IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

SHEMSU DABESA GANE
OCTOBER 2022

Outline

Executive Summary

- Summary of methodologies
  - Collecting the Data with an API: Gathering SpaceX launch data by using the SpaceX REST API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Analysis Using SQL and Data visualization
  - Interactive Visual Analytics/Map with Folium
  - Interactive Dashboard Using Plotly Dash
  - Predictive Analysis (For each classification model)
- Summary of all results
  - Exploratory Data Analysis(EDA) result
  - Predictive Analytics result from Machine Learning Lab
  - Interactive analytics in screenshots

Introduction

- Project background and context

  The commercial space age is here, companies are making space travel affordable for everyone. Virgin Galactic is providing suborbital spaceflights. Rocket Lab is a small satellite provider. Blue Origin manufactures sub-orbital and orbital reusable rockets. Perhaps the most successful is SpaceX. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upwards 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. Stage two, or the second stage, helps bring the payload to orbit, but most of the work is done by the first stage. The first stage does most of the work and is much larger than the second stage. This stage is quite large and expensive. Unlike other rocket providers, SpaceX's Falcon 9 Can recover the first stage. Sometimes the first stage does not land. Sometimes it will crash . Other times, Space X will sacrifice the first stage due to the mission parameters like payload, orbit, and customer. The 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 needed to increase the probability of successful landing/the best result

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
    - Data was collected by using SpaceX REST API and web scrapping from Wikipedia using Python BeautifulSoup package
- Perform data wrangling
    - One-hot encoding was applied to categorical features, Sampling Data(to remove/filter unnecessary features),Dealing with Nulls
- 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

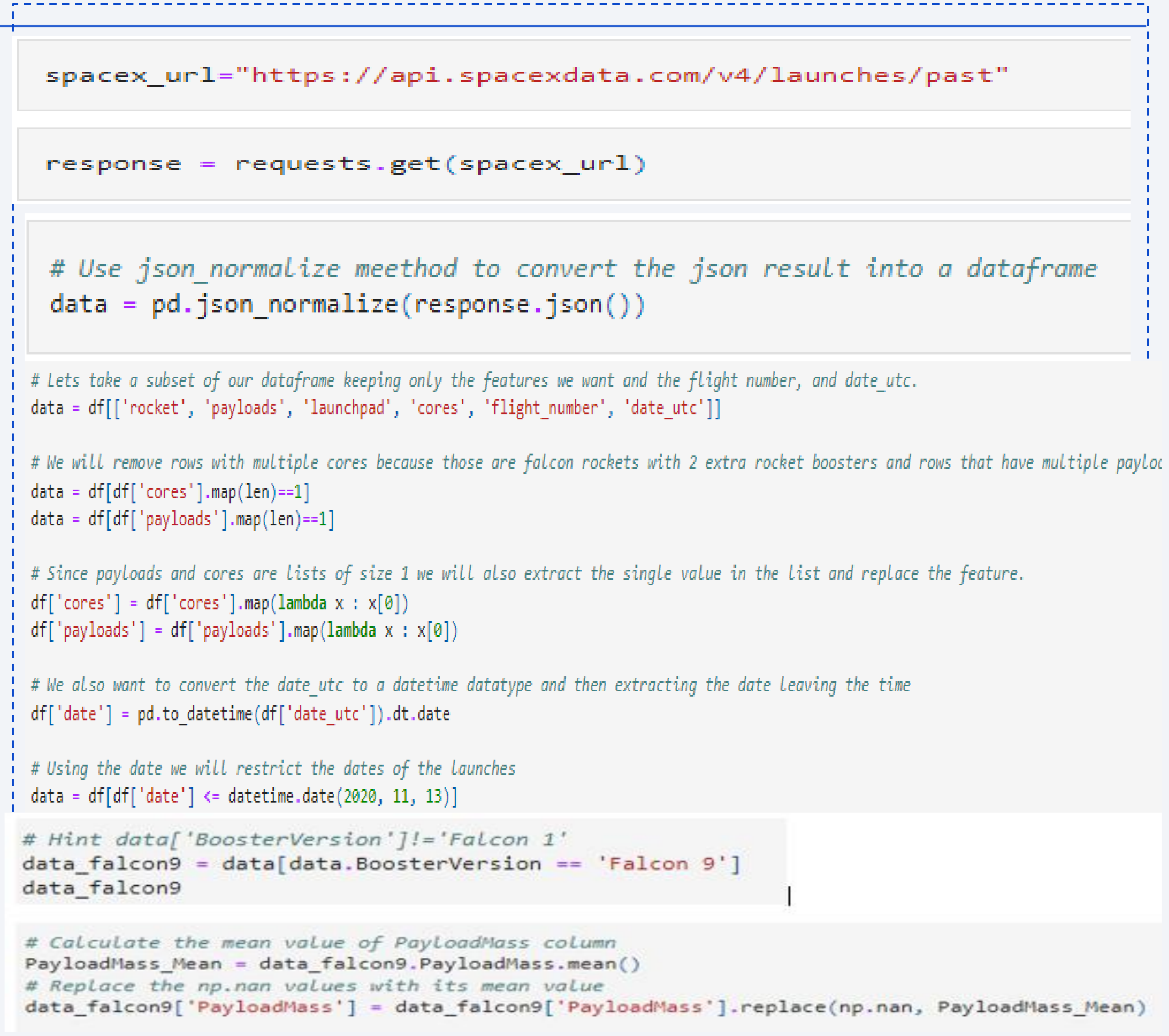
Data Collection

- Describe how data sets were collected.
  - As mentioned in the methodology section, the dataset was collected by using get request to the SpaceX REST API and web scrapping from Wikipedia
  - Next, we decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
  - Then we Filter the dataframe to only include Falcon 9 launches by removing data related to the Falcon 1 launches
  - Then we deal with Missing Values: We calculate the mean of the PayloadMass data and then replace the null values in PayloadMass with the mean to clean the data.
  - 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.
  - Finally we export the data to a **CSV** for future use.

Data Collection – SpaceX API

- The dataset was collected by using get request to the SpaceX REST API
- We Use json_normalize method to convert json result to dataframe
- Performed data cleaning(feature selection, filtering, conversion) and filling the missing value
- GitHub URL :

https://github.com/shemsud1/Applied-Data-Science-Capstone/blob/main/jupyter-labs-spacex-data-collection-api%20.ipynb



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

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

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

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = df[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have multiple paylo
data = df[df['cores'].map(len)==1]
data = df[df['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature.
df['cores'] = df['cores'].map(lambda x : x[0])
df['payloads'] = df['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
df['date'] = pd.to_datetime(df['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = df[df['date'] <= datetime.date(2020, 11, 13)]
```

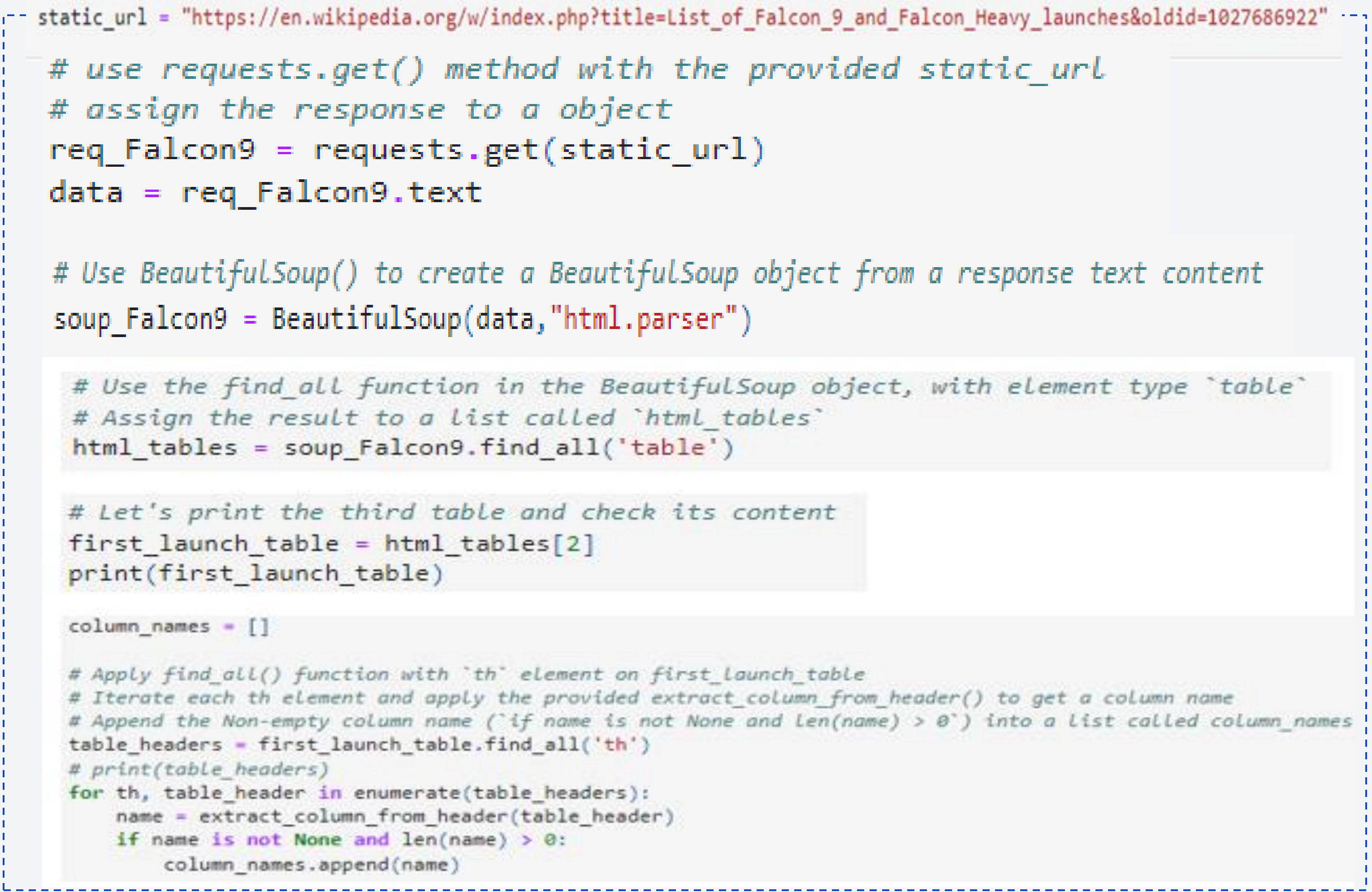
```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = data[data.BoosterVersion == 'Falcon 9']
data_falcon9
```

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

Data Collection - Scraping

- Request the Falcon9 Launch Wiki page from its URL
- Create a BeautifulSoup object from the HTML response
- Extract all column names from the HTML table header and
- Create a data frame by parsing the launch HTML tables
- GitHub URL:

https://github.com/shemsud1/Applied-Data-Science-Capstone/blob/main/jupyter-labs-webscraping.ipynb



```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

```
# use requests.get() method with the provided static_url
# assign the response to a object
req_Falcon9 = requests.get(static_url)
data = req_Falcon9.text

# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup_Falcon9 = BeautifulSoup(data,"html.parser")
```

```
# 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_Falcon9.find_all('table')
```

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

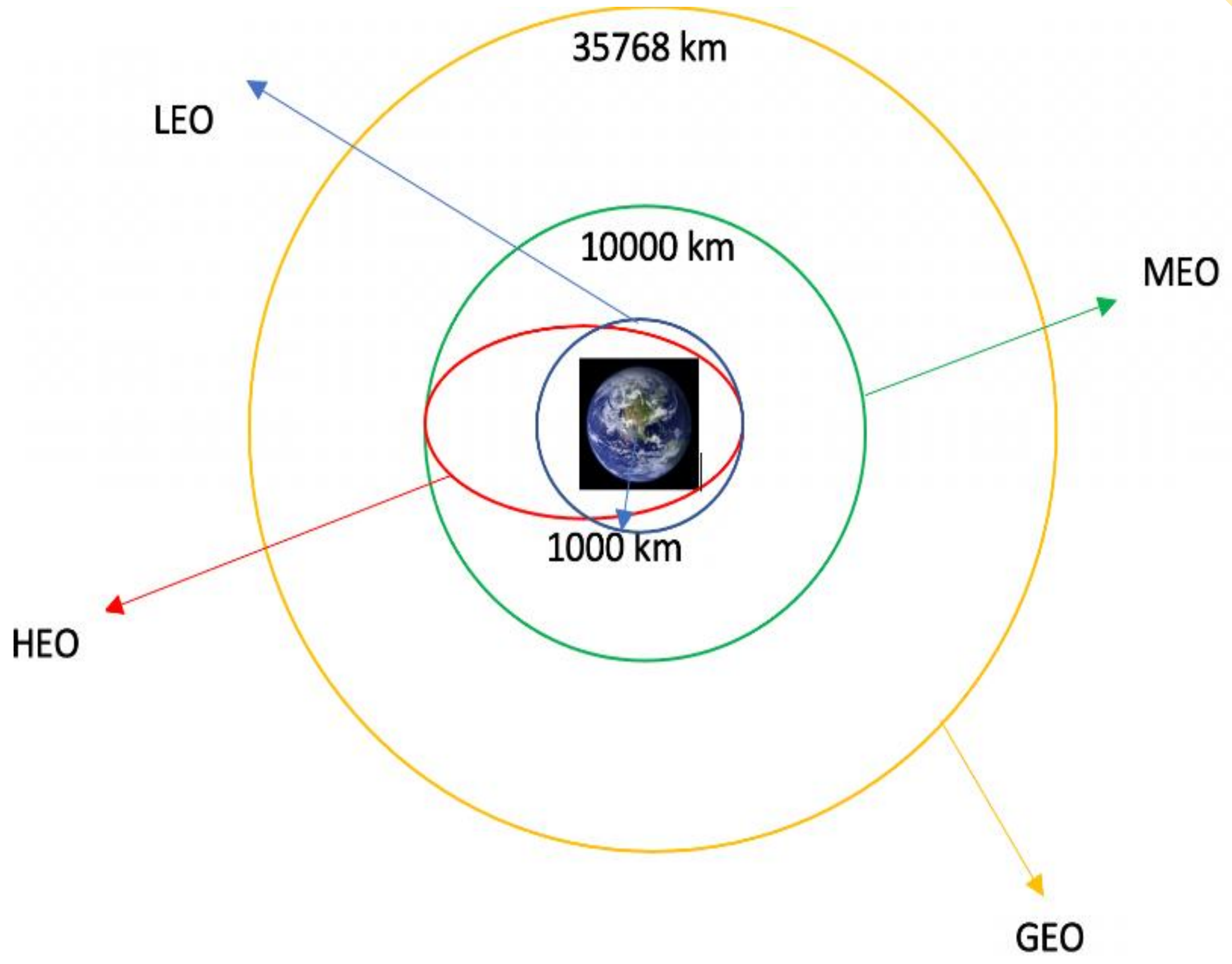
```
column_names = []

# 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
table_headers = first_launch_table.find_all('th')
# print(table_headers)
for th, table_header in enumerate(table_headers):
    name = extract_column_from_header(table_header)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

Data Wrangling

- Perform some EDA) to find some patterns in the data and determine what would be the label for training supervised models
- Calculate the number of launches on each site
- Calculate the number and occurrence of each orbit
- Calculate the number and occurrence of mission outcome per orbit type
- Create a landing outcome label from Outcome column & export the result to csv
- GitHub URL:

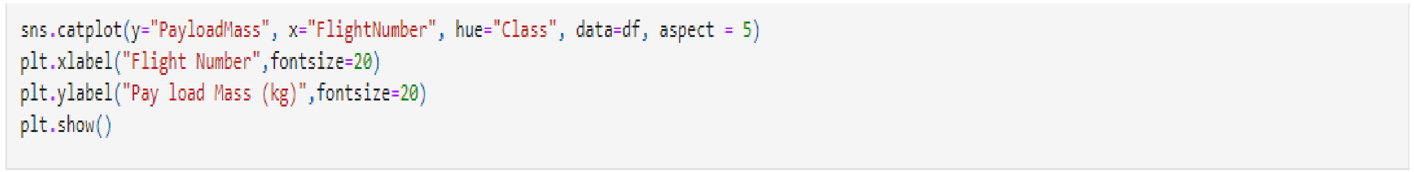
https://github.com/shemsud1/Applied-Data-Science-Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb



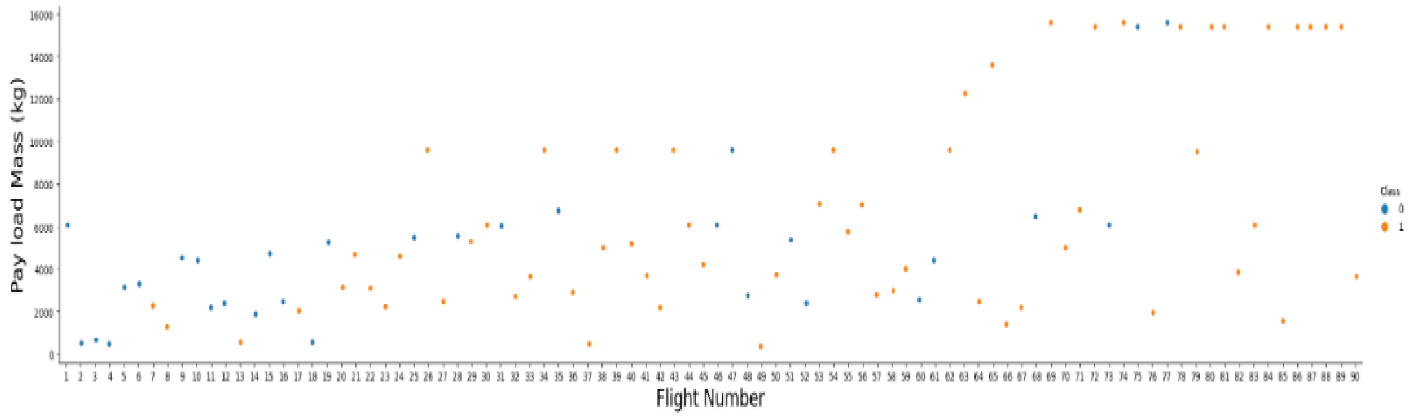
EDA with Data Visualization

- We first started by using scatter plot to find the relationship between the variables such as between:
    - PayloadMass and Flight Number.
    - Flight Number and Launch Site.
    - PayloadMass and Launch Site.
    - Flight Number and Orbit Type.
    - PayloadMass and Orbit Type.
- GitHub URL:

https://github.com/shemsud1/Applied-Data-Science-Capstone/blob/main/jupyter-labs-eda-dataviz.ipynb



```
sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```

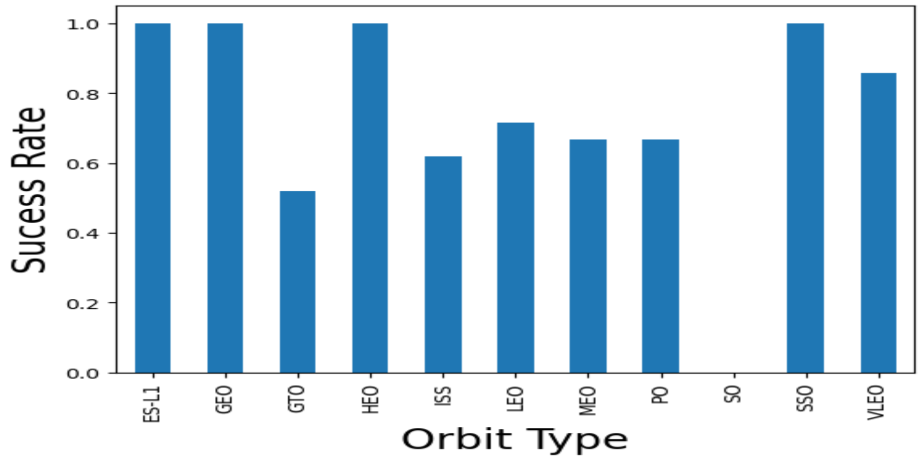


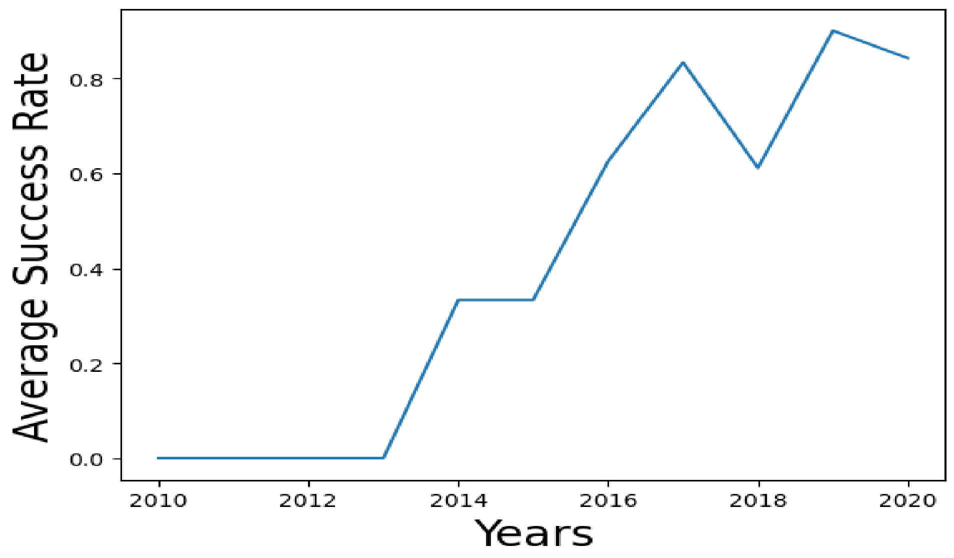
- Scatter plots show dependency of attributes on each other. Once a pattern is determined from the graphs. It’s very easy to see which factors affecting the most to the success of the landing outcomes.
- We see that as the flight number increases, the first stage is more likely to land successfully. The payload mass is also important; it seems the more massive the payload, the less likely the first stage will return.

EDA with Data Visualization(Cont..)

- To Visualize the relationship between success rate of each orbit type, we created Bar Chart. From this bar graph we can easily determine which orbits have the highest probability of success.
- To visualize and get the average launch success yearly trend, we created/plotted a line chart.
- We can observe that the success rate since 2013 kept increasing till 2020.
- We then use Feature Engineering to select the features that will be used in success prediction in the future module. We create the dummy variables to categorical columns.
- GitHub URL:
- https://github.com/shemsud1/Applied-Data-Science-Capstone/blob/main/jupyter-labs-eda-dataviz.ipynb

```
# HINT use groupby method on Orbit column and get the mean of Class column
df.groupby(['Orbit']).mean()['Class'].plot(kind='bar')
plt.xlabel("Orbit Type",fontsize=20)
plt.ylabel("Sucess Rate",fontsize=20)
plt.show()
```





EDA with SQL

Using SQL, we had performed many queries to get better understanding of the Spacex DataSet:

- We Load the dataset into the corresponding table in a Db2 database & execute SQL queries against the database. For instance:
  - Display the names of the unique launch sites in the space mission
  - Display the total payload mass carried by boosters launched by NASA (CRS)
  - Display average payload mass carried by booster version F9 v1.1
  - 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 successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.
- GitHub URL: https://github.com/shemsud1/Applied-Data-Science-Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- To visualize the launch data into an interactive map, we marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters(Red for 0 & Green for 1, we identified which launch sites have relatively high success rate.
- We then calculated the distances between a launch site to its proximities. We answered some questions. For instance:
    - Are launch sites in close proximity to railways, highways and coastlines?
    - Do launch sites keep certain distance away from cities?
- GitHub URL: https://github.com/shemsud1/Applied-Data-Science-Capstone/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash which allows users to input data as needed.
- We Add a dropdown list to enable Launch Site selection
- We plotted pie charts to show the total successful launches count for all sites
- We Add a slider to select payload range. We then plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- GitHub URL:

https://github.com/shemsud1/Applied-Data-Science-Capstone/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning. We plotted the confusion matrix.
- We found the best performing classification model. The model with the best accuracy score will be the best performing model.
- GitHub URL:
- https://github.com/shemsud1/Applied-Data-Science-Capstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

The results will be categorized to 3 main results which is:

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

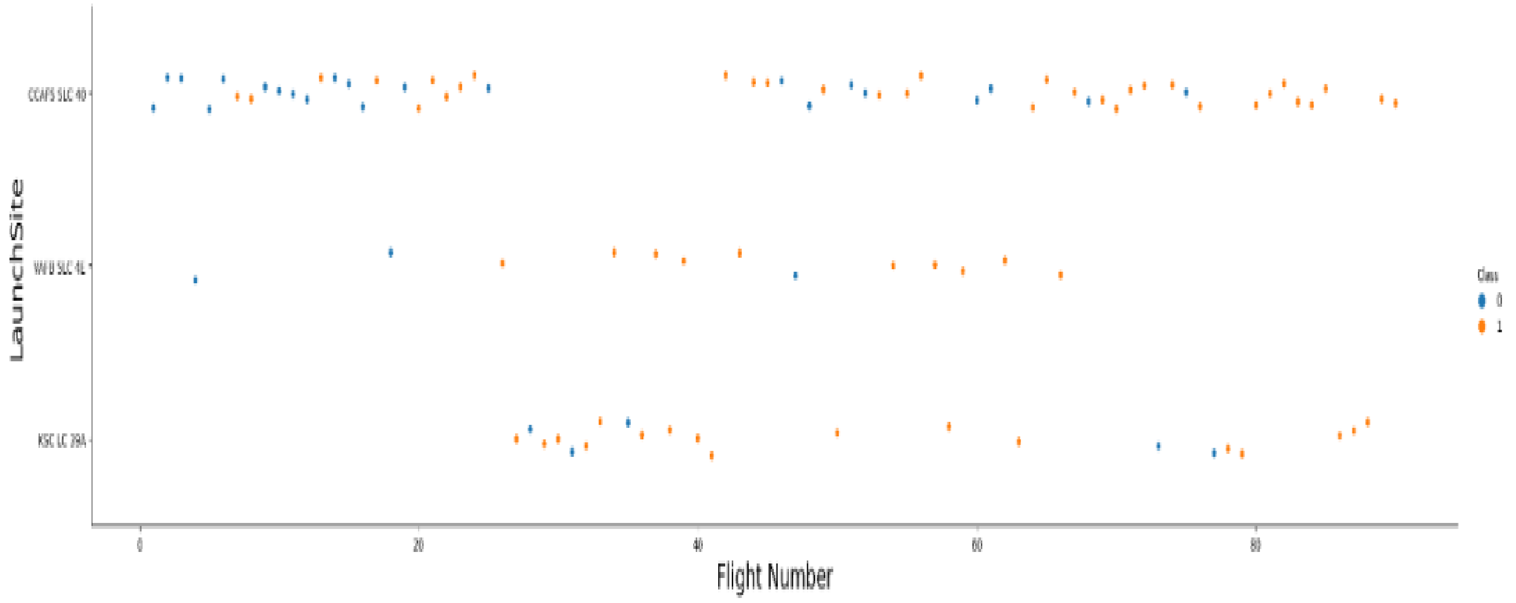
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

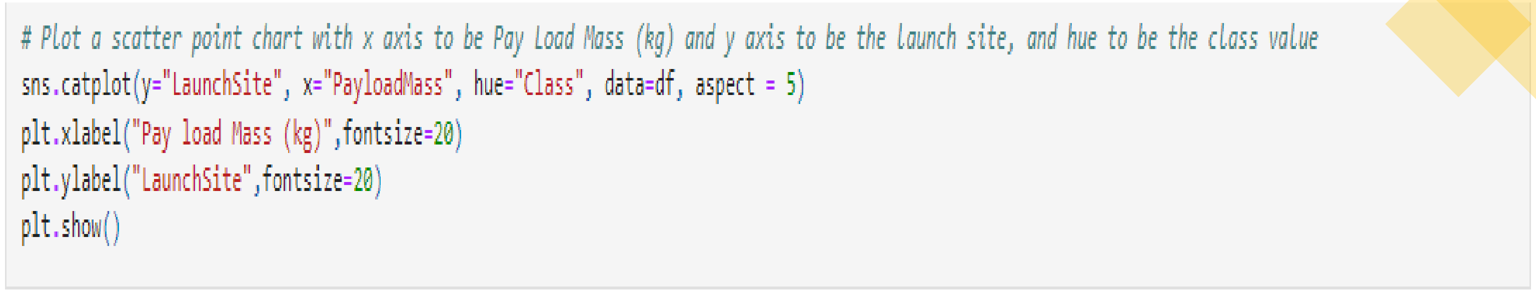
- This scatter plot shows that the larger the flights amount of the launch site, the greater the success rate will be.

```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```

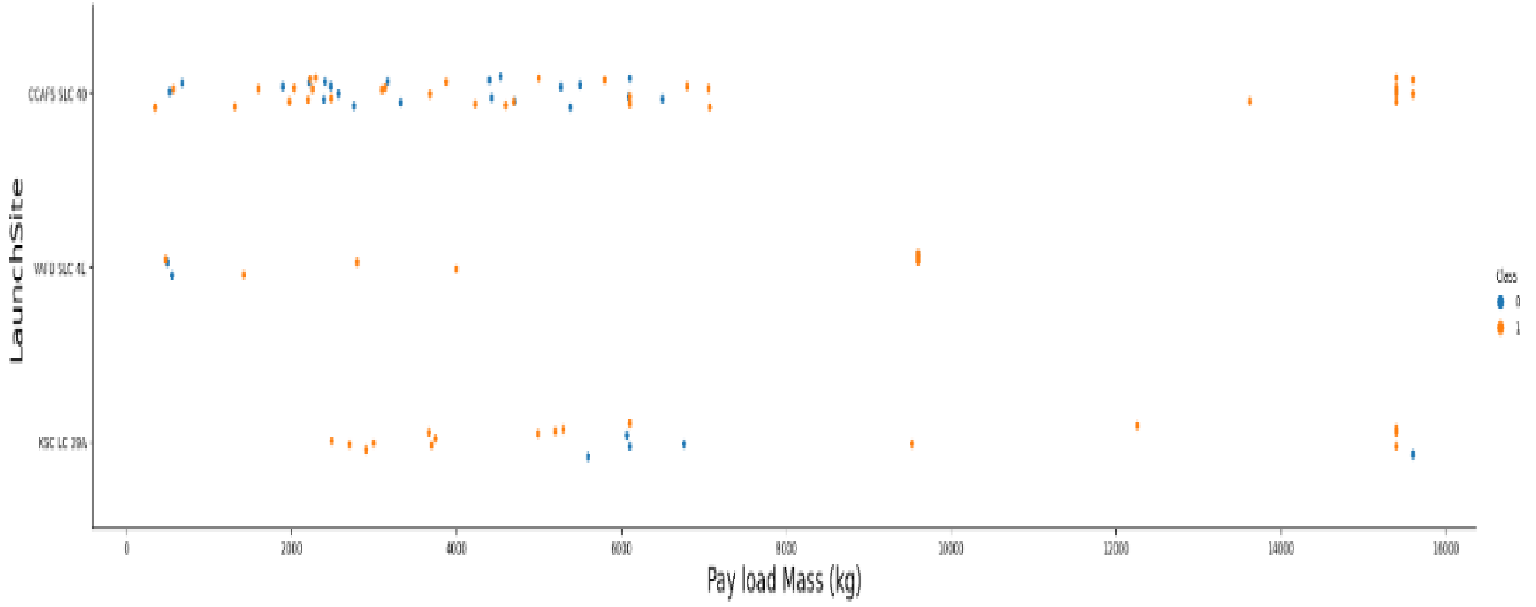


Payload vs. Launch Site

- This scatter plot shows, the greater the payload mass for the launch site CCAFS SLC 40, the higher the success rate for the rocket.
- The greater the payload mass (greater than 7000 Kg), the higher the success rate for the rocket. But there is no clear pattern to take a decision, if the launch site is dependent on Payload Mass for a success launch.
- For the VAFB-SLC launch site there are no rockets launched for heavy-payload mass(greater than 10000).

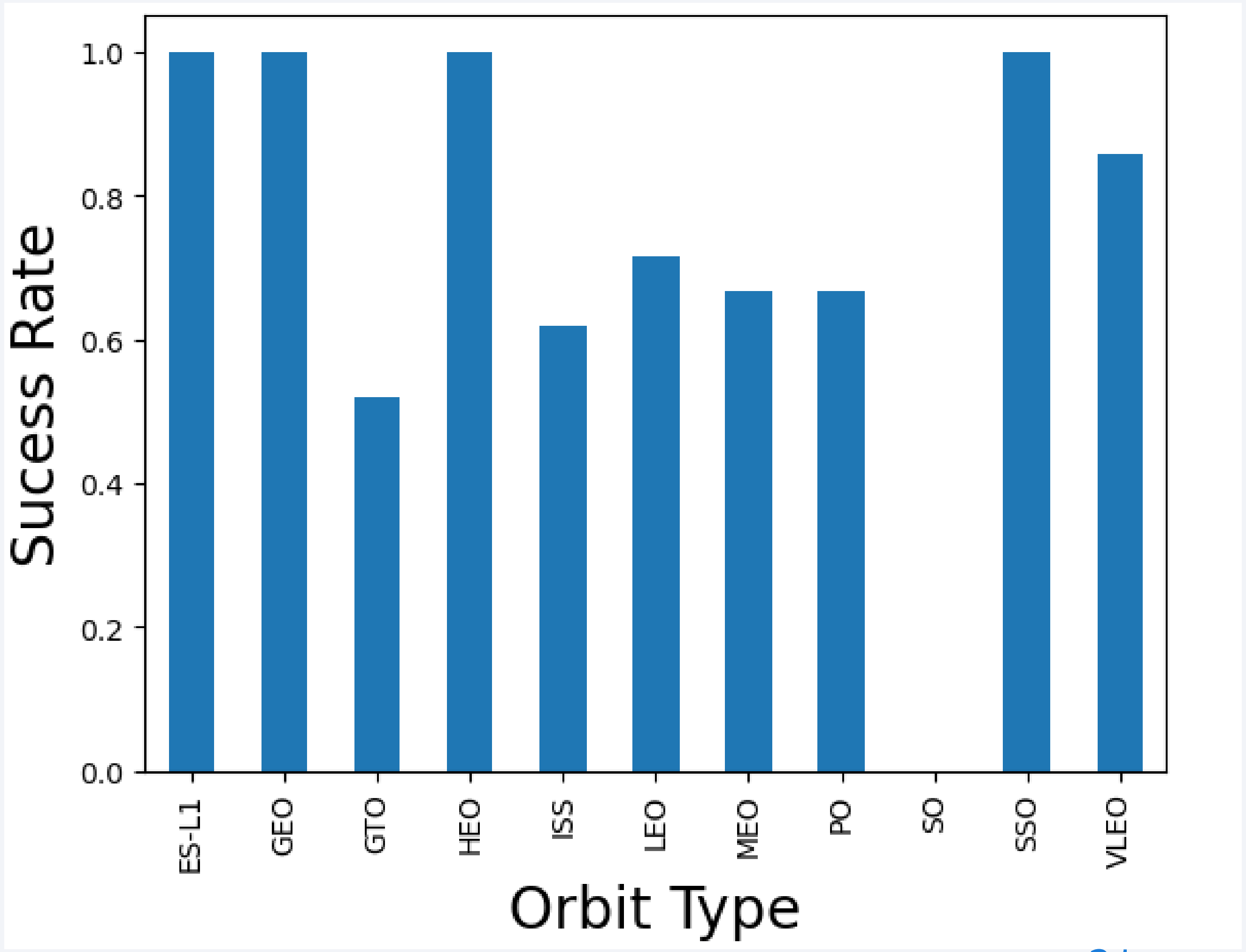


```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Pay load Mass (kg)",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```



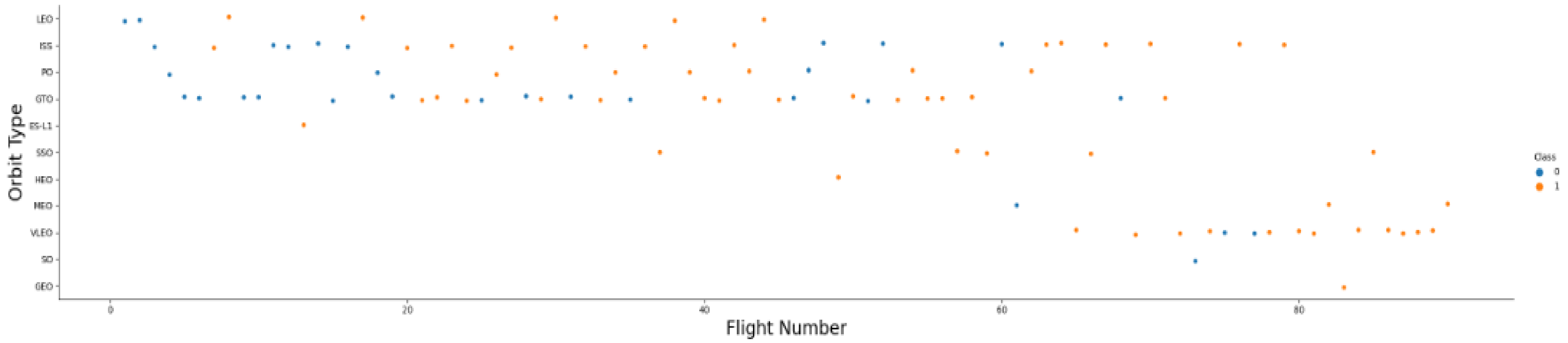
Success Rate vs. Orbit Type

- From this bar chart , we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate while SO orbit produced 0% rate of success.



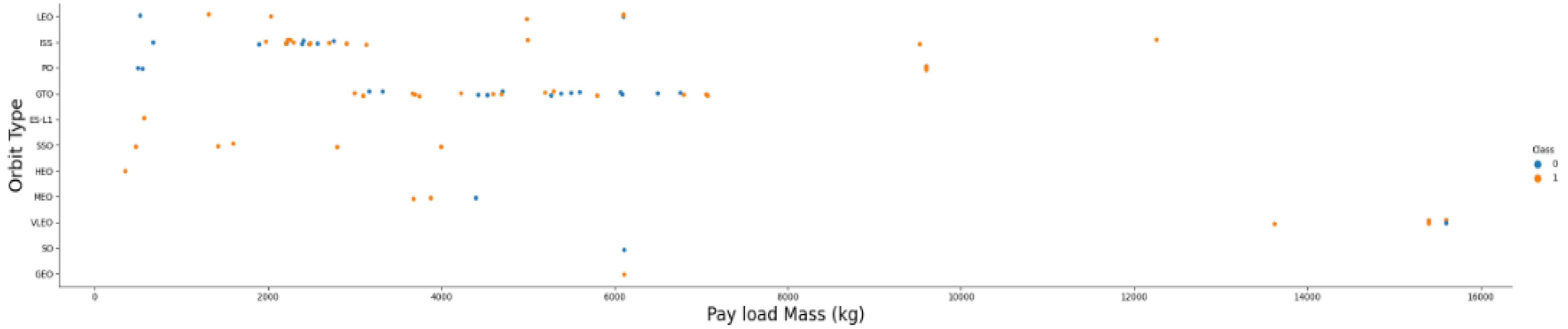
Flight Number vs. Orbit Type

- We observe that for the LEO orbit, the success is related to the number of flights (i.e. success increases with the number of flights)
- On the other hand, it seems there is no relationship between flight number and the GTO orbit.



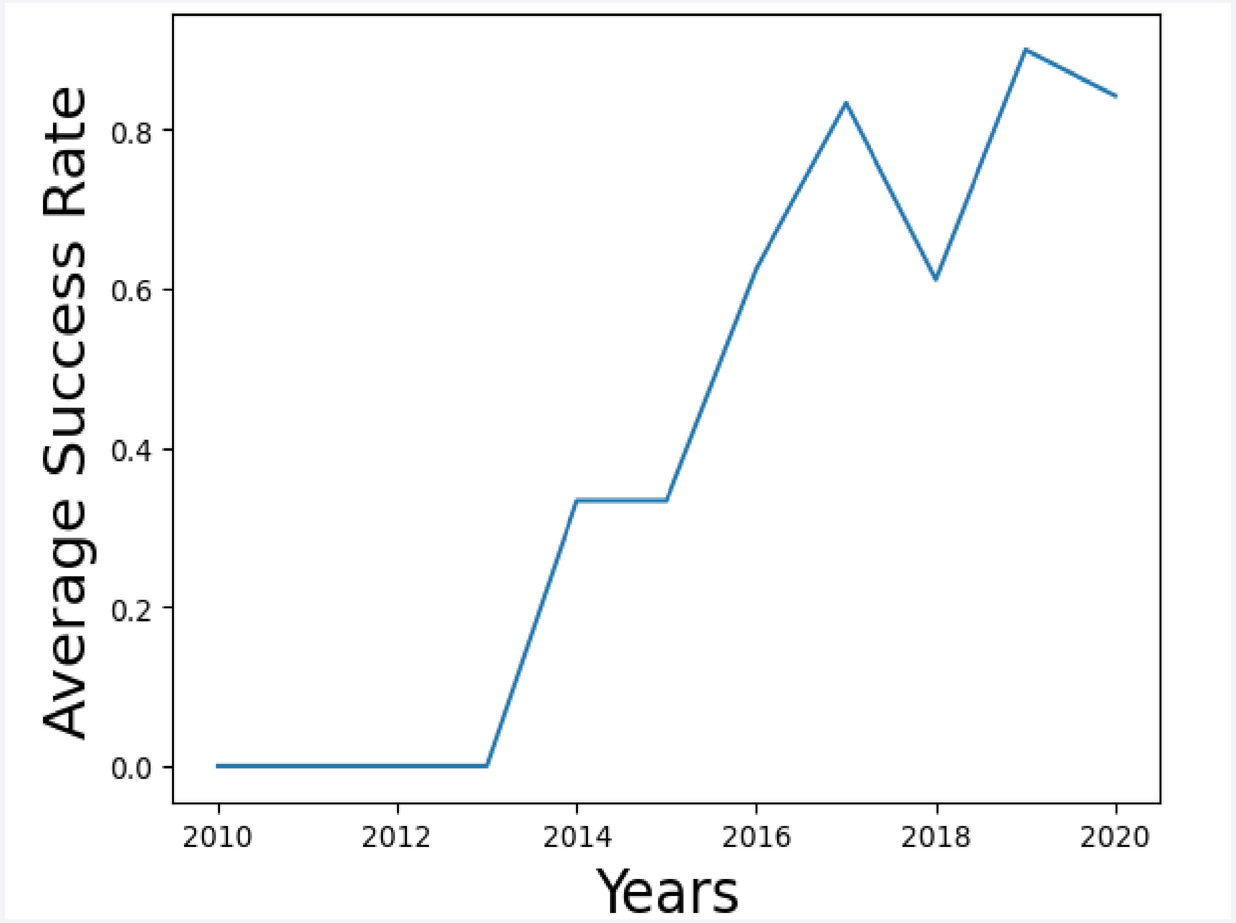
Payload vs. Orbit Type

- We can observe that with heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.
- We can also observe that heavy payloads have a negative influence on MEO and VLEO orbits.
- 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

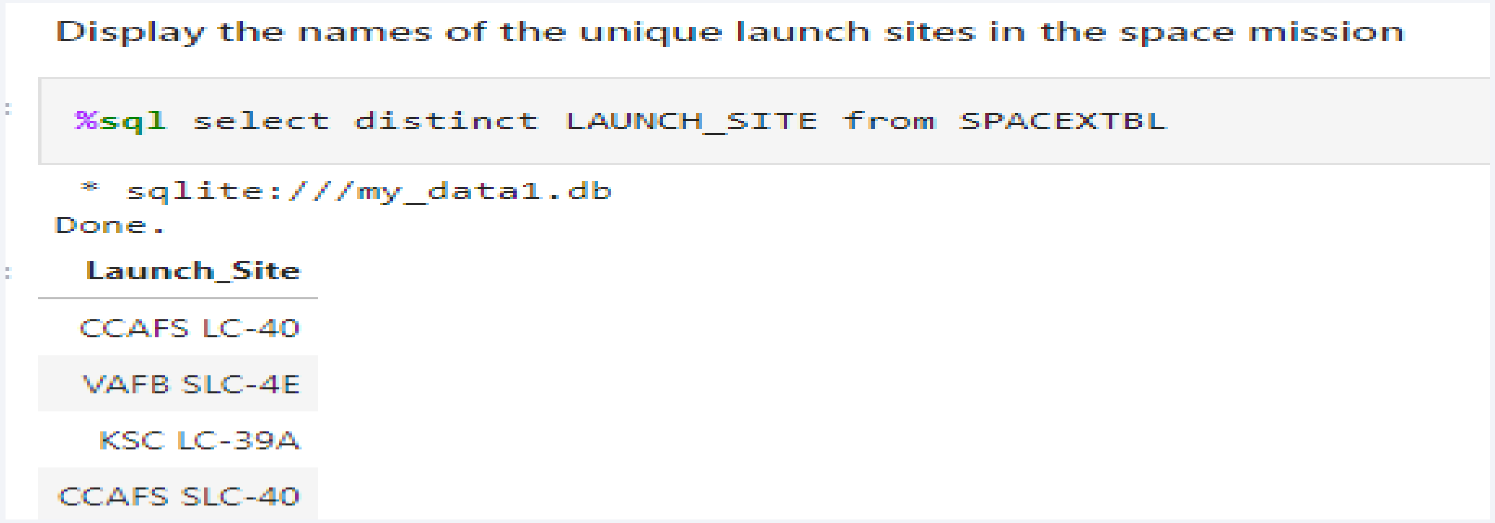
- We can observe that the success rate since 2013 kept increasing till 2020.



All Launch Site Names

Explanation:

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


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

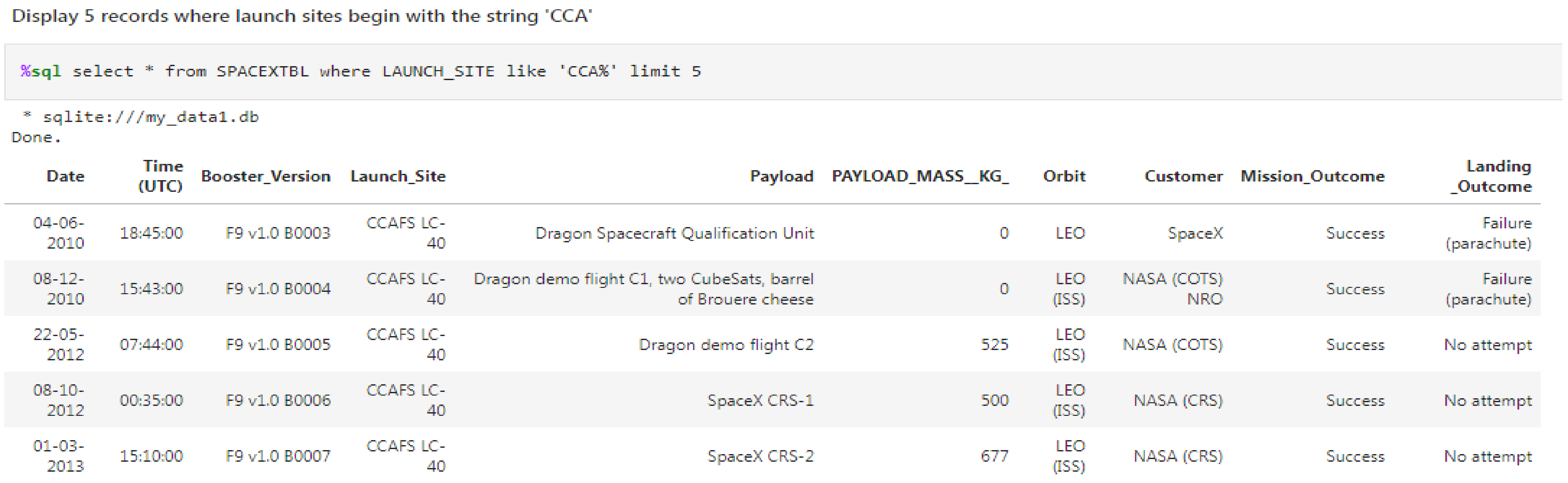
```
%sql select distinct LAUNCH_SITE from SPACEXTBL
```

```
 * 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'

- **Explanation:**
- We used the key word 'LIMIT 5' in the query to fetch 5 records from the table SPACEX and with condition LIKE keyword and wildcard 'CCA%'. The percentage at the end suggests that the launch_site name must start with CCA characters.



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

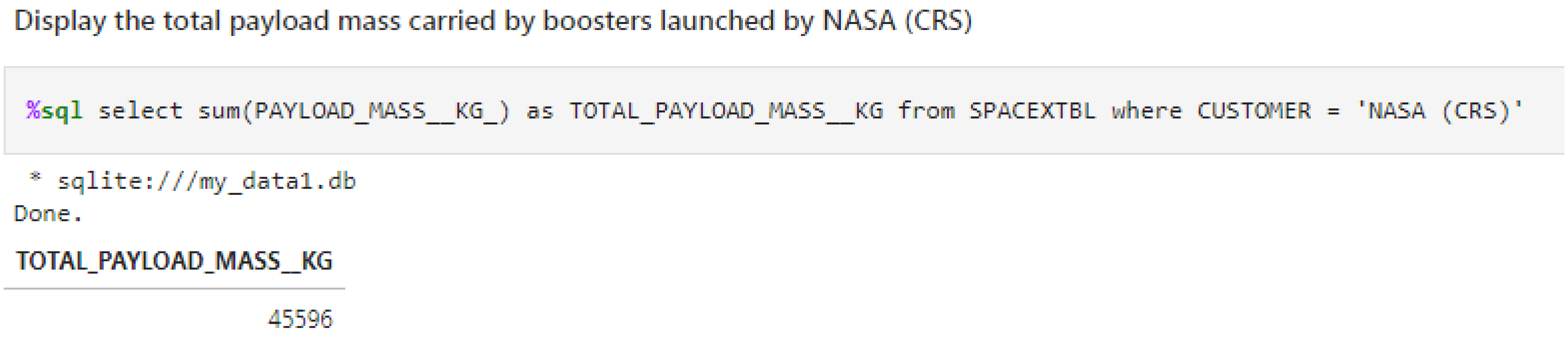
```
%sql select * from SPACEXTBL where LAUNCH_SITE like 'CCA%' limit 5
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- **Explanation:**
- We calculated the total payload carried by boosters from NASA as 45596. Using the function SUM() to calculate the total in the column PAYLOAD_MASS__KG_ and the WHERE clause filters the data to fetch Customer's by name "NASA (CRS)".



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

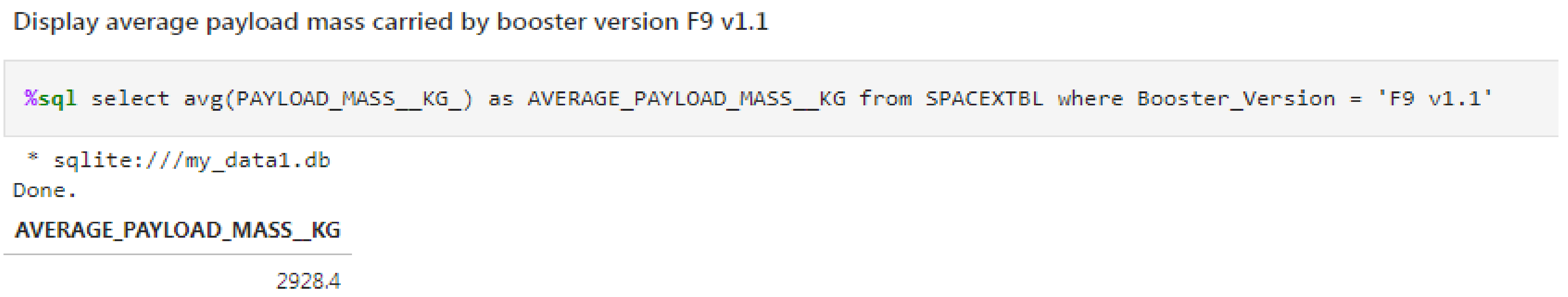
```
%sql select sum(PAYLOAD_MASS__KG_) as TOTAL_PAYLOAD_MASS__KG from SPACEXTBL where CUSTOMER = 'NASA (CRS)'
```

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

TOTAL_PAYLOAD_MASS__KG
45596

Average Payload Mass by F9 v1.1

- **Explanation:**
- We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4. Using the function AVG() to calculate the average in the column PAYLOAD_MASS__KG_ and the WHERE clause filters the dataset to only perform calculations on Booster version F9 v1.1.



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

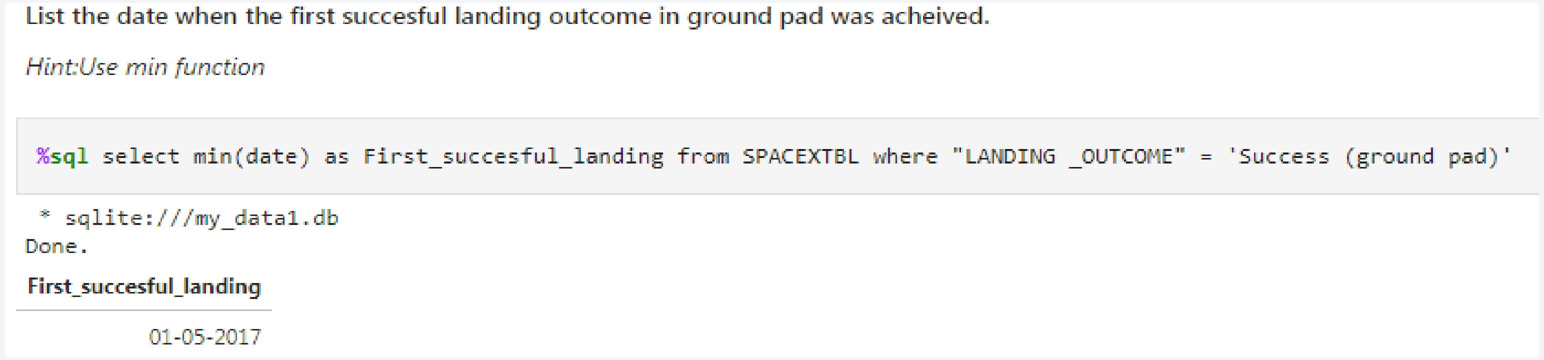
```
%sql select avg(PAYLOAD_MASS__KG_) as AVERAGE_PAYLOAD_MASS__KG from SPACEXTBL where Booster_Version = 'F9 v1.1'
```

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

AVERAGE_PAYLOAD_MASS__KG
2928.4

First Successful Ground Landing Date

- **Explanation:**
- We used the min() function to find the result. We observed that the dates of the first successful landing outcome on ground pad was 5th January 2017



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

Hint:Use min function

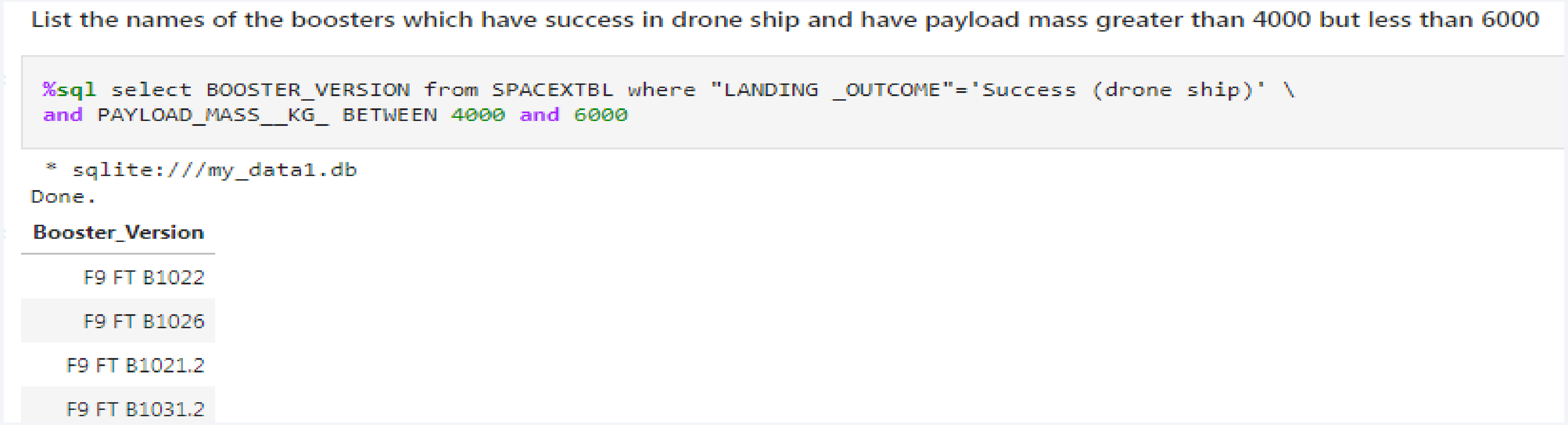
```
%sql select min(date) as First_succesful_landing from SPACEXTBL where "LANDING _OUTCOME" = 'Success (ground pad)'
```

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

First_succesful_landing
01-05-2017

Successful Drone Ship Landing with Payload between 4000 and 6000

Explanation: 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



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

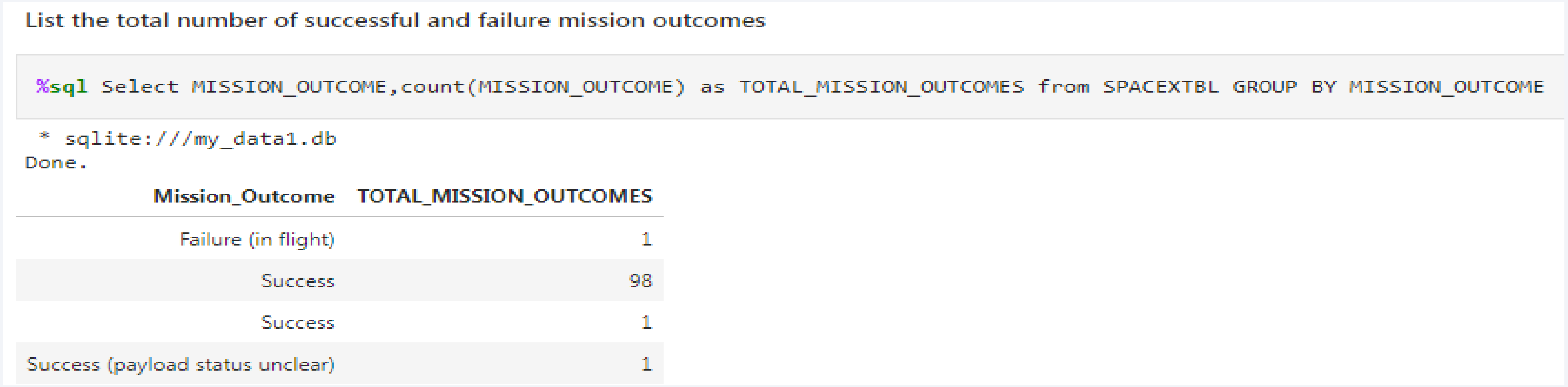
```
%sql select BOOSTER_VERSION from SPACEXTBL where "LANDING _OUTCOME"='Success (drone ship)' \
and PAYLOAD_MASS__KG_ BETWEEN 4000 and 6000
```

```
 * 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

Explanation: We used count() function on Mission_Outcome column and group them on the same column to get a success or a failure total counter.



List the total number of successful and failure mission outcomes

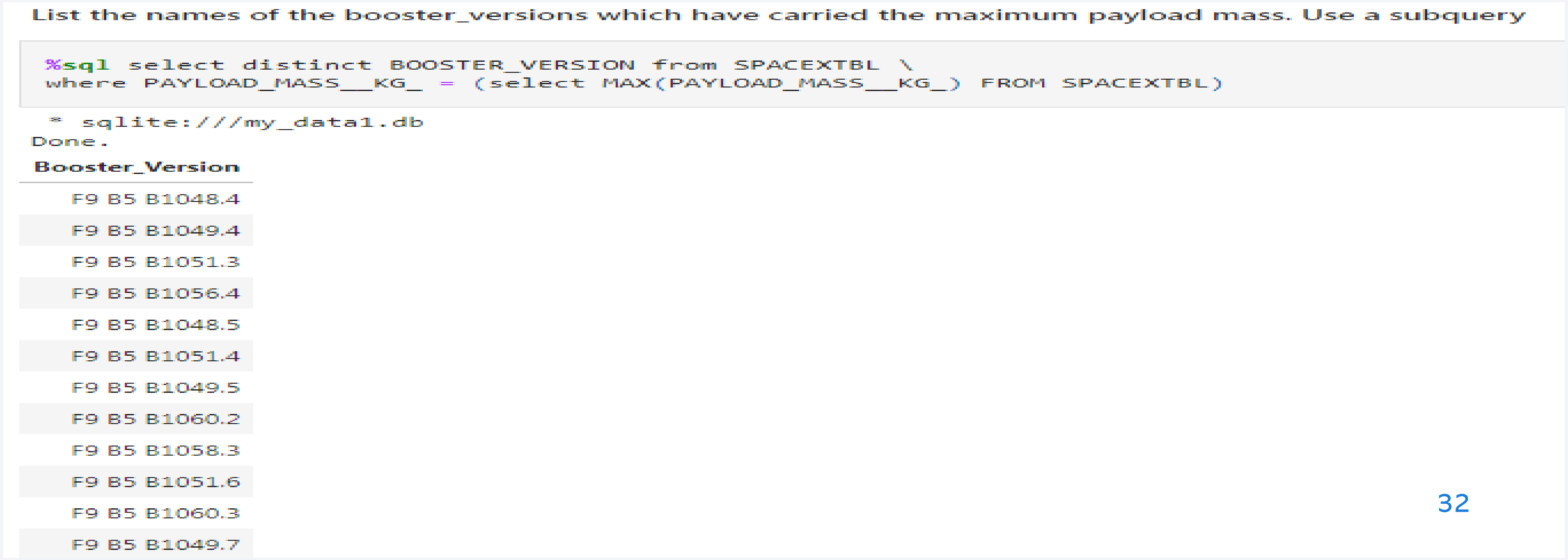
```
%sql Select MISSION_OUTCOME,count(MISSION_OUTCOME) as TOTAL_MISSION_OUTCOMES from SPACEXTBL GROUP BY MISSION_OUTCOME
```

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

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

Boosters Carried Maximum Payload

Explanation: We used the MAX() function to work out the maximum payload in the column PAYLOAD_MASS__KG_ in the sub query. The WHERE clause filters the Booster Version which had that maximum payload.



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

```
%sql select distinct BOOSTER_VERSION from SPACEXTBL \
where PAYLOAD_MASS__KG_ = (select MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
```

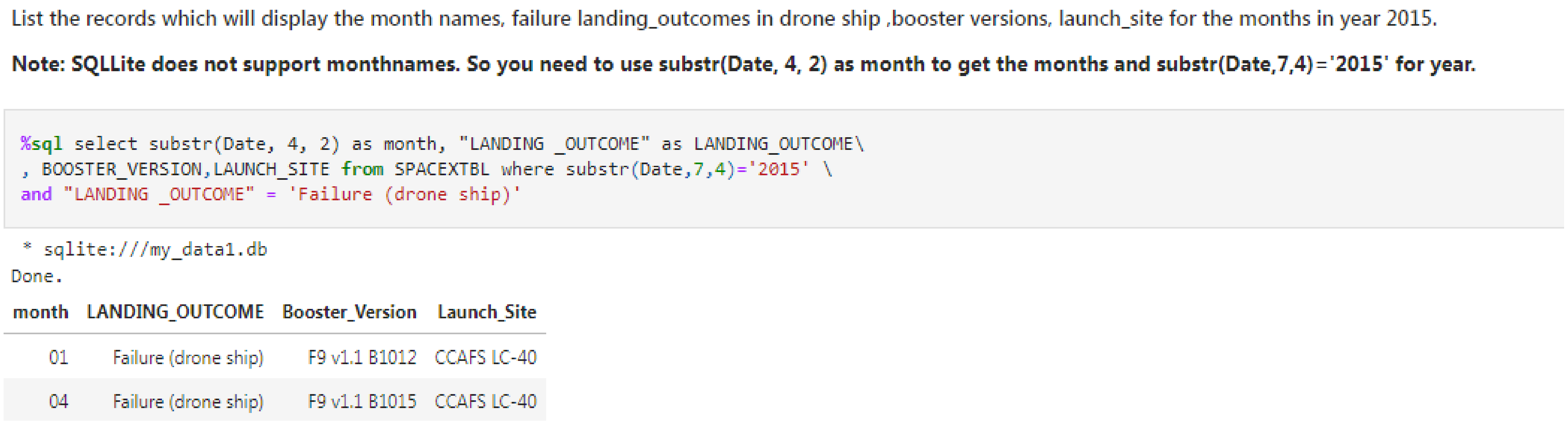
```
 * sqlite:///my_data1.db
Done.
```

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

Explanation:

- We need to list the records which will display the name of the Month, failure landing_outcomes in drone ship, booster versions, launch_site for months in the year 2015. We also used substr() function to extract months and year from the Date column.



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

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

```
%sql select substr(Date, 4, 2) as month, "LANDING _OUTCOME" as LANDING_OUTCOME\
, BOOSTER_VERSION,LAUNCH_SITE from SPACEXTBL where substr(Date,7,4)='2015' \
and "LANDING _OUTCOME" = 'Failure (drone ship)'
```

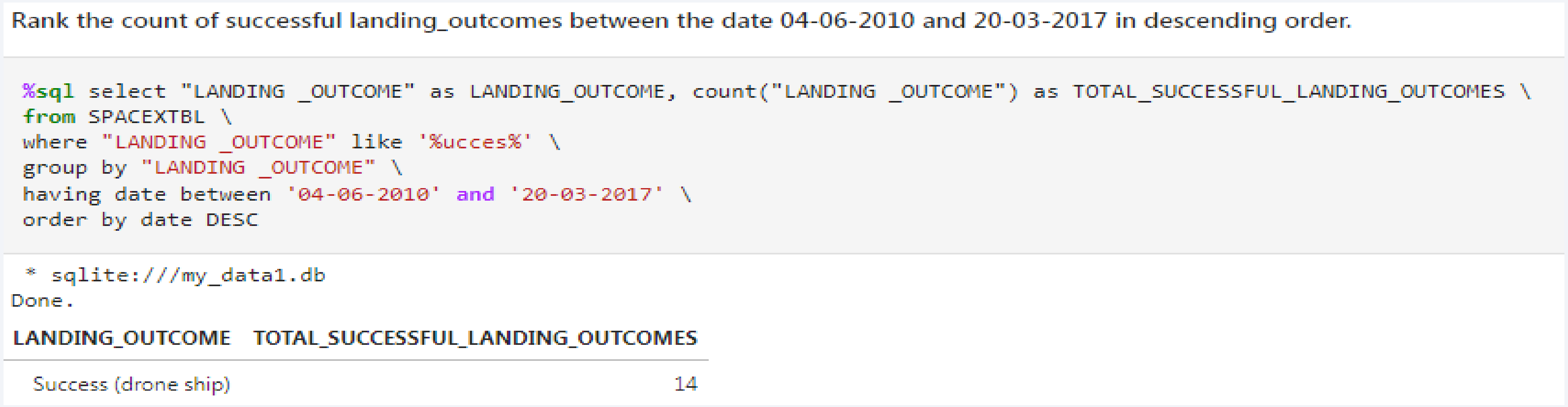
```
 * sqlite:///my_data1.db
Done.
```

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 WHERE clause and like '%ucces%' wildcard to filter for successful landing outcomes .

We applied the GROUP BY clause to group the landing outcomes and the having clause to filter date between '04-06-2010' and '20-03-2017' . Finally, we applied the ORDER BY clause to order the grouped landing outcome in descending order.



Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

```
%sql select "LANDING _OUTCOME" as LANDING_OUTCOME, count("LANDING _OUTCOME") as TOTAL_SUCCESSFUL_LANDING_OUTCOMES \
from SPACEXTBL \
where "LANDING _OUTCOME" like '%ucces%' \
group by "LANDING _OUTCOME" \
having date between '04-06-2010' and '20-03-2017' \
order by date DESC
```

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

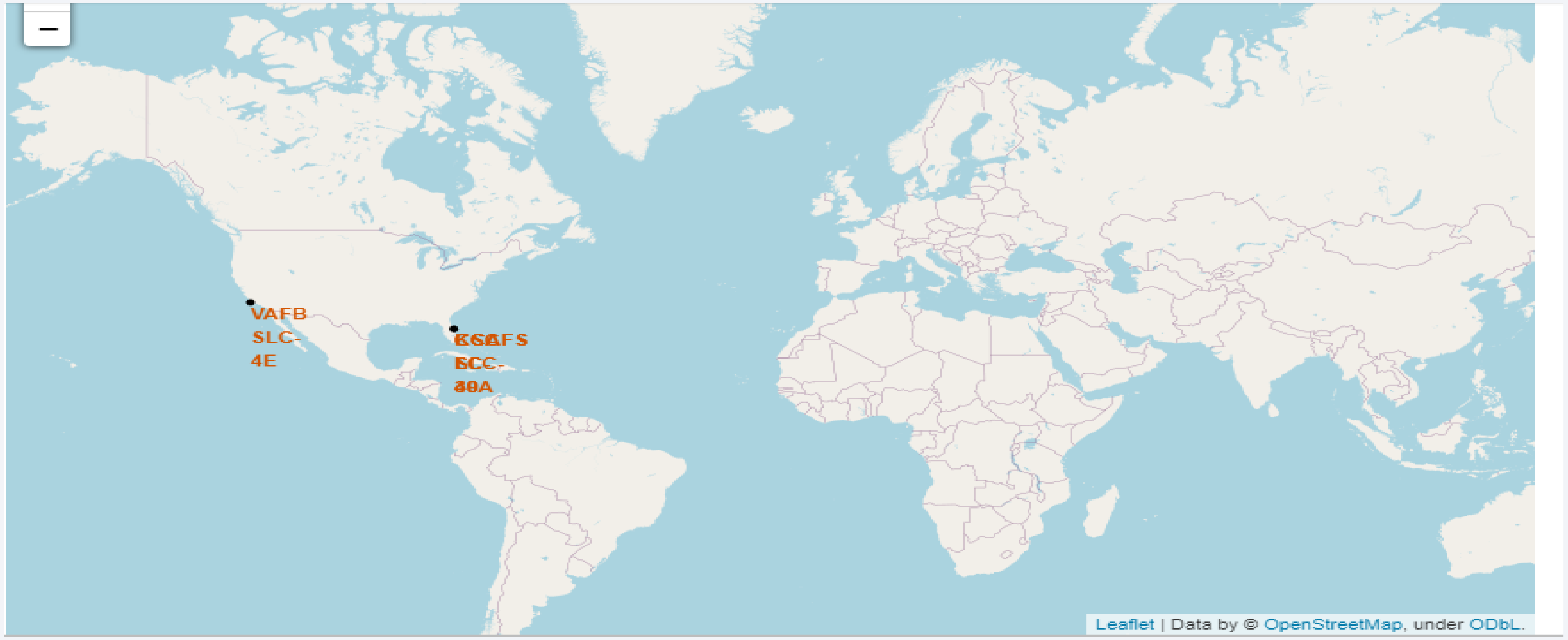
LANDING_OUTCOME	TOTAL_SUCCESSFUL_LANDING_OUTCOMES
Success (drone ship)	14

Section 3

Launch Sites Proximities Analysis

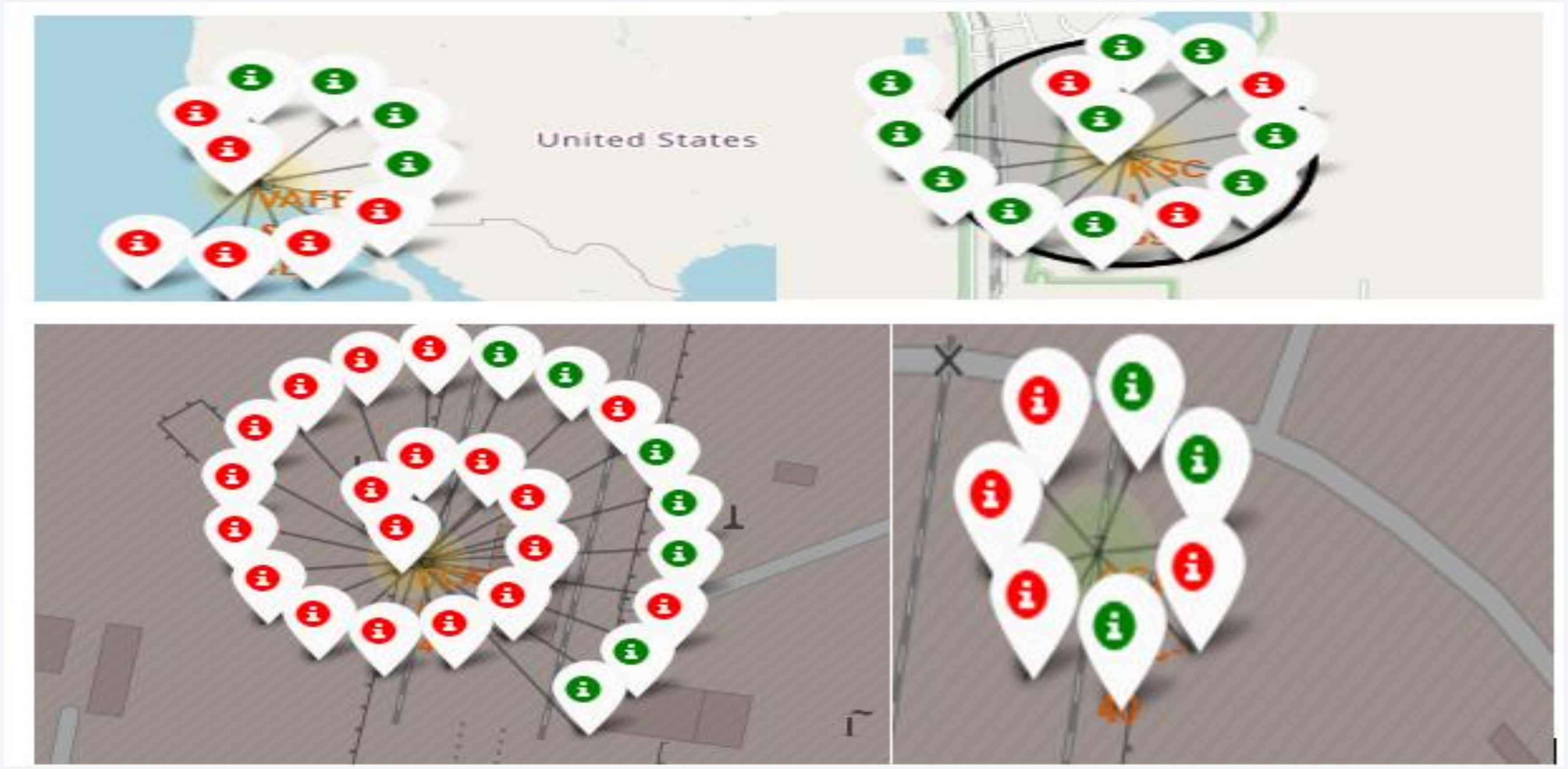
All launch sites' location markers on a global map

- We can see that all the SpaceX launch sites are located inside the United States California and Florida Regions



Markers showing launch sites with color labels

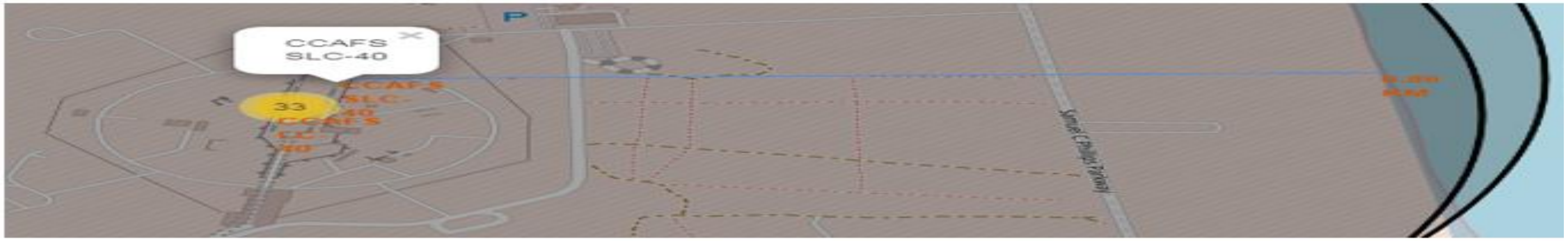
- Green Marker shows successful launches and Red Marker shows failures launches

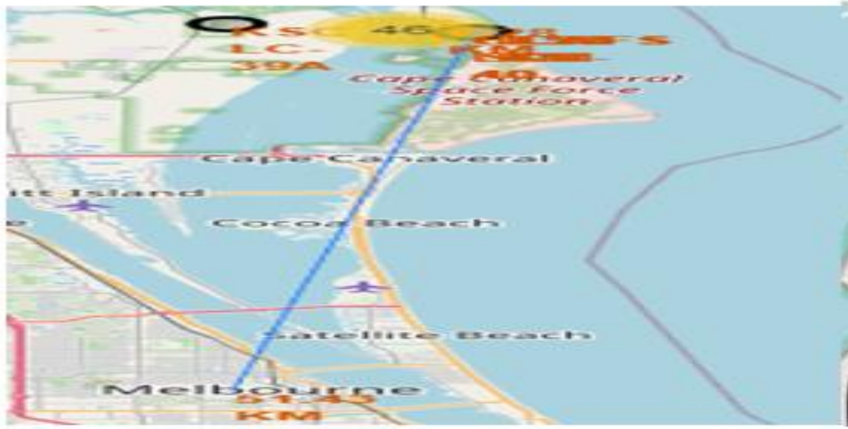


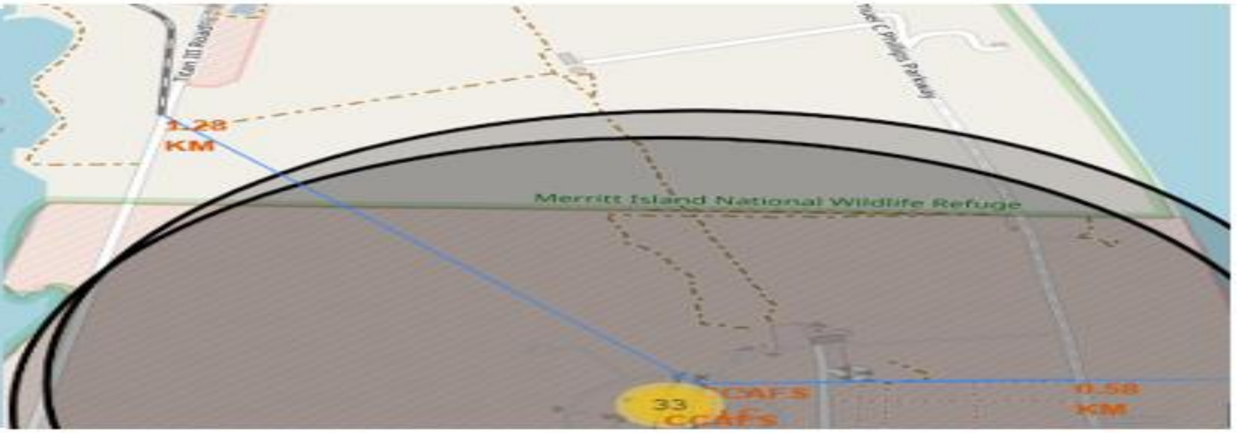
Launch Sites Distance to Landmarks

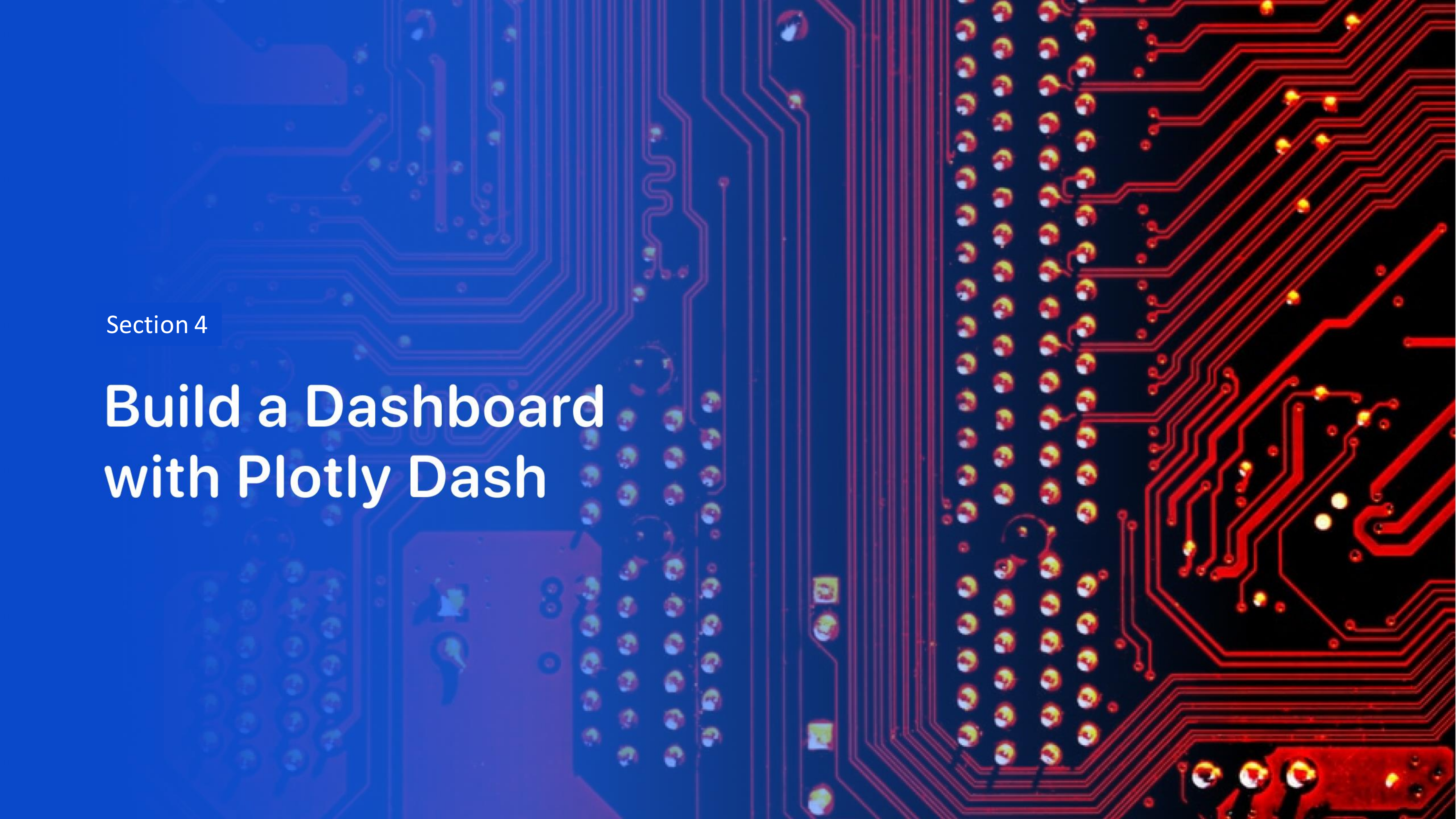
The distance calculated was as follows:

- distance_highway = 0.5834695366934144 km
- distance_railroad = 1.2845344718142522 km
- distance_city = 51.43416999517233 km







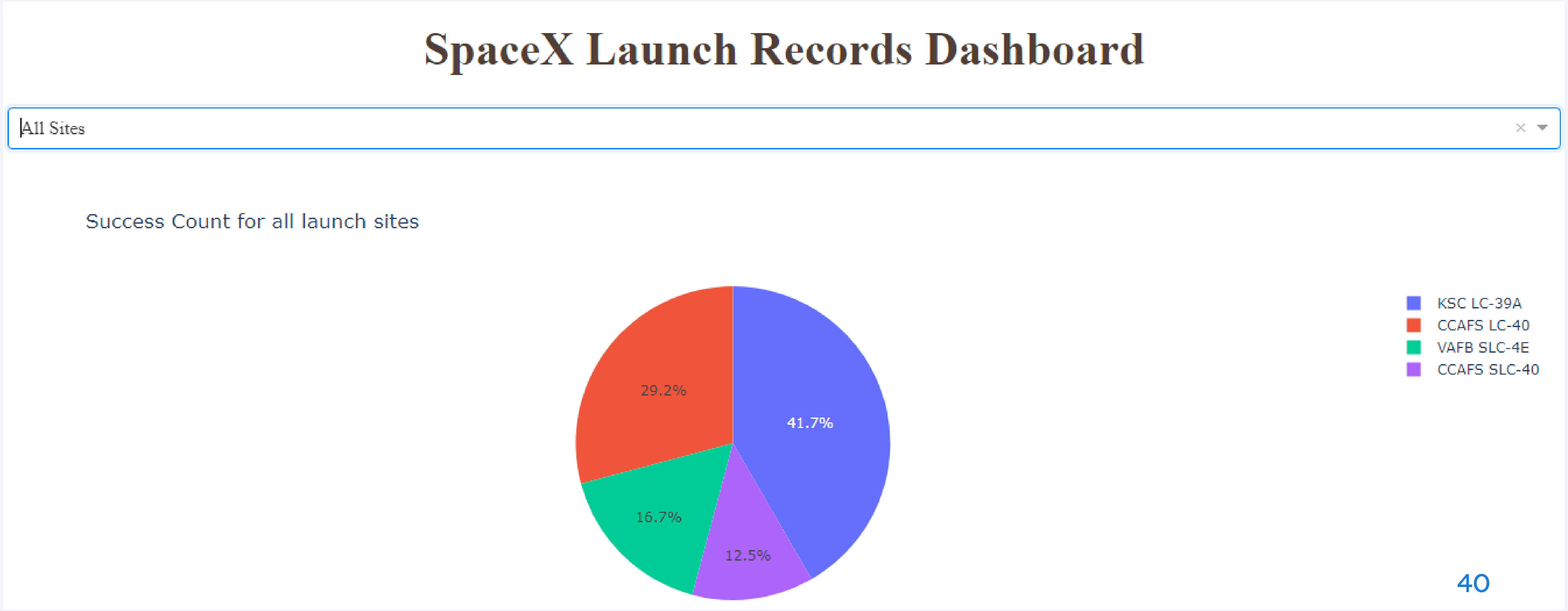


Section 4

Build a Dashboard with Plotly Dash

Launch success count for all sites

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

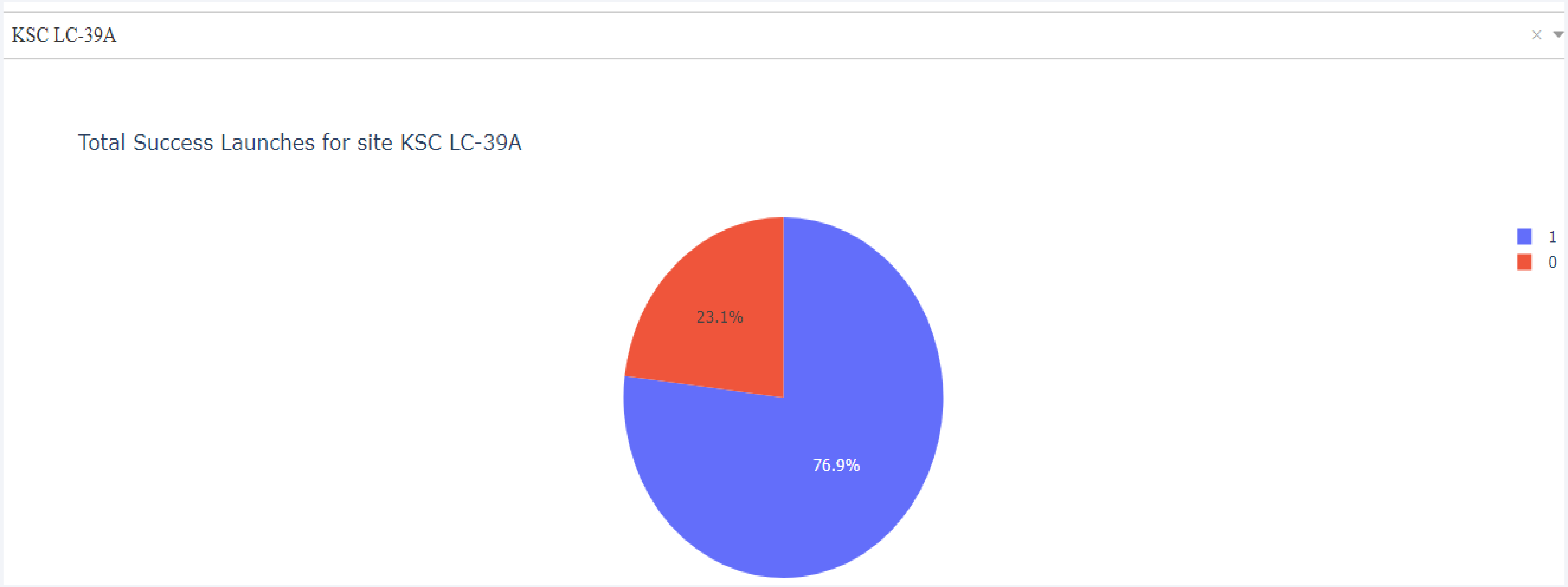


SpaceX Launch Records Dashboard

All Sites

Launch site with highest launch success ratio

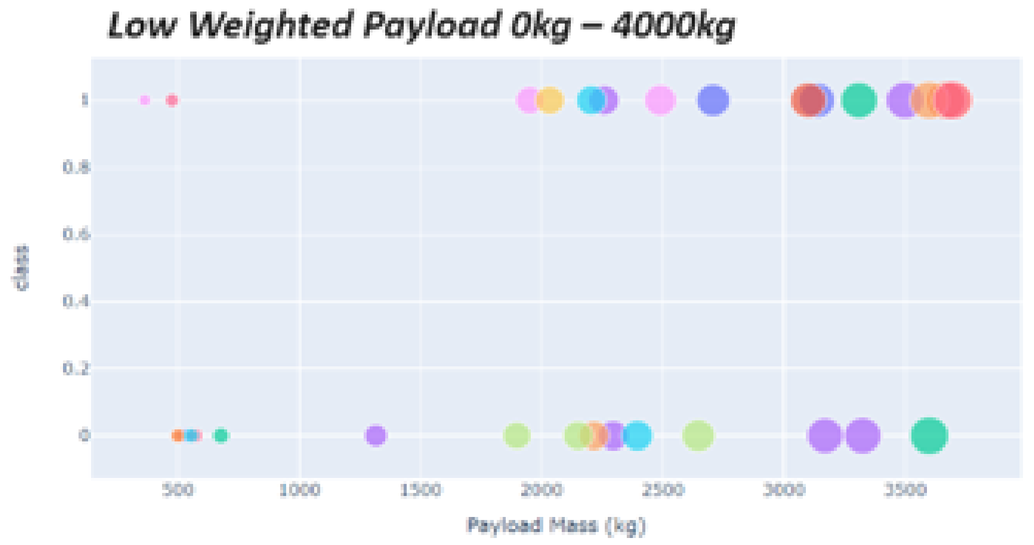
KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate.

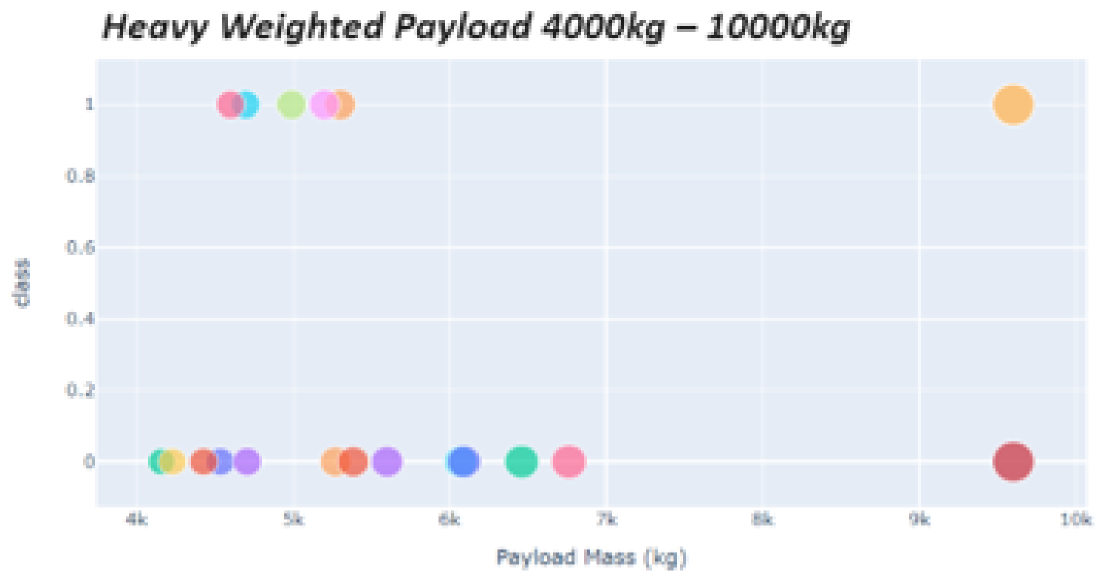

KSC LC-39A

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

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



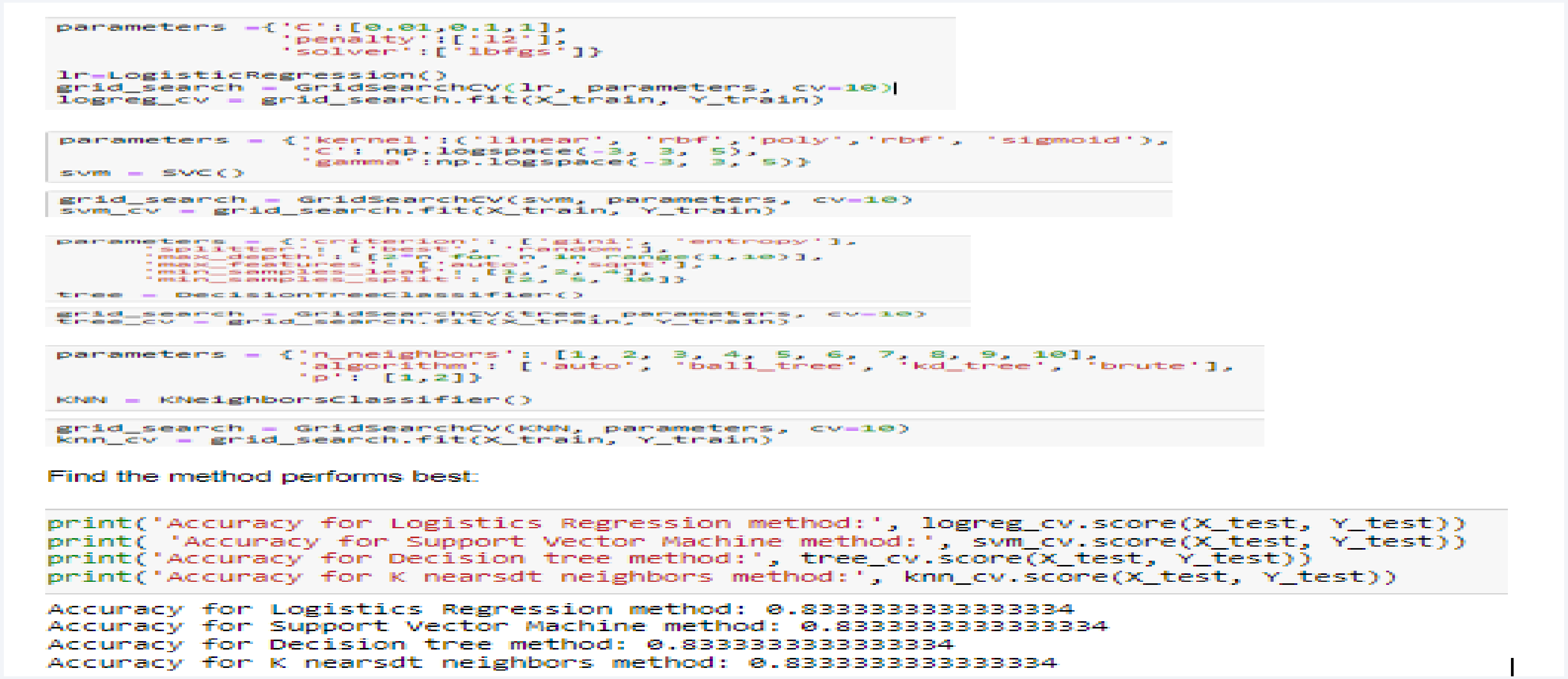


Section 5

Predictive Analysis (Classification)

Classification Accuracy

As we can see below, it seems that ALL the algorithms have equal accuracy



```
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}

lr=LogisticRegression()
grid_search = GridSearchCV(lr, parameters, cv=10)
logreg_cv = grid_search.fit(X_train, Y_train)
```

```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
```

```
grid_search = GridSearchCV(svm, parameters, cv=10)
svm_cv = grid_search.fit(X_train, Y_train)
```

```
parameters = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['auto', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}

tree = DecisionTreeClassifier()
```

```
grid_search = GridSearchCV(tree, parameters, cv=10)
tree_cv = grid_search.fit(X_train, Y_train)
```

```
parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'p': [1,2]}

KNN = KNeighborsClassifier()
```

```
grid_search = GridSearchCV(KNN, parameters, cv=10)
knn_cv = grid_search.fit(X_train, Y_train)
```

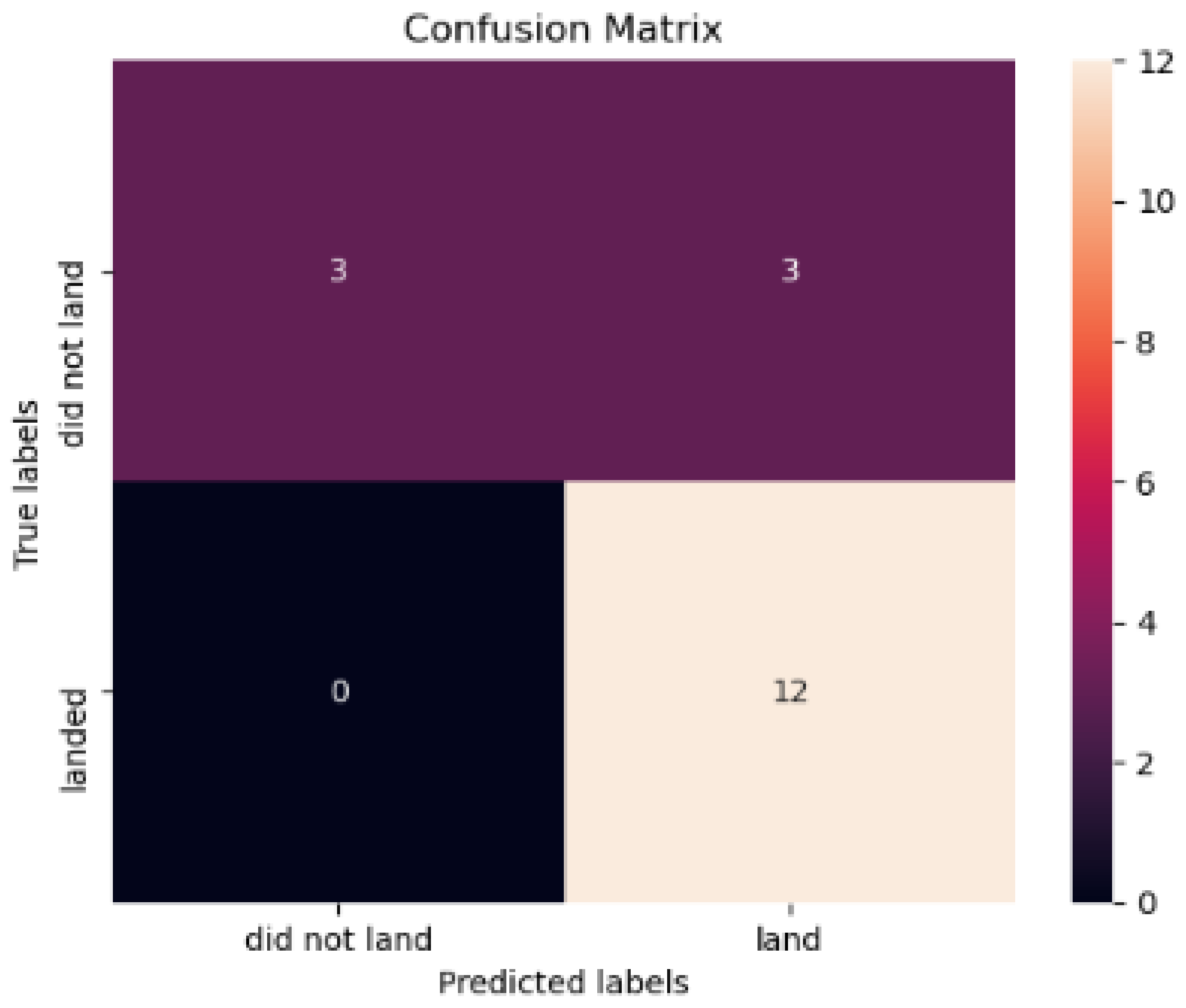
Find the method performs best:

```
print('Accuracy for Logistics Regression method:', logreg_cv.score(X_test, Y_test))
print( 'Accuracy for Support Vector Machine method:', svm_cv.score(X_test, Y_test))
print('Accuracy for Decision tree method:', tree_cv.score(X_test, Y_test))
print('Accuracy for K nearsdt neighbors method:', knn_cv.score(X_test, Y_test))
```

```
Accuracy for Logistics Regression method: 0.8333333333333334
Accuracy for Support Vector Machine method: 0.8333333333333334
Accuracy for Decision tree method: 0.8333333333333334
Accuracy for K nearsdt neighbors method: 0.8333333333333334
```

Confusion Matrix

- Examining the confusion matrix, we see that logistic regression can distinguish between the different classes. We see that the major problem is false positives.



Conclusions

We can conclude that:

- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate while SO orbit produced 0% rate of success.
- 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. We can say that success rates for SpaceX launches has been increasing relatively with time and it looks soon they will reach the required target.
- KSC LC-39A had the most successful launches of any sites.

Thank you!