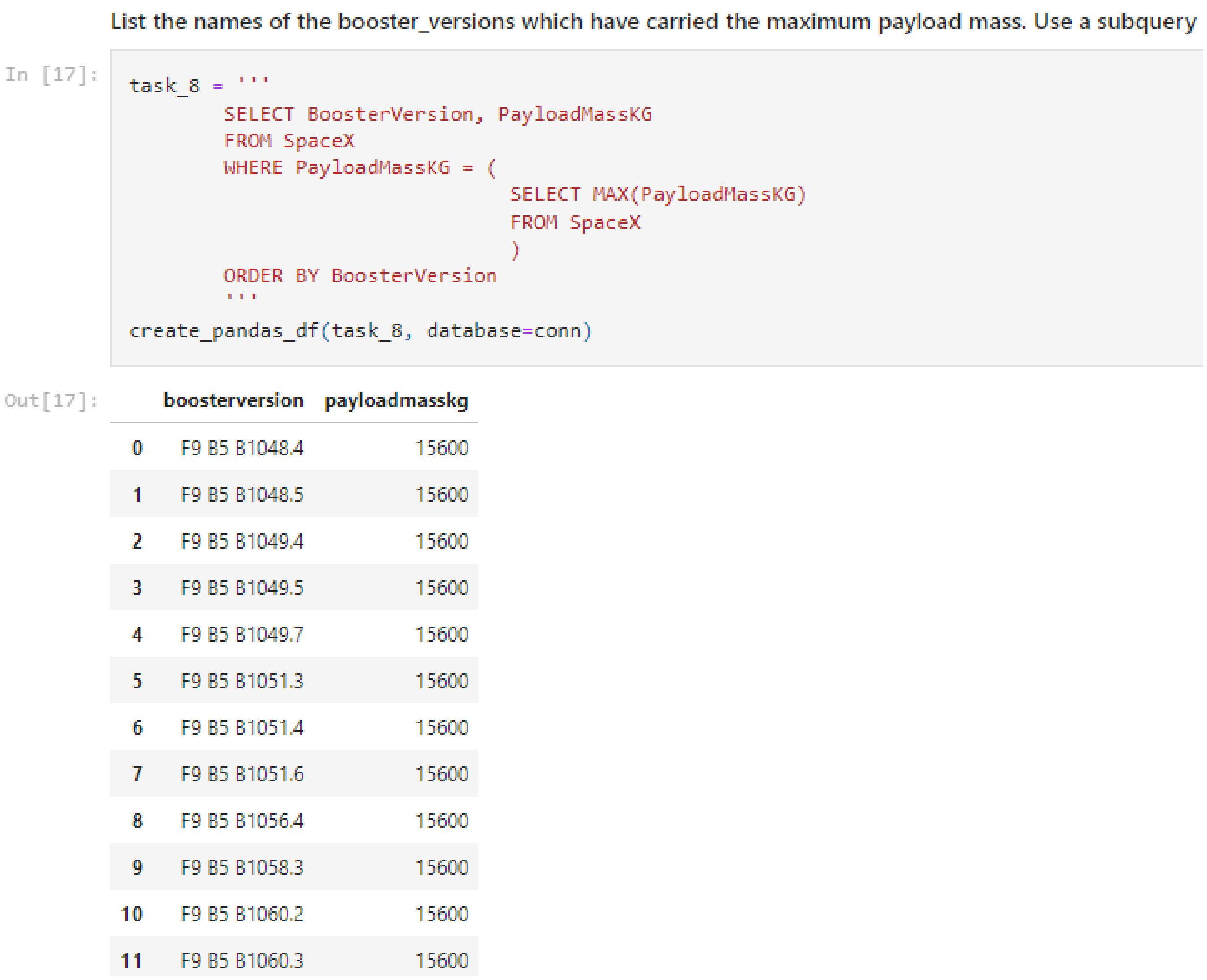
Out[16]:

	failureoutcome
0	1

- We utilized a wildcard, such as '%', to filter for WHERE MissionOutcome was successful or unsuccessful.

Boosters Carried Maximum Payload

- We used a subquery in the WHERE clause and the MAX() method to discover which booster had the most payload.



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

In [17]:

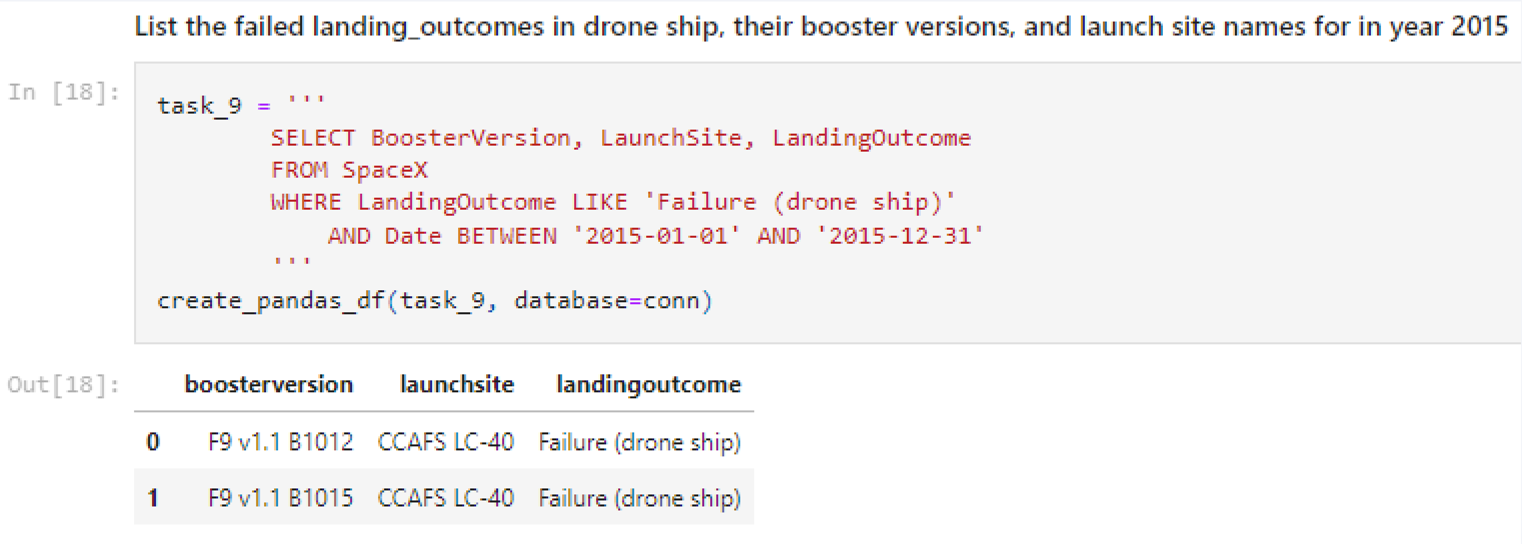
```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, 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

2015 Launch Records

- We utilized the WHERE clause, LIKE, AND, and BETWEEN conditions to filter for unsuccessful landings in drone ships, booster versions, and launch location names in 2015.



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

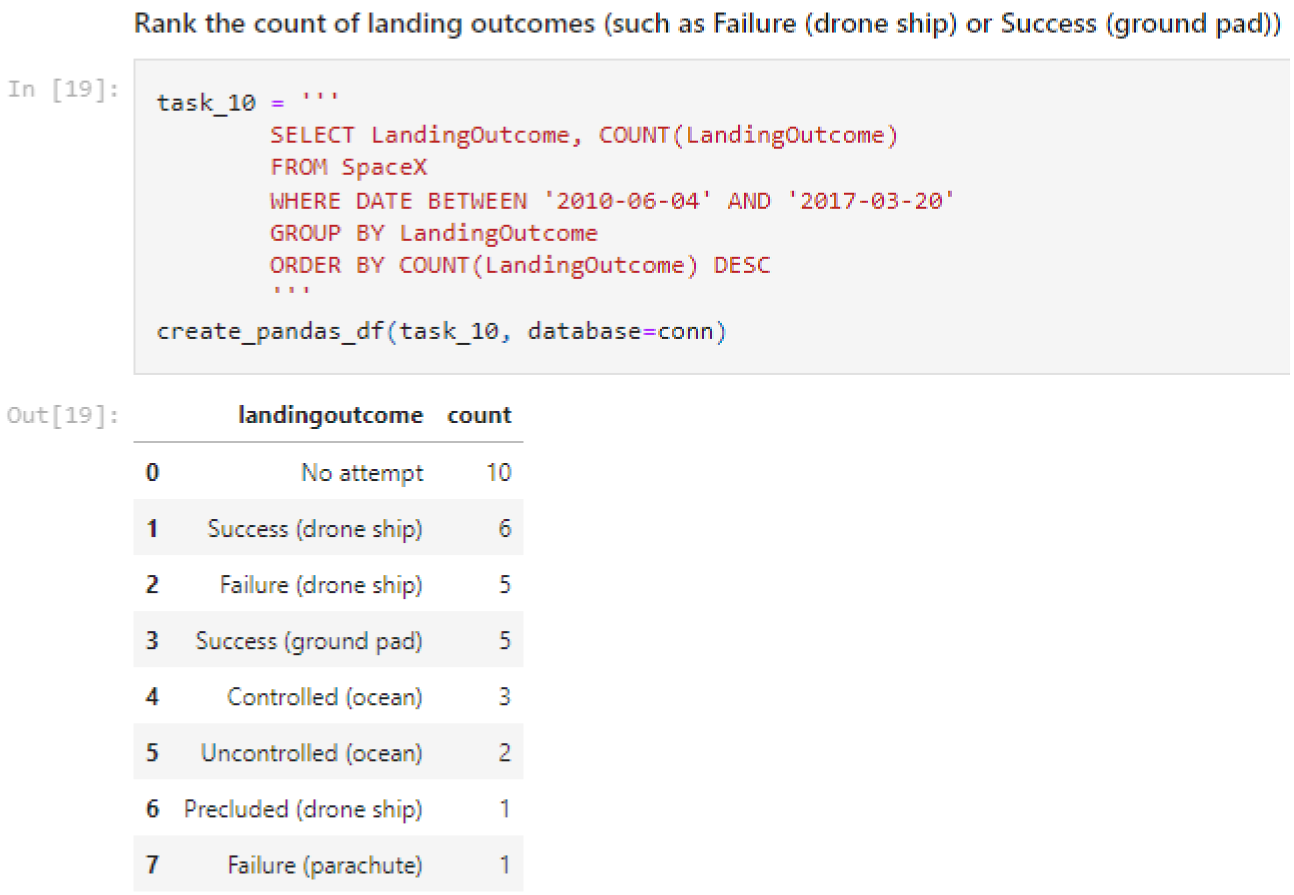
In [18]:

```
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)

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



Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad))

```
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]:

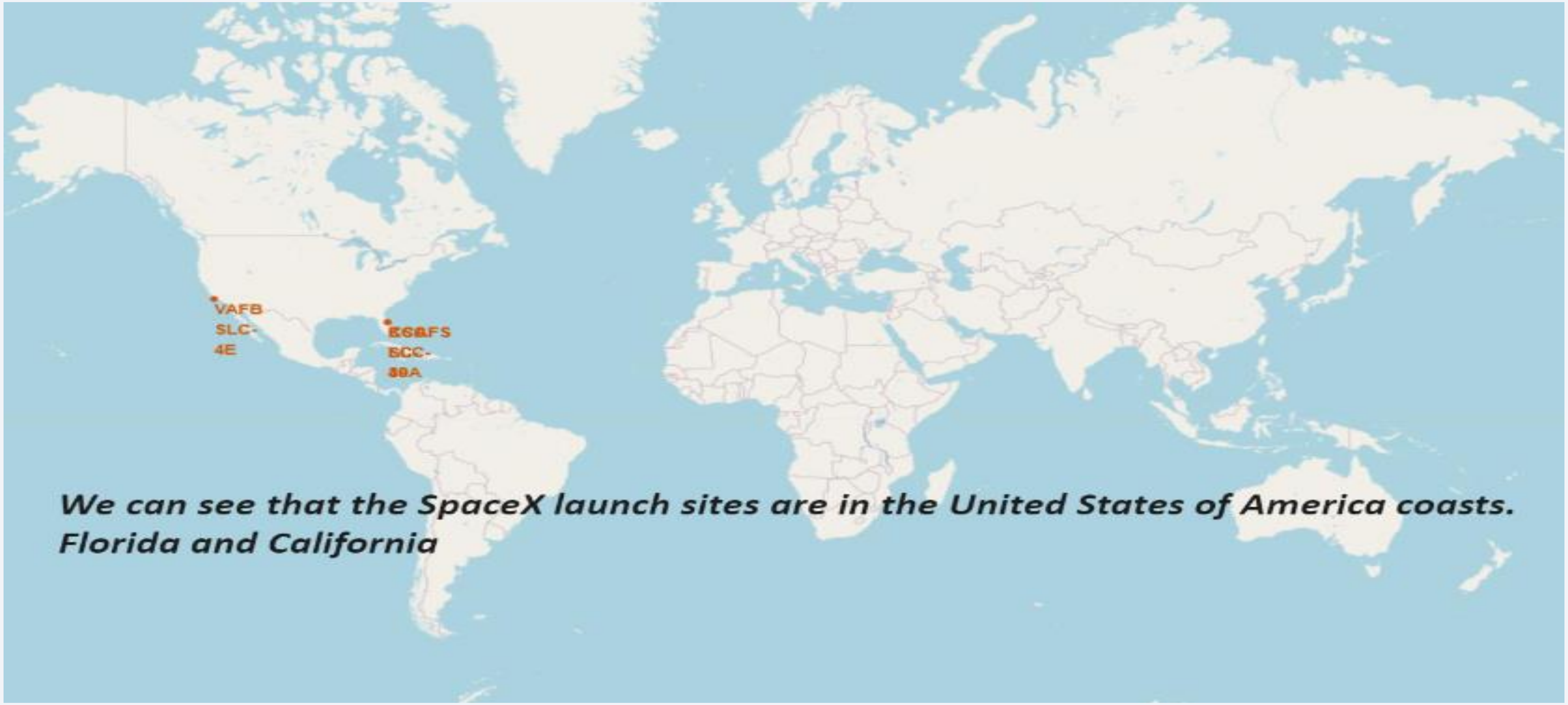
	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

- We chose Landing outcomes and the COUNT of landing outcomes from the data, and then used the WHERE clause to filter for landing outcomes between 2010-06-04 and 2010-03-20.
- We used the categorize BY clause to categorize the landing results, then the ORDER BY clause to sort them in decreasing order.

Section 4

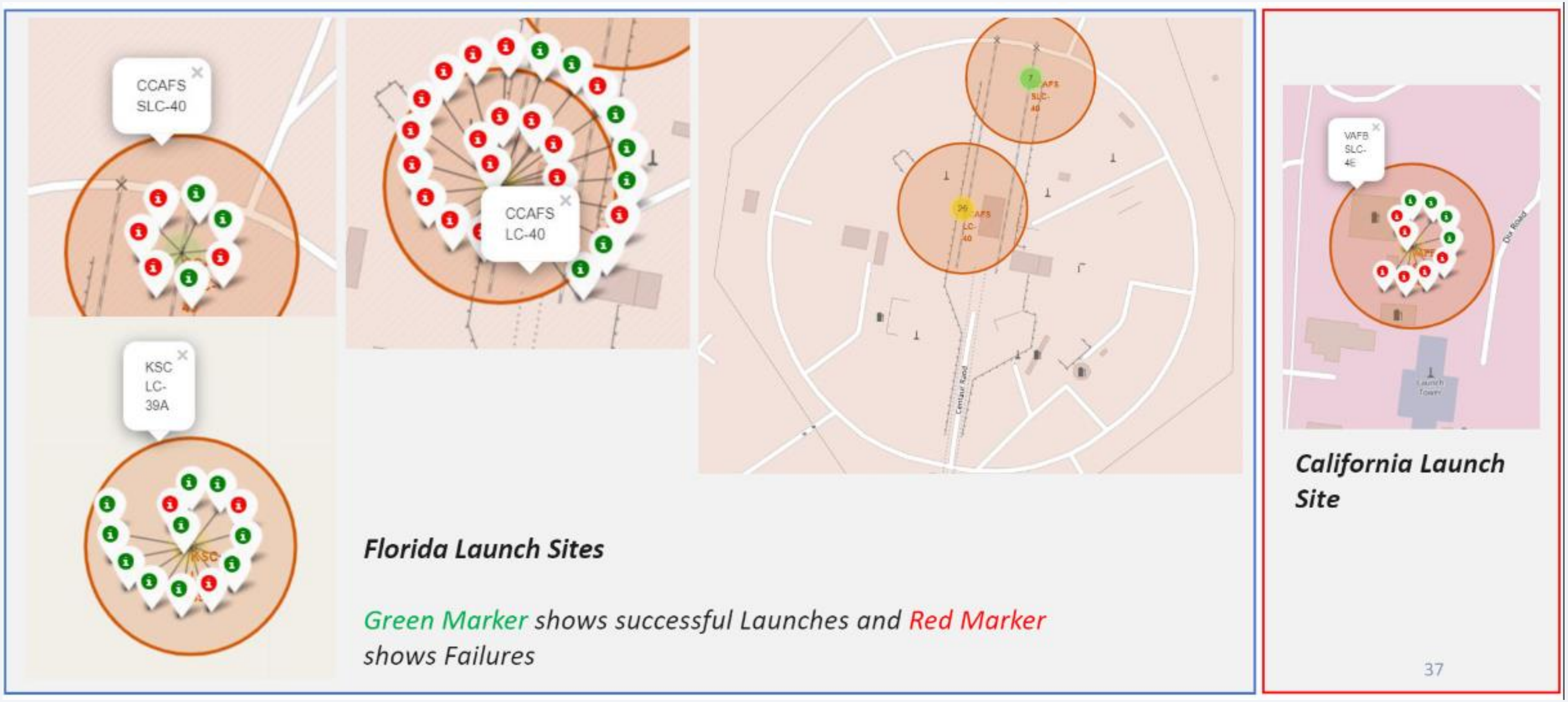
Launch Sites Proximities Analysis

All launch sites global map markers



We can see that the SpaceX launch sites are in the United States of America coasts. Florida and California

Markers showing launch sites with color labels

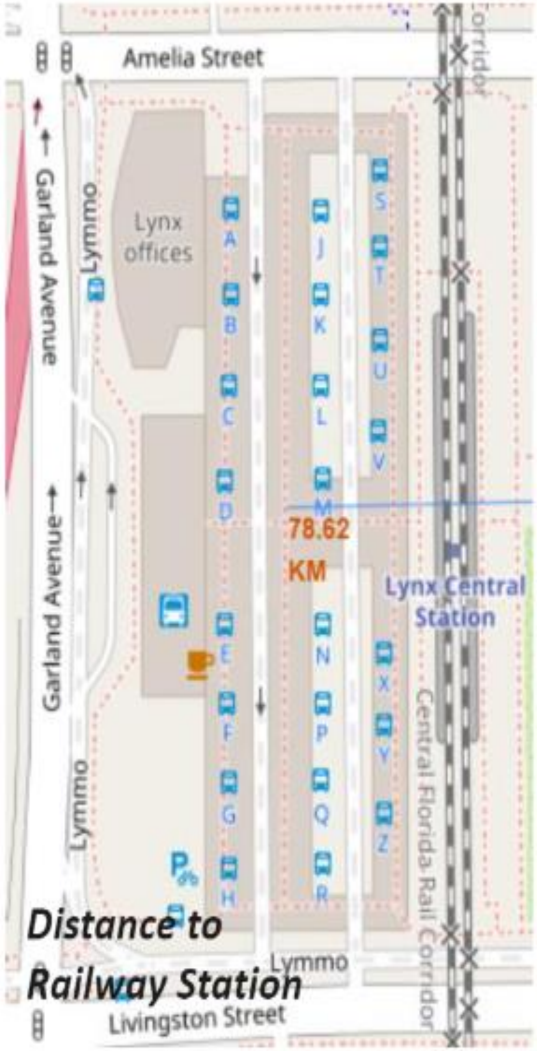


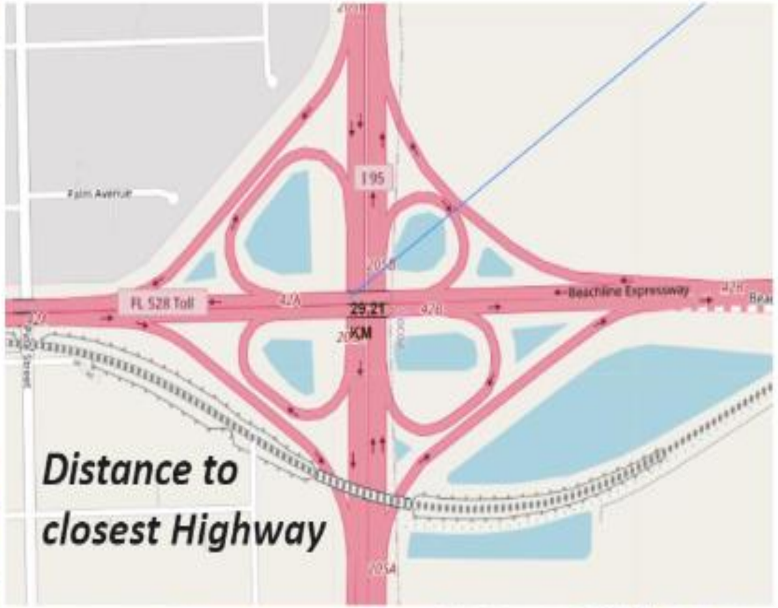
Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows Failures

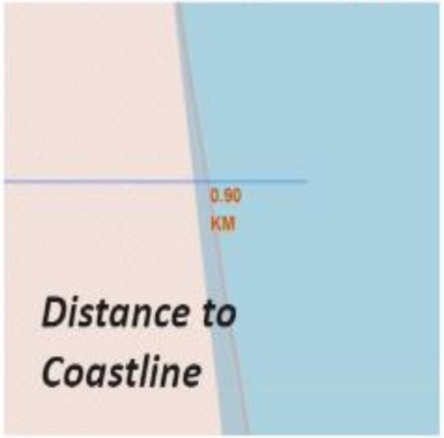
California Launch Site

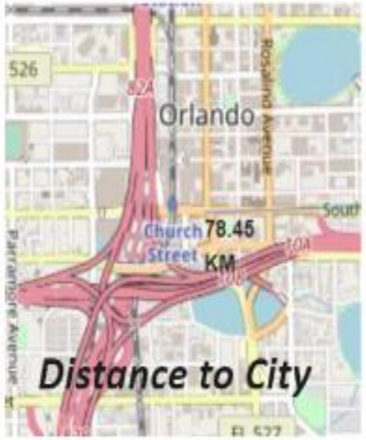
Launch Site distance to landmarks



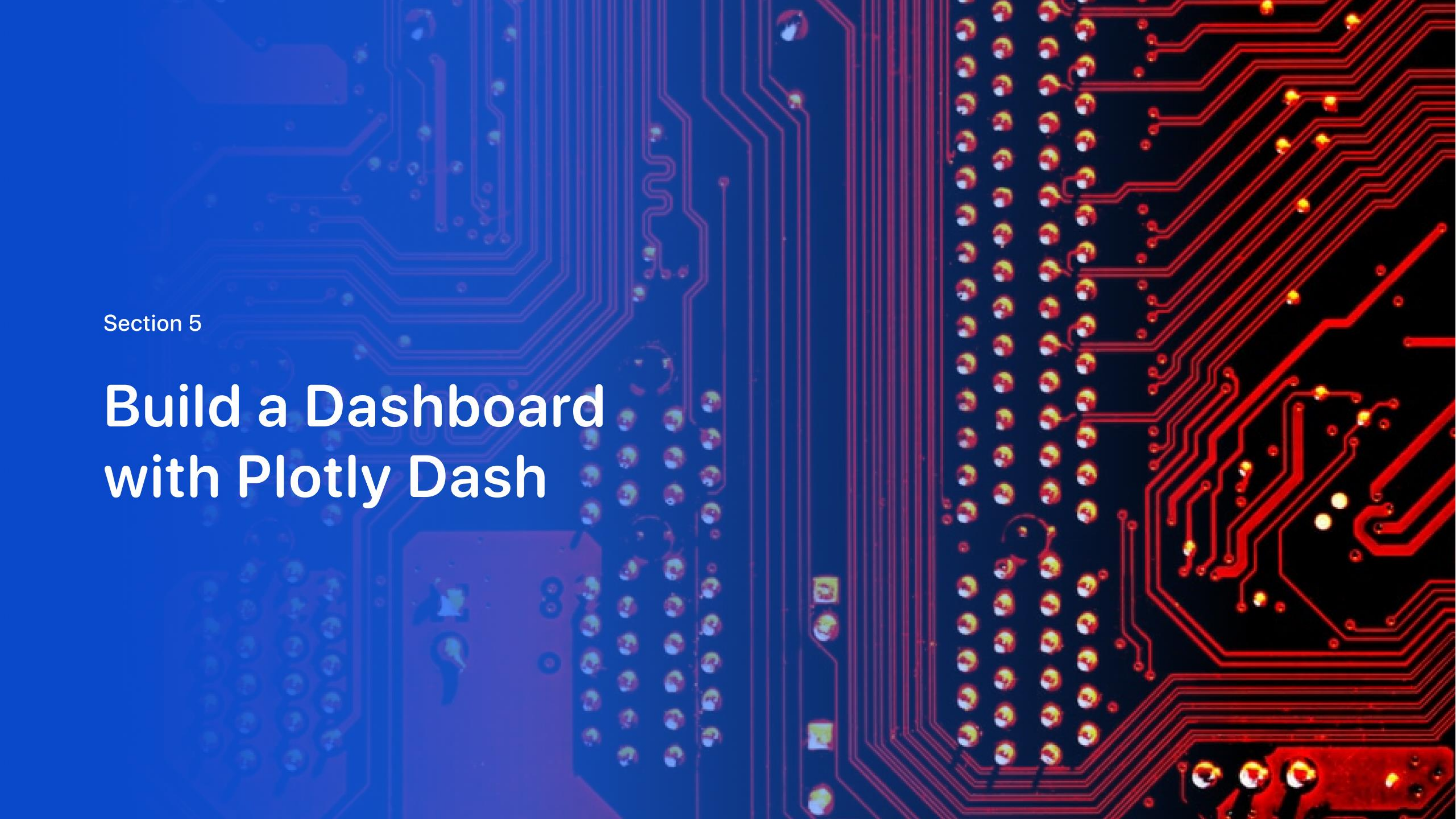


7
CCAFS SLC-40
26
CCAFS LC-40
Distance to coast





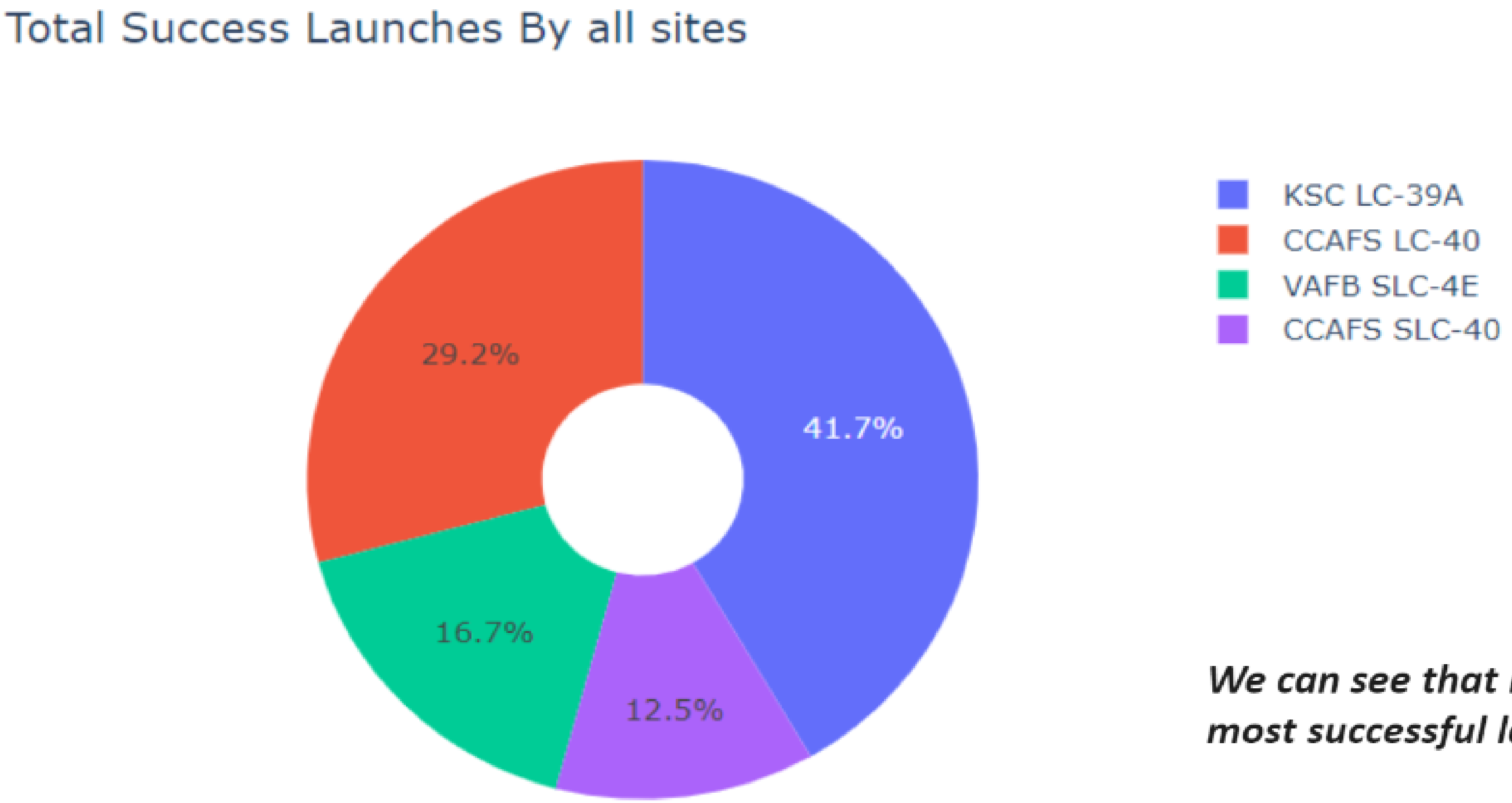
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes



Section 5

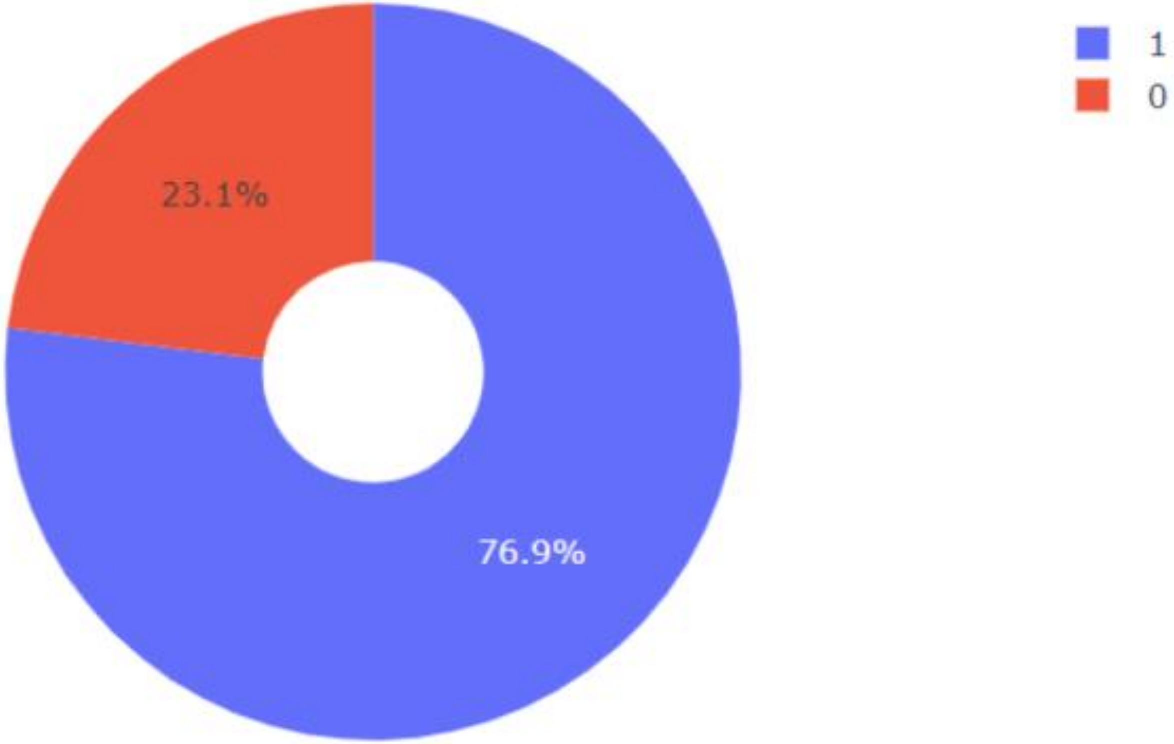
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



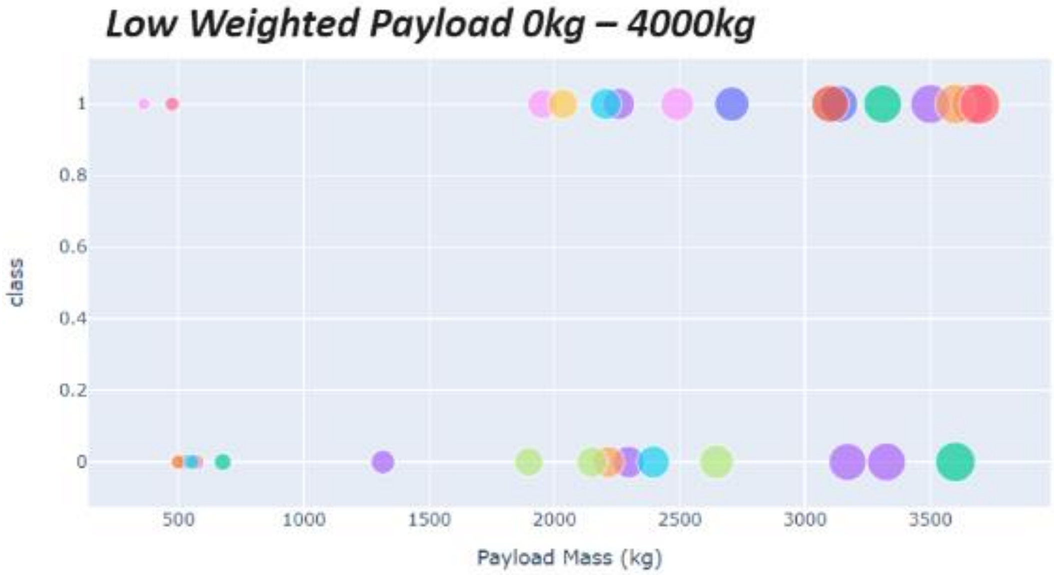
We can see that KSC LC-39A had the most successful launches from all the sites

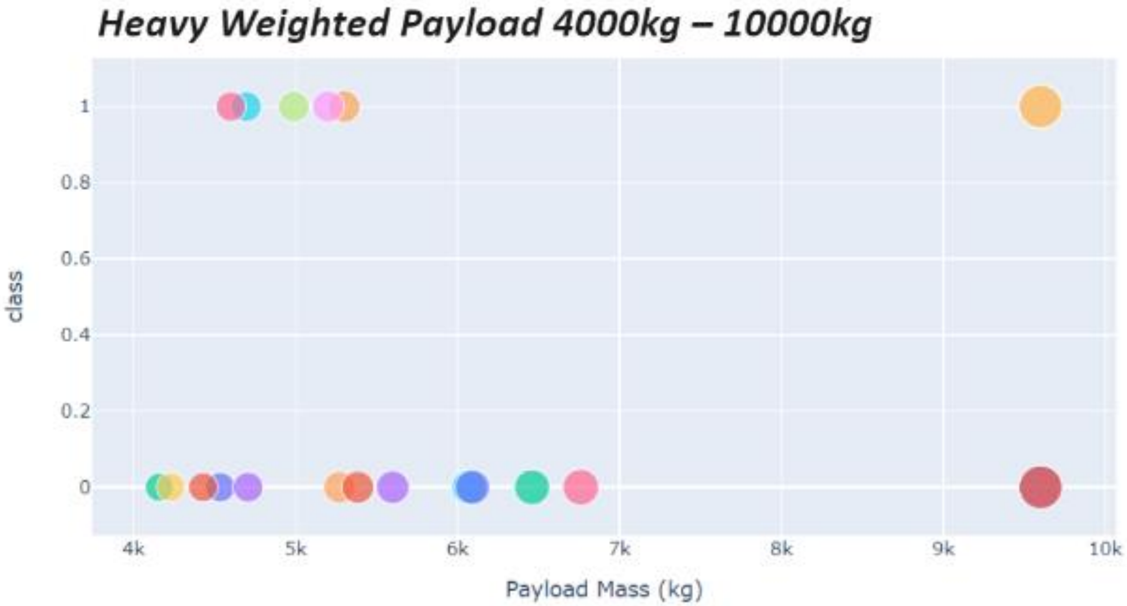
Pie chart showing the Launch site with the highest launch success ratio



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

Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider





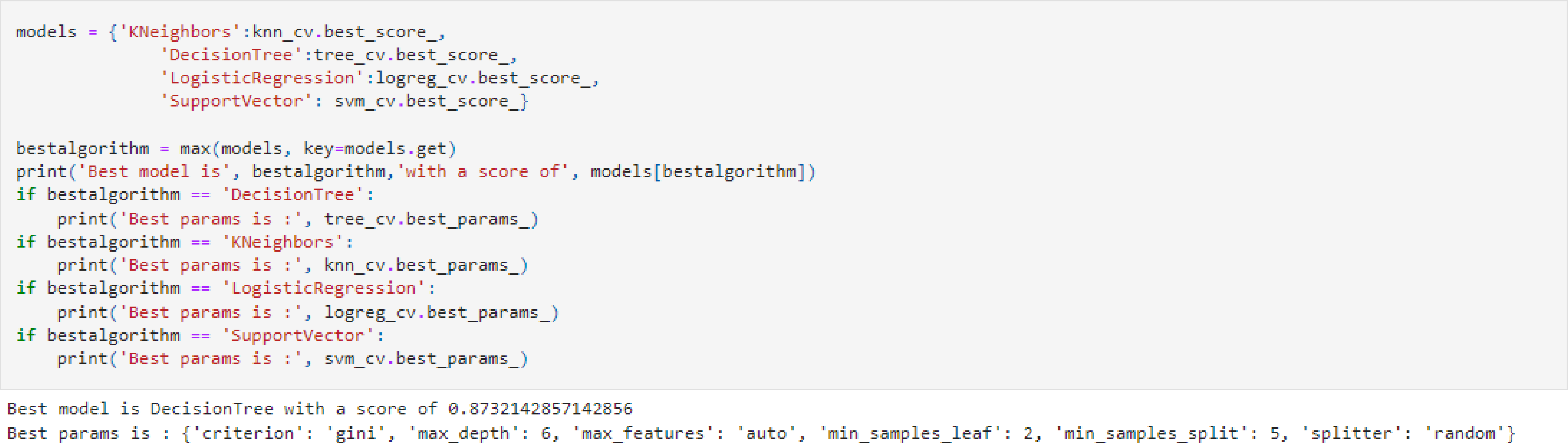
We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 6

Predictive Analysis (Classification)

Classification Accuracy

- The decision tree classifier has the highest classification accuracy.



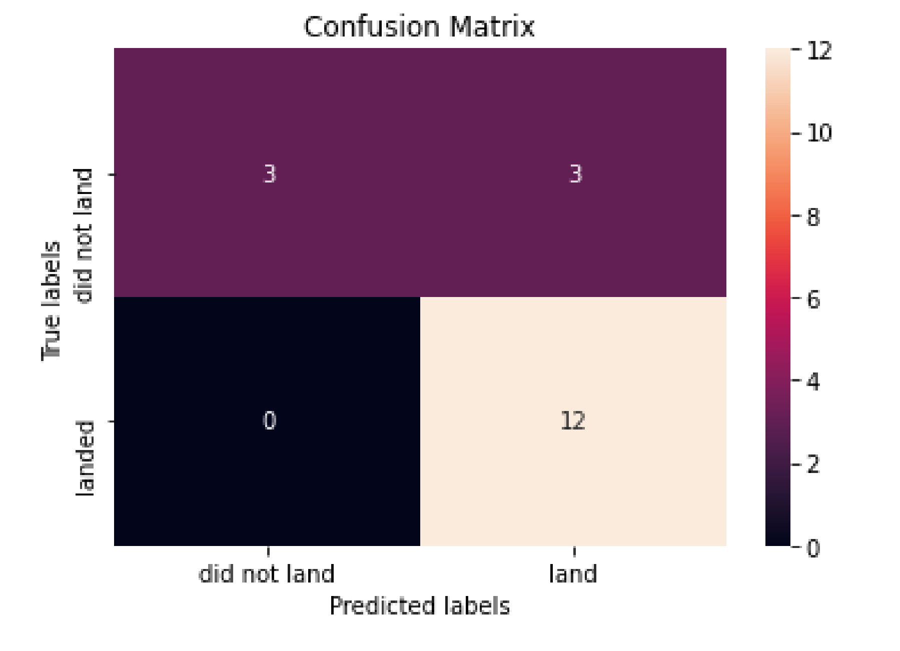
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

Confusion Matrix

- The confusion matrix for the decision tree classifier demonstrates that it can discriminate between the various classes. The biggest issue is false positives.i.e., a failed landing is recorded as successful by the classifier.



Conclusions

We can conclude that:

- Increasing the number of flights at a launch site correlates with higher success rates.
- From 2013 to 2020, the launch success rate increased.
- Orbits ES-L1, GEO, HEO, SSO, and VLEO achieved the highest success rate.
- KSC LC-39A was the most successful launch of all locations.
- The Decision tree classifier is the most effective machine learning method for this problem.

Thank you!