IBM Developer
SKILLS NETWORK

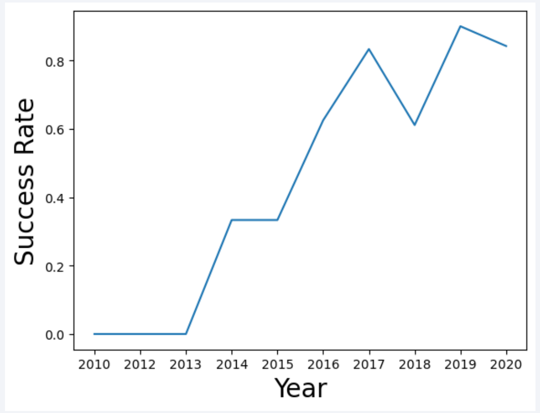
Winning Space Race with Data Science

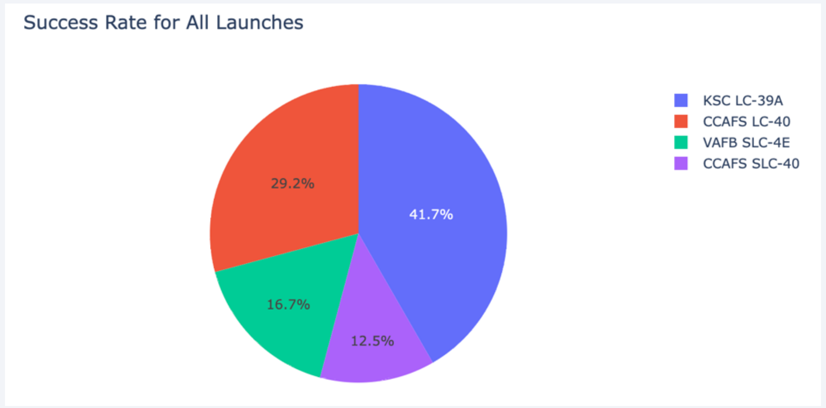
Jarrod Eisma
January 6, 2025

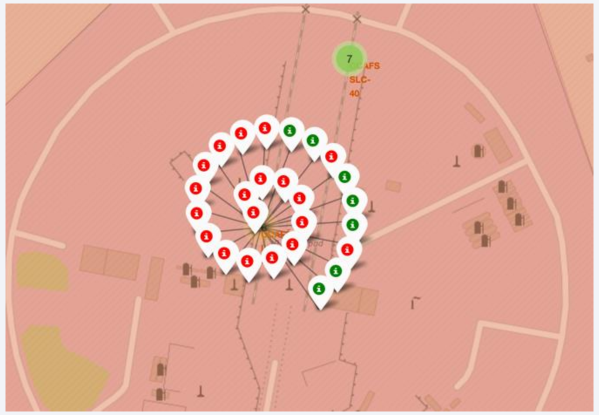
Outline

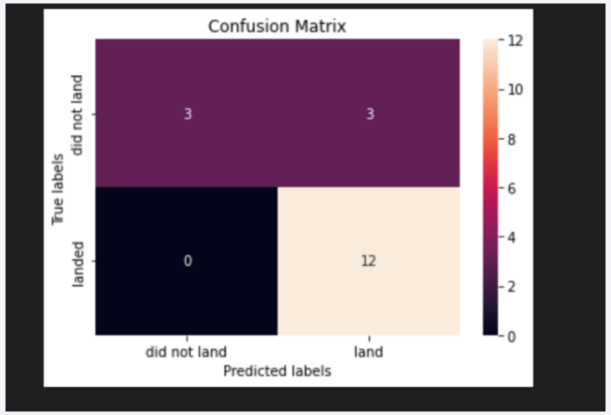
Executive Summary

- Summary of Methodologies
  - Data Collection
  - Data Wrangling
  - Exploratory Data Analysis
  - Interactive Visual Analytics
  - Predictive Analytics (Classification)
- Summary of Results
  - Exploratory Data Analysis Results
  - Geospatial Analytics
  - Interactive Dashboard
  - Predictive Analysis Results









Introduction

- 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 if an alternate company wants to bid against SpaceX for a rocket launch.
- The goal of this project is to predict if the Falcon 9 first stage will land successfully, through a series of data collection, cleaning, visualization, and analysis.

Section 1

Methodology

Methodology Overview

Executive Summary

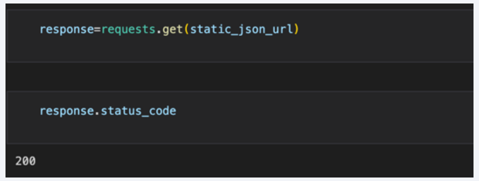
- Data Collection
  - GET requests to SpaceX REST API
  - Web scraping
- Data Wrangling
  - Removing NaN values
  - Determining number of launches at each site, occurrences of each orbit, and mission outcomes
  - Creating landing outcome variable
- Exploratory Data Analysis
  - Using SQL to manipulate SpaceX dataset
  - Using Matplotlib to visualize relationships between variables of interest
- Interactive Visual Analytics
  - Geospatial analytics using Folium
  - Interactive dashboard using Plotly Dash
- Predictive Analysis
  - Using Scikit-Learn to train machine learning models for classification

Data Collection – SpaceX API

1. Make a GET response to the SpaceX REST API
2. Convert the response to a Pandas data frame
3. Clean the data
4. Filter data to only include Falcon 9 launches, reset FlightNumber, replace NaNs in PayloadMass column with mean value

https://github.com/jjeisma/Applied_Data_Science_Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

1

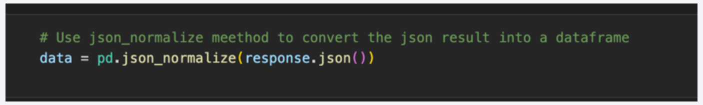


```
response=requests.get(static_json_url)
```

```
response.status_code
```

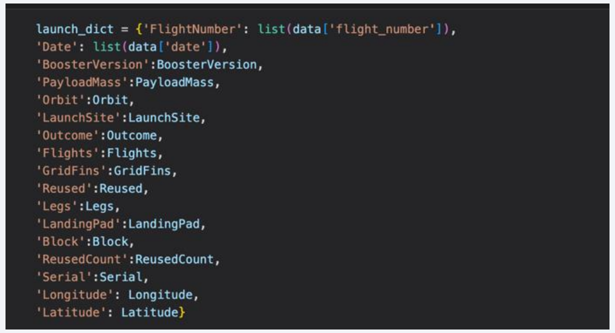
```
200
```

2

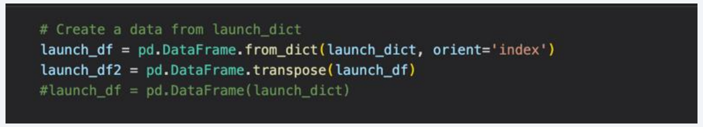


```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

3

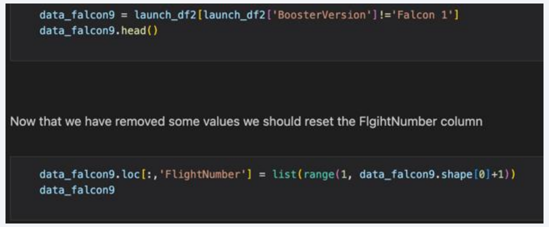


```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```



```
# Create a data from launch_dict
launch_df = pd.DataFrame.from_dict(launch_dict, orient='index')
launch_df2 = pd.DataFrame.transpose(launch_df)
#launch_df = pd.DataFrame(launch_dict)
```

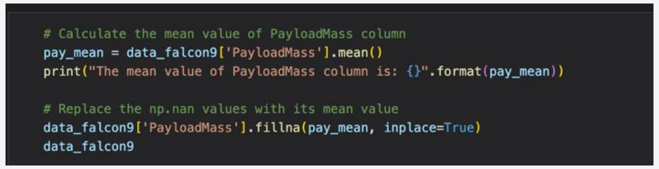
4



```
data_falcon9 = launch_df2[launch_df2['BoosterVersion']!='Falcon 1']
data_falcon9.head()
```

Now that we have removed some values we should reset the FlgihtNumber column

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```



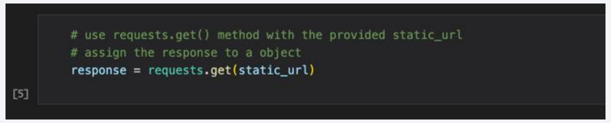
```
# Calculate the mean value of PayloadMass column
pay_mean = data_falcon9['PayloadMass'].mean()
print("The mean value of PayloadMass column is: {}".format(pay_mean))

# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].fillna(pay_mean, inplace=True)
data_falcon9
```

Data Collection - Scraping

1. Request HTML
2. Create BeautifulSoup object from HTML response
3. Find all tables
4. Collect all column header names
5. Parse launch tables to fill dictionary of column names
6. Convert dictionary to Pandas dataframe

https://github.com/jjeisma/Applied_Data_Science_Capstone/blob/main/jupyter-labs-webscraping.ipynb

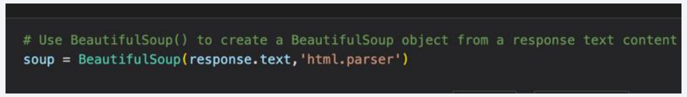
1



```python
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
```

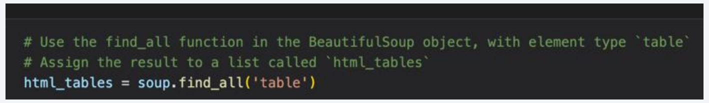
[5]

2



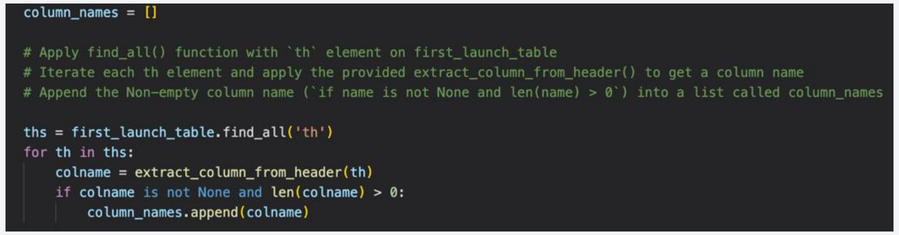
```python
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text,'html.parser')
```

3



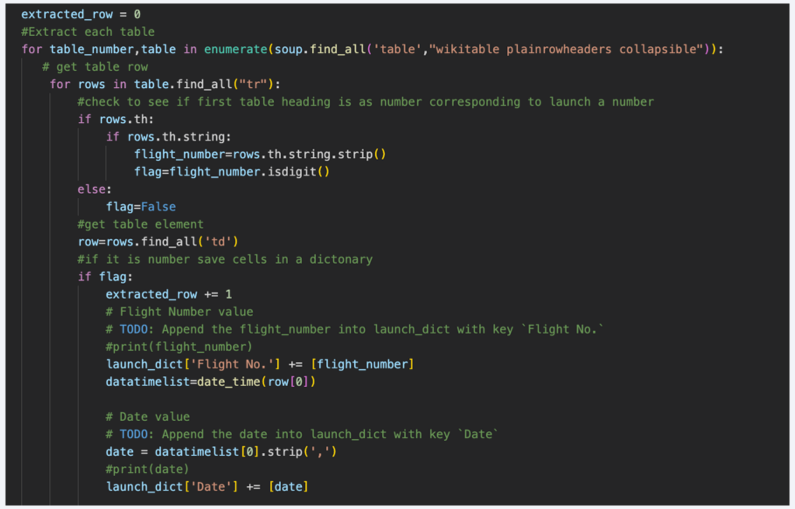
```python
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

4



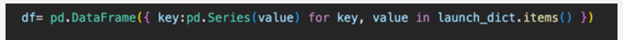
```python
column_names = []

# 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

ths = first_launch_table.find_all('th')
for th in ths:
    colname = extract_column_from_header(th)
    if colname is not None and len(colname) > 0:
        column_names.append(colname)
```

5



```python
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.'] += [flight_number]
            datatimelist=date_time(row[0])

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

6

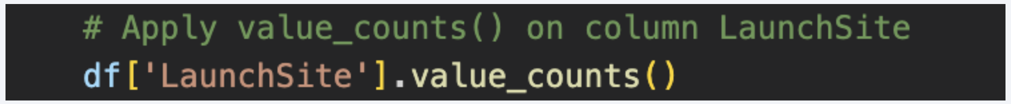


```python
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

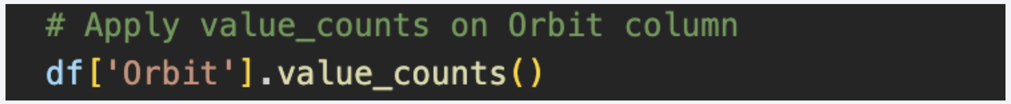
Data Wrangling

1. Determine number of launches on each site
2. Determine number of occurrences of each orbit
3. Determine success rate of each orbit type

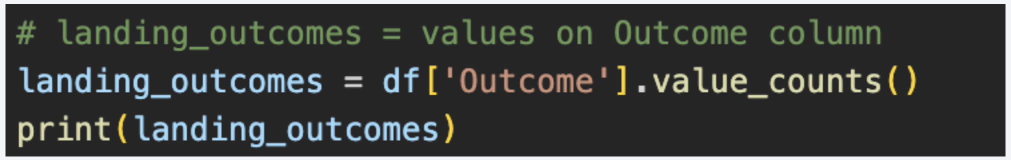
https://github.com/jjeisma/Applied_Data_Science_Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

1


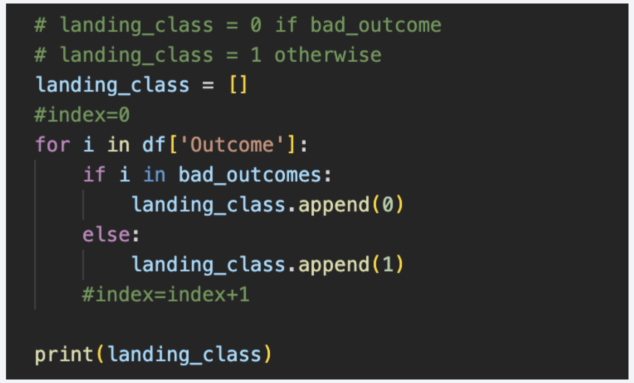
```python
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

2


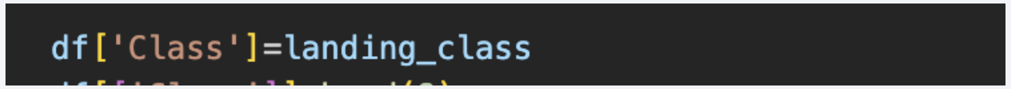
```python
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

3


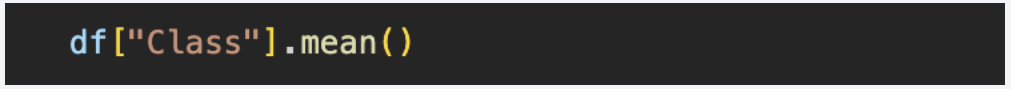
```python
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
print(landing_outcomes)
```



```python
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
#index=0
for i in df['Outcome']:
    if i in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
    #index=index+1

print(landing_class)
```



```python
df['Class']=landing_class
```



```python
df["Class"].mean()
```

EDA with Data Visualization

- Scatter charts: chosen to visualize relationships between numerical variables
  - Show relationships between flight number and launch site, payload and launch site, orbit type and flight number, and payload and orbit type
- Bar charts: chosen to compare numerical values within a categorical variable
  - Show comparison of success rates between orbit types
- Line charts: chosen to show change in numerical variable over time
  - Show relationship between success rate and time

https://github.com/jjeisma/Applied_Data_Science_Capstone/blob/main/edadataviz.ipynb

EDA with SQL

- Display names of unique launch sites
- Display 5 records where launch site begins with 'CCA'
- Display total payload mass carried by boosters launched by NASA
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was achieved
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster versions which have carried the maximum payload mass. Use a subquery
- 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.
- 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.

https://github.com/jjeisma/Applied_Data_Science_Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Mark all launch sites on a map with a folium.Circle and folium.Marker object
- Mark the success/failure class for each launch at each site using different colors and MarkerCluster objects
- Calculate the distances between a launch site and its nearest city, highway, and railway and mark the distance with a folium.Marker and folium.Polyline object

https://github.com/jjeisma/Applied_Data_Science_Capstone/blob/main/lab_jupyter_launch_site_location%20(1).ipynb

Build a Dashboard with Plotly Dash

- Pie charts (px.pie()) were added to the dashboard showing the portion of successful launches from site
  - Can be filtered to look at success rate for individual sites via. dcc.Dropdown() object
- Scatter plots (px.scatter()) were added to the dashboard showing the correlation between launch outcome and payload mass
  - Can be filtered by range of payload mass via RangeSlider() object and by booster version

https://github.com/jjeisma/Applied_Data_Science_Capstone/blob/main/spacex_dash_app%20(2).py

Predictive Analysis (Classification)

- Model Development
  - Load dataset
  - Preprocessing, standardization
  - Split data into training and test datasets using train_test_split()
  - Create GridSearchCV object and dictionary of parameters for chosen algorithm
  - Use training dataset to train model

- Model Evaluation
  - From the GridSearchCV object, check the hyperparameters (best_params_) and accuracy (score and best_score_)
  - Plot and examine the confusion matrix
  - Review accuracy scores for all chosen algorithms and chose best performing model appropriately

https://github.com/jjeisma/Applied_Data_Science_Capstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

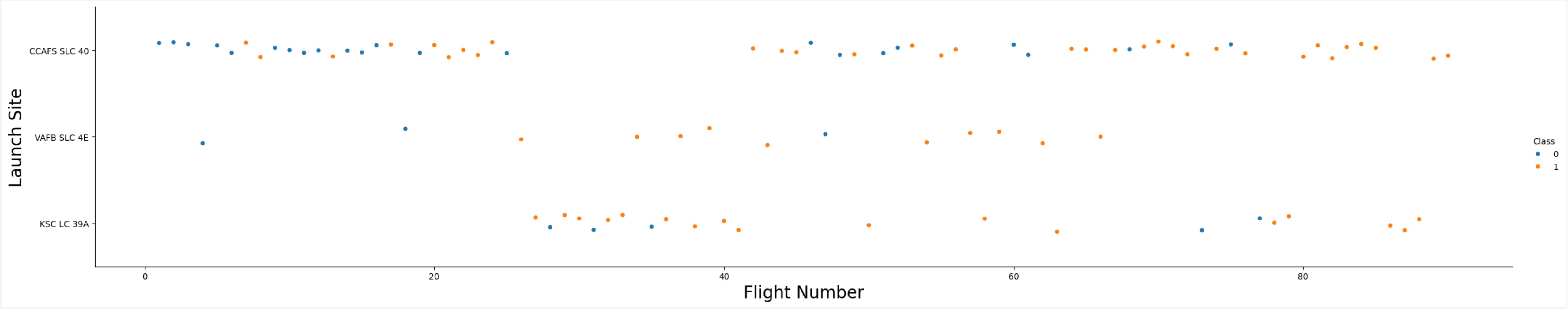
Results

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

Section 2

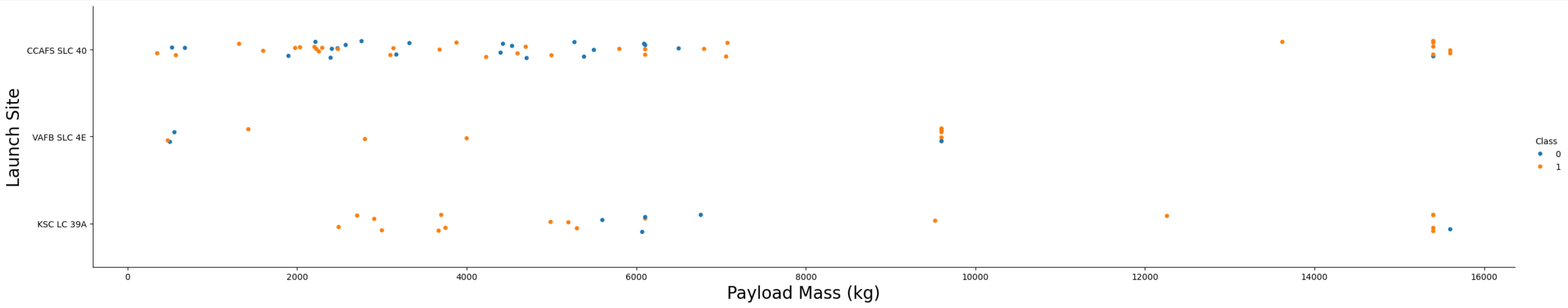
Insights drawn from EDA

Flight Number vs. Launch Site



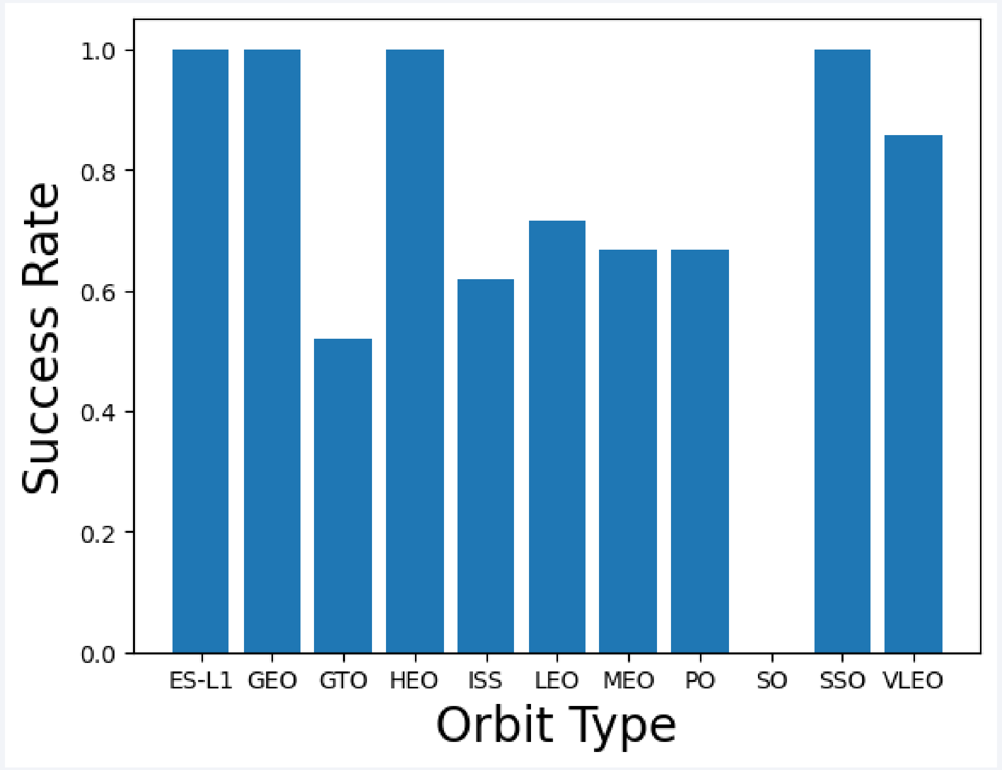
- As the flight number increases, so does the success rate
- Most of the low flight numbers (<30) were launched from CCAFS SLC 40 and were unsuccessful
- The low flight numbers (<30) from VAFB SLC 4E also were unsuccessful
- Most of the flight numbers >40 were successful

Payload vs. Launch Site



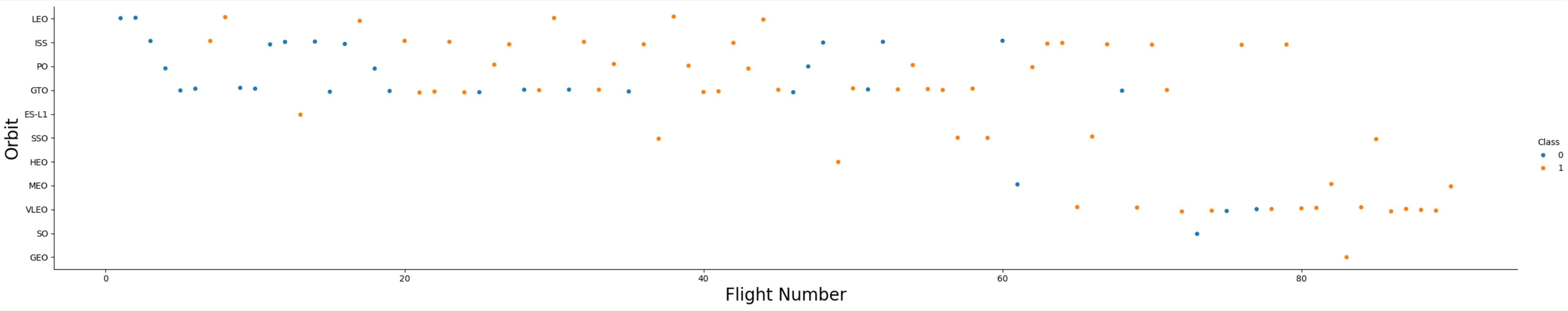
- Above a payload mass of 7000 kg, there are few unsuccessful launches
- There is no clear correlation between payload mass and success rate for any particular site

Success Rate vs. Orbit Type



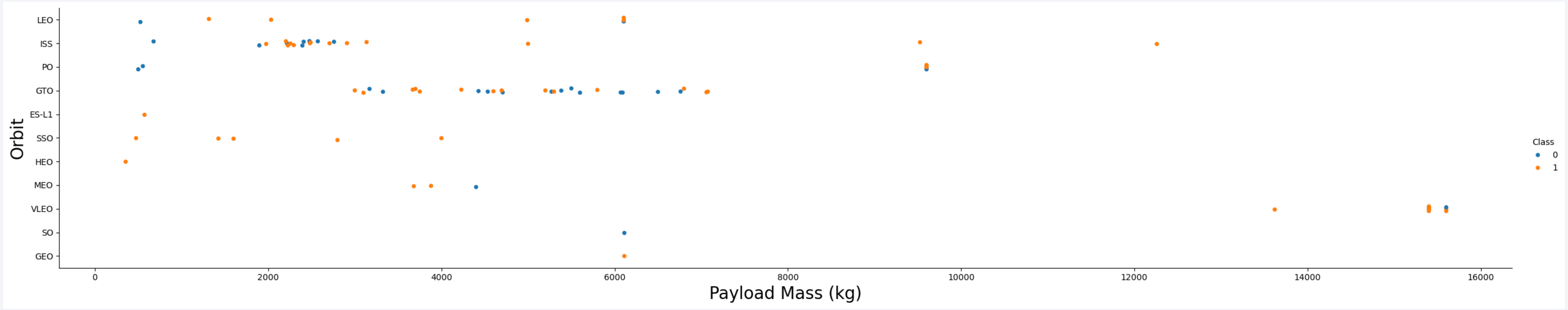
- ES-L1, GEO, HEO, and SSO orbit types have the highest success rates
- SO orbit type has the lowest success rate

Flight Number vs. Orbit Type



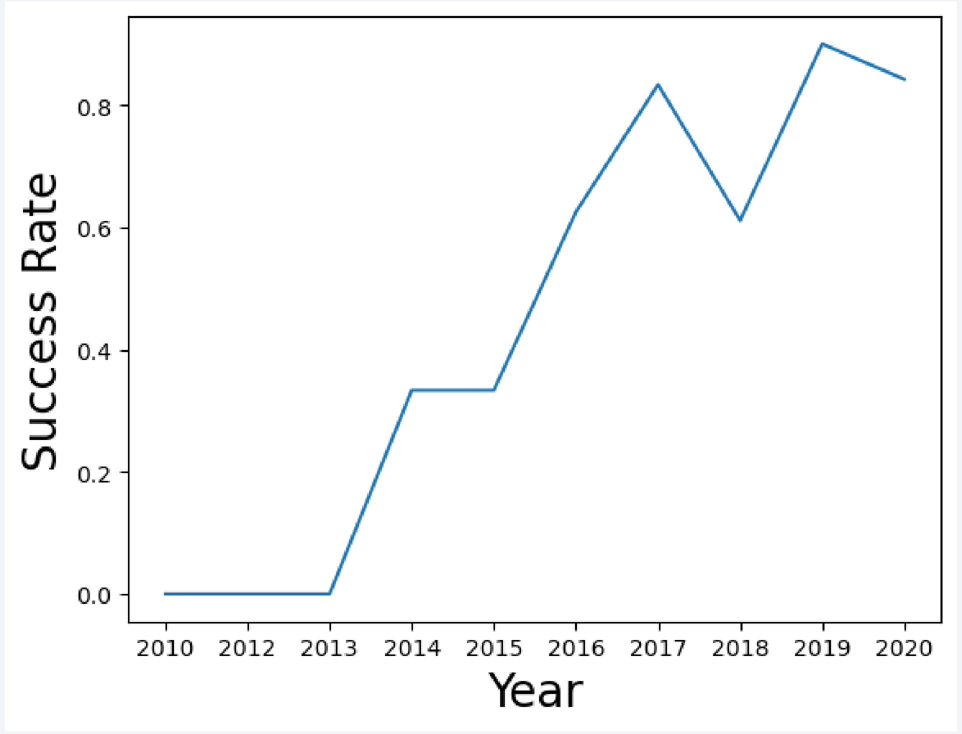
- The high success rates of GEO, HEO, and ES-L1 orbits are partially explained by them only having one data point
- SSO has a high success rate and at least 5 data points
- There is a general correlation between flight number and success rate, with higher flight numbers having a higher success rate (most clearly seen with LEO orbit)

Payload vs. Orbit Type



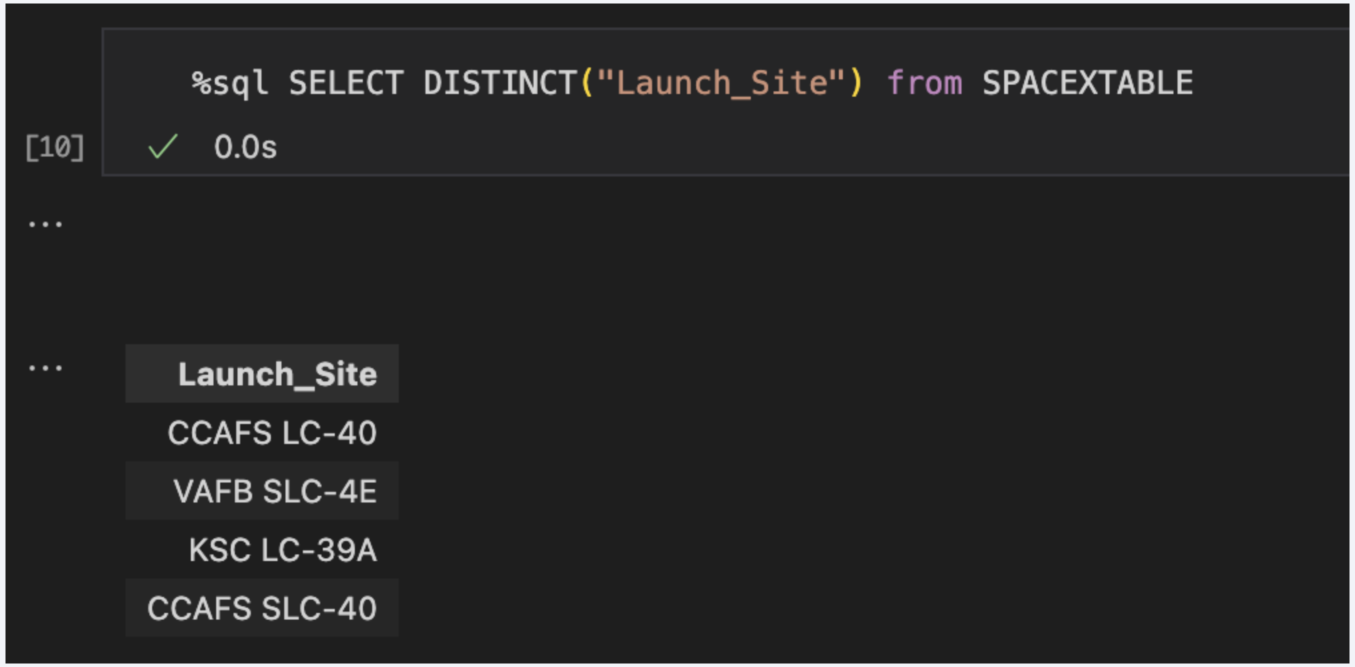
- LEO and ISS orbits have higher success with larger payload masses
- There is an unclear relationship between payload mass and success rate for GTO orbit

Launch Success Yearly Trend



- Between 2010 and 2013, there were no successful launches
- The success rate generally increased from 2013 to 2020, with small dips in 2015, 2018, and 2020

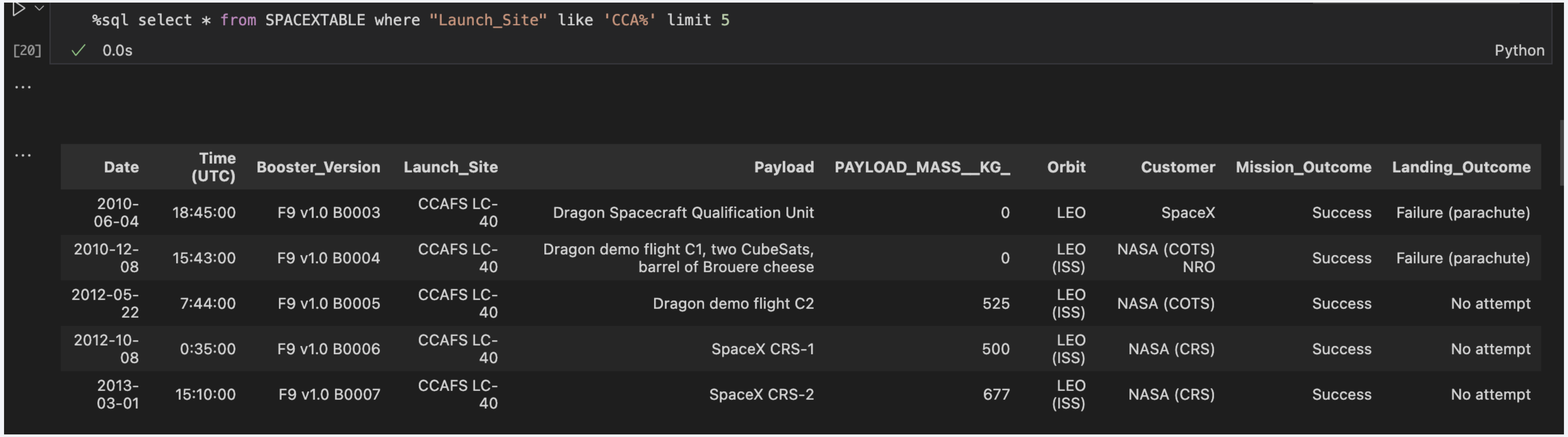
All Launch Site Names



```
%sql SELECT DISTINCT("Launch_Site") from SPACEXTABLE
[10]  ✓  0.0s
...
...
   Launch_Site
  CCAFS LC-40
  VAFB SLC-4E
   KSC LC-39A
 CCAFS SLC-40
```

- Find the names of the unique launch sites
- DISTINCT includes only the distinct values from the “Launch_Site” column

Launch Site Names Begin with 'CCA'



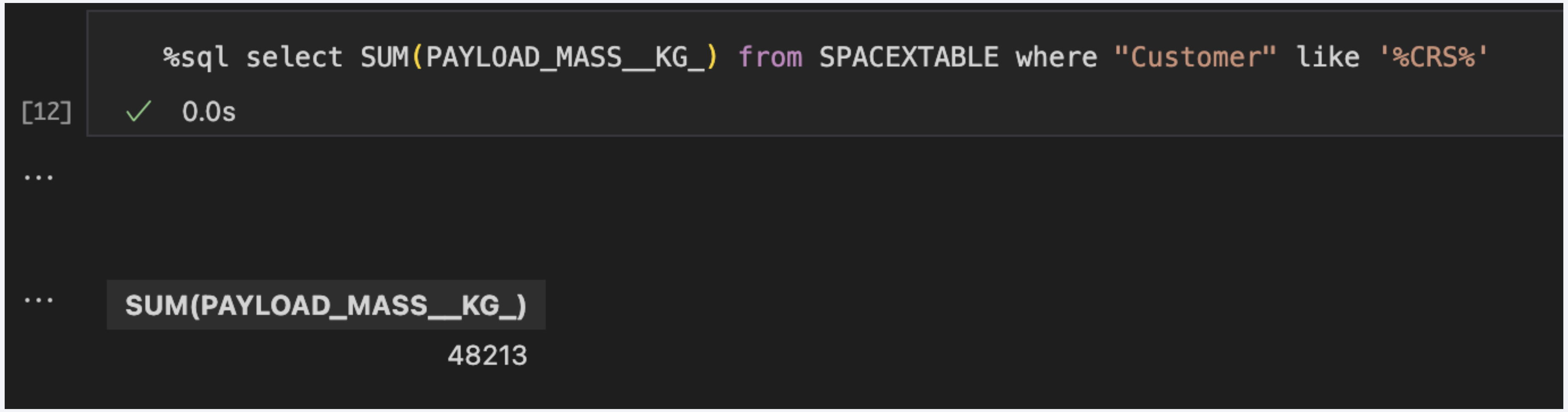
```
%sql select * from SPACEXTABLE where "Launch_Site" like 'CCA%' limit 5
```

[20] ✓ 0.0s Python

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

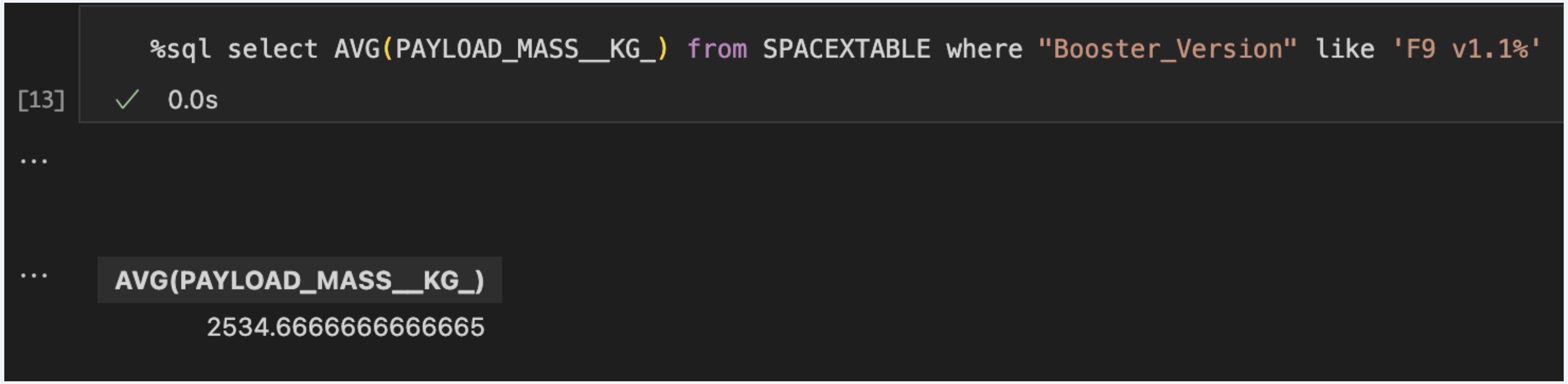
- Find 5 records where launch sites begin with `CCA`
- "limit 5" displays the first 5 entries
- "like 'CCA%'" shows entries in the Launch_Site column starting with 'CCA' string

Total Payload Mass


```
[12]   %sql select SUM(PAYLOAD_MASS__KG_) from SPACEXTABLE where "Customer" like '%CRS%'
       ✓ 0.0s
...
...    SUM(PAYLOAD_MASS__KG_)
                        48213
```

- Calculate the total payload carried by boosters from NASA
- SUM() calculates the total from the PAYLOAD_MASS__KG_ column
- ”like ‘%CRS%’” includes entries where Customer column contains the ‘CRS’ string, indicative of NASA launches

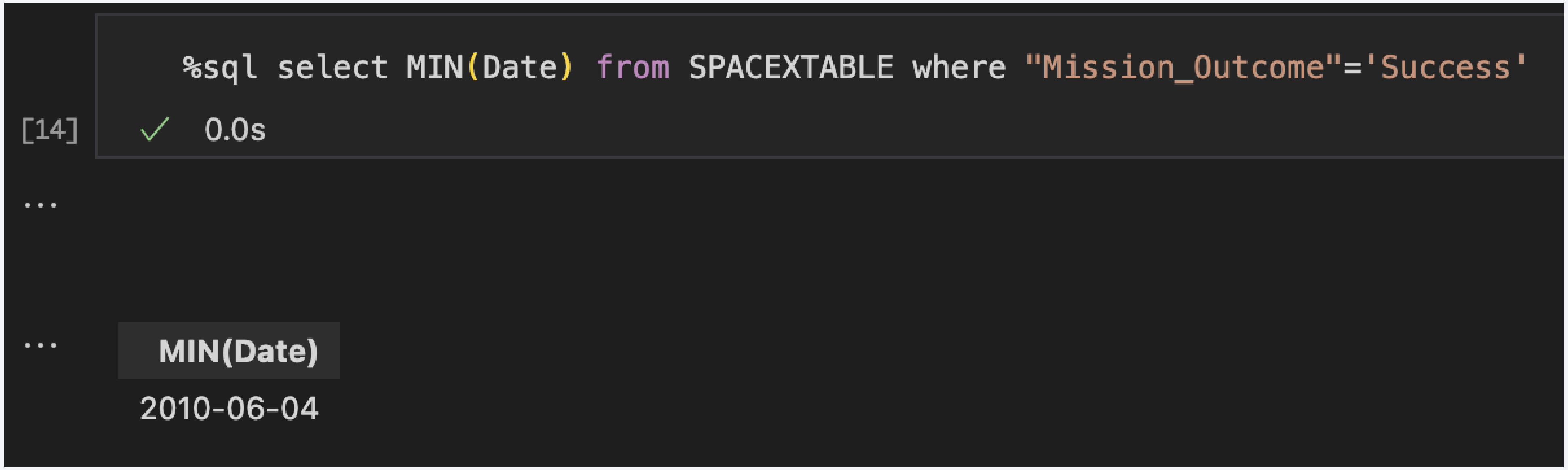
Average Payload Mass by F9 v1.1



```
%sql select AVG(PAYLOAD_MASS__KG_) from SPACEXTABLE where "Booster_Version" like 'F9 v1.1%'
[13] ✓ 0.0s
...
... AVG(PAYLOAD_MASS__KG_)
          2534.6666666666665
```

- Calculate the average payload mass carried by booster version F9 v1.1
- AVG() calculates the average from the PAYLOAD_MASS__KG_ column
- “like ‘F9 v1.1%’” includes entries where Booster_Version column contains ’F9 v1.1’ string

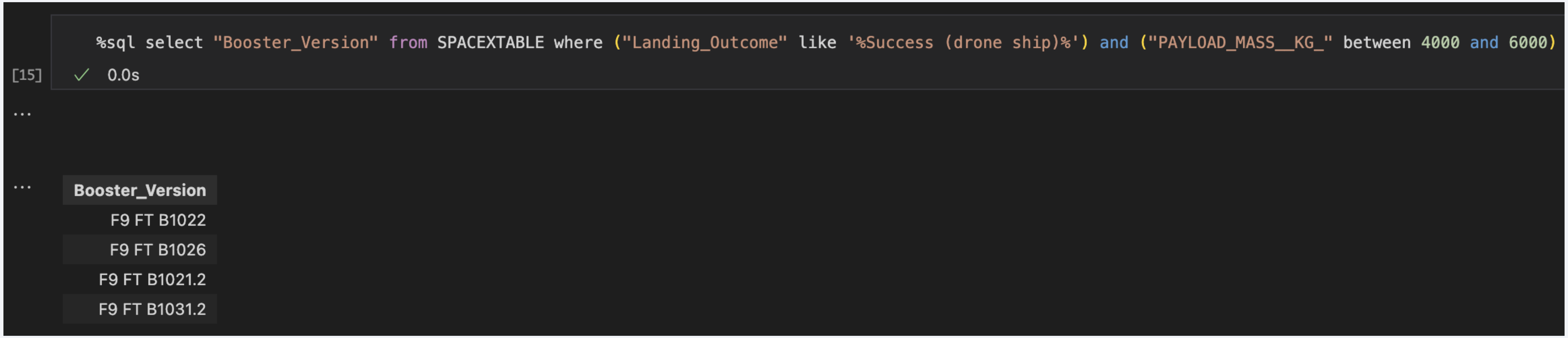
First Successful Ground Landing Date



```
%sql select MIN(Date) from SPACEXTABLE where "Mission_Outcome"='Success'
[14] ✓ 0.0s
...
...
MIN(Date)
2010-06-04
```

- Find the dates of the first successful landing outcome on ground pad
- MIN() finds the earliest date in the Date column
- ”where “Mission_Outcome”=’Success’” filters the search to successful landings only

Successful Drone Ship Landing with Payload between 4000 and 6000

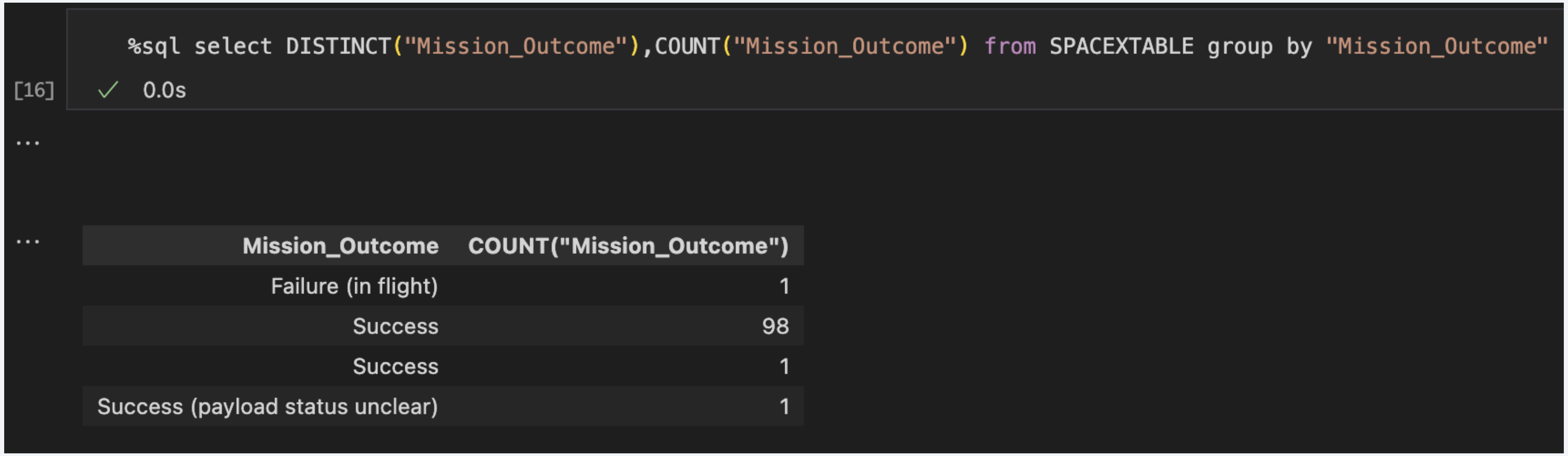


```
%sql select "Booster_Version" from SPACEXTABLE where ("Landing_Outcome" like '%Success (drone ship)%') and ("PAYLOAD_MASS__KG_" between 4000 and 6000)
[15] ✓ 0.0s
...
```

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

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000
- ”like ‘%Success (drone ship)%’” filters search to just successful landings on drone ships
- ””PAYLOAD_MASS__KG_ between 4000 and 6000”” filters search to payload masses between 4000 and 6000 kg

Total Number of Successful and Failure Mission Outcomes

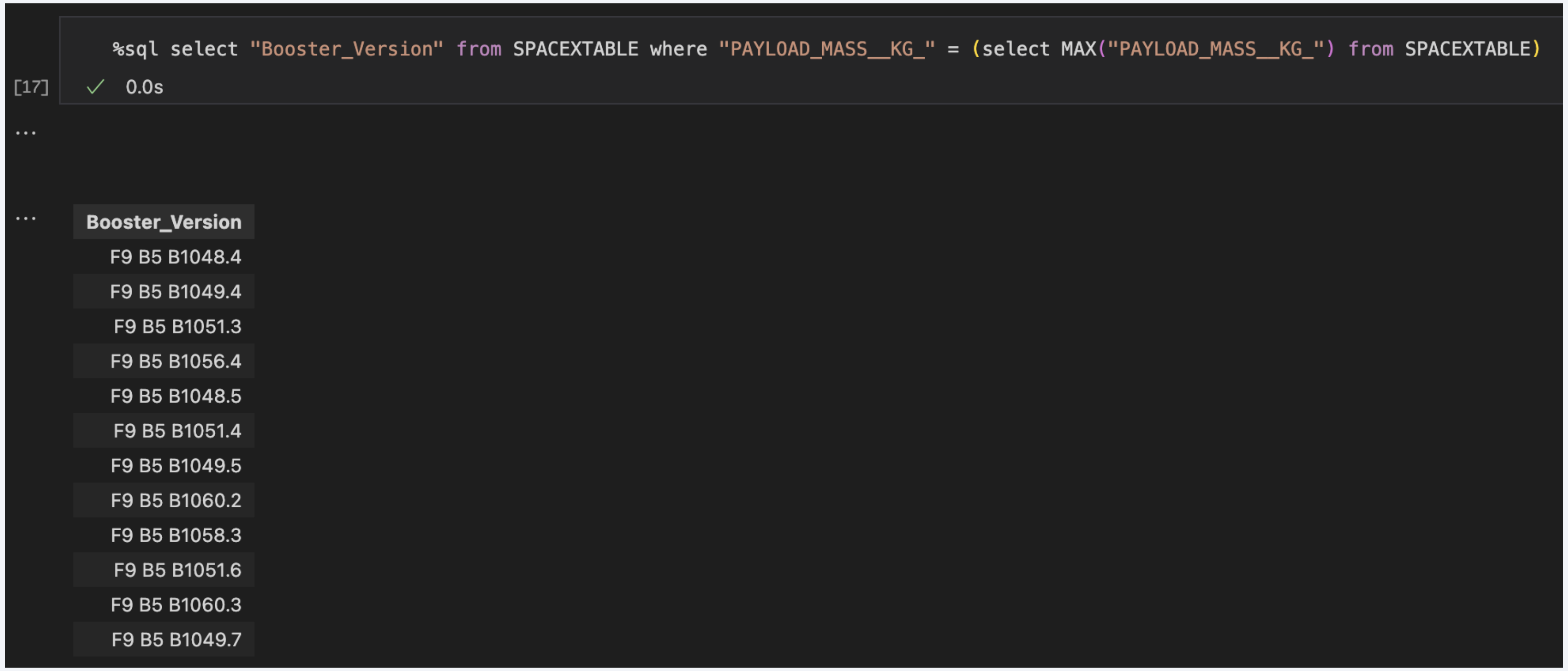


```
%sql select DISTINCT("Mission_Outcome"),COUNT("Mission_Outcome") from SPACEXTABLE group by "Mission_Outcome"
[16]  ✓ 0.0s
```

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

- Calculate the total number of successful and failure mission outcomes
- COUNT() displays the count for each Mission Outcome
- “group by ”Mission_Outcome”” groups the counts by Mission Outcome

Boosters Carried Maximum Payload



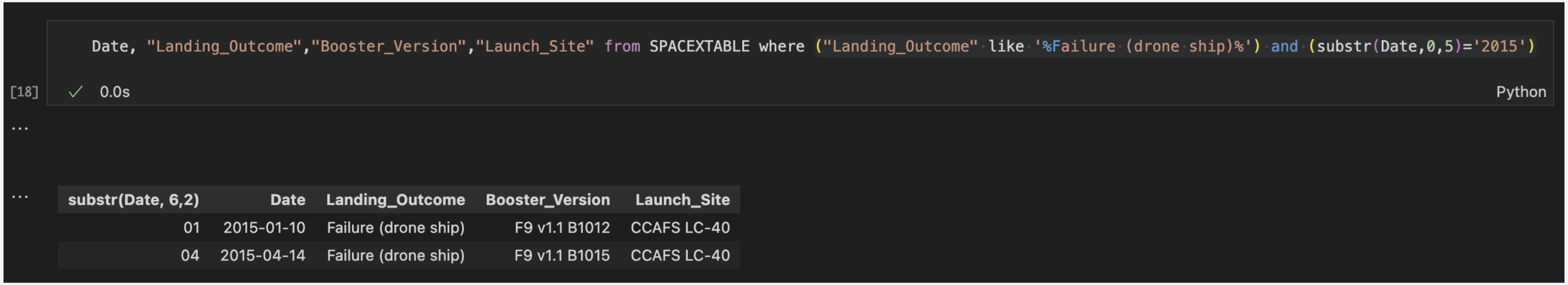
```
%sql select "Booster_Version" from SPACEXTABLE where "PAYLOAD_MASS__KG_" = (select MAX("PAYLOAD_MASS__KG_") from SPACEXTABLE)
```

[17] ✓ 0.0s

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

- List the names of the booster which have carried the maximum payload mass
- Using a subquery, the inner select statement finds the maximum payload mass, and the where statement uses this maximum value to filter the displayed results

2015 Launch Records

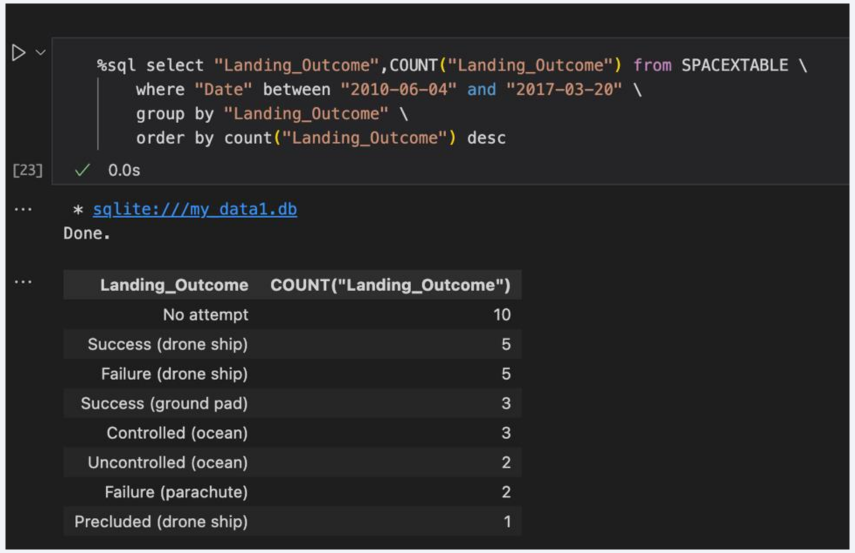


```
    Date, "Landing_Outcome","Booster_Version","Launch_Site" from SPACEXTABLE where ("Landing_Outcome" like '%Failure (drone ship)%') and (substr(Date,0,5)='2015')
[18]  ✓  0.0s                                                                                                              Python
...
```

substr(Date, 6,2)	Date	Landing_Outcome	Booster_Version	Launch_Site
01	2015-01-10	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	2015-04-14	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
- “where "Landing_Outcome" like '%Failure (drone ship)%') and (substr(Date,0,5)='2015’)” filters the search to only failures on drone ships in the year 2015

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
[23]  ✓ 0.0s

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

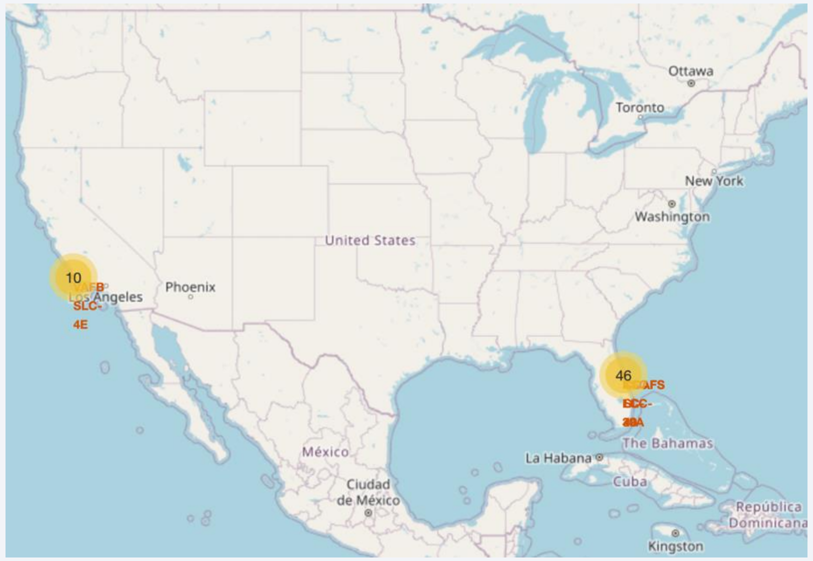
Landing_Outcome	COUNT("Landing_Outcome")
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

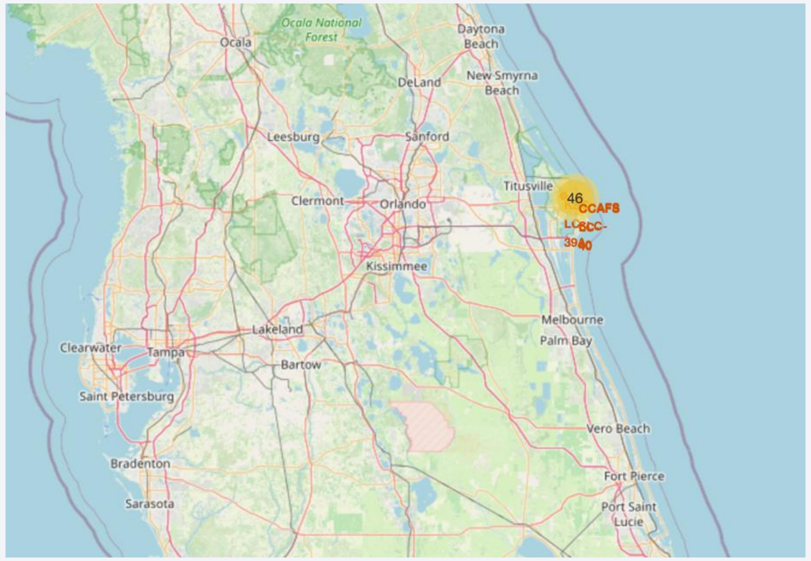
- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order
- ”desc” puts the list in descending order
- “group by "Landing_Outcome" order by count("Landing_Outcome")” groups and orders the results by count in “Landing_Outcome” column
- “where "Date" between "2010-06-04" and "2017-03-20”” filters results to be between the specified dates

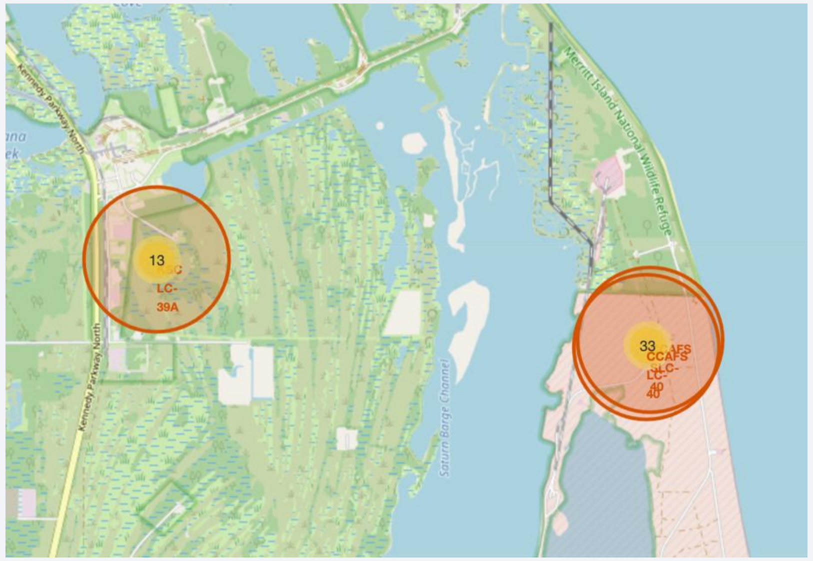
Section 3

Launch Sites Proximities Analysis

All Launch Sites on Map

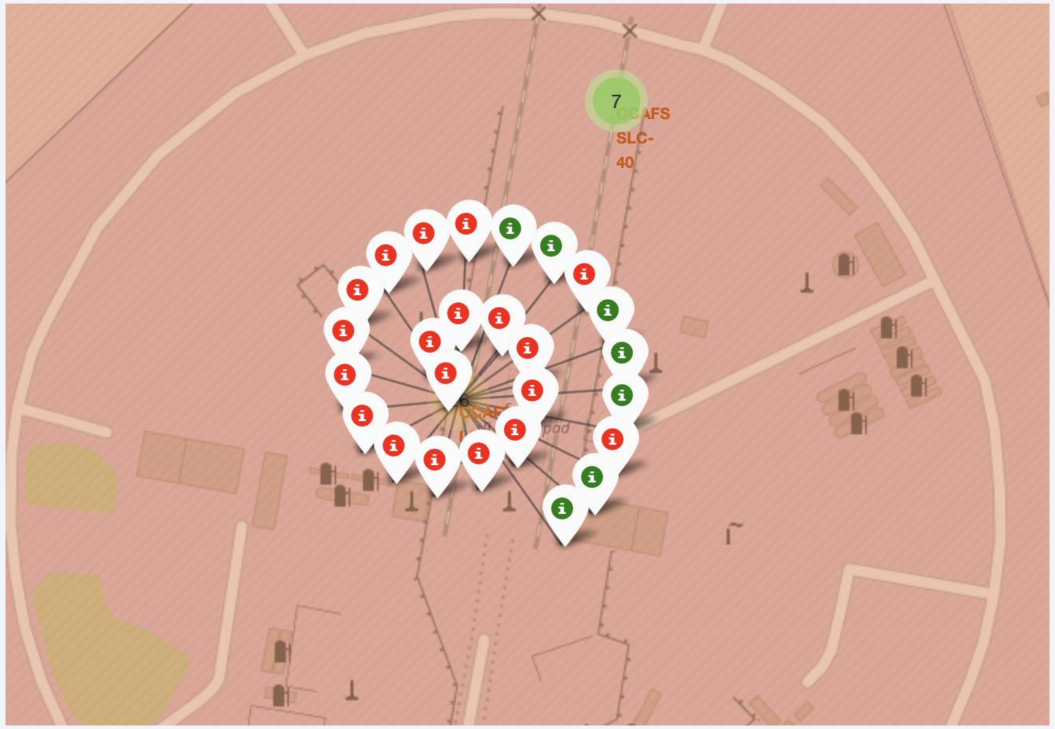






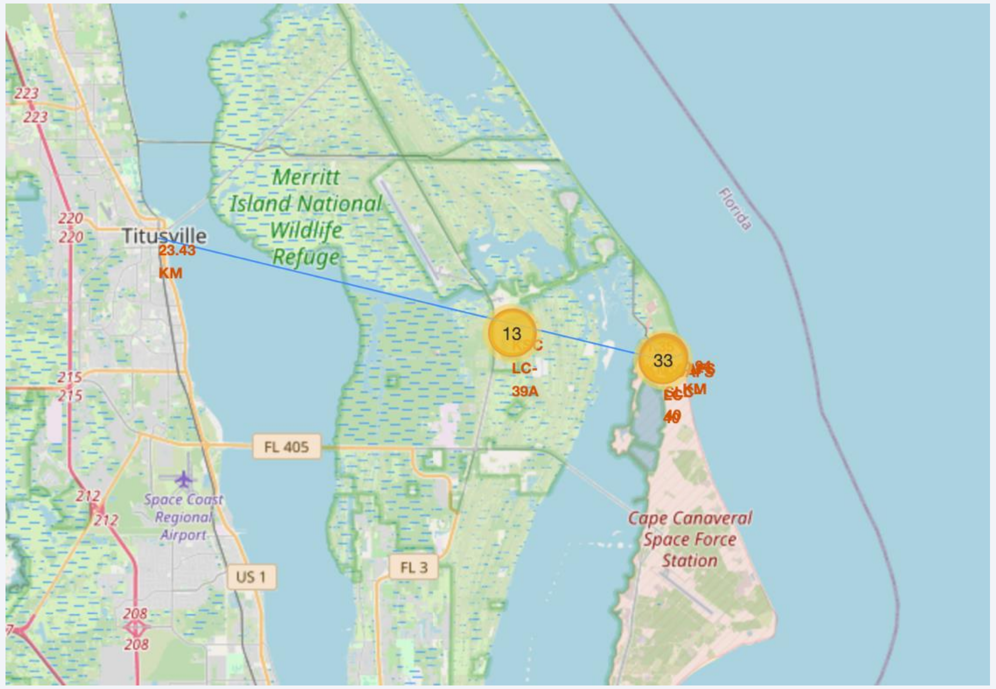
- All launch sites are shown in the map on the left, which appear to be concentrated on the west and east coasts of the United States.
- Progressing from left to right, the maps zoom in on the launch sites in Cape Canaveral, Florida

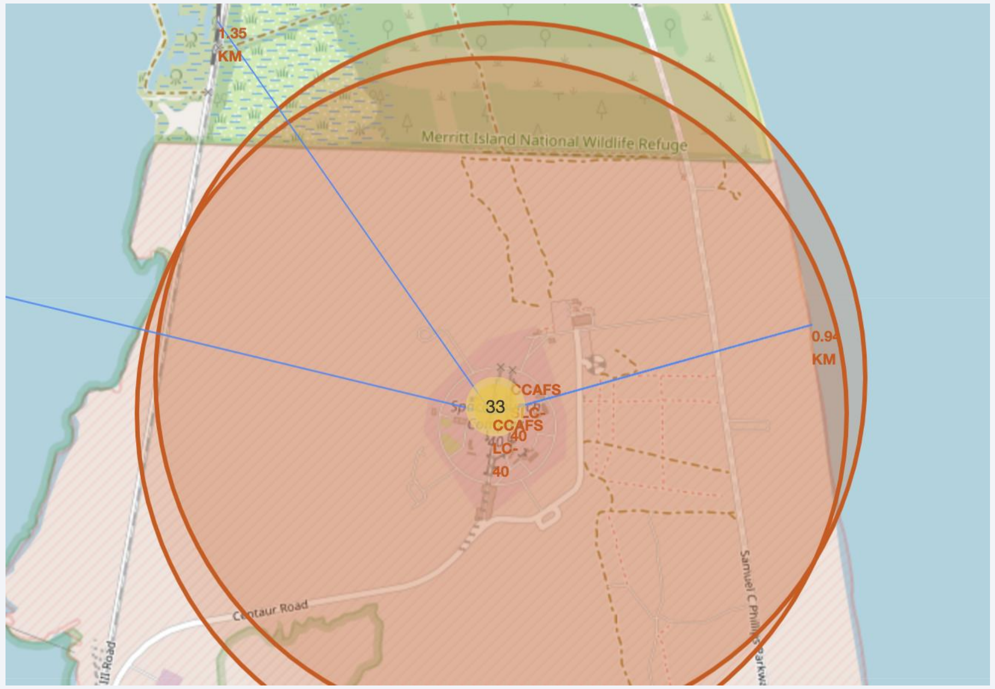
Color-Coded Outcomes for Each Launch Site



- Launches have been annotated with green icons for successful launches and red icons for failed launches
- Launches also have been clustered to ease the display of several closely-located launches

Distances Between Launch Site and Points of Interest


Merritt Island National Wildlife Refuge
Titusville
23.43 KM
13
LC-39A
33
FL 405
Space Coast Regional Airport
Cape Canaveral Space Force Station
US 1
FL 3
Florida

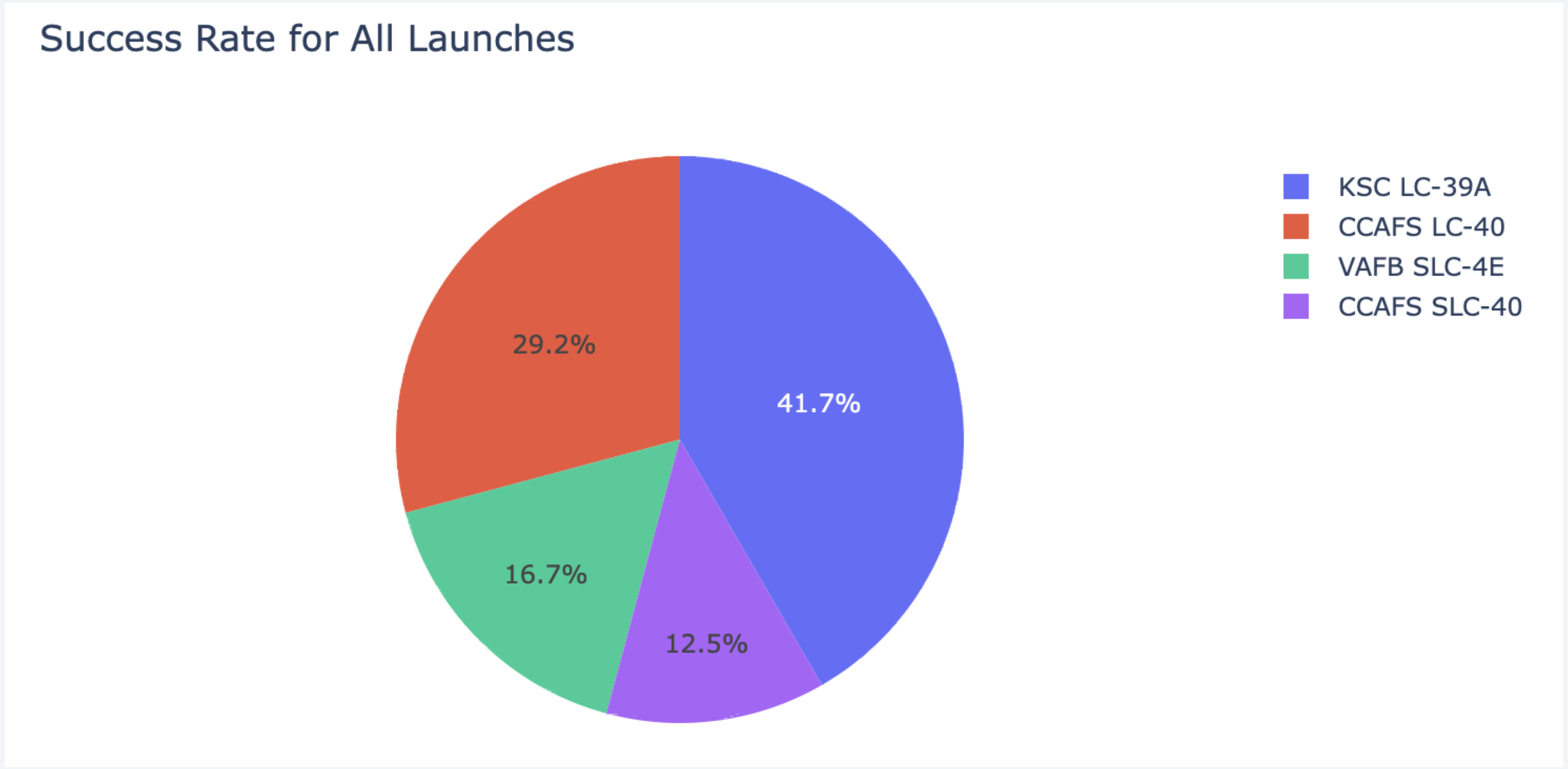

1.35 KM
Merritt Island National Wildlife Refuge
0.94 KM
33
CCAFS
LC-40
Centaur Road
Samuel C Phillips Parkway

- The screenshots display the distances between the Cape Canaveral Launch Site and the nearest coastline, railway, and city, which are all within close proximity.

Section 4

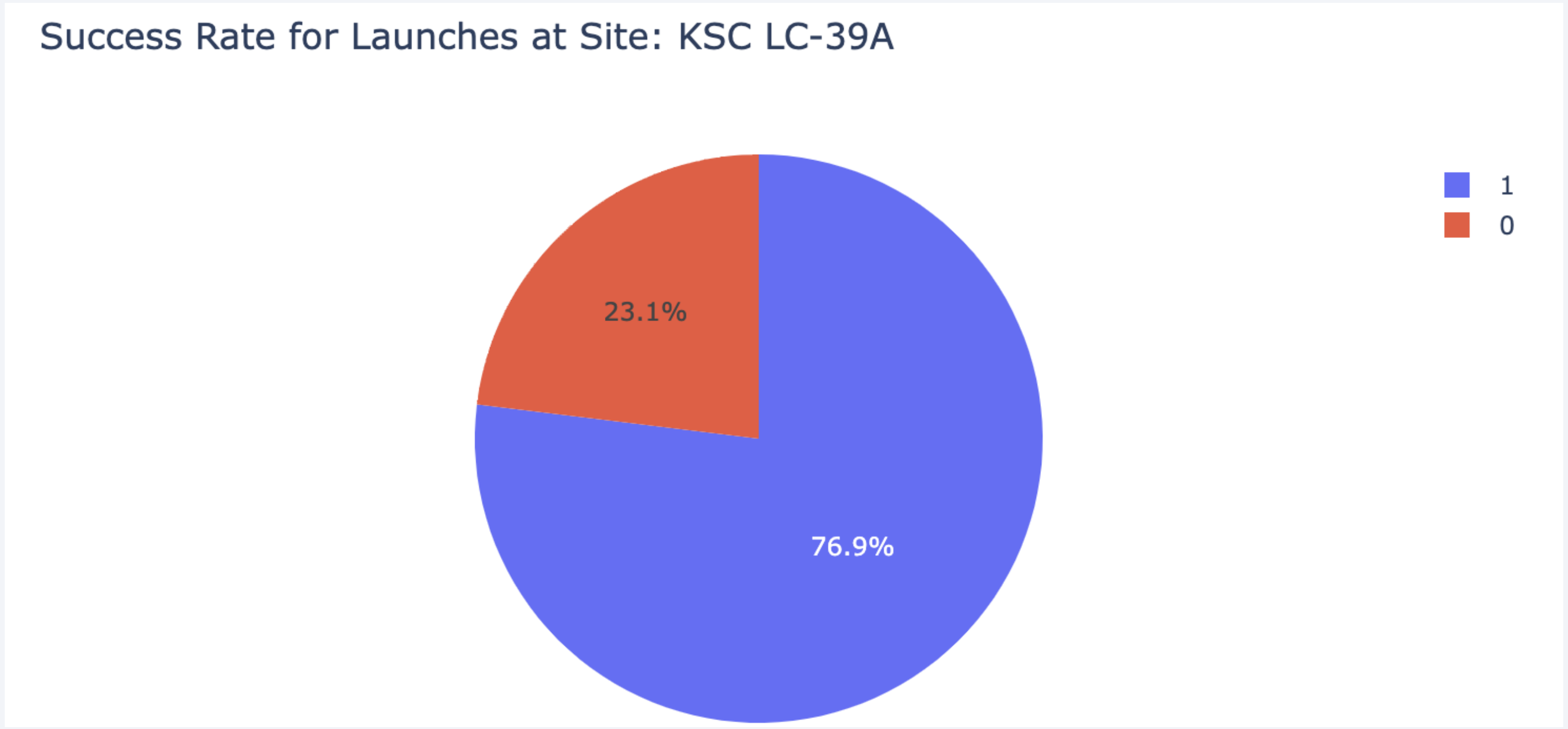
Build a Dashboard with Plotly Dash

Total Launch Success by Site



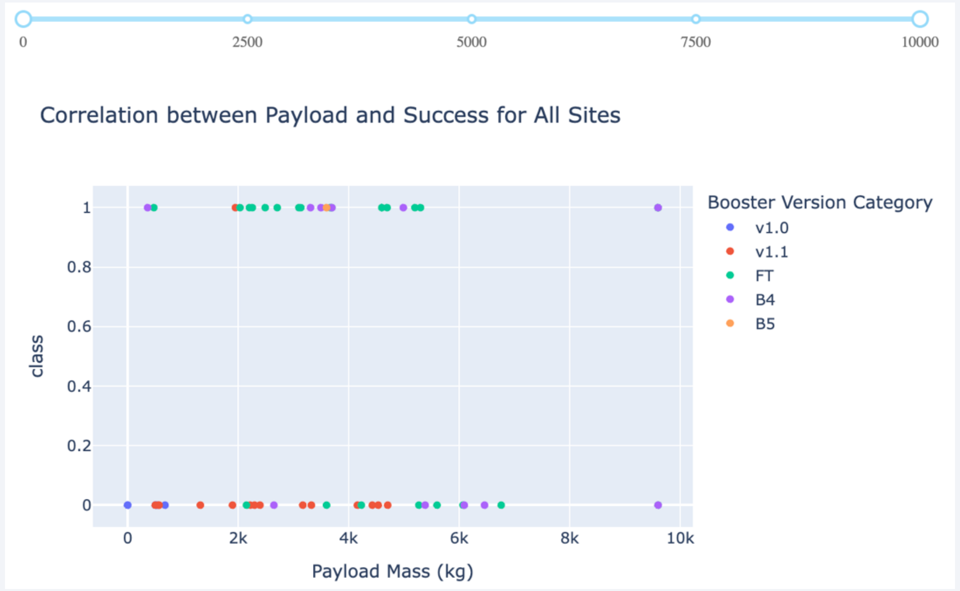
- The KSC LC-39A had the most successful launches, with 41.7% of the total launches.

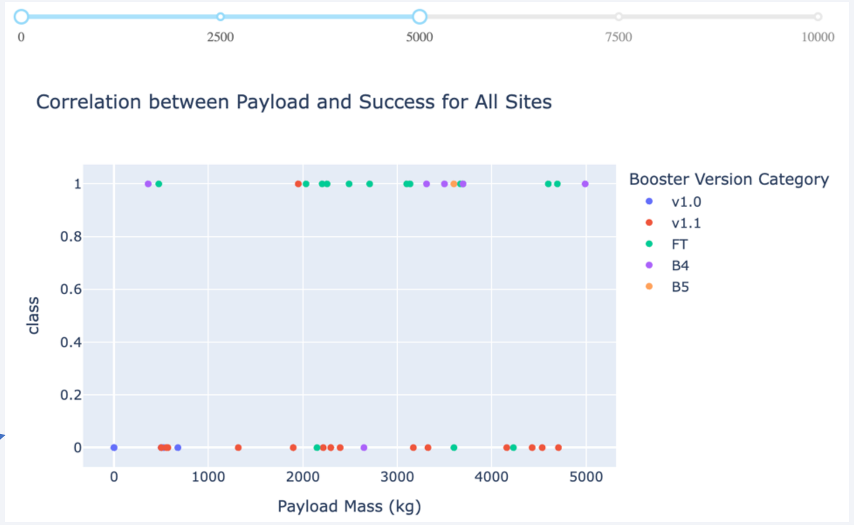
Launch Success Rate at the Most Successful Site

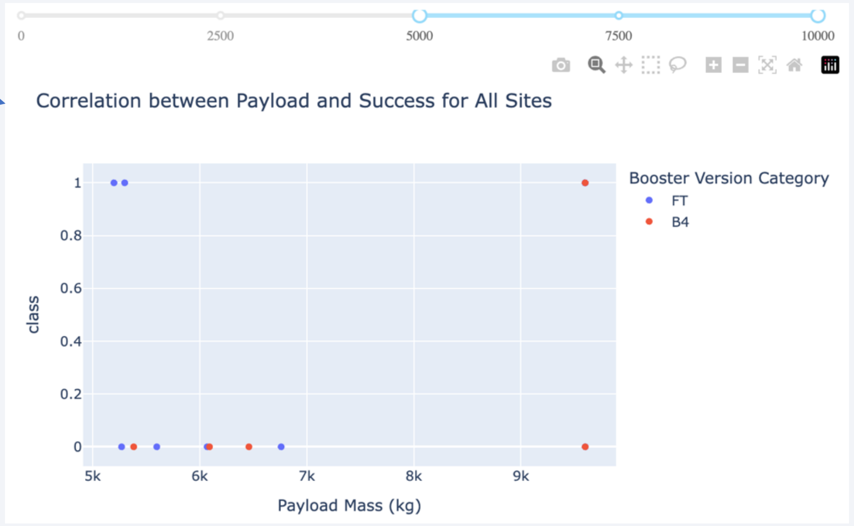


- The KSC LC-39A launch site had the highest success rate at 76.9%.

Launch Outcome vs. Payload Mass







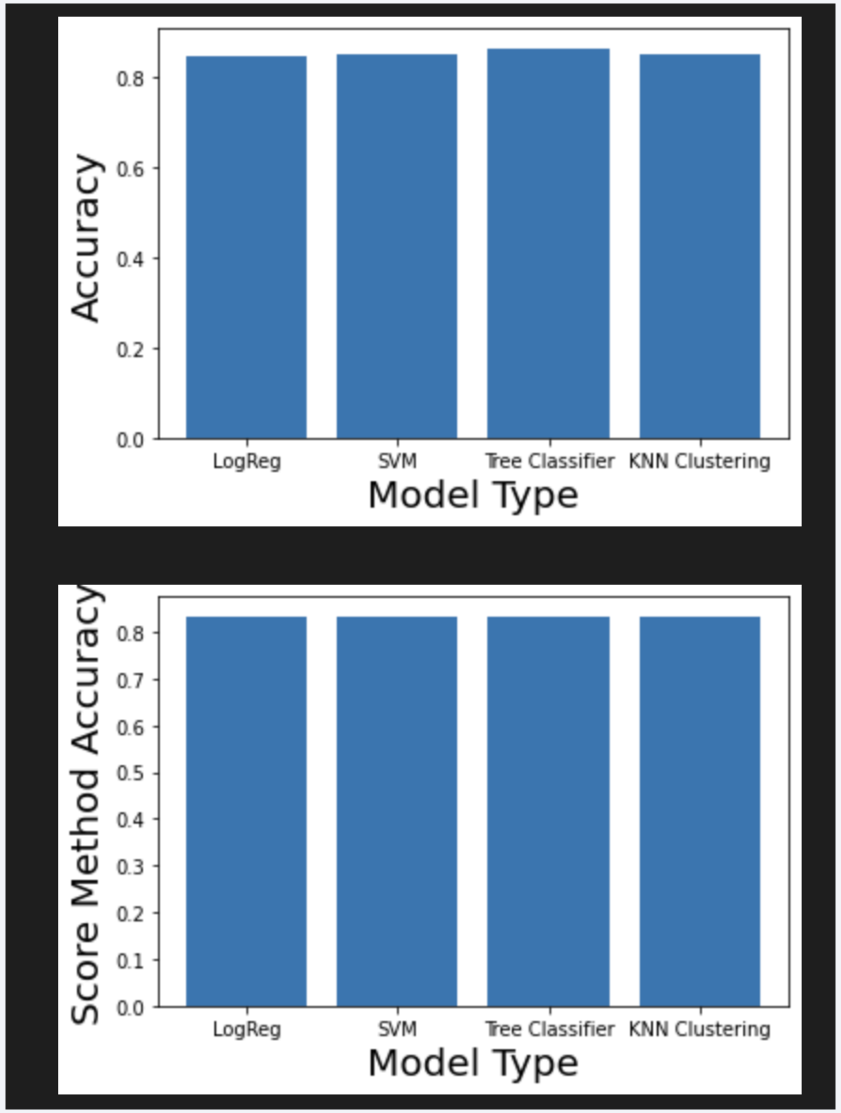
- Separating the above plot by payload mass < 5000 kg and > 5000 kg allows one to see that larger payloads have less successful launches across all sites.

Section 5

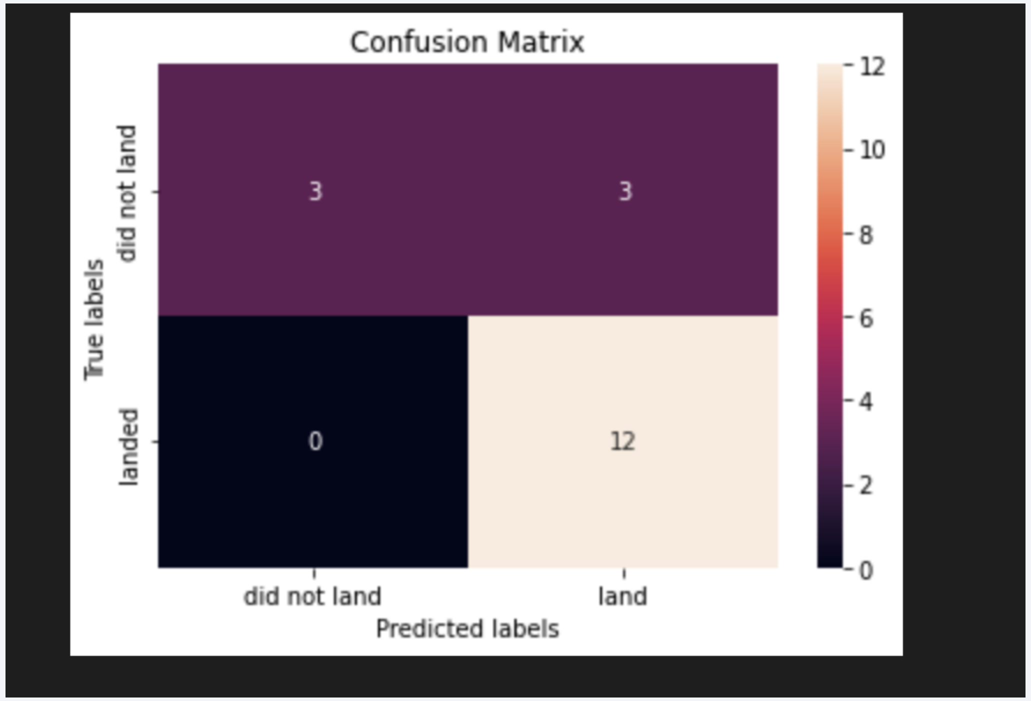
Predictive Analysis (Classification)

Classification Accuracy

- The accuracies of the four different models are shown on the right.
- The Tree Classifier performed marginally better than the others, with an accuracy of 86.25%.



Confusion Matrix



- The confusion matrix from the Tree Classifier model is shown above, with an accuracy of 86.25%.
- 3 out of 18 results were incorrectly predicted as successes (false positive), and there were 0 false negatives.
- 15 out of 18 results were correctly classified.

Conclusions

- As the flight number increases, the success rate increases across launch sites.
- Orbit types ES-L1, GEO, HEO, and SSO have the highest success rates.
- The launch site KSC LC-39A has the most successful launches and highest rate of successful launches.
- The best performing classification model is the Tree Classifier model.

Thank you!