IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

By: Greg Bolzon
SEP 2021

Outline

Executive Summary

- The purpose of this report is to predict the successful landings of Falcon 9 first stage rockets.
- Through web scraping publicly available data, analyzing historical launches and their successes, multiple predictive models were developed to determine if the first stage of the rocket will land successfully.
- Through successful analysis, a categorical model using (decision trees, logicstic regression, Support Vector Model?, or other) was able to predict a successful landing with XX% accuracy using historical data.

Introduction

- SpaceX's Falcon 9 rocket is a reusable first stage rocket which.
- By being able to reuse the first stage rocket, the cost of a launch is determined to be $62 million compared to other flights which cost $162 million.
- If a successful landing can be predicted, the cost of a launch can be determined.
- Using all public data available, the project aims to develop a predictive model to accurately predict if a landing will be successful.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Accessing publicly available data through webscraping and modifying that data into a usable format.
- Perform data wrangling
  - Analyzing the performing calculation on the data for further exploratory analysis.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Building models, tuning the models through parameters, and evaluating the results

Main GitHub Site

https://github.com/GB72-creator/Capstone_Project

Data Collection

- Public data was made available and collected through Wikipedia sites. Each site is in the respective workbook.
- Information collected from tables were converted to DataFrames where further analysis was conducted.
- Data was collected through API and Webscraping functions in python.

Data Collection – SpaceX API

- With the public data available, the data was imported using the API calls.
- Once the data was imported, the DataFrame was modified to remove unnecessary values.
- Through exploring the dataset there were a few missing values in the PayLoad, so the mean of values were used to replace the missing values.
- https://github.com/GB72-creator/Capstone_Project/blob/main/1_GB_Data_Collection_With_API.ipynb

SPACEX API Calls Flowchart

Import Libraries > Get Booster Names from Rocket Column > Get Launch Pad data and name LaunchSite > get Payload Data and name payloads > get the outcomes of the core data like reuse, leg reuse, and more > request rocket launch data from SpaceX API >

Check response code (200) > import data as DataFrame and call it data > check type(data) > get specifics like booster name, payload, launch site and store it as new DataFrame > Store data and define dictionary >

Filter DataFrame to only include Falcon 9 launches > Remove Flight Number Column >

Replace missing values in payload mass with average

Data Collection - Scraping

- Through the public data we extract the information on the launches based on the html tags like table through the find all function in BeautifulSoup.
- Next we build the dataframes for data analysis.
- https://github.com/GB72-creator/Capstone_Project/blob/main/2_Webscraping.ipynb

Get website > extract data > filter data into DataFrame > add values > finalize DataFrame

Data Wrangling

- The data was imported to a DataFrame.
- There the data was analyzed for missing and categorical values
- It was identified there were multiple outcomes of good and bad landings
    - Utilized code to separate out the good landings (1) and the bad landings (0).
- Calculated the mean of successful landings.
- https://github.com/GB72-creator/Capstone_Project/blob/main/3_GB_labs-jupyter-spacex-Data%20wrangling.ipynb

Importing libraries > reading in data as DataFrame > calculating percentage of missing values > identifying categorical vs numerical columns >

Value Counts to determine number of launch sites > performed Value Counts on orbits > Value Counts on landing outcomes > identify landing outcomes as keys > identify bad outcomes and classify their keys > apply landing class identifier for bad outcomes as 0 value, 1 value if it is a good outcome. > determine the mean number of successful landings

EDA with Data Visualization

- Depicted in the table is a summary of all the charts that were plotted as well as a justification for each plot.

- https://github.com/GB72-creator/Capstone_Project/blob/main/4_GB_Data_Visualization.ipynb

Chart Type	Chart Representation	Reason
Scatter	Flight Number vs PayloadMass vs Outcome	Show the payload mass and the outcome over the number of flights (or time)
Catplot (Categorical)	Flight Number vs Launch Site vs Outcome	Show the Launch Sites and the Outcome of the flight as well as the number (or time) the flight was initiated.
Catplot (Categorical)	Flight Number vs Payload Mass vs Outcome	Show the Launch Sites and the Outcome of the flight as well as the payload carried during the flight.
Bar Chart	Orbit and Success Rate	Show the success rate of each orbit of launch.
Catplot (Categorical)	Flight Number vs Orbit vs Outcome	Show the Launch Sites and the Outcome of the flight as well as the orbit during the flight.
Catplot (Categorical)	Orbitvs Payload Mass vs Outcome	Show the Orbit and the Outcome of the flight as well as the Payload Mass during the flight.
Line	Year vs Success Rate	Show the success rate of each launch by year

EDA with SQL

- Performed SQL queries to find:
  - Launch sites
  - First launch dates
  - Average payload masses of boosters from NASA
  - Number of successful missions
  - Landing outcomes
  - 2017 Launch records
  - Maximum payload booster versions

- https://github.com/GB72-creator/Capstone_Project/blob/main/5_EDX_With_SQL.ipynb

Build an Interactive Map with Folium

- Utilized folium map to mark the launch sites on an interactive map
- Further explored each site with coloured markers to indicate successful and unsuccessful landings at each site location.
- Added in key points of interest like railways and coastlines to highlight the proximity of important areas to the launch sites itself.

- https://github.com/GB72-creator/Capstone_Project/blob/main/6_Folium_Lab_GB.ipynb

Build a Dashboard with Plotly Dash

- Dashboard was not created as I was struggling and unable to create the proper code for it to work.
  - Apologies
- The plots are important because they allow any stakeholder who is interested in the data to easily go and acquire the information.
- https://github.com/GB72-creator/Capstone_Project/blob/main/7_GB_Daahboards.ipynb

Predictive Analysis (Classification)

- Took the full dataset and split it into test and training sets.
- Brought in 4 different classifier models to predict the outcomes of a landing.
- Utilized multiple parameters and GridSearchCV function to find the best model parameters to use in the model.
- Used to model to predict the outcome and compare to the actual value.
- 3 of the 4 models all predicted an outcome of 83% accuracy with no False Negatives
- Model outcomes changed with the version of python used even with random seed set
- https://github.com/GB72-creator/Capstone_Project/blob/main/8_GB_ML_Prediction.ipynb

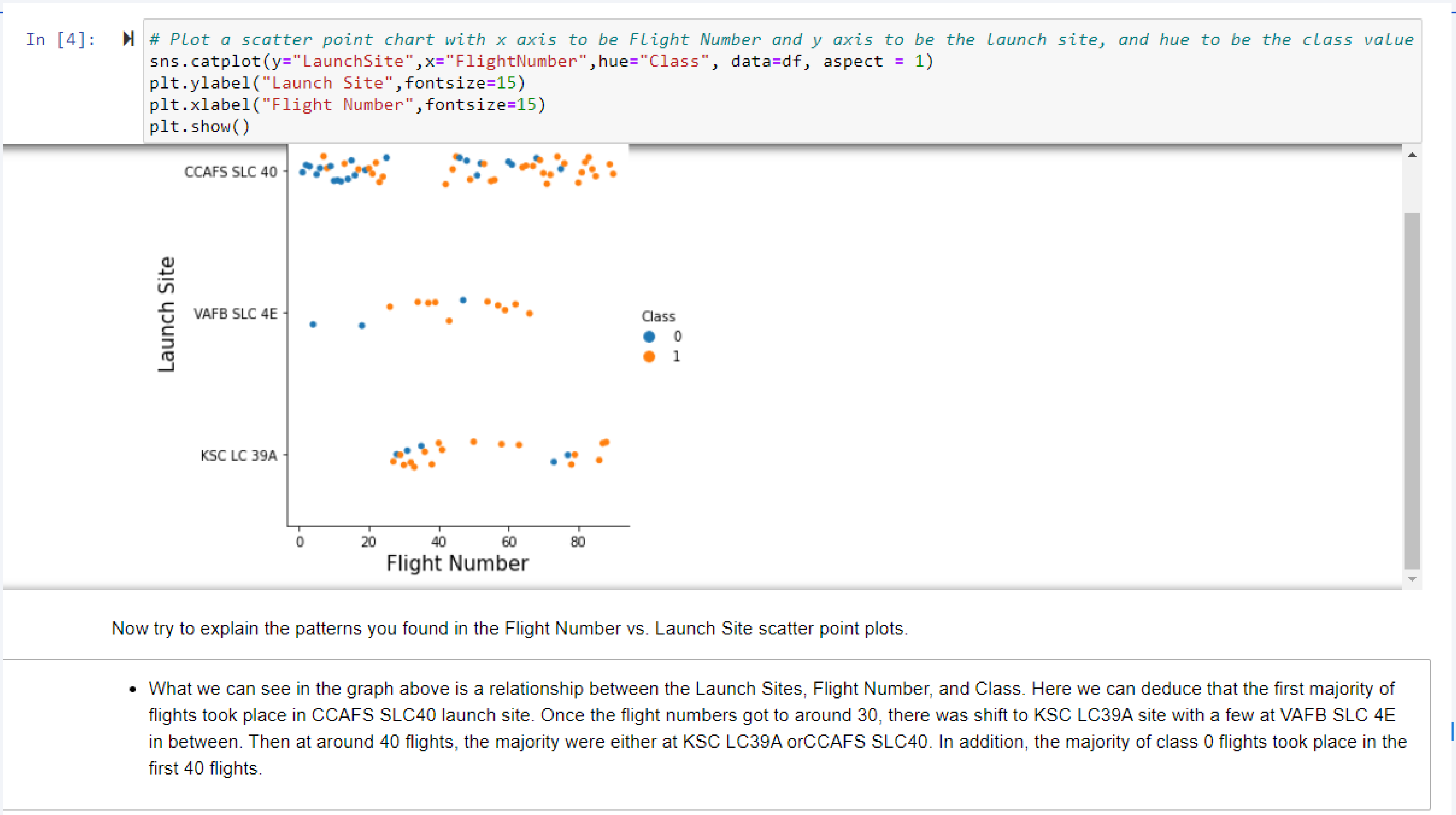
Results

- The next section of slides present the results of the analysis:
    - Exploratory data analysis results
    - Interactive analytics demo in screenshots
    - Predictive analysis results

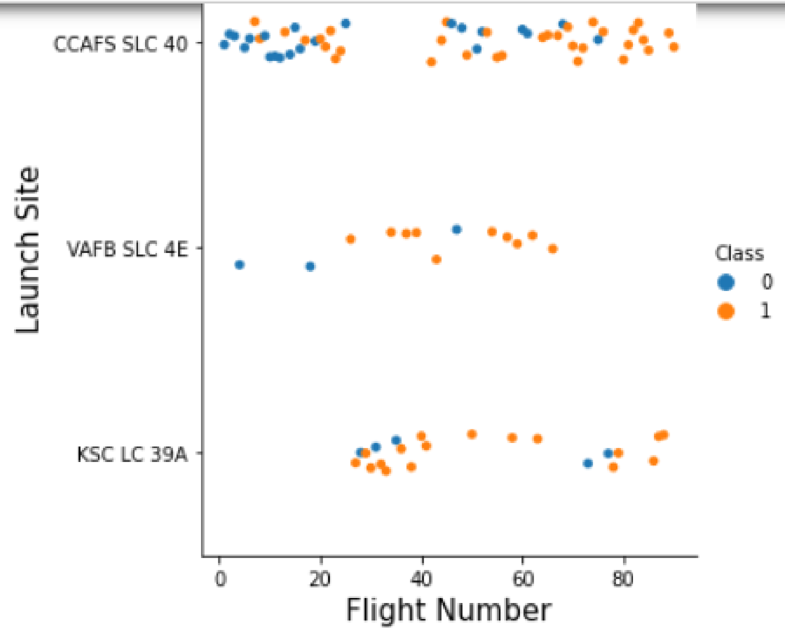
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site



```
In [4]:  # 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 = 1)
         plt.ylabel("Launch Site",fontsize=15)
         plt.xlabel("Flight Number",fontsize=15)
         plt.show()
```

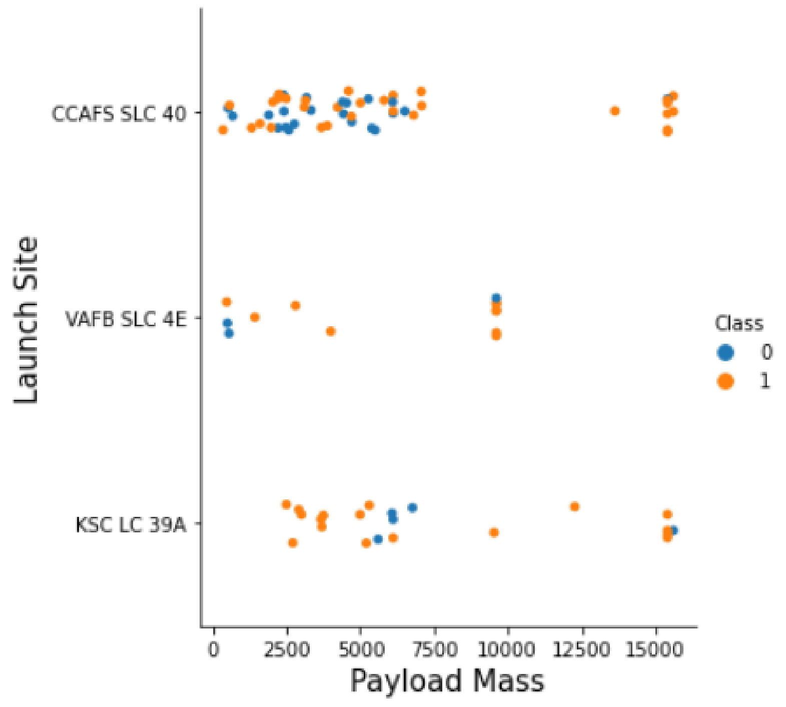


Now try to explain the patterns you found in the Flight Number vs. Launch Site scatter point plots.

- What we can see in the graph above is a relationship between the Launch Sites, Flight Number, and Class. Here we can deduce that the first majority of flights took place in CCAFS SLC40 launch site. Once the flight numbers got to around 30, there was shift to KSC LC39A site with a few at VAFB SLC 4E in between. Then at around 40 flights, the majority were either at KSC LC39A orCCAFS SLC40. In addition, the majority of class 0 flights took place in the first 40 flights.

Payload vs. Launch Site

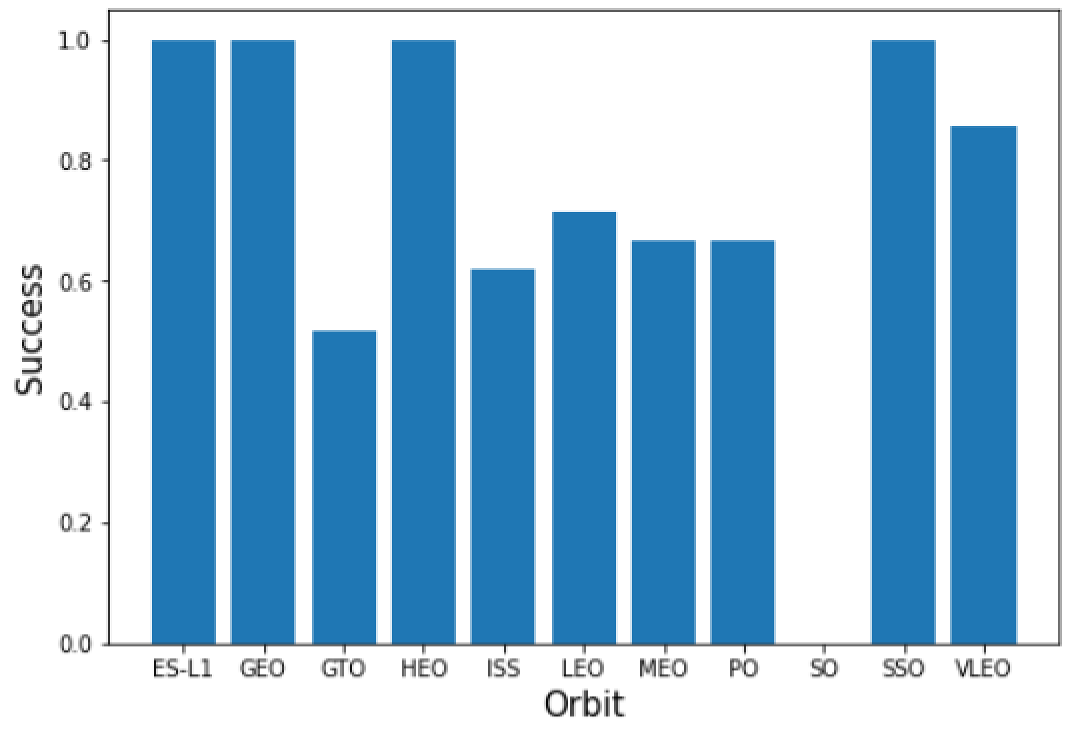
```
In [5]:  # Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class
         sns.catplot(y="LaunchSite",x="PayloadMass",hue="Class", data=df, aspect = 1)
         plt.ylabel("Launch Site",fontsize=15)
         plt.xlabel("Payload Mass",fontsize=15)
         plt.show()
```



Now try to explain any patterns you found in the Payload Vs. Launch Site scatter point chart.

There is a pattern for payload mass and launch site. Payload mass looks to be scattered in up to 7500. KSC was not able to handle the higher payloads as CCAFS does not have a distinguishable patter in the lower mass tier. Once the mass went above 7500, ,there were few instances of Class 0 at any of the launch facilities.

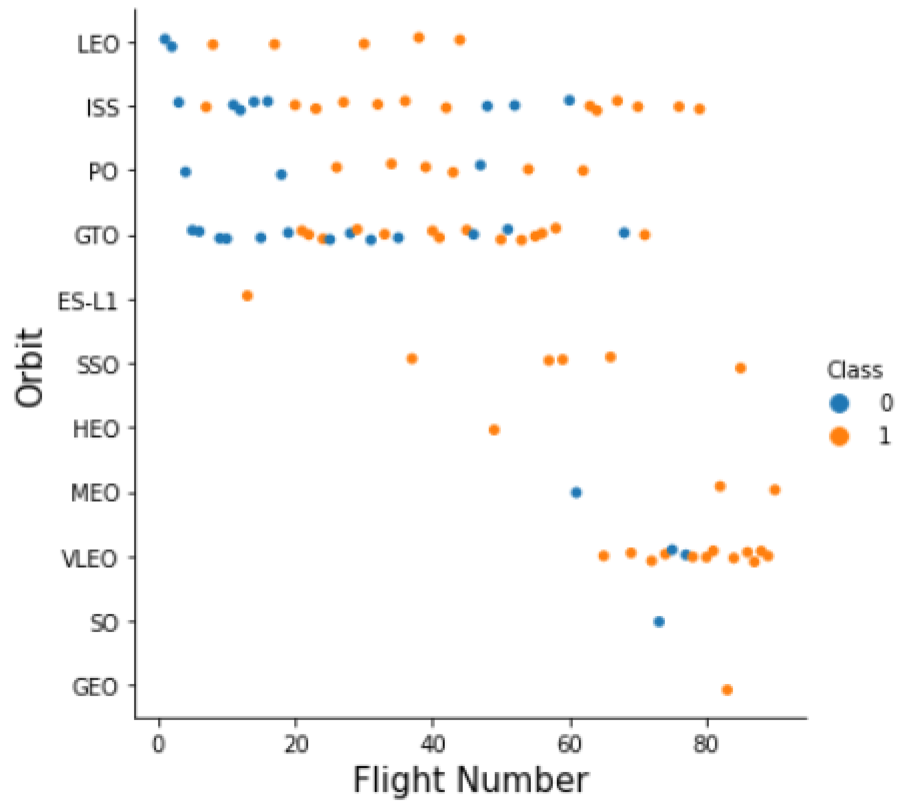
Success Rate vs. Orbit Type



Analyze the ploted bar chart try to find which orbits have high sucess rate.

```
In the Bar graph above there is a clear mark of success rates based on the orbit. LES-L1, GEO, Hight Earth Orbit (HEO), SSO were
all 100% succes sartes. SO had none and all the other had a success rate between 50 and 75%. It is unknown in this graph how
many launches were in that orbit which could contribut to their success rates.
```

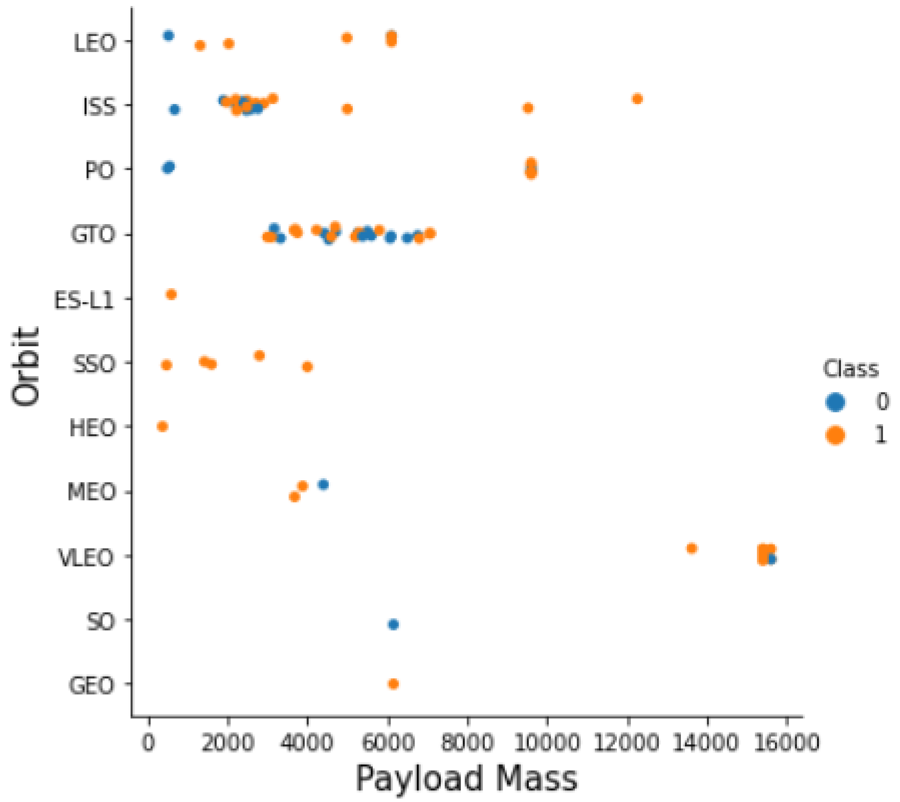
Flight Number vs. Orbit Type



You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

```
In addition, there is a clear that the GEO and ES-L1 success rates were contrinuted by the fact there was only 1 launch in this
orbit, as well as SO having no successful flights as only 1 flight in that orbit is in the dataset and it was unsuccessful. In
addition, the majority of the early flights were LWO and ISS and GTO, where the majority of later flight orbits were int he VLEO
category.
```

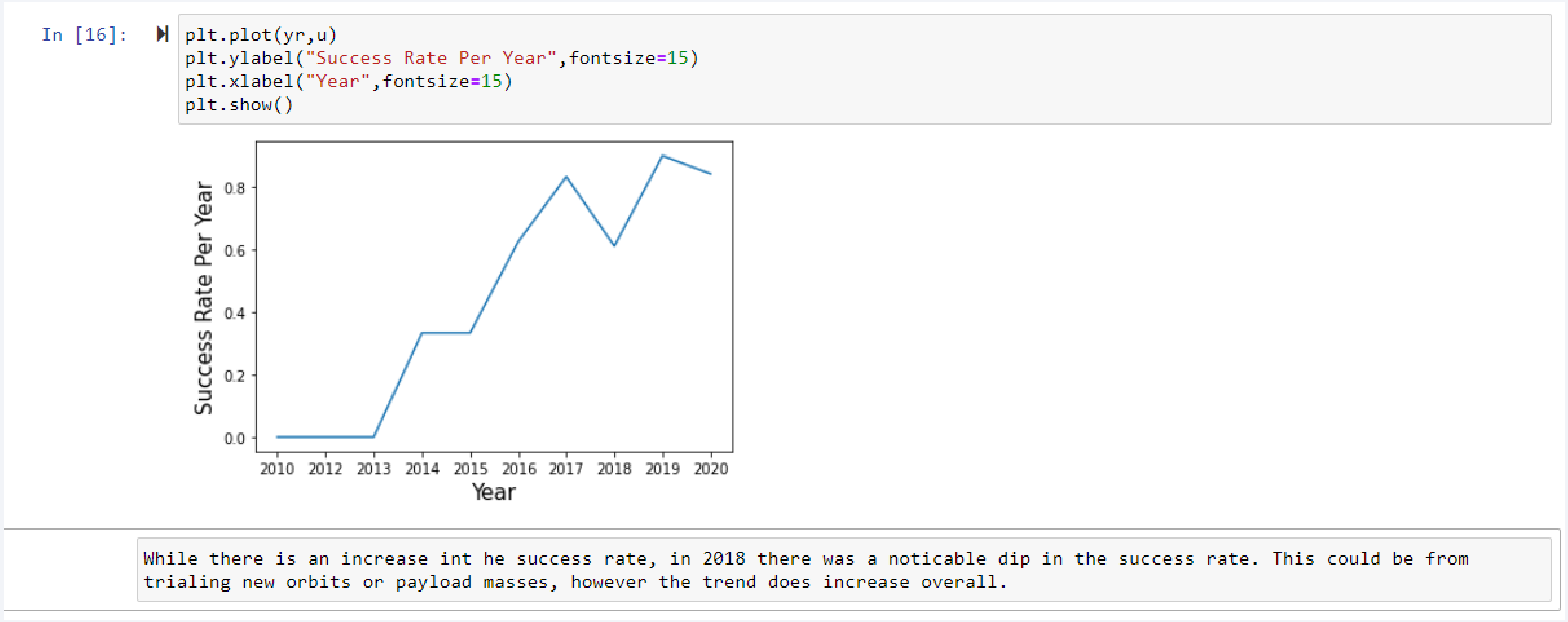
Payload vs. Orbit Type



You should observe that Heavy payloads have a negative influence on GTO orbits and positive on GTO and Polar LEO (ISS) orbits.

```
In addition, the VLEO, ISS and PO orbit types seem to handle the highest payloads, although there is little data to confirm.,
SSO is able to handle all payloads so far, but the mass is quite small compared to the payload masses which have gone into orbit
for other orbit types.
```

Launch Success Yearly Trend

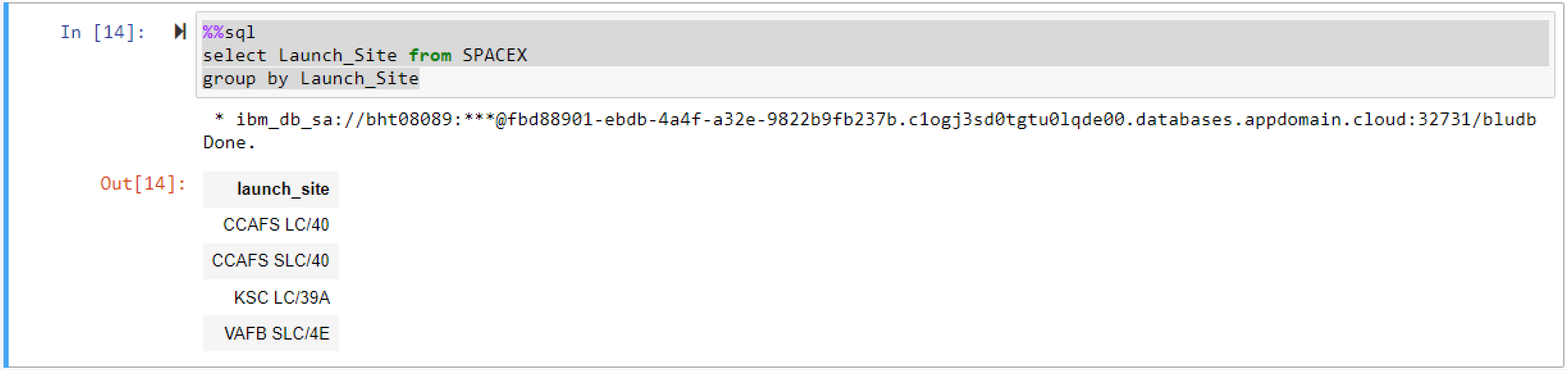


```
In [16]:  plt.plot(yr,u)
          plt.ylabel("Success Rate Per Year",fontsize=15)
          plt.xlabel("Year",fontsize=15)
          plt.show()
```

While there is an increase int he success rate, in 2018 there was a noticable dip in the success rate. This could be from trialing new orbits or payload masses, however the trend does increase overall.

All Launch Site Names

- By selecting launch site and grouping by the launch site, we were able to get the unique Launch Site names in a simple table.
- See Reference code screenshot below



In [14]:
```
%%sql
select Launch_Site from SPACEX
group by Launch_Site
```

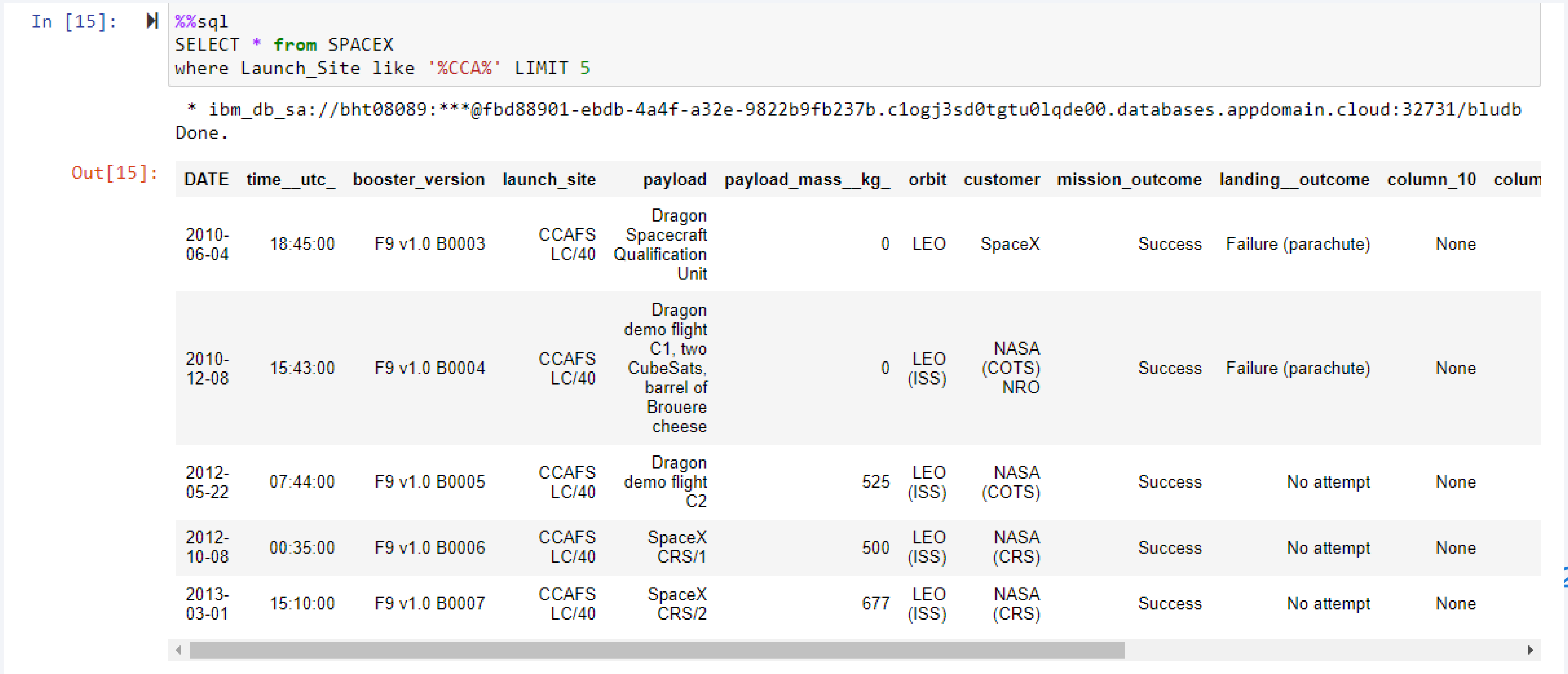
```
 * ibm_db_sa://bht08089:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Out[14]:

launch_site
CCAFS LC/40
CCAFS SLC/40
KSC LC/39A
VAFB SLC/4E

Launch Site Names Begin with 'CCA'

- By Selecting all and filtering the Launch Site to find the instance of CCA and limiting to 5 SQL was able to retrieve the table.
- See Reference code screenshot below: please not the notebook asked for KSC which is on the next slide.



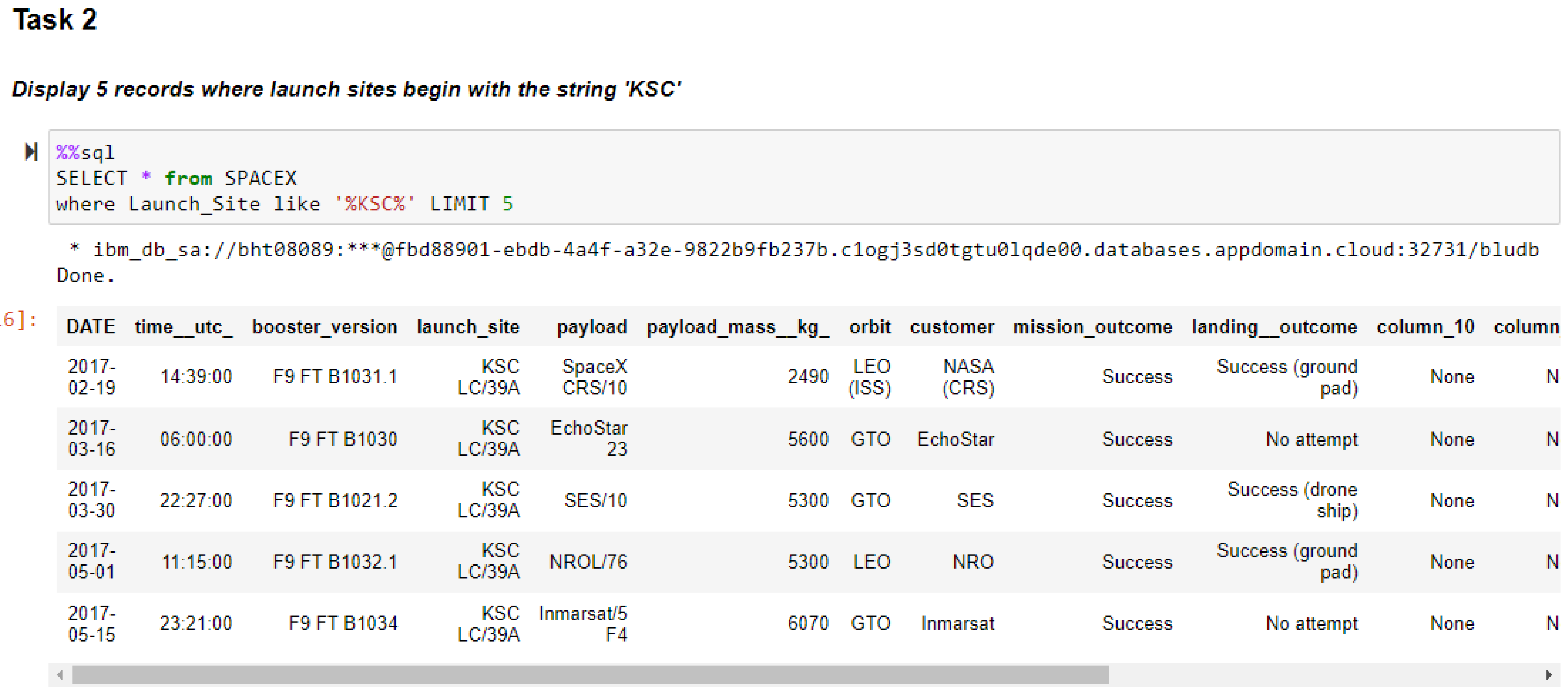
```
In [15]:  %%sql
          SELECT * from SPACEX
          where Launch_Site like '%CCA%' LIMIT 5

 * ibm_db_sa://bht08089:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Out[15]:

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome	column_10	colum
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC/40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)	None	
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)	None	
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC/40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt	None	
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC/40	SpaceX CRS/1	500	LEO (ISS)	NASA (CRS)	Success	No attempt	None	
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC/40	SpaceX CRS/2	677	LEO (ISS)	NASA (CRS)	Success	No attempt	None	

Launch Site Names Begin with 'KSC'

- By Selecting all and filtering the Launch Site to find the instance of KSC (which is what the notebook asked) and limiting to 5 SQL was able to retrieve the table.
- See Reference code screenshot below.



Task 2

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

In [16]:
```
%%sql
SELECT * from SPACEX
where Launch_Site like '%KSC%' LIMIT 5
```

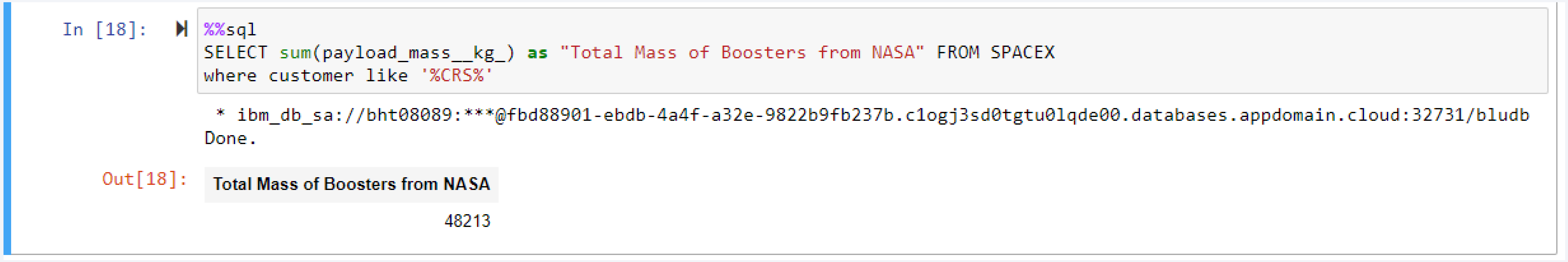
```
 * ibm_db_sa://bht08089:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Out[16]:

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome	column_10	column
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC/39A	SpaceX CRS/10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)	None	N
2017-03-16	06:00:00	F9 FT B1030	KSC LC/39A	EchoStar 23	5600	GTO	EchoStar	Success	No attempt	None	N
2017-03-30	22:27:00	F9 FT B1021.2	KSC LC/39A	SES/10	5300	GTO	SES	Success	Success (drone ship)	None	N
2017-05-01	11:15:00	F9 FT B1032.1	KSC LC/39A	NROL/76	5300	LEO	NRO	Success	Success (ground pad)	None	N
2017-05-15	23:21:00	F9 FT B1034	KSC LC/39A	Inmarsat/5 F4	6070	GTO	Inmarsat	Success	No attempt	None	N

Total Payload Mass

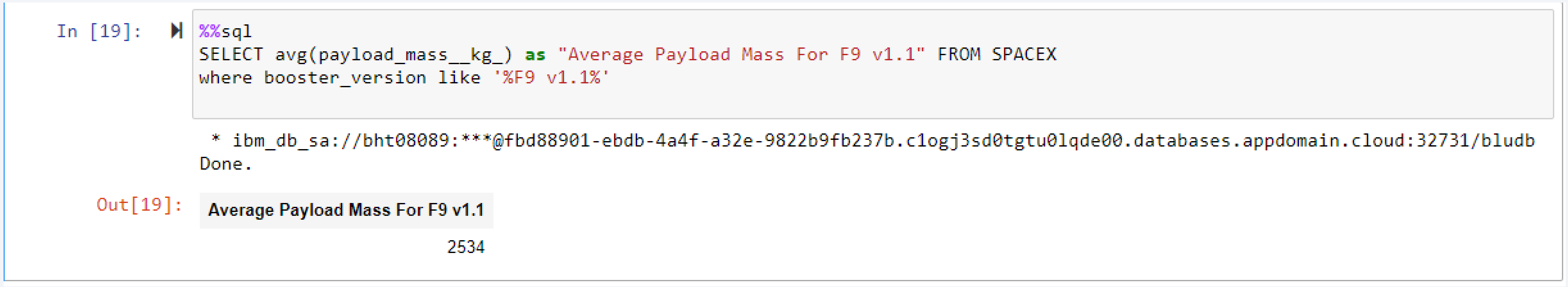
- By performing a sum on the payload mass and filtering the data to only include NASA boosters which contain CRS the total payload mass can be calculated
- See reference screenshot below



```
In [18]:  %%sql
          SELECT sum(payload_mass__kg_) as "Total Mass of Boosters from NASA" FROM SPACEX
          where customer like '%CRS%'

           * ibm_db_sa://bht08089:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
          Done.

Out[18]:  Total Mass of Boosters from NASA
                                     48213
```

Average Payload Mass by F9 v1.1

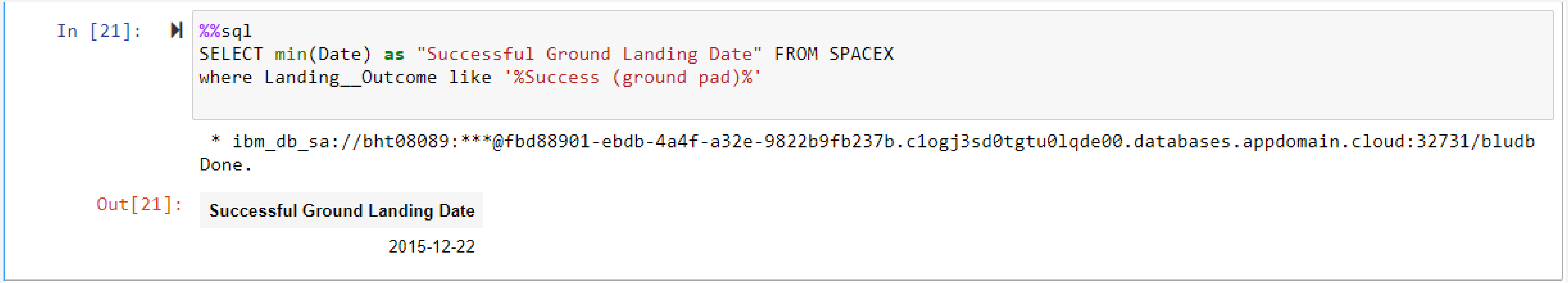
- By performing an Average function on the payload mass and filtering the data to only include booster version F9 v 1.1 by finding the like value string the average was calculated.
- See reference screenshot below



```
In [19]:  %%sql
          SELECT avg(payload_mass__kg_) as "Average Payload Mass For F9 v1.1" FROM SPACEX
          where booster_version like '%F9 v1.1%'

           * ibm_db_sa://bht08089:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
          Done.

Out[19]:  Average Payload Mass For F9 v1.1
                                      2534
```

First Successful Ground Landing Date

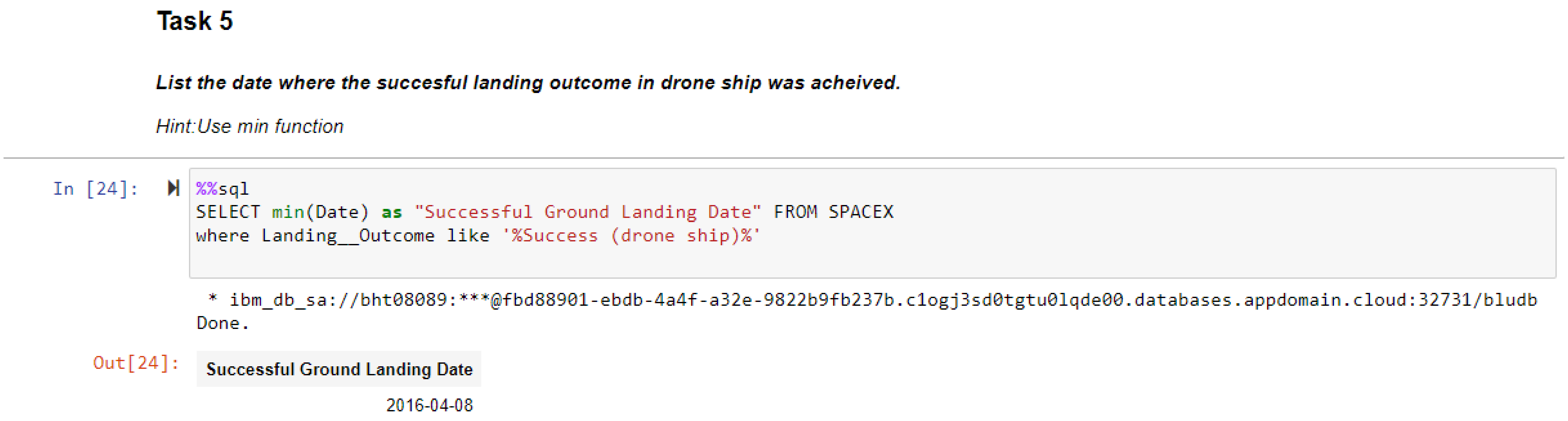
- By selecting the minimum date where a like function was passed to the landing outcome searching for the "Success (ground pad)" string the earliest date was obtained.
- PLEASE NOTE THE NOTEBOOK QUESTION ASKED FOR DRONE SHIP LANDING DATE (see next slide)
- See reference screenshot below



```
In [21]:  %%sql
          SELECT min(Date) as "Successful Ground Landing Date" FROM SPACEX
          where Landing__Outcome like '%Success (ground pad)%'

           * ibm_db_sa://bht08089:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
          Done.
Out[21]:  Successful Ground Landing Date
                                2015-12-22
```

First Successful Drone Ship Date

- By selecting the minimum date where a like function was passed to the landing outcome searching for the “Success (drone ship)” string the earliest date was obtained.
- See reference screenshot below



Task 5

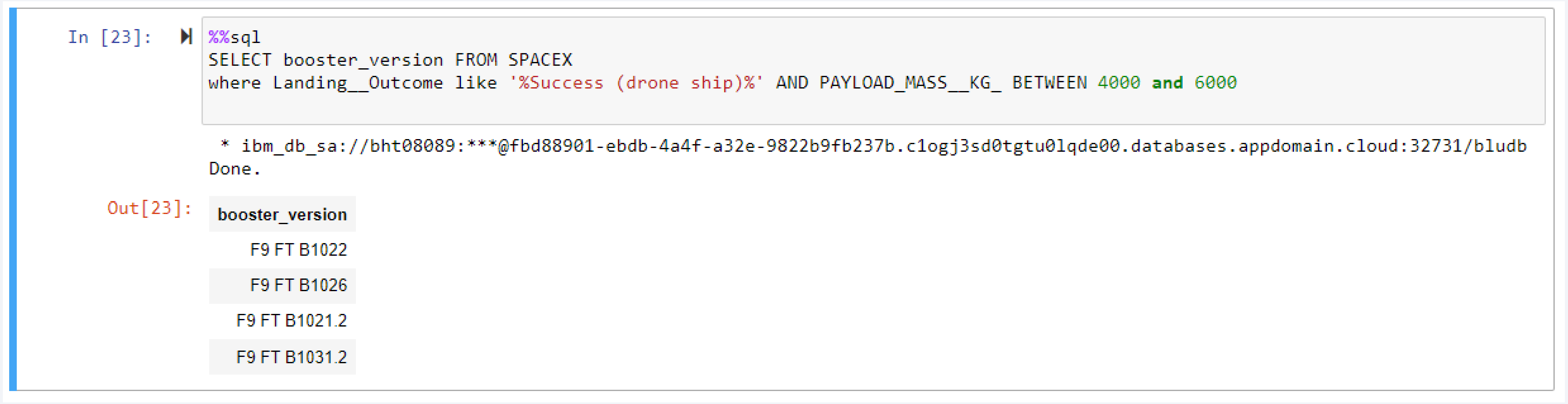
List the date where the succesful landing outcome in drone ship was acheived.

Hint:Use min function

```
In [24]:  %%sql
          SELECT min(Date) as "Successful Ground Landing Date" FROM SPACEX
          where Landing__Outcome like '%Success (drone ship)%'

 * ibm_db_sa://bht08089:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu01qde00.databases.appdomain.cloud:32731/bludb
Done.

Out[24]:  Successful Ground Landing Date
                            2016-04-08
```

Successful Drone Ship Landing with Payload between 4000 and 6000

- The booster version was selected form the table and was filtered by 2 and statements:
    - One where the landing outcome was success on drone ship (note the question asked for ground pad – see next slide)
    - One where the payload was BETWEEN 4000 AND 6000
- See reference screenshot below

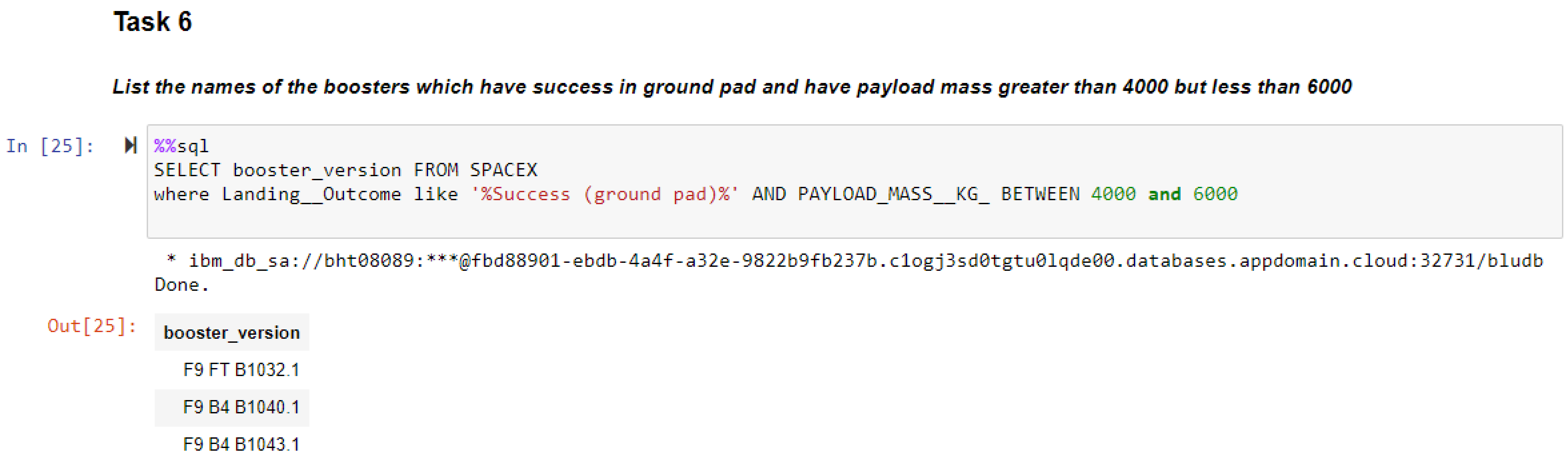


```
In [23]:  %%sql
          SELECT booster_version FROM SPACEX
          where Landing__Outcome like '%Success (drone ship)%' AND PAYLOAD_MASS__KG_ BETWEEN 4000 and 6000

           * ibm_db_sa://bht08089:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu01qde00.databases.appdomain.cloud:32731/bludb
          Done.

Out[23]:
```

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

Successful Ground Pad Landing with Payload between 4000 and 6000

- The booster version was selected form the table and was filtered by 2 and statements:
  - One where the landing outcome was success on ground pad
  - One where the payload was BETWEEN 4000 AND 6000
- See reference screenshot below



Task 6

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

```
In [25]:  %%sql
          SELECT booster_version FROM SPACEX
          where Landing__Outcome like '%Success (ground pad)%' AND PAYLOAD_MASS__KG_ BETWEEN 4000 and 6000
```

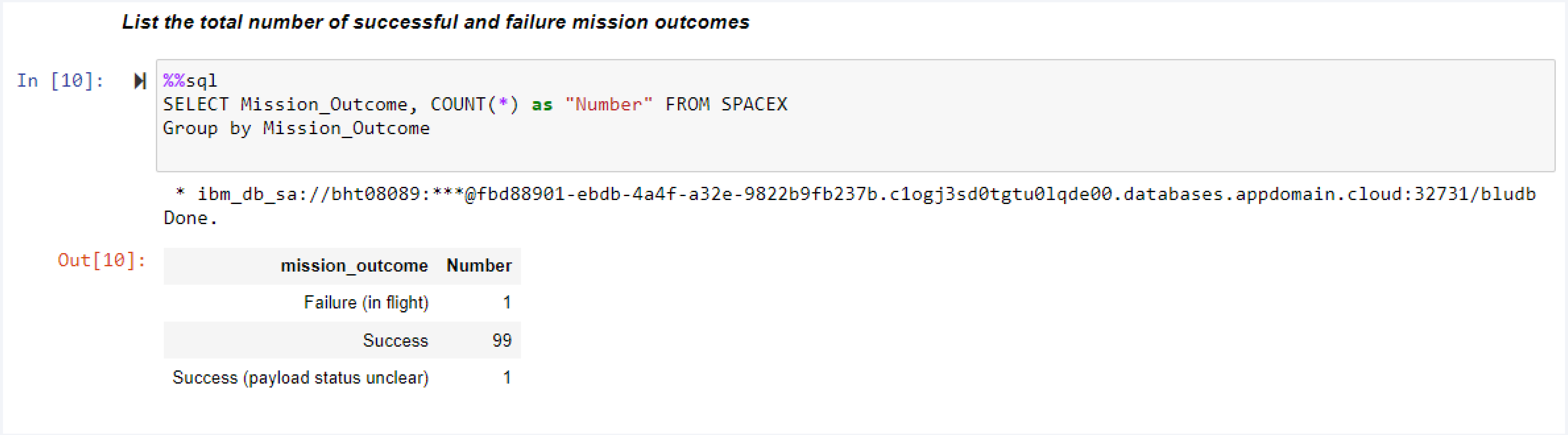
```
 * ibm_db_sa://bht08089:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Out[25]:

booster_version
F9 FT B1032.1
F9 B4 B1040.1
F9 B4 B1043.1

Total Number of Successful and Failure Mission Outcomes

- The mission outcome column was selected and a new variable of count was created. It was names number for the output table. Then the group by function was passed on the mission outcome variable to count the number of successful and failure missions.
- See reference screenshot below



List the total number of successful and failure mission outcomes

```
In [10]:  %%sql
          SELECT Mission_Outcome, COUNT(*) as "Number" FROM SPACEX
          Group by Mission_Outcome

 * ibm_db_sa://bht08089:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

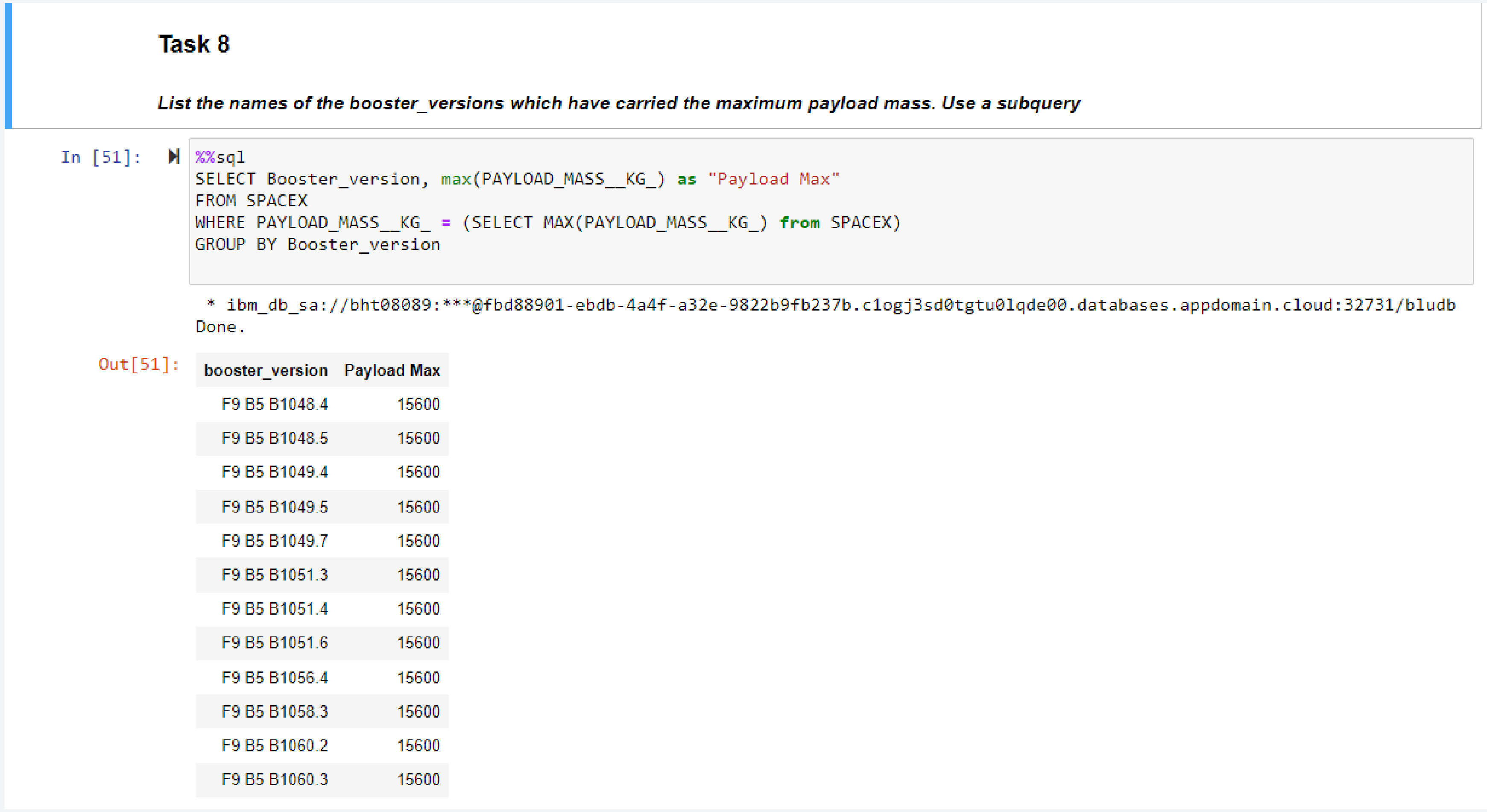
Out[10]:

mission_outcome	Number
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- The booster version and the max of Payload are selected and the only values which are grouped are the ones which have a max payload equal to the highest number in the payload column.

- See reference screenshot below



Task 8

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

In [51]:

```
%%sql
SELECT Booster_version, max(PAYLOAD_MASS__KG_) as "Payload Max"
FROM SPACEX
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) from SPACEX)
GROUP BY Booster_version
```

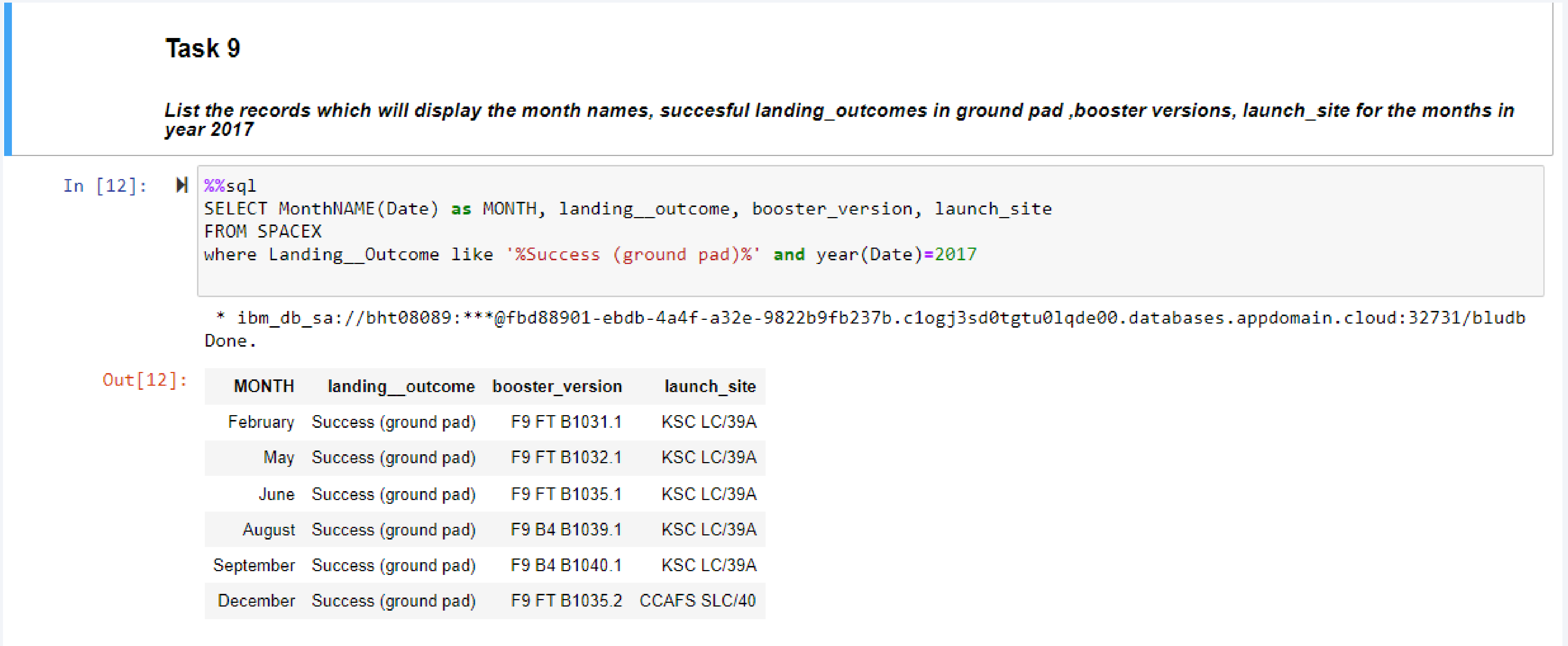
```
 * ibm_db_sa://bht08089:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Out[51]:

booster_version	Payload Max
F9 B5 B1048.4	15600
F9 B5 B1048.5	15600
F9 B5 B1049.4	15600
F9 B5 B1049.5	15600
F9 B5 B1049.7	15600
F9 B5 B1051.3	15600
F9 B5 B1051.4	15600
F9 B5 B1051.6	15600
F9 B5 B1056.4	15600
F9 B5 B1058.3	15600
F9 B5 B1060.2	15600
F9 B5 B1060.3	15600

2017 Launch Records

- The MONTHNAME was selected from the date and labelled as month and selected along with the landing outcomes, booster version, and launch site. Then the landing outcome was filtered through a like statement searching for the string "Success ground pad" AND searching the date values for the year 2017. The filtered table is in the screenshot below with the code



Task 9

List the records which will display the month names, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017

In [12]:
```
%%sql
SELECT MonthNAME(Date) as MONTH, landing__outcome, booster_version, launch_site
FROM SPACEX
where Landing__Outcome like '%Success (ground pad)%' and year(Date)=2017
```

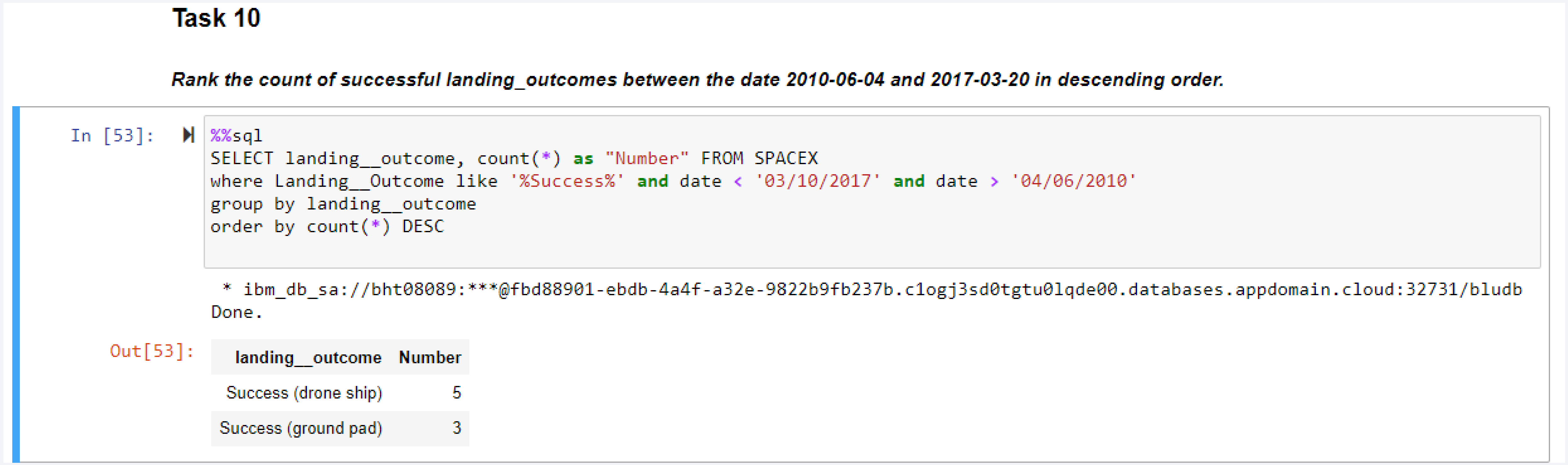
```
 * ibm_db_sa://bht08089:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Out[12]:

MONTH	landing__outcome	booster_version	launch_site
February	Success (ground pad)	F9 FT B1031.1	KSC LC/39A
May	Success (ground pad)	F9 FT B1032.1	KSC LC/39A
June	Success (ground pad)	F9 FT B1035.1	KSC LC/39A
August	Success (ground pad)	F9 B4 B1039.1	KSC LC/39A
September	Success (ground pad)	F9 B4 B1040.1	KSC LC/39A
December	Success (ground pad)	F9 FT B1035.2	CCAFS SLC/40

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

- The landing outcome was selected as the COUNT variable which was labelled number. The landing outcomes was parsed to search for the word "SUCCESS" as well as searching the date value for instances between 2010-06-04 and 2017-03-20



Task 10

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

In [53]:

```
%%sql
SELECT landing__outcome, count(*) as "Number" FROM SPACEX
where Landing__Outcome like '%Success%' and date < '03/10/2017' and date > '04/06/2010'
group by landing__outcome
order by count(*) DESC
```

```
 * ibm_db_sa://bht08089:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Out[53]:

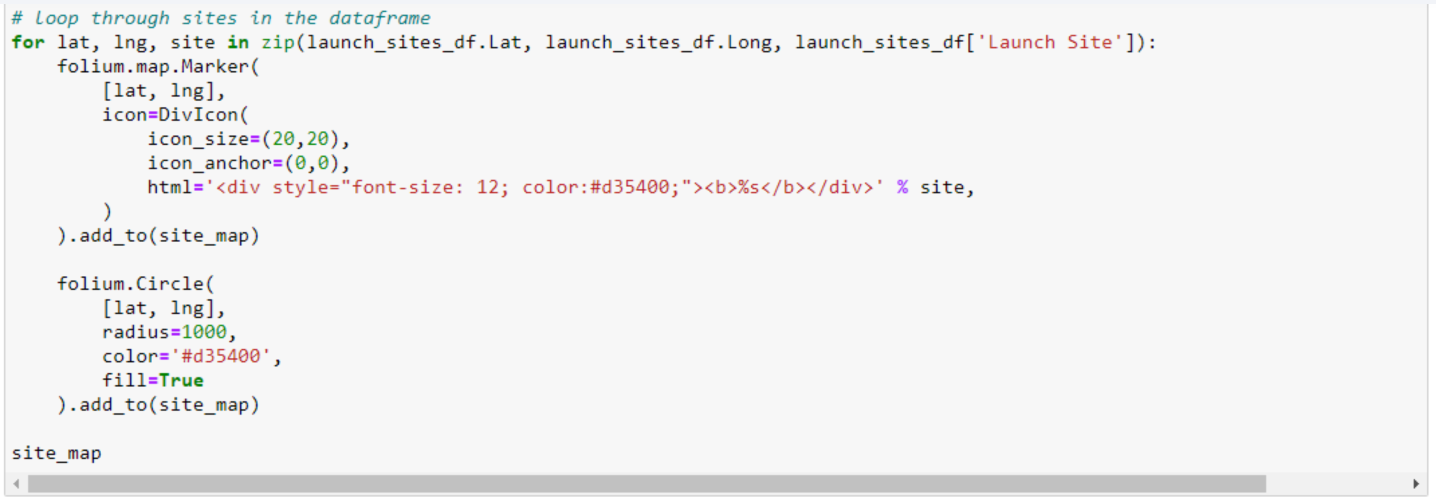
landing__outcome	Number
Success (drone ship)	5
Success (ground pad)	3

Section 4

Launch Sites Proximities Analysis

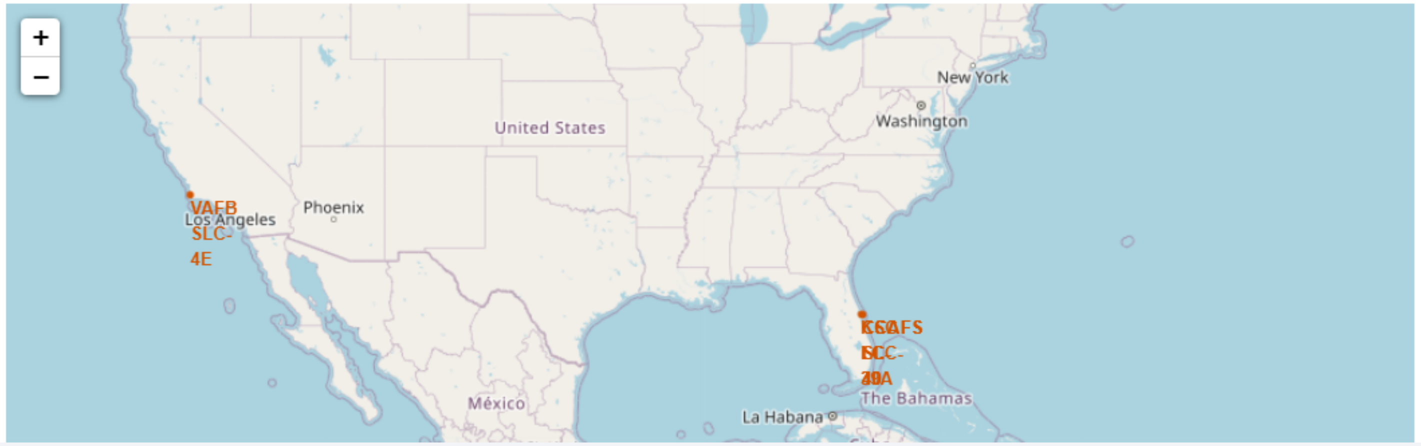
Launch Site Locations

- Locations of the launch sites are all in the Southern United States
- Locations are on the East and West Coast, close to the oceans.
- Multiple location site names are close to each other on each respective coast.



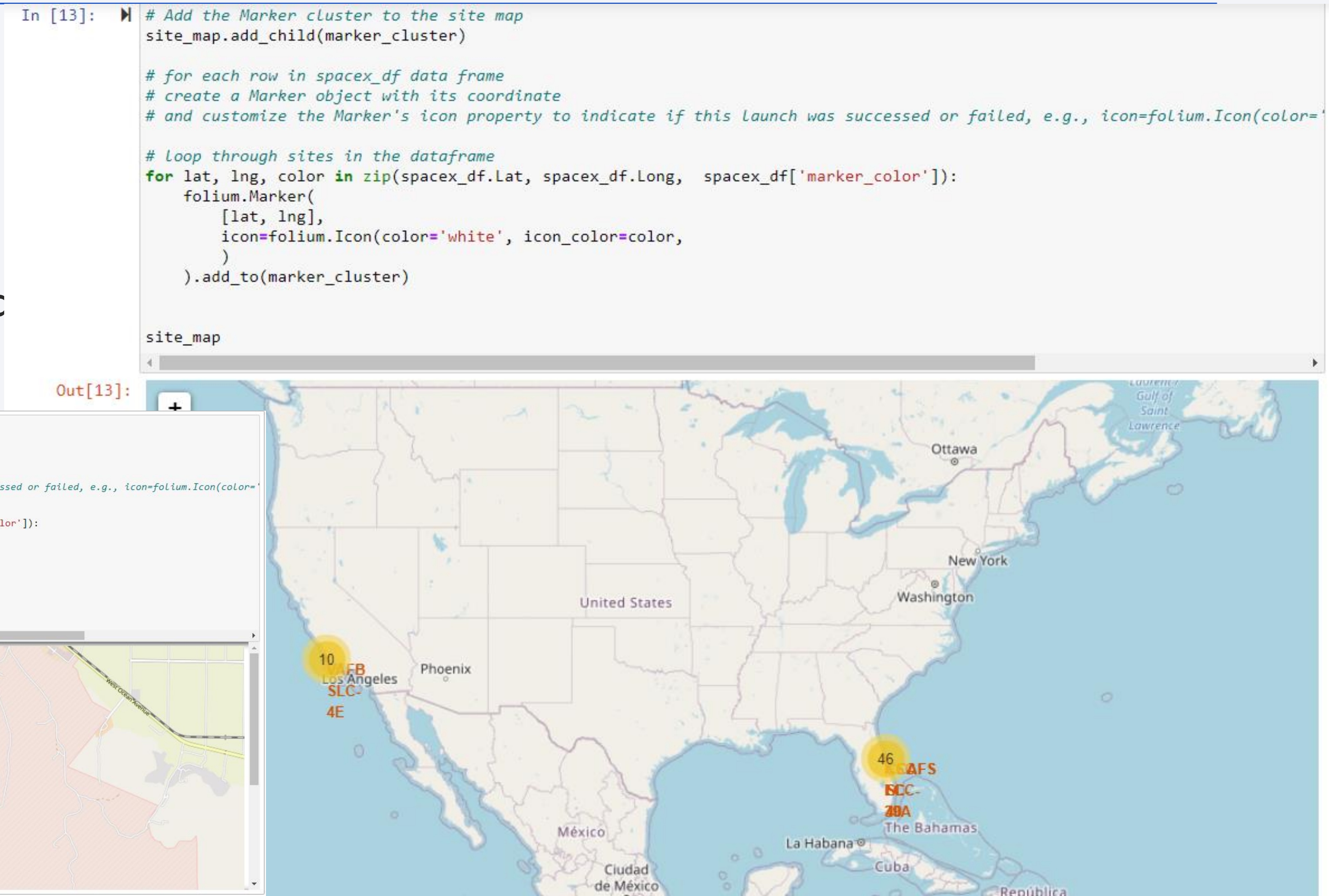
```python
# loop through sites in the dataframe
for lat, lng, site in zip(launch_sites_df.Lat, launch_sites_df.Long, launch_sites_df['Launch Site']):
    folium.map.Marker(
        [lat, lng],
        icon=DivIcon(
            icon_size=(20,20),
            icon_anchor=(0,0),
            html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % site,
        )
    ).add_to(site_map)

    folium.Circle(
        [lat, lng],
        radius=1000,
        color='#d35400',
        fill=True
    ).add_to(site_map)

site_map
```

Out[8]:

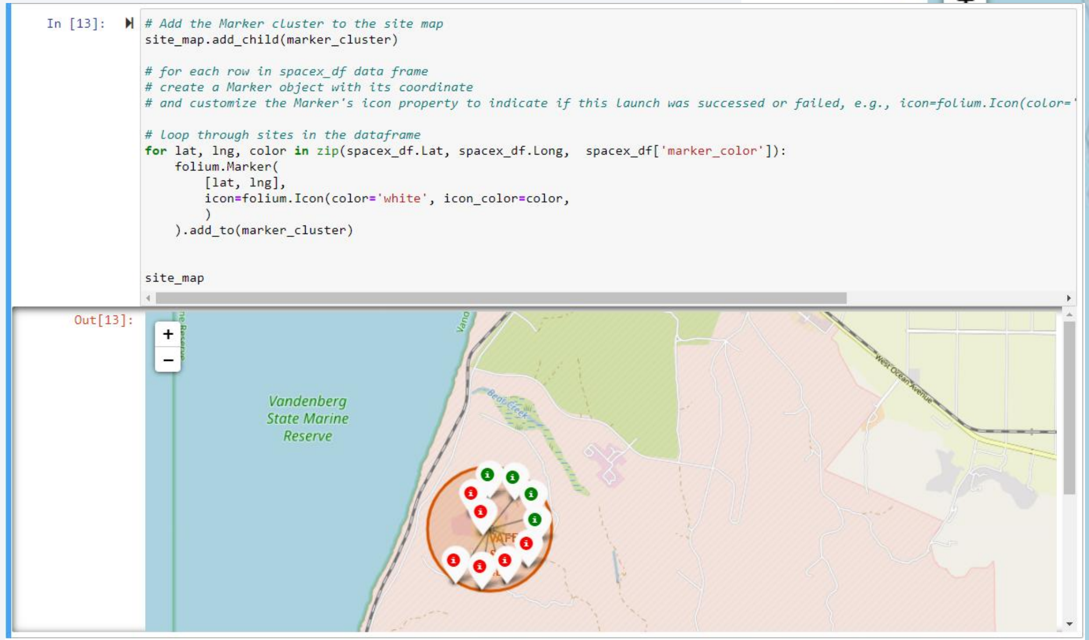


Successful Launch Outcomes

- Explain the important elements and findings on the screenshot
- Each launch site has multiple successful an unsuccessful landings.



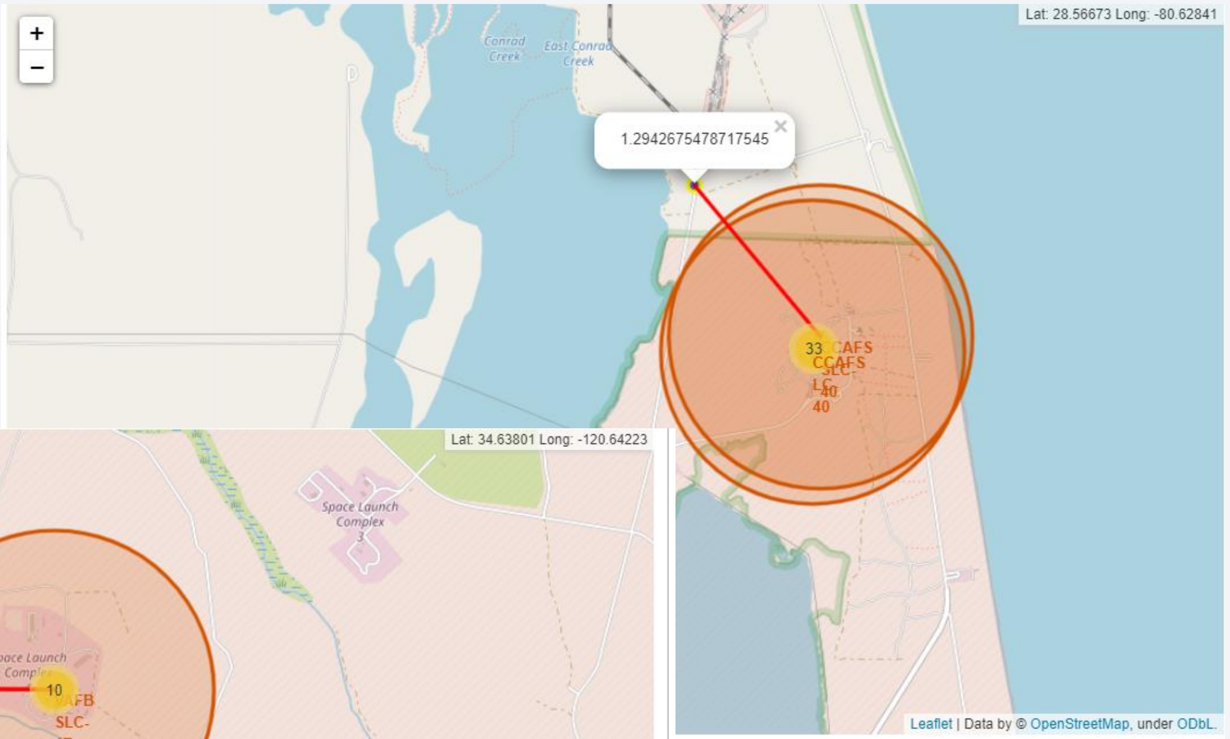
```
In [13]:  # Add the Marker cluster to the site map
          site_map.add_child(marker_cluster)

          # for each row in spacex_df data frame
          # create a Marker object with its coordinate
          # and customize the Marker's icon property to indicate if this launch was successed or failed, e.g., icon=folium.Icon(color=

          # loop through sites in the dataframe
          for lat, lng, color in zip(spacex_df.Lat, spacex_df.Long,  spacex_df['marker_color']):
              folium.Marker(
                  [lat, lng],
                  icon=folium.Icon(color='white', icon_color=color,
                  )
              ).add_to(marker_cluster)


          site_map
Out[13]:
```

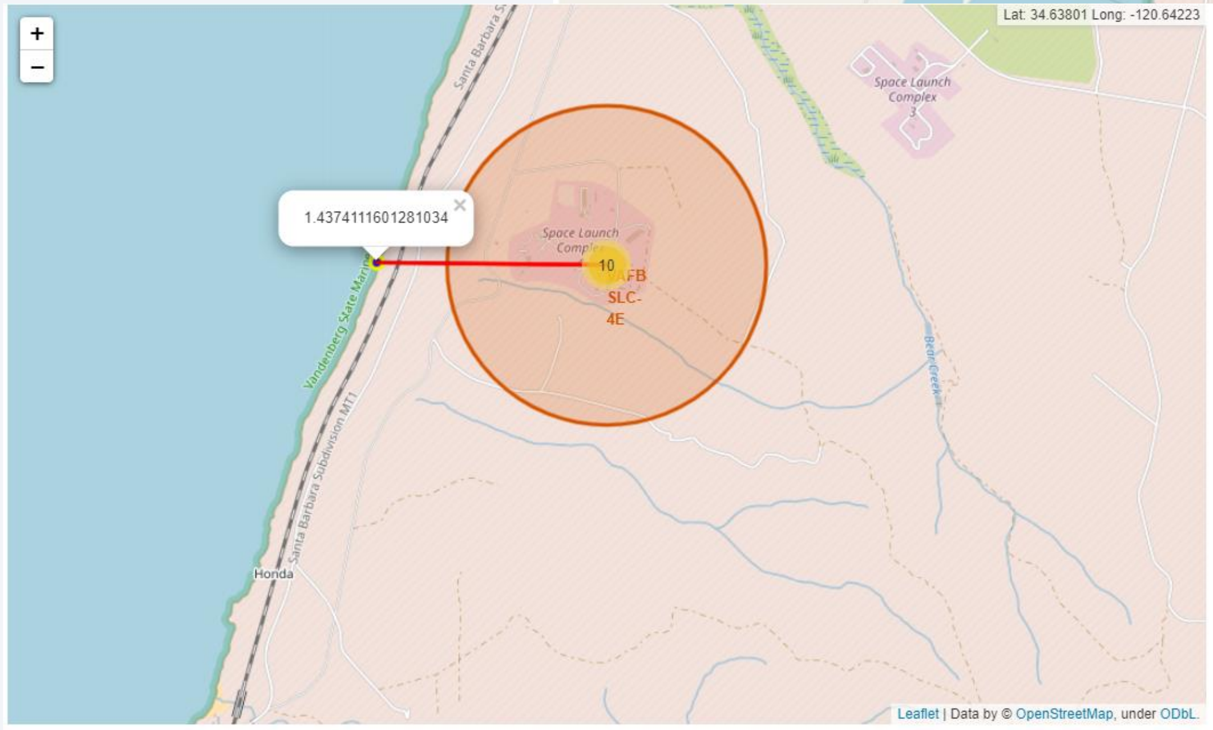


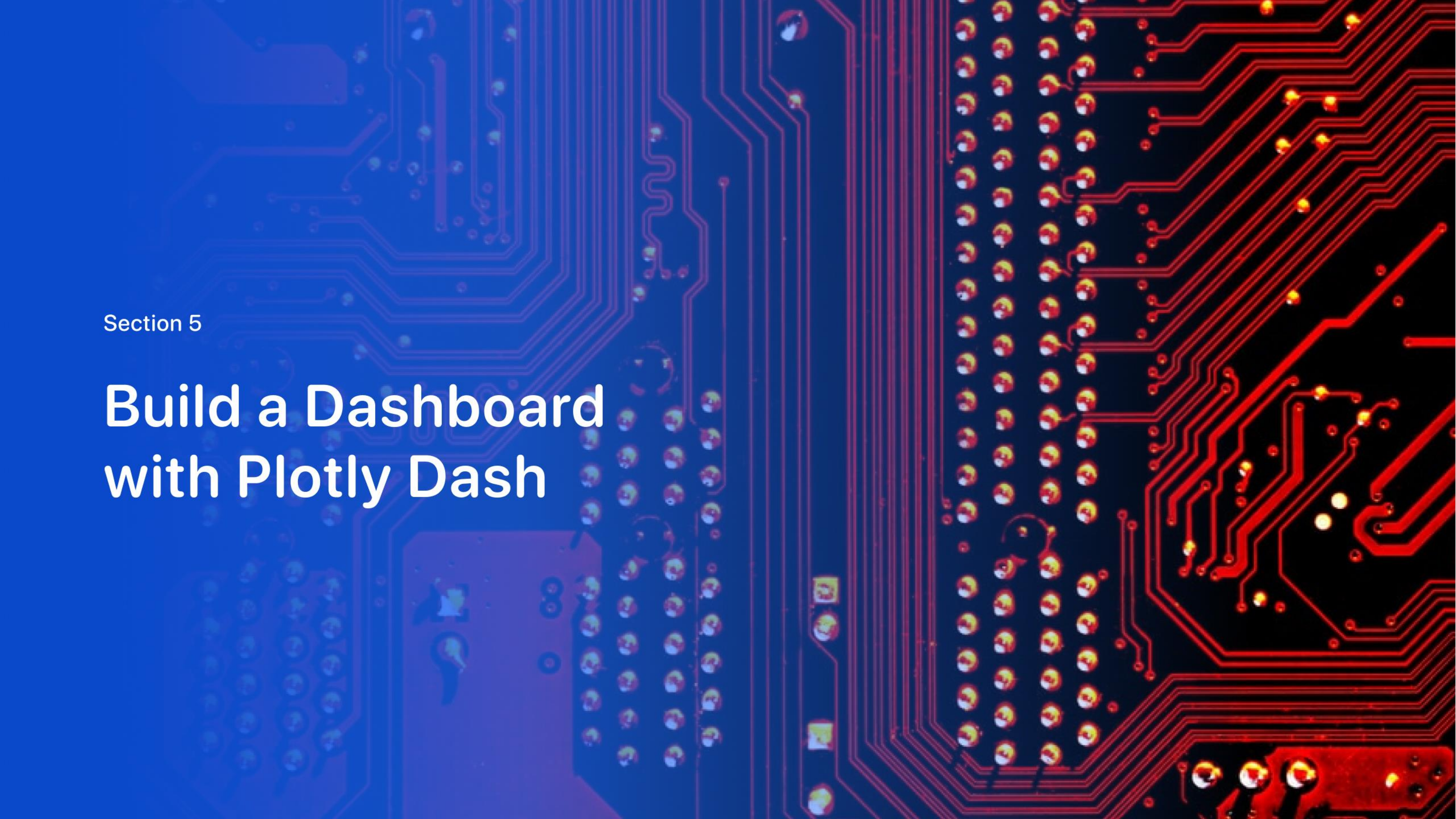
```
In [13]:  # Add the Marker cluster to the site map
          site_map.add_child(marker_cluster)

          # for each row in spacex_df data frame
          # create a Marker object with its coordinate
          # and customize the Marker's icon property to indicate if this launch was successed or failed, e.g., icon=folium.Icon(color=

          # loop through sites in the dataframe
          for lat, lng, color in zip(spacex_df.Lat, spacex_df.Long,  spacex_df['marker_color']):
              folium.Marker(
                  [lat, lng],
                  icon=folium.Icon(color='white', icon_color=color,
                  )
              ).add_to(marker_cluster)


          site_map
Out[13]:
```

Launch Site Locations to Areas of Interest

- The locations of the launch sites are relatively close to coastlines and railroads and some highways.
  - This is expected as large components for rockets need to be transported to the site.
  - Distances to coastlines is close as it is suspected by this author that any disasters would result in failures on the water


Lat: 28.56673 Long: -80.62841
1.2942675478717545
33
CCAFS SLC-40
CCAFS LC-40
Conrad Creek
East Conrad Creek
Leaflet | Data by © OpenStreetMap, under ODbL.





Section 5

Build a Dashboard with Plotly Dash

Interactive Dashboard 1

- Did not complete
- Unfortunately this was too difficult and I just didn’t get it

Interactive Dashboard 2

- Did not complete
- Unfortunately this was too difficult and I just didn’t get it

Interactive Dashboard 3

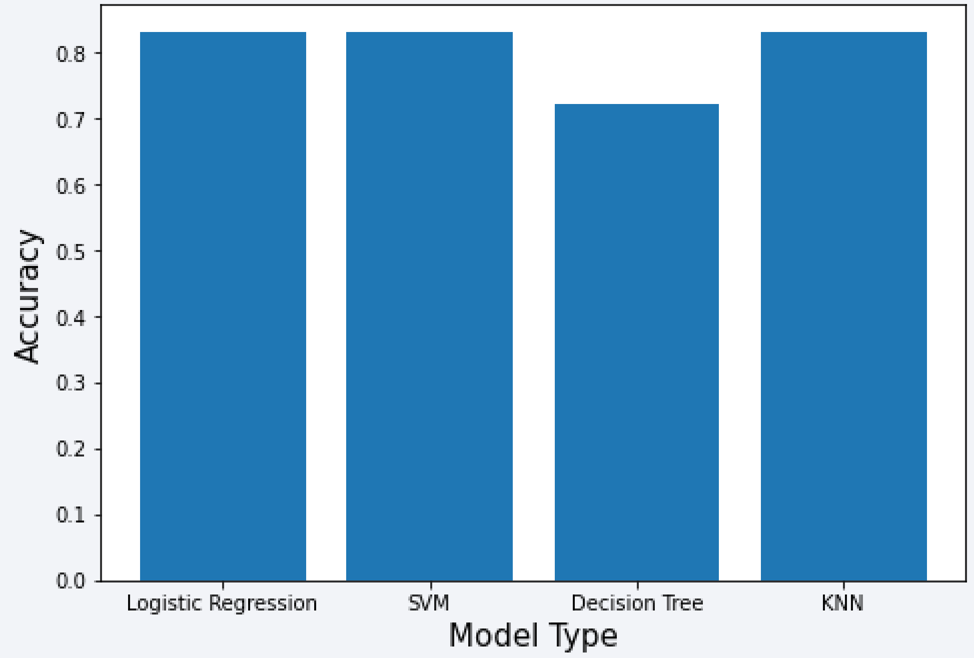
- Did not complete
- Unfortunately this was too difficult and I just didn’t get it

Section 6

Predictive Analysis (Classification)

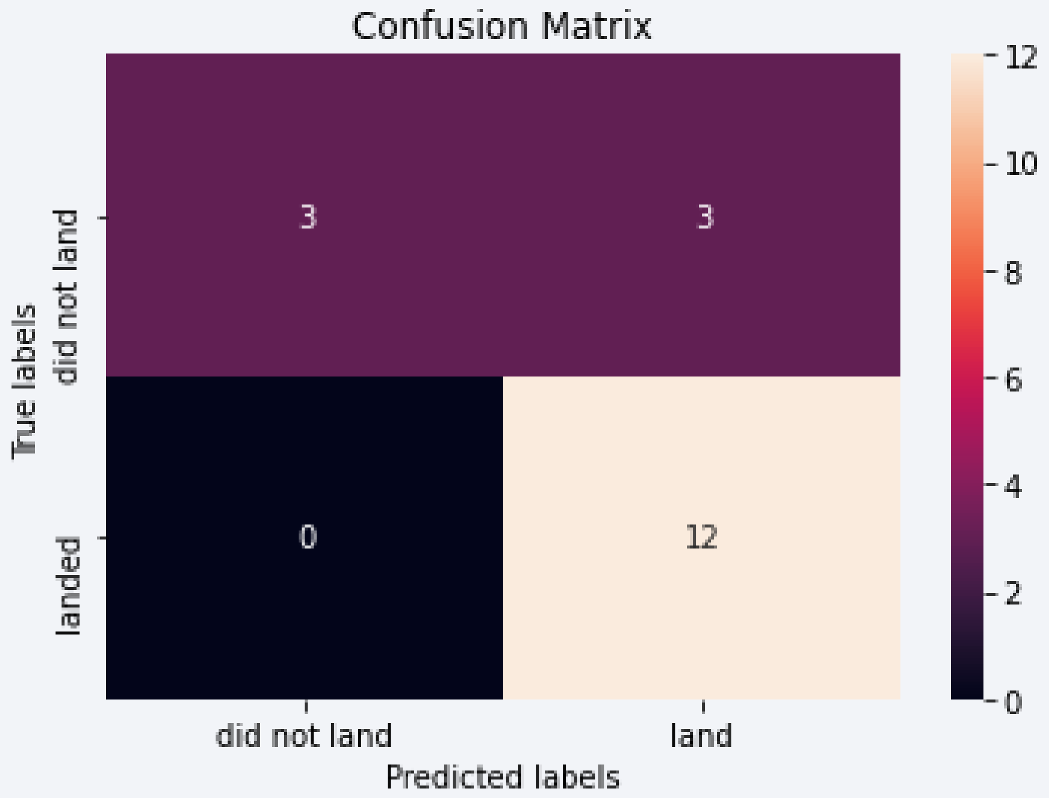
Classification Accuracy

- Model was developed using 80% of data in training.
- LR, SVM, and KNN models were able to provide an accuracy on the test data equal to 83.33%.
- The Logistic Regression and Decision Tree models had an accuracy of 72.22%
- Accuracy on the **train set** was also similar:
  - LR was 84.64%
  - KNN was 84.82%
  - SVM was 84.82%



Confusion Matrix

- Confusion matrix for KNN, LR, SVM depicts the number of predictions compared to the actual outcomes.
- Data shows 3 false positives (predict land when it did not)
- Model predicts no false negatives (predict no land when it did land).



Conclusions

- SVM, Logistic Regression, and KNN all performed the same in terms of model.
- Model results changed when the python version changed.
  - Even when the random seed was set to 2 for each.
- 3 false positives were predicted.
- No false negatives in the 3 models.

Thank you!