IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Ankur Bhargava
17-06-2024

## Outline

## 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 advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Problems you want to find answers

  - What factors determine if the rocket will land successfully?
  - The interaction amongst various features that determine the success rate of a successful landing.
  - What conditions needs to be in place to ensure 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

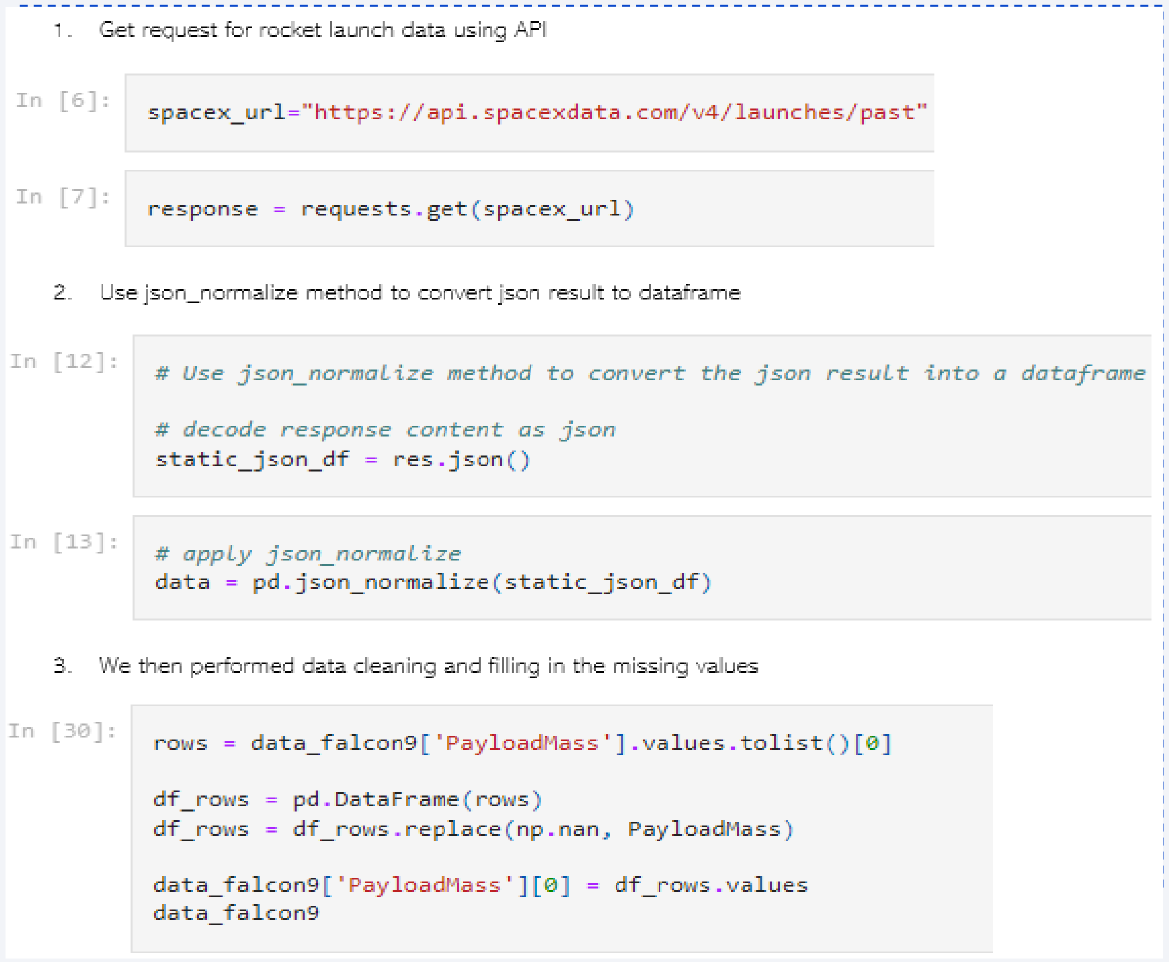
## Data Collection

- Data collection Methods
  - Data collection was done using SpaceX API which provide response with Json
  - 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

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

- Add the GitHub URL of the completed SpaceX API calls notebook (https://github.com/ankurbhargava0511/-IBM-Data-Science-Capstone-SpaceX/blob/main/Week%201a%20jupyter-labs-spacex-data-collection-api.ipynb), as an external reference and peer-review purpose



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)
```

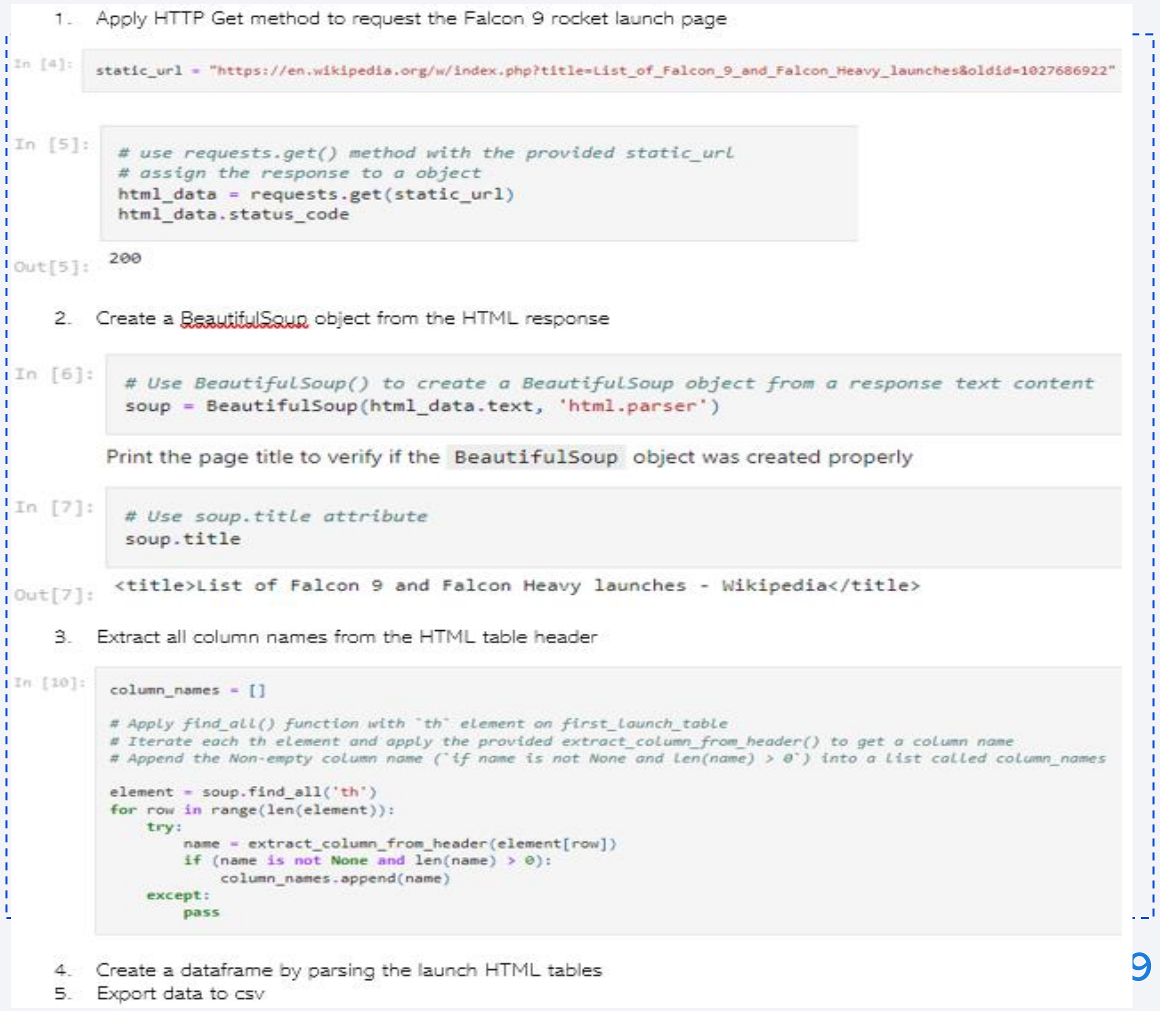
3. We then performed data cleaning and filling in the missing values

```
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

- Present your web scraping process using key phrases and flowcharts

- [https://github.com/ankurbhargava0511/-IBM-Data-Science-Capstone-SpaceX/blob/main/Week%201b%20jupyter-labs-webscraping.ipynb](https://github.com/ankurbhargava0511/-IBM-Data-Science-Capstone-SpaceX/blob/main/Week%201b%20jupyter-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>
```

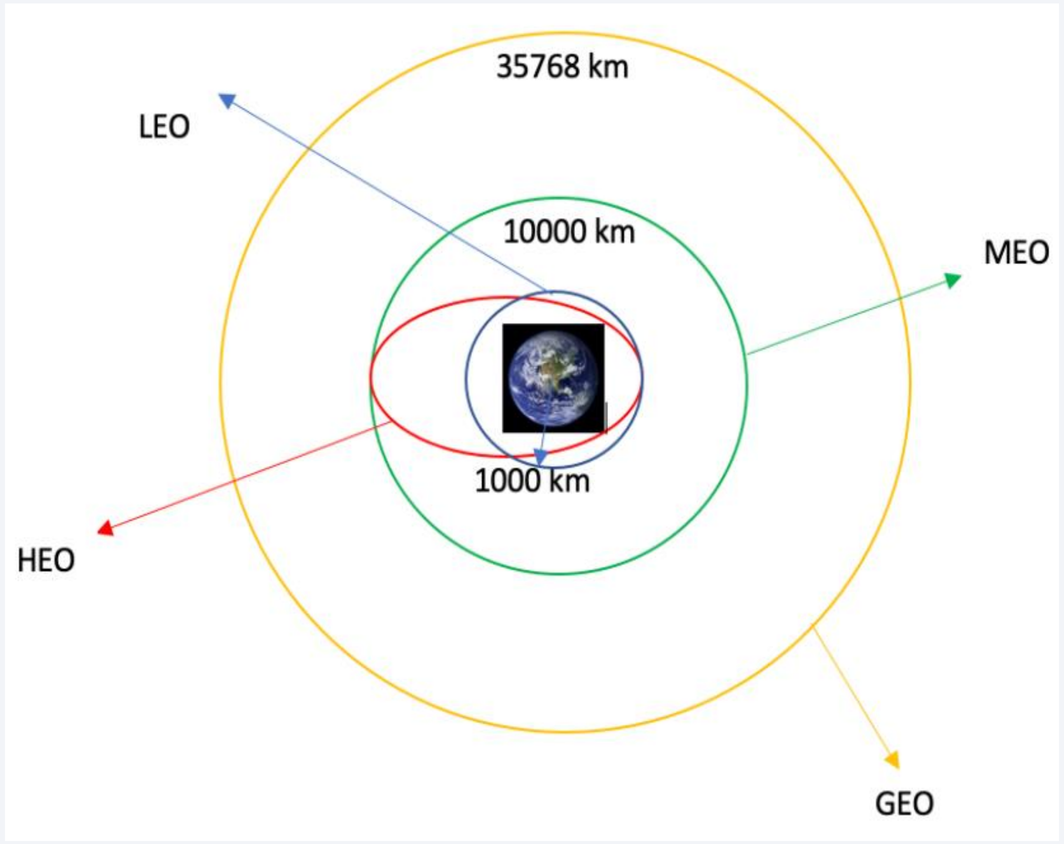
3. Extract all column names from the HTML table header

```
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

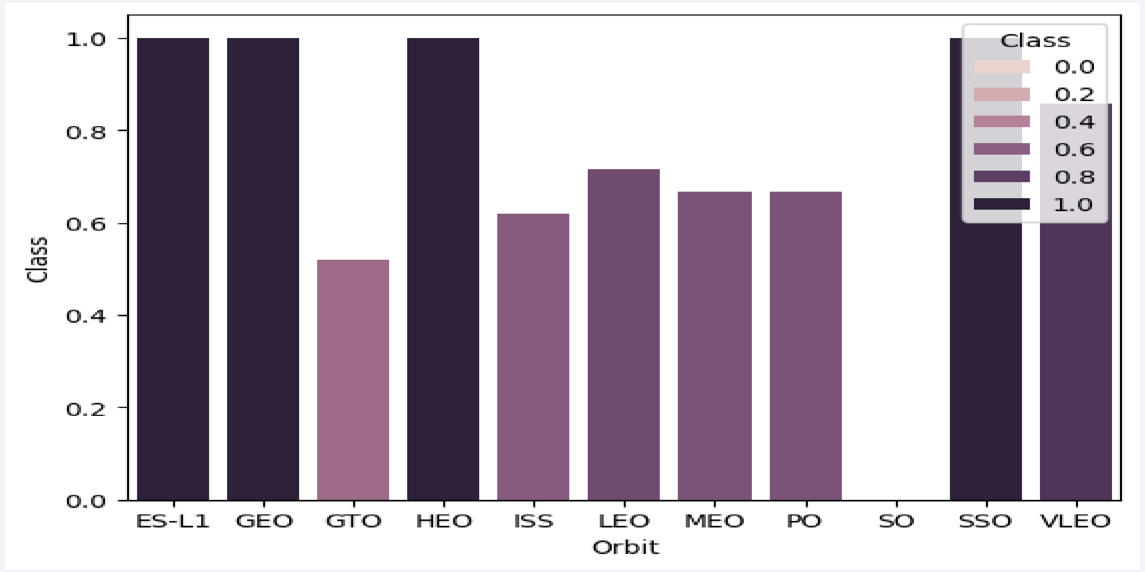
## Data Wrangling

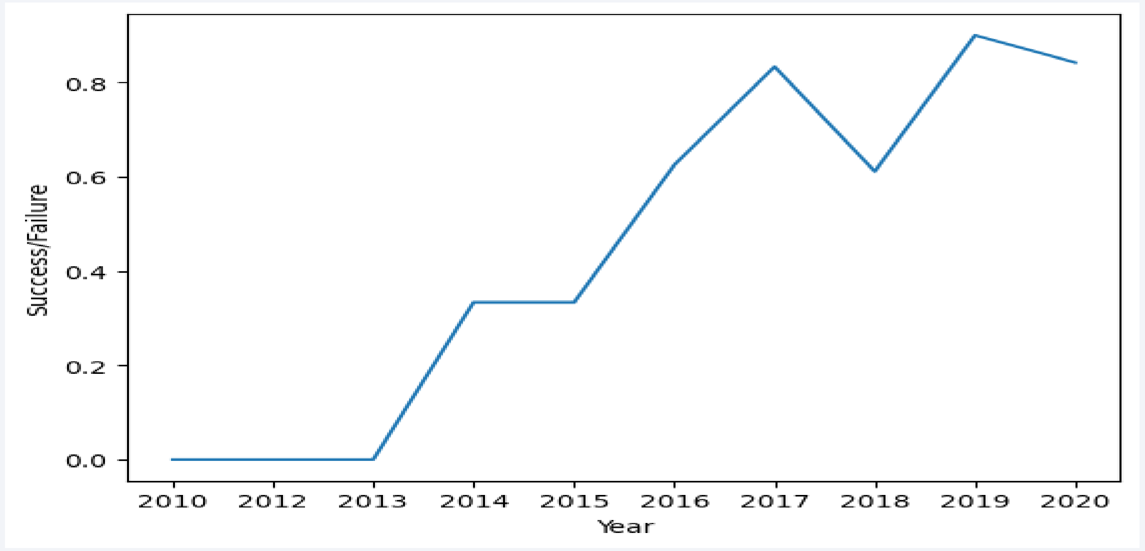
- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- [https://github.com/ankurbhargava0511/-IBM-Data-Science-Capstone-SpaceX/blob/main/Week%201b%20jupyter-labs-webscraping.ipynb](https://github.com/ankurbhargava0511/-IBM-Data-Science-Capstone-SpaceX/blob/main/Week%201b%20jupyter-labs-webscraping.ipynb)



# EDA with Data Visualization

- We explored the data by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.
- https://github.com/ankurbhargava0511/-IBM-Data-Science-Capstone-SpaceX/blob/main/Week%202b%20edadataviz.ipynb





## EDA with SQL

- We loaded the SpaceX dataset into a PostgreSQL database without leaving the jupyter notebook.
- We applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:
  - The names of unique launch sites in the space mission.
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - The total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names.
- Notebook- https://github.com/ankurbhargava0511/-IBM-Data-Science-Capstone-SpaceX/blob/main/Week%202a%20jupyter-labs-eda-sql-coursera_sqllite.ipynb

## Build an Interactive Map with Folium

- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:
  - Are launch sites near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities.

## Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The link to the notebook is https://github.com/ankurbhargava0511/-IBM-Data-Science-Capstone-SpaceX/blob/main/Week%203b%20Plotly%20spacex_dash_app%20(1).py

## Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- The link to the notebook is https://github.com/ankurbhargava0511/-IBM-Data-Science-Capstone-SpaceX/blob/main/Week%204aSpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

## Results

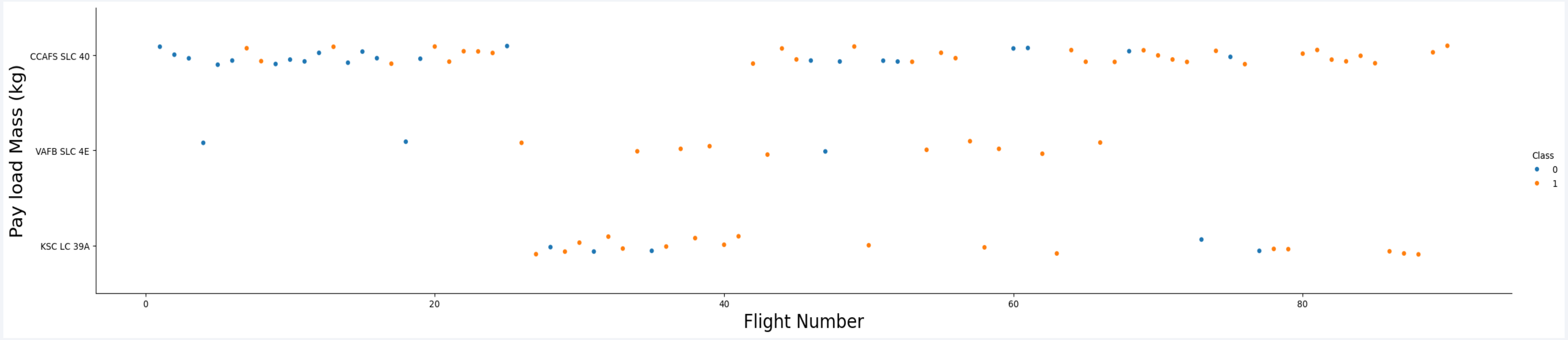
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

## Insights drawn from EDA

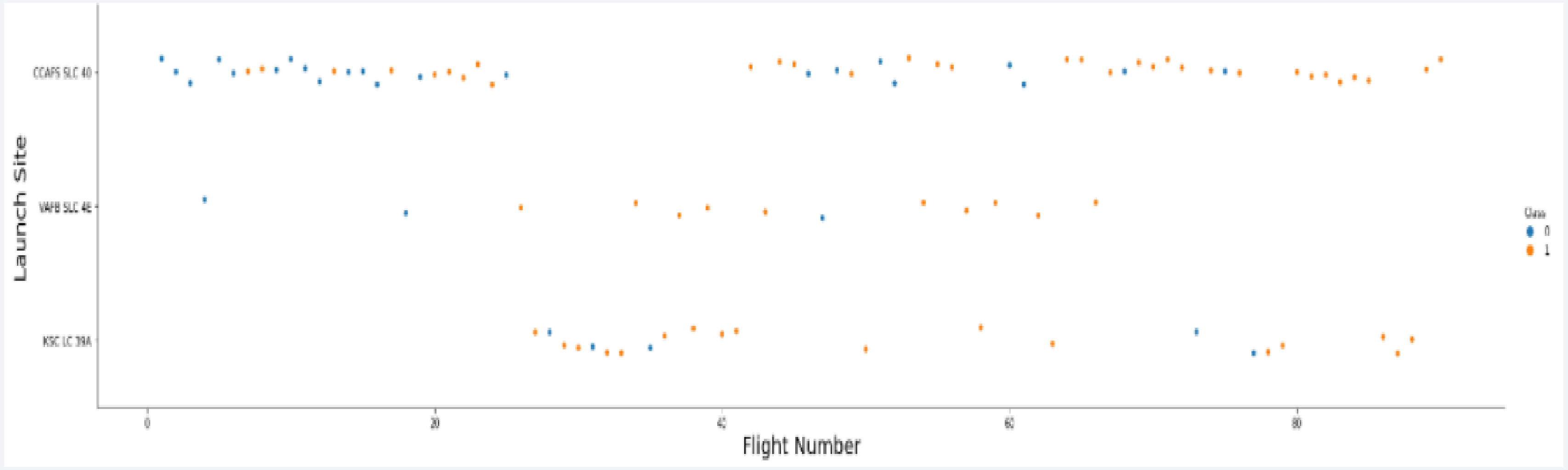
# Flight Number vs. Launch Site

- Show a scatter plot of Flight Number vs. Launch Site

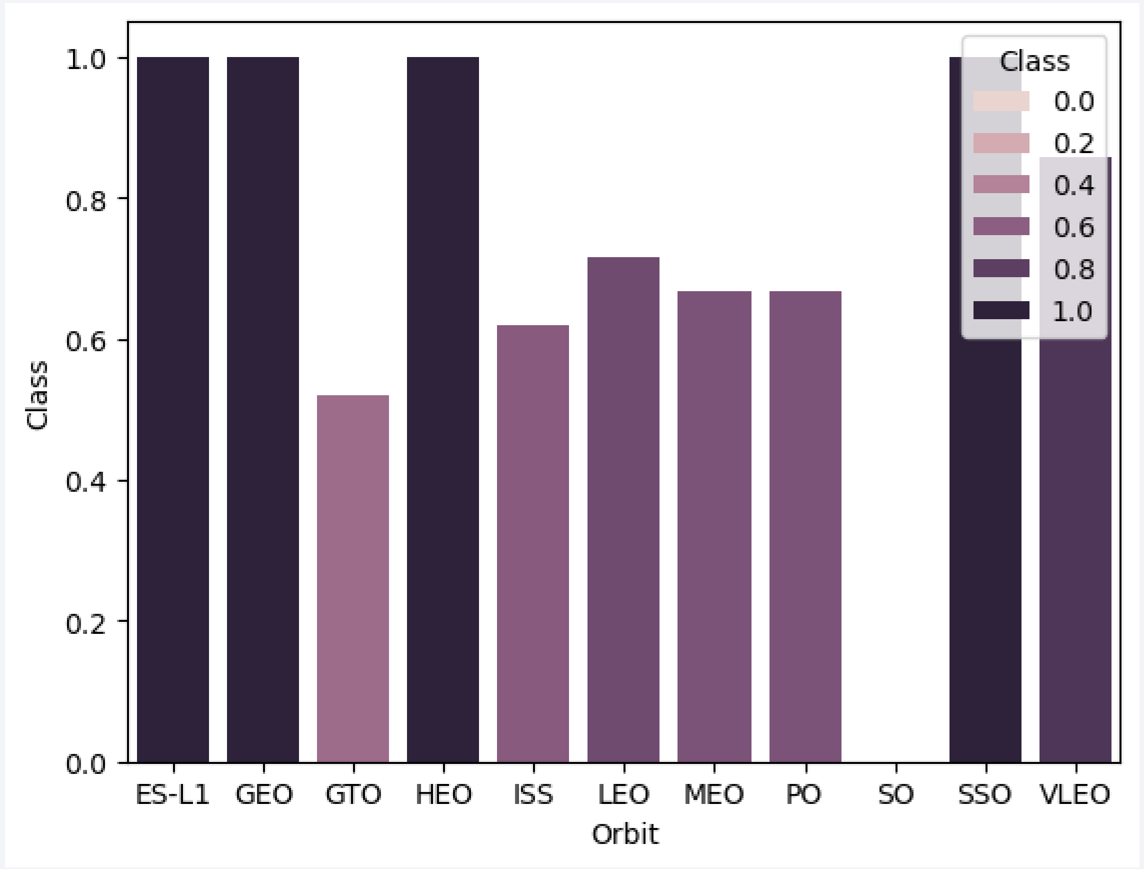


# Payload vs. Launch Site

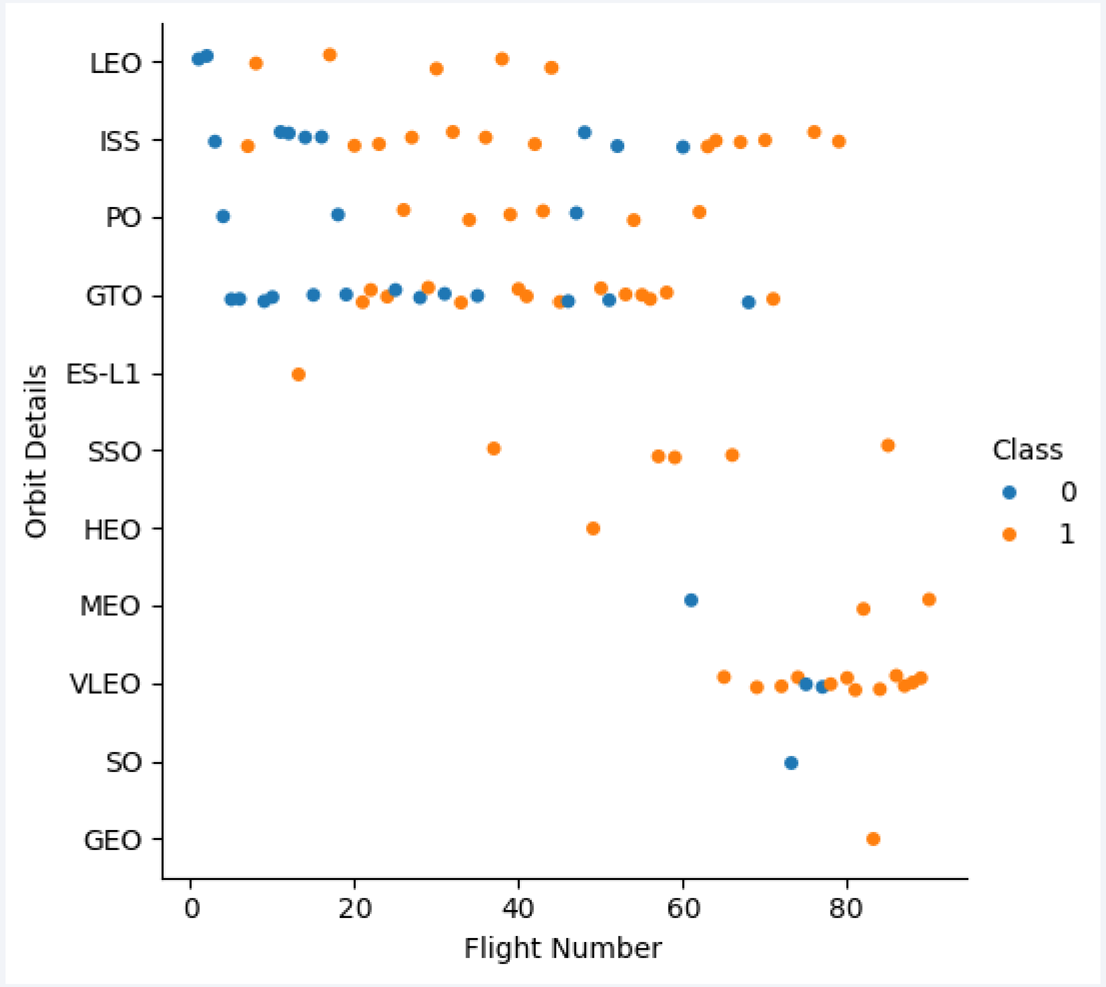
- Show a scatter plot of Payload vs. Launch Site



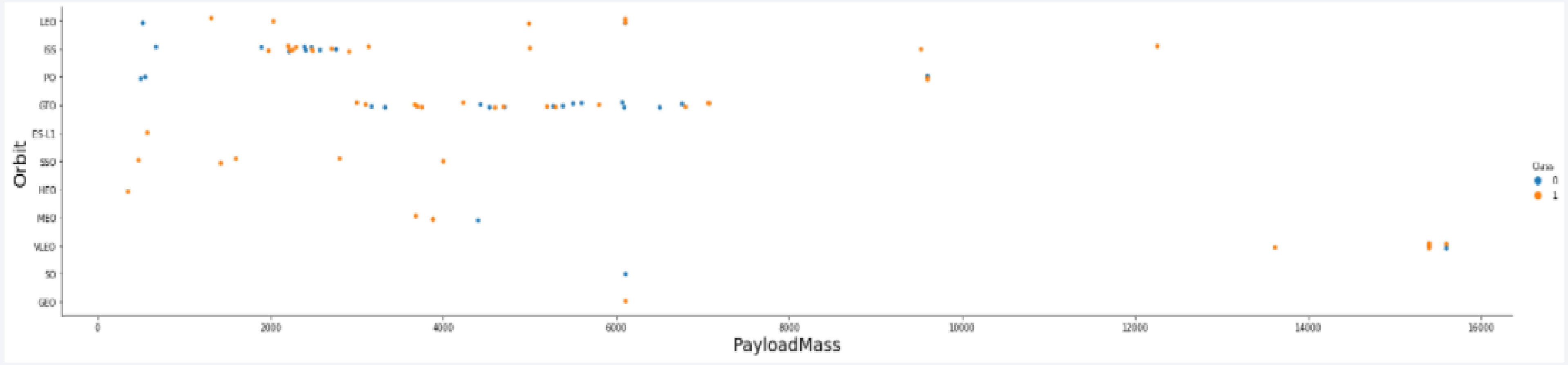
## Success Rate vs. Orbit Type



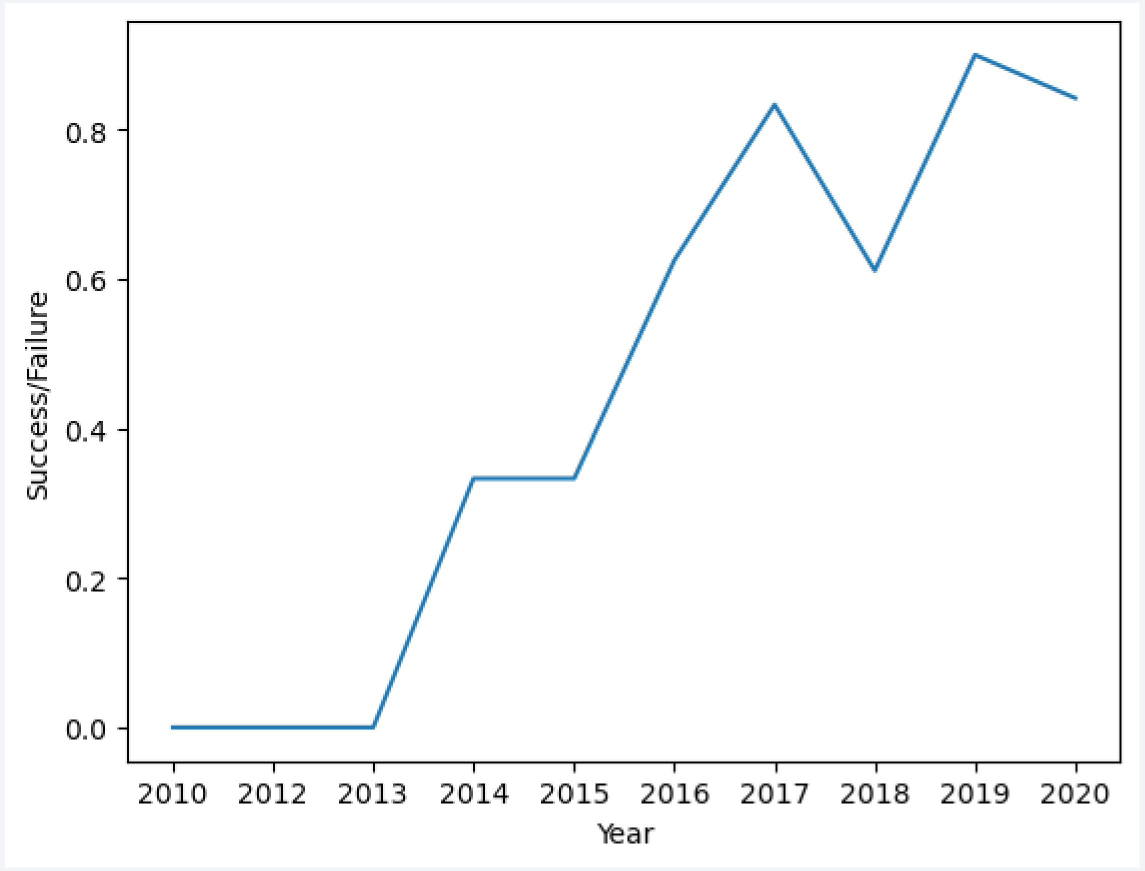
# Flight Number vs. Orbit Type



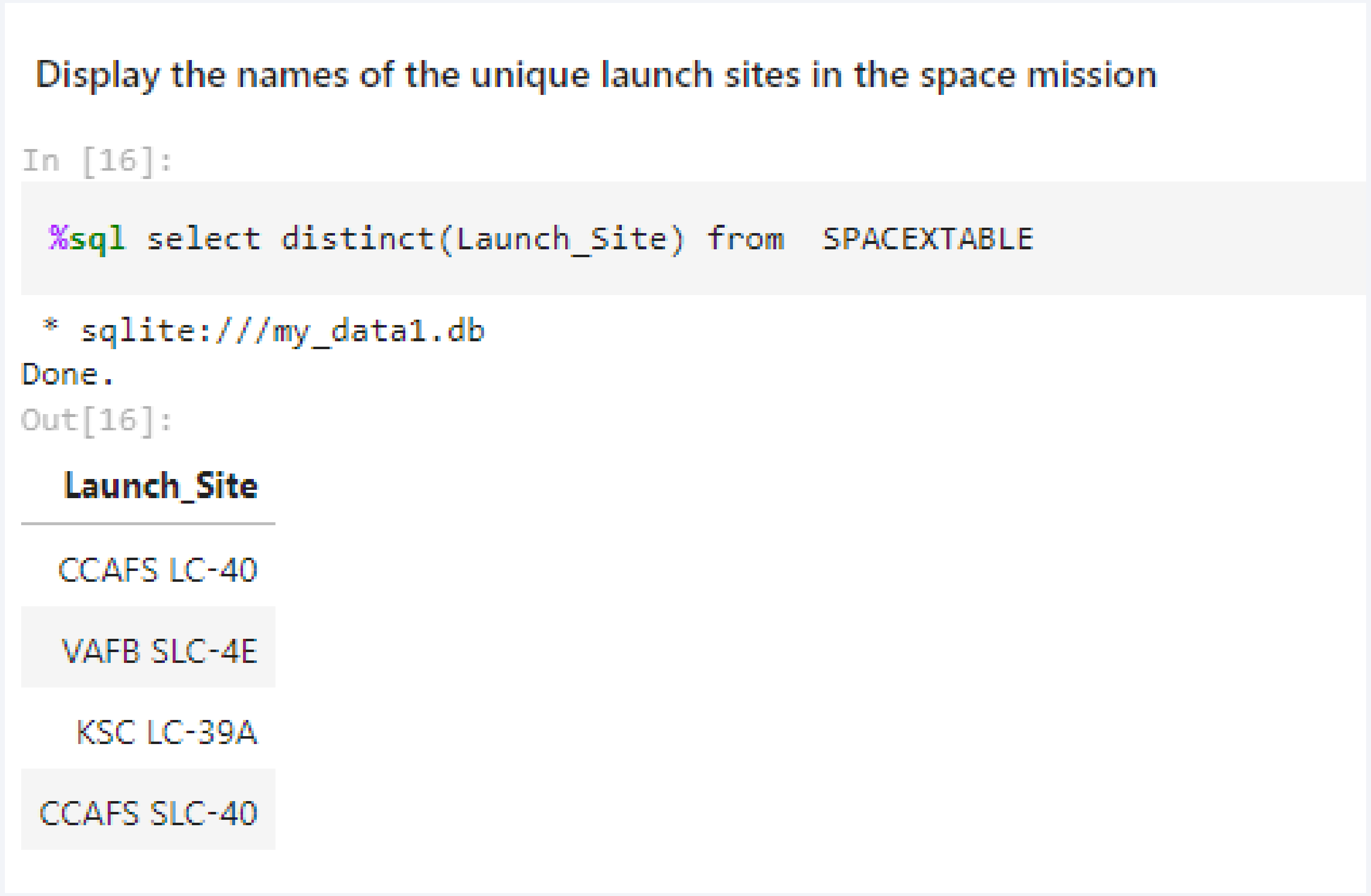
# Payload vs. Orbit Type



## Launch Success Yearly Trend



# All Launch Site Names



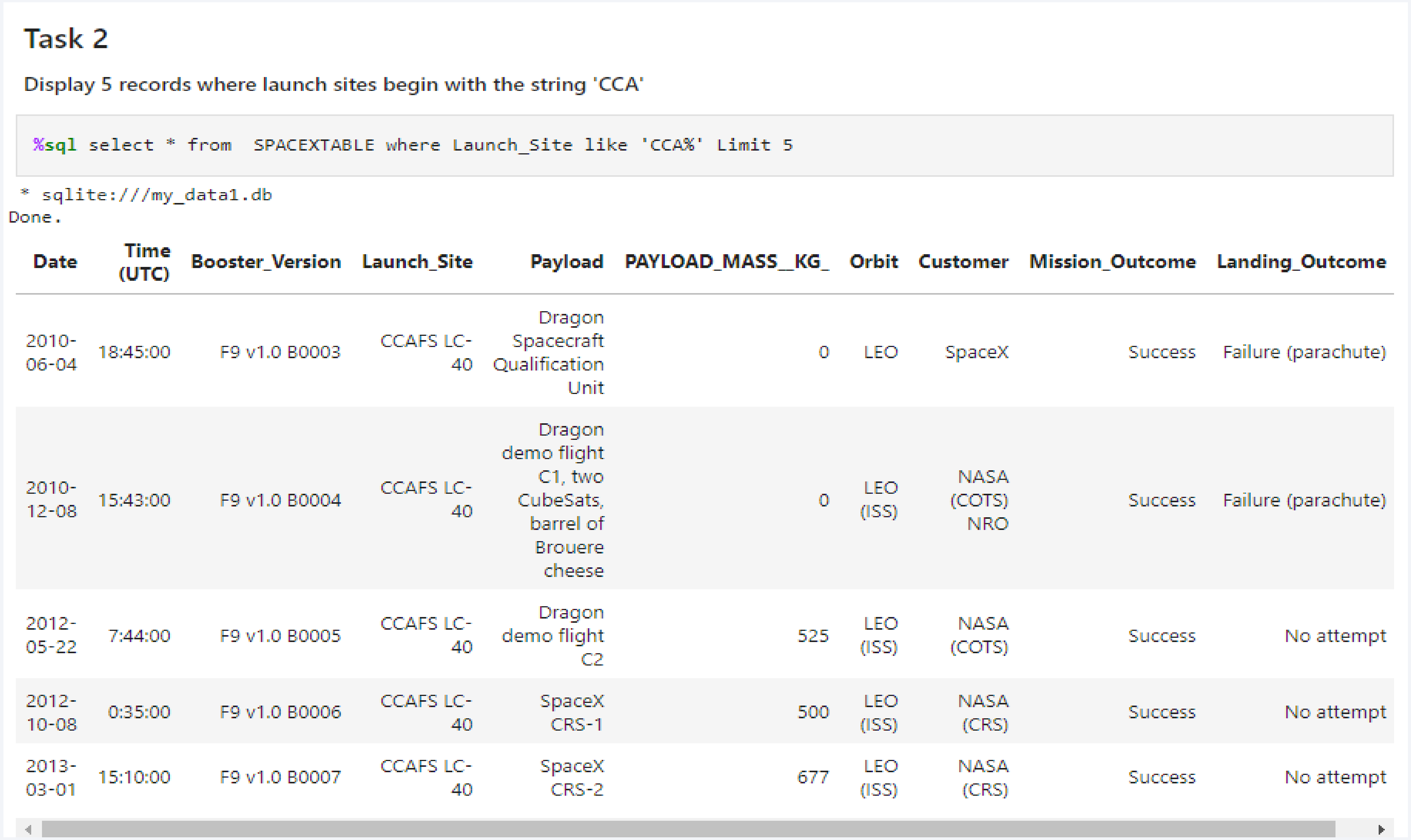
Display the names of the unique launch sites in the space mission

```
In [16]:
%sql select distinct(Launch_Site) from  SPACEXTABLE
```

```
 * sqlite:///my_data1.db
Done.
Out[16]:
```

| Launch_Site |
|---|
| CCAFS LC-40 |
| VAFB SLC-4E |
| KSC LC-39A |
| CCAFS SLC-40 |

## Launch Site Names Begin with 'CCA'



### Task 2

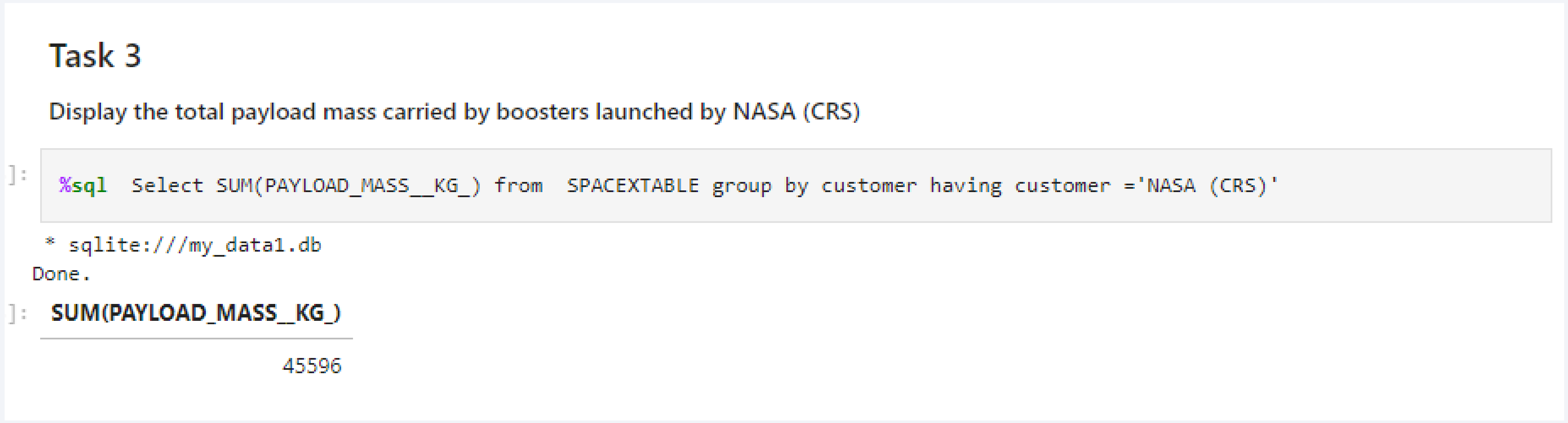
Display 5 records where launch sites begin with the string 'CCA'

```
%sql select * from  SPACEXTABLE where Launch_Site like 'CCA%' Limit 5
```

```
 * sqlite:///my_data1.db
Done.
```

| Date | Time (UTC) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS__KG_ | Orbit | Customer | Mission_Outcome | Landing_Outcome |
|---|---|---|---|---|---|---|---|---|---|
| 2010-06-04 | 18:45:00 | F9 v1.0 B0003 | CCAFS LC-40 | Dragon Spacecraft Qualification Unit | 0 | LEO | SpaceX | Success | Failure (parachute) |
| 2010-12-08 | 15:43:00 | F9 v1.0 B0004 | CCAFS LC-40 | Dragon demo flight C1, two CubeSats, barrel of Brouere cheese | 0 | LEO (ISS) | NASA (COTS) NRO | Success | Failure (parachute) |
| 2012-05-22 | 7:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 | LEO (ISS) | NASA (COTS) | Success | No attempt |
| 2012-10-08 | 0:35:00 | F9 v1.0 B0006 | CCAFS LC-40 | SpaceX CRS-1 | 500 | LEO (ISS) | NASA (CRS) | Success | No attempt |
| 2013-03-01 | 15:10:00 | F9 v1.0 B0007 | CCAFS LC-40 | SpaceX CRS-2 | 677 | LEO (ISS) | NASA (CRS) | Success | No attempt |

## Total Payload Mass



### Task 3

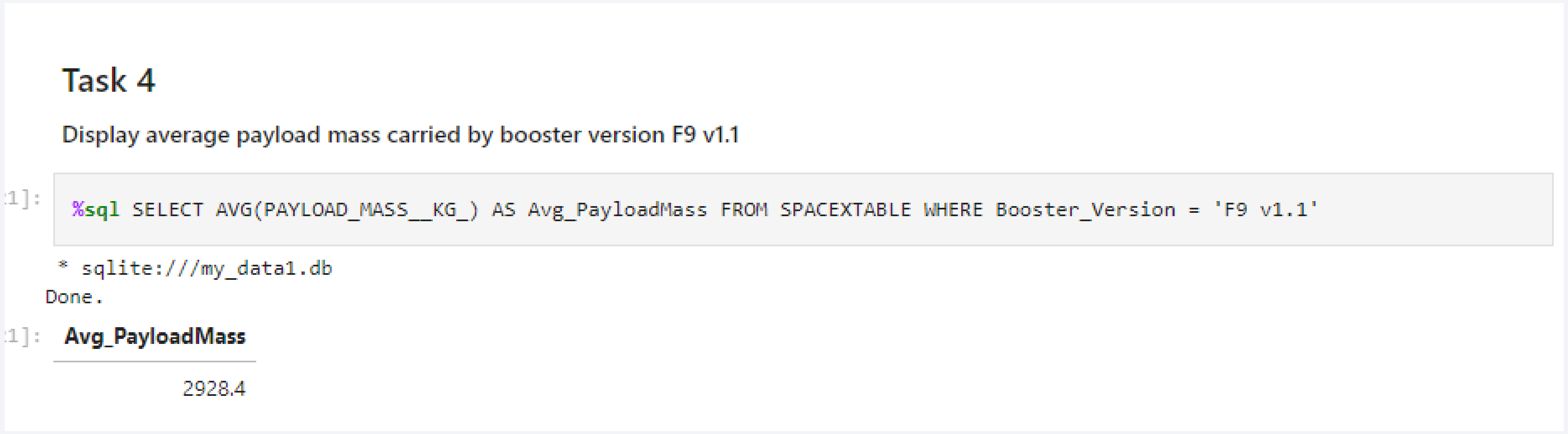
Display the total payload mass carried by boosters launched by NASA (CRS)

```
%sql  Select SUM(PAYLOAD_MASS__KG_) from  SPACEXTABLE group by customer having customer ='NASA (CRS)'
```

```
 * sqlite:///my_data1.db
Done.
```

| SUM(PAYLOAD_MASS__KG_) |
|---|
| 45596 |

# Average Payload Mass by F9 v1.1



### Task 4

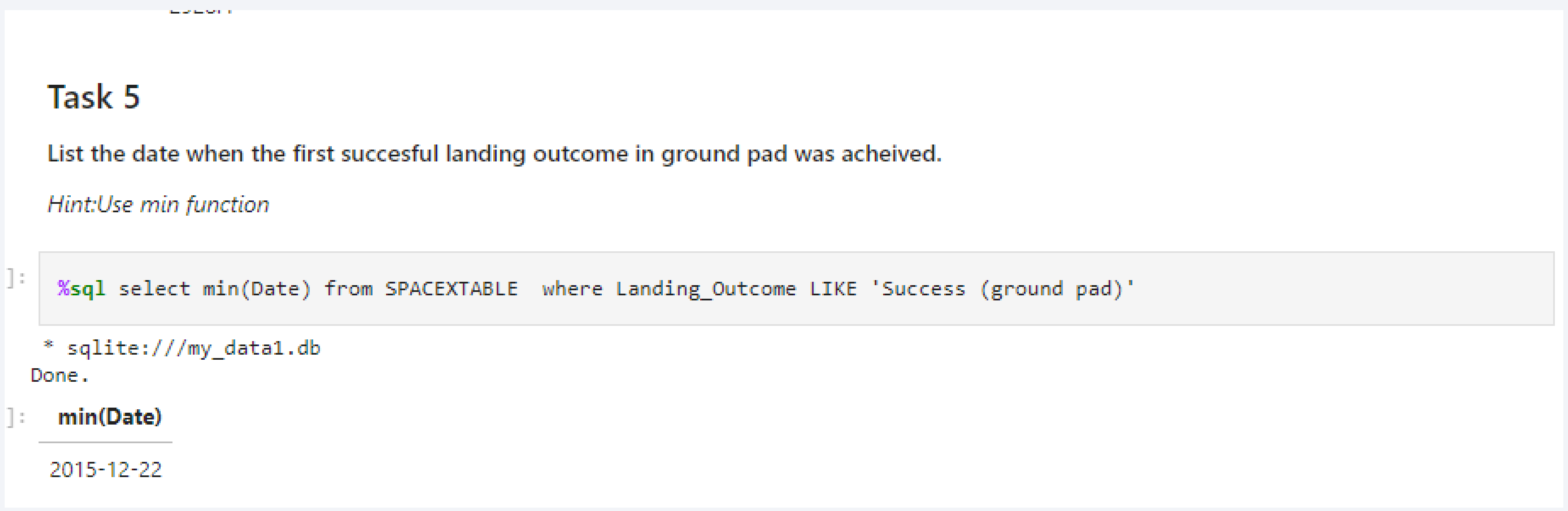
Display average payload mass carried by booster version F9 v1.1

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS Avg_PayloadMass FROM SPACEXTABLE WHERE Booster_Version = 'F9 v1.1'
```

```
 * sqlite:///my_data1.db
Done.
```

| Avg_PayloadMass |
|---|
| 2928.4 |

## First Successful Ground Landing Date



### Task 5

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

*Hint:Use min function*

```
]: %sql select min(Date) from SPACEXTABLE  where Landing_Outcome LIKE 'Success (ground pad)'
```

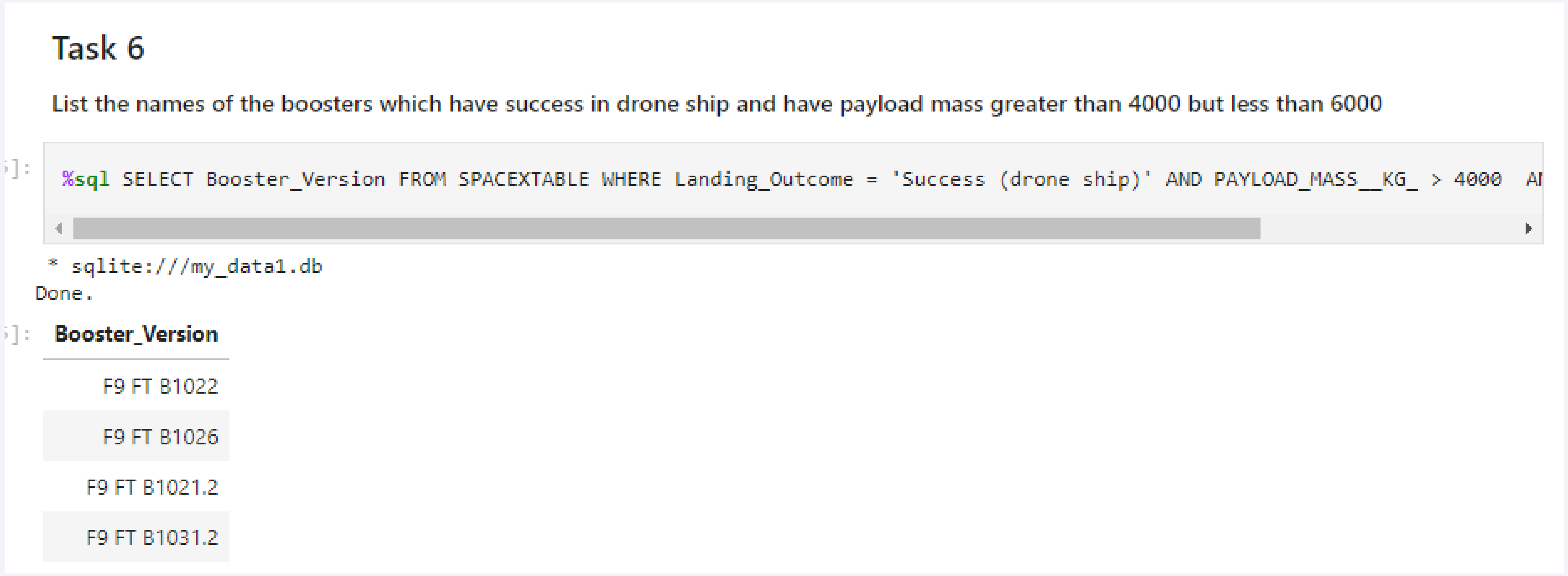
```
 * sqlite:///my_data1.db
Done.
```

]:

| min(Date) |
|---|
| 2015-12-22 |

## Successful Drone Ship Landing with Payload between 4000 and 6000

%sql SELECT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000



### Task 6

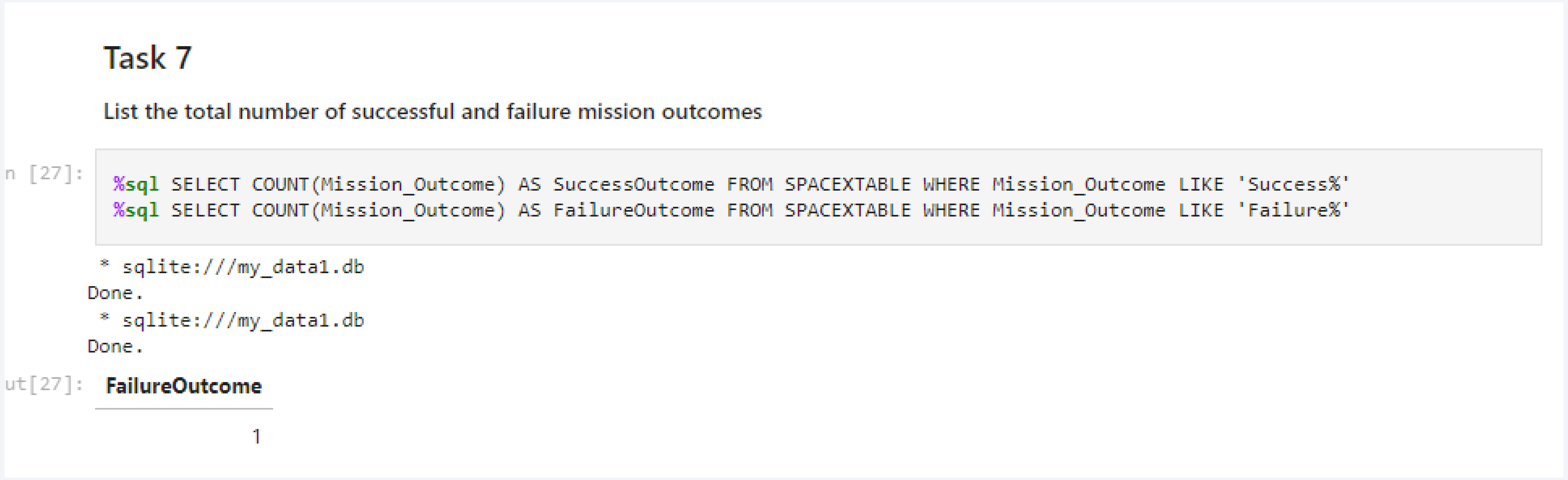
List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
]: %sql SELECT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000  AI
```

```
 * sqlite:///my_data1.db
Done.
```

| Booster_Version |
|---|
| F9 FT B1022 |
| F9 FT B1026 |
| F9 FT B1021.2 |
| F9 FT B1031.2 |

## Total Number of Successful and Failure Mission Outcomes



### Task 7

List the total number of successful and failure mission outcomes

```
n [27]: %sql SELECT COUNT(Mission_Outcome) AS SuccessOutcome FROM SPACEXTABLE WHERE Mission_Outcome LIKE 'Success%'
        %sql SELECT COUNT(Mission_Outcome) AS FailureOutcome FROM SPACEXTABLE WHERE Mission_Outcome LIKE 'Failure%'
```

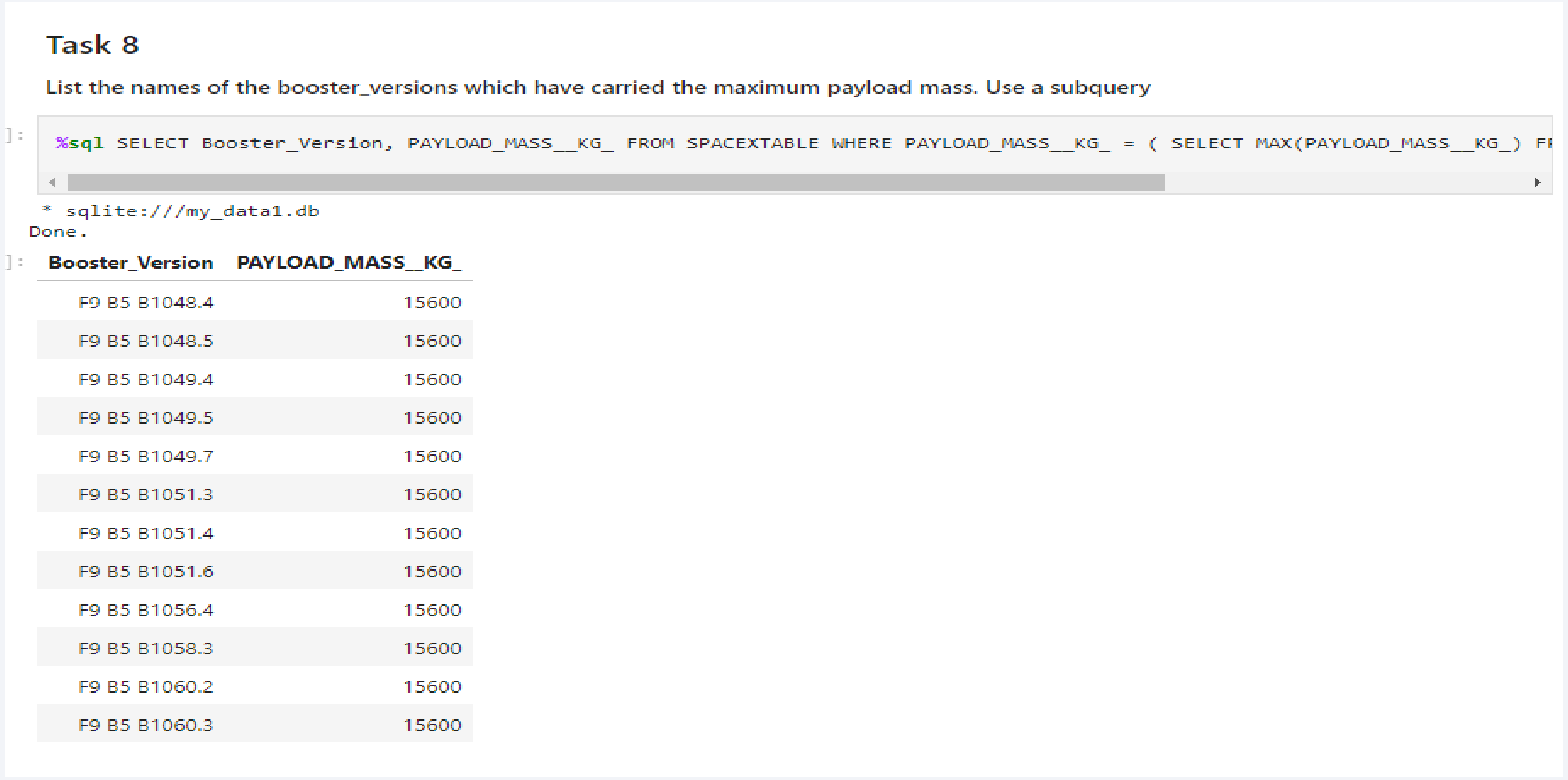
```
 * sqlite:///my_data1.db
Done.
 * sqlite:///my_data1.db
Done.
```

ut[27]:

| FailureOutcome |
|---|
| 1 |

# Boosters Carried Maximum Payload

- **%sql** SELECT Booster_Version, PAYLOAD_MASS__KG_ FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = ( SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE) ORDER BY Booster_Version



Task 8

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

```
]: %sql SELECT Booster_Version, PAYLOAD_MASS__KG_ FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = ( SELECT MAX(PAYLOAD_MASS__KG_) FR
```

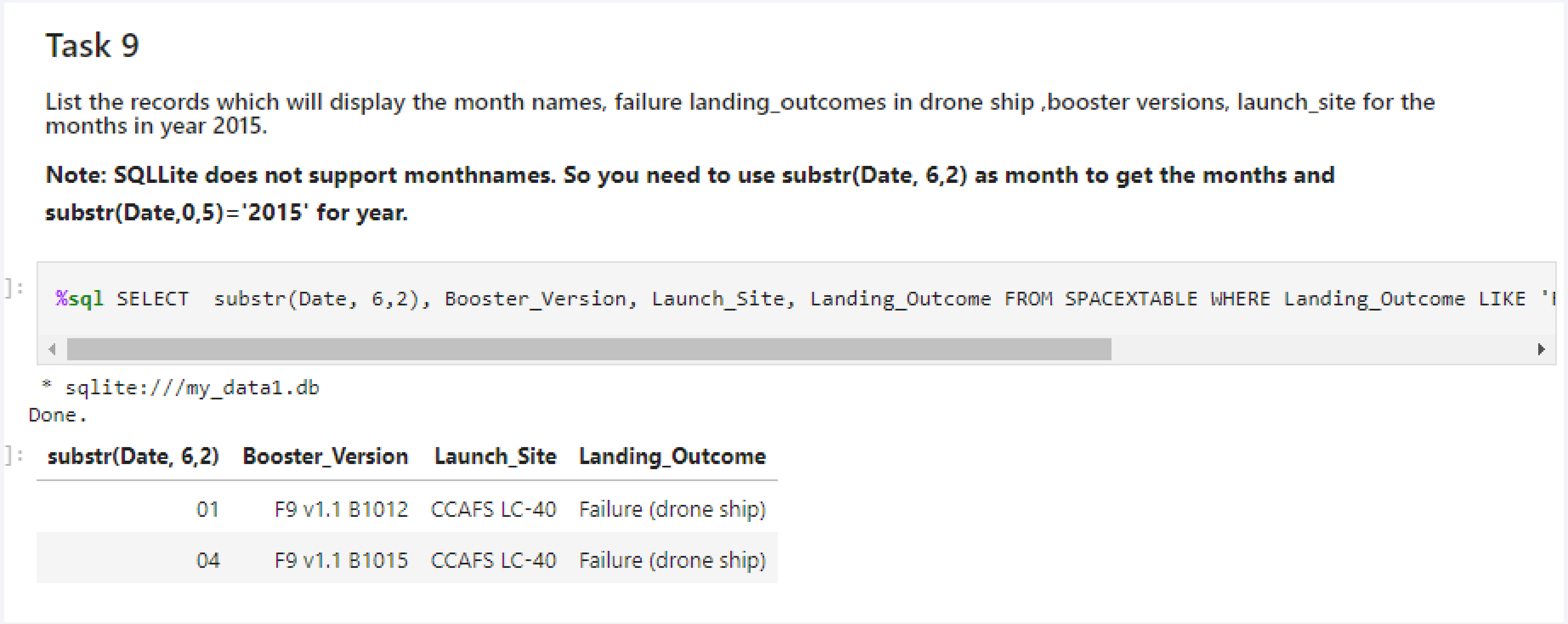
```
 * sqlite:///my_data1.db
Done.
```

]:

| Booster_Version | PAYLOAD_MASS__KG_ |
|---|---|
| F9 B5 B1048.4 | 15600 |
| F9 B5 B1048.5 | 15600 |
| F9 B5 B1049.4 | 15600 |
| F9 B5 B1049.5 | 15600 |
| F9 B5 B1049.7 | 15600 |
| F9 B5 B1051.3 | 15600 |
| F9 B5 B1051.4 | 15600 |
| F9 B5 B1051.6 | 15600 |
| F9 B5 B1056.4 | 15600 |
| F9 B5 B1058.3 | 15600 |
| F9 B5 B1060.2 | 15600 |
| F9 B5 B1060.3 | 15600 |

# 2015 Launch Records

%sql SELECT substr(Date, 6,2), Booster_Version, Launch_Site, Landing_Outcome FROM SPACEXTABLE WHERE Landing_Outcome LIKE 'Failure (drone ship)' AND substr(Date,0,5)='2015'



## Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

**Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.**

```
]: %sql SELECT  substr(Date, 6,2), Booster_Version, Launch_Site, Landing_Outcome FROM SPACEXTABLE WHERE Landing_Outcome LIKE 'F
```

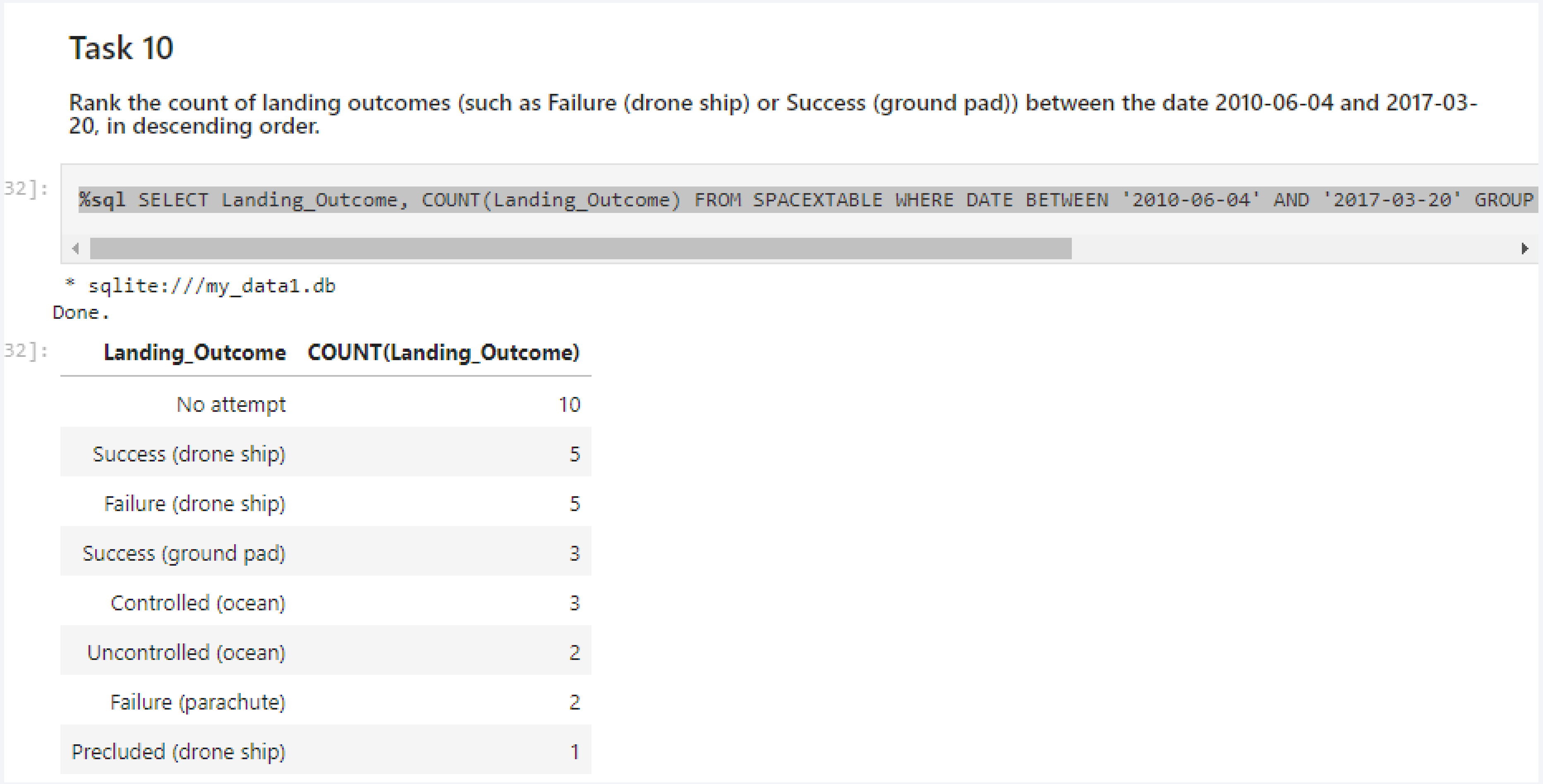
```
 * sqlite:///my_data1.db
Done.
```

]:

| substr(Date, 6,2) | Booster_Version | Launch_Site | Landing_Outcome |
|---|---|---|---|
| 01 | F9 v1.1 B1012 | CCAFS LC-40 | Failure (drone ship) |
| 04 | F9 v1.1 B1015 | CCAFS LC-40 | Failure (drone ship) |

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

%sql SELECT Landing_Outcome, COUNT(Landing_Outcome) FROM SPACEXTABLE WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outcome ORDER BY COUNT(Landing_Outcome) DESC



### 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.

```
32]:  %sql SELECT Landing_Outcome, COUNT(Landing_Outcome) FROM SPACEXTABLE WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP
```

```
 * sqlite:///my_data1.db
Done.
```

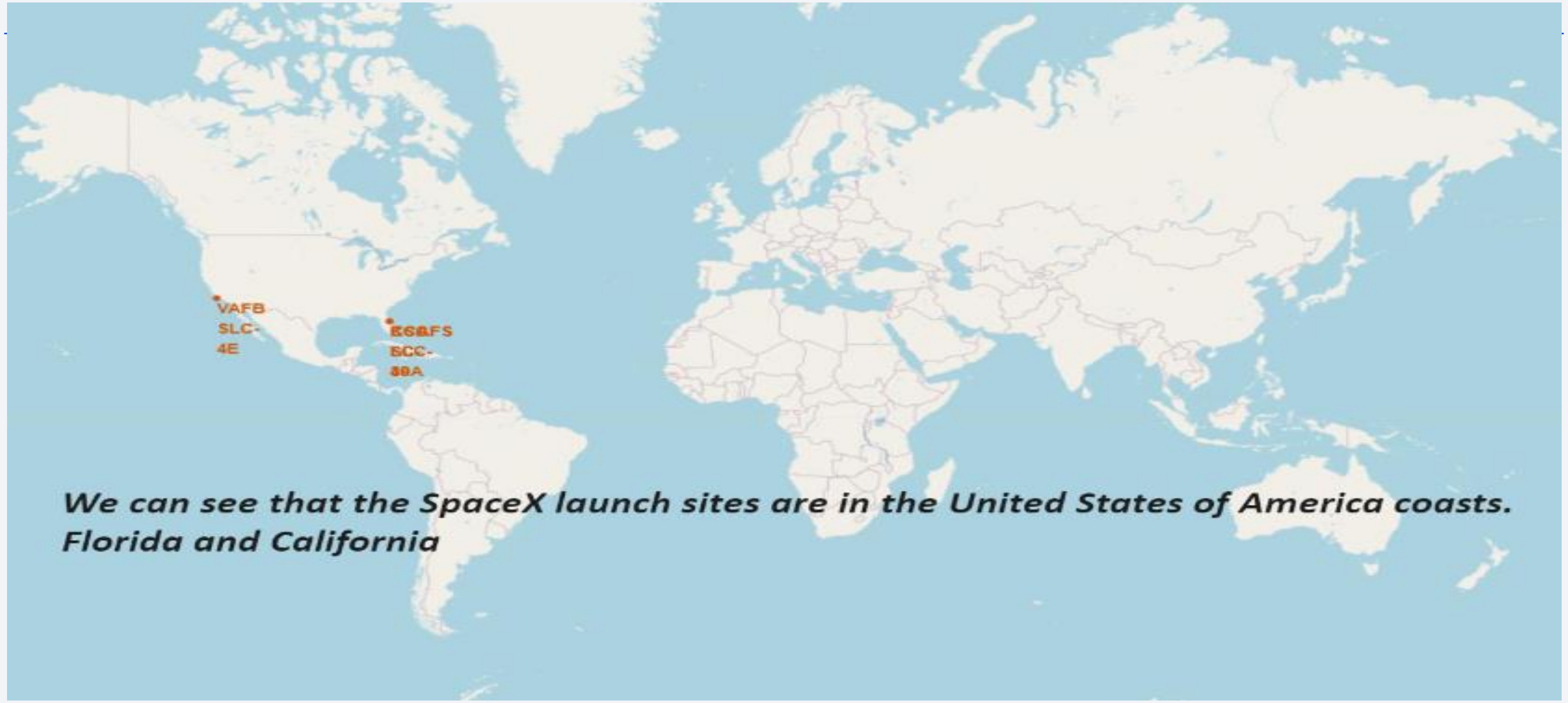
32]:

| Landing_Outcome | COUNT(Landing_Outcome) |
|---|---|
| No attempt | 10 |
| Success (drone ship) | 5 |
| Failure (drone ship) | 5 |
| Success (ground pad) | 3 |
| Controlled (ocean) | 3 |
| Uncontrolled (ocean) | 2 |
| Failure (parachute) | 2 |
| Precluded (drone ship) | 1 |

Section 3

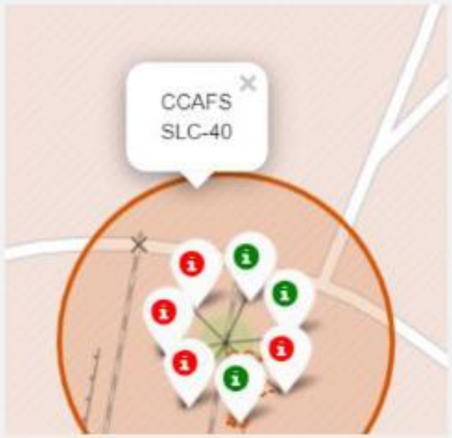
## Launch Sites Proximities Analysis

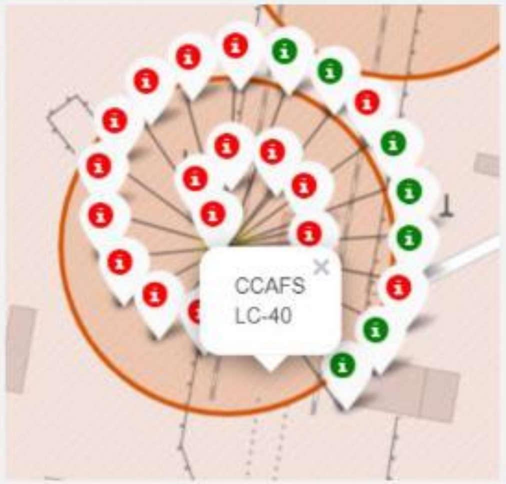
## All launch sites global map markers

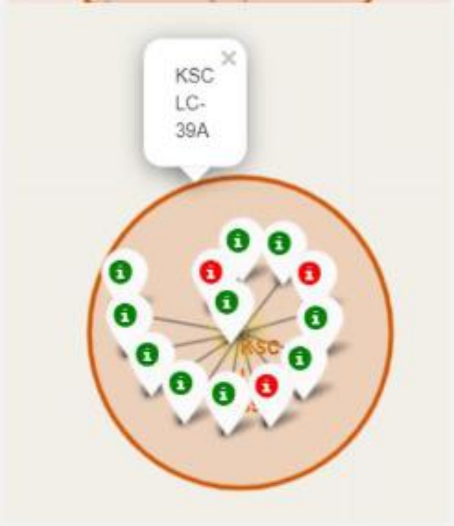


***We can see that the SpaceX launch sites are in the United States of America coasts. Florida and California***

# Markers showing launch sites with color labels

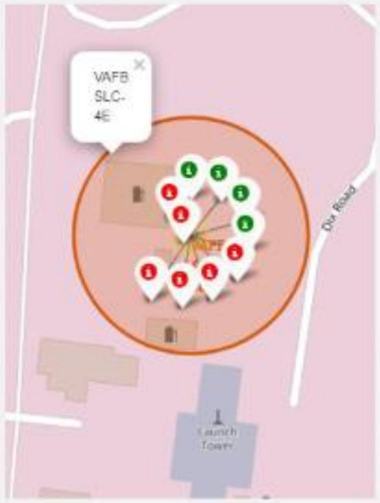






***Florida Launch Sites***

*Green Marker shows successful Launches and Red Marker shows Failures*



***California Launch Site***

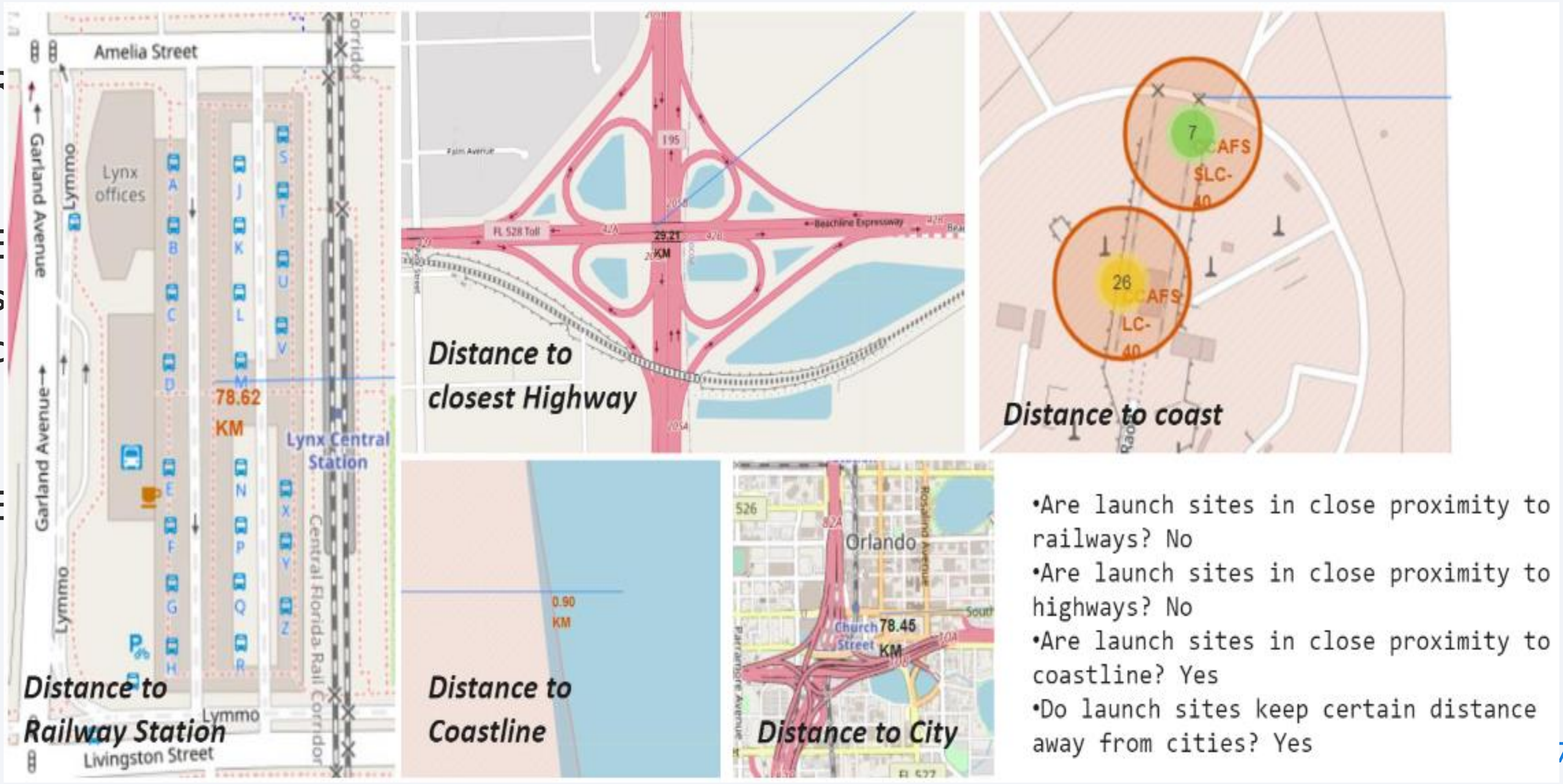
37

# Launch Site distance to landmarks

• F

• E
s
c

• E

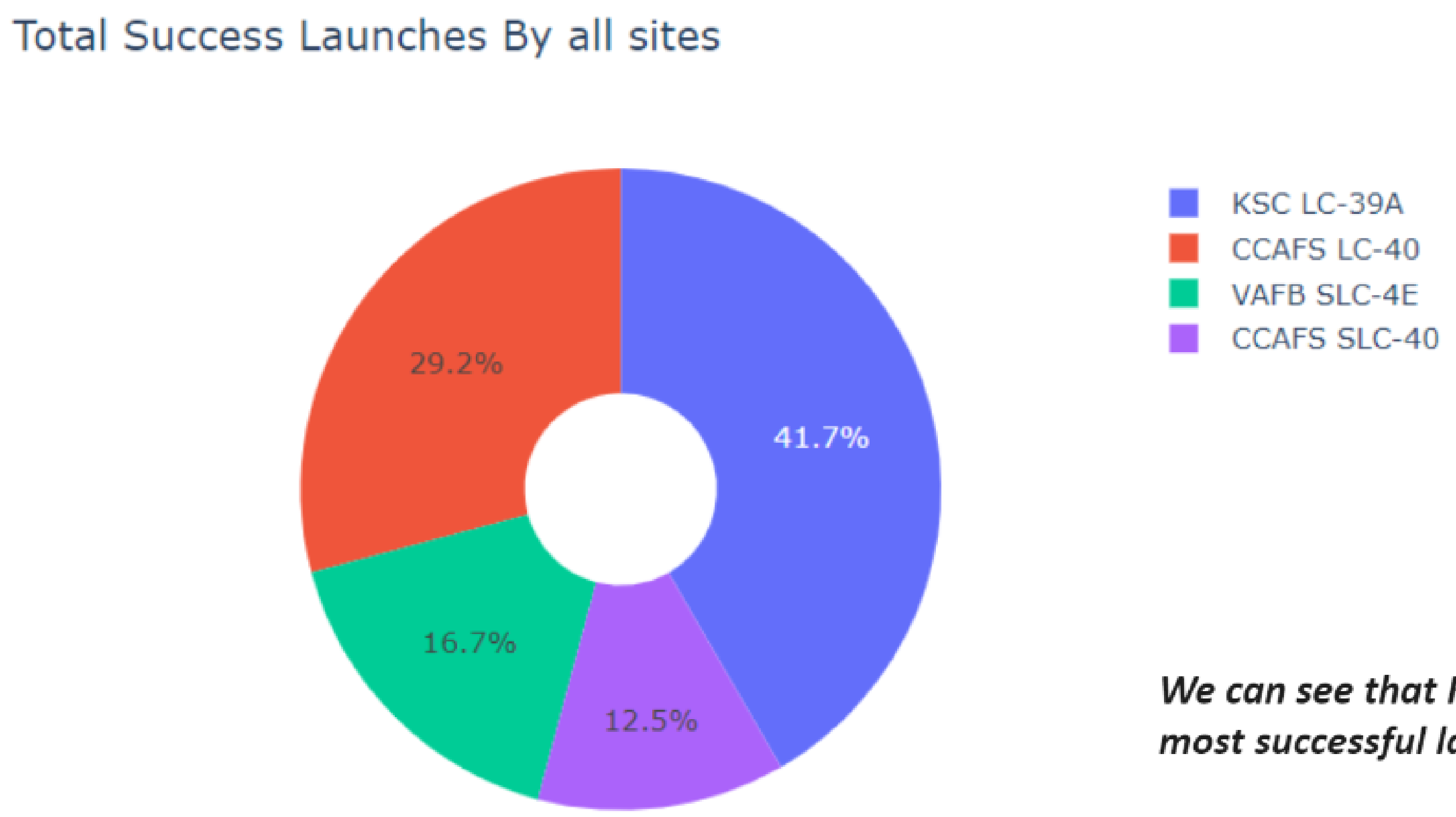


•Are launch sites in close proximity to railways? No
•Are launch sites in close proximity to highways? No
•Are launch sites in close proximity to coastline? Yes
•Do launch sites keep certain distance away from cities? Yes

Section 4

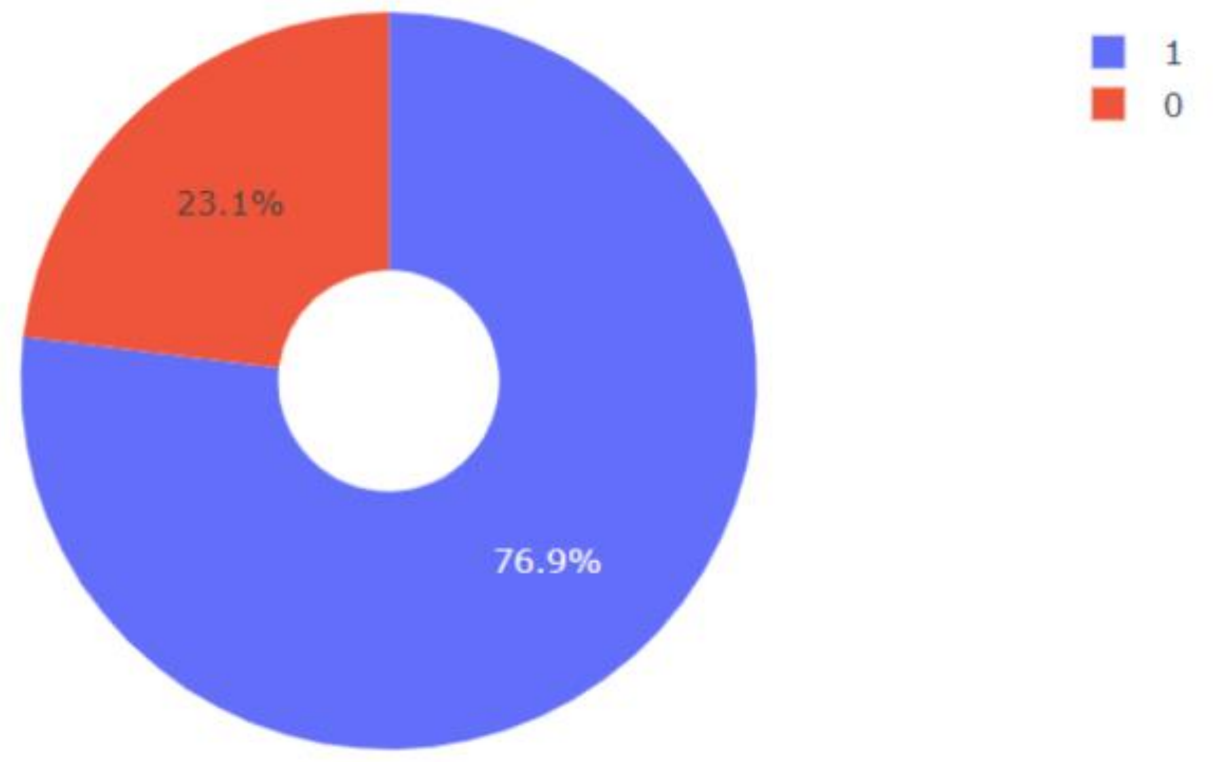
## Build a Dashboard with Plotly Dash

## Pie chart showing the success percentage achieved by each launch site



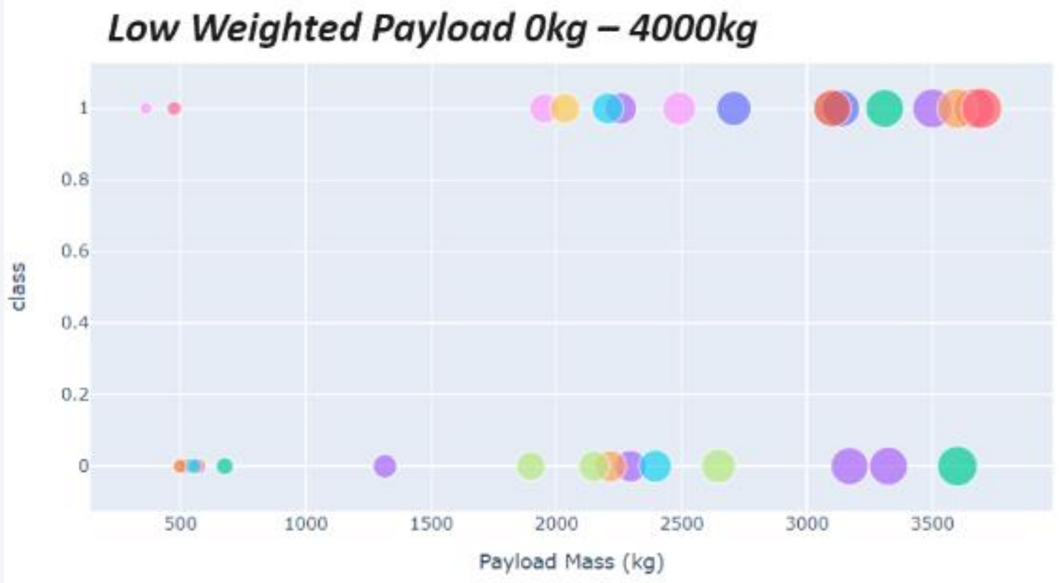
***We can see that KSC LC-39A had the most successful launches from all the sites***

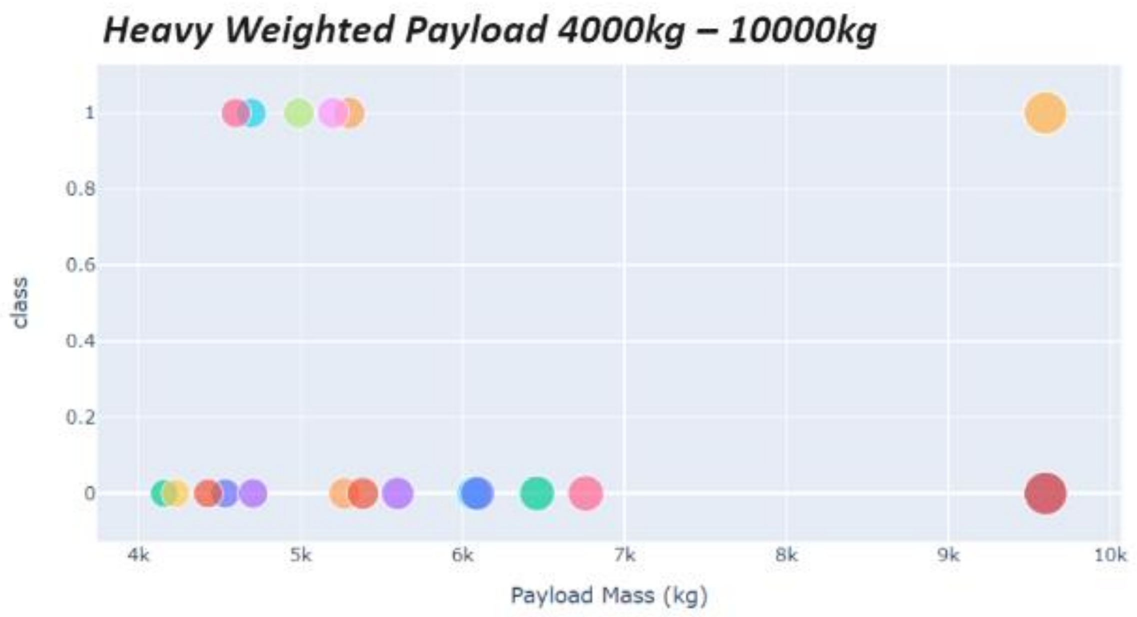
## Pie chart showing the Launch site with the highest launch success ratio



*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



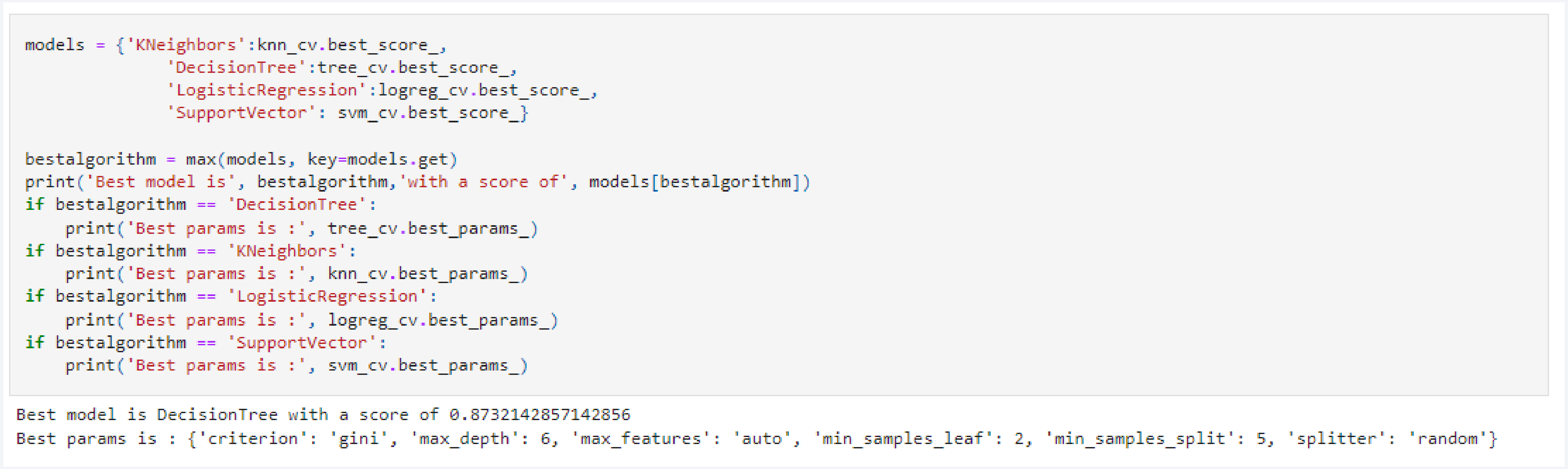


*We can see the success rates for low weighted payloads is higher than the heavy weighted payloads*

Section 5

## Predictive Analysis (Classification)

# Classification Accuracy

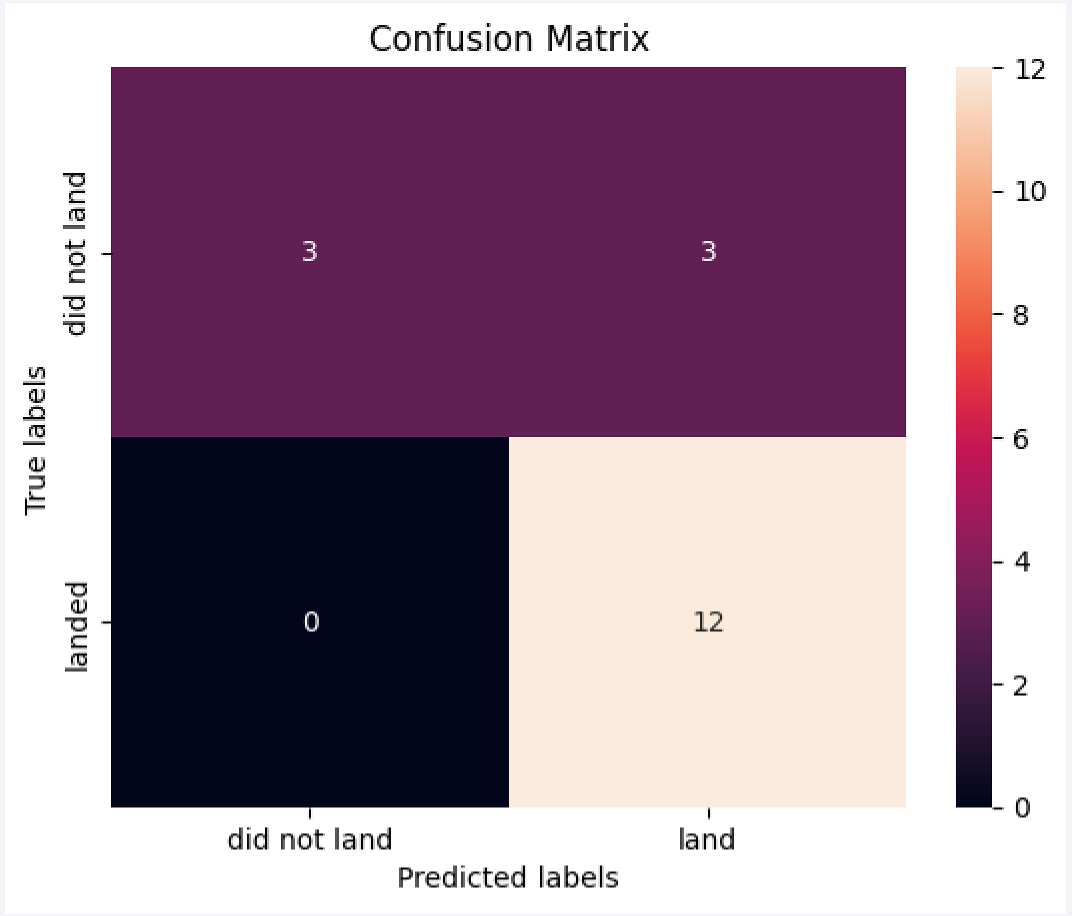


```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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 shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.



## Conclusions

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A had the most successful launches of any sites.
- 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!