IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Emmanuel Ani
3/13/223

Outline

Executive Summary

- **Summary of methodologies**
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling (Manipulation)
  - 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
  - Screenshots of interactive analytics
  - Predictive Analytics result

Introduction

This project is focused on the commercial space transport service, SpaceX, who have relatively succeeded in the space program due to their relatively inexpensive rocket launches. The aim of the project is to use the SpaceX datasets to determine the cost of a launch by predicting if the first stage of the rocket would land. The data sets were collected via an API. I used SQL and Python primarily for the whole project. Data Wrangling and EDA and visualization were done to understand some of the variables in the dataset. Folium was used in plotting the sites of the launches and other relative information. Finally, a Machine Learning model was created to predict the success of failure of each rocker launches using different classification algorithms.

The results gotten would be presented in details in the subsequent slides for your perusal.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
    - Data was collected by web scraping and via spacex API
- Perform data wrangling
    - Data was processed using pandas
- 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 by different methods:

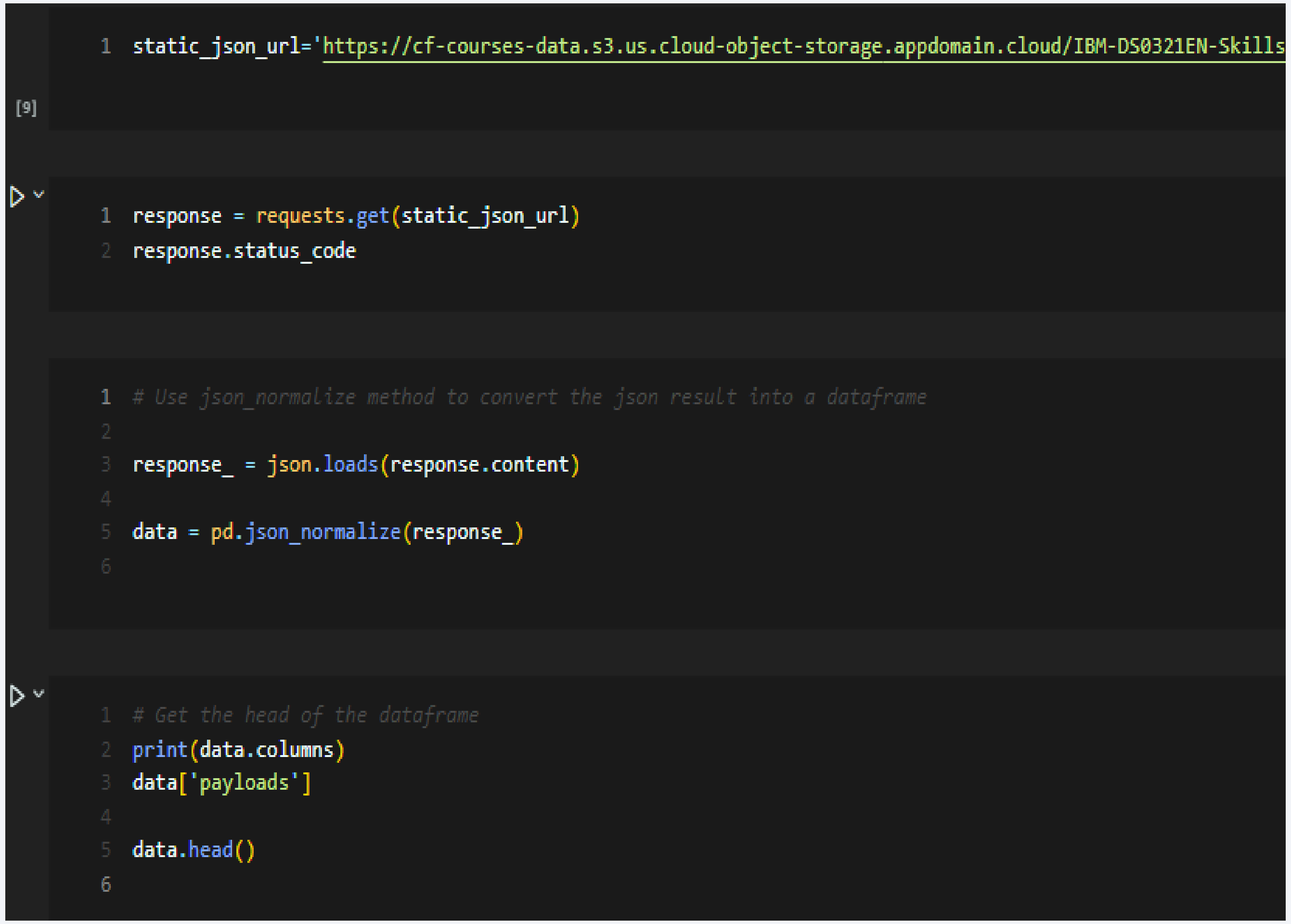
Via SpaceX API

This requires me sending an API request to the SpaceX server and getting a json response that can be parsed and analyzed. This was done using the request library.

Via Webscraping

Here, the data was collected by scraping Wikipedia page for information containing SpaceX rockets using beautiful soup library. The scraped html data is then parsed using html parser and the necessary data were extracted and used for the analysis.

Data Collection – SpaceX API



```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-Skills
```

[9]

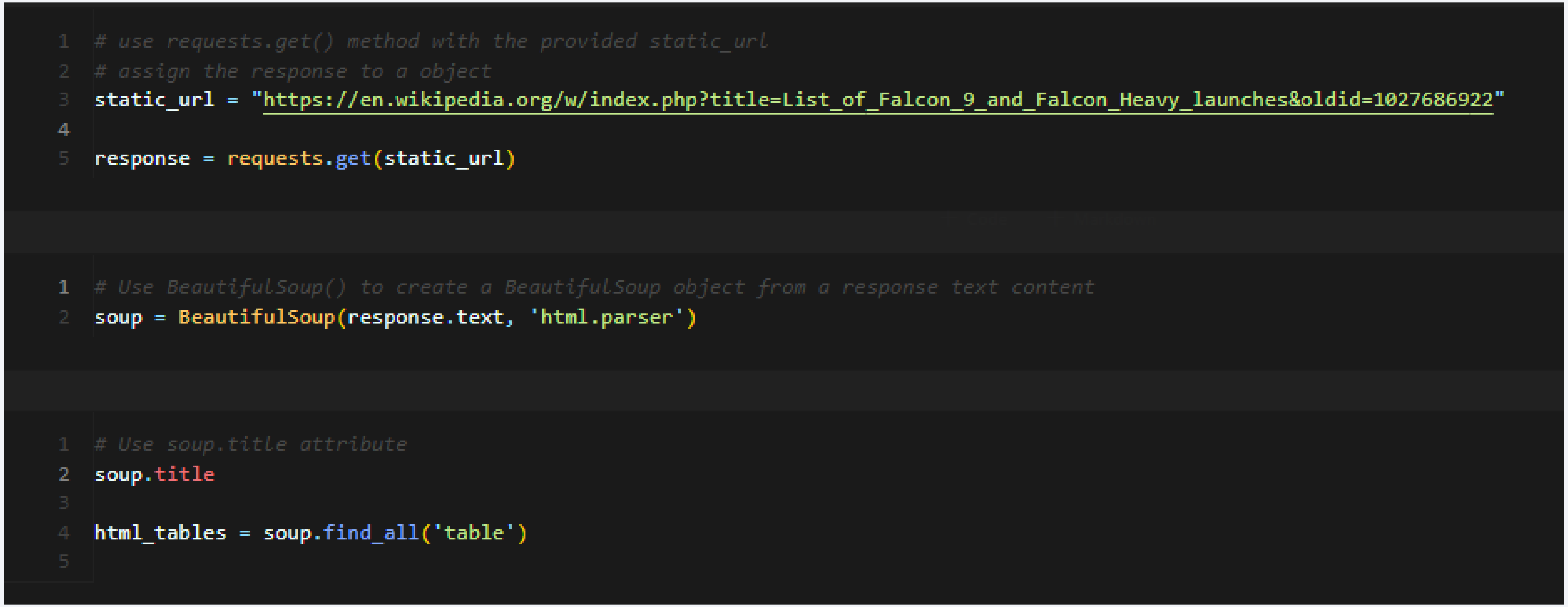
```
response = requests.get(static_json_url)
response.status_code
```

```
# Use json_normalize method to convert the json result into a dataframe

response_ = json.loads(response.content)

data = pd.json_normalize(response_)

```

```
# Get the head of the dataframe
print(data.columns)
data['payloads']

data.head()

```

github: https://github.com/emmanuelani/ibm_ds_capstone/blob/master/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping



```
# use requests.get() method with the provided static_url
# assign the response to a object
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"

response = requests.get(static_url)
```

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

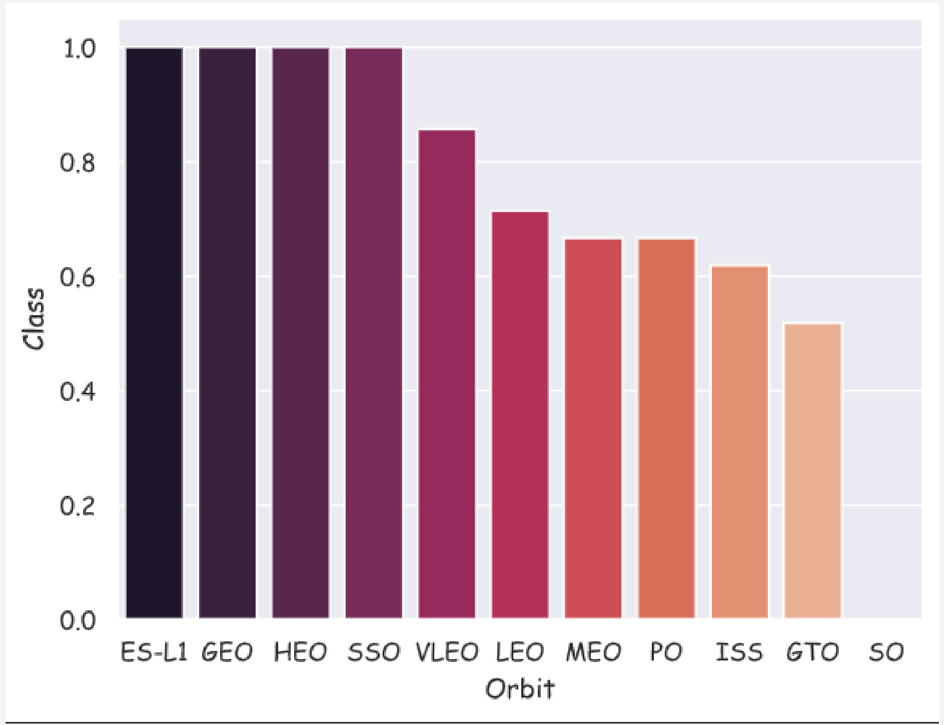
```
# Use soup.title attribute
soup.title

html_tables = soup.find_all('table')

```

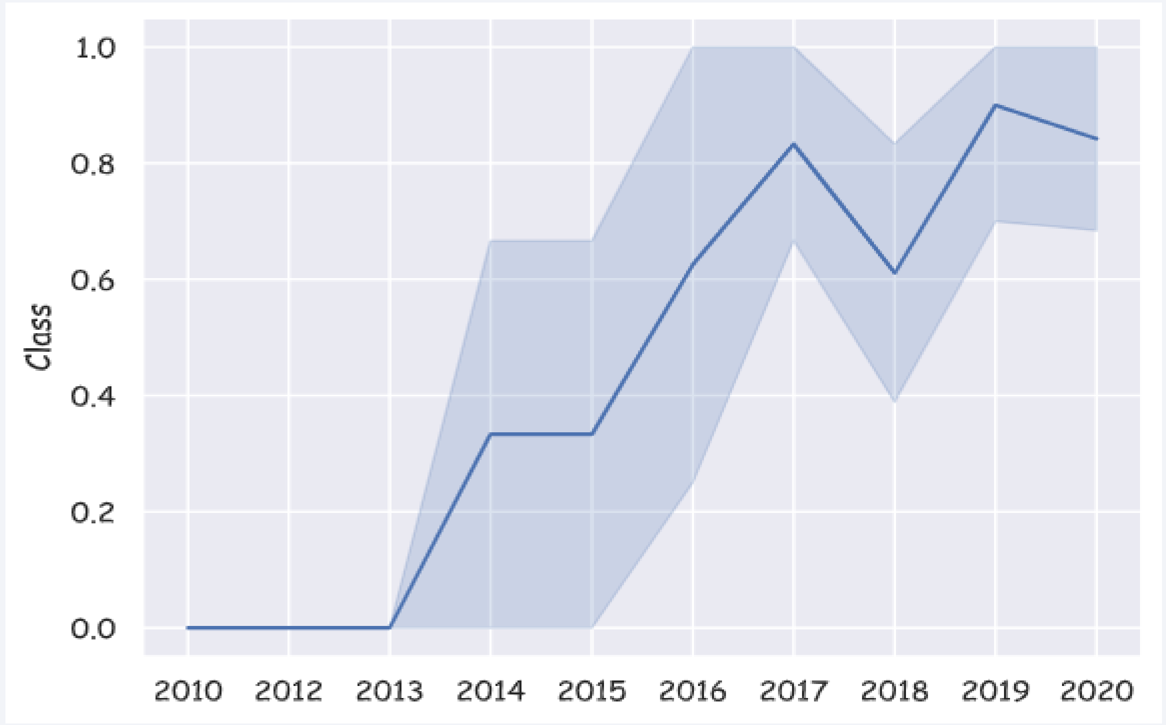
Github url: https://github.com/emmanuelani/ibm_ds_capstone/blob/master/Webscraping.ipynb

EDA with Data Visualization

Bar charts and Line charts were plotted. The bar chart was used to visualize the success rate for each orbit while the line chart was used to show the trend of success rates over the years.

Github: https://github.com/emmanuelani/ibm_ds_capstone/blob/master/jupyter-labs-eda-dataviz.ipynb





EDA with SQL

- The data used for the EDA was loaded into SQL Server and then a connection was made to the server using pyodbc module.
- A cursor object was then created from the connection string which was used to run the SQL queries in python.
- The link below would provide a more detailed look into the EDA using SQL

Github:
https://github.com/emmanuelani/ibm_ds_capstone/blob/master/SQL%20EDA.ipynb

Build an Interactive Map with Folium

- Using folium API, I was able to create interactive maps that visualized the launch sites.
- I was also able to display all the different launches at the different sites and label the successful and the failed launches using MarkerCluster object.
- I used MousePosition to get the coordinates of coastline, railway, highway and a city and calculated the distance between the launch centre and these locations.

Github:
https://github.com/emmanuelani/ibm_ds_capstone/blob/master/Launch%20Sites%20Locations%20Analysis%20using%20Folium.ipynb

Predictive Analysis (Classification)

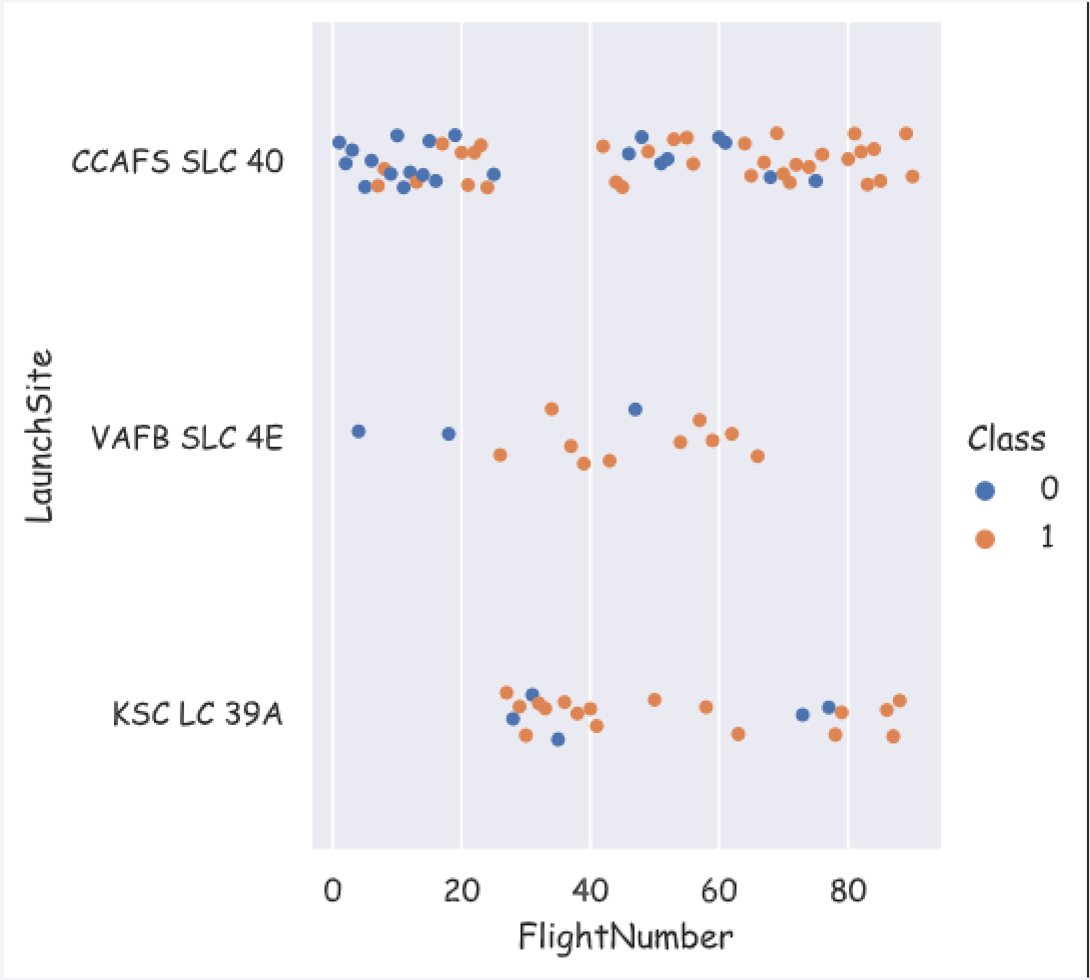
- Using several classification algorithm. I was able to find the best model that most accurately predicted if the falcon 9 rocket would land or not.
- Using standard scaler, the data was normalized before passing it over to the algorithms.
- It was then split into train and test data using train_test_split function.
- After training the data on different algorithms. The accuracy score of each model was plotted on a bar chart and Decision tree came out to be the model that performed best.

Github:
https://github.com/emmanuelani/ibm_ds_capstone/blob/master/Prediction%20using%20Machine%20Learning.ipynb

Section 2

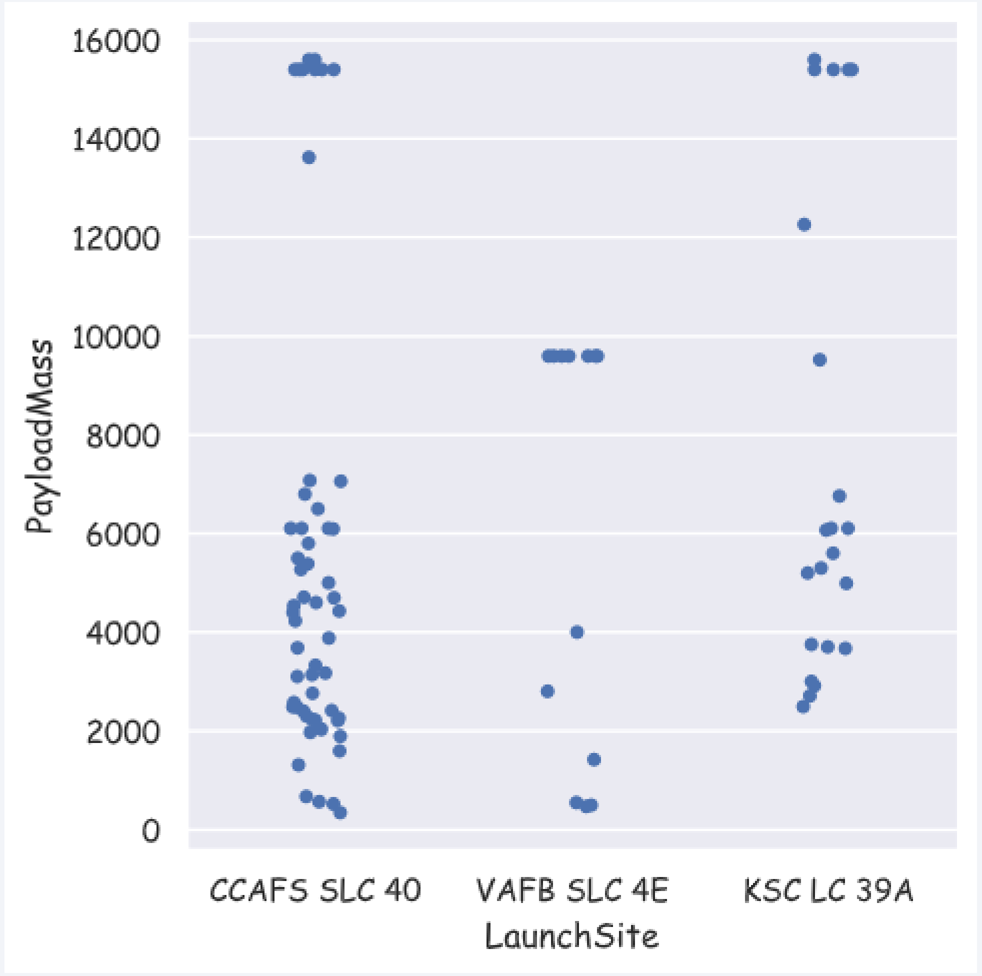
Insights drawn from EDA

Flight Number vs. Launch Site



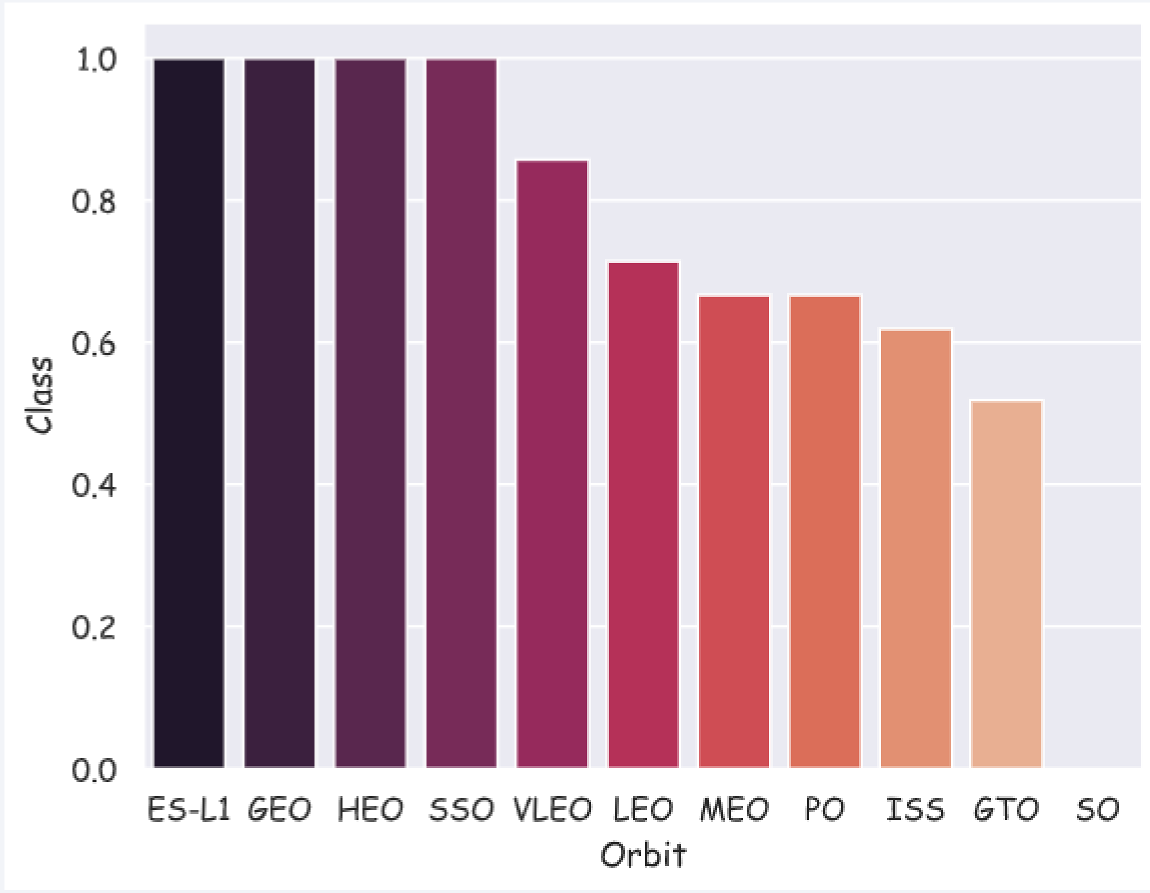
We can see clearly from the chart the number of flights that succeeded and failed in each launch site. KSC LC 39A and VAFB SLC 4E clearly had lesser number of flights and a higher success rate than CCAFS SLC 40.

Payload vs. Launch Site



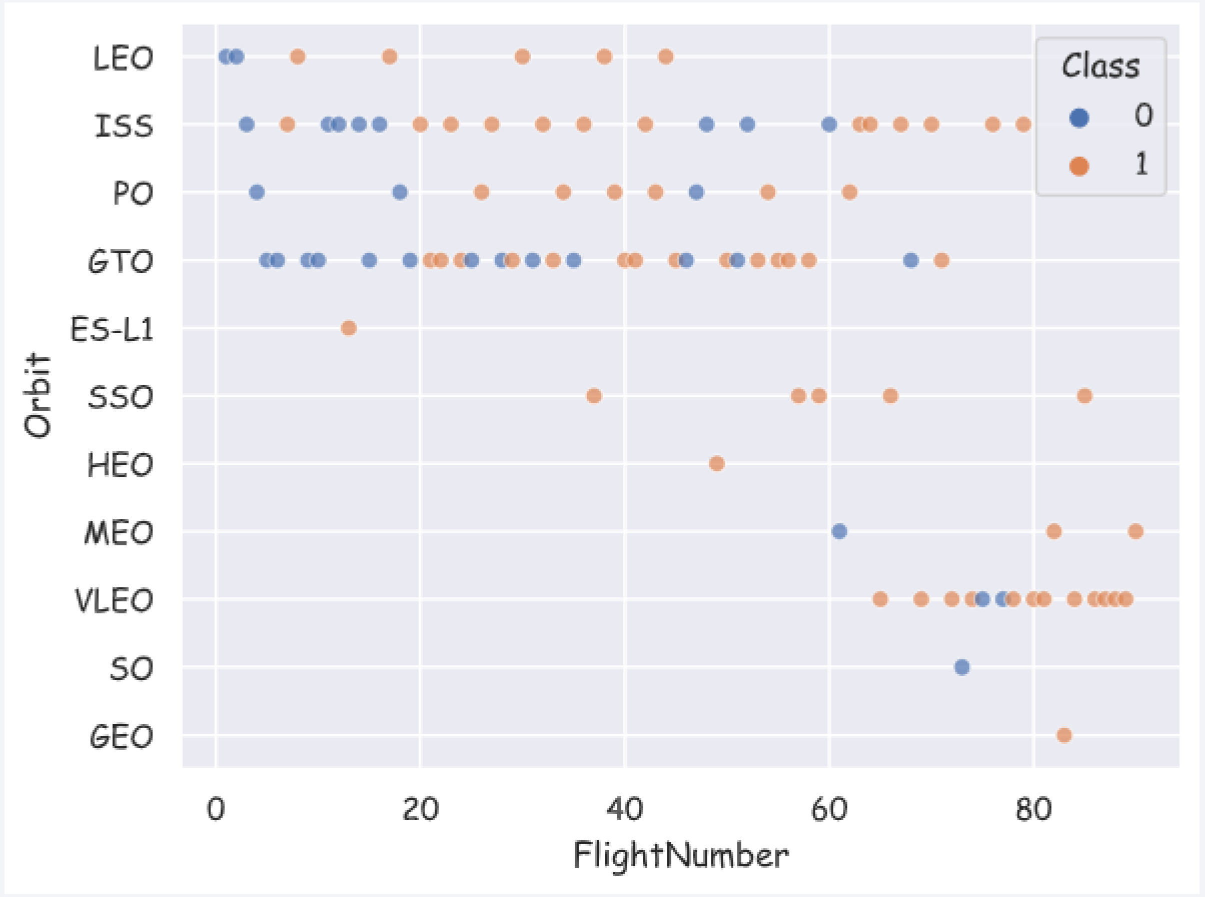
We can see clearly from the chart the launch sites and their distribution of payload masses for each launch. CCAFS SLC 40 and KSC LC 39A both had flights with payload mass above 15000kg

Success Rate vs. Orbit Type



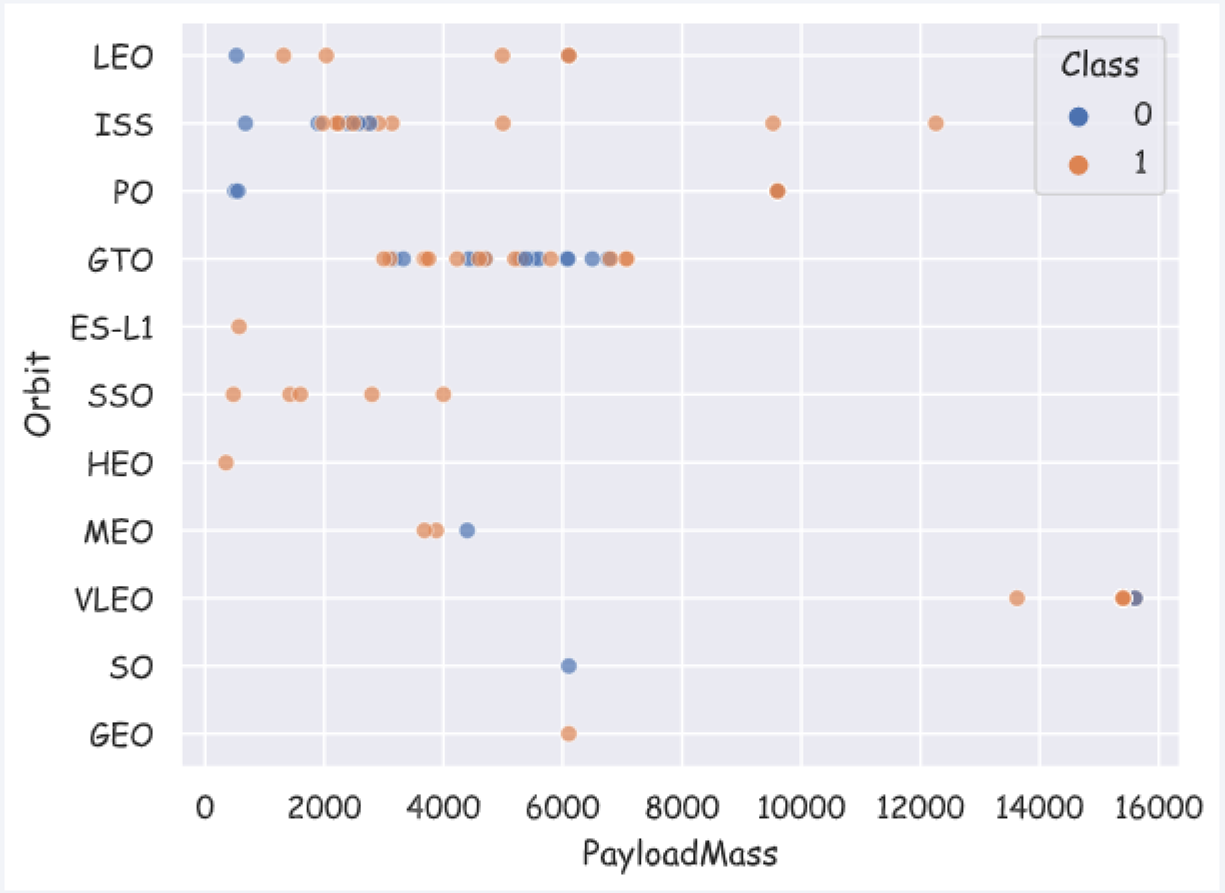
This chart shows the success rate for each orbit. Orbits ES-L1, GEO, HEO, and SSO has the highest success rates among all orbits.

Flight Number vs. Orbit Type



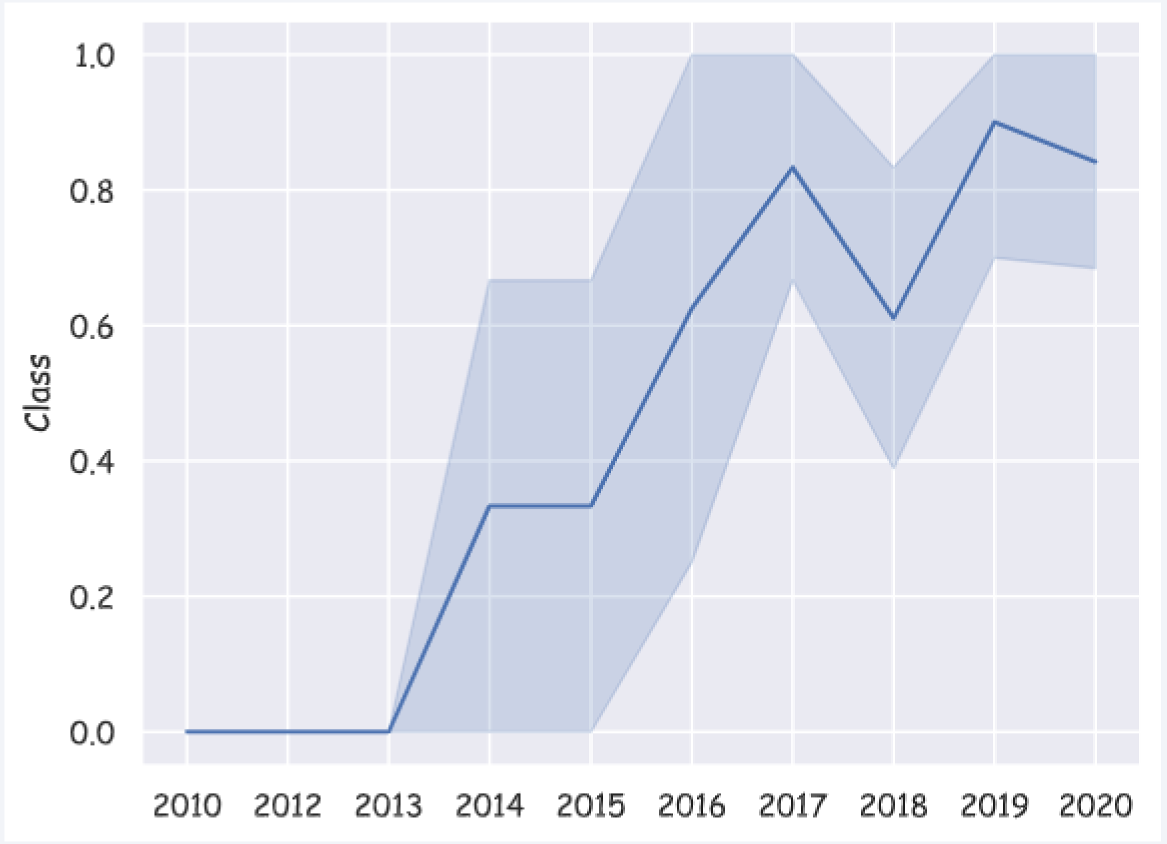
The chart here shows the success rate for each orbit. You would notice that the more the number of flights for each orbit the lesser the success rate.

Payload vs. Orbit Type



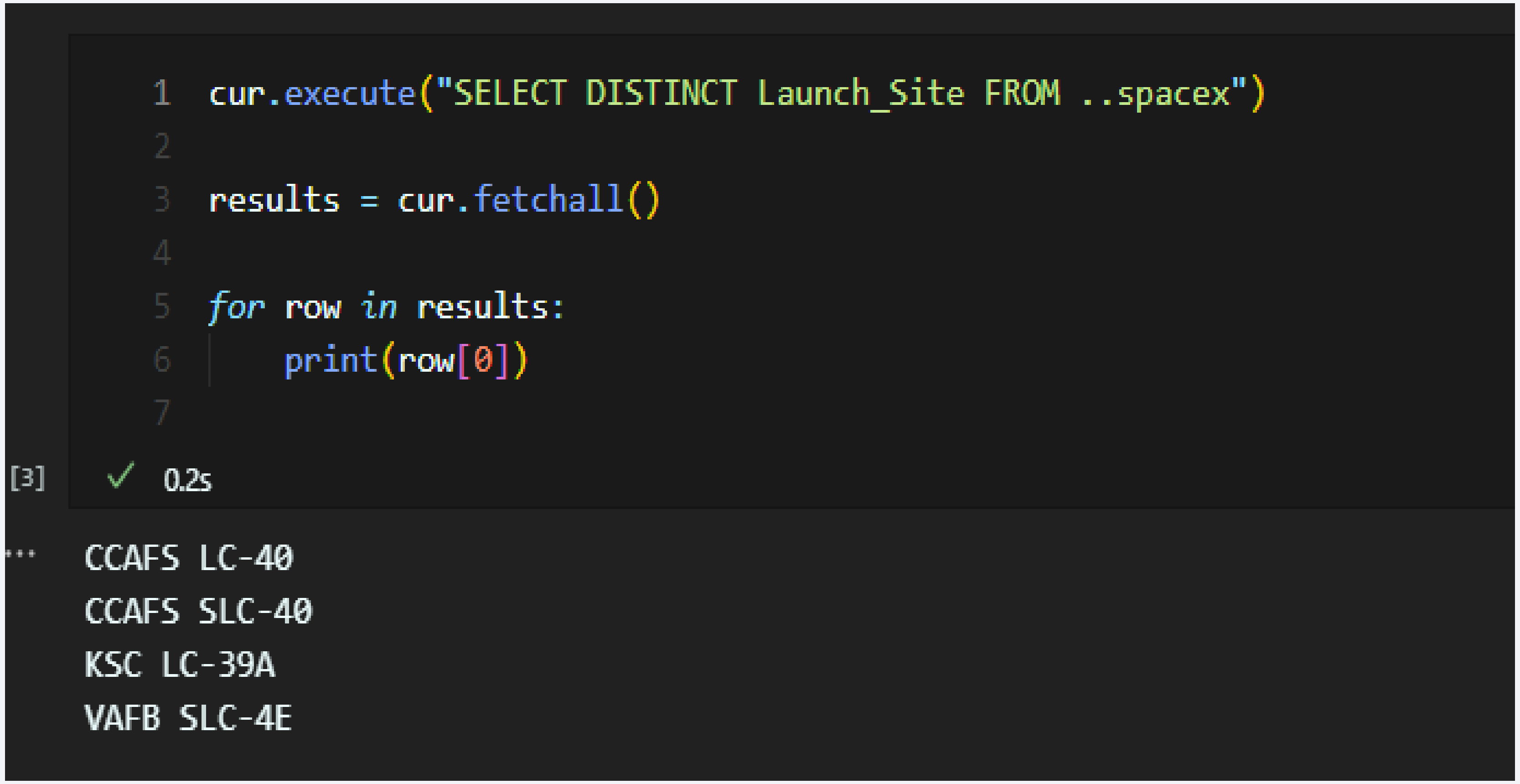
This chart shows the payload mass for each orbit type.

Launch Success Yearly Trend



This chart shows the trend of the success and the failure rates for each year. We can see a markedly increase of the success rate from 2013 and a sharp drop between 2017 and 2018.

All Launch Site Names

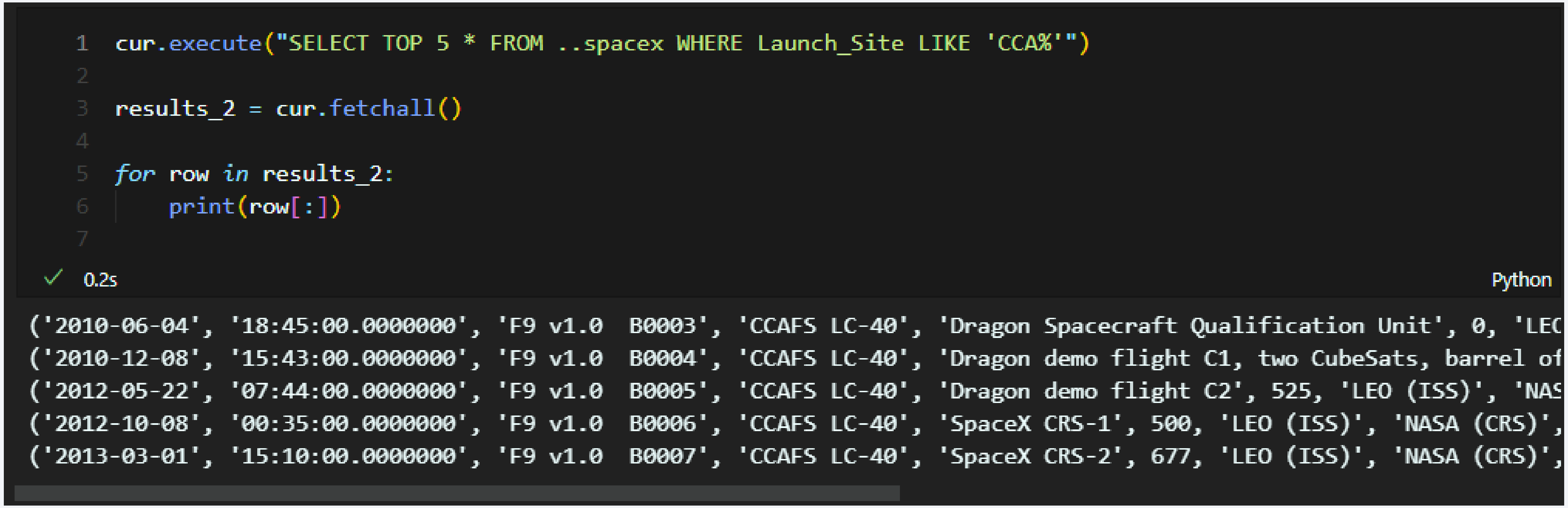


```
cur.execute("SELECT DISTINCT Launch_Site FROM ..spacex")

results = cur.fetchall()

for row in results:
    print(row[0])

[3]  ✓ 0.2s
```

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

The query shows the different sites names. The distinct sites names were gotten by using the ‘DISTINCT’ keyword to show the unique launch sites.

Launch Site Names Begin with 'CCA'



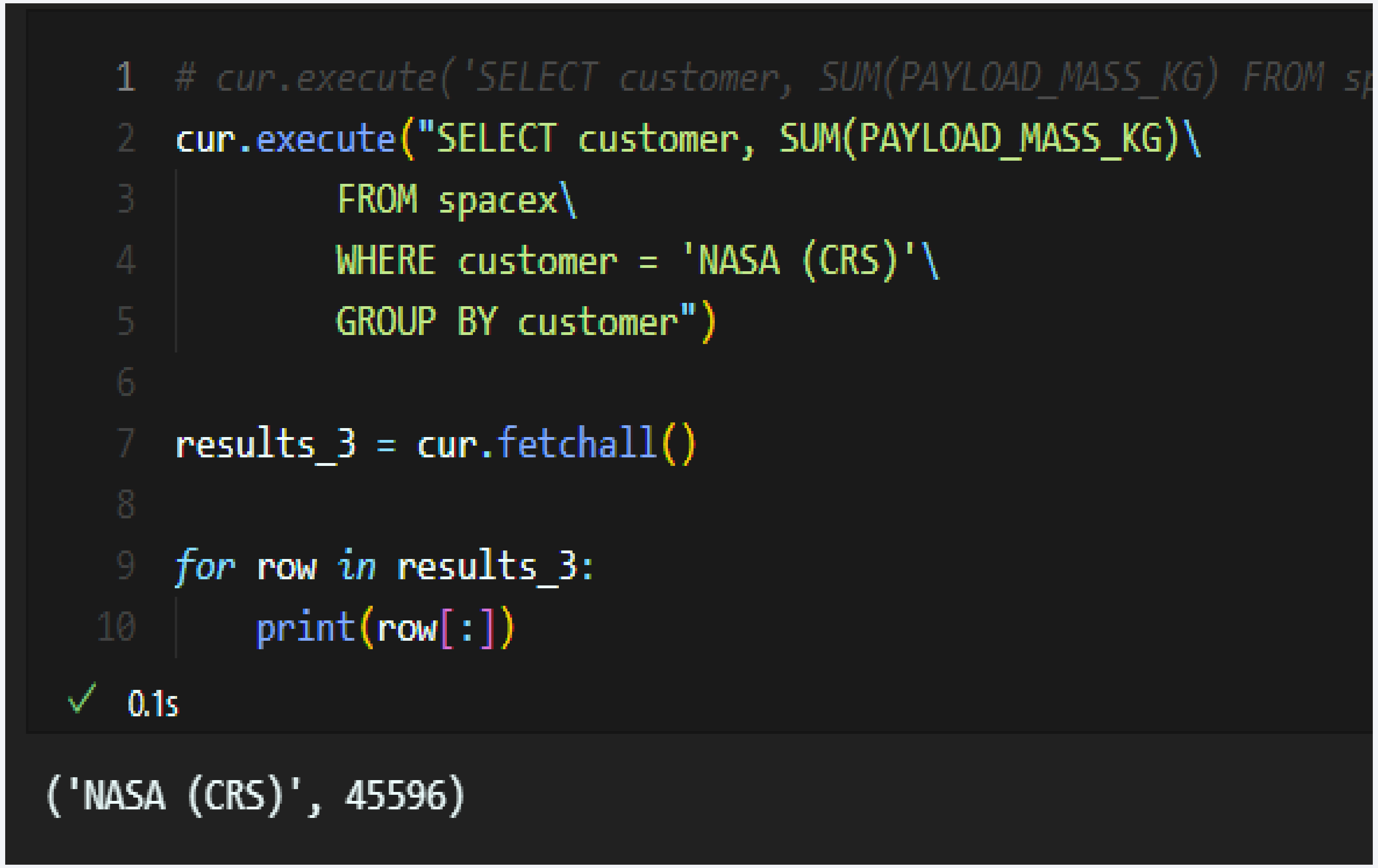
```
cur.execute("SELECT TOP 5 * FROM ..spacex WHERE Launch_Site LIKE 'CCA%'")

results_2 = cur.fetchall()

for row in results_2:
    print(row[:])

```

✓ 0.2s Python

```
('2010-06-04', '18:45:00.0000000', 'F9 v1.0  B0003', 'CCAFS LC-40', 'Dragon Spacecraft Qualification Unit', 0, 'LEO
('2010-12-08', '15:43:00.0000000', 'F9 v1.0  B0004', 'CCAFS LC-40', 'Dragon demo flight C1, two CubeSats, barrel of
('2012-05-22', '07:44:00.0000000', 'F9 v1.0  B0005', 'CCAFS LC-40', 'Dragon demo flight C2', 525, 'LEO (ISS)', 'NAS
('2012-10-08', '00:35:00.0000000', 'F9 v1.0  B0006', 'CCAFS LC-40', 'SpaceX CRS-1', 500, 'LEO (ISS)', 'NASA (CRS)',
('2013-03-01', '15:10:00.0000000', 'F9 v1.0  B0007', 'CCAFS LC-40', 'SpaceX CRS-2', 677, 'LEO (ISS)', 'NASA (CRS)',
```

Here, you can see the top 5 records of the launch sites that starts with ‘CCA’. I used the ‘TOP’, ‘WHERE’, and ‘LIKE’ keywords to get this information.

Total Payload Mass



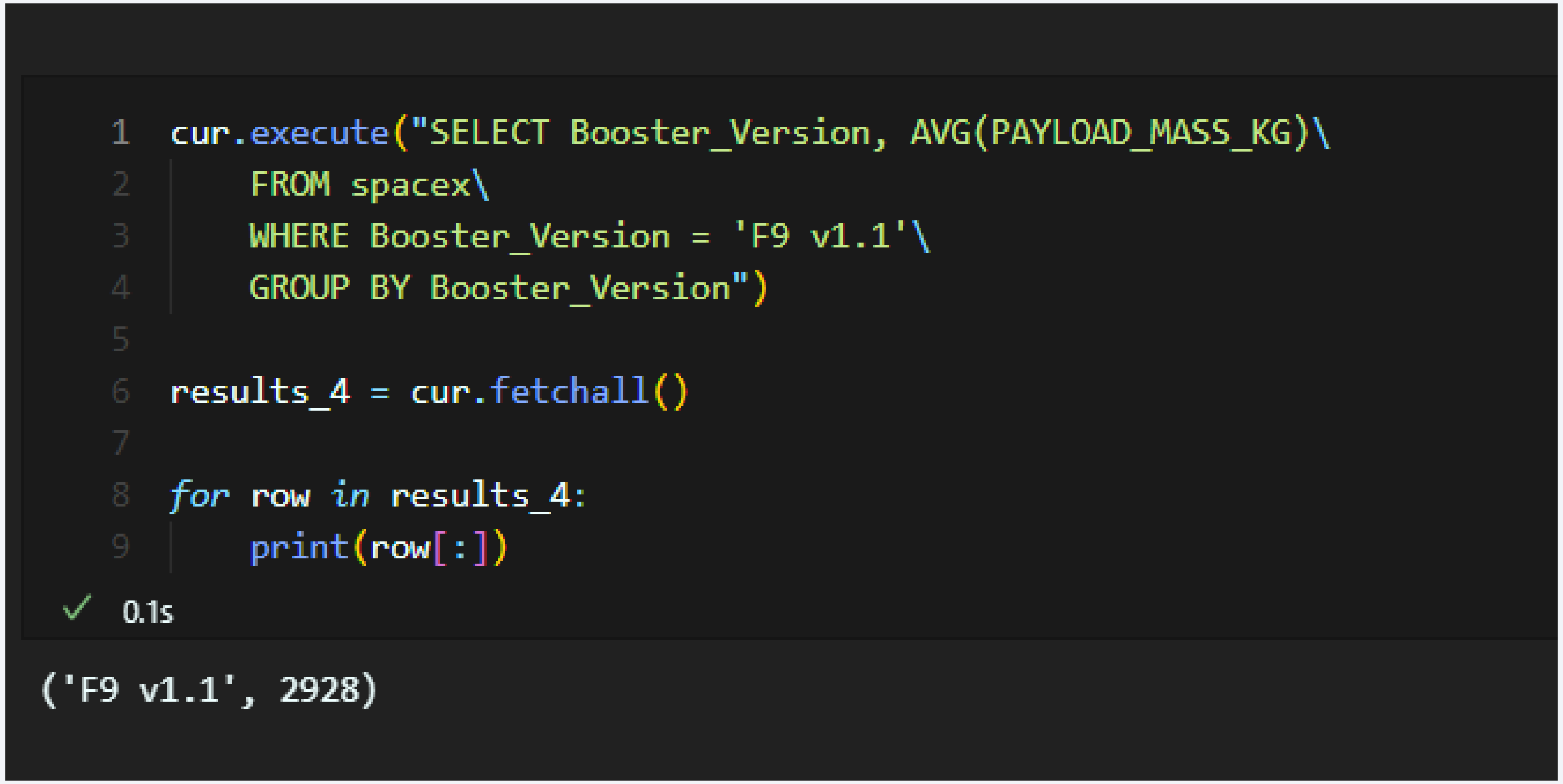
```
# cur.execute('SELECT customer, SUM(PAYLOAD_MASS_KG) FROM sp
cur.execute("SELECT customer, SUM(PAYLOAD_MASS_KG)\
        FROM spacex\
        WHERE customer = 'NASA (CRS)'\
        GROUP BY customer")

results_3 = cur.fetchall()

for row in results_3:
    print(row[:])
```

✓ 0.1s

```
('NASA (CRS)', 45596)
```

This screenshot shows the total payload mass carried by boosters launched by NASA (CRS). The total payload mass is 45,596kg.

Average Payload Mass by F9 v1.1



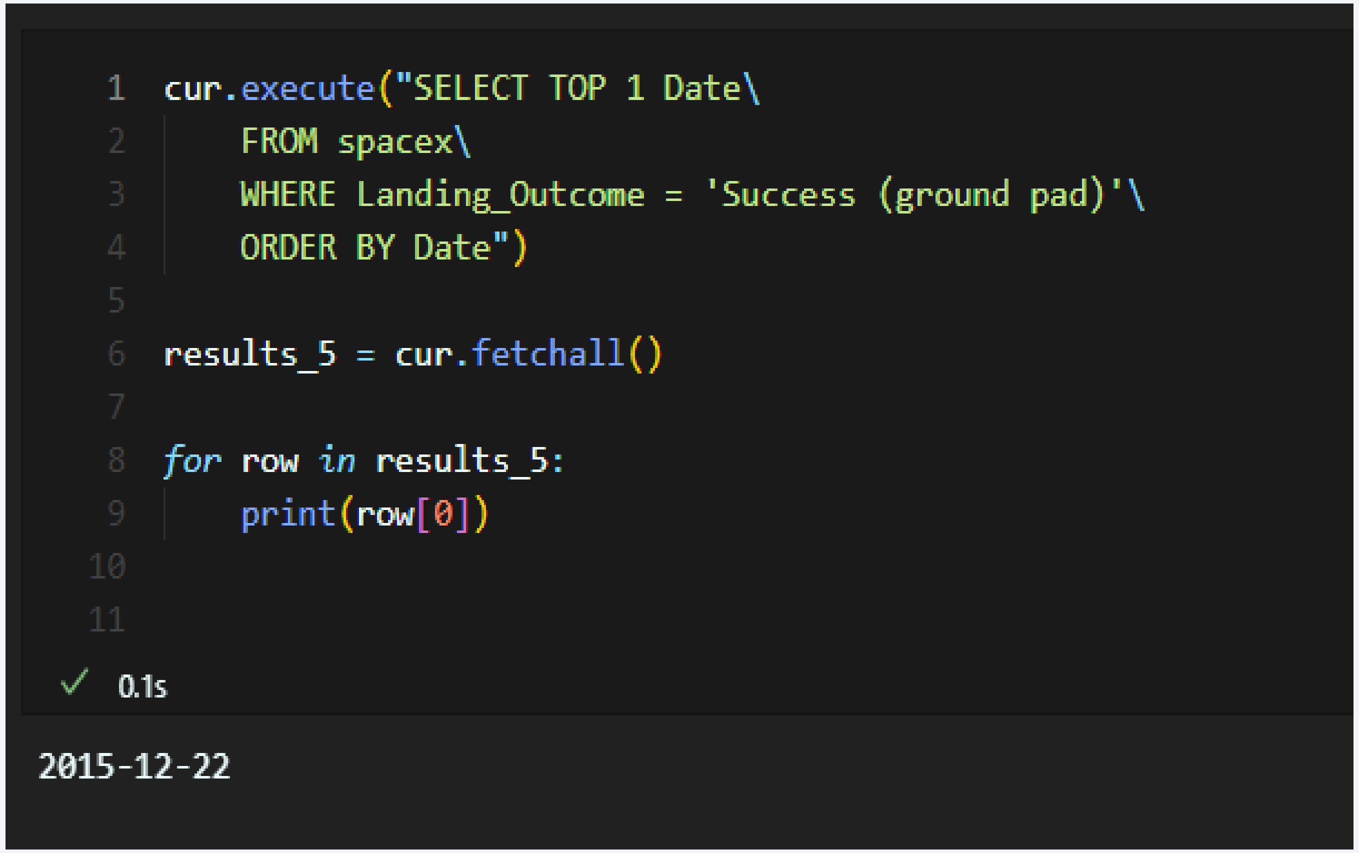
```
cur.execute("SELECT Booster_Version, AVG(PAYLOAD_MASS_KG)\
    FROM spacex\
    WHERE Booster_Version = 'F9 v1.1'\
    GROUP BY Booster_Version")

results_4 = cur.fetchall()

for row in results_4:
    print(row[:])
```

✓ 0.1s

```
('F9 v1.1', 2928)
```

This shows the average payload mass by F9 v1.1 booster which is 2928.

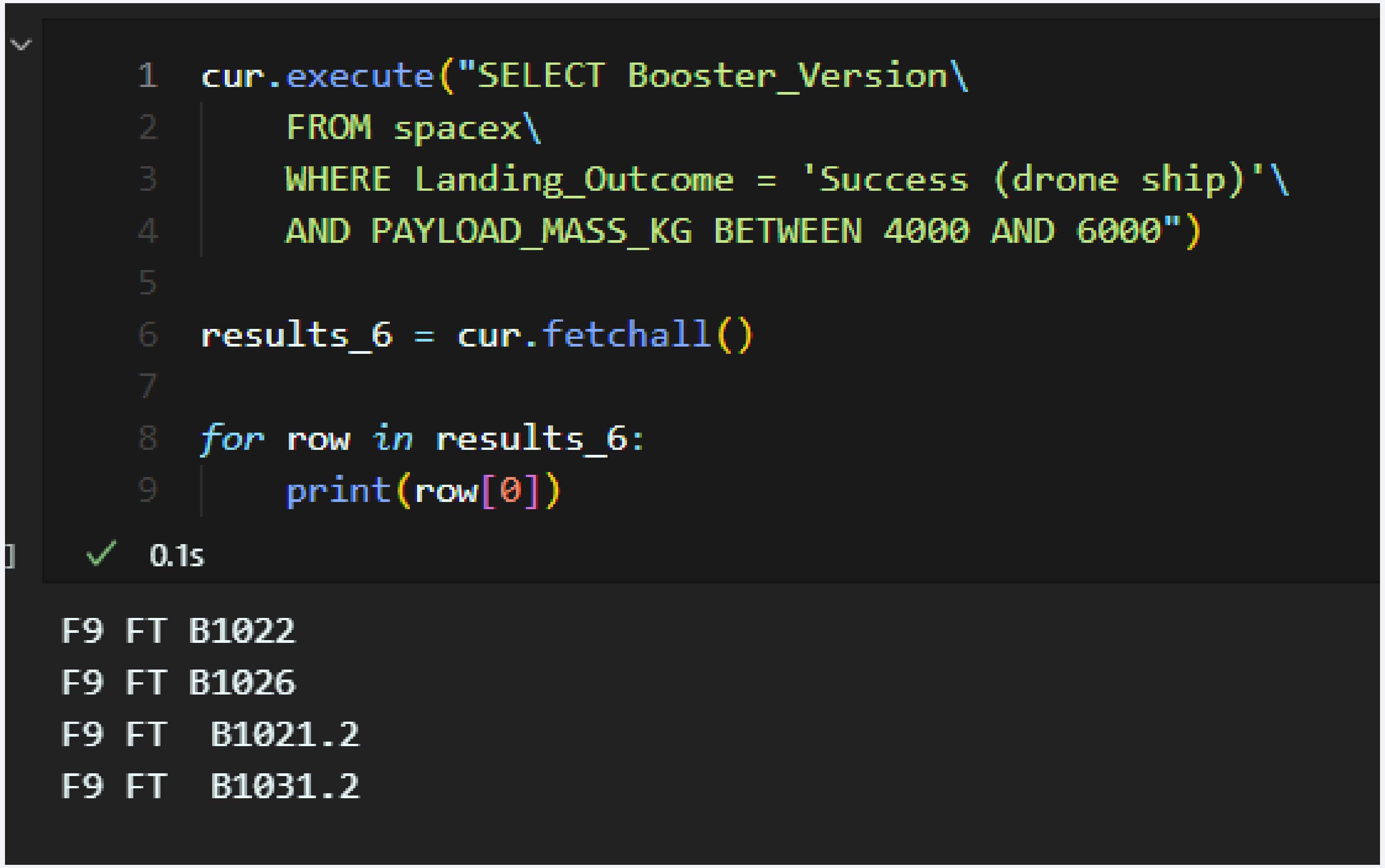
First Successful Ground Landing Date



```
cur.execute("SELECT TOP 1 Date\
    FROM spacex\
    WHERE Landing_Outcome = 'Success (ground pad)'\
    ORDER BY Date")

results_5 = cur.fetchall()

for row in results_5:
    print(row[0])


✓ 0.1s

2015-12-22
```

This shows the data for the first successful landing on the ground pad. The date is shown to be 2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000



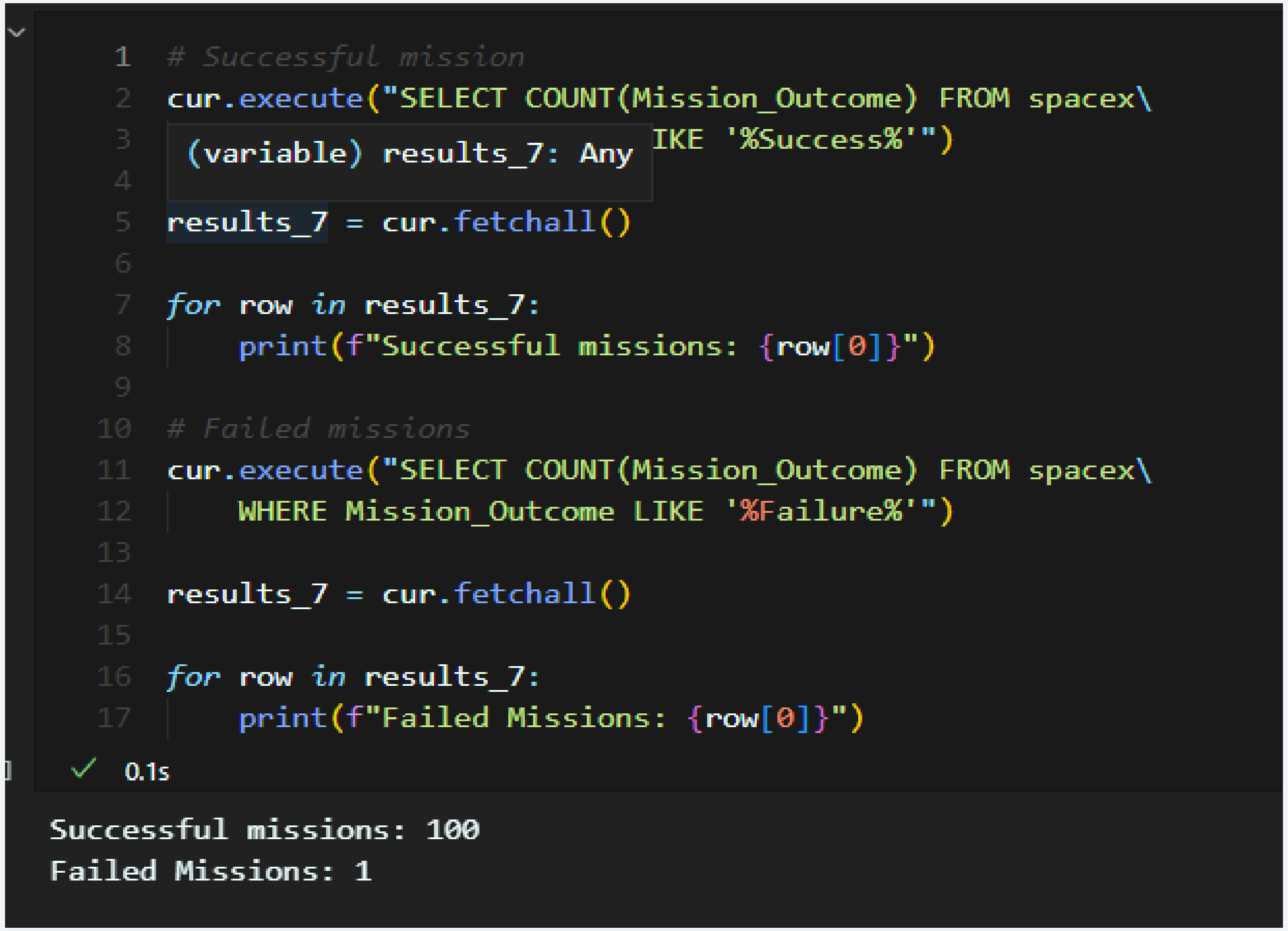
```
cur.execute("SELECT Booster_Version\
    FROM spacex\
    WHERE Landing_Outcome = 'Success (drone ship)'\
    AND PAYLOAD_MASS_KG BETWEEN 4000 AND 6000")

results_6 = cur.fetchall()

for row in results_6:
    print(row[0])
```

✓ 0.1s

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

This shows the boosters which have successfully landed on a drone ship with a payload mass between 4000 and 6000.

Total Number of Successful and Failure Mission Outcomes



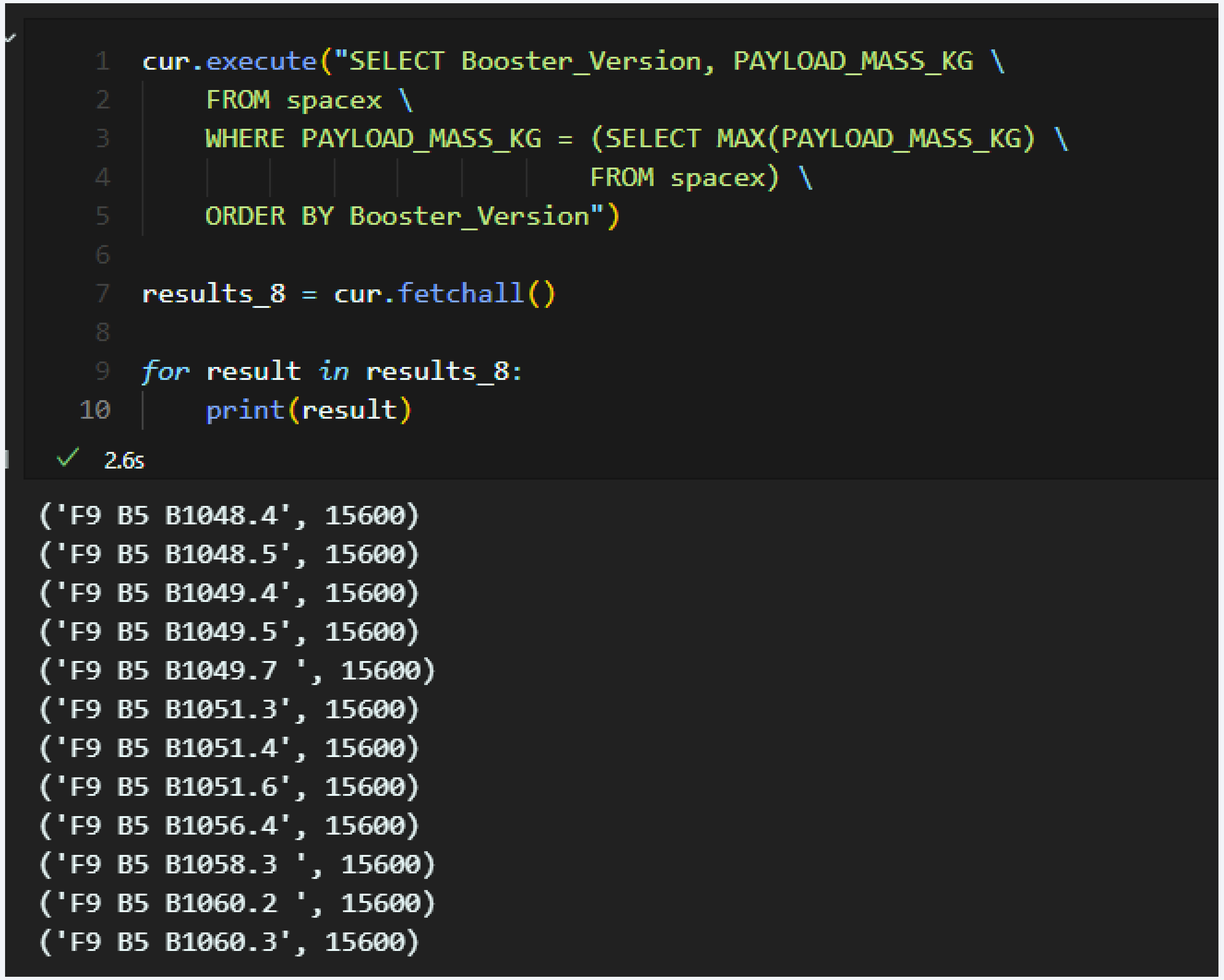
```
# Successful mission
cur.execute("SELECT COUNT(Mission_Outcome) FROM spacex\
 (variable) results_7: Any IKE '%Success%'")

results_7 = cur.fetchall()

for row in results_7:
    print(f"Successful missions: {row[0]}")

# Failed missions
cur.execute("SELECT COUNT(Mission_Outcome) FROM spacex\
    WHERE Mission_Outcome LIKE '%Failure%'")

results_7 = cur.fetchall()

for row in results_7:
    print(f"Failed Missions: {row[0]}")
```

✓ 0.1s

```
Successful missions: 100
Failed Missions: 1
```

This shows the number of successful and failed mission outcomes. Here, we have 100 successful missions and just 1 failed missions.

Boosters Carried Maximum Payload



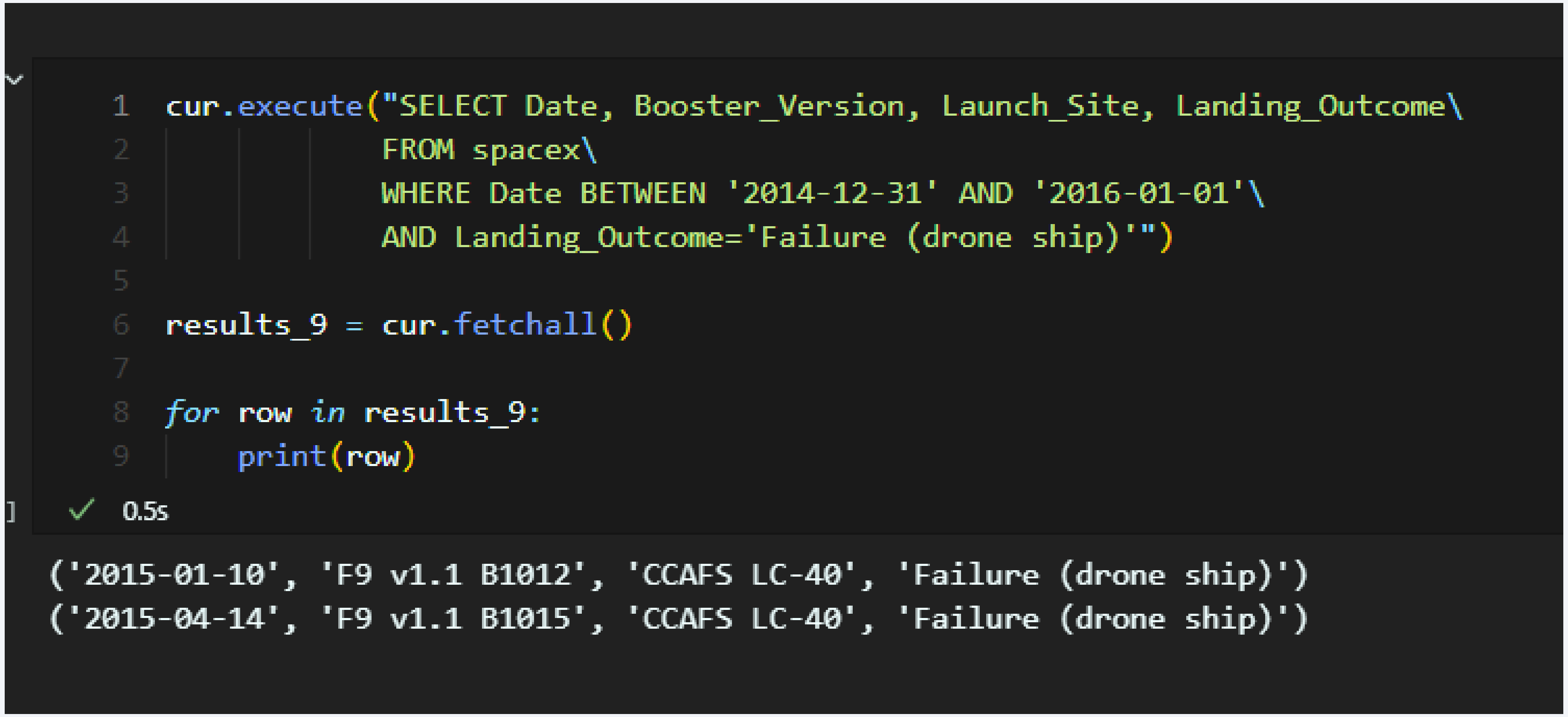
```
cur.execute("SELECT Booster_Version, PAYLOAD_MASS_KG \
    FROM spacex \
    WHERE PAYLOAD_MASS_KG = (SELECT MAX(PAYLOAD_MASS_KG) \
                            FROM spacex) \
    ORDER BY Booster_Version")

results_8 = cur.fetchall()

for result in results_8:
    print(result)
```

✓ 2.6s

```
('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)
```

This shows the boosters that carried the maximum payload. We have 12 of them listed.

2015 Launch Records



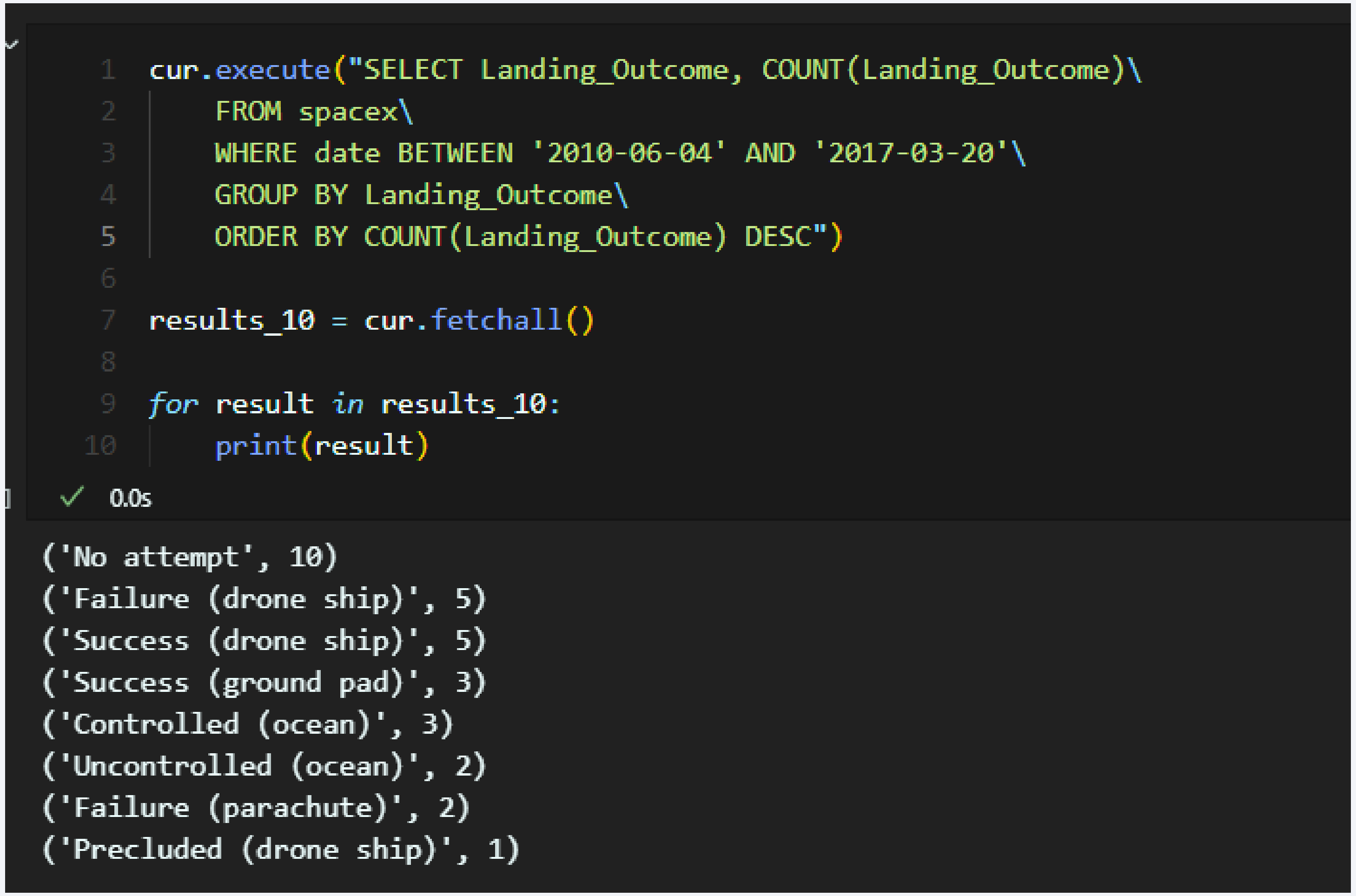
```
cur.execute("SELECT Date, Booster_Version, Launch_Site, Landing_Outcome\
            FROM spacex\
            WHERE Date BETWEEN '2014-12-31' AND '2016-01-01'\
            AND Landing_Outcome='Failure (drone ship)'")

results_9 = cur.fetchall()

for row in results_9:
    print(row)
```

✓ 0.5s

```
('2015-01-10', 'F9 v1.1 B1012', 'CCAFS LC-40', 'Failure (drone ship)')
('2015-04-14', 'F9 v1.1 B1015', 'CCAFS LC-40', 'Failure (drone ship)')
```

This shows the launch records for 2015. Here, we have just 2 records from 2015.

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



```
cur.execute("SELECT Landing_Outcome, COUNT(Landing_Outcome)\
    FROM spacex\
    WHERE date BETWEEN '2010-06-04' AND '2017-03-20'\
    GROUP BY Landing_Outcome\
    ORDER BY COUNT(Landing_Outcome) DESC")

results_10 = cur.fetchall()

for result in results_10:
    print(result)
```

✓ 0.0s

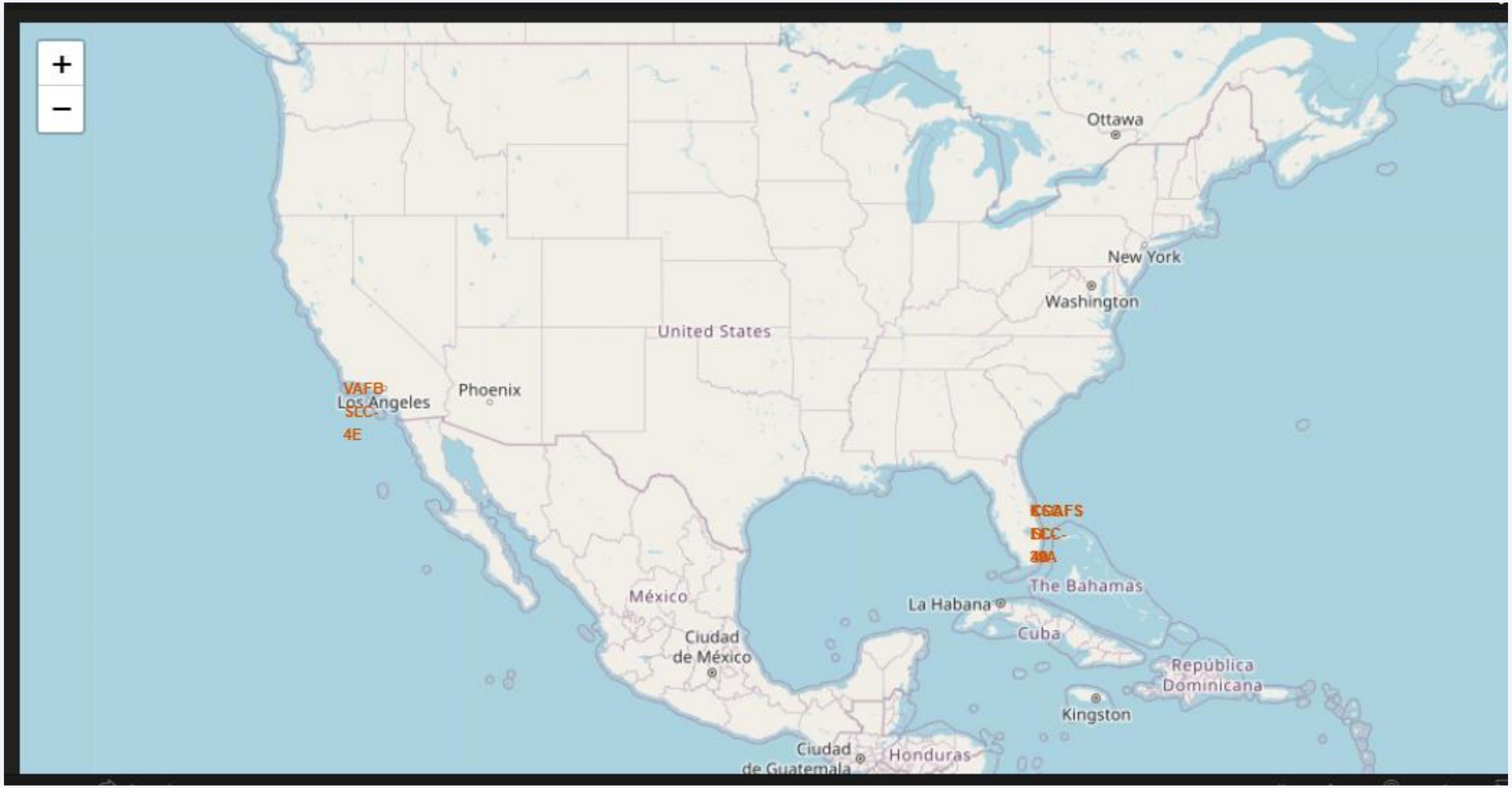
```
('No attempt', 10)
('Failure (drone ship)', 5)
('Success (drone ship)', 5)
('Success (ground pad)', 3)
('Controlled (ocean)', 3)
('Uncontrolled (ocean)', 2)
('Failure (parachute)', 2)
('Precluded (drone ship)', 1)
```

This shows the count of successful landing outcomes between 04-06-2010 and 20-03-2017.

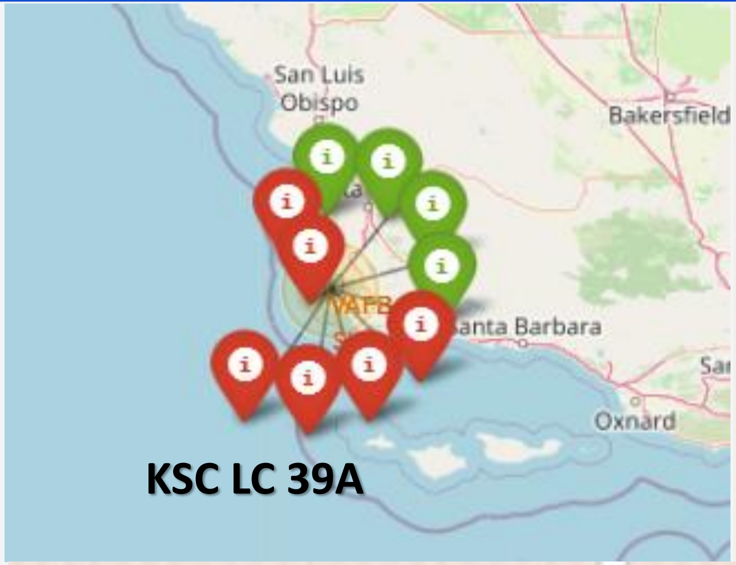
Section 3

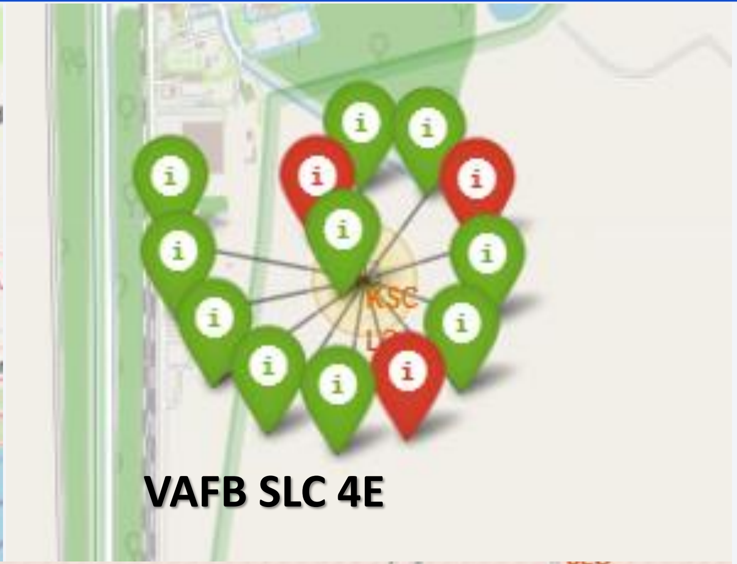
Launch Sites Proximities Analysis

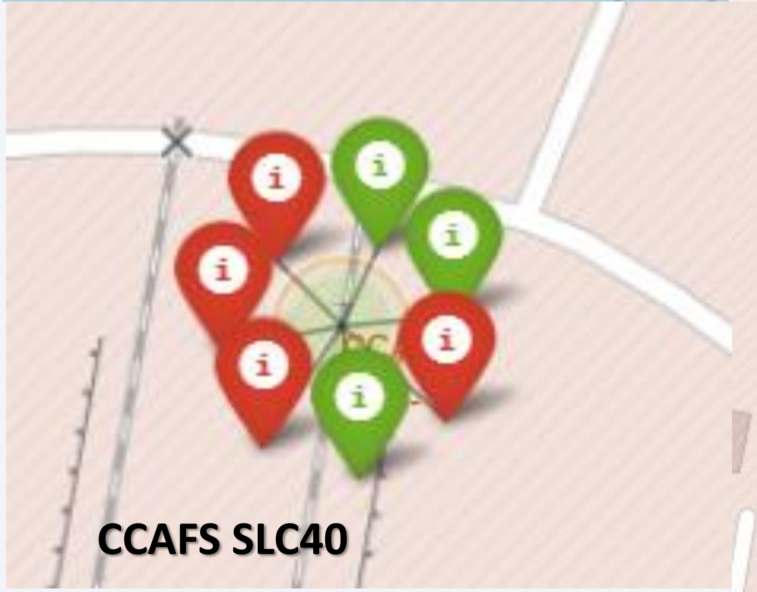
Launch Sites

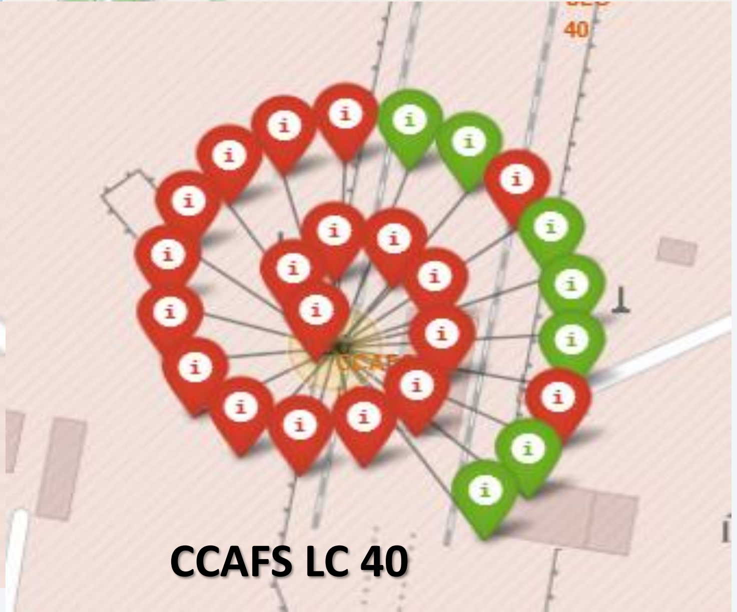


Successful and Failed Launches at different launch sites



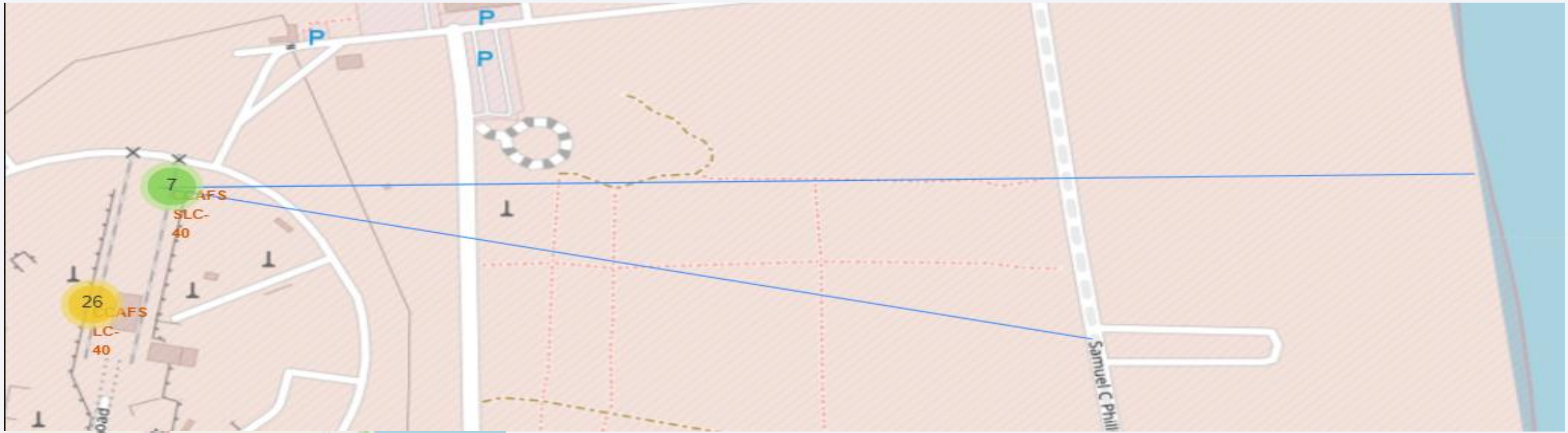


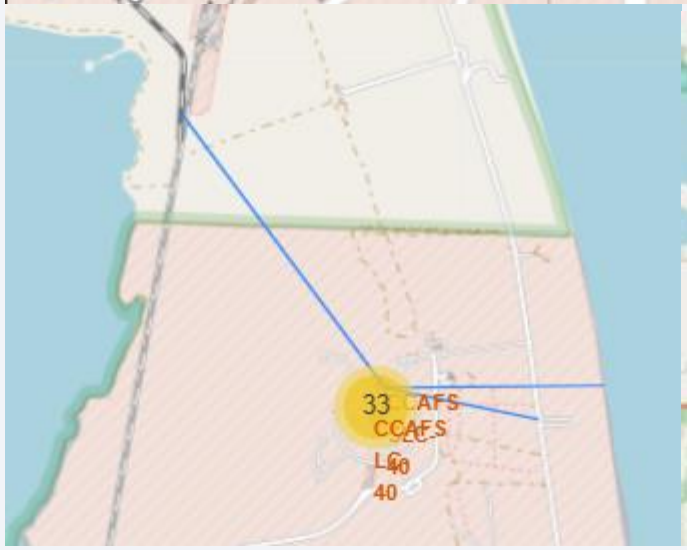


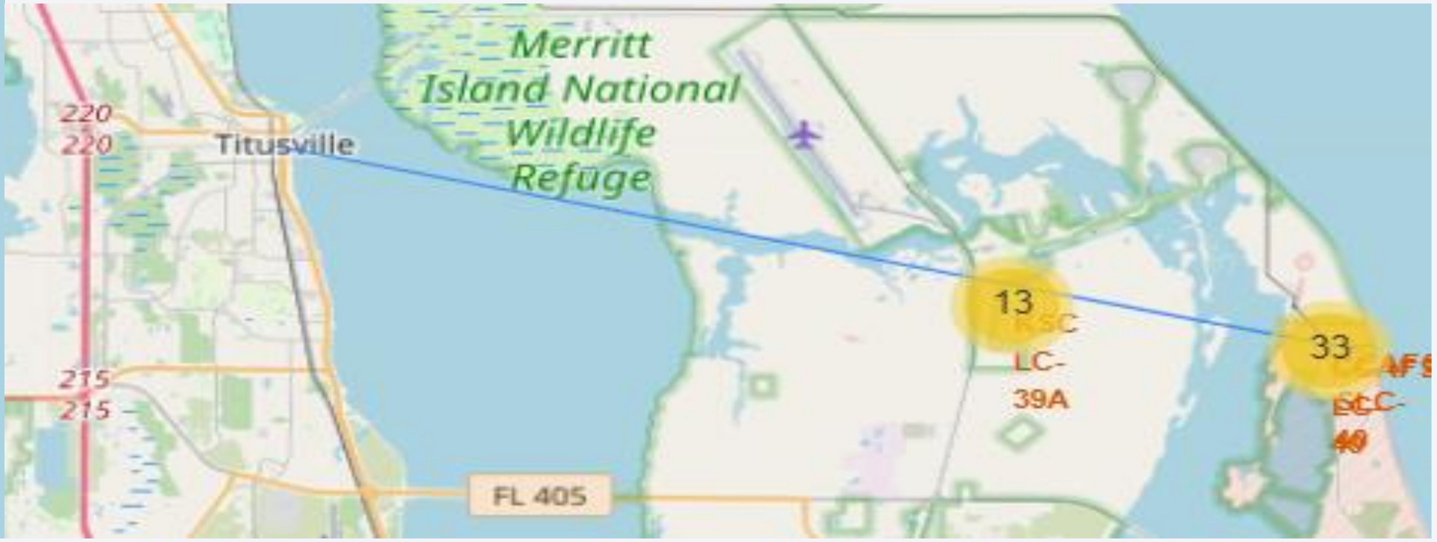


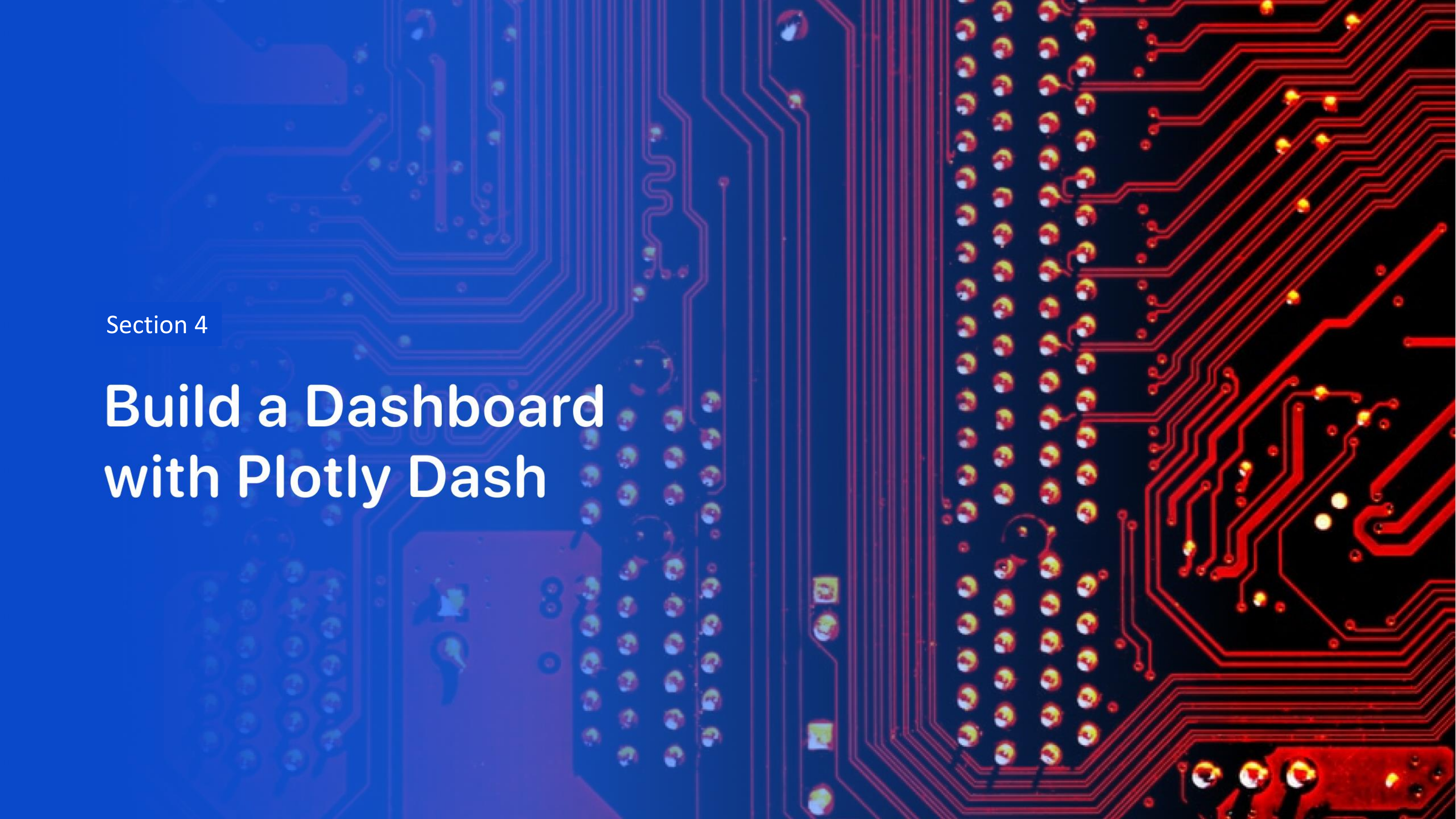
The following screenshots shows the different launch sites with successful launches in green and failed launches in red.

Launch Sites to Markers distance





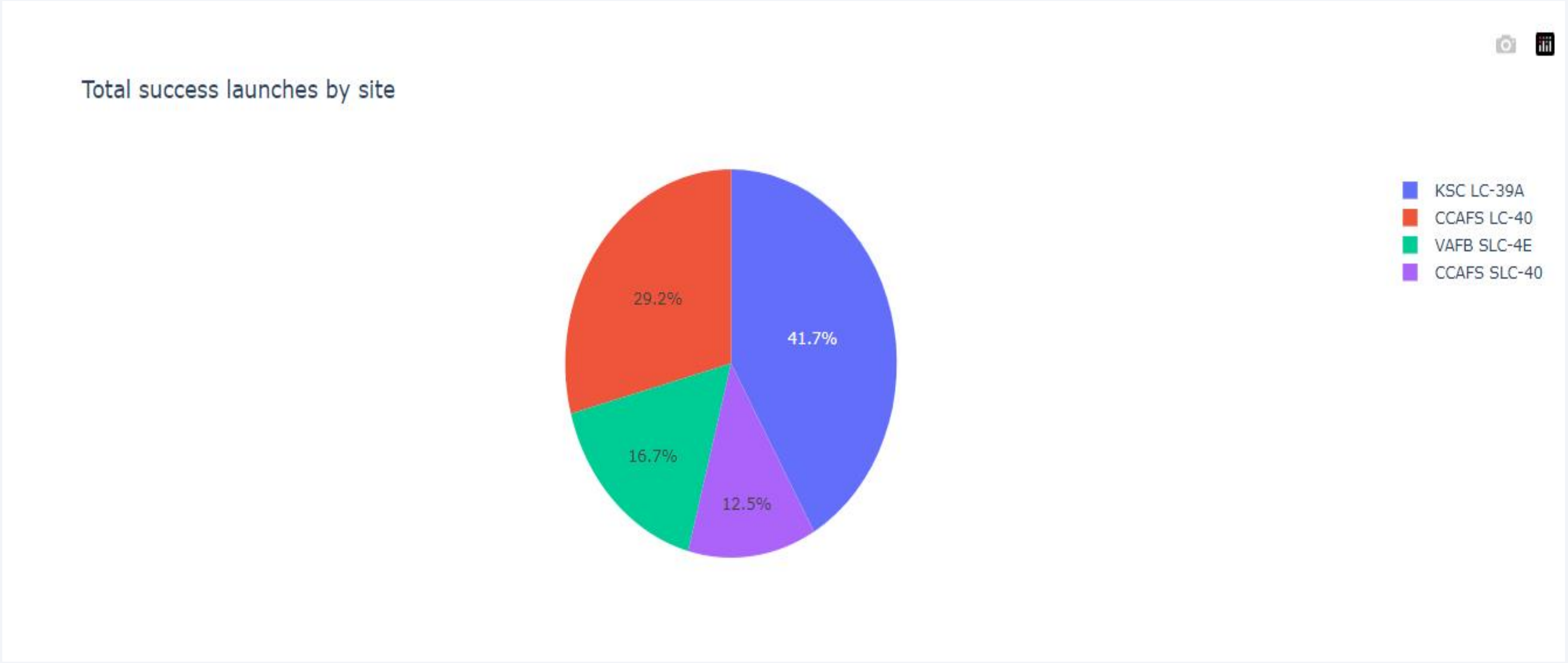




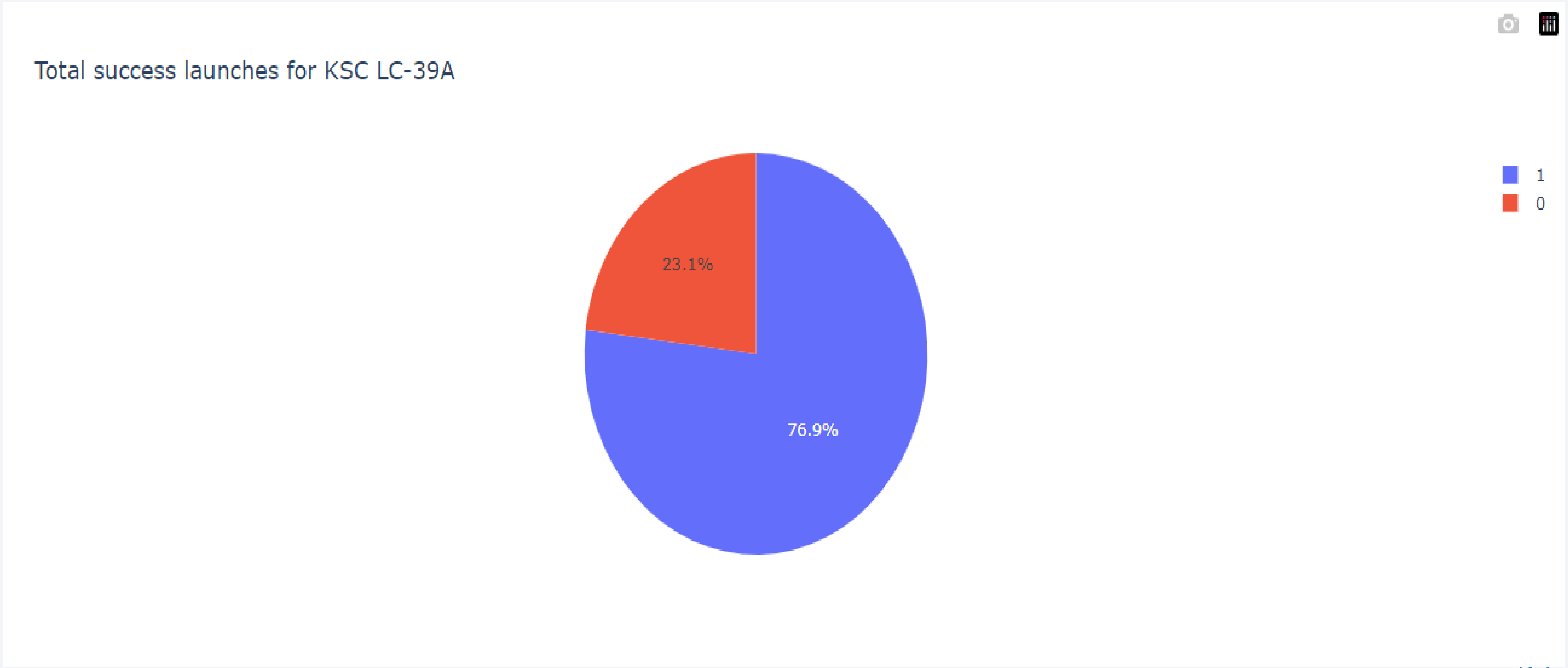
Section 4

Build a Dashboard with Plotly Dash

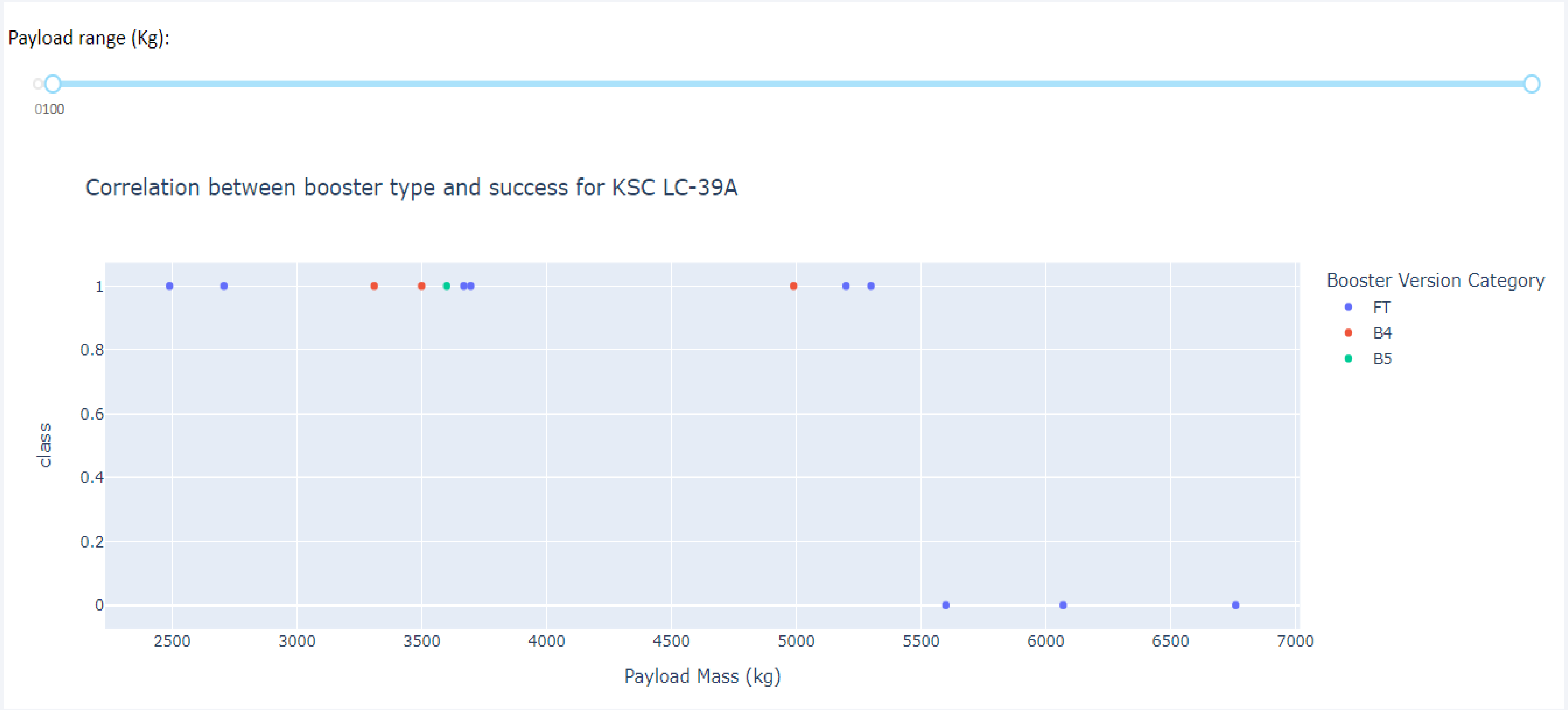
Total success launches by sites



Success ratio for KSC LC-39A



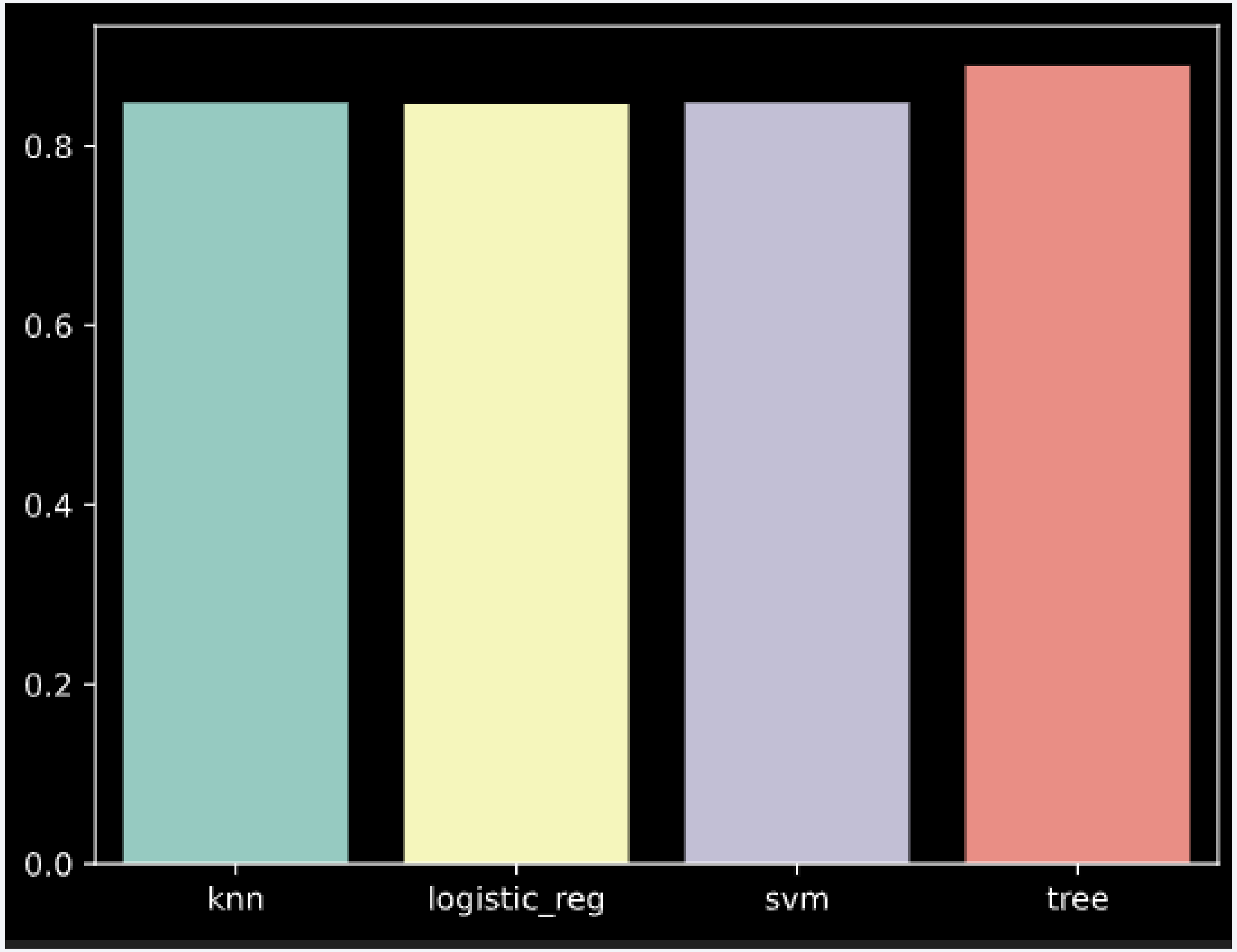
Payload vs Launch outcome scatter plot



Section 5

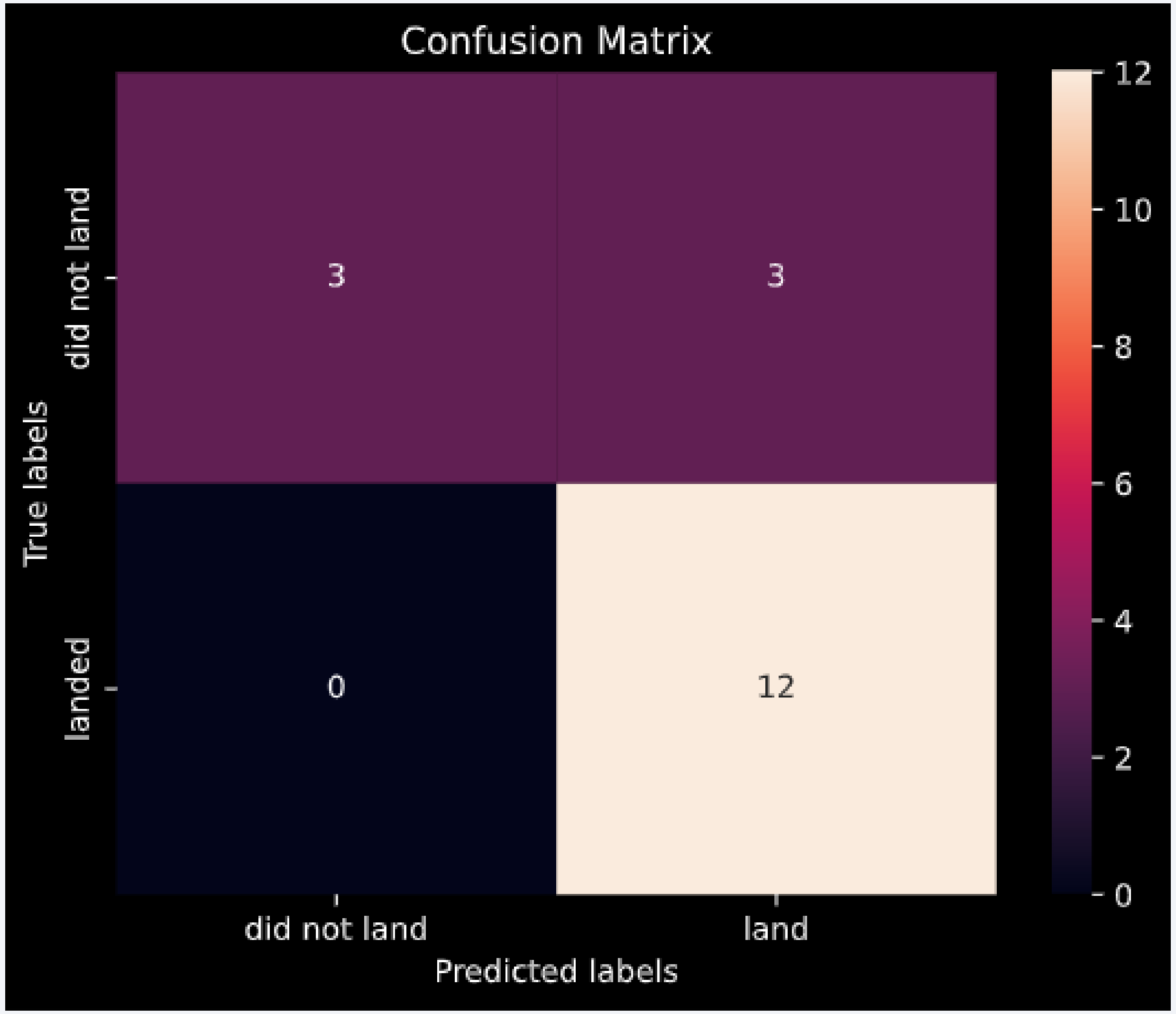
Predictive Analysis (Classification)

Classification Accuracy



From the chart, you can see that out of all the classification algorithms, the decision tree algorithm had the best accuracy score.

Confusion Matrix



The confusion matrix was plotted for the best performing model (Decision tree classifier) showing the True Positives, False positives, True Negatives and False Negatives.
True Positives: 12
True Negatives: 3
False Positives: 3
False Negatives: 0

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!