IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Karina Oportus>
<2024-08-06>

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion

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 operating 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

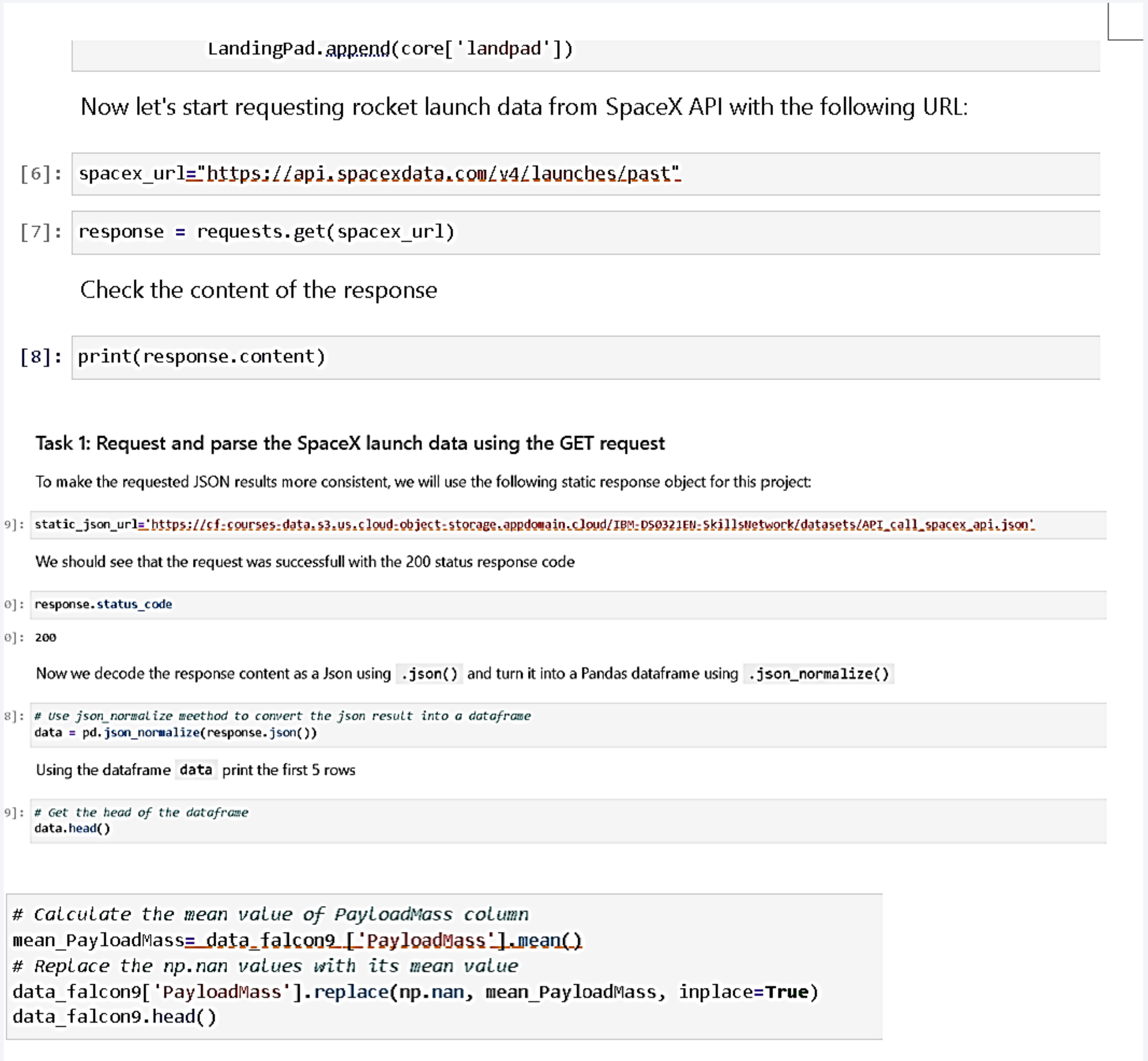
The data was compiled as follows:

- 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

- Used the SpaceX API get request to collect and clean requested data and performed some basic data formatting and modifications.
- Link de github

https://github.com/karina1404/Ciencia-de-datos-aplicada-Capstone.git



```
        LandingPad.append(core['landpad'])
```

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)
```

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

0]:
```
response.status_code
```

0]:
```
200
```

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

8]:
```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

Using the dataframe `data` print the first 5 rows

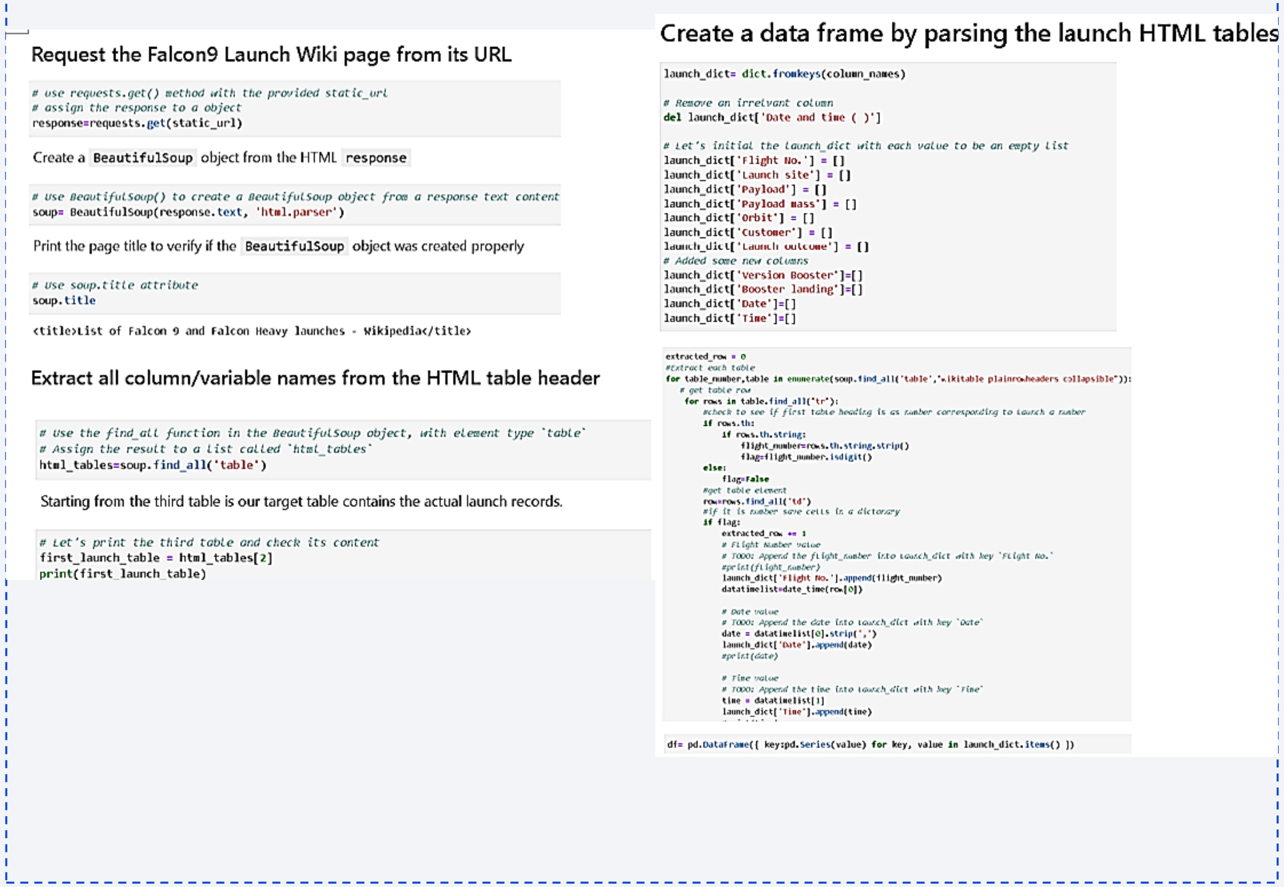
9]:
```
# Get the head of the dataframe
data.head()
```

```
# Calculate the mean value of PayloadMass column
mean_PayloadMass= data_falcon9 ['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].replace(np.nan, mean_PayloadMass, inplace=True)
data_falcon9.head()
```

Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas dataframe.
- Link de github

https://github.com/karina1404/Ciencia-de-datos-aplicada-Capstone.git



Request the Falcon9 Launch Wiki page from its URL

```
# 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`

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

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

```
# Use soup.title attribute
soup.title
```

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

Extract all column/variable names from the HTML table header

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

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

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

Create a data frame by parsing the launch HTML tables

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            #print(flight_number)
            launch_dict['Flight No.'].append(flight_number)
            datatimelist=date_time(row[0])

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date)
            #print(date)

            # Time value
            # TODO: Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            launch_dict['Time'].append(time)
```

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

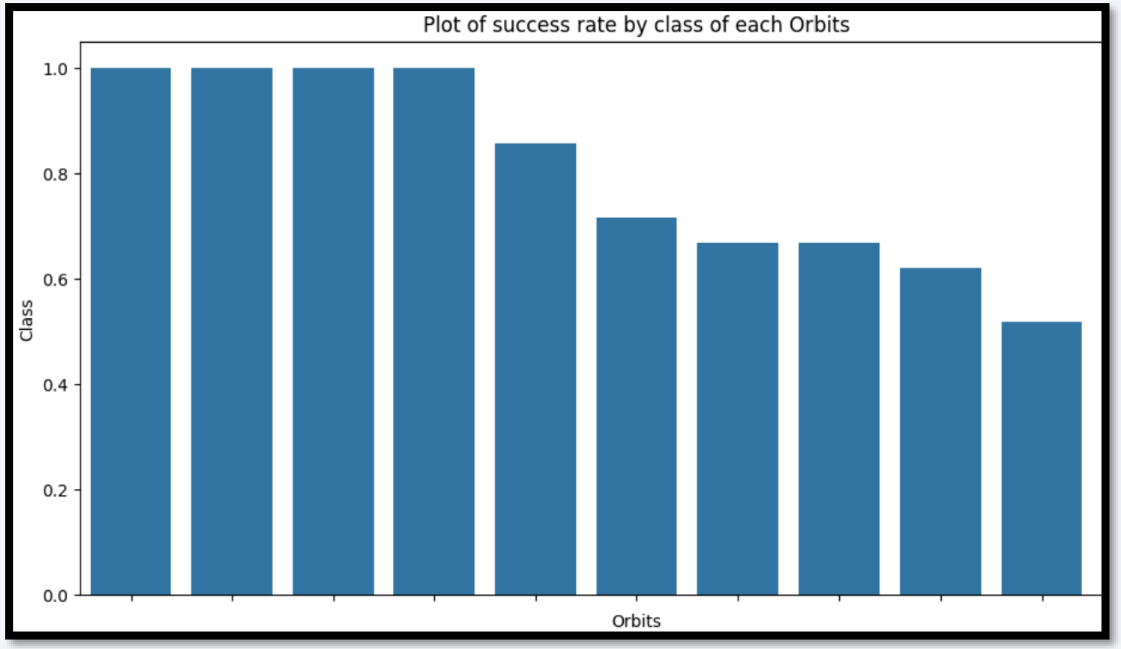
Data Wrangling

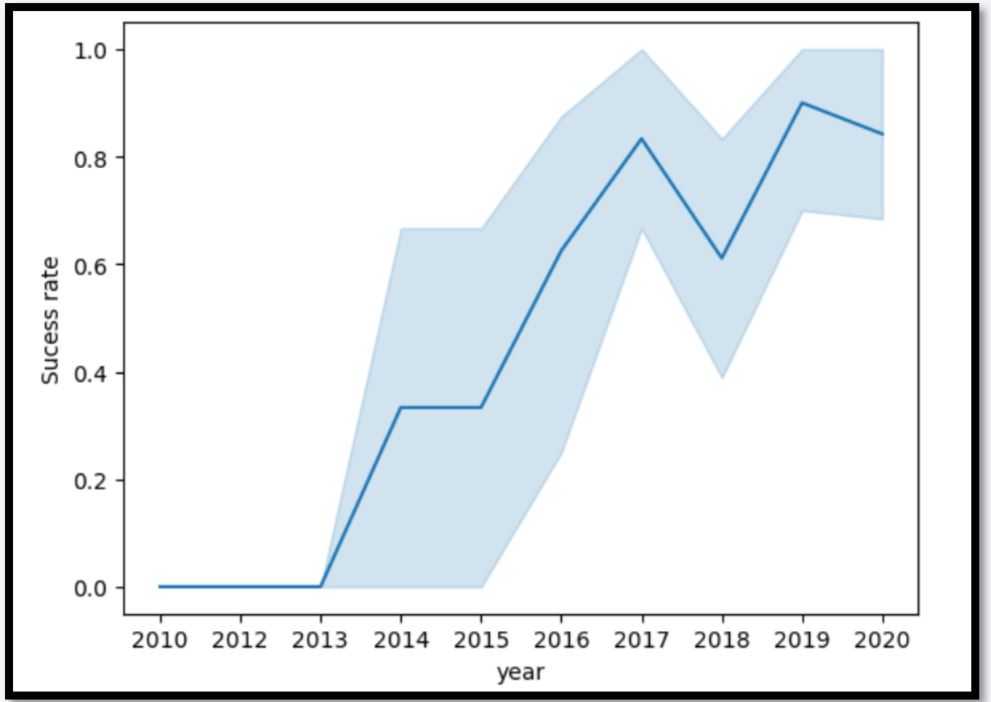
- Exploratory data analysis was performed, and training labels were determined.
- The number of launches at each site and the number and occurrence of each orbit were calculated.
- Created a landing result label from the results column and exported the results to csv.
- Link de github

https://github.com/karina1404/Ciencia-de-datos-aplicada-Capstone.git

EDA with Data Visualization

- The data was explored by visualizing the relationship between flight number and launch site, payload and launch site, the success rate of each orbit type, flight number and orbit type, and the trend annual launch success.





Link de github

https://github.com/karina1404/Ciencia-de-datos-aplicada-Capstone.git

EDA with SQL

- The SpaceX dataset was loaded into a PostgreSQL database without leaving the Jupyter notebook.
- EDA with SQL was applied to obtain information from the data. Consultations will be made to obtain:

  - the names of launch sites unique to the space mission.

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

  - the average payload mass carried by the F9 v1.1 booster version

  - the total number of successful and failed mission results.

  - the results of the unsuccessful landing on an unmanned ship, its booster version and the names of the launch sites.

- Link de GitHub

https://github.com/karina1404/Ciencia-de-datos-aplicada-Capstone.git

Build an Interactive Map with Folium

- Launch sites were marked and map objects such as markers, circles and lines were added to mark the success or failure of launches for each site on the folium map.
- The results of the function launch were assigned failure(0) or success(1).
- It was identified which launch sites have a relatively high success rate. The distances between a launch site and its surroundings were calculated. Answers are given to certain questions such as:

  - Do the launch sites maintain a certain distance from cities?

- Link de GitHub

https://github.com/karina1404/Ciencia-de-datos-aplicada-Capstone.git

Build a Dashboard with Plotly Dash

- Created an interactive dashboard with Plotly Dash
- Plotting pie charts showing total launches for certain sites.
- Additionally, a scatter plot was plotted showing the relationship with the result and the payload mass (kg) for the different booster versions.
- Link de GitHub

https://github.com/karina1404/Ciencia-de-datos-aplicada-Capstone.git

Predictive Analysis (Classification)

- The data was loaded with numpy and pandas, transforming the data, dividing it and training and testing.
- We built different machine learning models and tuned different hyperparameters using GridSearchCV.
- Accuracy was used as a metric for our model, we improved the model through feature engineering and algorithm tuning.
- The classification model with the best performance was chosen.
- Link de GitHub

https://github.com/karina1404/Ciencia-de-datos-aplicada-Capstone.git

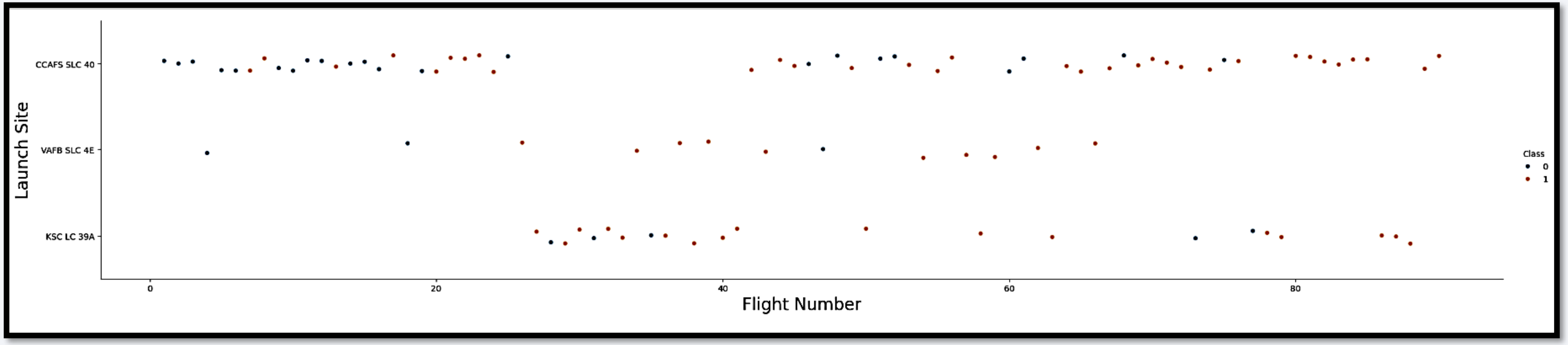
Results

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

Section 2

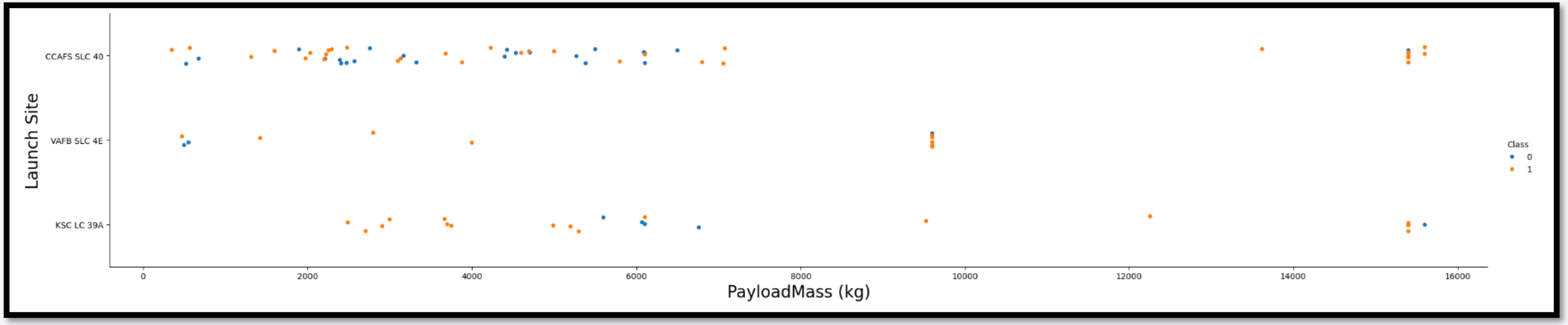
Insights drawn from EDA

Flight Number vs. Launch Site



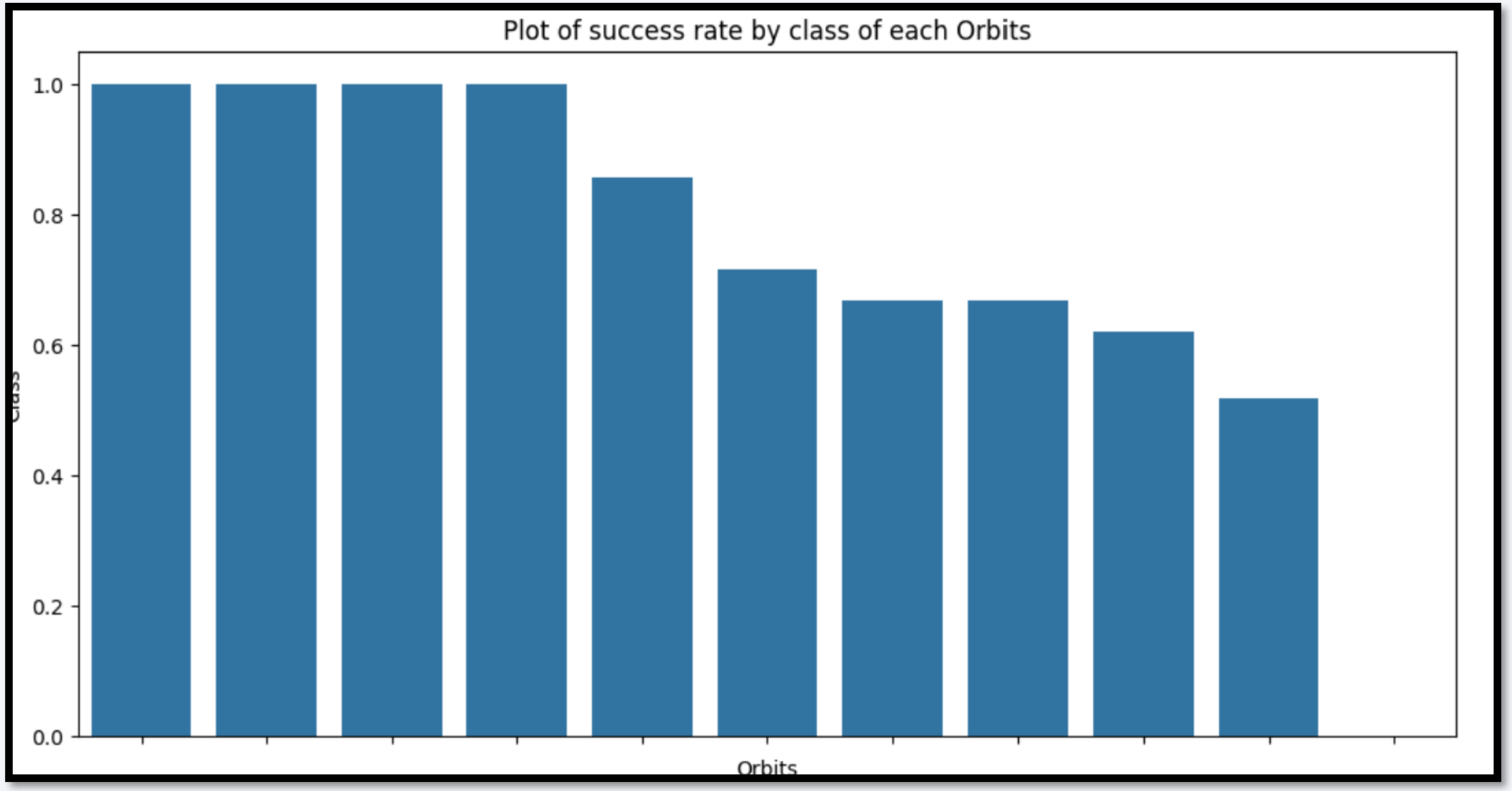
This graph shows flights launched from three different sites, with the flight number on the horizontal axis and the launch site on the vertical axis. Blue dots indicate unsuccessful flights and orange dots indicate successful flights. It is noted that there is a mix of successes and failures across all sites, but the distribution varies by launch site.

Payload vs. Launch Site



The graph shows the relationship between payload mass and launch sites, with different colored dots representing successful (Class 1) and unsuccessful (Class 0) launches. The launches are spread across three main sites, with the highest concentration at CCAFS SLC 40. The graph suggests that both the launch site and the mass of the payload could influence the outcome of the launch.

Success Rate vs. Orbit Type

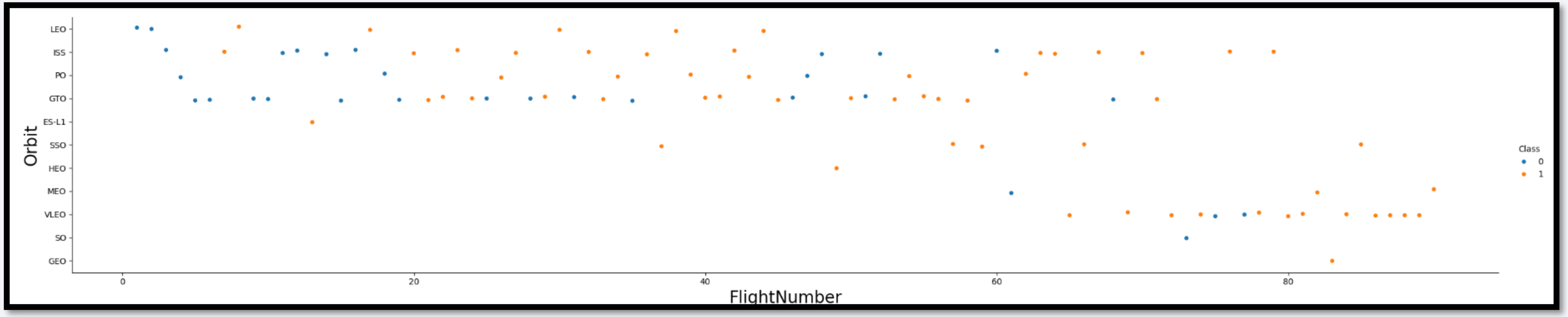


The bar graph shows the success rate of launches in different orbits. The vertical (y) axis represents the success rate, while the horizontal (x) axis represents the different orbits. Higher bars indicate a higher success rate.

Some orbits have a 100% success rate.

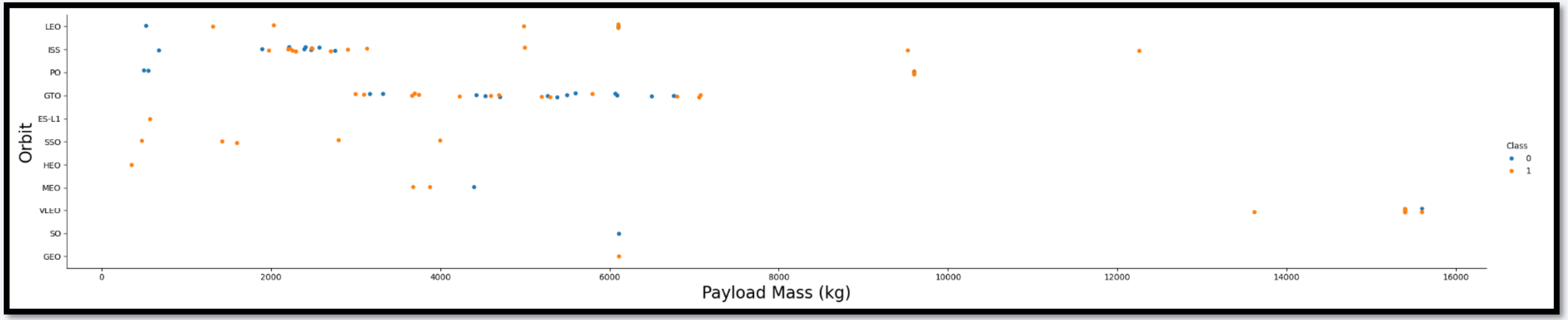
As you move to the right on the graph, the success rate decreases, indicating that certain orbits have a higher risk of launch failures.

Flight Number vs. Orbit Type



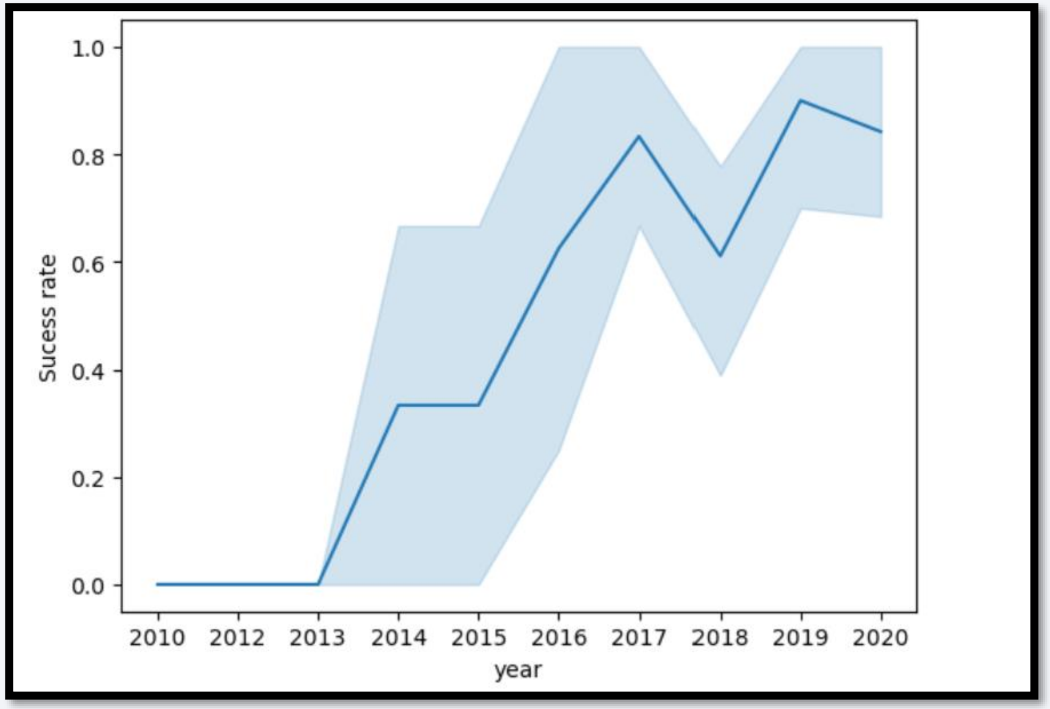
The plot below shows the Flight Number vs. Orbit type. We observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.

Payload vs. Orbit Type



We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.

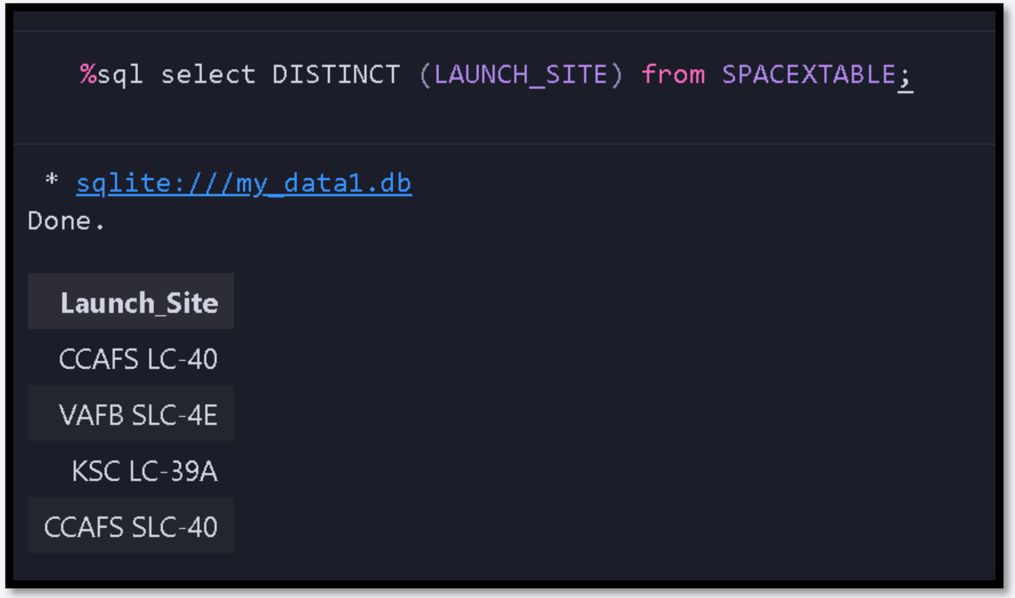
Launch Success Yearly Trend



From the plot, we can observe that success rate since 2013 kept on increasing till 2020.

All Launch Site Names

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

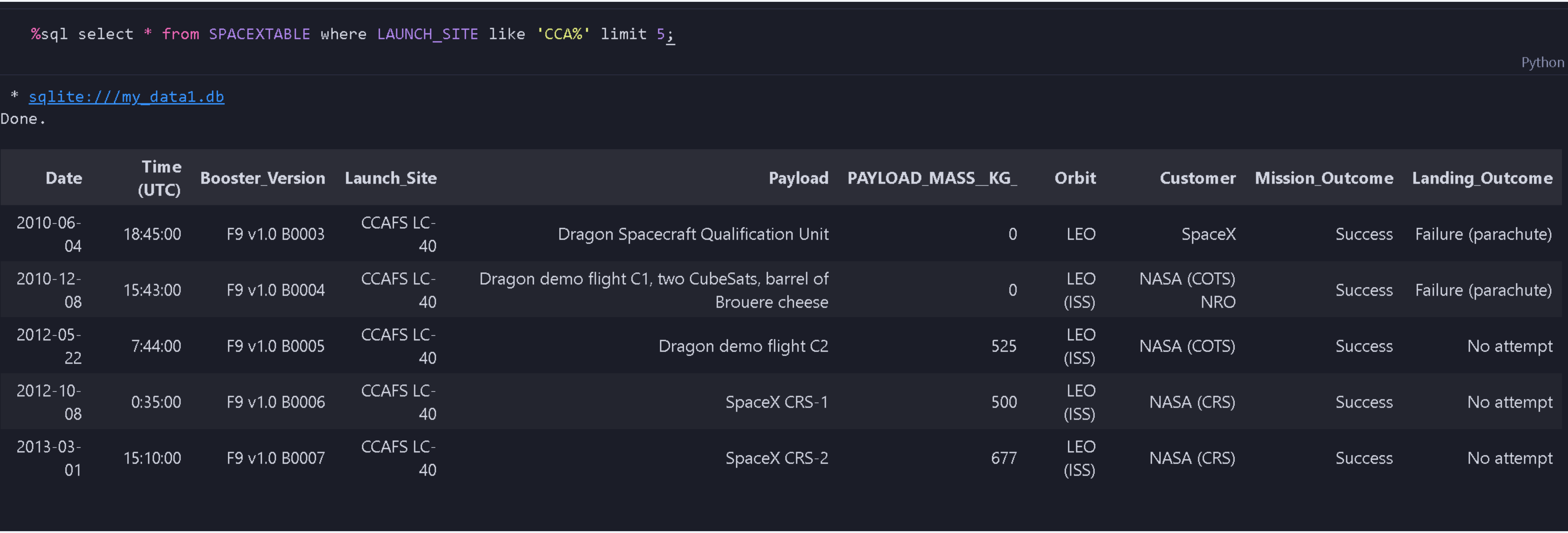


```
%sql select DISTINCT (LAUNCH_SITE) from SPACEXTABLE;
```

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

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

Launch Site Names Begin with 'CCA'



```
%sql select * from SPACEXTABLE where LAUNCH_SITE like 'CCA%' limit 5;
```

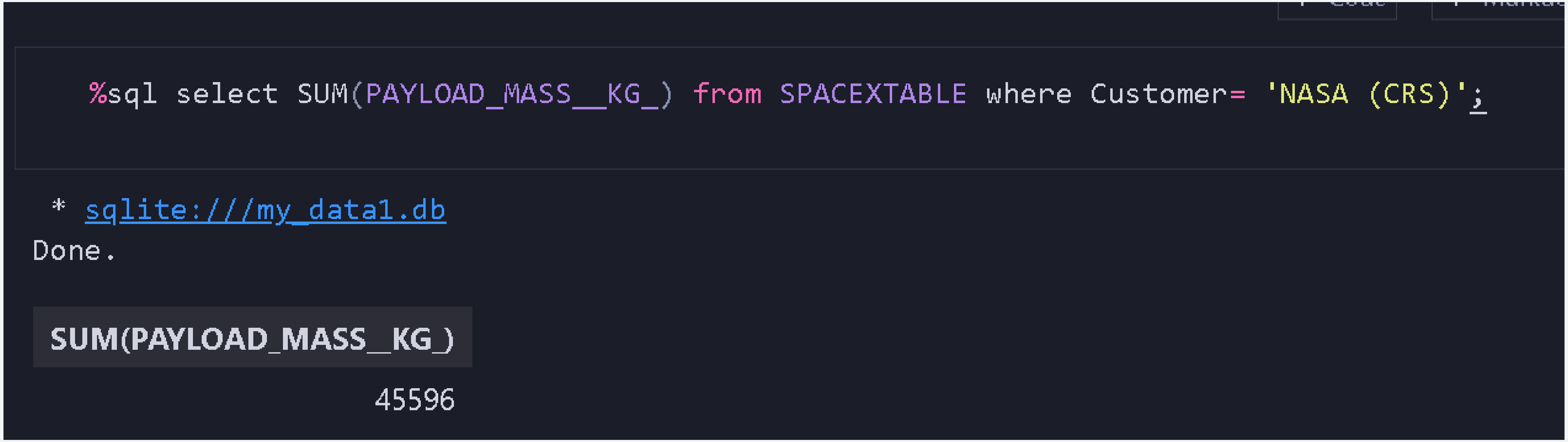
Python

```
 * 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

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

Total Payload Mass



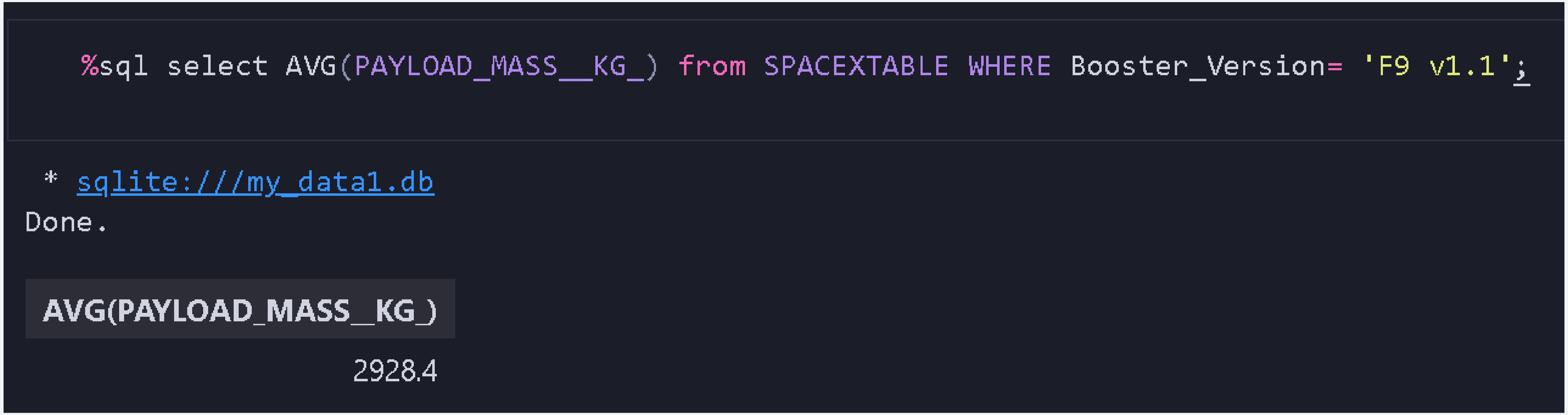
```
%sql select SUM(PAYLOAD_MASS__KG_) from SPACEXTABLE where Customer= 'NASA (CRS)';
```

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

SUM(PAYLOAD_MASS__KG_)
                 45596
```

We calculated the total payload carried by boosters from NASA as 45596

Average Payload Mass by F9 v1.1



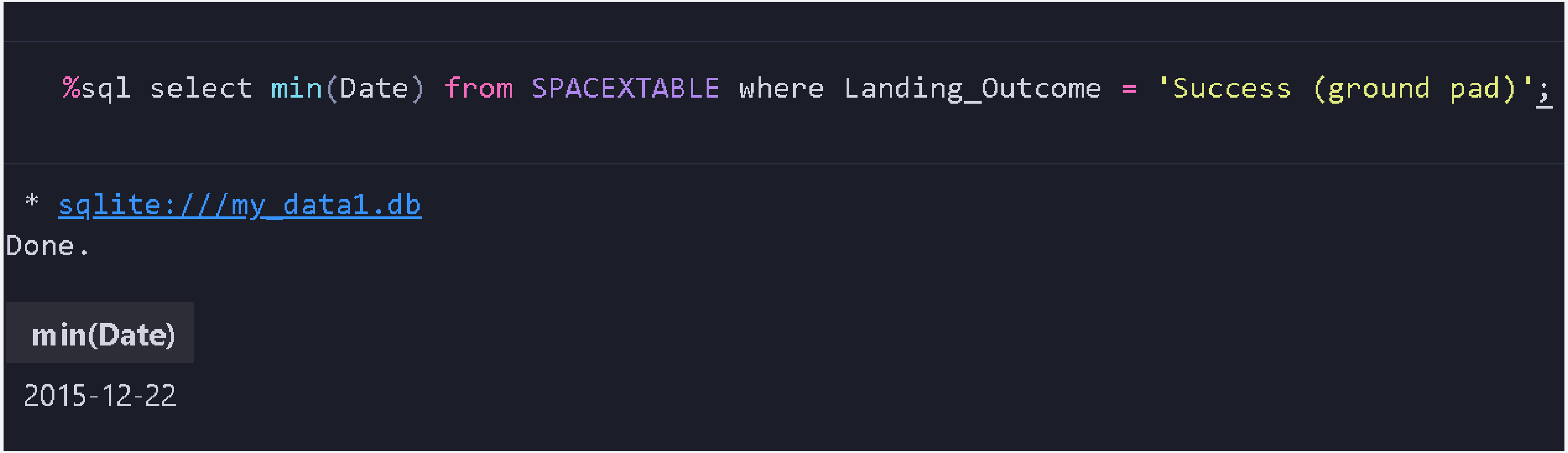
```
%sql select AVG(PAYLOAD_MASS__KG_) from SPACEXTABLE WHERE Booster_Version= 'F9 v1.1';
```

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

AVG(PAYLOAD_MASS__KG_)
2928.4

We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4

First Successful Ground Landing Date



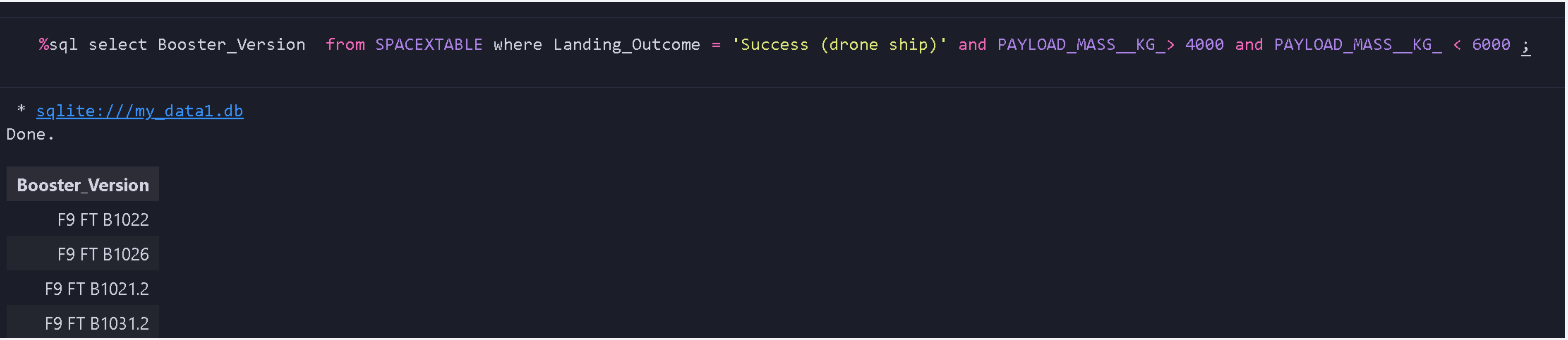
```
%sql select min(Date) from SPACEXTABLE where Landing_Outcome = 'Success (ground pad)';
```

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

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

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

Successful Drone Ship Landing with Payload between 4000 and 6000



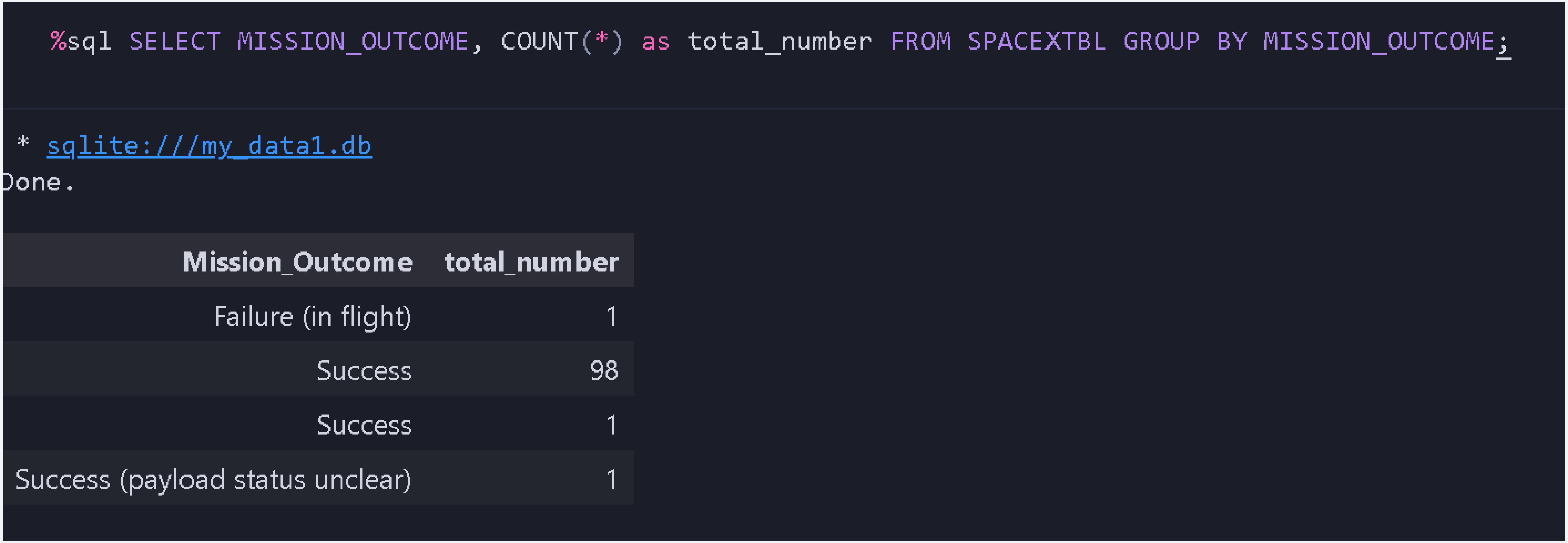
```
%sql select Booster_Version  from SPACEXTABLE where Landing_Outcome = 'Success (drone ship)' and PAYLOAD_MASS__KG_> 4000 and PAYLOAD_MASS__KG_ < 6000 ;
```

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

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



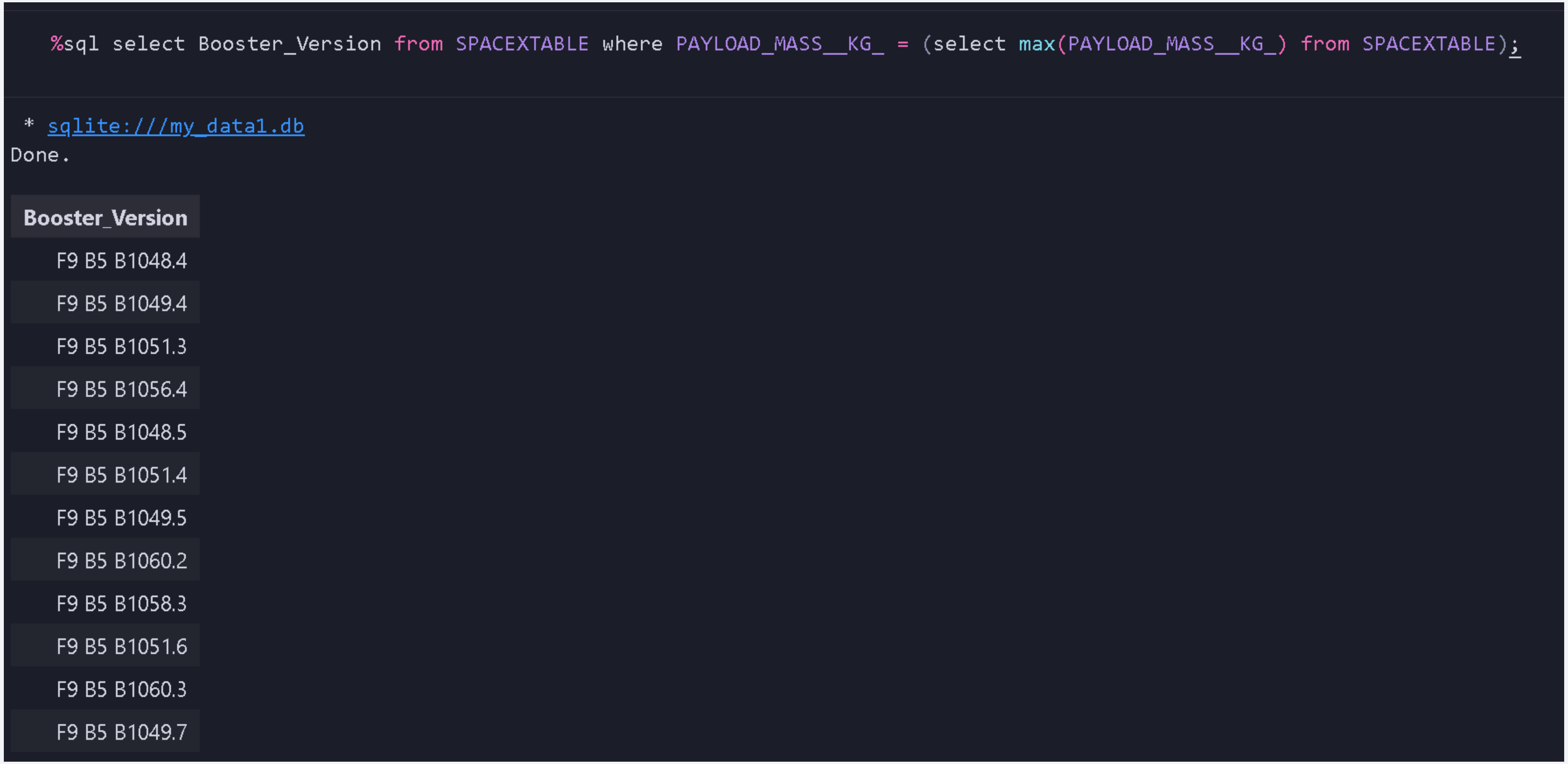
```
%sql SELECT MISSION_OUTCOME, COUNT(*) as total_number FROM SPACEXTBL GROUP BY MISSION_OUTCOME;
```

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

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

We count(*) for Mission_Outcome was a success or a failure.

Boosters Carried Maximum Payload



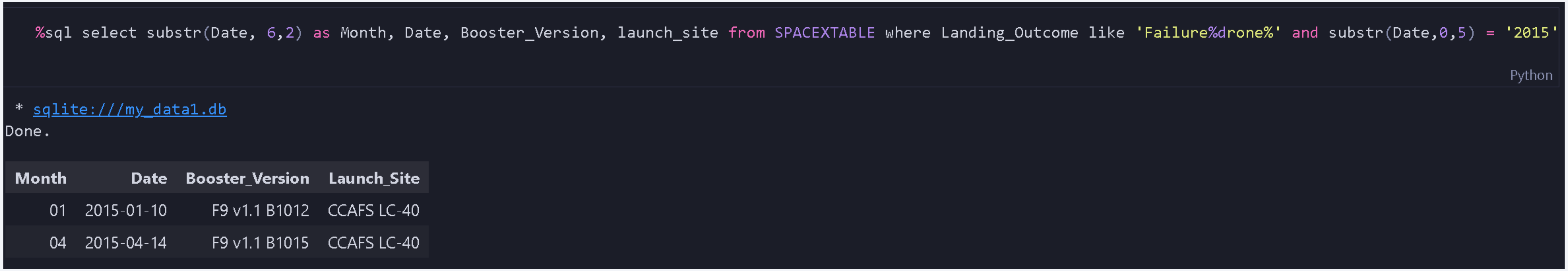
```
%sql select Booster_Version from SPACEXTABLE where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTABLE);
```

```
 * 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

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

2015 Launch Records



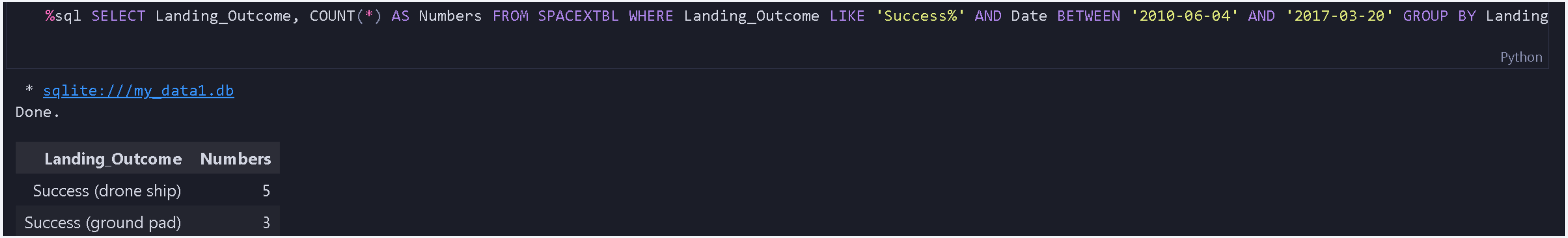
```
%sql select substr(Date, 6,2) as Month, Date, Booster_Version, launch_site from SPACEXTABLE where Landing_Outcome like 'Failure%drone%' and substr(Date,0,5) = '2015'
```

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

Month	Date	Booster_Version	Launch_Site
01	2015-01-10	F9 v1.1 B1012	CCAFS LC-40
04	2015-04-14	F9 v1.1 B1015	CCAFS LC-40

We used a combinations of the **WHERE** clause **LIKE and AND**, 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



```
%sql SELECT Landing_Outcome, COUNT(*) AS Numbers FROM SPACEXTBL WHERE Landing_Outcome LIKE 'Success%' AND Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing
```

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

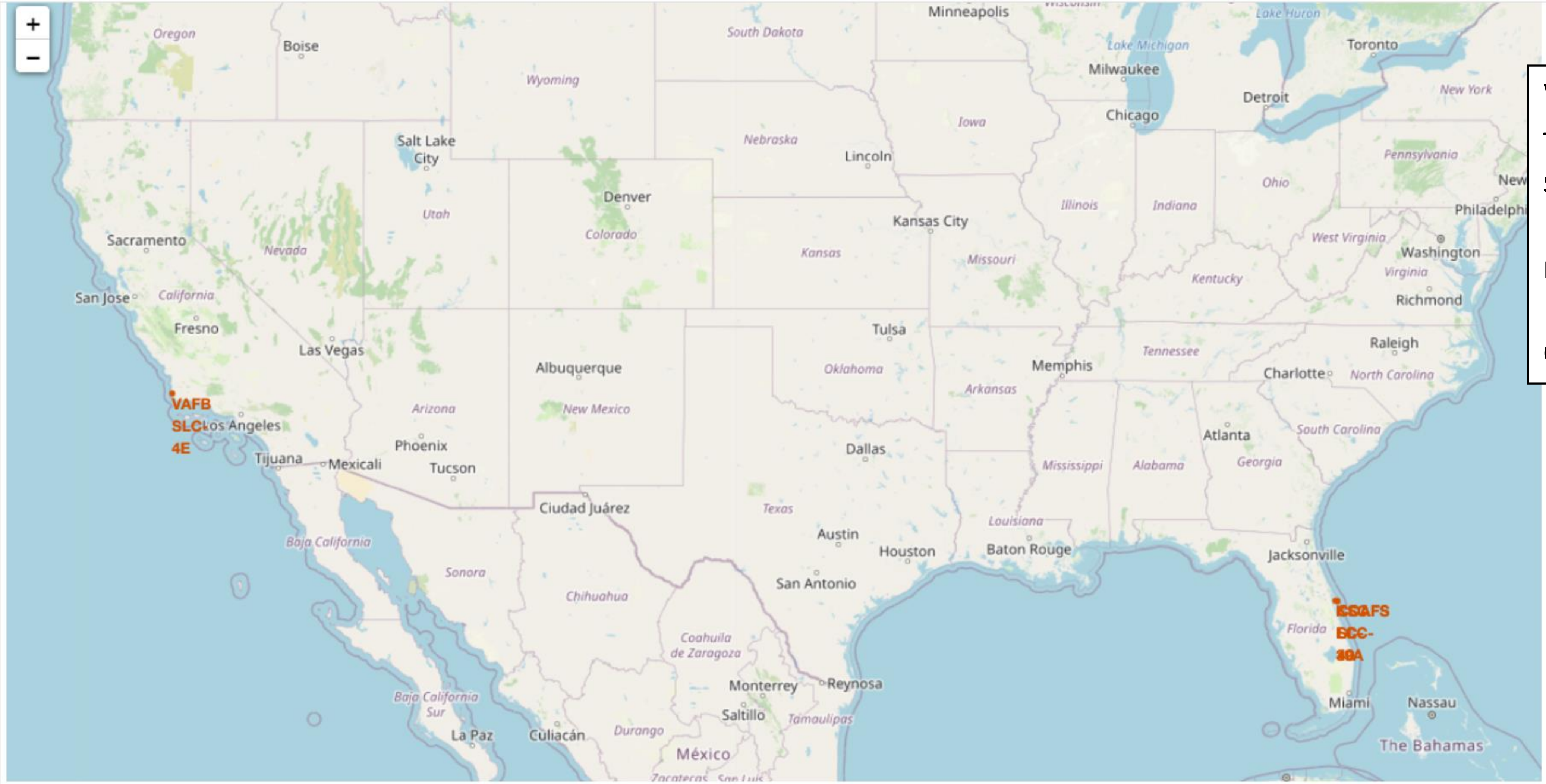
Landing_Outcome	Numbers
Success (drone ship)	5
Success (ground pad)	3

We 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

Section 3

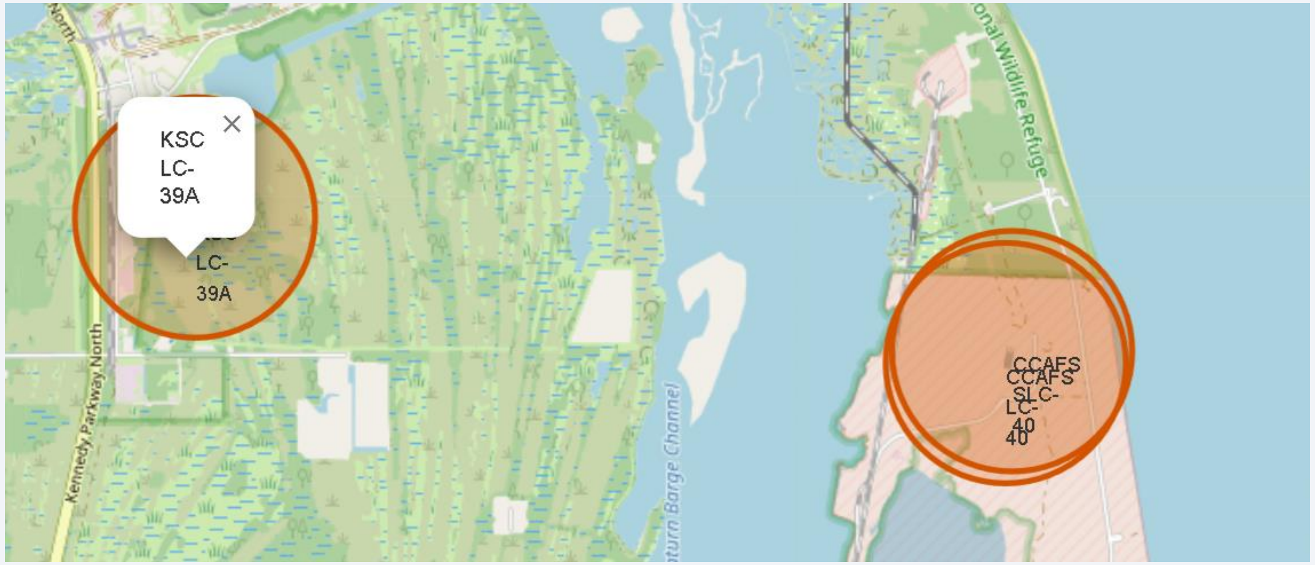
Launch Sites Proximities Analysis

All launch sites global map markers



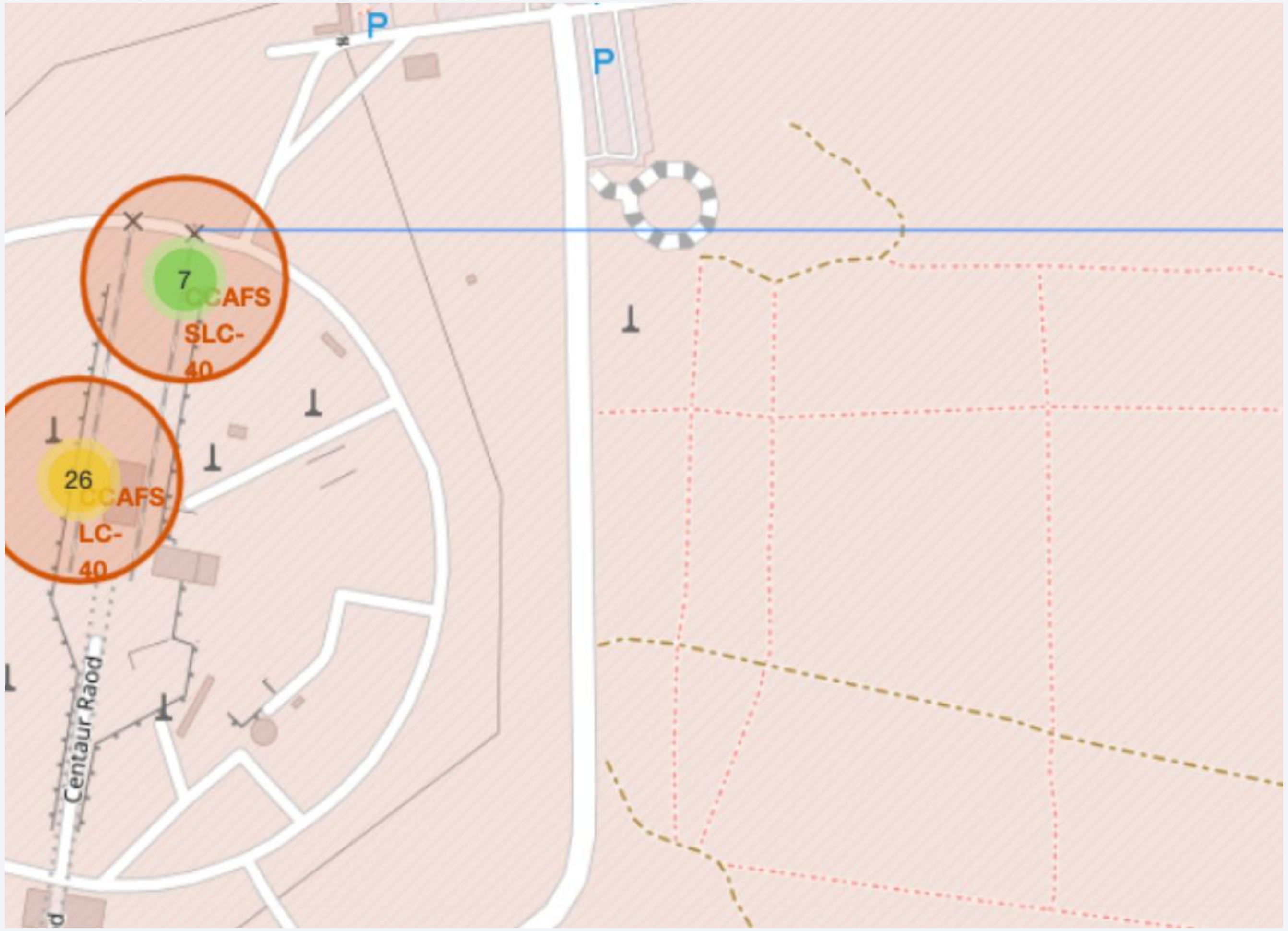
We can see that the Space X launch sites are in the United States of A merica costas. Florida and California.

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

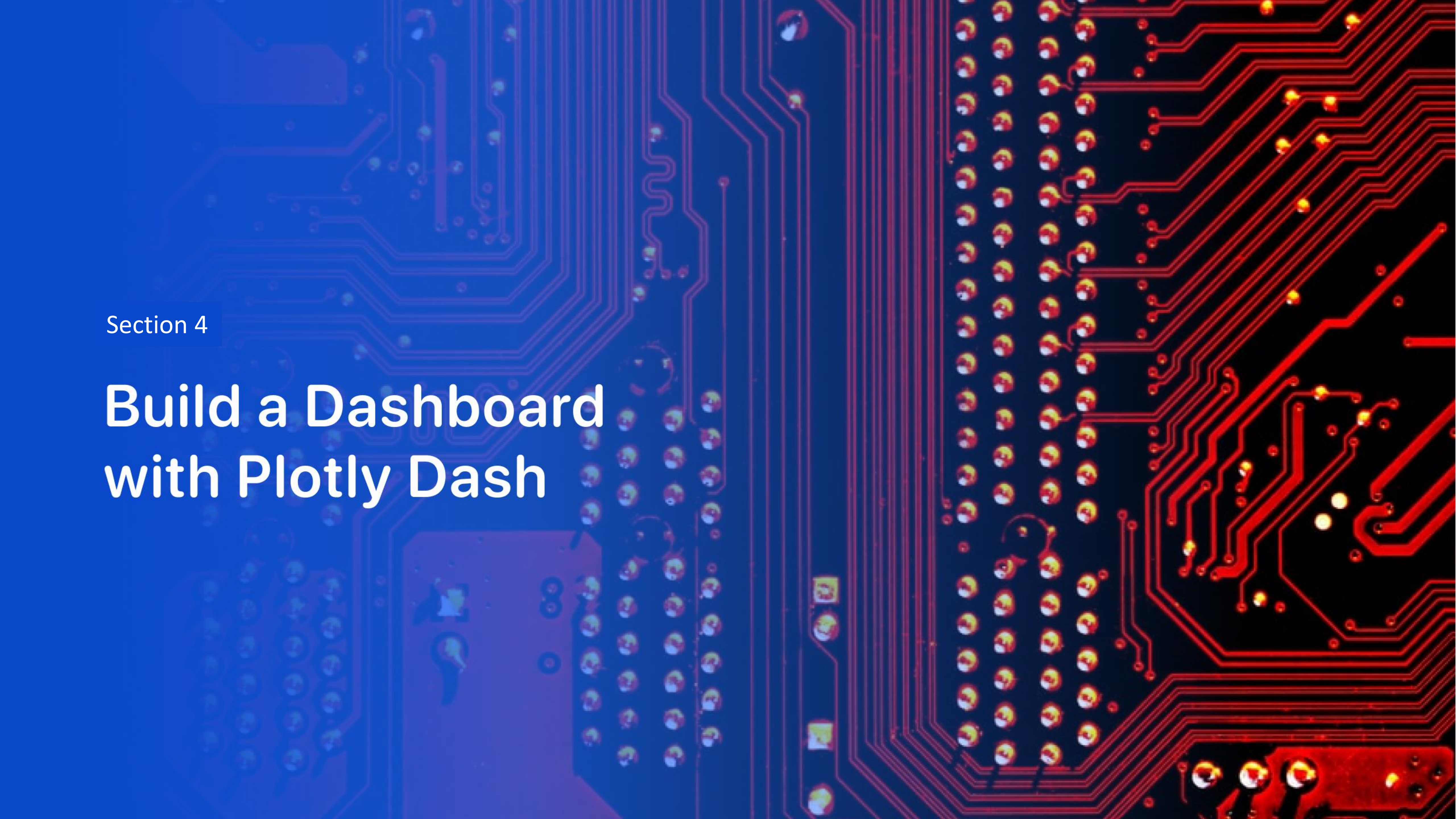


0 (red): launch failed
1(green): successful launch

Distances between a launch site to its proximities



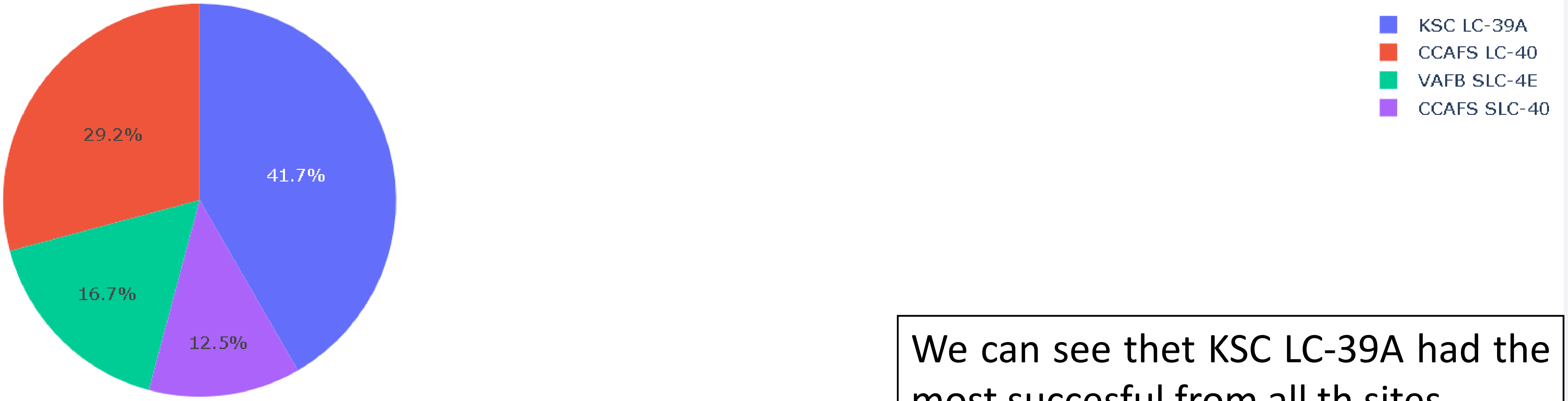
1. Are launch sites in close proximity to railways? no
2. Are launch sites in close proximity to highways? no
3. Are launch sites in close proximity to coastline? yes
4. Do launch sites keep certain distance away from cities? yes



Section 4

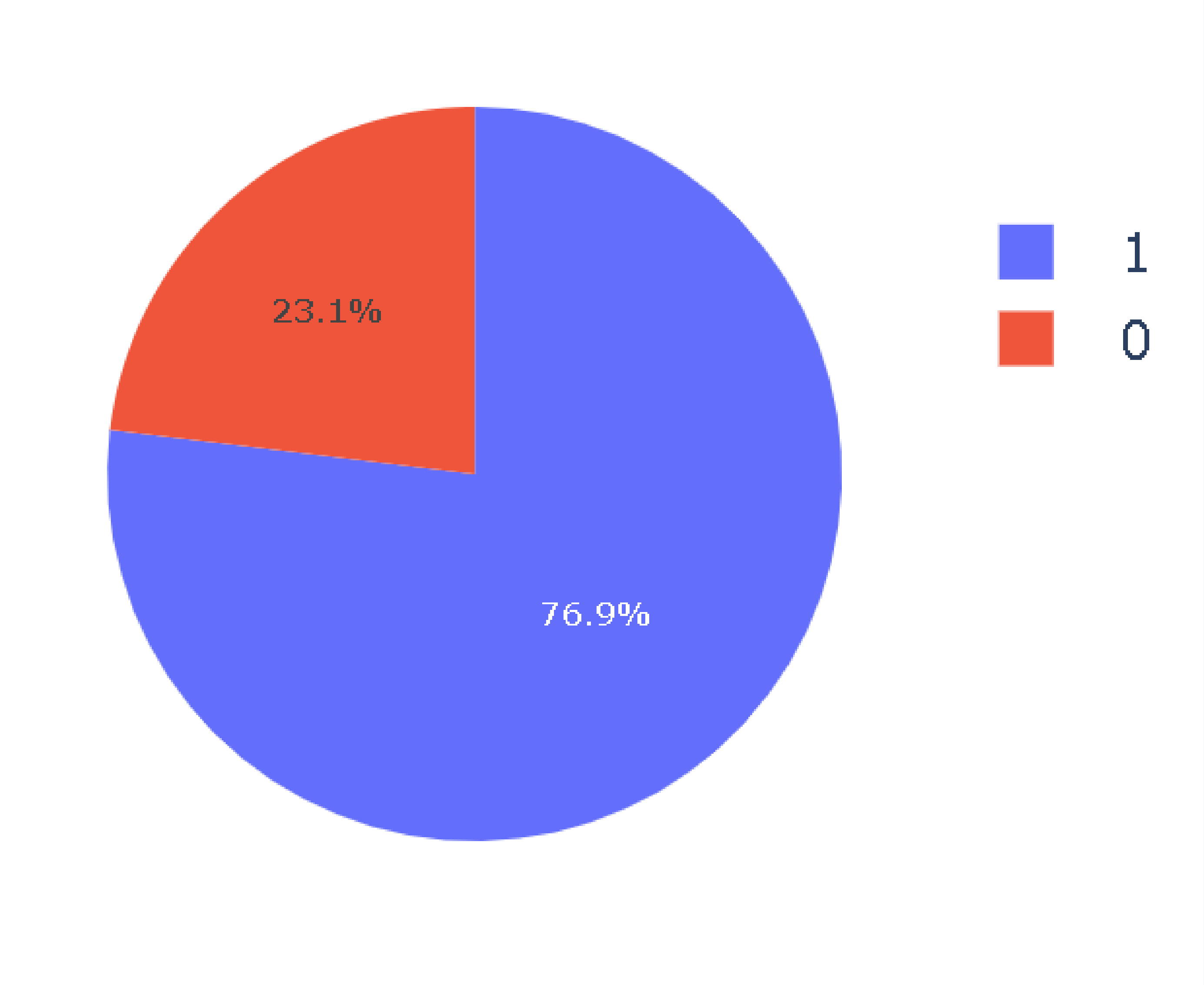
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



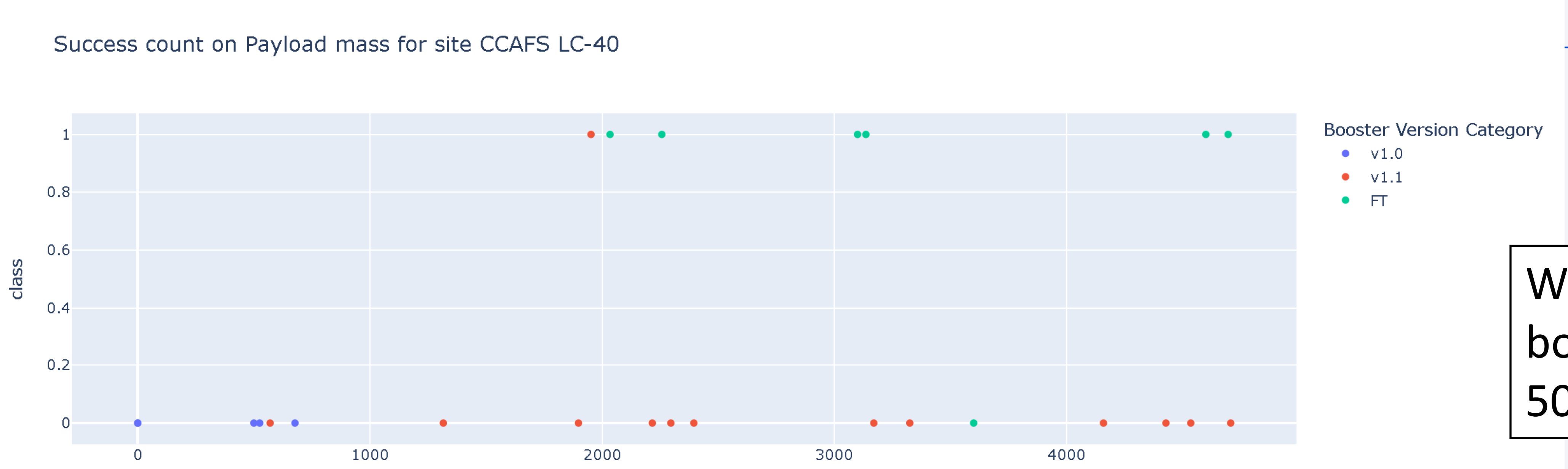
We can see thet KSC LC-39A had the most succesful from all th sites.

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

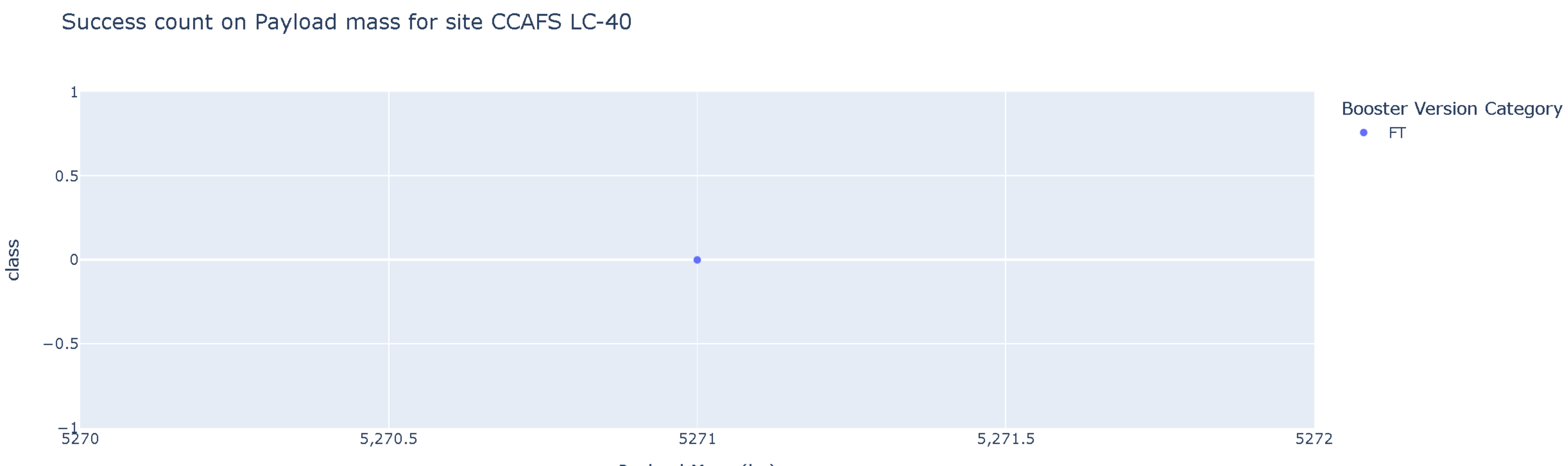


KSC LC-39A achieved a 76.9% succes rate while getting a 23.1% failure rate

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



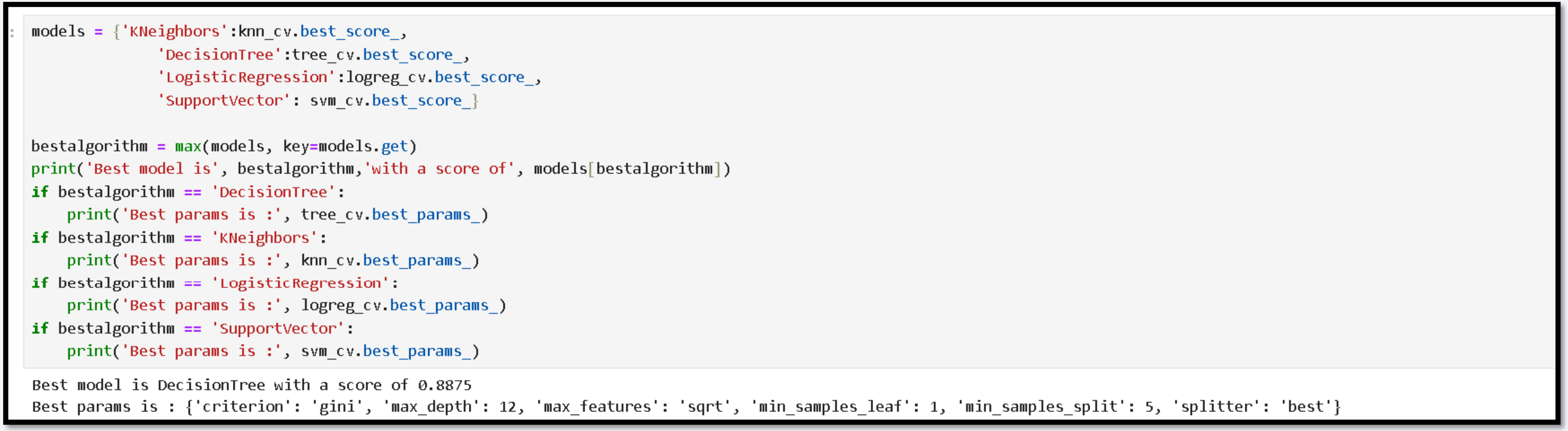
We can see many boster in range 0-5000 kg



Section 5

Predictive Analysis (Classification)

Classification Accuracy



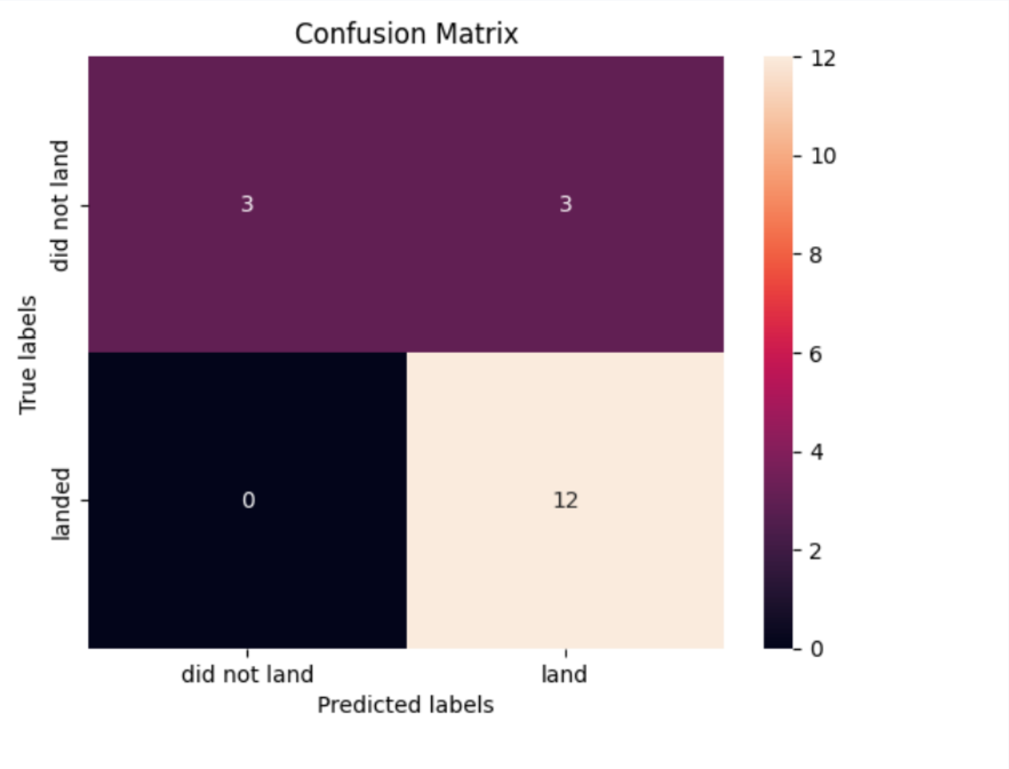
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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 0.8875
Best params is : {'criterion': 'gini', 'max_depth': 12, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5, 'splitter': 'best'}
```

The decision tree classifier is the model with the highest classification 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

- 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.

Thank you!