IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Henrique Dalla Valle de Siqueira
August 21, 2025

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Executive Summary

- Summary of methodologies
  - Data collection
  - Data wrangling
  - EDA with data visualization
  - EDA with SQL
  - Building an interactive map with Folium
  - Building a Dashboard with Plotly Dash
  - Predictive analysis (Classification)
- Summary of all results
  - EDA results
  - Interactive analytics
  - Predictive analysis

Introduction

Project background and context

This project is part of an ambitious mission to revolutionize the space industry. As we enter the commercial space age, companies like Virgin Galactic, Rocket Lab, Blue Origin, and SpaceX are making space travel more accessible and affordable. SpaceX, in particular, has become a leader in this field due to its innovative approach to reducing the cost of space travel. One of the key factors contributing to their success is the reusability of their Falcon 9 rocket's first stage. Unlike traditional rockets, SpaceX can recover and reuse this critical part of the rocket, significantly cutting down on launch costs.

Our project aims to help a new rocket company, SpaceY, which is founded to compete directly with SpaceX. We will focus on predicting whether SpaceX's first-stage rocket will land successfully, as this directly affects the cost of their launches. By using data science and machine learning, we will develop models and dashboards to inform strategic decisions for SpaceY.

Business Problem

SpaceX has revolutionized the space industry with its cost-efficient approach to launching rockets. While a traditional rocket launch could cost upwards of 165 million dollars, SpaceX has managed to reduce its launch costs to around 62 million dollars, primarily through the reusability of the Falcon 9's first stage. This reusability saves millions, as the first stage alone would normally cost over 15 million dollars to build and launch.

However, there are instances when SpaceX cannot recover the first stage, due to factors such as heavy payloads or specific mission requirements. This results in significant additional costs. The business problem, therefore, is predicting whether the first stage will land successfully or not, as this decision directly impacts the financial outcome of the launch. By using machine learning to forecast the likelihood of a successful landing, we can provide valuable insights to SpaceY and potentially help them reduce their own launch costs.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data sourced from SpaceX Open Source REST API and Wikipedia's Falcon 9 Launches page via web scraping.
- Perform data wrangling
  - Cleaned and processed data using One-Hot Encoding for categorical variables.
  - Removed unnecessary or missing information.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Applied Logistic Regression, KNN, SVM, and Decision Tree models.
  - Evaluated model performance to identify the best classification method.

Data Collection

The datasets were gathered using two main approaches:

Request to the SpaceX API:

- Historical launch data was retrieved from SpaceX's public open-source API.
- A GET request was used to collect the data, focusing exclusively on Falcon 9 missions.
- Missing payload weights for classified missions were estimated using average values to ensure consistency.

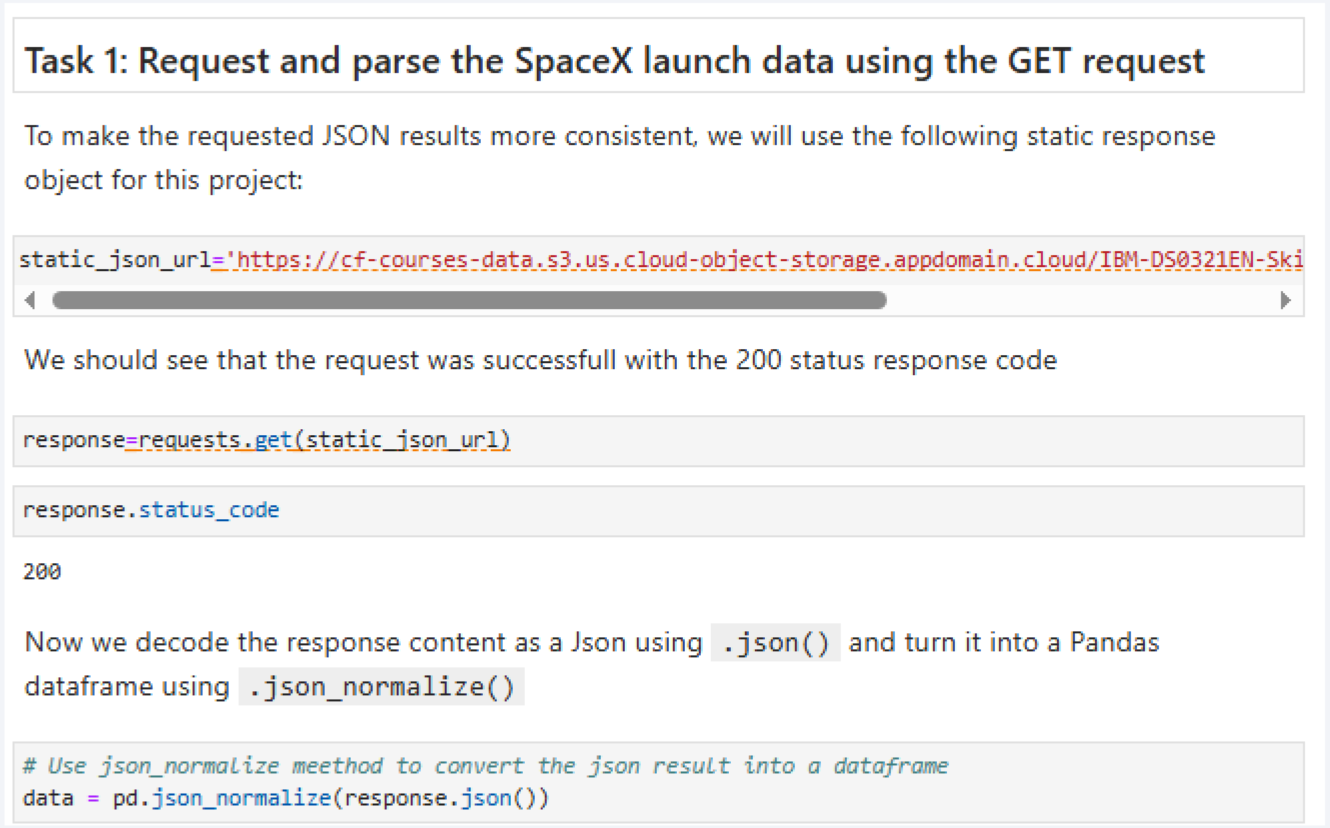
Web Scraping:

- Additional data was sourced from Wikipedia's page listing Falcon 9 and Falcon Heavy launches.
- The HTML table was accessed through a direct link.
- The launch data was extracted, parsed, and transformed into a Pandas DataFrame for analysis.

Data Collection – SpaceX API

1) Requested and parsed SpaceX launch data using GET request from a static JSON endpoint.

Data was requested from a static JSON URL simulating a SpaceX API call. The JSON response was parsed into a Pandas DataFrame for further processing.



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:

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

We should see that the request was successfull with the 200 status response code

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

```
response.status_code
```

```
200
```

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

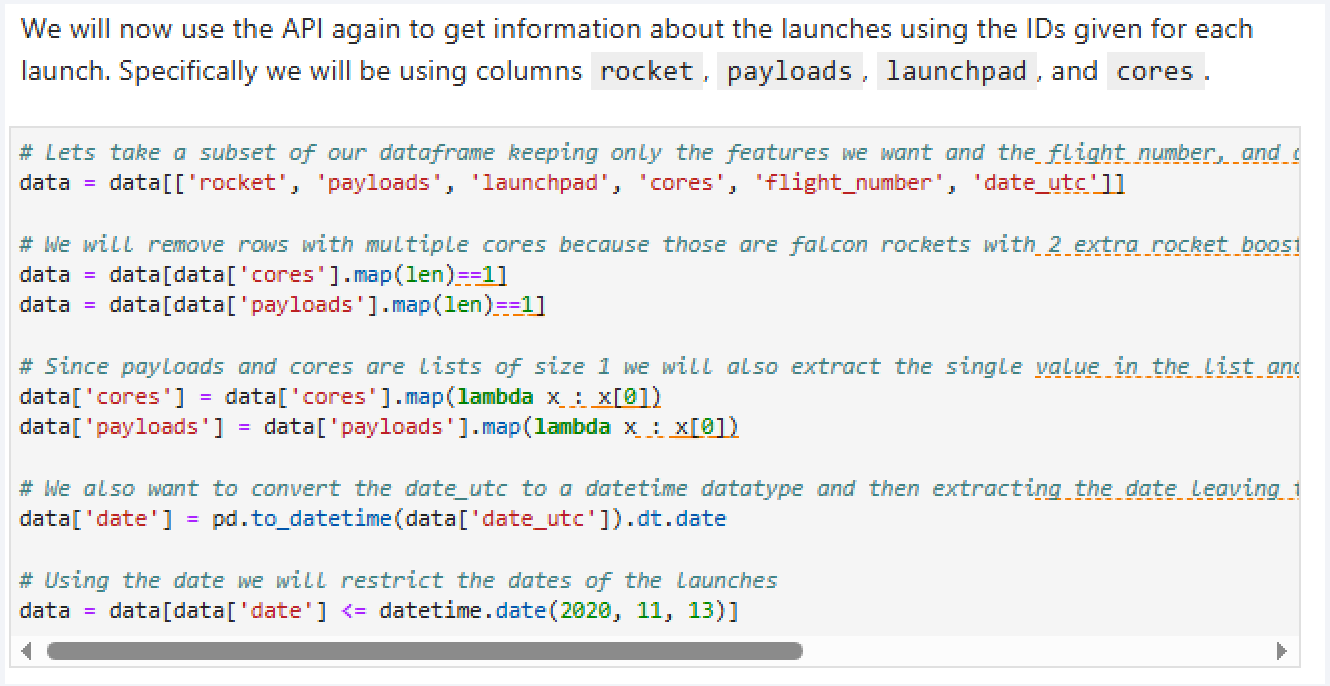
```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

Click here to check on my GitHub

Data Collection – SpaceX API

2) Filtered valid entries and cleaned data for single-core and single-payload missions.

Selected only relevant columns and filtered out records with multiple cores or payloads. Dates were also formatted and filtered to include launches before November 2020.



We will now use the API again to get information about the launches using the IDs given for each launch. Specifically we will be using columns `rocket`, `payloads`, `launchpad`, and `cores`.

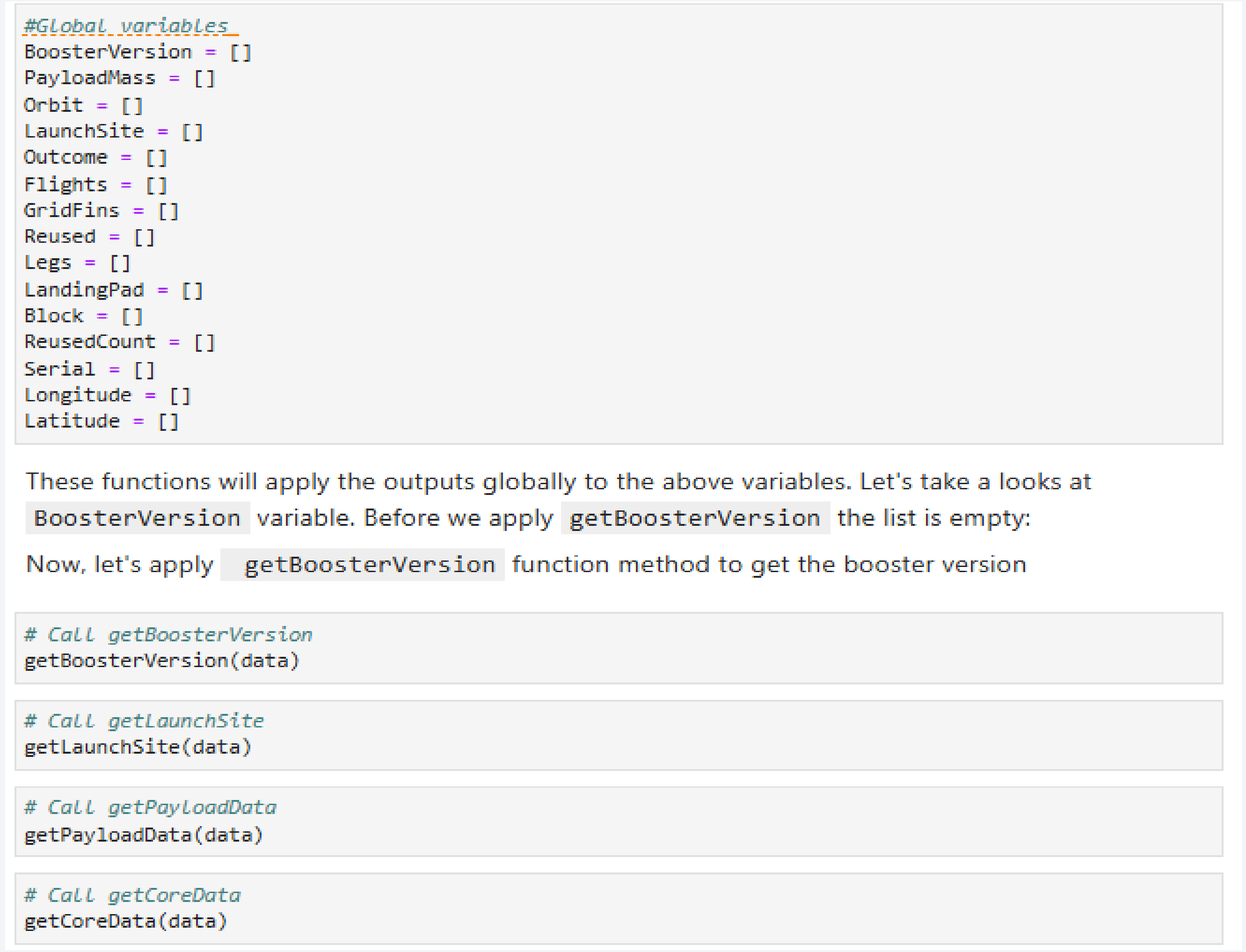
```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and c
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

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

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

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving t
data['date'] = pd.to_datetime(data['date_utc']).dt.date

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

Click here to check on my GitHub

Data Collection – SpaceX API

3) Retrieved extended features using rocket, payload, launchpad, and core IDs.

The rocket, payload, core, and launchpad IDs were used to fetch detailed information via secondary API endpoints, such as booster version, payload mass, orbit type, and landing outcome.



```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

These functions will apply the outputs globally to the above variables. Let's take a looks at `BoosterVersion` variable. Before we apply `getBoosterVersion` the list is empty:

Now, let's apply `getBoosterVersion` function method to get the booster version

```
# Call getBoosterVersion
getBoosterVersion(data)
```

```
# Call getLaunchSite
getLaunchSite(data)
```

```
# Call getPayloadData
getPayloadData(data)
```

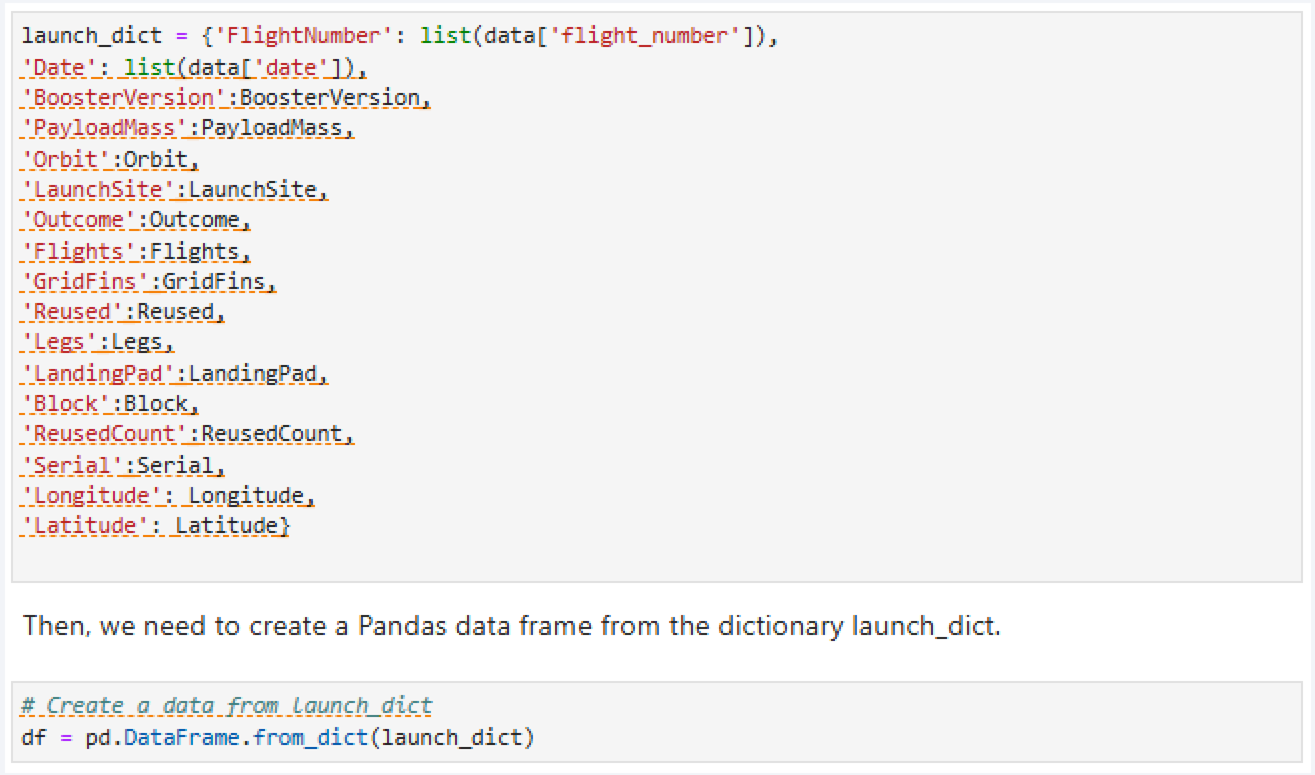
```
# Call getCoreData
getCoreData(data)
```

Click here to check on my GitHub

Data Collection – SpaceX API

4) Compiled all retrieved features into a structured DataFrame for analysis.

All extracted data was organized into a dictionary and converted into a Pandas DataFrame. This dataset is now ready for visualization, EDA, and machine learning modeling.



```
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}
```

Then, we need to create a Pandas data frame from the dictionary launch_dict.

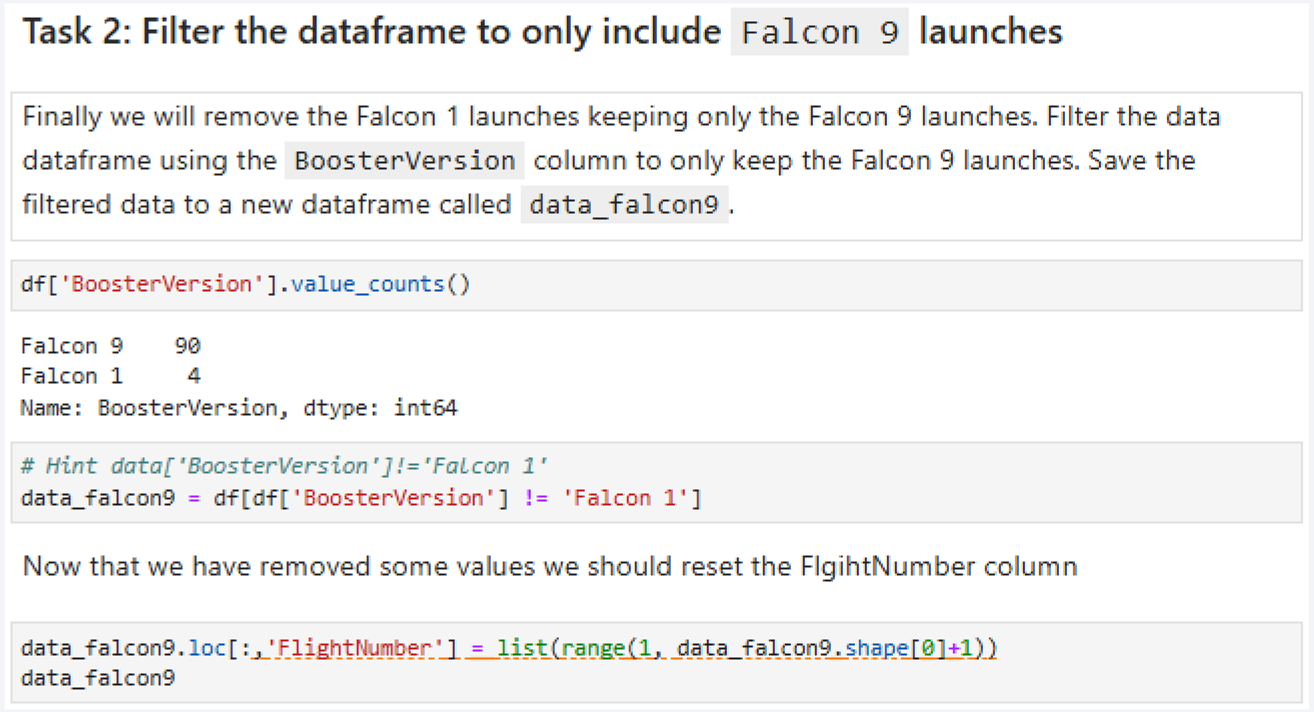
```
# Create a data from launch_dict
df = pd.DataFrame.from_dict(launch_dict)
```

Click here to check on my GitHub

Data Collection – SpaceX API

5) Filtered dataset to include only Falcon 9 launches and reset flight numbers.

Removed all Falcon 1 entries by filtering the `BoosterVersion` column. The dataset was updated to include only Falcon 9 launches. After filtering, the `FlightNumber` column was reset to ensure sequential indexing.



Task 2: Filter the dataframe to only include `Falcon 9` launches

Finally we will remove the Falcon 1 launches keeping only the Falcon 9 launches. Filter the data dataframe using the `BoosterVersion` column to only keep the Falcon 9 launches. Save the filtered data to a new dataframe called `data_falcon9`.

```
df['BoosterVersion'].value_counts()
```

```
Falcon 9    90
Falcon 1     4
Name: BoosterVersion, dtype: int64
```

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

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

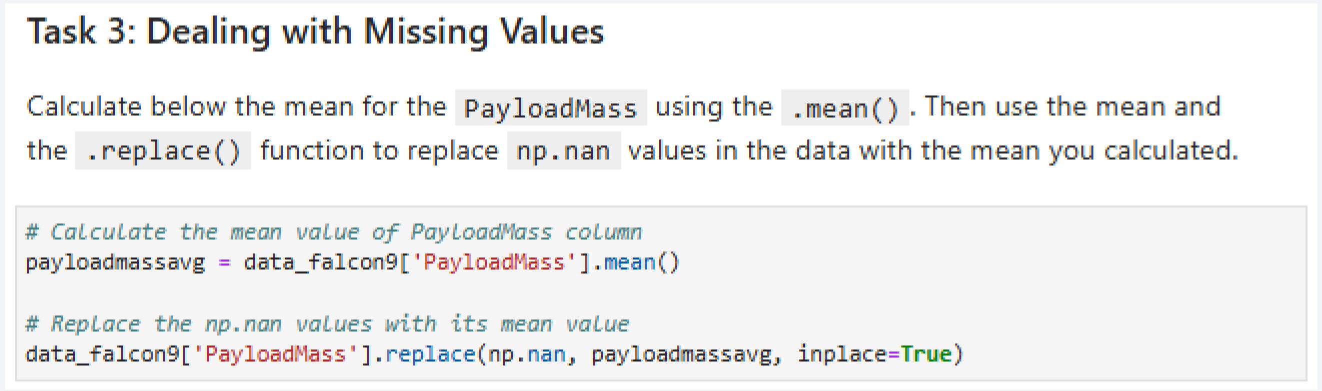
```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

Click here to check on my GitHub

Data Collection – SpaceX API

6) Calculated mean PayloadMass and replaced missing values with the mean.

Computed the average of the PayloadMass column and replaced all NaN values in the dataset with this mean to handle missing data. Verified that no missing values remain.



Task 3: Dealing with Missing Values

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

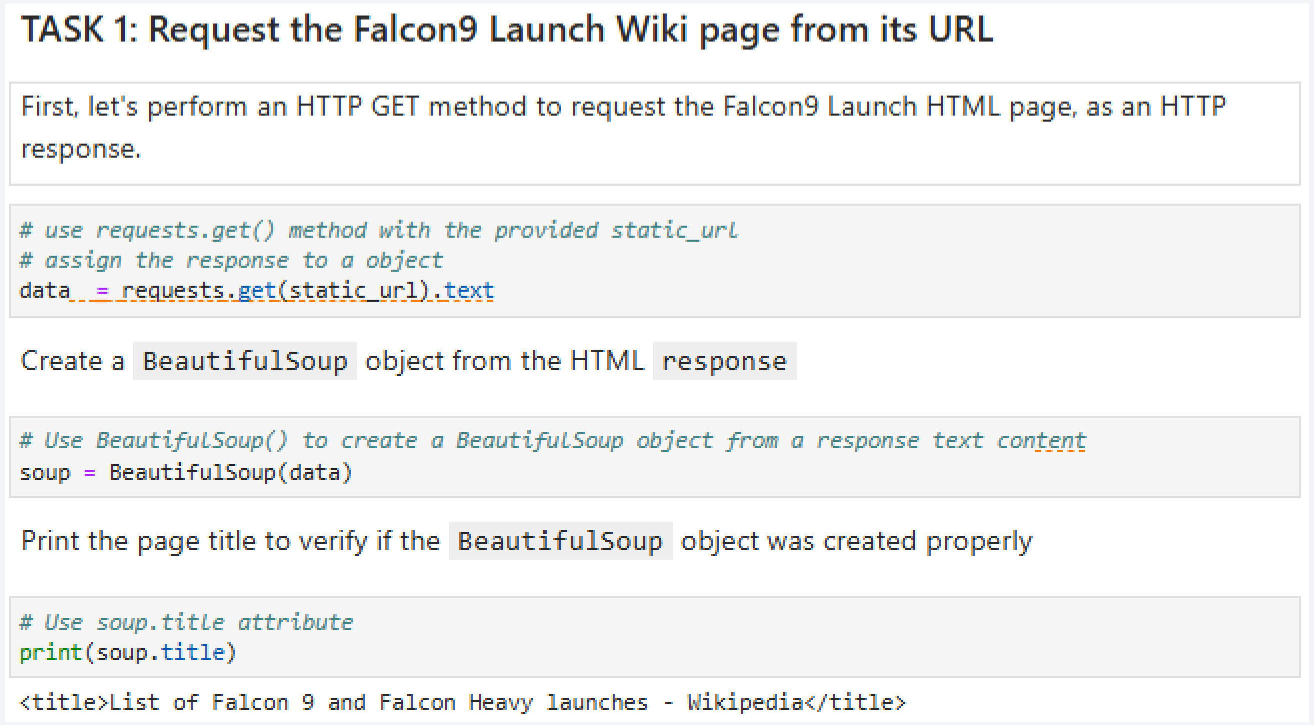
```
# Calculate the mean value of PayloadMass column
payloadmassavg = data_falcon9['PayloadMass'].mean()

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

Click here to check on my GitHub

Data Collection - Scraping

1) Fetched Falcon 9 launch page HTML and created a BeautifulSoup object.

Sent a GET request to obtain the HTML content of the Falcon 9 launches Wikipedia page. Then, parsed the HTML with BeautifulSoup to facilitate data extraction. Verified the object by printing the page title.



TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

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

Create a `BeautifulSoup` object from the HTML `response`

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data)
```

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

```
# Use soup.title attribute
print(soup.title)
```

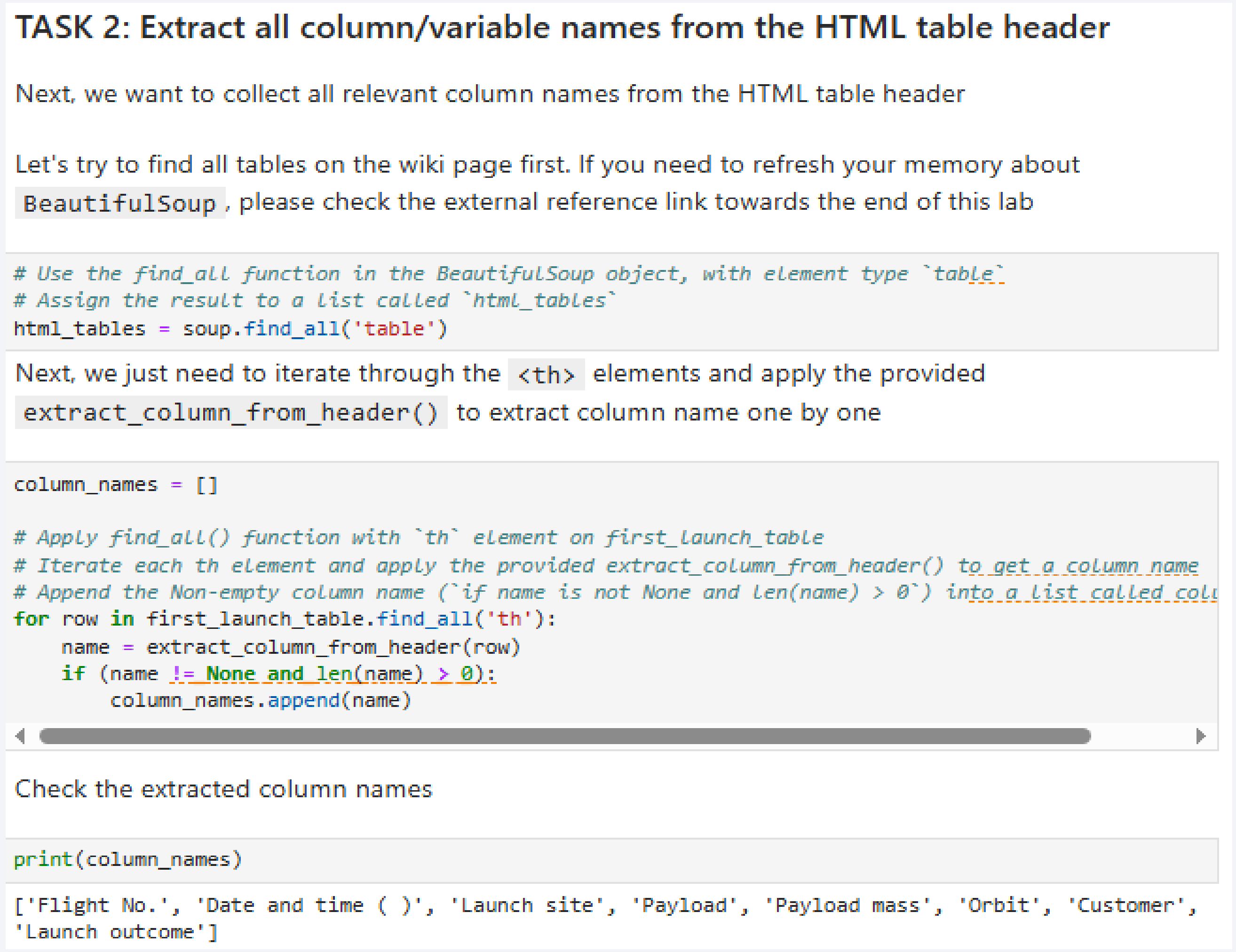
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

Click here to check on my GitHub

Data Collection - Scraping

2) Extracted column names from the Falcon 9 launches table header.

Located all tables in the HTML content, identified the relevant launch table, and extracted its column headers. Processed the <th> elements to collect all column names for later use in data structuring.



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

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

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

```
# 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')
```

Next, we just need to iterate through the `<th>` elements and apply the provided `extract_column_from_header()` to extract column name one by one

```
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 colu
for row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if (name != None and len(name) > 0):
        column_names.append(name)
```

Check the extracted column names

```
print(column_names)
```

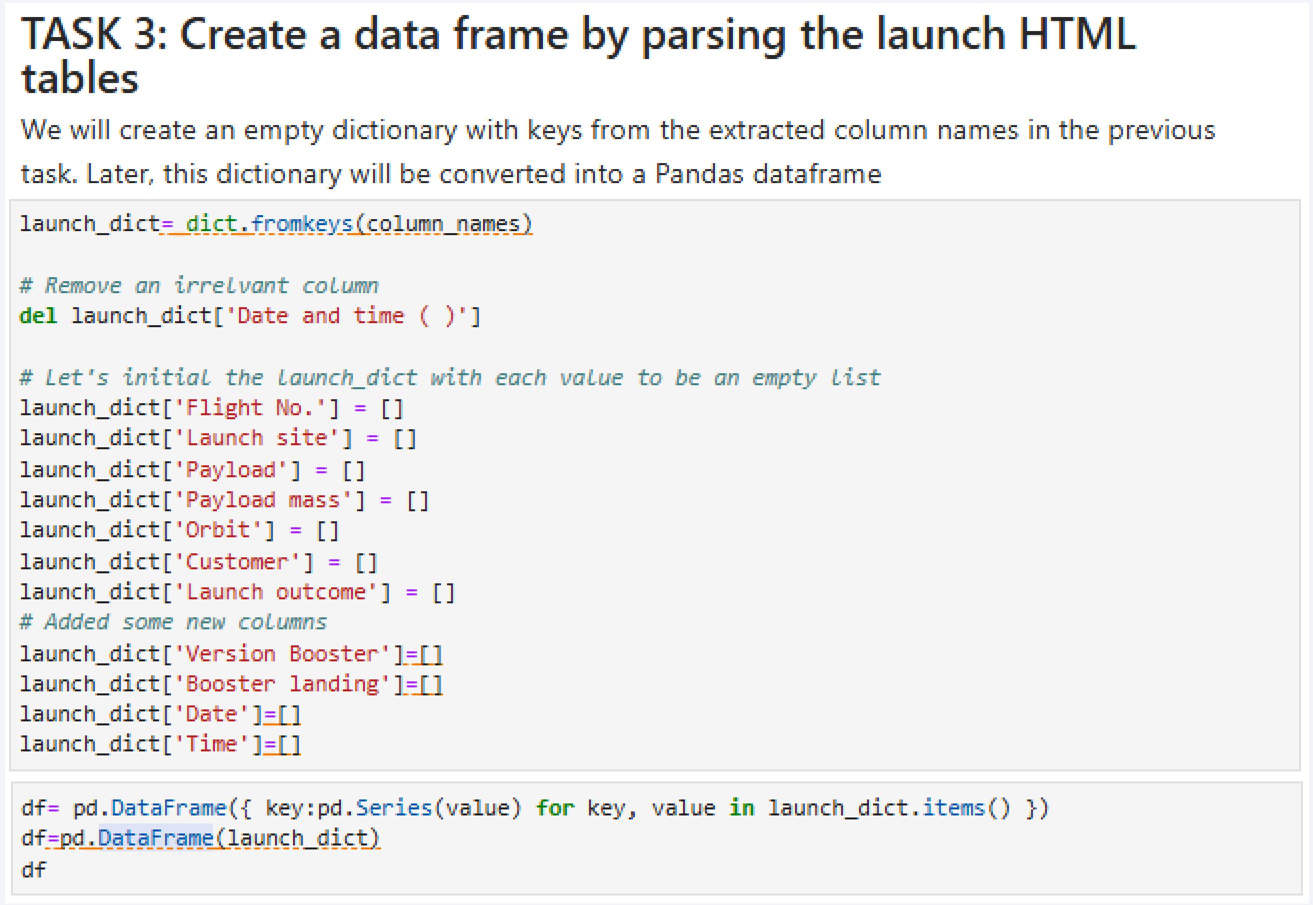
```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer',
'Launch outcome']
```

Click here to check on my GitHub

Data Collection - Scraping

3) Parsed launch data from HTML tables into a structured dictionary, then created a DataFrame.

Initialized a dictionary with column names as keys and empty lists as values. Iterated through the table rows, extracted relevant data for each launch, and appended them to the dictionary. Finally, converted the dictionary to a Pandas DataFrame for analysis.



TASK 3: Create a data frame by parsing the launch HTML tables

We will create an empty dictionary with keys from the extracted column names in the previous task. Later, this dictionary will be converted into a Pandas dataframe

```
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']=[]
```

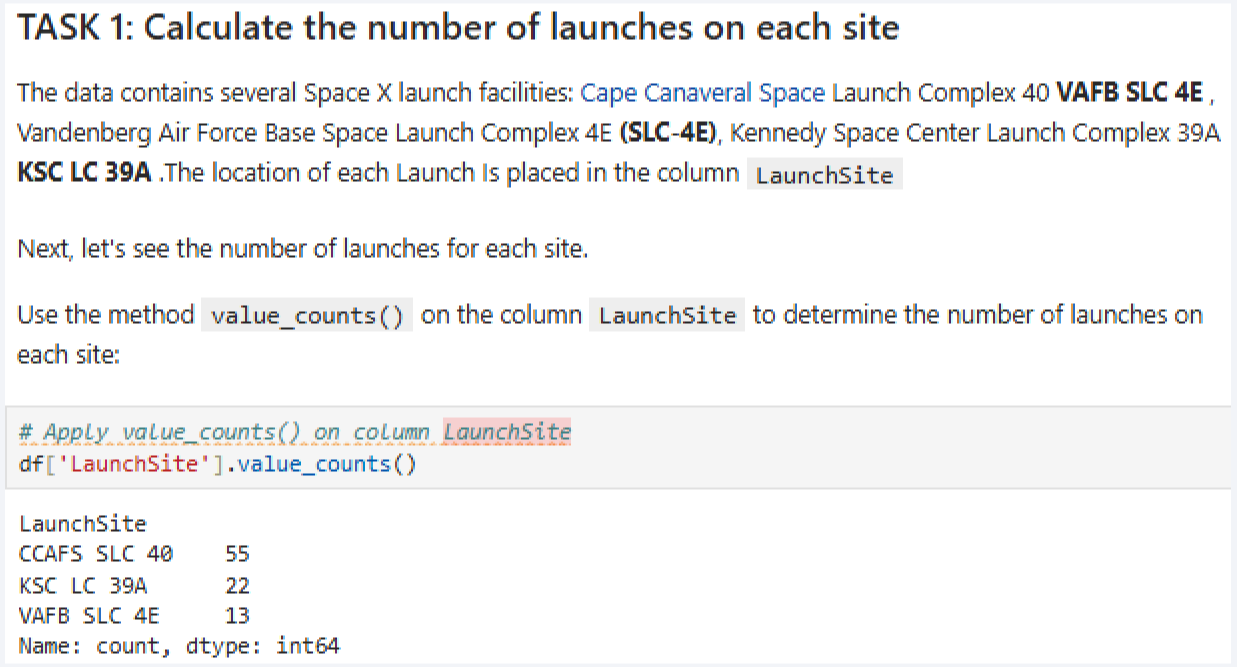
```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
df=pd.DataFrame(launch_dict)
df
```

Click here to check on my GitHub

Data Wrangling

1) Calculated the number of launches at each launch site.

Counted how many launches were made from each SpaceX launch facility using the `LaunchSite` column. This helps identify the most used launch sites for analysis.



TASK 1: Calculate the number of launches on each site

The data contains several Space X launch facilities: Cape Canaveral Space Launch Complex 40 **VAFB SLC 4E** , Vandenberg Air Force Base Space Launch Complex 4E **(SLC-4E)**, Kennedy Space Center Launch Complex 39A **KSC LC 39A** .The location of each Launch Is placed in the column `LaunchSite`

Next, let's see the number of launches for each site.

Use the method `value_counts()` on the column `LaunchSite` to determine the number of launches on each site:

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

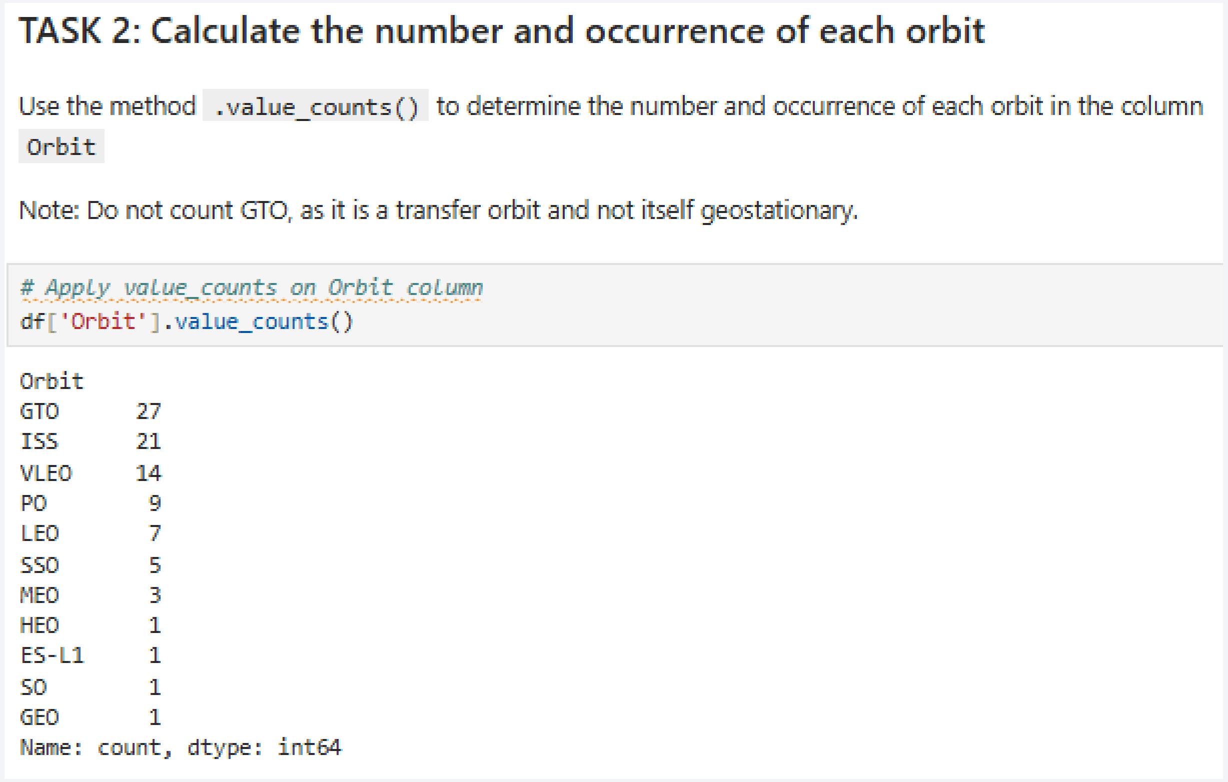
```
LaunchSite
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: count, dtype: int64
```

Click here to check on my GitHub

Data Wrangling

2) Calculated the number and occurrence of each orbit type, excluding transfer orbit GTO

Used `.value_counts()` on the `Orbit` column to count how many launches occurred in each orbit. GTO is excluded from some analyses because it is a transfer orbit, not geostationary.



TASK 2: Calculate the number and occurrence of each orbit

Use the method `.value_counts()` to determine the number and occurrence of each orbit in the column `Orbit`

Note: Do not count GTO, as it is a transfer orbit and not itself geostationary.

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

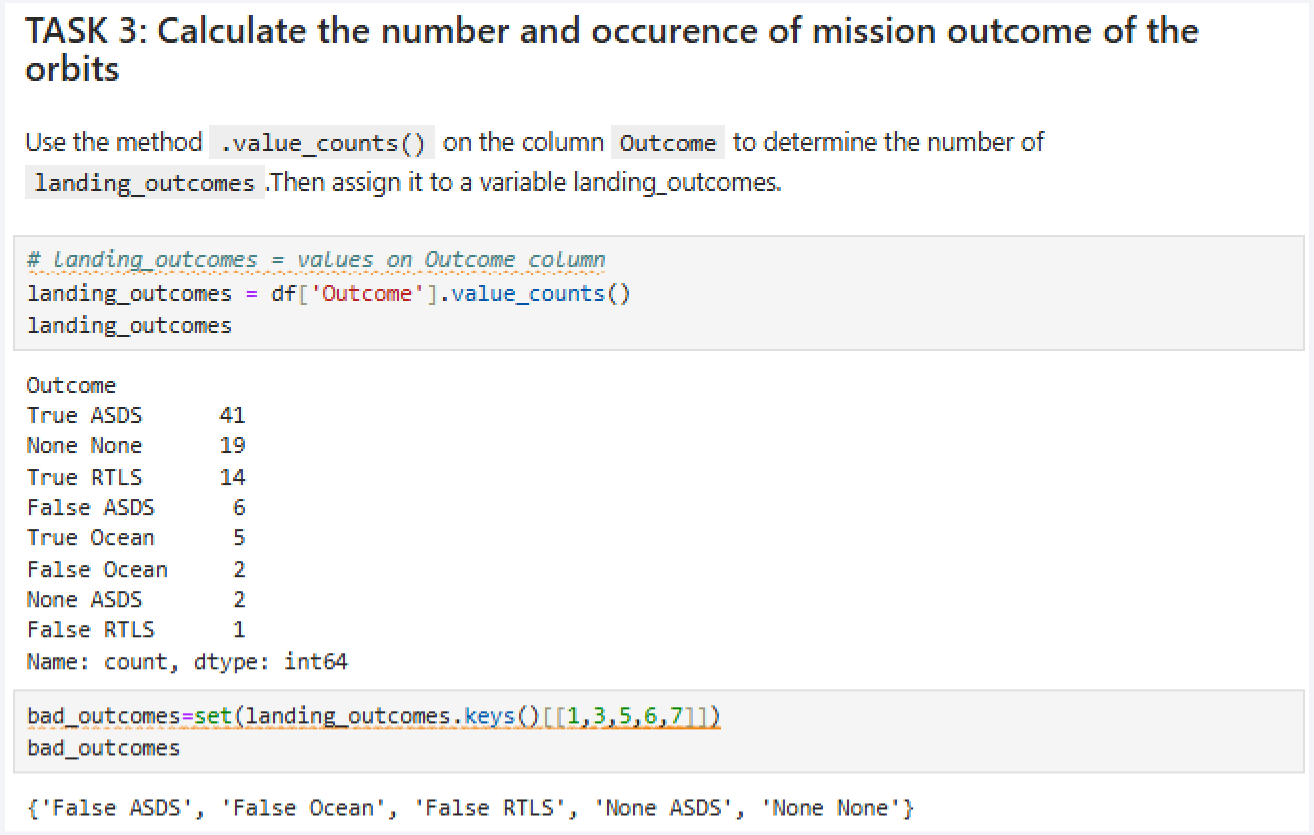
```
Orbit
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
HEO       1
ES-L1     1
SO        1
GEO       1
Name: count, dtype: int64
```

Click here to check on my GitHub

Data Wrangling

3) Calculated the count and categories of mission outcomes using the Outcome column.

Used `.value_counts()` on the `Outcome` column to count different landing results, such as successful landings on drone ships (ASDS), ground pads (RTLS), and ocean areas. Created a set of bad outcomes where landings failed.



TASK 3: Calculate the number and occurence of mission outcome of the orbits

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes` .Then assign it to a variable landing_outcomes.

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

```
Outcome
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: count, dtype: int64
```

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

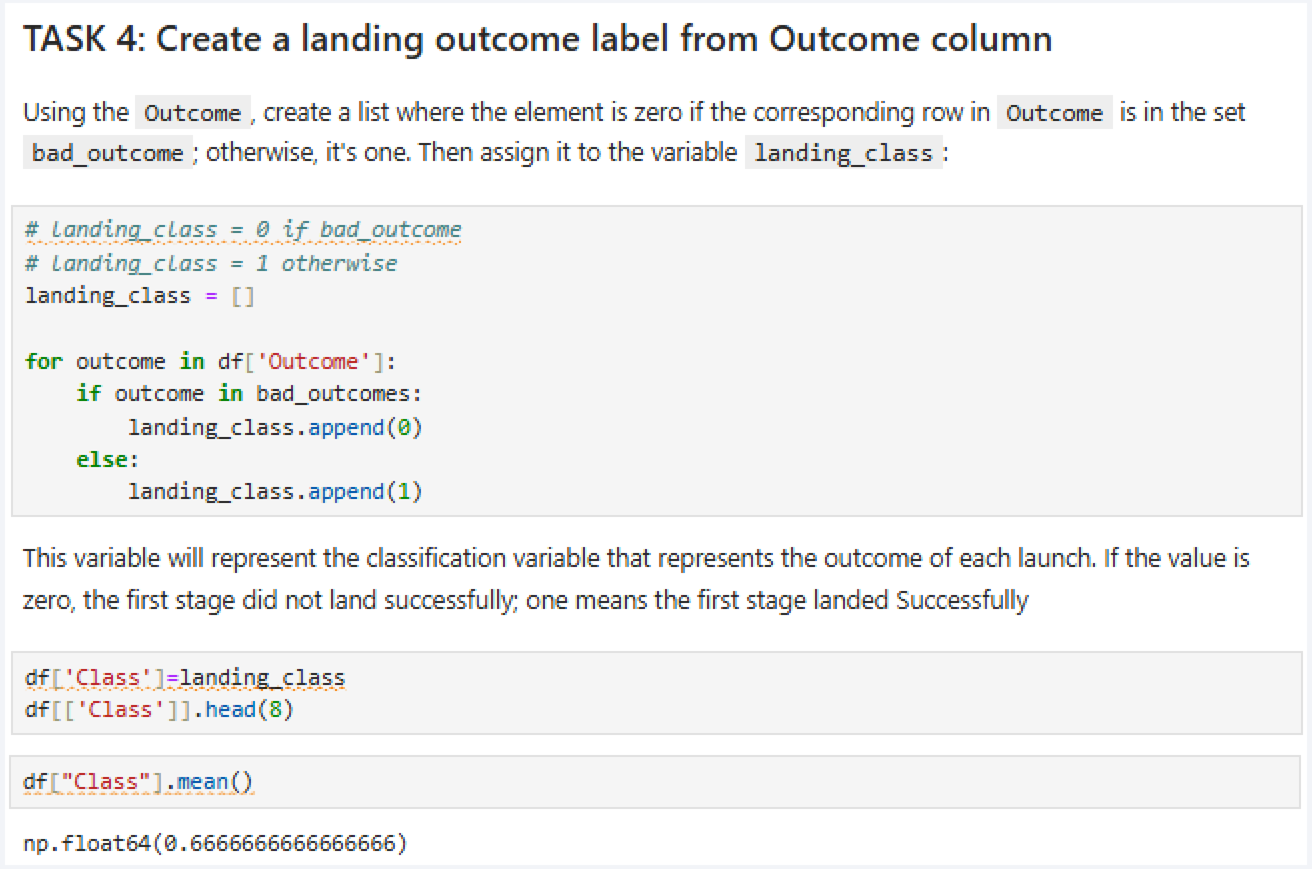
```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

Click here to check on my GitHub

Data Wrangling

4) Created a new classification column representing landing success: 0 for failure, 1 for success.

Assigned the variable `landing_class` to the DataFrame as the 'Class' column. This variable flags whether the first stage landed successfully (1) or not (0). Calculated the average success rate using `.mean()`.



TASK 4: Create a landing outcome label from Outcome column

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []

for outcome in df['Outcome']:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

```
df['Class']=landing_class
df[['Class']].head(8)
```

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

```
np.float64(0.6666666666666666)
```

Click here to check on my GitHub

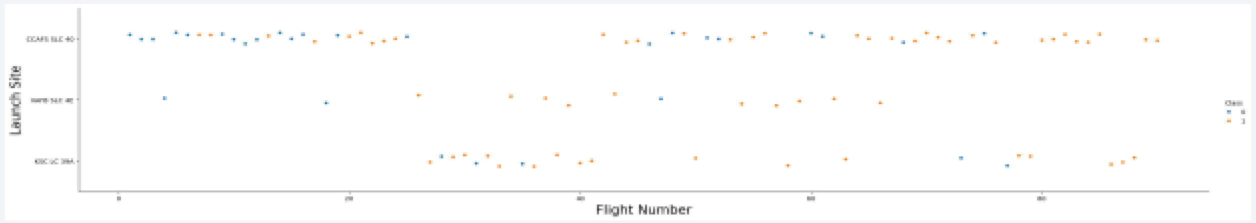
EDA with Data Visualization

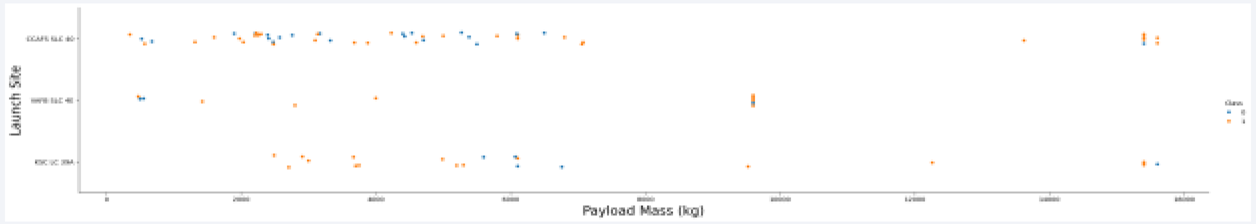
1) Used scatter plots to visualize relationships between flight details and launch/orbit parameters.

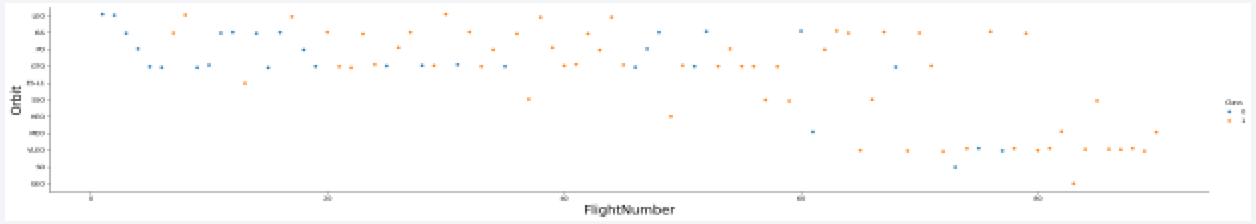
Created scatter plots to analyze relationships between:

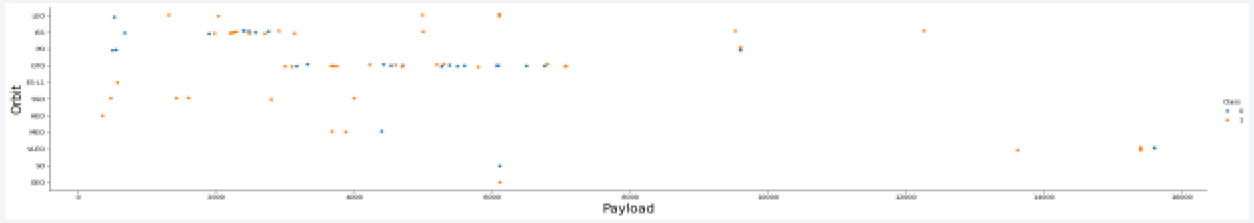
- Flight Number vs Launch Site
- Payload Mass vs Launch Site
- Flight Number vs Orbit type
- Payload vs Orbit type

These plots help identify patterns or clusters in launch data.







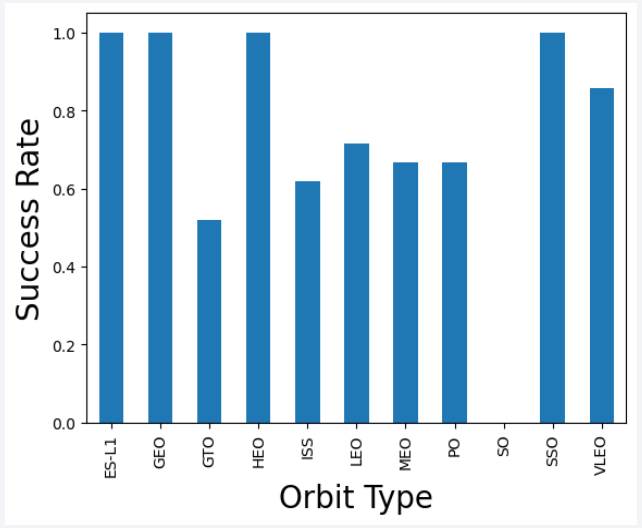


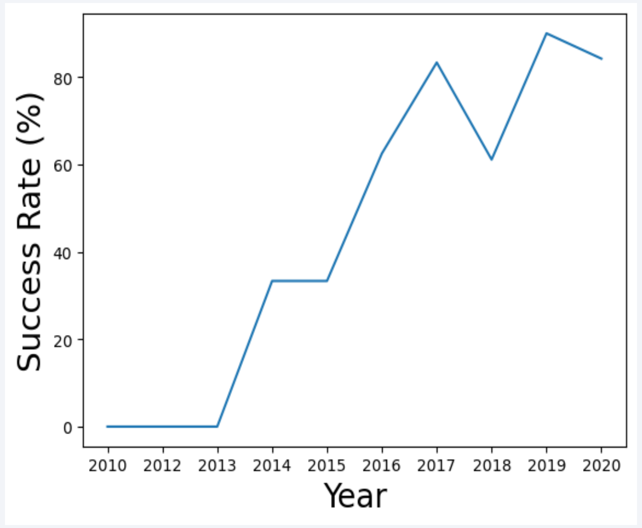
[Click here to check on my GitHub]

EDA with Data Visualization

2) Used bar charts to show success rates by orbit type and line charts to display yearly launch success trends.

- Bar chart visualizes success rates for each orbit type, showing which orbits have higher mission success.
- Line chart depicts launch success trends over the years, helping to identify improvements or patterns over time.





Click here to check on my GitHub

EDA with SQL

SQL queries were performed to:

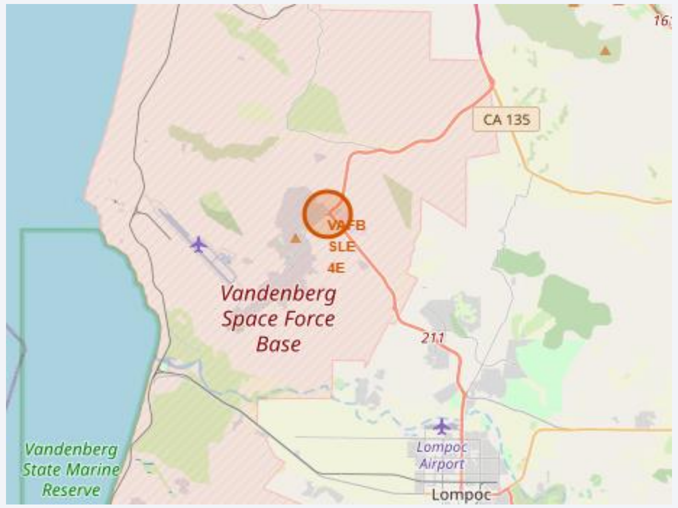
1. Identification of unique launch sites in space missions.
2. Extraction of the first 5 records where launch sites start with "CCA".
3. Calculation of total payload mass carried by boosters launched by NASA (CRS).
4. Calculation of average payload mass carried by booster version F9 v1.1.
5. Identification of the date of the first successful landing on ground pad.
6. Listing of boosters with successful drone ship landings and payload between 4000 and 6000 kg.
7. Counting total successful and failed mission outcomes.
8. Identification of booster versions that carried the maximum payload mass using an aggregated subquery.
9. Listing of months in 2015 with drone ship landing failures, including booster version and launch site.
10. Ranking of landing outcome types between 2010 and 2017, ordered by descending frequency.

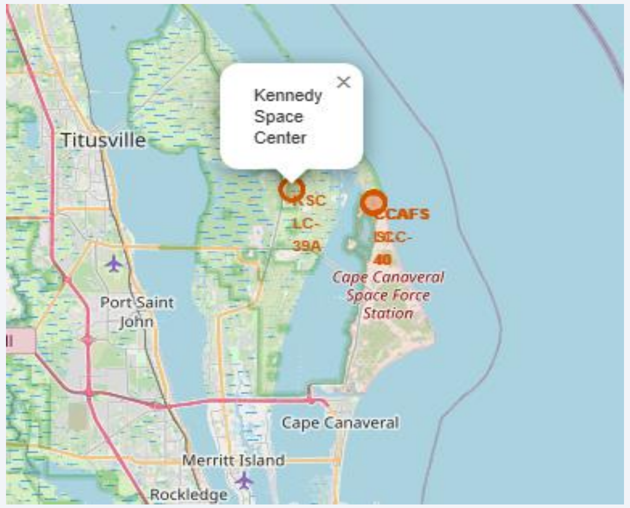
Click here to check on my GitHub

Build an Interactive Map with Folium

1) We added circles and markers for each SpaceX launch site on the interactive Folium map.

- Circles highlight the launch site area with a radius around the coordinates.
- Markers label each site with its name, displayed as a popup or label on the map.
- Circles provide a clear visual boundary around each launch site, helping viewers quickly locate the general area.
- Markers provide precise site identification with labels, improving map usability and user interaction.



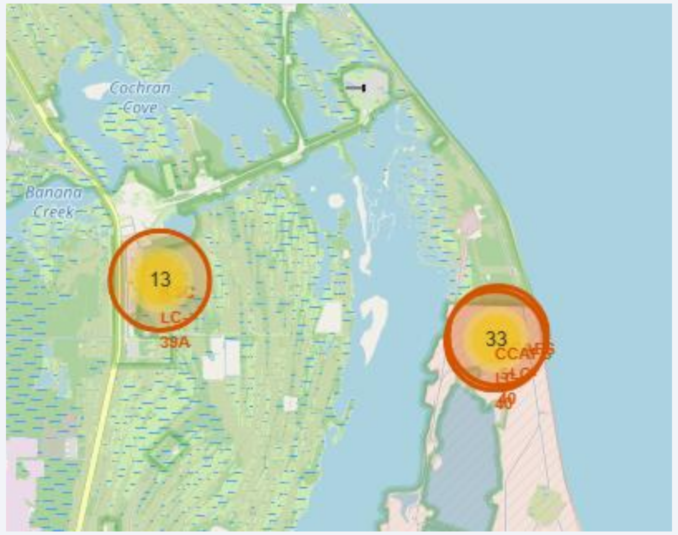


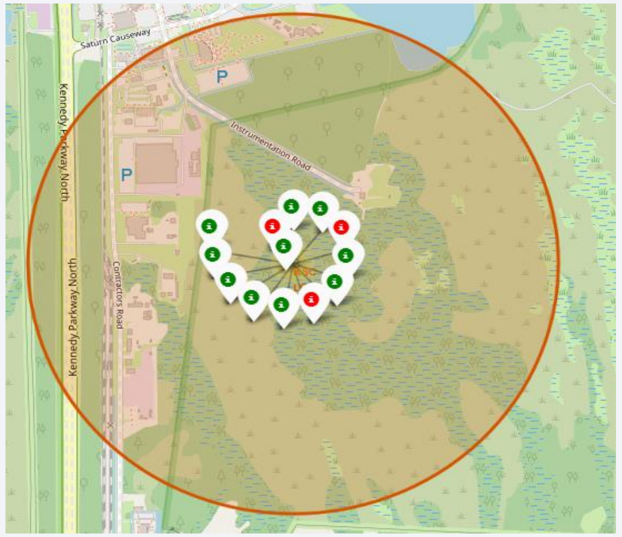
Click here to check on my GitHub

Build an Interactive Map with Folium

2) Added markers on the map for each launch record, colored green for successful launches and red for failed ones, based on the class column in the dataset.

- Used MarkerCluster to group markers with the same coordinates to avoid clutter.
- Color coding (green/red) immediately communicates launch success or failure.
- Marker clustering improves map readability when many launches share the same location.



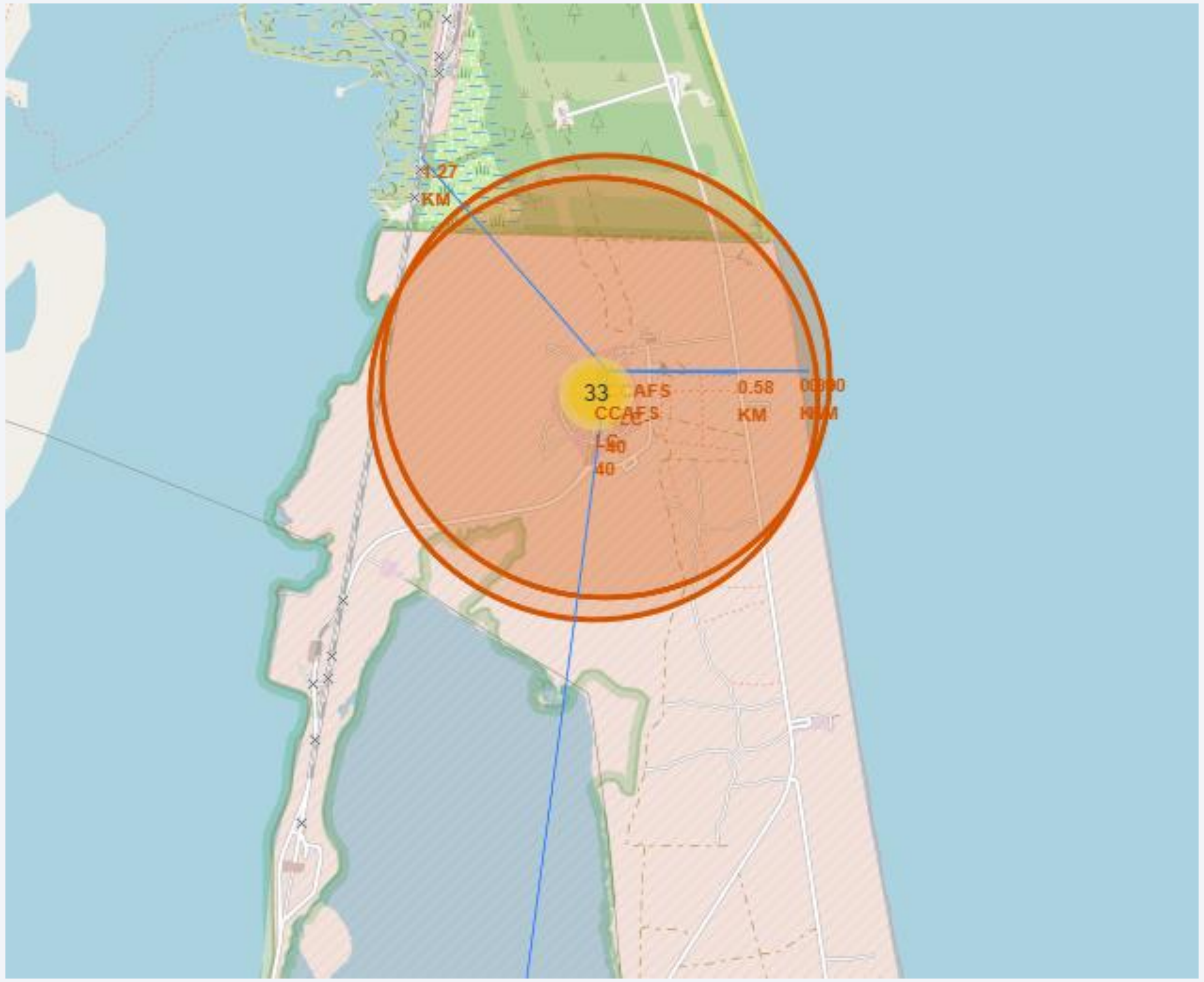


Click here to check on my GitHub

Build an Interactive Map with Folium

3) Added a MousePosition plugin to the map to help get latitude and longitude interactively while exploring the map.

- Implemented a function to calculate the distance between two geographic points (launh site and nearby points).
- Marked closest points of interest (coastline, highway, railroad, city) on the map with distance labels.



[Click here to check on my GitHub]

Build a Dashboard with Plotly Dash

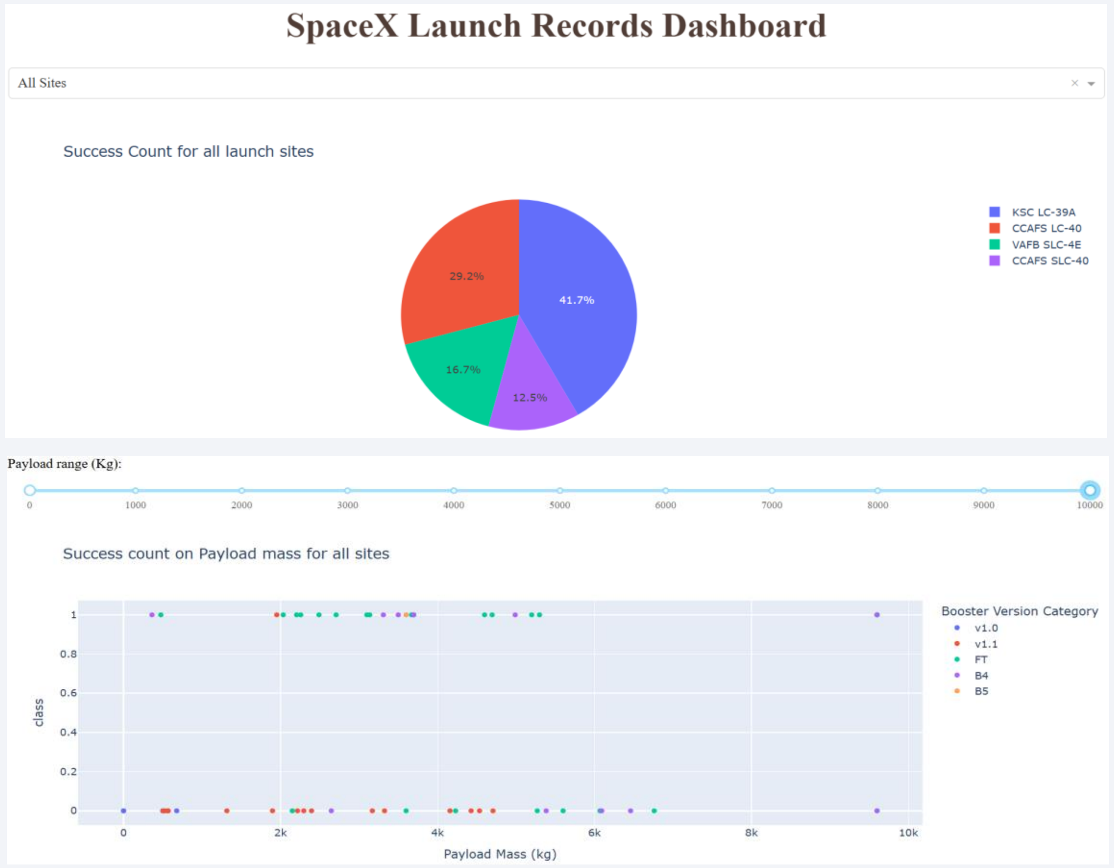
Dropdown Option for Pie Chart and Scatter Plot with Payload Slider

In this task, we created an interactive dropdown menu to visualize the success rates of launches for all launch sites combined or individually. The dropdown allows the user to select a specific launch site or view data from all sites.

The pie chart displays the proportion of successful versus failed launches for the selected site(s), making it easy to compare success rates.

Additionally, a scatter plot was developed with a payload mass range slider. This enables filtering of launches by payload size, providing insights into how payload mass correlates with launch success across different sites.

Together, these interactive visualizations enhance the ability to explore launch outcomes by site and payload dynamically.



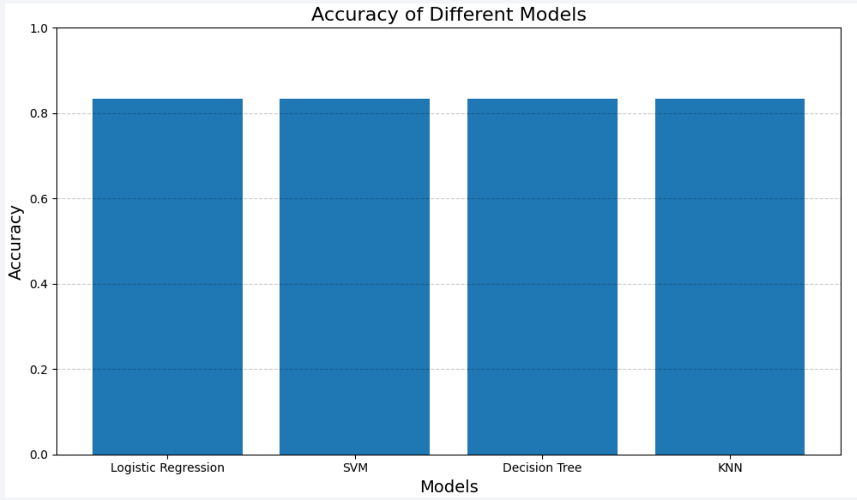
Click here to check on my GitHub

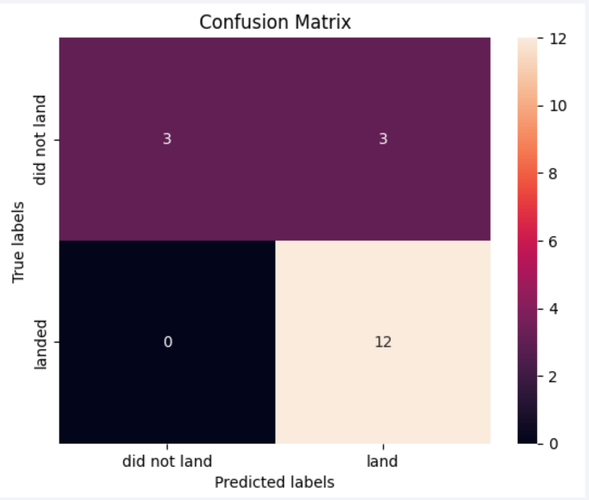
Predictive Analysis (Classification)

Predictive Analysis (Classification)

In this task, we developed multiple classification models—including Logistic Regression, Support Vector Machine (SVM), Decision Tree, and K-Nearest Neighbors (KNN)—to predict the success of launches. The process involved data preprocessing, feature selection, training, and evaluating each model using accuracy metrics. We compared the models' performances visually with a bar chart to easily identify differences in accuracy.

All models achieved a similar accuracy rate of approximately 83.33%, demonstrating consistent predictive capability across the different algorithms.

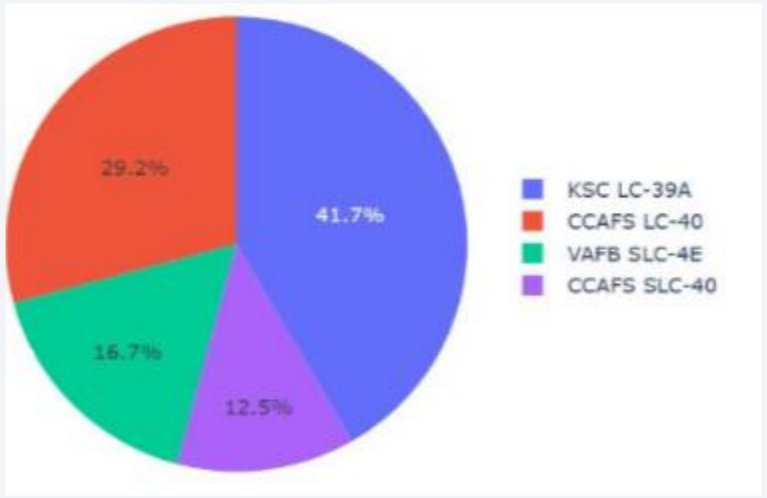


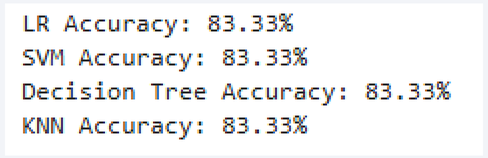


Click here to check on my GitHub

Results

- Logistic Regression, SVM, Decision Tree, and KNN all demonstrated comparable high performance in predicting launch outcomes in this dataset.
- Launches with lighter payloads consistently demonstrate higher success rates compared to those carrying heavier payloads.
- The likelihood of a SpaceX launch succeeding increases with operational experience over the years, indicating a trend towards near-flawless launches as the company matures.
- Kennedy Space Center’s Launch Complex 39A records the highest number of successful launches among all launch sites analyzed.
- Orbits such as GEO, HEO, SSO, and ES L1 stand out for their superior success rates, showing the most reliable launch outcomes.



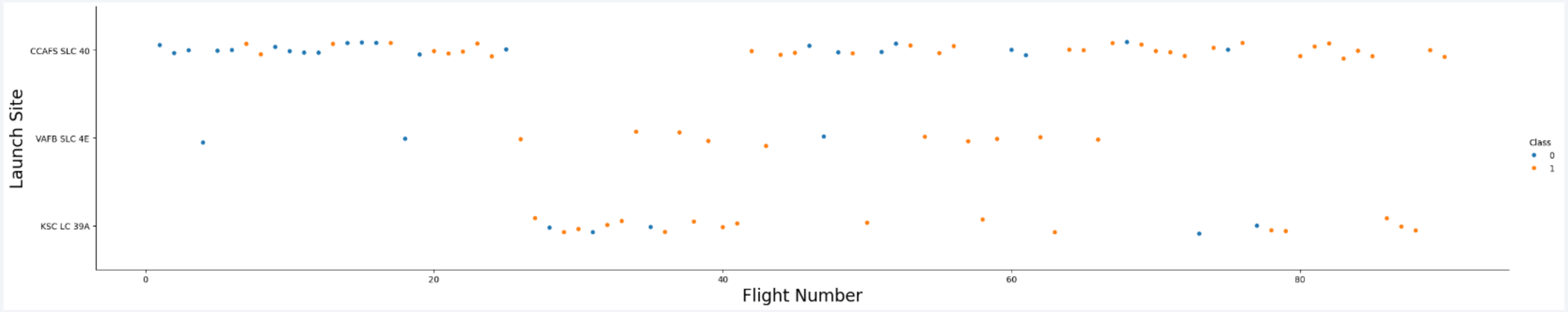

```
LR Accuracy: 83.33%
SVM Accuracy: 83.33%
Decision Tree Accuracy: 83.33%
KNN Accuracy: 83.33%
```

Section 2

Insights drawn from EDA

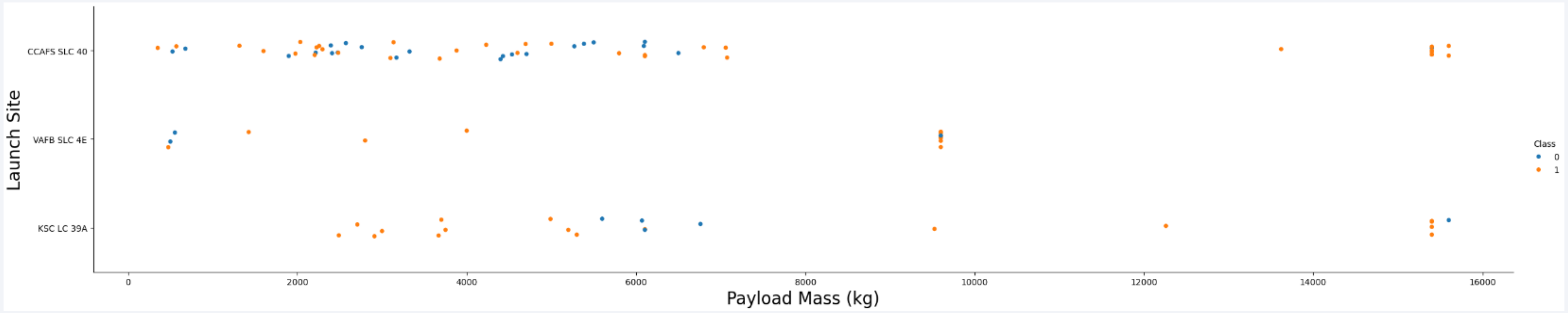
Flight Number vs. Launch Site

- The scatter plot presented here illustrates the relationship between Flight Number and Launch Site, with color coding representing the Class values.
- The visualization highlights a significantly higher number of launches originating from the CCAFS SLC 40 launch site when compared to other sites, revealing a more intensive launch activity pattern at this particular location.



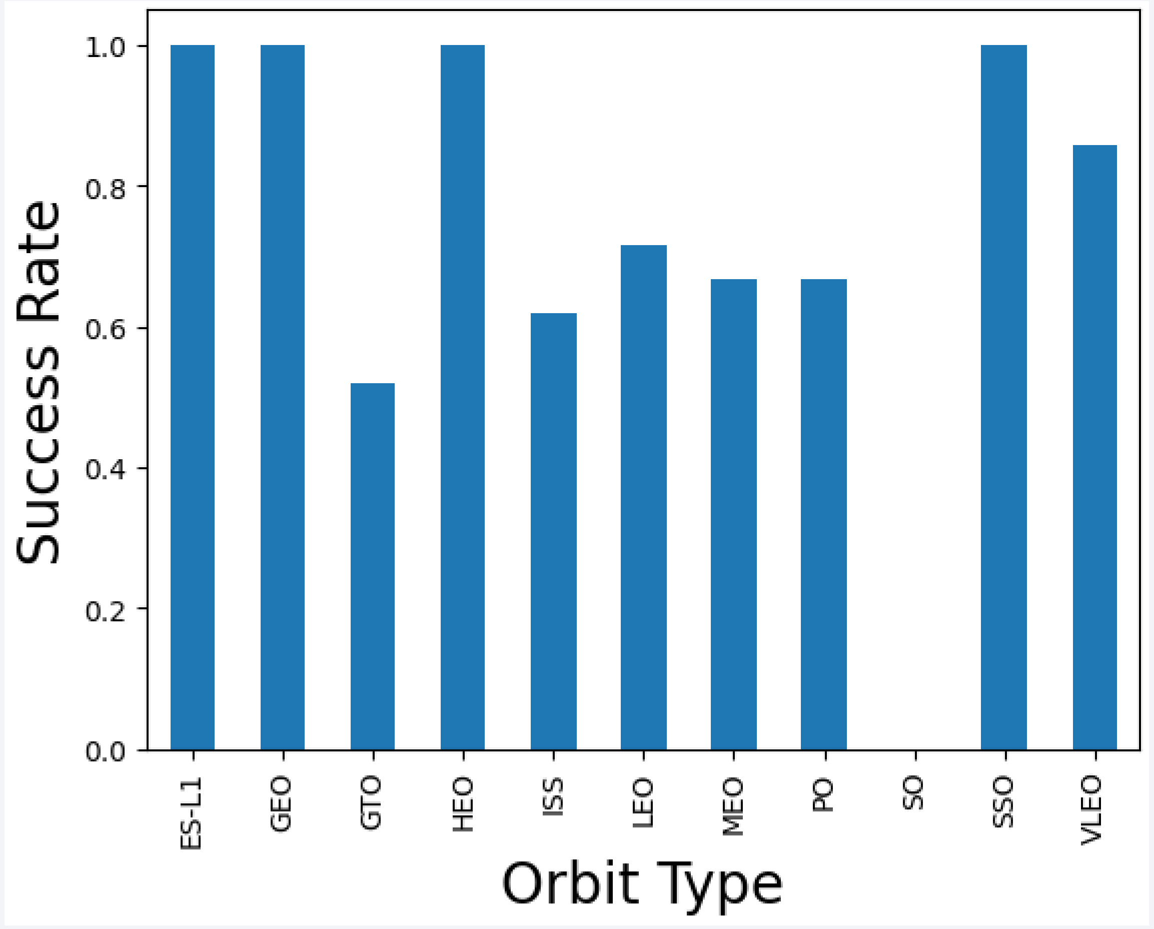
Payload vs. Launch Site

- The scatter plot displayed here shows the relationship between Payload Mass (kg) and Launch Site, with hue representing the Class values.
- It is apparent that launches with lower payload mass are more frequent across all launch sites, while payloads with higher mass seem to correlate with a higher occurrence of Class 1 (success) launches, suggesting that heavier payloads are more likely to be successfully launched.



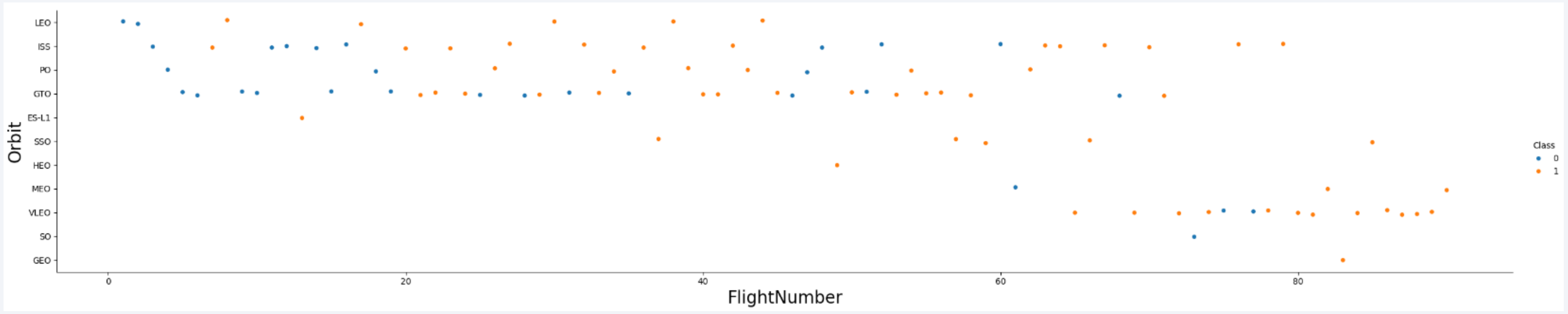
Success Rate vs. Orbit Type

- The bar chart presented here showcases the success rate for each orbit type, calculated as the mean of the Class values.
- It is evident that orbit types such as **ES-L1**, **GEO**, **HEO**, and **SSO** exhibit the highest success rates compared to other orbit categories, highlighting the superior reliability of these orbits in achieving successful launches.
- Additionally, the orbit type **SO** stands out with a notably low or zero success rate, indicating a significant deviation in performance compared to the others.



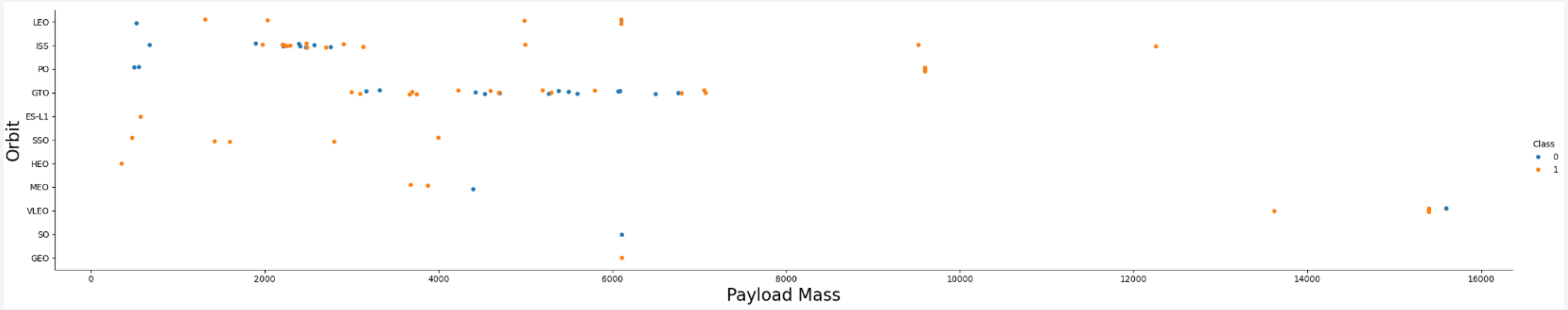
Flight Number vs. Orbit Type

- The scatter plot presented here illustrates the relationship between Flight Number and Orbit Type, with color differentiation by Class values.
- It is apparent that orbits such as LEO, ISS, PO, and GTO had the highest number of launches in the earlier years, but over time, there has been a gradual shift towards the VLEO orbit, reflecting changing mission preferences and/or advancements in technology.



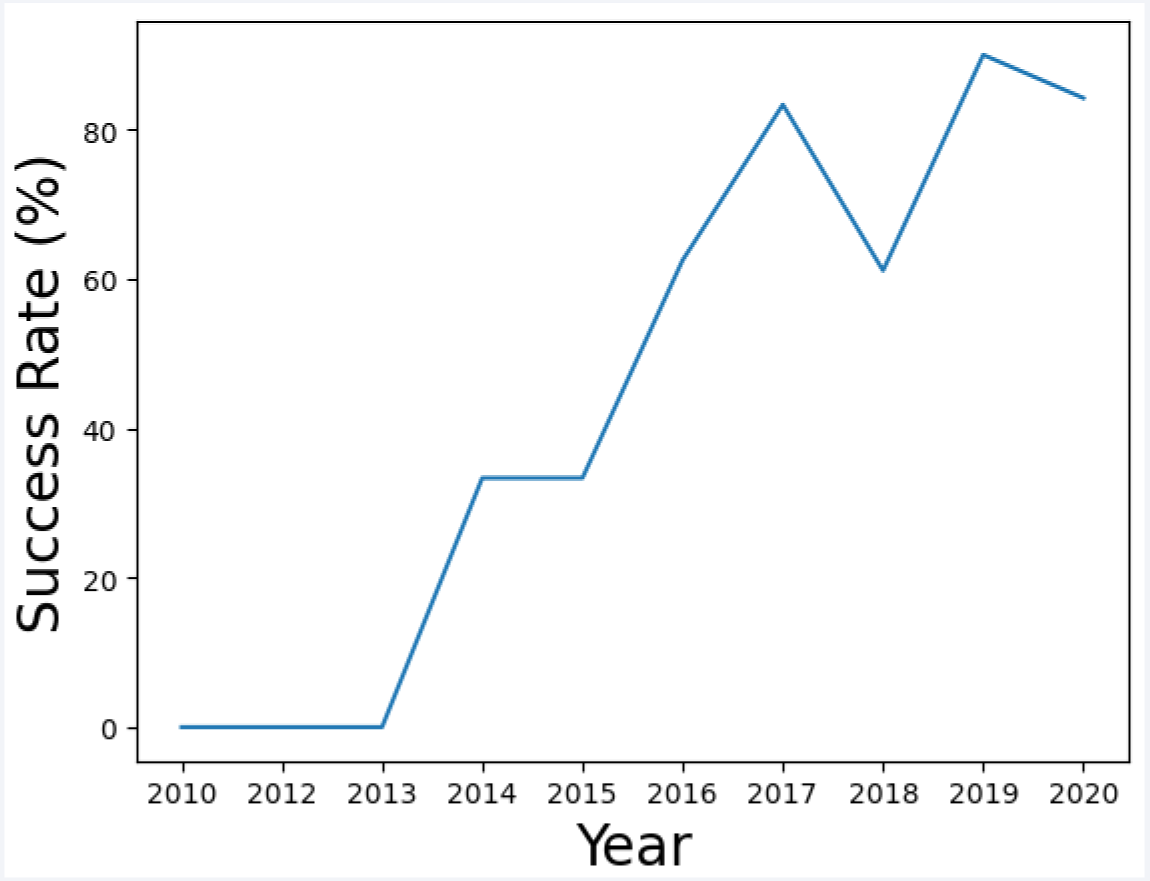
Payload vs. Orbit Type

- The scatter plot presented here shows the relationship between Payload Mass and Orbit Type, with color encoding by Class values.
- It reveals that heavier payloads are generally associated with higher success rates in orbits like PO, LEO, and ISS.
- However, for GTO orbit, the success rate appears less consistent, with a near-equal distribution of both successful and unsuccessful launches.
- Notably, for the SSO orbit, the success rate remains consistently at 100%, regardless of payload mass, indicating flawless performance in this orbit type.



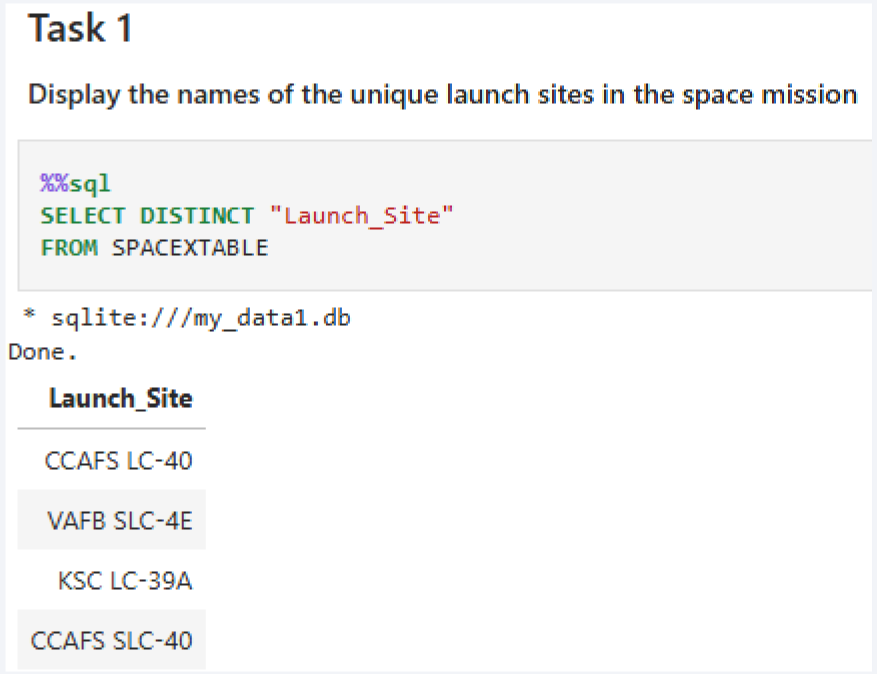
Launch Success Yearly Trend

- The line chart illustrates the yearly average success rate of launches, showing that there were notable failures up until 2013.
- From 2013 to 2020, a clear upward trend is observed, likely driven by technological advancements and accumulated operational experience.
- A dip in success rate occurred in 2018, but was followed by a recovery and continued improvement in 2019, highlighting the resilience and ongoing enhancement in launch performance.



All Launch Site Names

- The query retrieves the distinct names of launch sites from the dataset, providing an overview of all unique locations where launches have taken place.
- This information is essential for understanding the geographical distribution of launch activities and for further analysis related to site-specific performance or trends



Task 1

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

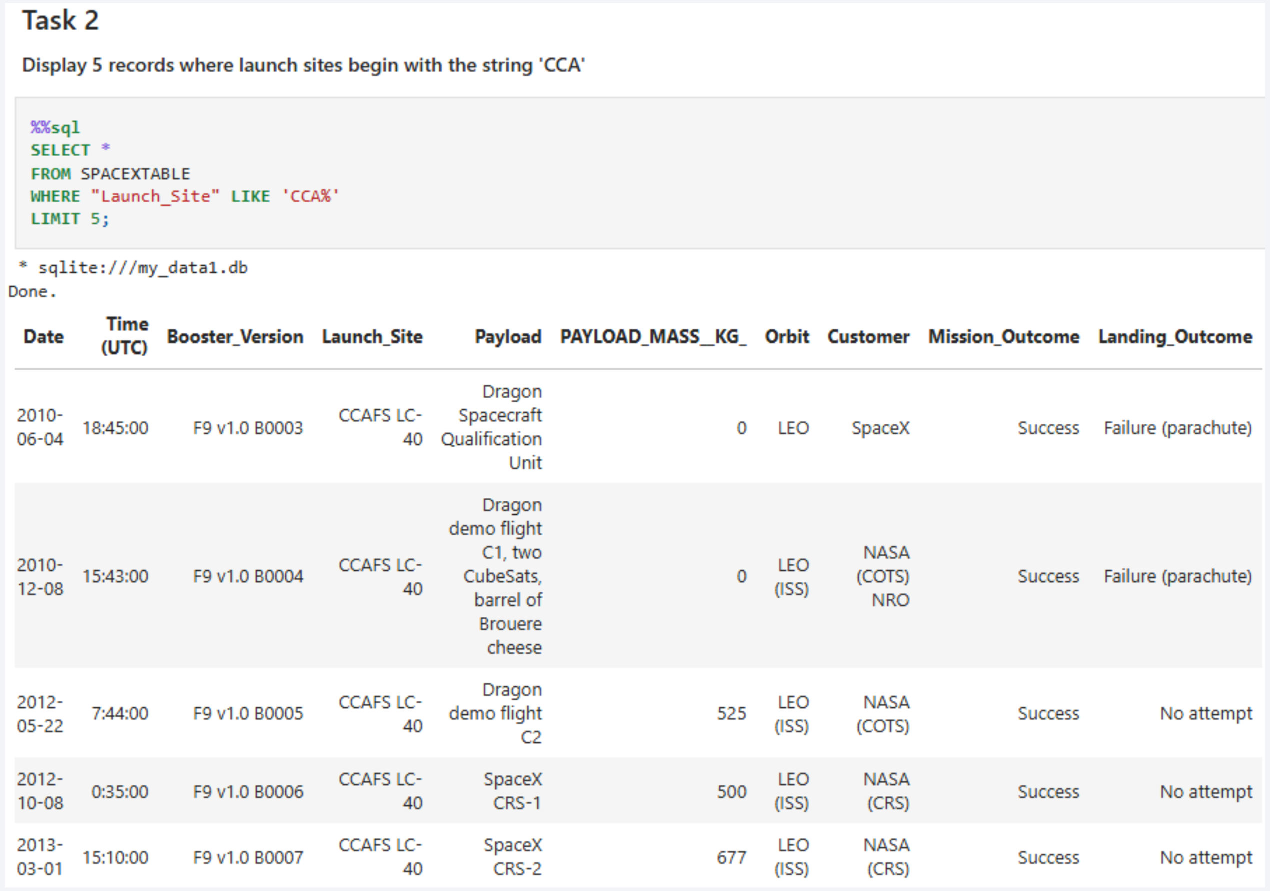
```
%%sql
SELECT DISTINCT "Launch_Site"
FROM SPACEXTABLE
```

```
 * 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'

- The query extracts five records from the dataset where the launch site names start with the prefix 'CCA'.
- This selection helps to focus on specific launch sites beginning with 'CCA', enabling a closer examination of their related launch data and characteristics.



Task 2

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

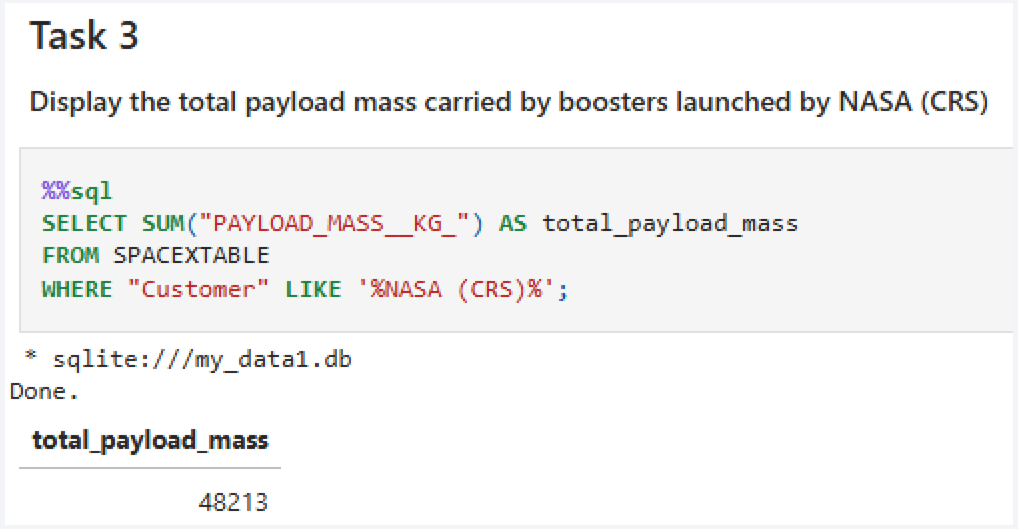
```
%%sql
SELECT *
FROM SPACEXTABLE
WHERE "Launch_Site" LIKE 'CCA%'
LIMIT 5;
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- The query calculates the total payload mass carried by boosters for missions where NASA (CRS) is the customer.
- This aggregate value provides insight into the cumulative payload capacity delivered by these specific NASA-contracted launches.



Task 3

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

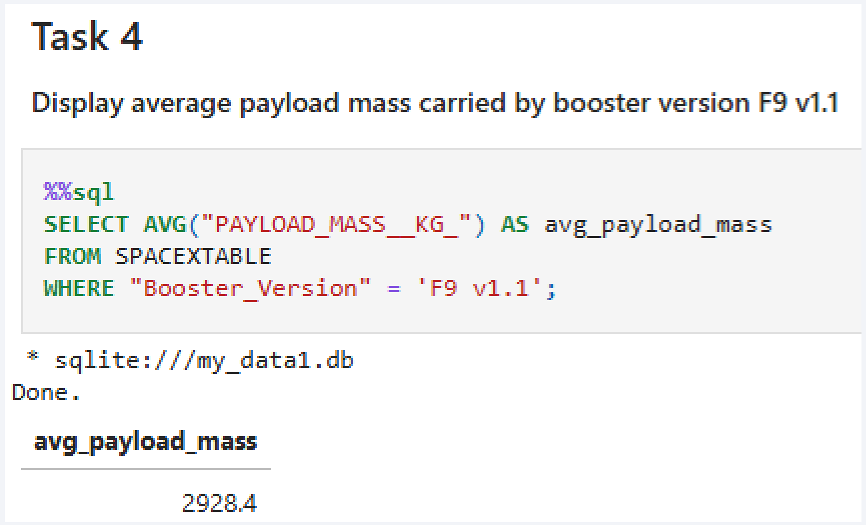
```
%%sql
SELECT SUM("PAYLOAD_MASS__KG_") AS total_payload_mass
FROM SPACEXTABLE
WHERE "Customer" LIKE '%NASA (CRS)%';
```

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

total_payload_mass
48213

Average Payload Mass by F9 v1.1

- The query computes the average payload mass carried by boosters of version F9 v1.1.
- This average value helps to understand the typical payload capacity handled by this specific booster model across its missions.



Task 4

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

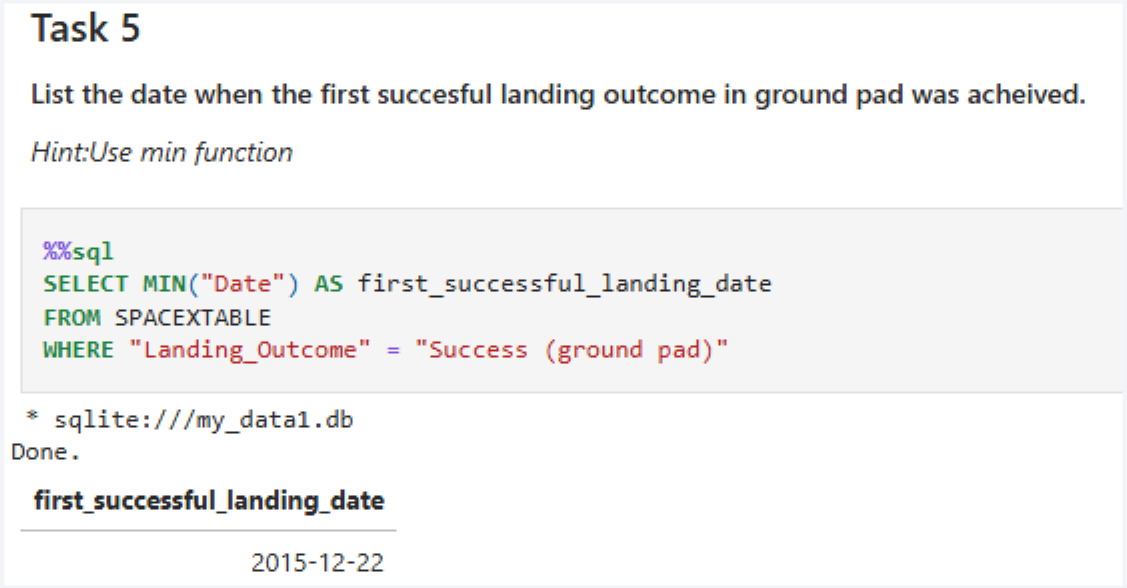
```
%%sql
SELECT AVG("PAYLOAD_MASS__KG_") AS avg_payload_mass
FROM SPACEXTABLE
WHERE "Booster_Version" = 'F9 v1.1';
```

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

avg_payload_mass
2928.4

First Successful Ground Landing Date

- The query identifies the earliest date of a successful landing on a ground pad, which occurred on December 22, 2015.
- This date marks a significant milestone in the progress of landing technology, representing the first time a booster was successfully recovered on a ground pad.



Task 5

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

Hint:Use min function

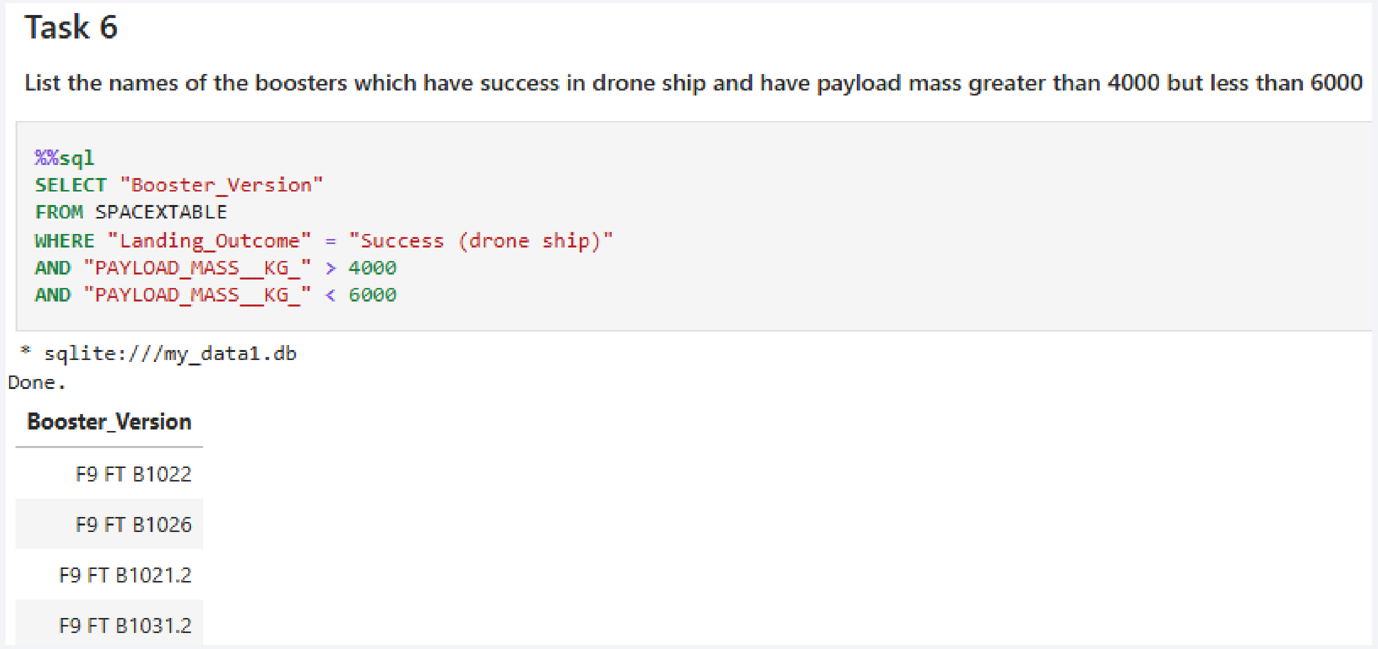
```
%%sql
SELECT MIN("Date") AS first_successful_landing_date
FROM SPACEXTABLE
WHERE "Landing_Outcome" = "Success (ground pad)"
```

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

first_successful_landing_date
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- The query lists the names of booster versions that have successfully landed on a drone ship and carried payloads with masses between 4000 kg and 6000 kg.
- This information highlights boosters capable of handling mid-range payloads while achieving successful drone ship landings



Task 6

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

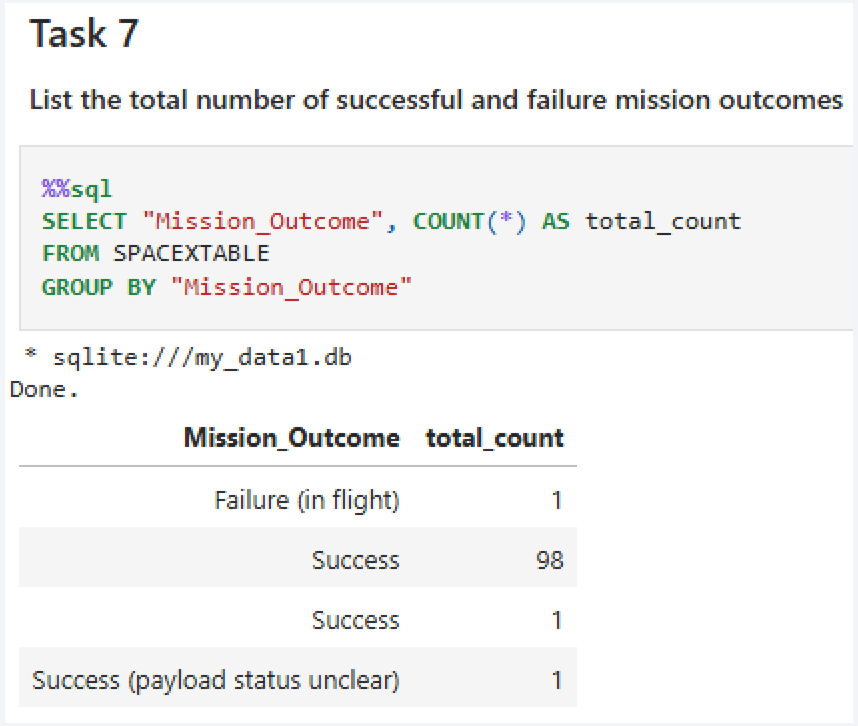
```
%%sql
SELECT "Booster_Version"
FROM SPACEXTABLE
WHERE "Landing_Outcome" = "Success (drone ship)"
AND "PAYLOAD_MASS__KG_" > 4000
AND "PAYLOAD_MASS__KG_" < 6000
```

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

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

Total Number of Successful and Failure Mission Outcomes

- The query counts the total number of missions grouped by their outcomes.
- The results reveal a dominant number of successful missions, totaling 99 when combining the exact and slightly unclear success categories.
- There is only one recorded mission failure during flight, indicating a high overall mission success rate in the dataset.



Task 7

List the total number of successful and failure mission outcomes

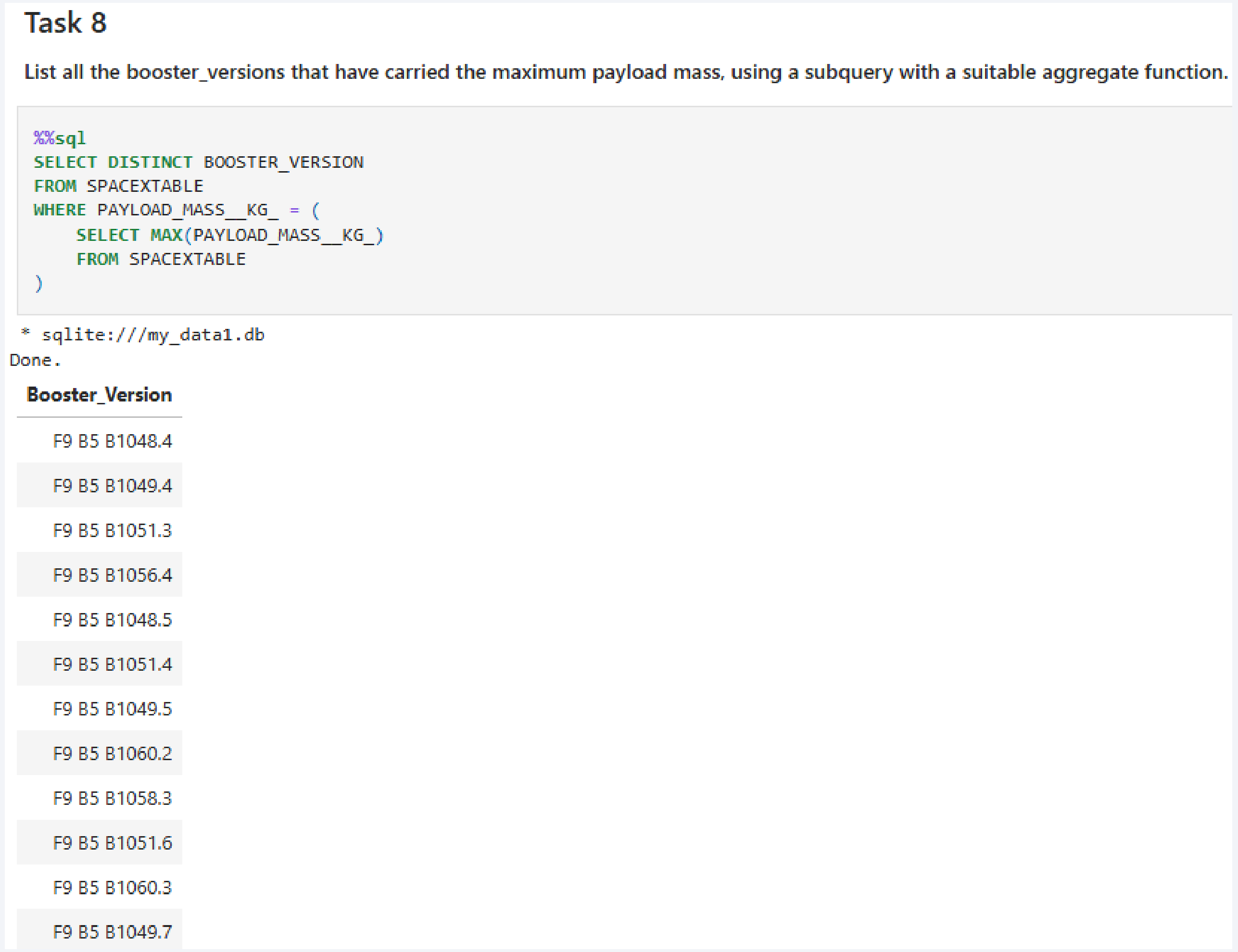
```
%%sql
SELECT "Mission_Outcome", COUNT(*) AS total_count
FROM SPACEXTABLE
GROUP BY "Mission_Outcome"
```

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

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

Boosters Carried Maximum Payload

- The query retrieves the distinct booster versions that have carried the maximum payload mass recorded in the dataset.
- This highlights the consistent capability of the F9 B5 series booster family in handling the heaviest payloads across multiple missions



Task 8

List all the booster_versions that have carried the maximum payload mass, using a subquery with a suitable aggregate function.

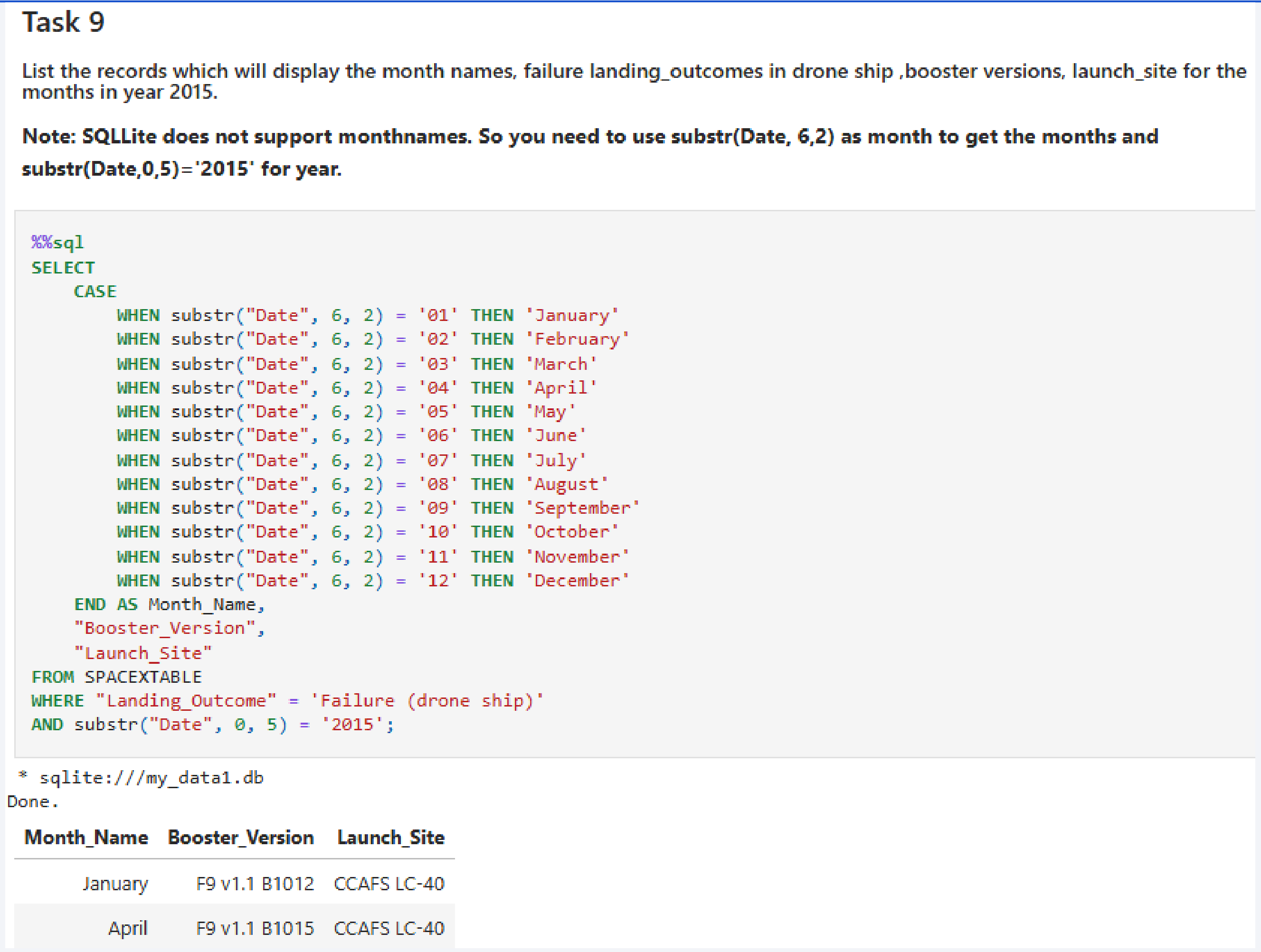
```
%%sql
SELECT DISTINCT BOOSTER_VERSION
FROM SPACEXTABLE
WHERE PAYLOAD_MASS__KG_ = (
    SELECT MAX(PAYLOAD_MASS__KG_)
    FROM SPACEXTABLE
)
```

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

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

2015 Launch Records

- The query identifies failed landing outcomes on drone ships during 2015, specifying the booster versions and launch sites involved.
- The data reveals two incidents: one in January with booster version F9 v1.1 B1012 and another in April with booster version F9 v1.1 B1015, both launched from CCAFS LC-40.
- This information highlights specific instances and equipment associated with landing failures in that year.



Task 9

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, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

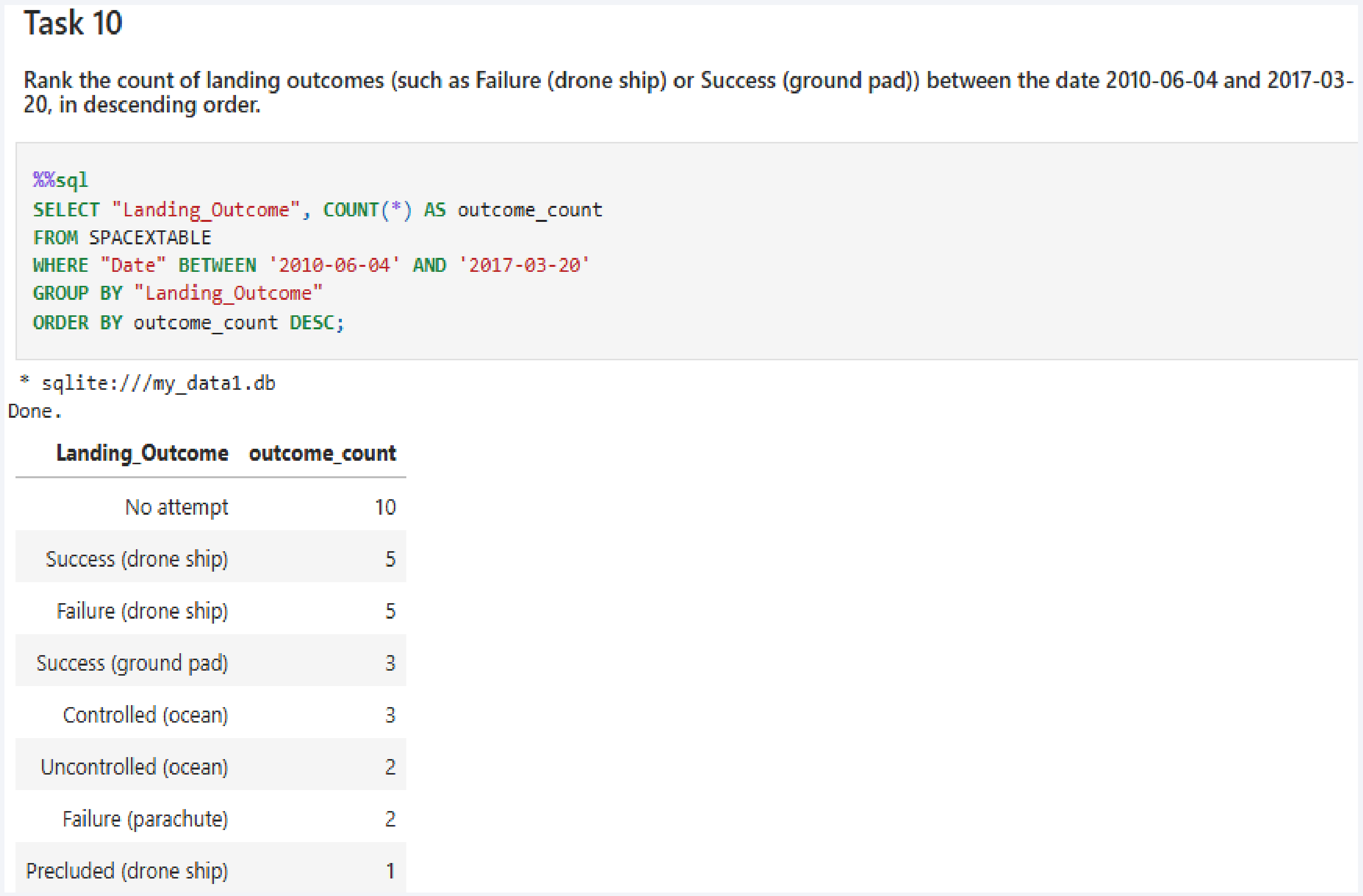
```
%%sql
SELECT
    CASE
        WHEN substr("Date", 6, 2) = '01' THEN 'January'
        WHEN substr("Date", 6, 2) = '02' THEN 'February'
        WHEN substr("Date", 6, 2) = '03' THEN 'March'
        WHEN substr("Date", 6, 2) = '04' THEN 'April'
        WHEN substr("Date", 6, 2) = '05' THEN 'May'
        WHEN substr("Date", 6, 2) = '06' THEN 'June'
        WHEN substr("Date", 6, 2) = '07' THEN 'July'
        WHEN substr("Date", 6, 2) = '08' THEN 'August'
        WHEN substr("Date", 6, 2) = '09' THEN 'September'
        WHEN substr("Date", 6, 2) = '10' THEN 'October'
        WHEN substr("Date", 6, 2) = '11' THEN 'November'
        WHEN substr("Date", 6, 2) = '12' THEN 'December'
    END AS Month_Name,
    "Booster_Version",
    "Launch_Site"
FROM SPACEXTABLE
WHERE "Landing_Outcome" = 'Failure (drone ship)'
AND substr("Date", 0, 5) = '2015';
```

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

Month_Name	Booster_Version	Launch_Site
January	F9 v1.1 B1012	CCAFS LC-40
April	F9 v1.1 B1015	CCAFS LC-40

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

- The query ranks the count of different landing outcomes recorded between June 4, 2010, and March 20, 2017, sorted in descending order.
- The results show that 'No attempt' outcomes are the most frequent, followed by equal counts of successful and failed drone ship landings.
- Other outcomes such as successful ground pad landings, controlled ocean landings, and failures by parachute are also represented, providing a comprehensive overview of landing performance and attempts during this period.



Task 10

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.

```
%%sql
SELECT "Landing_Outcome", COUNT(*) AS outcome_count
FROM SPACEXTABLE
WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY "Landing_Outcome"
ORDER BY outcome_count DESC;
```

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

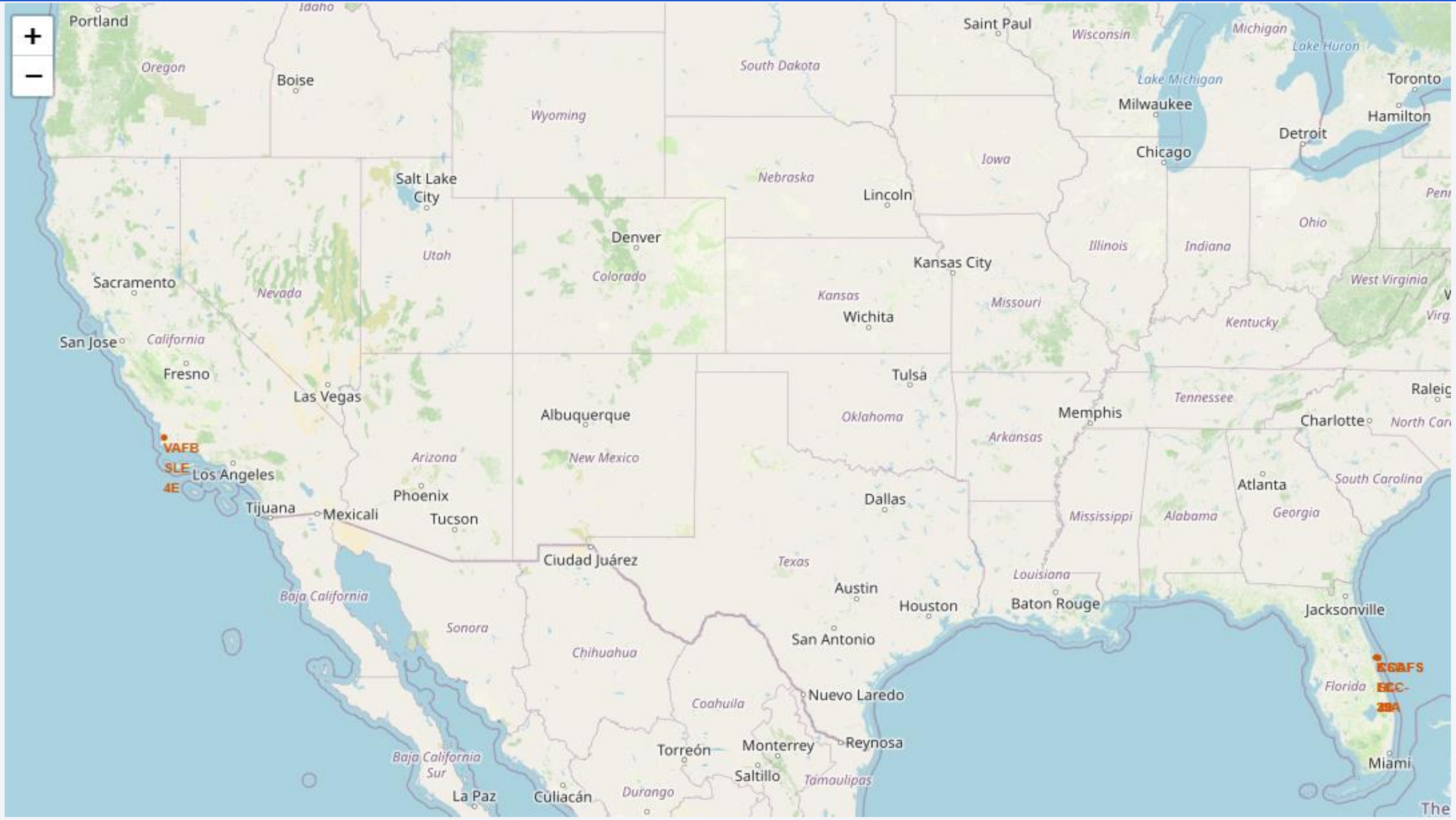
Landing_Outcome	outcome_count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

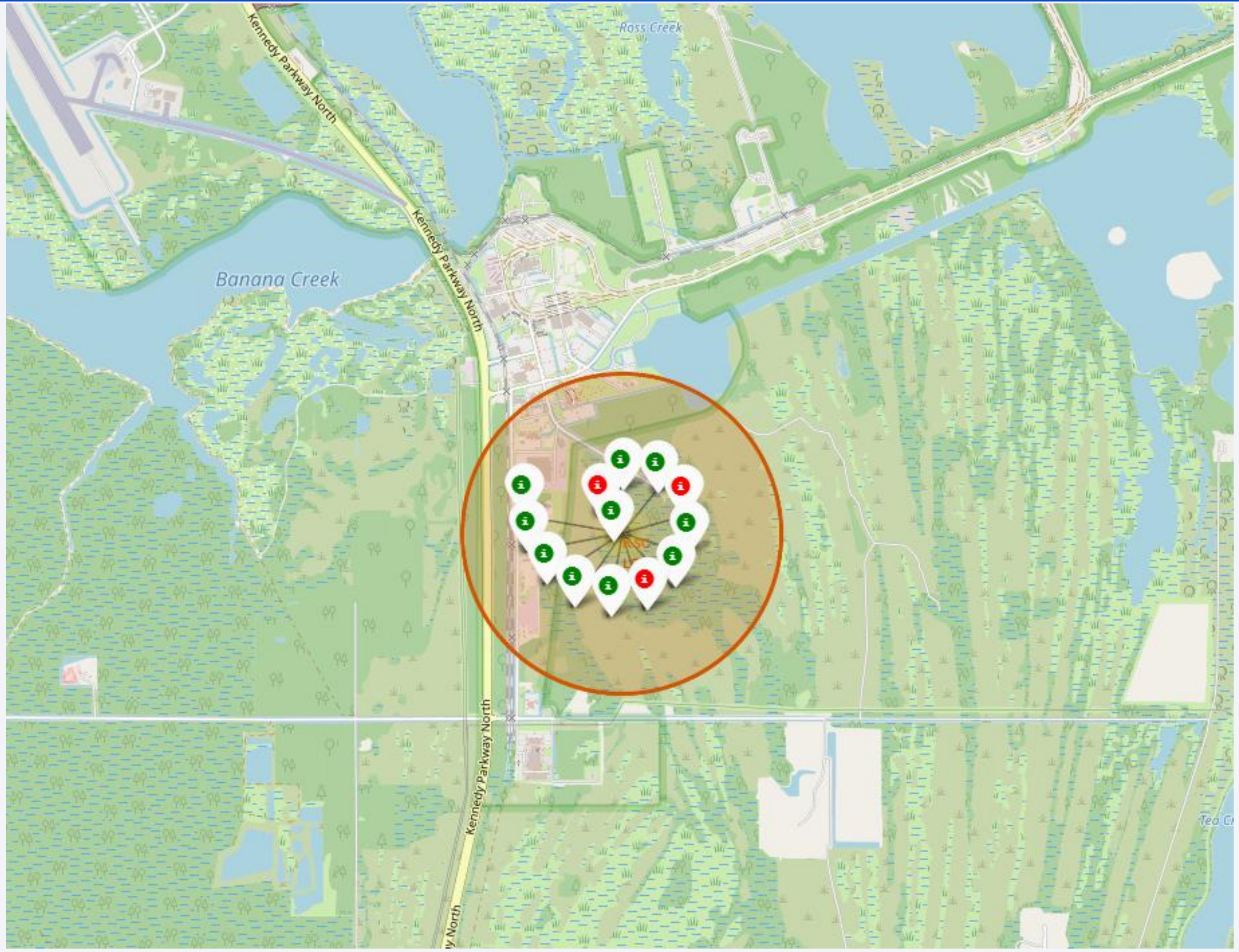
Geographic Distribution of SpaceX Launch Sites in the United States

- The map displays the geographic positions of SpaceX launch sites, each marked with a labeled icon and a circular area.
- Site names are clearly indicated for easy identification and comparison.



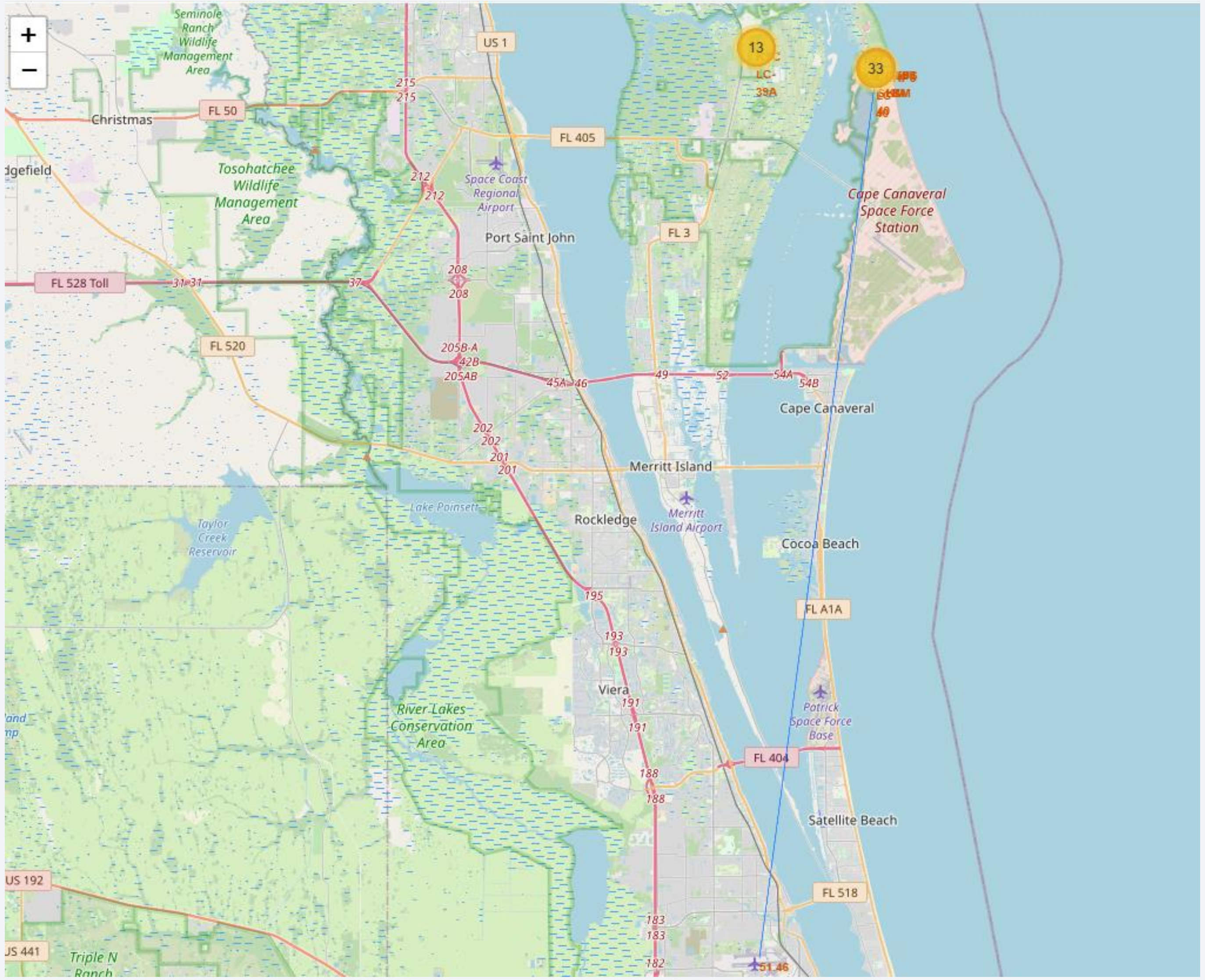
Color-Coded Launch Outcomes at SpaceX Launch Sites

- The map displays clustered launch markers, each color-coded to represent mission outcomes.
- Successes and failures are visually distinguished using different icon colors, enabling quick assessment of performance at various sites.



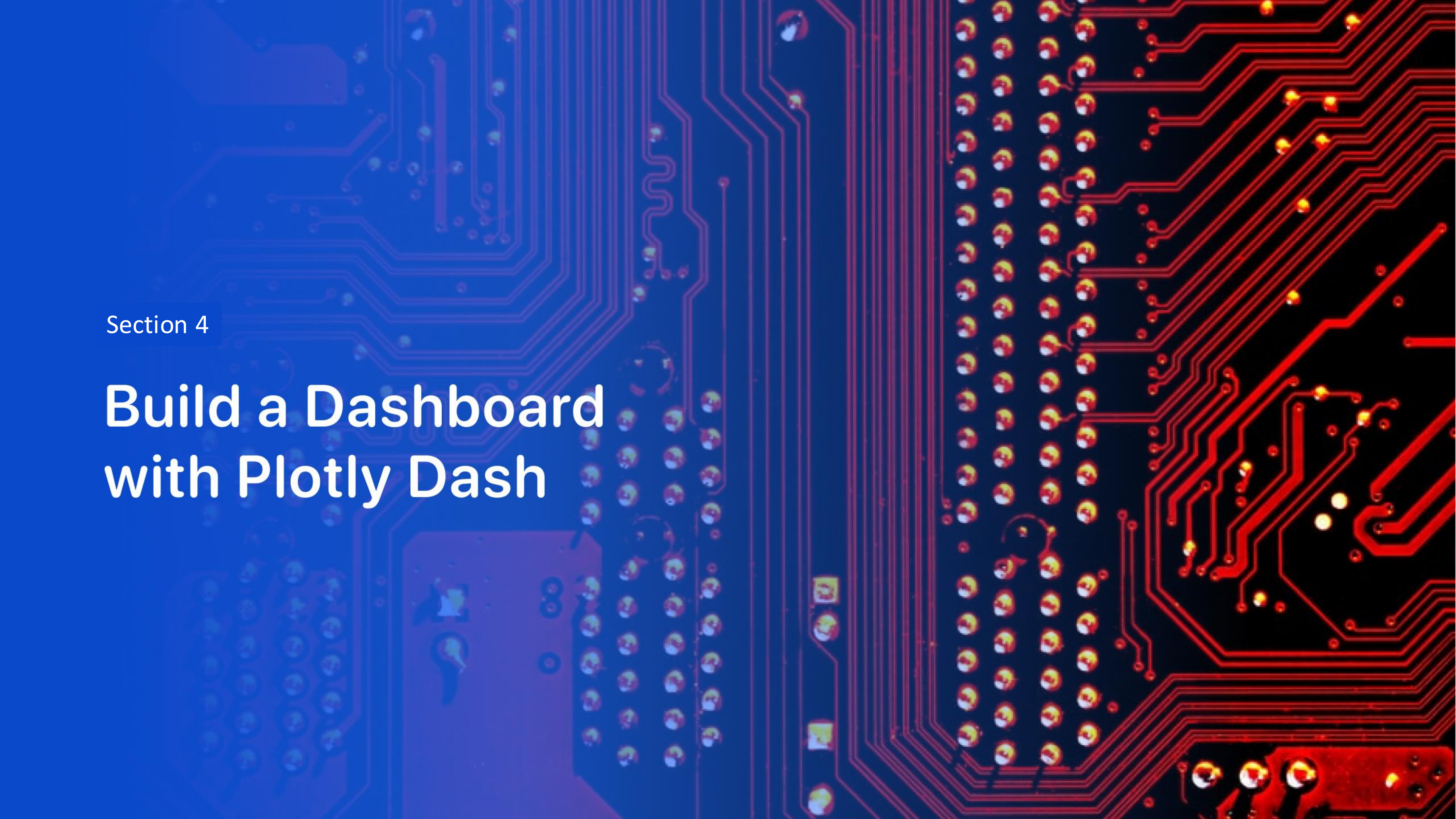
Launch Site Proximity to Infrastructure and Urban Areas

- The map highlights a selected launch site and its proximity to key infrastructure elements, including the nearest highway, railroad, and city.
- Distances to each point are displayed with labeled markers and connecting lines.
- This spatial analysis helps assess the site's logistical accessibility and potential constraints related to transportation and urban presence.



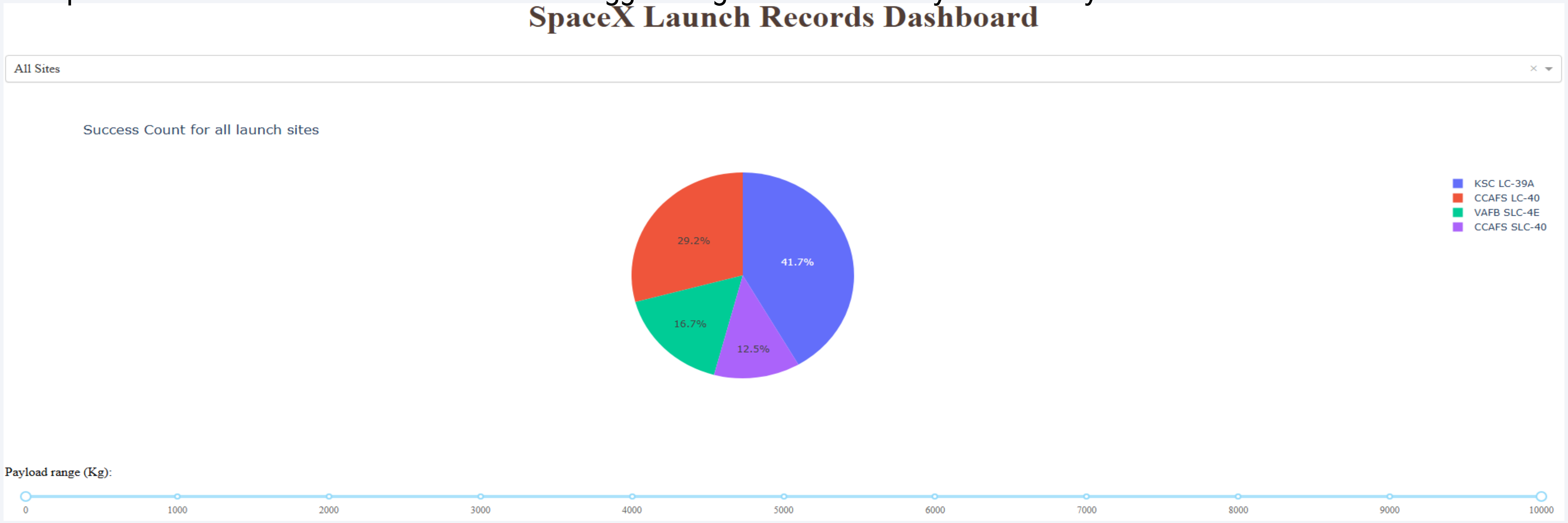
Section 4

Build a Dashboard with Plotly Dash



Distribution of Successful Launches by Site

- The pie chart illustrates the proportion of successful launches attributed to each launch site.
- Among them, **KSC LC-39A** stands out with the largest share, accounting for approximately **41.7%** of all recorded successes.
- In contrast, **CCAFS SLC-40** contributes the least, with a success rate around **12.5%**. This distribution highlights the operational dominance of KSC LC-39A and suggests higher launch activity or efficiency at that site.

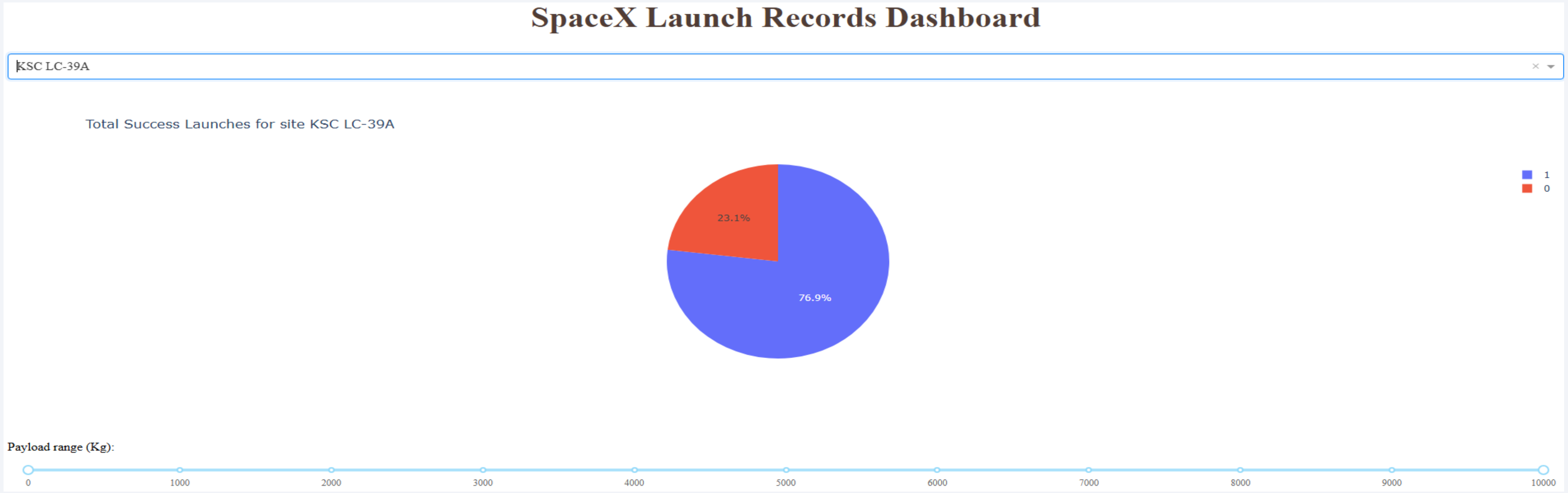


SpaceX Launch Records Dashboard

All Sites

Success vs. Failure Rate at KSC LC-39A Launch Site

- The pie chart provides a focused breakdown of launch outcomes specifically at **KSC LC-39A**, the site with the highest number of successful missions.
- The data reveals a **76.9% success rate**, indicating a strong operational track record, while **23.1%** of launches did not achieve full mission success.
- This highlights the site's reliability while still acknowledging the occurrence of failures.



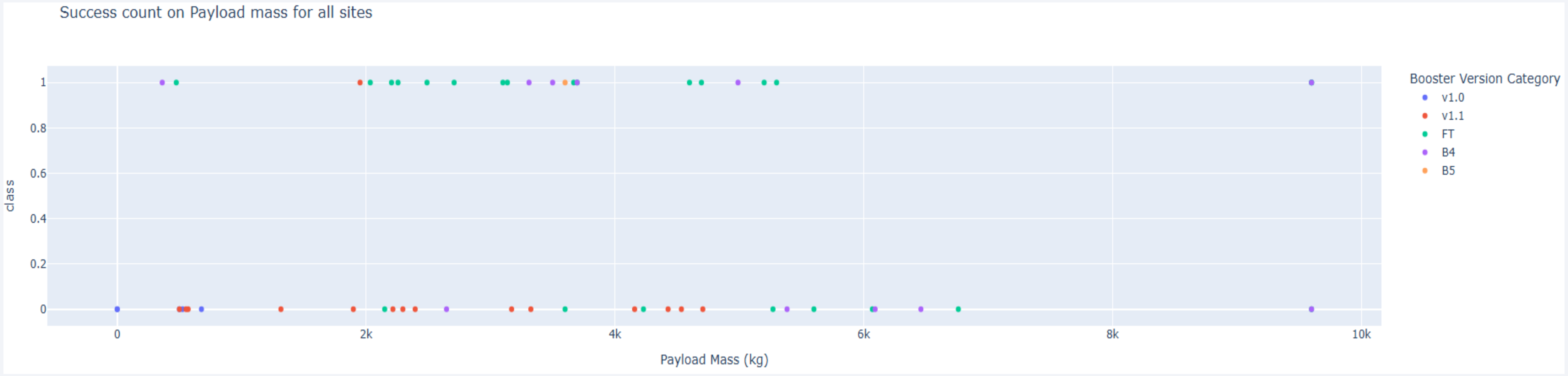
SpaceX Launch Records Dashboard

KSC LC-39A

Total Success Launches for site KSC LC-39A

Correlation Between Payload Mass and Launch Outcomes Across All Sites

- The scatter plot reveals that payloads in the **2,000 to 4,000 kg** range correspond to the highest concentration of successful launches, followed closely by the **4,000 to 6,000 kg** range. This pattern is evident from the dense clustering of success markers within these payload intervals.
- Additionally, among the various booster versions, **FT** (represented by green markers) demonstrates the **highest rate of success**, with **B4** (purple markers) following as the second most reliable configuration. This suggests a strong correlation between specific payload ranges and booster models in achieving mission success

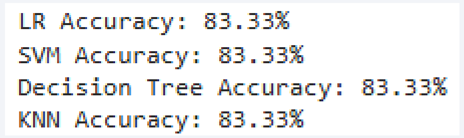


Section 5

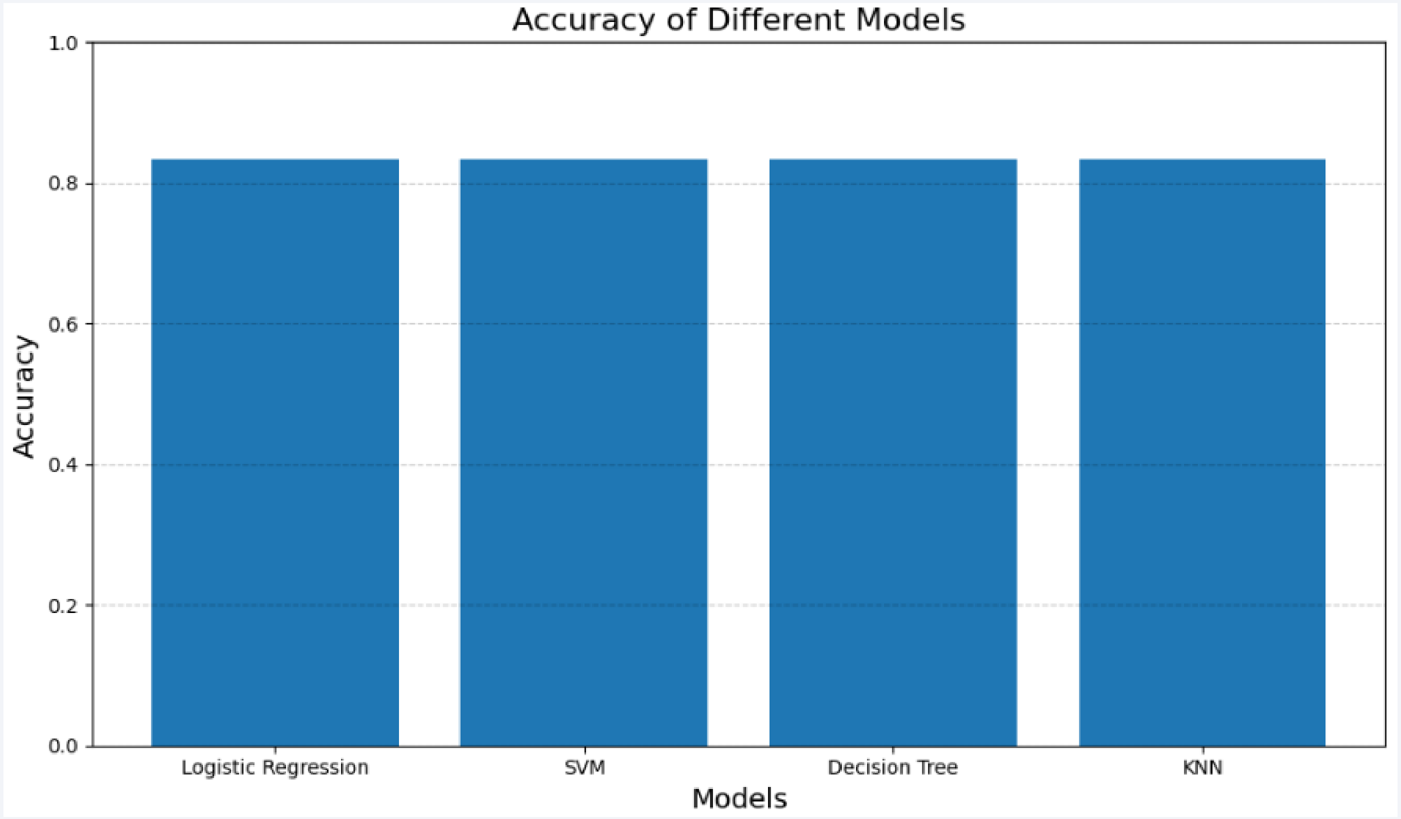
Predictive Analysis (Classification)

Classification Accuracy

- The bar chart presents a comparison of classification accuracy across four machine learning models: Logistic Regression, Support Vector Machine, Decision Tree, and K-Nearest Neighbors.
- The results indicate that **all models achieved an identical accuracy of 83.33%**, suggesting comparable performance on the evaluated dataset.
- No single algorithm outperformed the others in this specific case, indicating either a balanced dataset or limited feature differentiation.

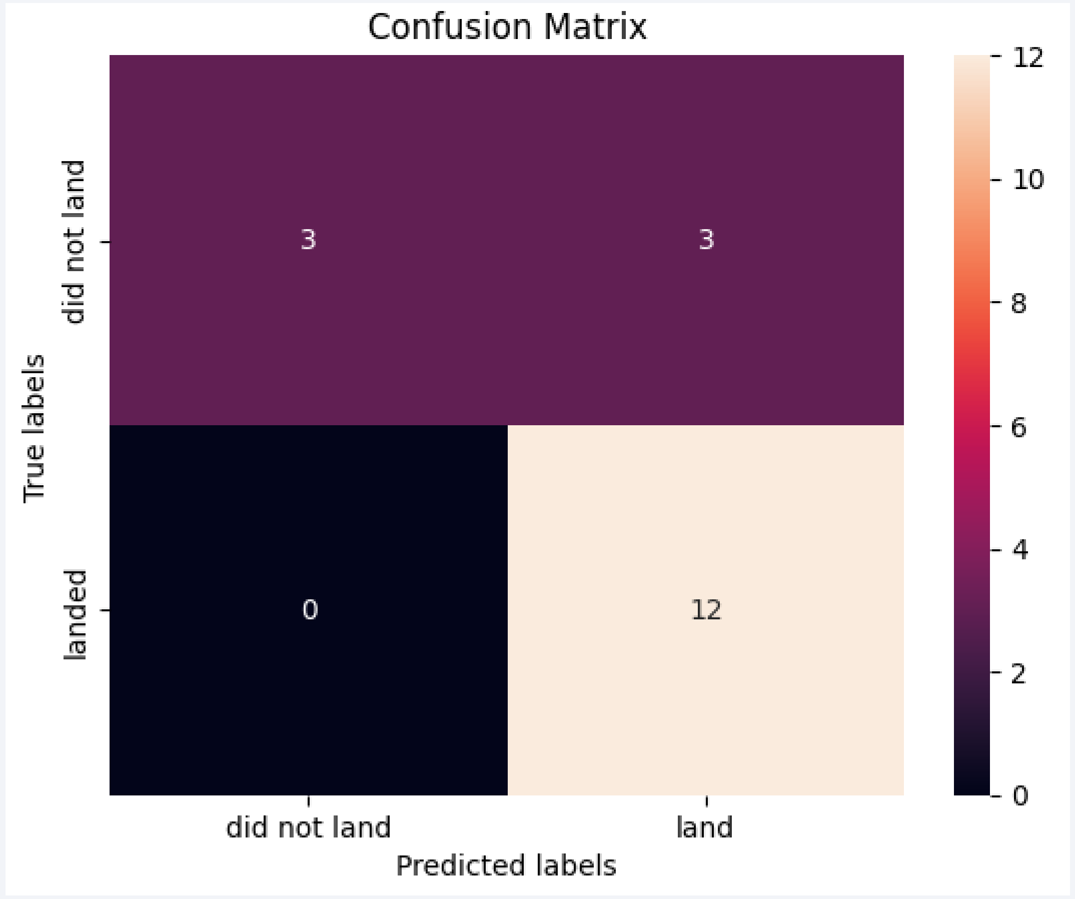


```
LR Accuracy: 83.33%
SVM Accuracy: 83.33%
Decision Tree Accuracy: 83.33%
KNN Accuracy: 83.33%
```



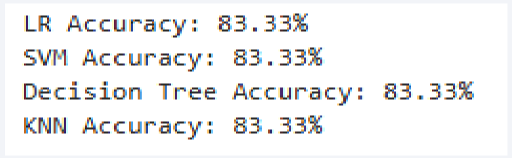
Confusion Matrix

- The confusion matrix is identical across all evaluated classification models, indicating consistent behavior in their predictions. Each model correctly classified **12 true positives** and **3 true negatives**, while misclassifying **3 false positives** and recording **no false negatives**.
- This suggests the models are **equally effective**, with a slight inclination toward overpredicting the positive class. The absence of false negatives is particularly relevant for applications where missing a positive case would be costly.

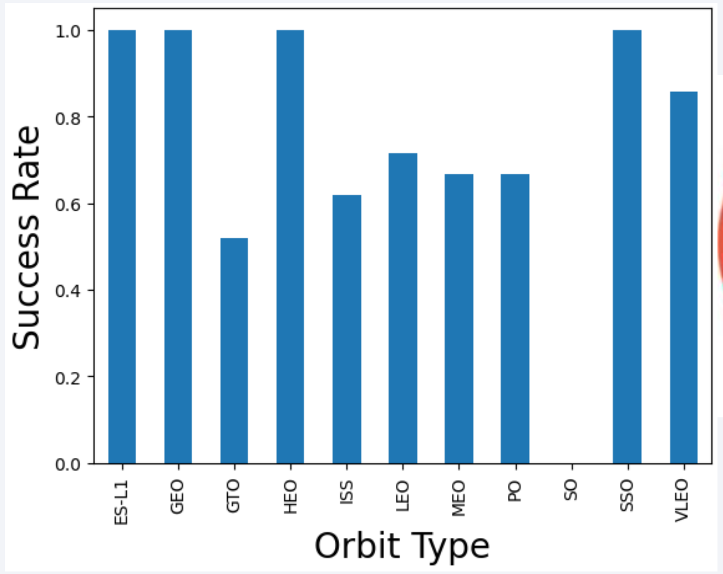


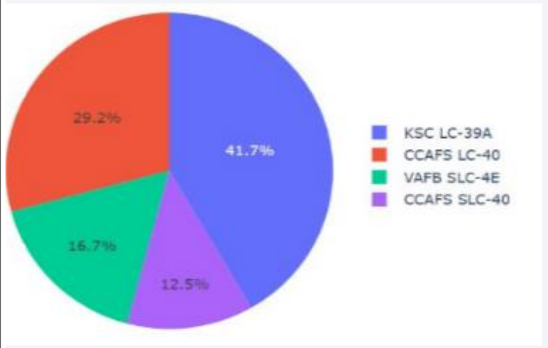
Conclusions

- Logistic Regression, SVM, Decision Tree, and KNN all demonstrated comparable high performance in predicting launch outcomes in this dataset.
- Launches with lighter payloads consistently demonstrate higher success rates compared to those carrying heavier payloads.
- The likelihood of a SpaceX launch succeeding increases with operational experience over the years, indicating a trend towards near-flawless launches as the company matures.
- Kennedy Space Center's Launch Complex 39A records the highest number of successful launches among all launch sites analyzed.
- Orbits such as GEO, HEO, SSO, and ES L1 stand out for their superior success rates, showing the most reliable launch outcomes.



```
LR Accuracy: 83.33%
SVM Accuracy: 83.33%
Decision Tree Accuracy: 83.33%
KNN Accuracy: 83.33%
```





Thank you!