IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Abhijeet Deepak Patil
22/02/2024

Outline

Executive Summary

- Summary of methodologies

  - Collection of Data using SpaceX API
  - Web scraping Falcon 9 and Falcon Heavy Launches Records from Wikipedia
  - Data Wrangling
  - Exploratory Data Analysis using SQL
  - Exploratory Data Analysis using Visualization
  - Launch Sites Locations Analysis with Folium
  - First Stage Landing Prediction using Machine Learning

- Summary of all results

  - Data Analysis Result
  - Prediction Result
  - Visuals

Introduction

- Project background and context

SpaceX 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 SpaceX 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 by the company to bid against SpaceX for a rocket launch. The goal of this project is to predict whether the first stage will land successfully or not.

- Problems you want to find answers

Determine whether the first will land successfully, what factors affect the landing and how they interact with each other.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using the SpaceX API and performing webscraping on wikipedia.
- Perform data wrangling
  - Data was cleaned, unwanted data was removed, nan values and categorical variables were taken care of.
- 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.

  - Data was collected using get request to SpaceX API.

  - Response was decoded using .json() and converted to a dataframe using .json_normalize()

  - Data was cleaned and data wrangling was performed.

  - Missing values were handled.

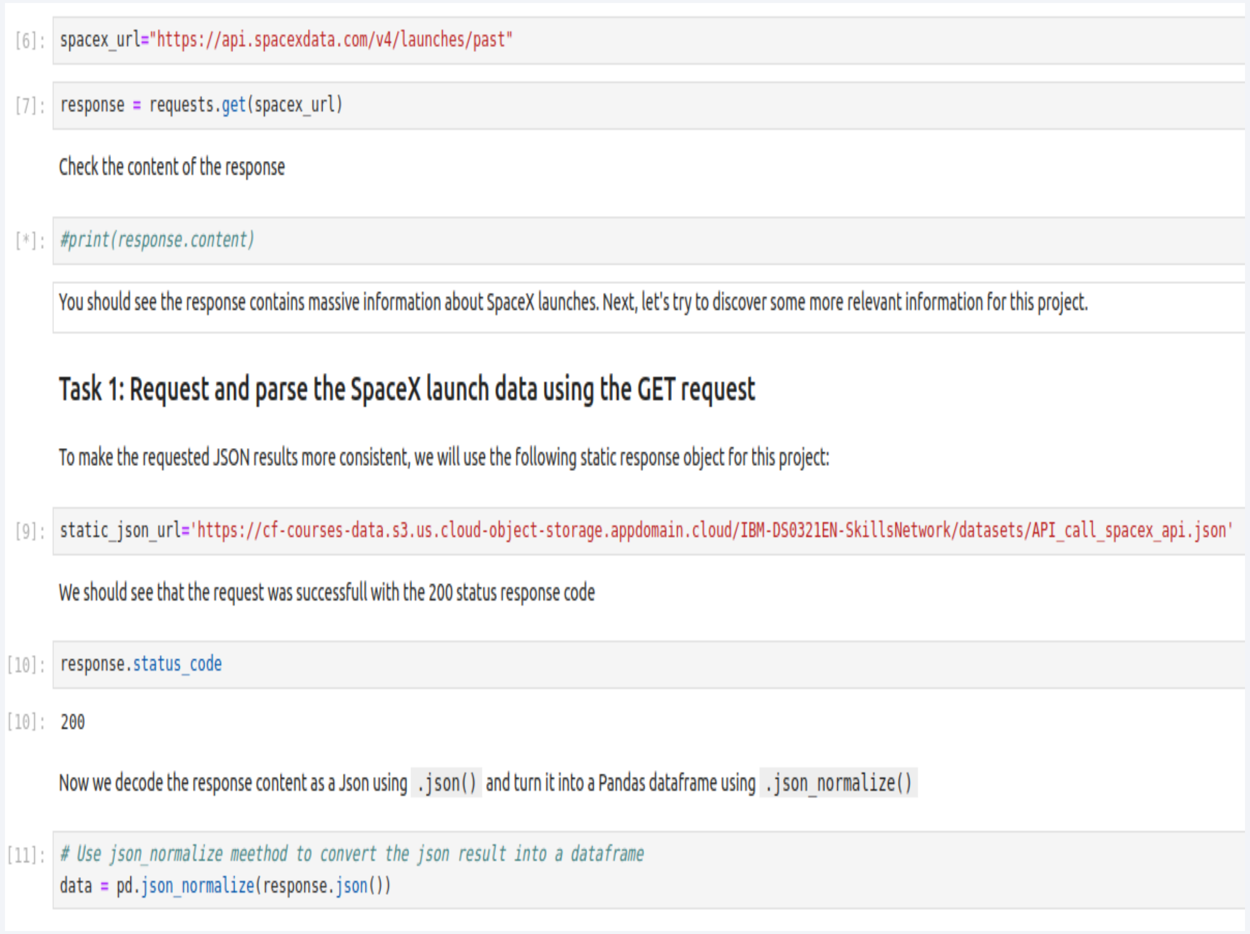
  - Additionally, webscrapping was performed on a wikipedia page using BeautifulSoup.

  - Falcon 9 data was extracted from HTML tables and converted to pandas dataframe.

Data Collection – SpaceX API

- We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.

- The link to the notebook is, https://github.com/abhiJeetP10/IBM-Applied-DS-Capstone/blob/main/Data%20Collection%20using%20API.ipynb .



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

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

Check the content of the response

```
[*]: #print(response.content)
```

You should see the response contains massive information about SpaceX launches. Next, let's try to discover some more relevant information for this project.

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

```
[9]: 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

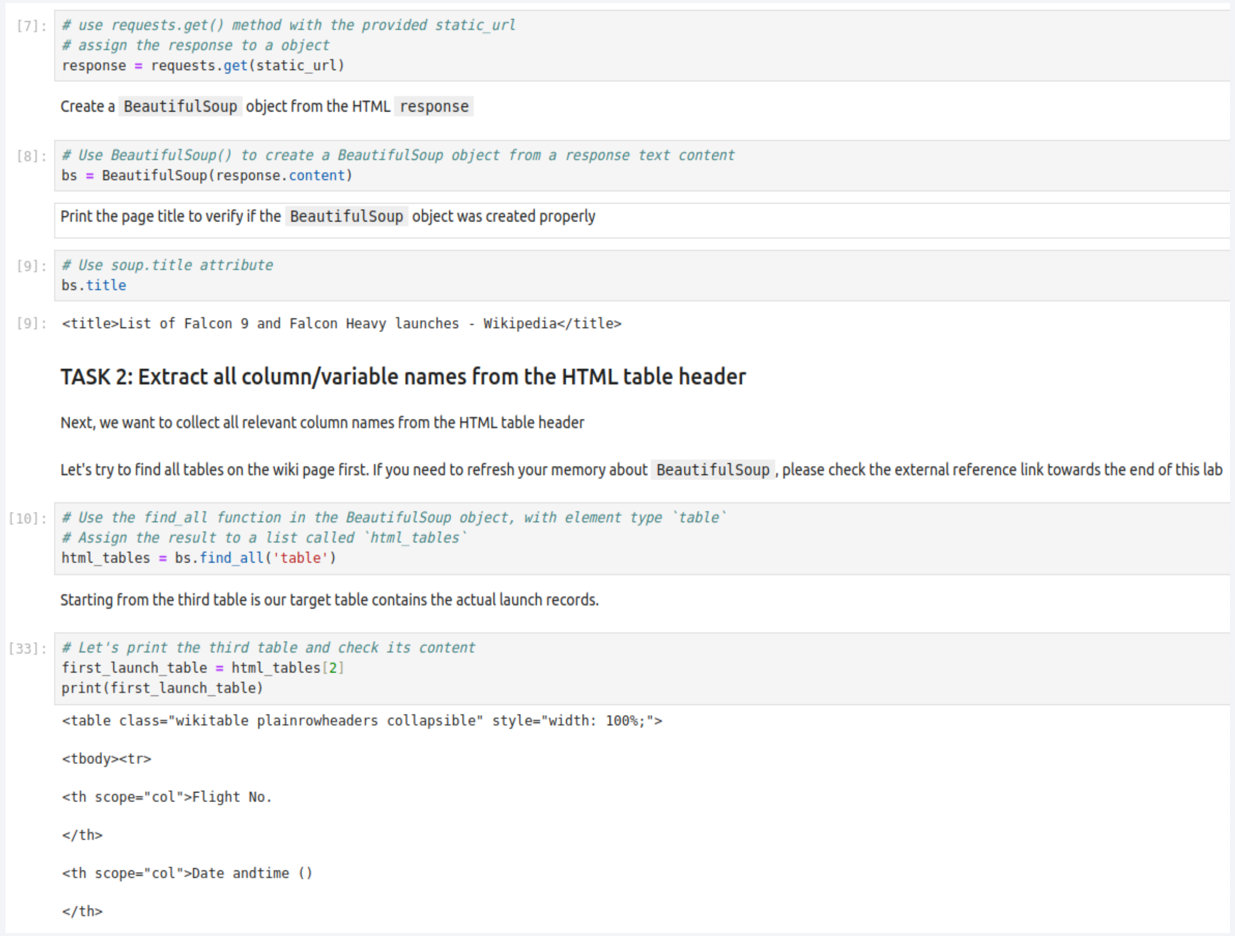
```
[10]: response.status_code

[10]: 200
```

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

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

Data Collection - Scraping

- We applied web scrapping to collect Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas dataframe.
- The link to the notebook is, https://github.com/abhiJeetP10/IBM-Applied-DS-Capstone/blob/main/Webscraping.ipynb .



```
[7]:  # 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`

```
[8]:  # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
      bs = BeautifulSoup(response.content)
```

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

```
[9]:  # Use soup.title attribute
      bs.title

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

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

```
[10]: # Use the find_all function in the BeautifulSoup object, with element type `table`
      # Assign the result to a list called `html_tables`
      html_tables = bs.find_all('table')
```

Starting from the third table is our target table contains the actual launch records.

```
[33]: # Let's print the third table and check its content
      first_launch_table = html_tables[2]
      print(first_launch_table)

      <table class="wikitable plainrowheaders collapsible" style="width: 100%;">

      <tbody><tr>

      <th scope="col">Flight No.

      </th>

      <th scope="col">Date andtime ()

      </th>
```

Data Wrangling

- We found the number of launches from each site.
- We also found the number of launches to each orbit type.
- We found the different type of outcomes.
- The successful outcomes were grouped together and so were the unsuccessful outcomes.
- A label variable was created, representing successful outcomes with 1 and unsuccessful outcomes with 0.
- The link to notebook is, https://github.com/abhiJeetP10/IBM-Applied-DS-Capstone/blob/main/Data%20Wrangling.ipynb .

EDA with Data Visualization

- Payload Mass vs Flight Number, Launch Site vs Flight Number, Launch Site vs Payload Mass, Success Rate vs Orbit Type, Orbit Type vs Flight Number, Orbit Type vs Payload Mass, Yearly Launch Success Trend were plotted.
- Categorical variables were handled using one-hot encoding.
- The link to the notebook is, https://github.com/abhiJeetP10/IBM-Applied-DS-Capstone/blob/main/EDA%20using%20Data%20Visualization.ipynb .

EDA with SQL

- The following sql queries were performed,

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

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

  - List the total number of successful and failure mission outcomes.

  - List the names of the booster_versions which have carried the maximum payload mass.

- The link to the notebook is, https://github.com/abhiJeetP10/IBM-Applied-DS-Capstone/blob/main/EDA%20with%20SQL.ipynb .

Build an Interactive Map with Folium

- We added markers to indicate the individual launch sites.
- Marker Clusters were used to represent the individual launches since launches from same site are in close proximity to each other.
- Polylines to closest roads, railway tracks, coasts and cities were shown with distances.
- This was done to understand the importance of the location of a launch site.
- The link to the notebook is, https://github.com/abhiJeetP10/IBM-Applied-DS-Capstone/blob/main/Site%20Locations%20using%20Folium.ipynb .

Build a Dashboard with Plotly Dash

- A dropdown that allows us to select all or individual sites.
- An interactive pie chart is added to the dashboard, displaying success and failure percentages.
- A slider allowing us to select the payload mass range.
- A scatter plot showing distribution of success and failure based on payload mass(in kg).
- The link to the dash app is, https://github.com/abhiJeetP10/IBM-Applied-DS-Capstone/tree/main/Dashboard .

Predictive Analysis (Classification)

- First, the target variable and the independent variables were separated.
- Next, we processed the data and standardized it.
- The data was split in training and testing data.
- GridSearchCV was used to find the best hyperparameters.
- Logistic Regression, Support Vector Machine, Decision Tree and K-nearest Neighbours algorithms were tested and confusion matrix was plotted for each.
- The best algorithm was found to be Decision Trees.
- The link to the notebook is, https://github.com/abhiJeetP10/IBM-Applied-DS-Capstone/blob/main/Machine%20Learning%20Prediction.ipynb .

Results

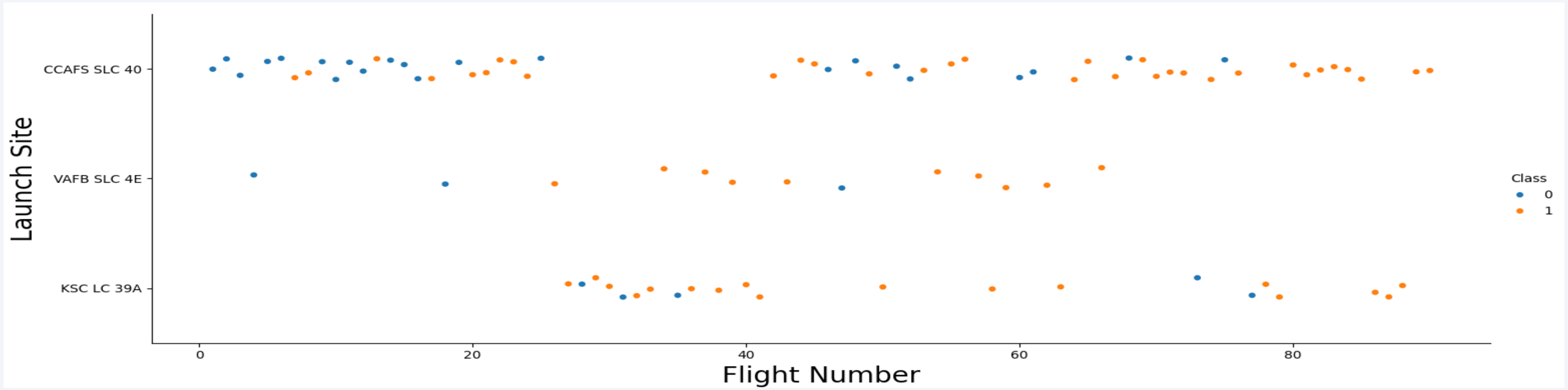
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

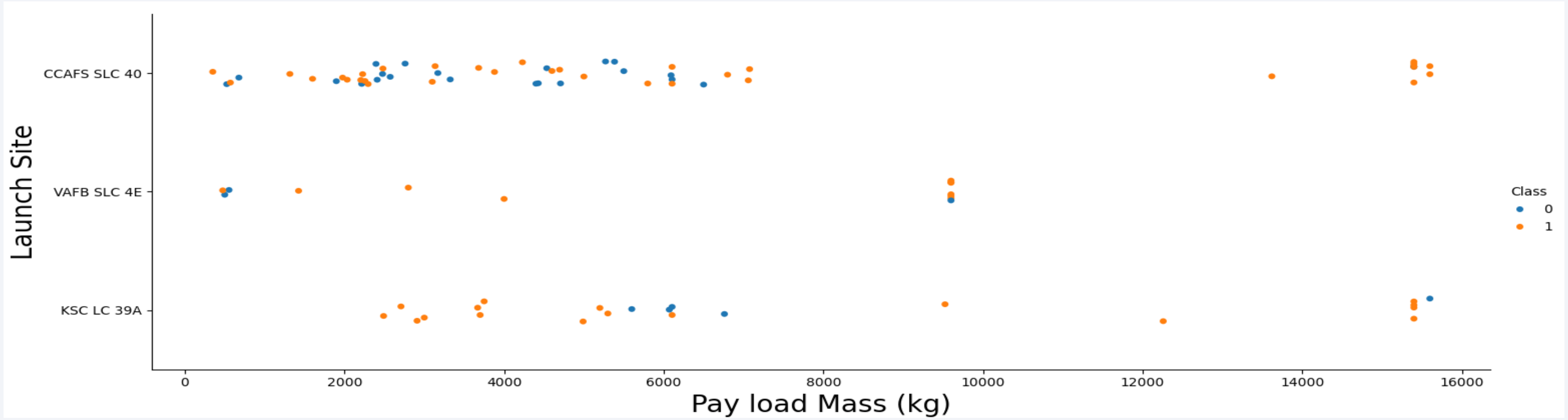
Flight Number vs. Launch Site

- We see that different launch sites have different success rates. CCAFS SLC-40, has a success rate of 60 %, while KSC LC-39A and VAFB SLC 4E has a success rate of 77%. We also see that as the flight number increases, the first stage is more likely to land successfully.



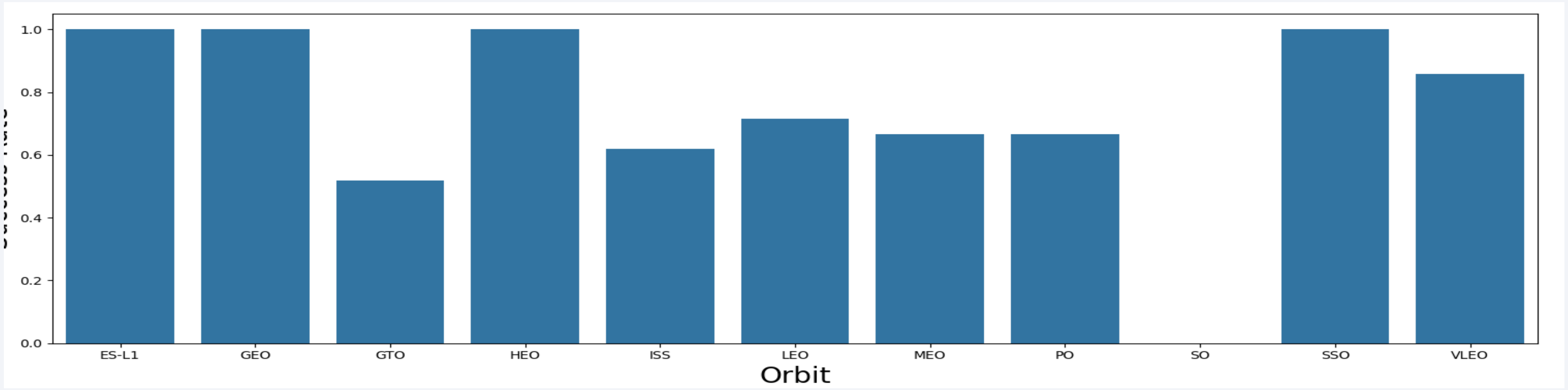
Payload vs. Launch Site

- We can see that CCAFS SLC-40 has higher success rate for high payload mass than for low payload mass.
- It is also visible that no launches with payload mass greater than 10000 kg took place from VAFB SLC-4E.
- We can also see that KSC LC-39A has good success rate across all payload ranges.



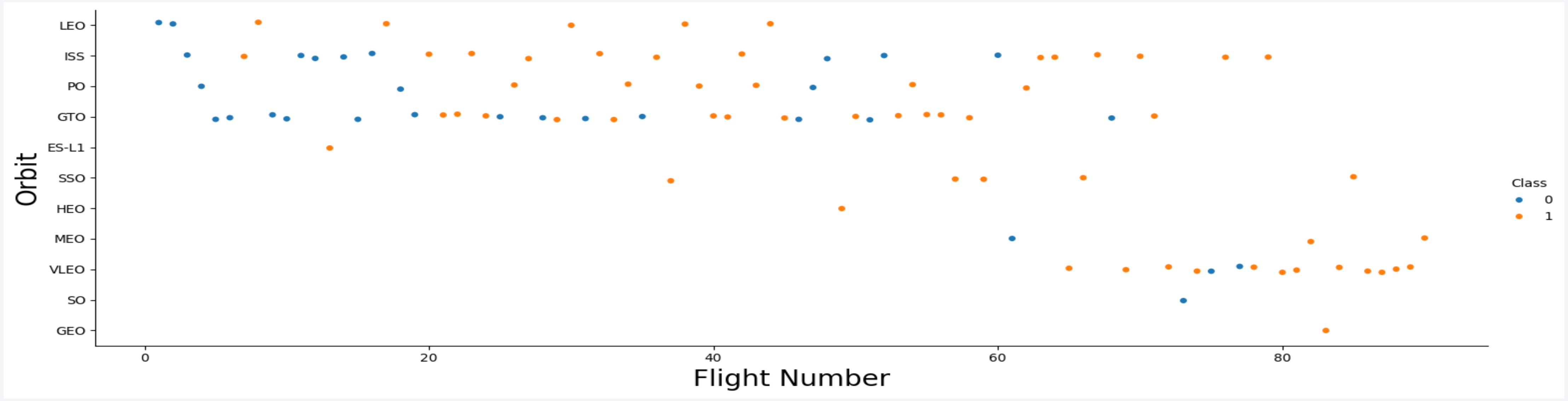
Success Rate vs. Orbit Type

- We can see ES-L1, GEO, HEO, SSO had the highest success rate.



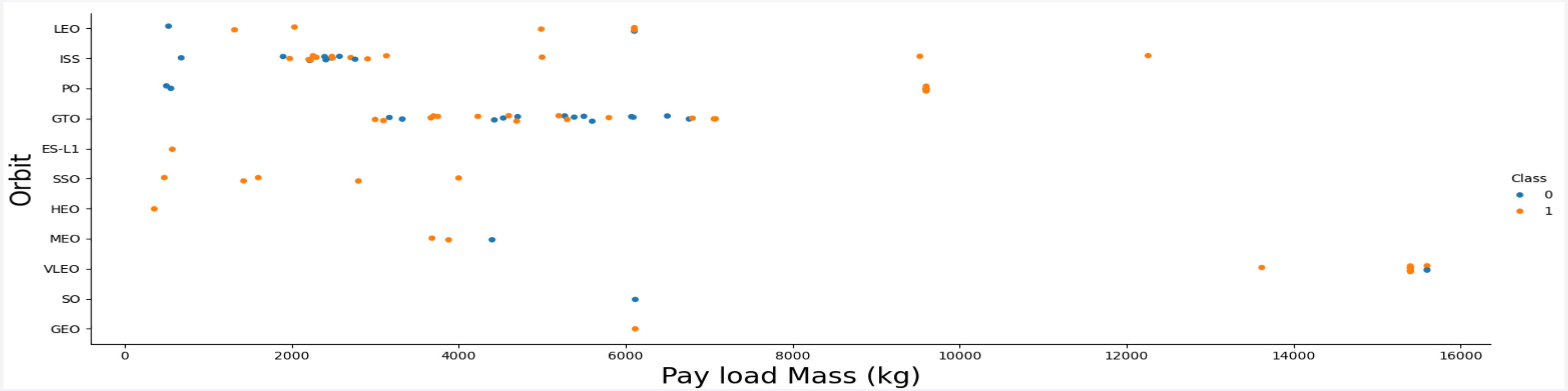
Flight Number vs. Orbit Type

- We can see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.



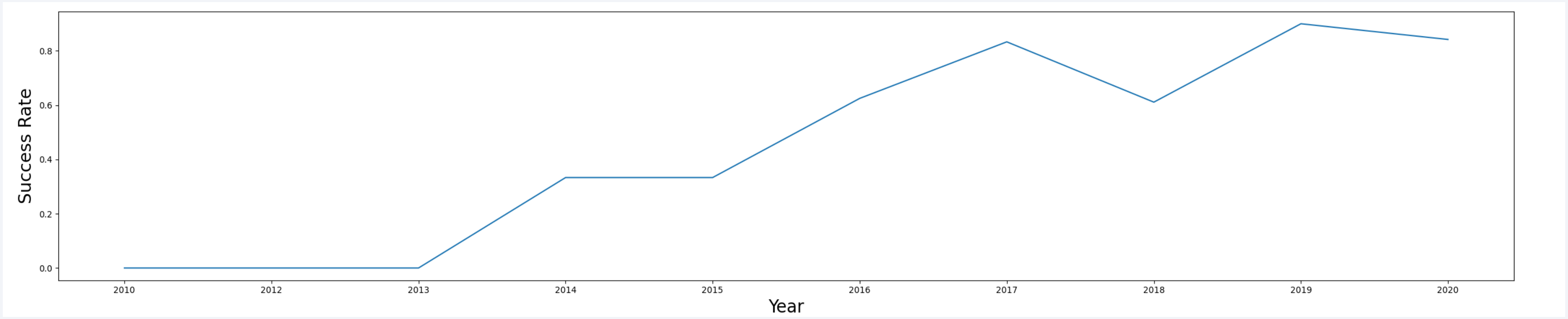
Payload vs. Orbit Type

- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.
- However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.



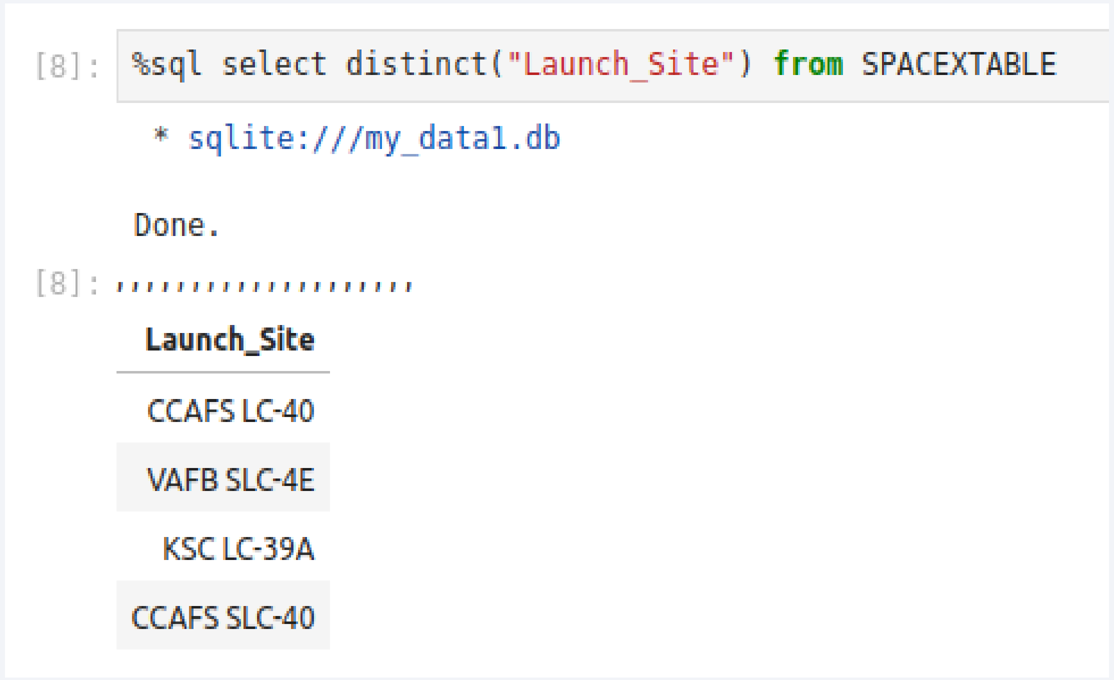
Launch Success Yearly Trend

- We can observe that the success rate since 2013 kept increasing till 2020



All Launch Site Names

- We used the key word DISTINCT to show only unique launch sites from the SpaceX data.



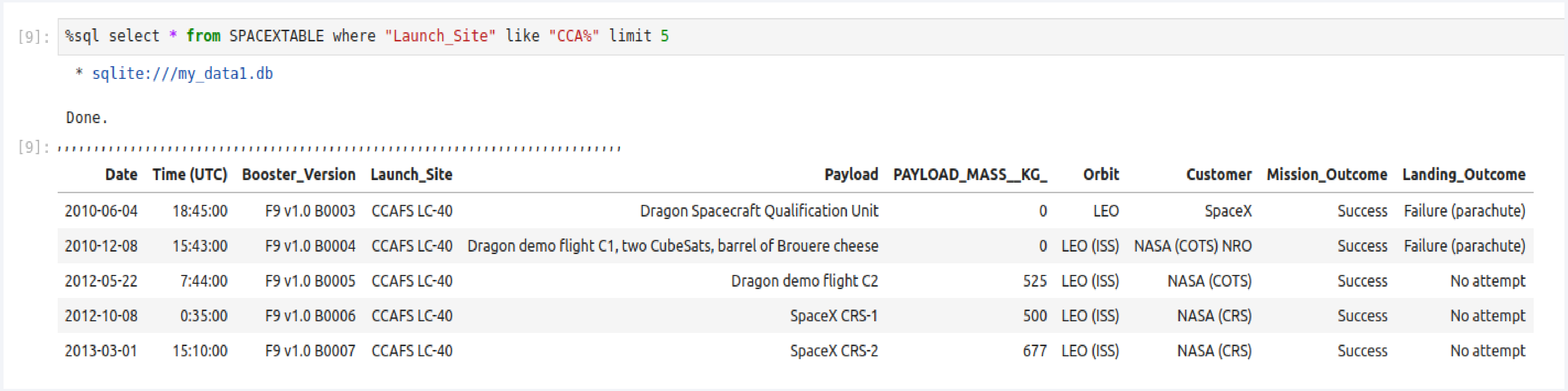
```
[8]: %sql select distinct("Launch_Site") from SPACEXTABLE

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

[8]:

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

Launch Site Names Begin with 'CCA'

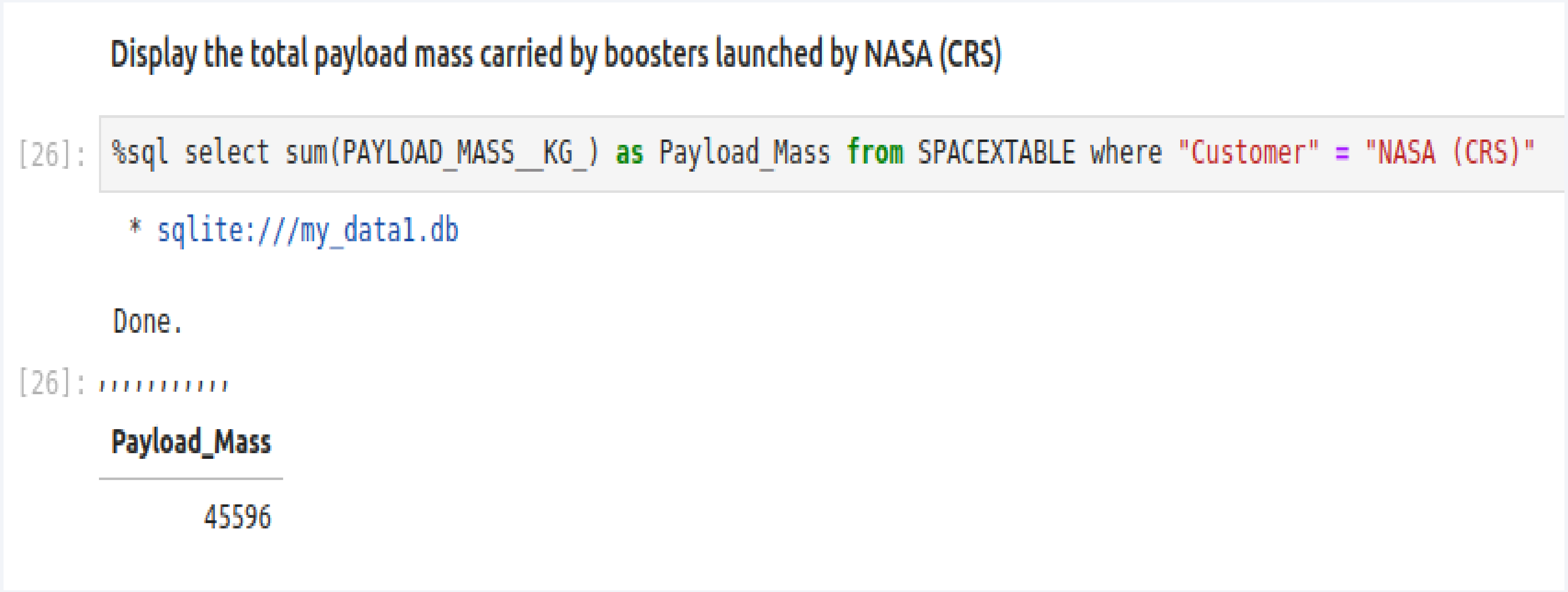
- We use like 'CCA%' to find names beginning with 'CCA'
- We use limit 5 to limit the number of queries to 5.



```
[9]: %sql select * from SPACEXTABLE where "Launch_Site" like "CCA%" limit 5

      * sqlite:///my_data1.db

     Done.
[9]:
```

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

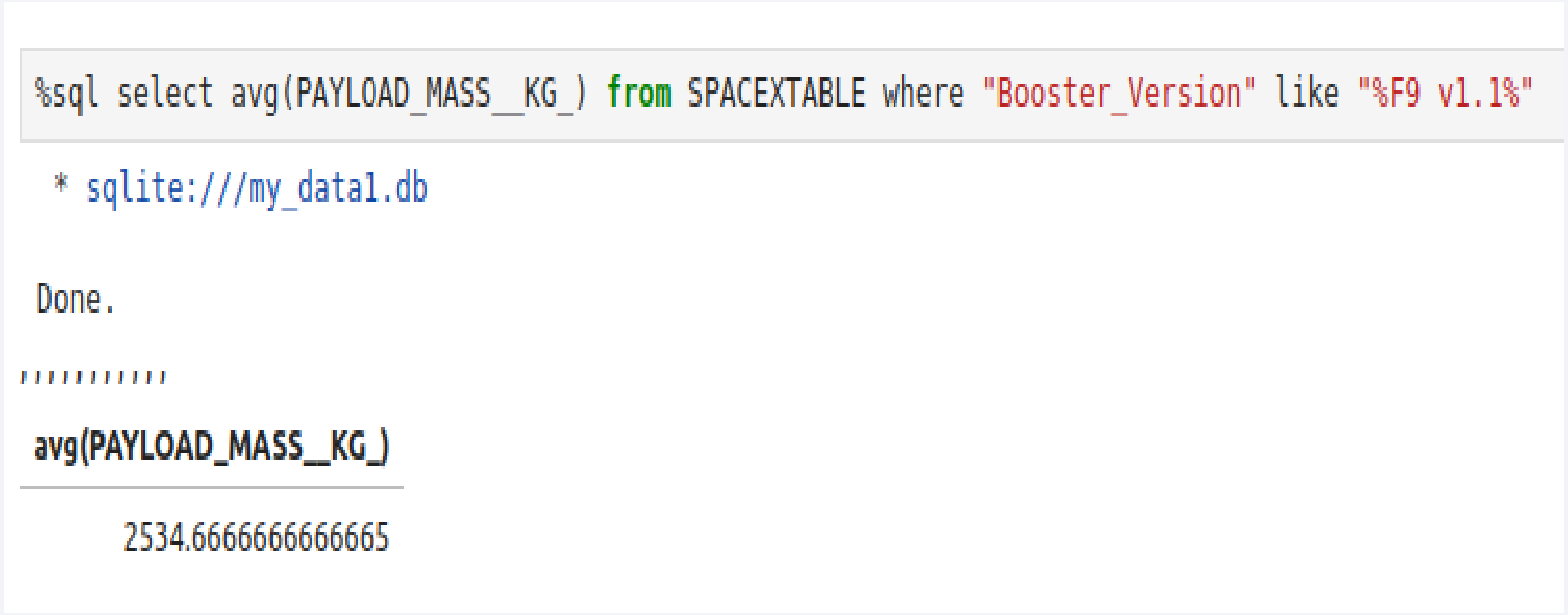


Display the total payload mass carried by boosters launched by NASA (CRS)

```
[26]: %sql select sum(PAYLOAD_MASS__KG_) as Payload_Mass from SPACEXTABLE where "Customer" = "NASA (CRS)"

 * sqlite:///my_data1.db

Done.

[26]: ''''''''''''
```

Payload_Mass
45596

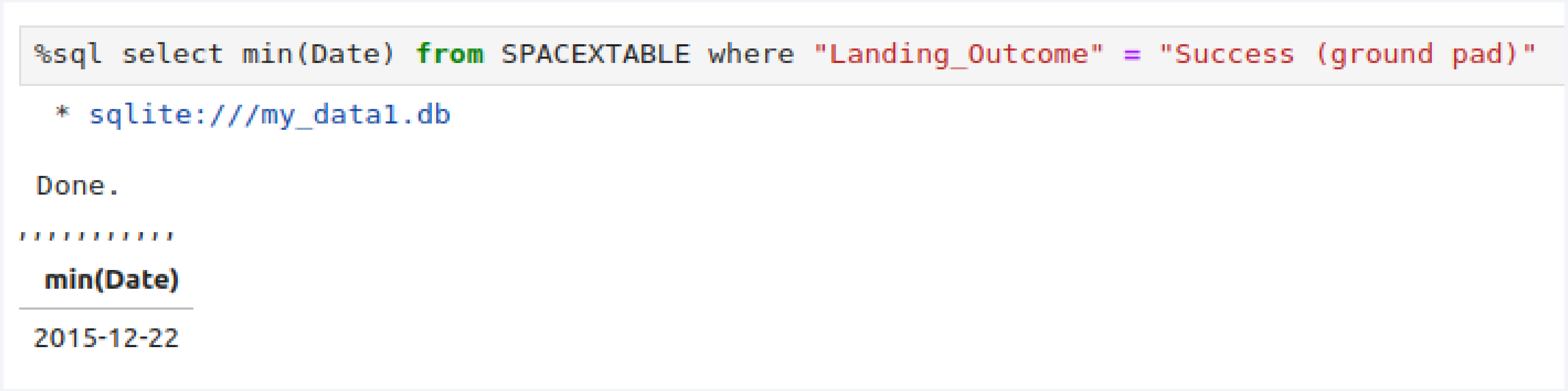
Average Payload Mass by F9 v1.1



```
%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTABLE where "Booster_Version" like "%F9 v1.1%"

 * sqlite:///my_data1.db

Done.

avg(PAYLOAD_MASS__KG_)
      2534.6666666666665
```

First Successful Ground Landing Date

- We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015

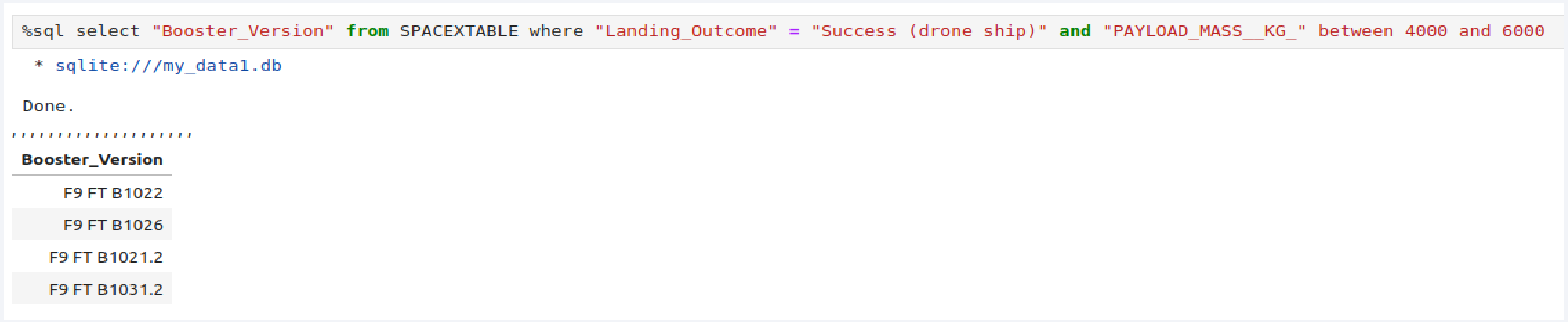


```
%sql select min(Date) from SPACEXTABLE where "Landing_Outcome" = "Success (ground pad)"

 * sqlite:///my_data1.db

Done.
```

min(Date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- 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.

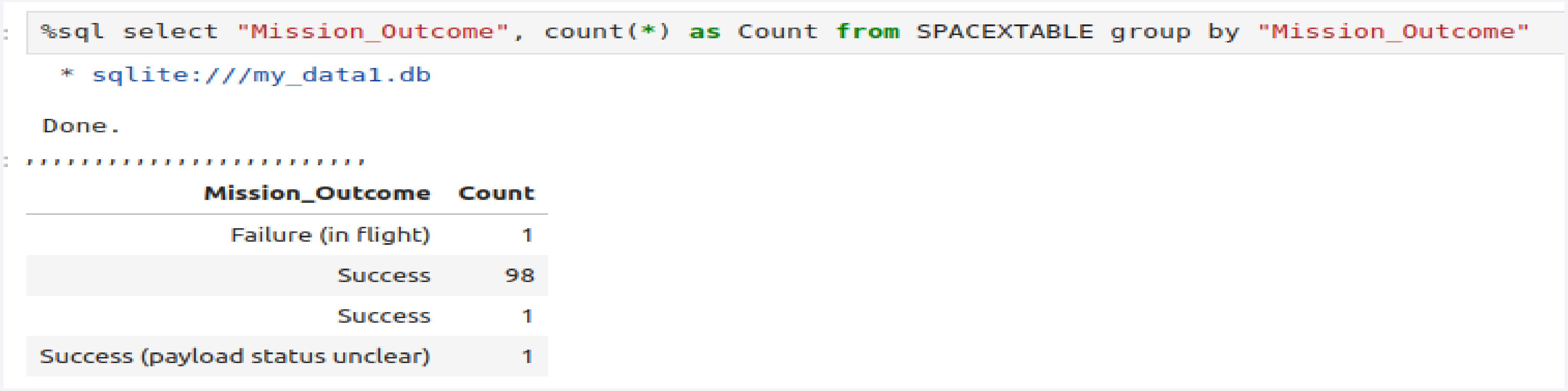


```
%sql select "Booster_Version" from SPACEXTABLE where "Landing_Outcome" = "Success (drone ship)" and "PAYLOAD_MASS__KG_" between 4000 and 6000
 * sqlite:///my_data1.db
Done.
```

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

Total Number of Successful and Failure Mission Outcomes

- We group the data by Mission Outcome.

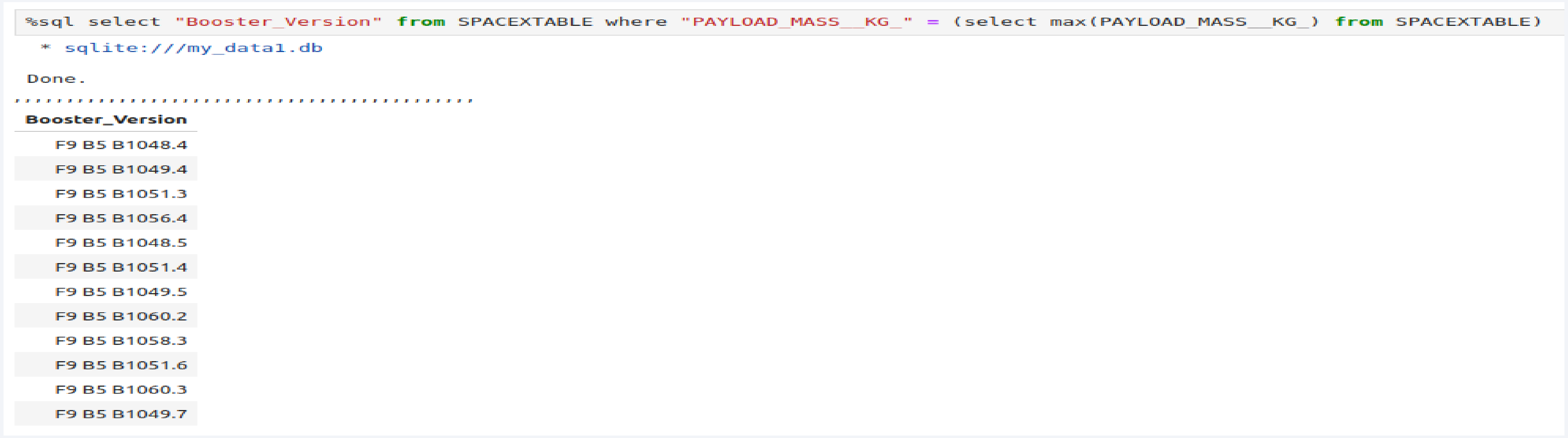


```
%sql select "Mission_Outcome", count(*) as Count from SPACEXTABLE group by "Mission_Outcome"
 * sqlite:///my_data1.db
Done.
```

Mission_Outcome	Count
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- We have used a subquery in the where clause and used the max function in the subquery to find the maximum payload.

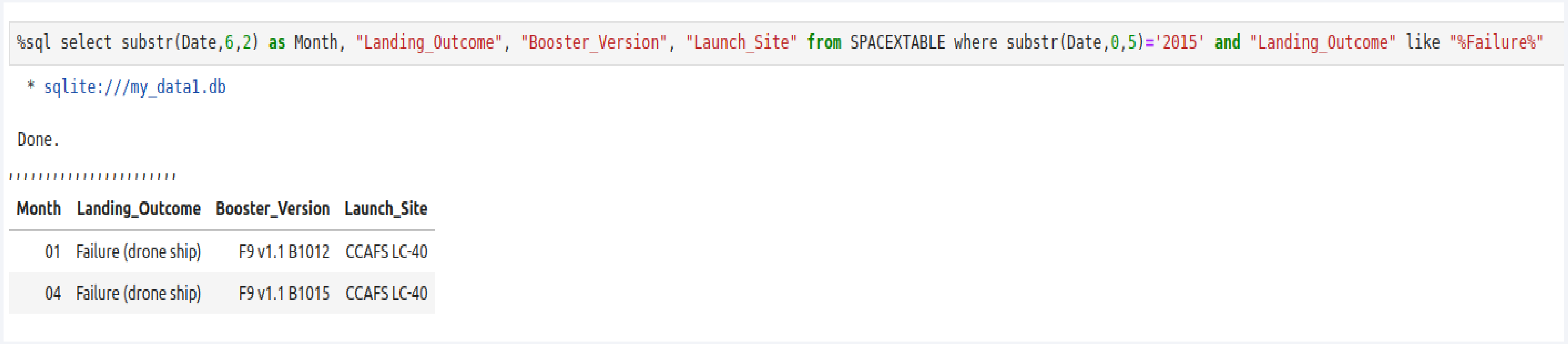


```
%sql select "Booster_Version" from SPACEXTABLE where "PAYLOAD_MASS__KG_" = (select max(PAYLOAD_MASS__KG_) from SPACEXTABLE)
 * sqlite:///my_data1.db
Done.
```

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

2015 Launch Records

- We have used the substr() function to get the month and year.

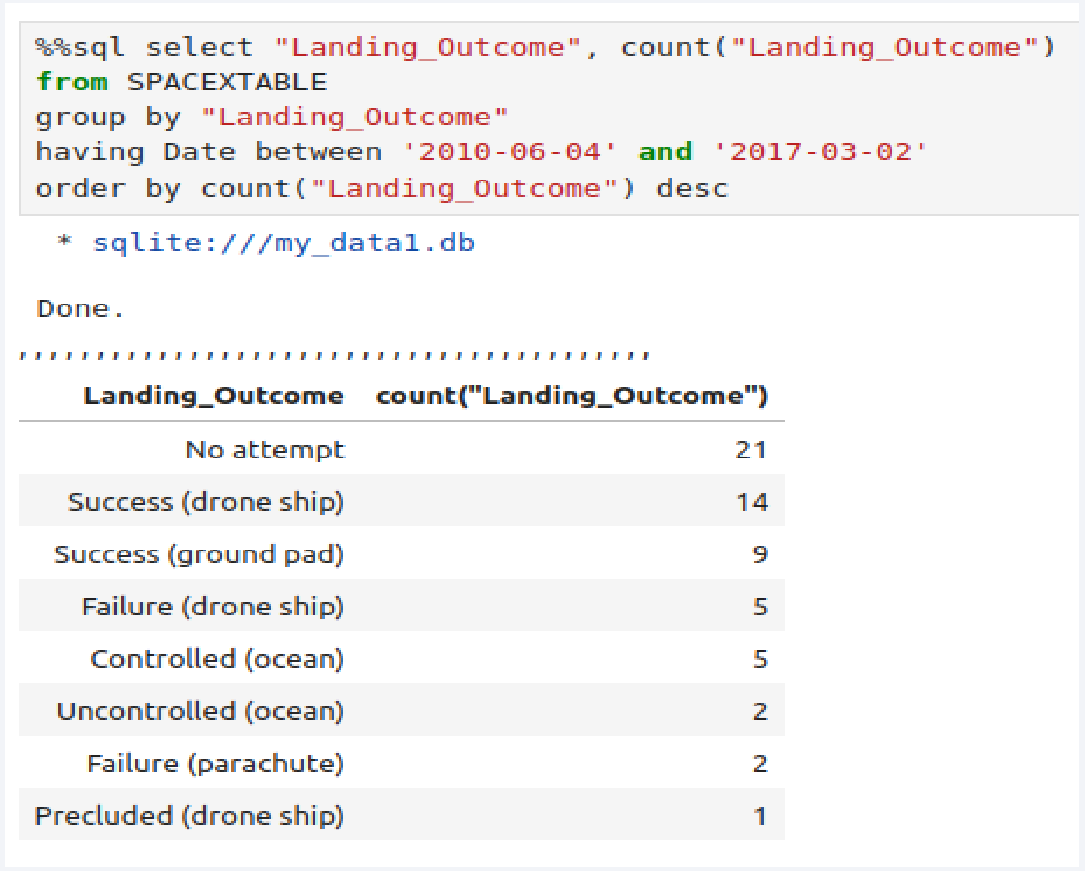


```
%sql select substr(Date,6,2) as Month, "Landing_Outcome", "Booster_Version", "Launch_Site" from SPACEXTABLE where substr(Date,0,5)='2015' and "Landing_Outcome" like "%Failure%"

 * sqlite:///my_data1.db

Done.
```

Month	Landing_Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

- We selected Landing outcomes and the COUNT of landing outcomes from the data and used the HAVING to filter for landing outcomes BETWEEN 2010-06-04 to 2017-03-20.
- We applied the GROUP BY clause to group the landing outcomes and the ORDER BY clause to order the grouped landing outcome in descending order.



```
%%sql select "Landing_Outcome", count("Landing_Outcome")
from SPACEXTABLE
group by "Landing_Outcome"
having Date between '2010-06-04' and '2017-03-02'
order by count("Landing_Outcome") desc
```

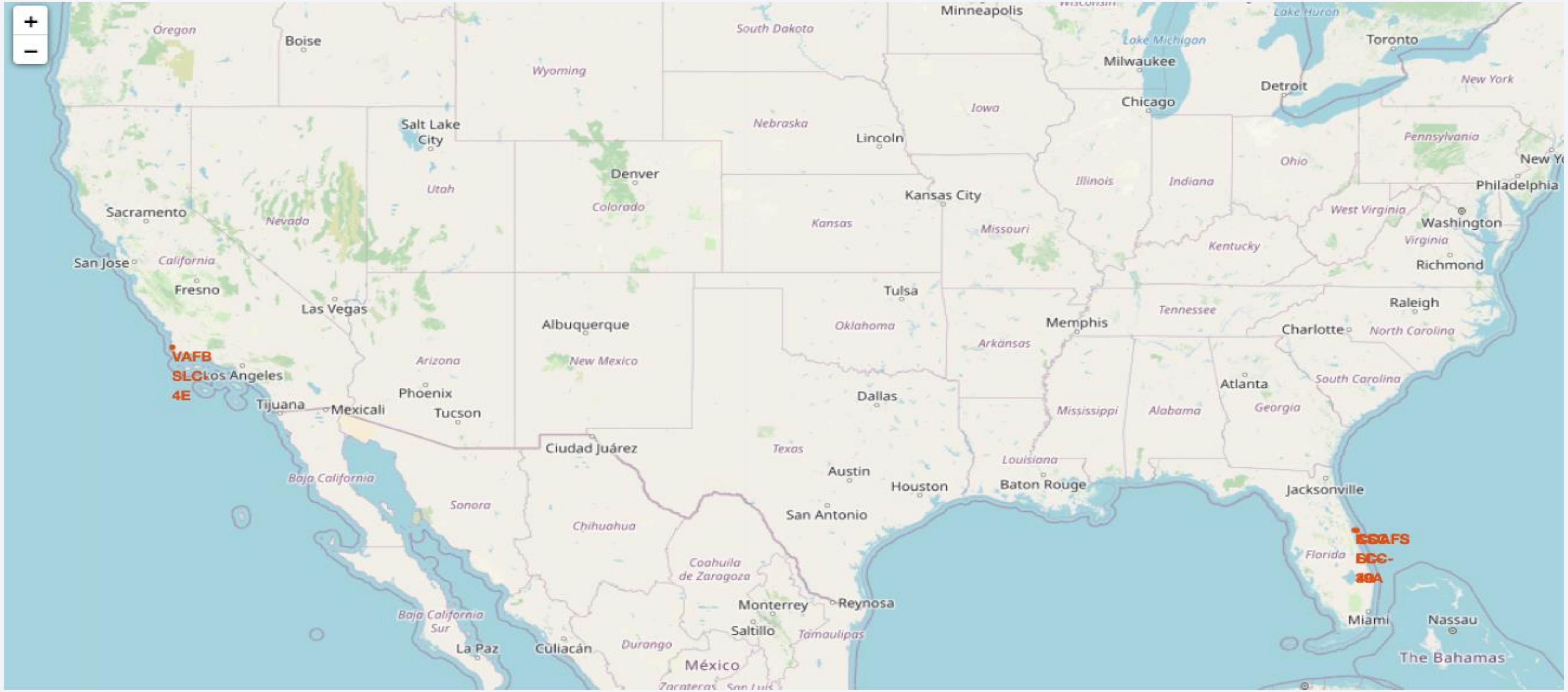
```
 * sqlite:///my_data1.db

Done.
```

Landing_Outcome	count("Landing_Outcome")
No attempt	21
Success (drone ship)	14
Success (ground pad)	9
Failure (drone ship)	5
Controlled (ocean)	5
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

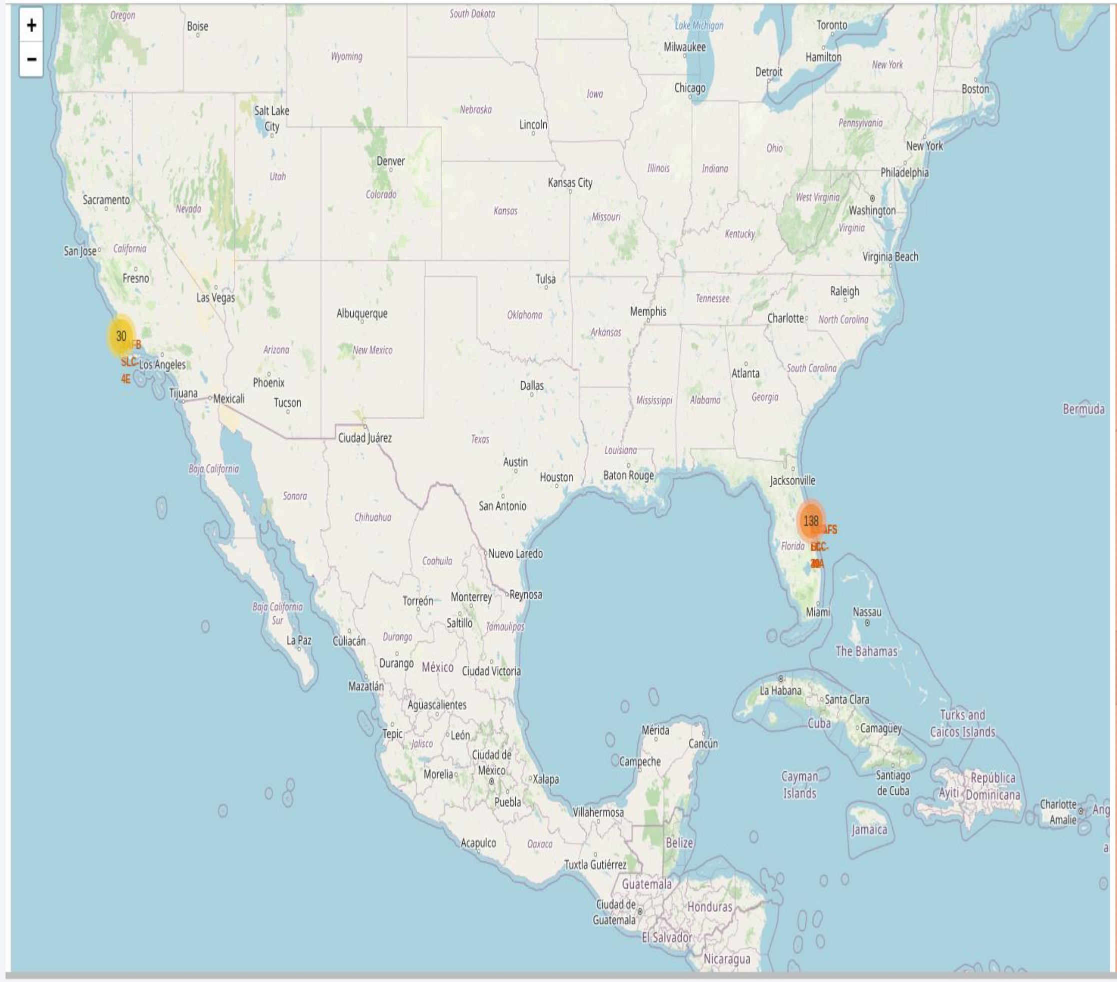
Section 3

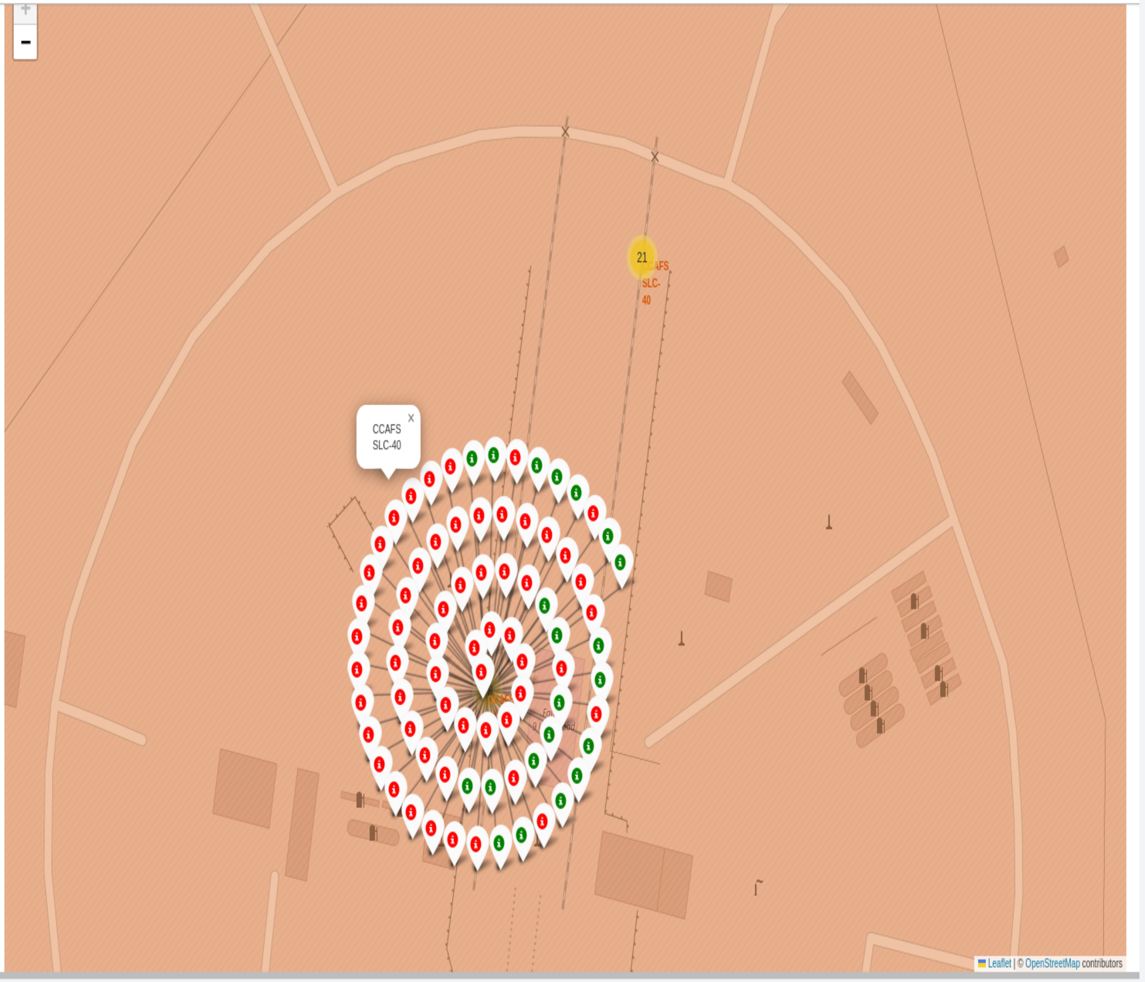
Launch Sites Proximities Analysis

All Launch Sites Map Markers

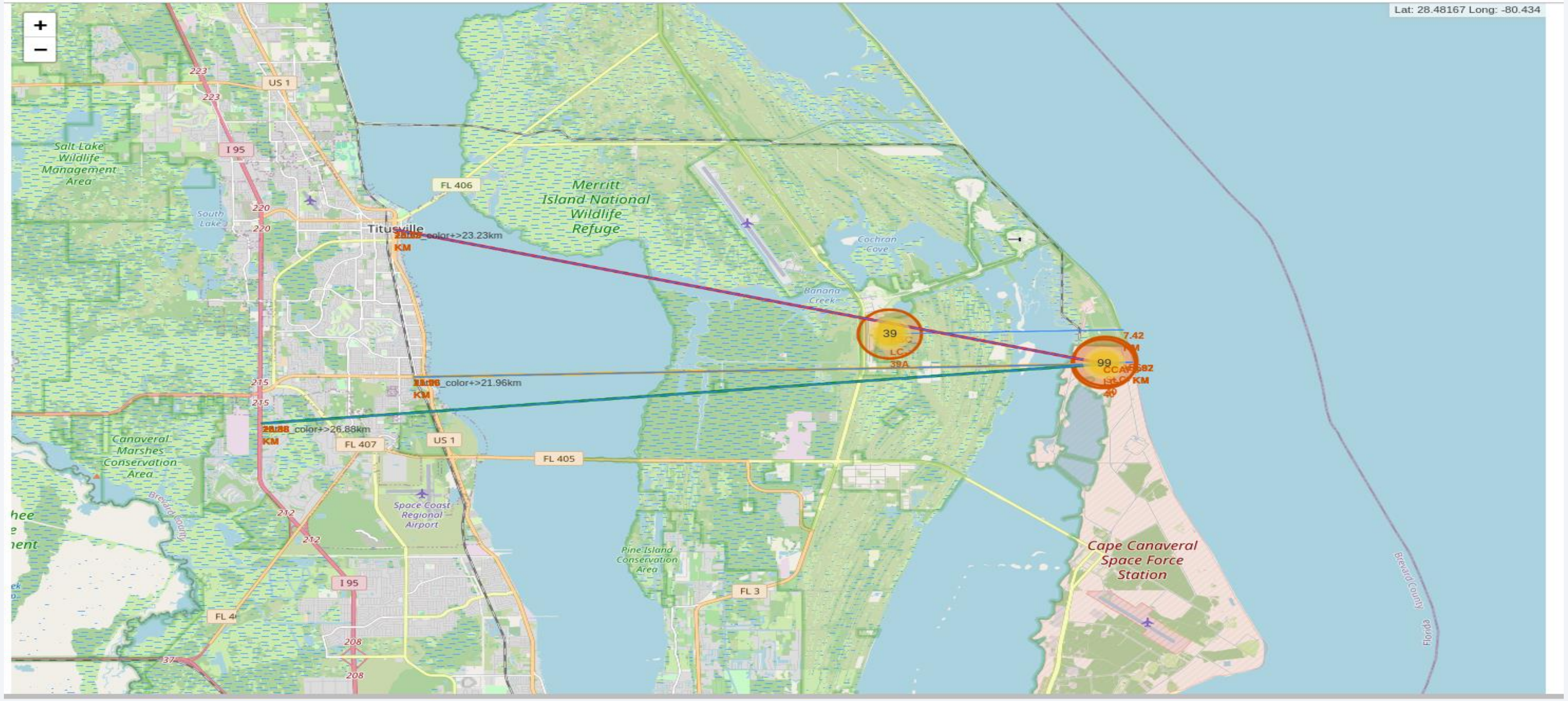


Marked the success/failed launches for each site on the map

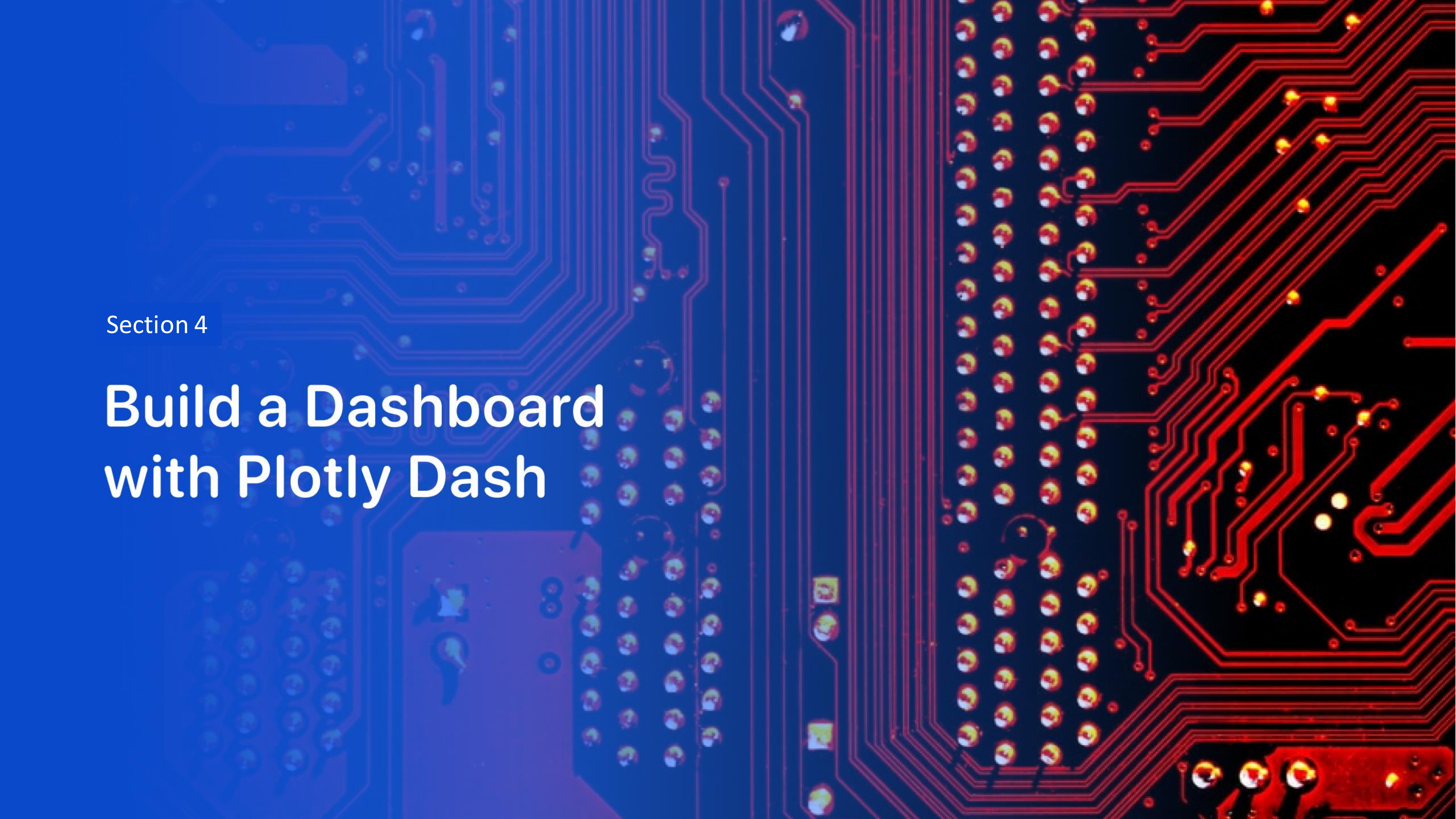




Launch Site distance to Landmarks



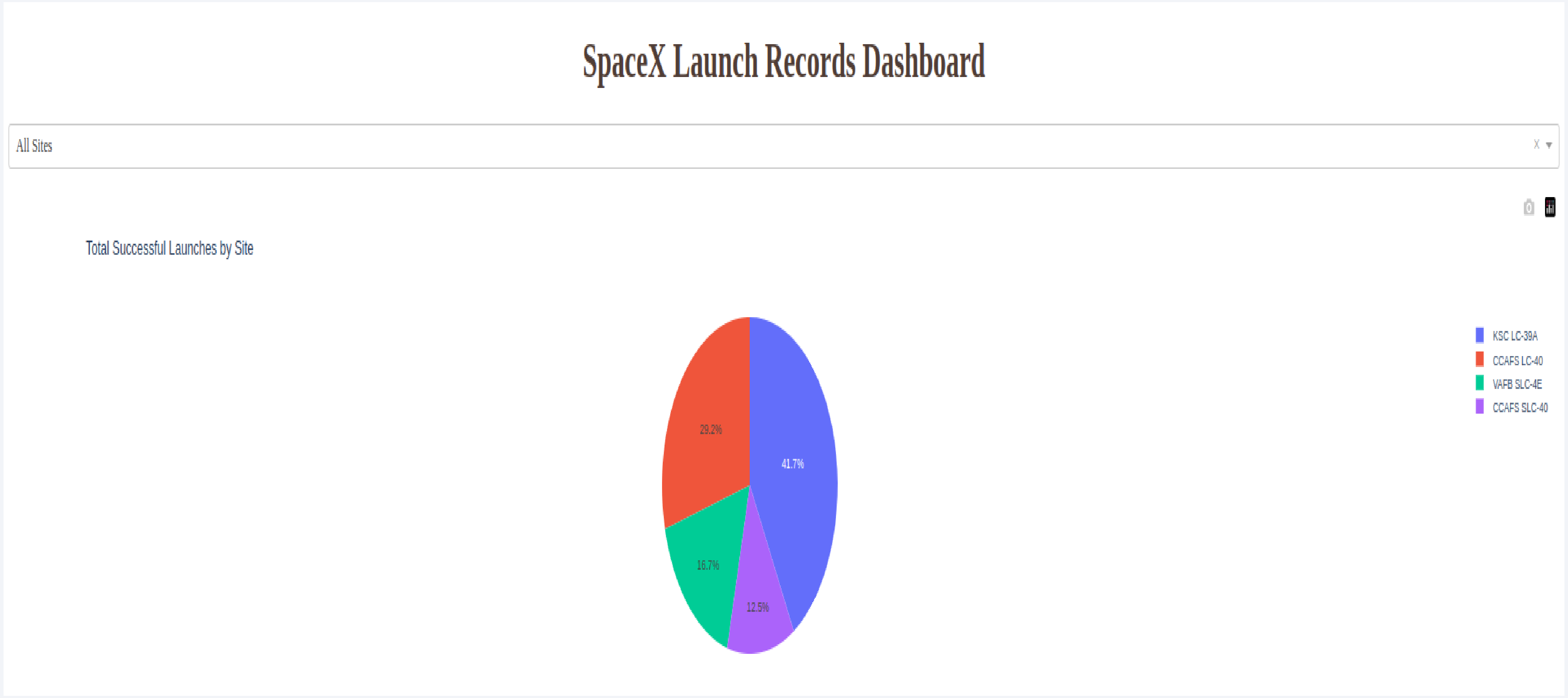
Lat: 28.48167 Long: -80.434
Titusville
color+>23.23km
color+>21.96km
color+>26.88km
7.42 KM
39
99
LC-39A
CCAFS
Merritt Island National Wildlife Refuge
Salt Lake Wildlife Management Area
Canaveral Marshes Conservation Area
Space Coast Regional Airport
Pine Island Conservation Area
Cape Canaveral Space Force Station
Banana Creek
Cochran Cove
South Lake
US 1
I 95
FL 406
FL 407
FL 405
FL 3



Section 4

Build a Dashboard with Plotly Dash

Pie Chart showing percentage of Total successful landings, achieved by each site

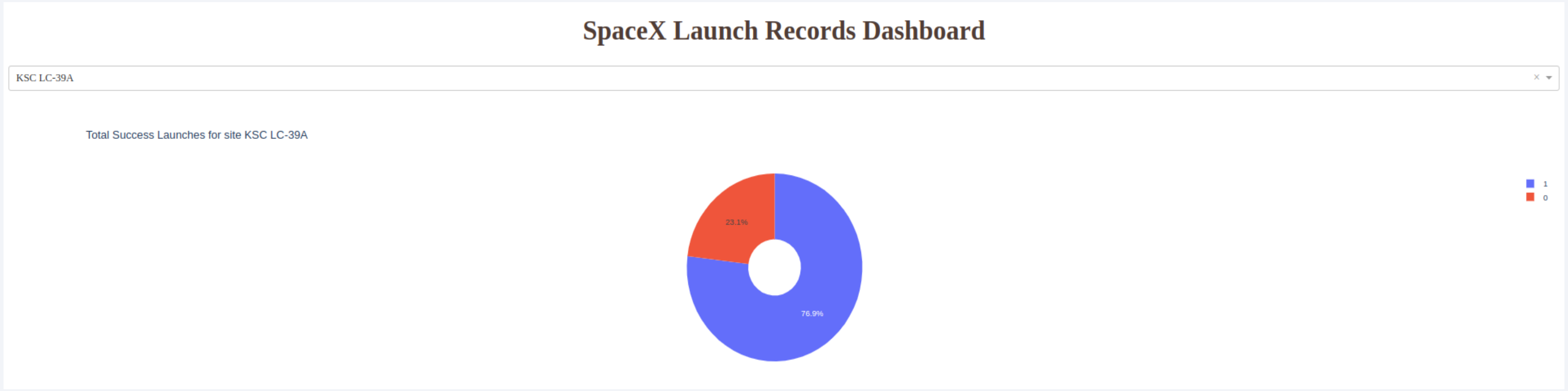

SpaceX Launch Records Dashboard

All Sites

Total Successful Launches by Site

Pie Chart showing outcome distribution for site with highest success rate

- KSC LC-39A achieved a 76.9% success rate and a 23.1% failure rate.

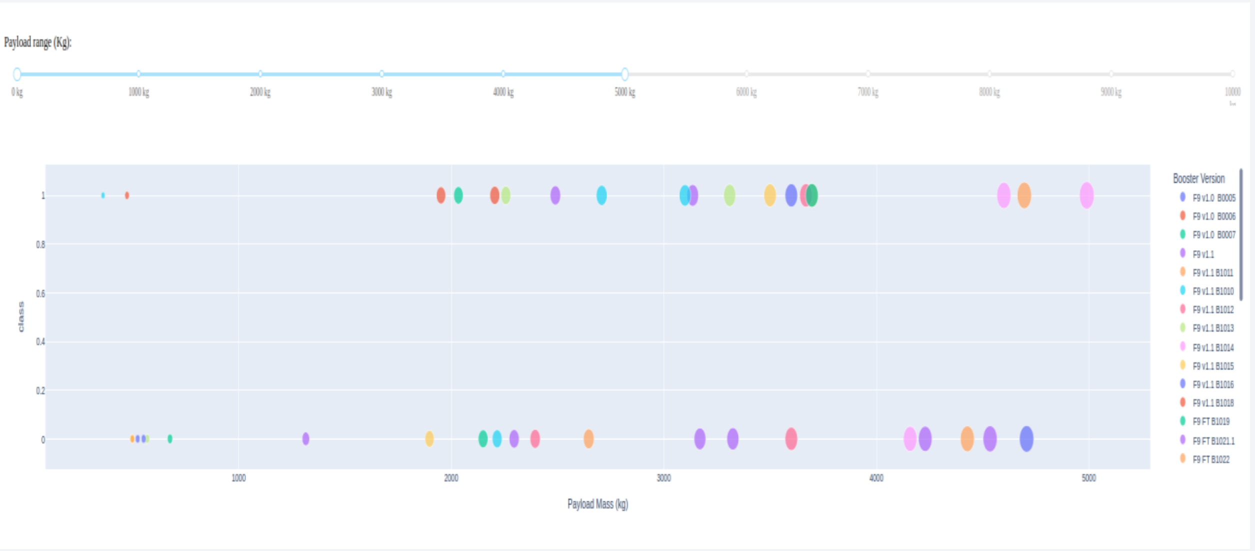

SpaceX Launch Records Dashboard

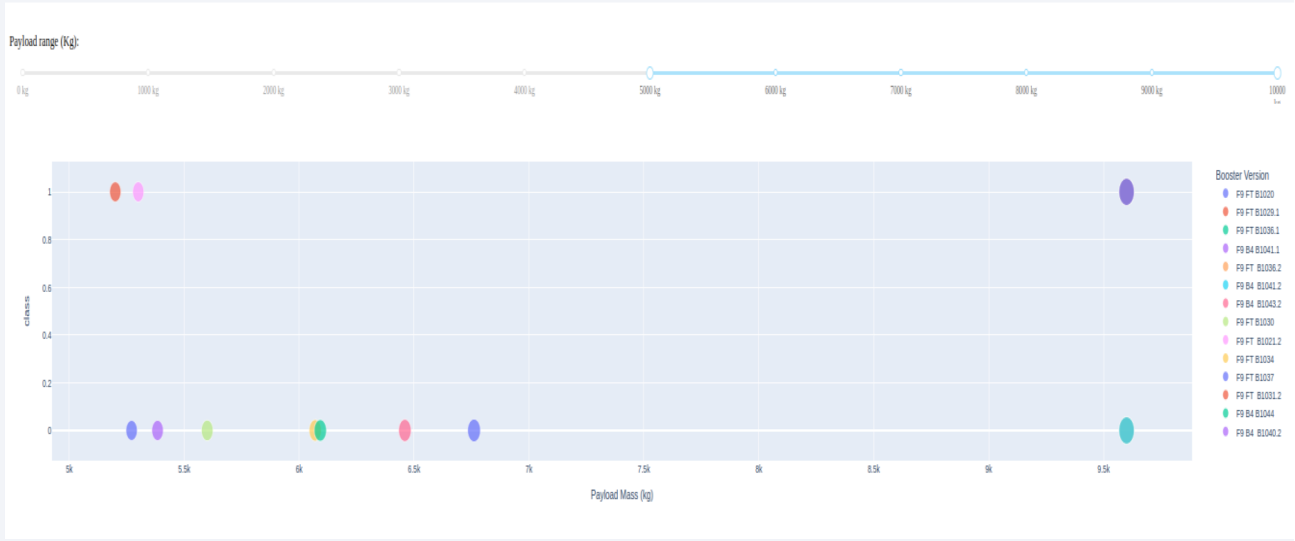
KSC LC-39A

Total Success Launches for site KSC LC-39A

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

- Outcomes for Low Payloads (0-5000) vs Outcomes for High Payloads (5000-10000)
- We can see that the success rate for low weighted payloads is higher than high weighted payloads.



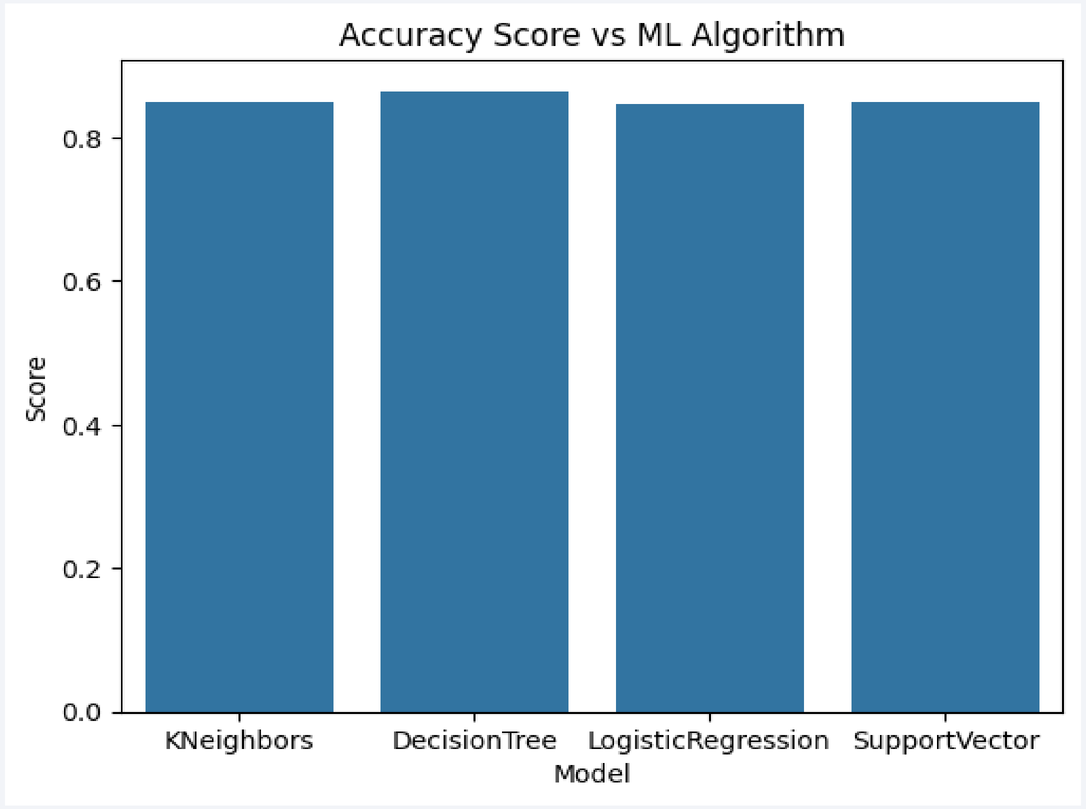


Section 5

Predictive Analysis (Classification)

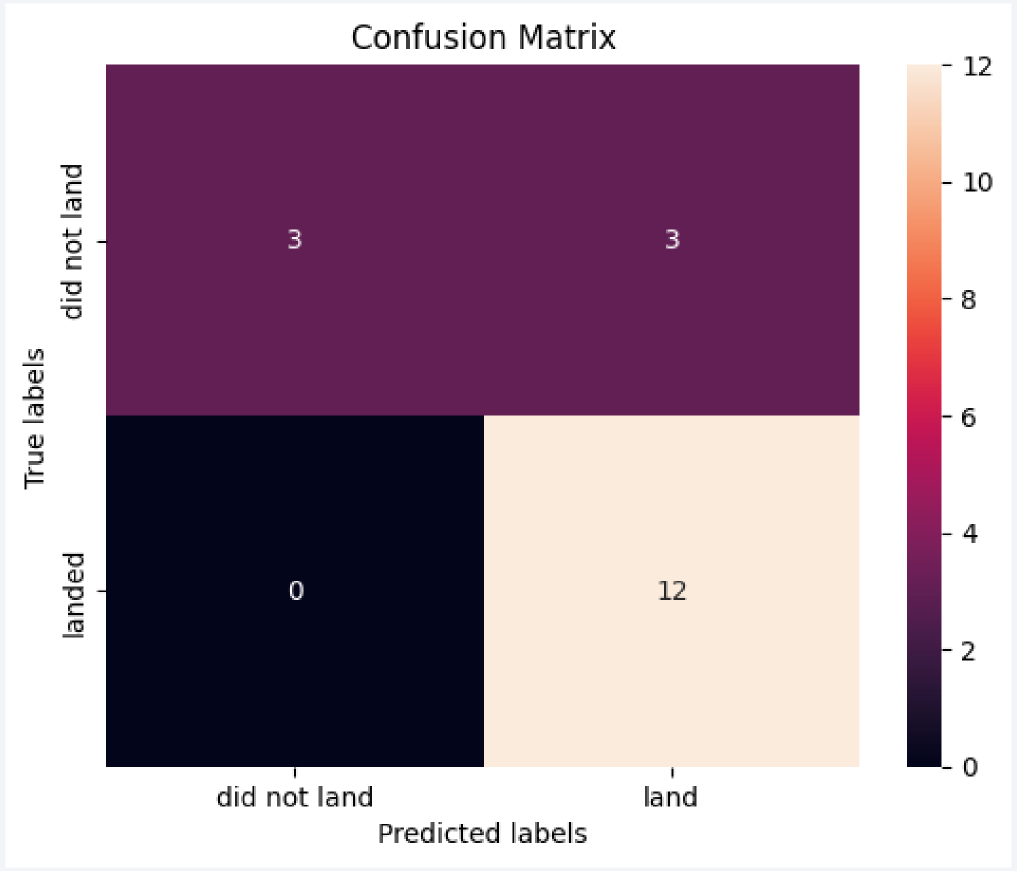
Classification Accuracy

- All the classification models are quite close in terms of accuracy.
- But, amongst them, the Decision Tree algorithm has the highest accuracy.



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

- We can conclude that:

  - 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 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

- Link to github repo, https://github.com/abhiJeetP10/IBM-Applied-DS-Capstone .

Thank you!