title: SpaceX Falcon web scraping

author: Wittawat Muangkot

date: 2022-03-10

Description

this paper will collection spaceX launches data from wikipedia by using web scraping and do some basic data wrangling following step below;

- Import essential package
- Request the Falcon9 Launch Wiki page from its URL
- Extract all column/variable names from the HTML table header
- Create a dataframe by parsing the launch HTML tables
- data wrangling and basic exploratory
- data visualization

More specifically, the launch records are stored in a HTML table shown in wikipedia page as below:

In [204…

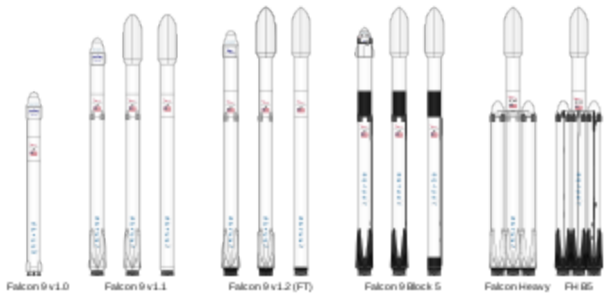
```
from IPython.display import IFrame

IFrame(src="https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches?utm_r
```

Out[204…

WIKIPEDIA

List of Falcon 9 and Falcon Heavy launches

Since June 2010, rockets from the Falcon 9 family have been launched 151 times, with 149 full mission successes, one partial failure and one total loss of the spacecraft. In addition, one rocket and its payload were destroyed on the launch pad during the fueling process before a static fire test was set to occur.



Left to right: Falcon 9 v1.0, v1.1, v1.2 "Full Thrust", Falcon 9 Block 5, Falcon Heavy, and Falcon Heavy Block 5.

Designed and operated by private manufacturer SpaceX, the Falcon 9 rocket family includes the retired versions Falcon 9 v1.0, v1.1, and v1.2 "Full Thrust" Block 1 to 4, along with the currently active Block 5 evolution. Falcon Heavy is a heavy-lift derivative of Falcon 9, combining a strengthened central core with two Falcon 9 first stages as the side boosters.[1]

The Falcon design features reusable first-stage boosters, which land either on a ground pad near the launch site or on a drone ship at sea.[2] In December 2015, Falcon 9 became the first rocket to land propulsively after delivering a payload into orbit.[3] This reusability has resulted in significantly reduced launch costs.[4] Falcon family core boosters have successfully landed 114 times in 125 attempts. A total of 29 boosters have flown multiple missions, with a record of

Import essential package

In [205…

```
import requests
from bs4 import BeautifulSoup
import re
import unicodedata
import pandas as pd
import datetime as dt
import warnings
import seaborn as sns
warnings.filterwarnings('ignore')
```

Request the Falcon9 Launch Wiki page from its URL

In [205…

```
url = "https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches?utm_mediur
```

Next, request the HTML page from the above URL and get a response object

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

In [205…

```
respons=requests.get(url)
```

```
text=respons.text
```

Create a BeautifulSoup object from the HTML response

In [205…
```
soup=BeautifulSoup(text,'html5lib')
```

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

In [205…
```
soup.title
```

Out[205…
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

Starting from the third table is the target table contains the actual launch records.

In [205…
```
table=soup.find_all('table')
table[2]
```

Out[205…
```
<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
<tbody><tr>
<th scope="col">Flight No.
</th>
<th scope="col">Date and<br/>time (<a href="/wiki/Coordinated_Universal_Time" title="Coord
inated Universal Time">UTC</a>)
</th>
<th scope="col"><a href="/wiki/List_of_Falcon_9_first-stage_boosters" title="List of Falco
n 9 first-stage boosters">Version,<br/>Booster</a> <sup class="reference" id="cite_ref-boo
ster_11-0"><a href="#cite_note-booster-11">[b]</a></sup>
</th>
<th scope="col">Launch site
</th>
<th scope="col">Payload<sup class="reference" id="cite_ref-Dragon_12-0"><a href="#cite_not
e-Dragon-12">[c]</a></sup>
</th>
<th scope="col">Payload mass
</th>
<th scope="col">Orbit
</th>
<th scope="col">Customer
</th>
<th scope="col">Launch<br/>outcome
</th>
<th scope="col"><a href="/wiki/Falcon_9_first-stage_landing_tests" title="Falcon 9 first-s
tage landing tests">Booster<br/>landing</a>
</th></tr>
<tr>
<th rowspan="2" scope="row" style="text-align:center;">1
</th>
<td>4 June 2010,<br/>18:45
</td>
<td><a href="/wiki/Falcon_9_v1.0" title="Falcon 9 v1.0">F9 v1.0</a><sup class="reference"
id="cite_ref-MuskMay2012_13-0"><a href="#cite_note-MuskMay2012-13">[7]</a></sup><br/>B0003
<sup class="reference" id="cite_ref-block_numbers_14-0"><a href="#cite_note-block_numbers-
14">[8]</a></sup>
</td>
<td><a href="/wiki/Cape_Canaveral_Space_Force_Station" title="Cape Canaveral Space Force S
tation">CCAFS</a>,<br/><a href="/wiki/Cape_Canaveral_Space_Launch_Complex_40" title="Cape
Canaveral Space Launch Complex 40">SLC-40</a>
</td>
<td><a href="/wiki/Dragon_Spacecraft_Qualification_Unit" title="Dragon Spacecraft Qualific
ation Unit">Dragon Spacecraft Qualification Unit</a>
</td>
<td class="table-na" data-sort-value="" style="background: #ececec; color: #2C2C2C; vertic
```

al-align: middle; text-align: center;">No payload (excl. Dragon Mass)
</td>
<td>LEO
</td>
<td>SpaceX
</td>
<td class="table-success" style="background: #9EFF9E; vertical-align: middle; text-align: center;">Success
</td>
<td class="table-failure" style="background: #FFC7C7; vertical-align: middle; text-align: center;">Failure^{[9]}^{[10]}
<small>(parachute)</small>
</td></tr>
<tr>
<td colspan="9">First flight of Falcon 9 v1.0.^{[11]} Used a boilerplate version of Dragon capsule which was not designed to separate from the second stage.<small>(more details below)</small> Attempted to recover the first stage by parachuting it into the ocean, but it burned up on reentry, before the parachutes even got to deploy.^{[12]}
</td></tr>
<tr>
<th rowspan="2" scope="row" style="text-align:center;">2
</th>
<td>8 December 2010,
15:43^{[13]}
</td>
<td>F9 v1.0^{[7]}
B0004 ^{[8]}
</td>
<td>CCAFS,
SLC-40
</td>
<td>Dragon demo flight C1
(Dragon C101)
</td>
<td class="table-na" data-sort-value="" style="background: #ececec; color: #2C2C2C; vertical-align: middle; text-align: center;">Classified (excl. Dragon Mass)
</td>
<td>LEO (ISS)
</td>
<td>
NASA (COTS)
NRO

</td>
<td class="table-success" style="background: #9EFF9E; vertical-align: middle; text-align: center;">Success^{[9]}
</td>
<td class="table-failure" style="background: #FFC7C7; vertical-align: middle; text-align: center;">Failure^{[9]}^{[14]}
<small>(parachute)</small>
</td></tr>
<tr>
<td colspan="9">Maiden flight of SpaceX's <a class="mw-redirect" href="/wiki/Dragon_capsul

e" title="Dragon capsule">Dragon capsule, consisting of over 3 hours of testing thruster maneuvering and then reentry.^{[15]} Attempted to recover the first stage by parachuting it into the ocean, but it disintegrated upon reentry, again before the parachutes were deployed.^{[12]} <small>(more details below)</small> It also included two CubeSats,^{[16]} and a wheel of Brouère cheese. Before the launch, SpaceX discovered that there was a crack in the nozzle of the 2nd stage's Merlin vacuum engine. So Elon just had them cut off the end of the nozzle with a pair of shears and launched the rocket a few days later. After SpaceX had trimmed the nozzle, NASA was notified of the change and they agreed to it.^{[17]}
</td></tr>
<tr>
<th rowspan="2" scope="row" style="text-align:center;">3
</th>
<td>22 May 2012,
07:44^{[18]}
</td>
<td>F9 v1.0^{[7]}
B0005 ^{[8]}
</td>
<td>CCAFS,
SLC-40
</td>
<td>Dragon demo flight C2+^{[19]}
(Dragon C102)
</td>
<td>525 kg (1,157 lb)^{[20]} (excl. Dragon mass)
</td>
<td>LEO (ISS)
</td>
<td>NASA (COTS)
</td>
<td class="table-success" style="background: #9EFF9E; vertical-align: middle; text-align: center;">Success^{[21]}
</td>
<td class="table-noAttempt" style="background: #EEE; vertical-align: middle; white-space: nowrap; text-align: center;">No attempt
</td></tr>
<tr>
<td colspan="9">The Dragon spacecraft demonstrated a series of tests before it was allowed to approach the International Space Station. Two days later, it became the first commercial spacecraft to board the ISS.^{[18]} <small>(more details below)</small>
</td></tr>
<tr>
<th rowspan="3" scope="row" style="text-align:center;">4
</th>
<td rowspan="2">8 October 2012,
00:35^{[22]}
</td>

</td>
<td rowspan="2">F9 v1.0<sup class
="reference" id="cite_ref-MuskMay2012_13-3">[7]</s
up>
B0006<sup class="reference" id="cite_ref-block_numbers_14-3"><a href="#cite_note-b
lock_numbers-14">[8]</sup>
</td>
<td rowspan="2"><a href="/wiki/Cape_Canaveral_Space_Force_Station" title="Cape Canaveral S
pace Force Station">CCAFS,
<a href="/wiki/Cape_Canaveral_Space_Launch_Complex_40"
title="Cape Canaveral Space Launch Complex 40">SLC-40
</td>
<td>SpaceX CRS-1<sup class="referenc
e" id="cite_ref-sxManifest20120925_29-0">[23]</
a></sup>
(Dragon C103)
</td>
<td>4,700 kg (10,400 lb) (excl. Dragon mass)
</td>
<td>LEO (<a href="/wiki/Intern
ational_Space_Station" title="International Space Station">ISS)
</td>
<td>NASA (<a href="/wiki/Commercial_Resupply_Service
s" title="Commercial Resupply Services">CRS)
</td>
<td class="table-success" style="background: #9EFF9E; vertical-align: middle; text-align:
center;">Success
</td>
<td rowspan="2" style="background:#ececec; text-align:center;">No att
empt
</td></tr>
<tr>
<td>Orbcomm-OG2<sup cl
ass="reference" id="cite_ref-Orbcomm_30-0">[24]</sup>
</td>
<td>172 kg (379 lb)<sup class="reference" id="cite_ref-gunter-og2_31-0"><a href="#cite_not
e-gunter-og2-31">[25]</sup>
</td>
<td>LEO
</td>
<td>Orbcomm
</td>
<td class="table-partial" style="background: #FE9; vertical-align: middle; text-align: cen
ter;">Partial failure<sup class="reference" id="cite_ref-nyt-20121030_32-0"><a href="#cite
_note-nyt-20121030-32">[26]</sup>
</td></tr>
<tr>
<td colspan="9">CRS-1 was successful, but the <a href="/wiki/Secondary_payload" title="Sec
ondary payload">secondary payload was inserted into an abnormally low orbit and subseq
uently lost. This was due to one of the nine <a href="/wiki/SpaceX_Merlin" title="SpaceX M
erlin">Merlin engines shutting down during the launch, and NASA declining a second rei
gnition, as per <a href="/wiki/International_Space_Station" title="International Space Sta
tion">ISS visiting vehicle safety rules, the primary payload owner is contractually al
lowed to decline a second reignition. NASA stated that this was because SpaceX could not g
uarantee a high enough likelihood of the second stage completing the second burn successfu
lly which was required to avoid any risk of secondary payload's collision with the ISS.<su
p class="reference" id="cite_ref-OrbcommTotalLoss_33-0"><a href="#cite_note-OrbcommTotalLo
ss-33">[27]</sup><sup class="reference" id="cite_ref-sn20121011_34-0"><a href="#cite_n
ote-sn20121011-34">[28]</sup><sup class="reference" id="cite_ref-35"><a href="#cite_no
te-35">[29]</sup>
</td></tr>
<tr>
<th rowspan="2" scope="row" style="text-align:center;">5
</th>
<td>1 March 2013,
15:10
</td>
<td>F9 v1.0<sup class="reference"
id="cite_ref-MuskMay2012_13-4">[7]</sup>
B0007
<sup class="reference" id="cite_ref-block_numbers_14-4"><a href="#cite_note-block_numbers-

14">[8]</sup>
</td>
<td><a href="/wiki/Cape_Canaveral_Space_Force_Station" title="Cape Canaveral Space Force S
tation">CCAFS,
<a href="/wiki/Cape_Canaveral_Space_Launch_Complex_40" title="Cape
Canaveral Space Launch Complex 40">SLC-40
</td>
<td>SpaceX CRS-2<sup class="referenc
e" id="cite_ref-sxManifest20120925_29-1">[23]</
a></sup>
(Dragon C104)
</td>
<td>4,877 kg (10,752 lb) (excl. Dragon mass)
</td>
<td>LEO (<a class="mw-redirec
t" href="/wiki/ISS" title="ISS">ISS)
</td>
<td>NASA (<a href="/wiki/Commercial_Resupply_Service
s" title="Commercial Resupply Services">CRS)
</td>
<td class="table-success" style="background: #9EFF9E; vertical-align: middle; text-align:
center;">Success
</td>
<td class="table-noAttempt" style="background: #EEE; vertical-align: middle; white-space:
nowrap; text-align: center;">No attempt
</td></tr>
<tr>
<td colspan="9">Last launch of the original Falcon 9 v1.0 <a href="/wiki/Launch_vehicle" t
itle="Launch vehicle">launch vehicle, first use of the unpressurized trunk section of
Dragon.<sup class="reference" id="cite_ref-sxf9_20110321_36-0"><a href="#cite_note-sxf9_20
110321-36">[30]</sup>
</td></tr>
<tr>
<th rowspan="2" scope="row" style="text-align:center;">6
</th>
<td>29 September 2013,
16:00<sup class="reference" id="cite_ref-pa20130930_37-0"><a hr
ef="#cite_note-pa20130930-37">[31]</sup>
</td>
<td>F9 v1.1<sup class="reference"
id="cite_ref-MuskMay2012_13-5">[7]</sup>
B1003
<sup class="reference" id="cite_ref-block_numbers_14-5"><a href="#cite_note-block_numbers-
14">[8]</sup>
</td>
<td><a class="mw-redirect" href="/wiki/Vandenberg_Air_Force_Base" title="Vandenberg Air Fo
rce Base">VAFB,
<a href="/wiki/Vandenberg_Space_Launch_Complex_4" title="Vandenber
g Space Launch Complex 4">SLC-4E
</td>
<td>CASSIOPE<sup class="reference" id="cite_
ref-sxManifest20120925_29-2">[23]</sup><sup
class="reference" id="cite_ref-CASSIOPE_MDA_38-0">[3
2]</sup>
</td>
<td>500 kg (1,100 lb)
</td>
<td>Polar orbit <a href="/wiki/Low_Ear
th_orbit" title="Low Earth orbit">LEO
</td>
<td>MDA
</td>
<td class="table-success" style="background: #9EFF9E; vertical-align: middle; text-align:
center;">Success<sup class="reference" id="cite_ref-pa20130930_37-1"><a href="#cite_note-p
a20130930-37">[31]</sup>
</td>
<td class="table-no2" style="background: #FFE3E3; color: black; vertical-align: middle; te
xt-align: center;">Uncontrolled
<small>(ocean)</small><sup class="reference" id="cite_
ref-ocean_landing_39-0">[d]</sup>
</td></tr>
<tr>

<td colspan="9">First commercial mission with a private customer, first launch from Vandenberg, and demonstration flight of Falcon 9 v1.1 with an improved 13-tonne to LEO capacity.^{[30]} After separation from the second stage carrying Canadian commercial and scientific satellites, the first stage booster performed a controlled reentry,^{[33]} and an ocean touchdown test for the first time. This provided good test data, even though the booster started rolling as it neared the ocean, leading to the shutdown of the central engine as the roll depleted it of fuel, resulting in a hard impact with the ocean.^{[31]} This was the first known attempt of a rocket engine being lit to perform a supersonic retro propulsion, and allowed SpaceX to enter a public-private partnership with NASA and its Mars entry, descent, and landing technologies research projects.^{[34]} <small>(more details below)</small>
</td></tr>
<tr>
<th rowspan="2" scope="row" style="text-align:center;">7
</th>
<td>3 December 2013,
22:41^{[35]}
</td>
<td>F9 v1.1
B1004
</td>
<td>CCAFS,
SLC-40
</td>
<td>SES-8^{[23]}^{[36]}^{[37]}
</td>
<td>3,170 kg (6,990 lb)
</td>
<td>GTO

</td>
<td>SES
</td>
<td class="table-success" style="background: #9EFF9E; vertical-align: middle; text-align: center;">Success^{[38]}
</td>
<td class="table-noAttempt" style="background: #EEE; vertical-align: middle; white-space: nowrap; text-align: center;">No attempt
^{[39]}
</td></tr>
<tr>
<td colspan="9">First Geostationary transfer orbit (GTO) launch for Falcon 9,^{[36]} and first successful reignition of the second stage.^{[40]} SES-8 was inserted into a Super-Synchronous Transfer Orbit of 79,341 km (49,300 mi) in apogee with an inclination of 20.55° to the equator.
</td></tr></tbody></table>

In [205…

```
first_table=table[2]
```

Extract all column/variable names from the HTML table header

In [205…
```
def extract_col_name(row):
    if (row.br):
        row.br.extract()
    if (row.a):
        row.a.extract()
    if (row.sup):
        row.sup.extract()

    col_name=" ".join(row.contents)
    if not(col_name.strip().isdigit()):
        col_name=col_name.strip()
    return col_name
```

In [205…
```
col_name=[]
th_element=first_table.find_all('th')
for element in th_element:
    name=extract_col_name(element)
    if name is not None and len(name)>0 and not(name.strip().isdigit()):
        col_name.append(name)
```

Check the extracted column names

In [205…
```
col_name
```

Out[205…
```
['Flight No.',
 'Date and time ( )',
 'Launch site',
 'Payload',
 'Payload mass',
 'Orbit',
 'Customer',
 'Launch outcome']
```

We will create an empty dictionary with keys from the extracted column name and the dictionary will be converted into a Pandas dataframe

In [206…
```
data_dict=dict.fromkeys(col_name)

del data_dict['Date and time ( )']
data_dict['Flight No.']=[]
data_dict['Launch site']=[]
data_dict['Payload']=[]
data_dict['Payload mass']=[]
data_dict['Orbit']=[]
data_dict['Customer']=[]
data_dict['Launch outcome']=[]
data_dict['Version booster']=[]
data_dict['Booster landing']=[]
data_dict['Date']=[]
data_dict['Time']=[]
```

In [206…
```
def date_time(data):
    date=[x.strip() for x in list(data.strings)][0:2]
    return date
def booster_version(data):
    boost=" ".join([x for i, x in enumerate(data.strings) if i%2==0][0:-1])
    return boost
```

In [206…

```
def landing_status(data):
    out=[i for i in data.strings][0]
    return out


def get_mass(data):
    mass=unicodedata.normalize("NFKD", data.text).strip()
    if mass:
        mass.find("kg")
        new_mass=mass[0:mass.find("kg")+2]
    else:
        new_mass=0
    return new_mass
```

Next, we just need to fill up the launch_dict with launch records extracted from table rows.

In [206…

```
extracted_row = 0
for table_number, table in enumerate(soup.find_all('table',"wikitable plainrowheaders coll
    for rows in table.find_all('tr'):
        if rows.th:
            if rows.th.string:
                flight_num=rows.th.string.strip()
                bools=flight_num.isdigit()
        else:
            bools=False
        row=rows.find_all('td')
        if bools:
            extracted_row+=1
            datetime=date_time(row[0])
            date=datetime[0].strip(',')
            data_dict['Date'].append(date)

            time=datetime[1]
            data_dict['Time'].append(time)

            data_dict['Flight No.'].append(flight_num)

            d=booster_version(row[1])
            if not(d):
                d=row[1].a.string
            data_dict['Version booster'].append(d)



            a=" ".join([x.strip() for i,x in enumerate(list(row[2].strings)) if i%2==0][0:
            data_dict['Launch site'].append(a)


            payload = row[3].a.string
            data_dict["Payload"].append(payload)


            payload_mass = get_mass(row[4])
            data_dict['Payload mass'].append(payload_mass)



            orbit = row[5].a.string
            data_dict['Orbit'].append(orbit)



            customer = row[6].text.strip()
```

```
            data_dict['Customer'].append(customer)


            launch_outcome = list(row[7].strings)[0]

            launch_outcome=launch_outcome.strip('\n')

            data_dict['Launch outcome'].append(launch_outcome)


            booster_landing = landing_status(row[8])
            data_dict['Booster landing'].append(booster_landing)
```

Create a dataframe by parsing the launch HTML tables

After fill in the parsed launch record values into data dictionary, create a dataframe from it.

In [206…

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

Out[206…

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version booster	Booster landing	Date	Tim
0	1	CCAFS SLC-40	Dragon Spacecraft Qualification Unit	N	LEO	SpaceX	Success	F9 v1.0 B0003	Failure	4 June 2010	18:4
1	2	CCAFS SLC-40	Dragon	C	LEO	NASA (COTS)\nNRO	Success	F9 v1.0 B0004	Failure	8 December 2010	15:4
2	3	CCAFS SLC-40	Dragon	525 kg	LEO	NASA (COTS)	Success	F9 v1.0 B0005	No attempt\n	22 May 2012	07:4
3	4	CCAFS SLC-40	SpaceX CRS-1	4,700 kg	LEO	NASA (CRS)	Success	F9 v1.0 B0006	No attempt	8 October 2012	00:3
4	5	CCAFS SLC-40	SpaceX CRS-2	4,877 kg	LEO	NASA (CRS)	Success	F9 v1.0 B0007	No attempt\n	1 March 2013	15:1
…	…	…	…	…	…	…	…	…	…	…	
143	144	CCSFS SLC-40	Starlink	~14,160 kg	LEO	SpaceX	Success	F9 B5 B1052.4	Success	9 March 2022	13:4
144	145	CCSFS SLC-40	Starlink	~16,250 kg	LEO	SpaceX	Success	F9 B5 B1051.12	Success	19 March 2022	03:2
145	146	CCSFS SLC-40	None	U	SSO	Various	Success	F9 B5 B1061.7	Success	1 April 2022	16:2
146	147	KSC LC-39A	Axiom-1	~13,000 kg	LEO	Axiom Space	Success	F9 B5 B1062.5	Success	8 April 2022	[77
147	148	VSFB SLC-4E	NROL-85	C	LEO	NRO	Success	F9 B5	Success	17 April 2022	[77

148 rows × 11 columns

In [206…

```
df['Booster landing'].value_counts()
```

Out[206…

```
Success          107
No attempt        18
```

```
Failure           10
Controlled         5
No attempt\n       4
Uncontrolled       2
Failure            1
Precluded          1
Name: Booster landing, dtype: int64
```

Export dataframe to file csv for next processing

In [206…
```
df.to_csv('xspace.csv')
```

import csv file to data and parse date columns to date time formate

In [206…
```
data=pd.read_csv('xspace.csv',usecols=range(1,12),parse_dates=['Date'])
```

Using the month and year function to convert date columns to new columns month and year columns

In [206…
```
data['Month']=data['Date'].dt.month
data['Year']=data['Date'].dt.year
```

Replace the columns name into the correct format name

In [206…
```
data.rename(columns={'Flight No.':'FlightNo','Launch site':'LaunchSite','Payload mass':'Pa
```

data wrangling and basic exploratory

As the unique value in Launchsite columns are same, this might be the space in the value. we need to strip the space from values

In [207…
```
site=data['LaunchSite'].value_counts()
site
```

Out[207…
```
KSC LC-39A               42
CCAFS SLC-40             39
CCSFS SLC-40             21
Cape Canaveral LC-40     19
VAFB SLC-4E              16
VSFB SLC-4E               5
CCSFS SLC-40              2
Cape Canaveral SLC-40     1
CCAFS                     1
KSC LC-39A                1
VSFB                      1
Name: LaunchSite, dtype: int64
```

In [207…
```
data['LaunchSite']=data['LaunchSite'].str.strip(" ")
data
```

Out[207…

	FlightNo	LaunchSite	Payload	PayloadMass(kg)	Orbit	Customer	LaunchOutcome	VersionBoost
0	1	CCAFS SLC-40	Dragon Spacecraft Qualification Unit	N	LEO	SpaceX	Success	F9 v1.0 B00(
1	2	CCAFS SLC-40	Dragon	C	LEO	NASA (COTS)\nNRO	Success	F9 v1.0 B00(

	FlightNo	LaunchSite	Payload	PayloadMass(kg)	Orbit	Customer	LaunchOutcome	VersionBoost
2	3	CCAFS SLC-40	Dragon	525 kg	LEO	NASA (COTS)	Success	F9 v1.0 B00(
3	4	CCAFS SLC-40	SpaceX CRS-1	4,700 kg	LEO	NASA (CRS)	Success	F9 v1.0 B00(
4	5	CCAFS SLC-40	SpaceX CRS-2	4,877 kg	LEO	NASA (CRS)	Success	F9 v1.0 B00(
...	...	...	...	...	...	...	...	
143	144	CCSFS SLC-40	Starlink	~14,160 kg	LEO	SpaceX	Success	F9 B5 B1052
144	145	CCSFS SLC-40	Starlink	~16,250 kg	LEO	SpaceX	Success	F9 B5 B1051.
145	146	CCSFS SLC-40	NaN	U	SSO	Various	Success	F9 B5 B106'
146	147	KSC LC-39A	Axiom-1	~13,000 kg	LEO	Axiom Space	Success	F9 B5 B1062
147	148	VSFB SLC-4E	NROL-85	C	LEO	NRO	Success	F9 I

148 rows × 13 columns

In [207…

```
site=data['LaunchSite'].value_counts()
site
```

Out[207…

```
KSC LC-39A              43
CCAFS SLC-40            39
CCSFS SLC-40            23
Cape Canaveral LC-40    19
VAFB SLC-4E             16
VSFB SLC-4E              5
Cape Canaveral SLC-40    1
CCAFS                    1
VSFB                     1
Name: LaunchSite, dtype: int64
```

Refer to the launchsite from wikipedia shown, we need to clean data as same in wikipedia

First group the unique values into the same group

- CCAFS
- VAFB

In [207…

```
for i,key in enumerate(site.keys()):
    print(i,key)
CCAFS=set(site.keys()[[1,2,3,6,7]])
VAFB=set(site.keys()[[4,5,8]])
```

```
0 KSC LC-39A
1 CCAFS SLC-40
2 CCSFS SLC-40
3 Cape Canaveral LC-40
4 VAFB SLC-4E
5 VSFB SLC-4E
6 Cape Canaveral SLC-40
```

```
7 CCAFS
8 VSFB
```

Using the lambda function to separate the value into each grouop as we group as above

In [207…
```
data['LaunchSite']=data['LaunchSite'].apply(lambda x: 'CCSFS SLC-40' if (x in CCAFS) else
```

Now we can group the unique value into 3 group

In [207…
```
data['LaunchSite'].value_counts()
```

Out[207…
```
CCSFS SLC-40    83
KSC LC-39A      43
VSFB SLC-4E     22
Name: LaunchSite, dtype: int64
```

BoosterLanding

As the unique value in BoosterLanding columns are same, this might be the space in the value. we need to strip the space from values

In [207…
```
data['BoosterLanding'].value_counts()
```

Out[207…
```
Success          107
No attempt        18
Failure           10
Controlled         5
No attempt\n       4
Uncontrolled       2
Failure            1
Precluded          1
Name: BoosterLanding, dtype: int64
```

In [207…
```
data['BoosterLanding']=data['BoosterLanding'].str.strip(" ")
```

In [207…
```
booster=data['BoosterLanding'].value_counts()
booster
```

Out[207…
```
Success          107
No attempt        18
Failure           11
Controlled         5
No attempt\n       4
Uncontrolled       2
Precluded          1
Name: BoosterLanding, dtype: int64
```

Change "No attempt\n" into "No attempt"

In [207…
```
data['BoosterLanding']=data['BoosterLanding'].apply(lambda x: 'No attempt' if (x == 'No at
```

In [208…
```
booster=data['BoosterLanding'].value_counts()
booster
```

Out[208…
```
Success          107
No attempt        22
Failure           11
Controlled         5
```

```
Uncontrolled      2
Precluded         1
Name: BoosterLanding, dtype: int64
```

PayloadMass

In [208…
```
data['PayloadMass(kg)'].value_counts()
```

Out[208…
```
15,600 kg     25
9,600 kg       7
C              6
~14,500 kg     4
~13,000 kg     3
              ..
6,070 kg       1
5,600 kg       1
2,490 kg       1
4,600 kg       1
~16,250 kg     1
Name: PayloadMass(kg), Length: 101, dtype: int64
```

Remove special character '~' and 'kg' from PayloadMass columns

In [208…
```
data['PayloadMass(kg)']= data['PayloadMass(kg)'].str.strip("~")
data['PayloadMass(kg)']= data['PayloadMass(kg)'].str.strip(" kg")
```

In [208…
```
data['PayloadMass(kg)'].value_counts()
```

Out[208…
```
15,600    25
9,600      7
C          6
14,500     4
13,000     3
          ..
6,070      1
5,600      1
2,490      1
4,600      1
16,250     1
Name: PayloadMass(kg), Length: 100, dtype: int64
```

As the values in columns PayloadMass(kg) contain 'C' for Classifier , 'N' for No load, 'U' for Unknown or not specific, we will replace it by 0 and change data type to integer

In [208…
```
payloadmass=data['PayloadMass(kg)'].value_counts()
```

In [208…
```
for i,key in enumerate(payloadmass.keys()):
    print(i,key)
```

```
0 15,600
1 9,600
2 C
3 14,500
4 13,000
5 U
6 2,205
7 5,300
8 4,311
9 3,600
10 7,000
```

11 3,500
12 6,500
13 12,055
14 2,500
15 4,400
16 4,850
17 15,410
18 12,530
19 1,977
20 12,050
21 2,268
22 2,495
23 6,956
24 2,617
25 13,620
26 4,200
27 5,000–6,000
28 N
29 3,130
30 14,932
31 12,519
32 14,160
33 13,900
34 14,750
35 13,600
36 2,989
37 4,500
38 325
39 624
40 15,635
41 13,260
42 15,440
43 2,200
44 4,331
45 3,328
46 14,000
47 5,000
48 2,972
49 1,192
50 12,500
51 3,000
52 4,000
53 7,075
54 7,060
55 2,216
56 5,271
57 553
58 2,034
59 1,952
60 4,707
61 1,898
62 4,159
63 570
64 2,395
65 4,428
66 5,800
67 4,535
68 1,316
69 2,296
70 3,325
71 3,170
72 500
73 4,877
74 4,700
75 525
76 3,136

```
77 4,696
78 3,100
79 2,257
80 2,697
81 5,384
82 6,460
83 362
84 2,647
85 6,092
86 2,150
87 4,230
88 5,400
89 4,990
90 475
91 3,310
92 6,761
93 3,669
94 2,708
95 6,070
96 5,600
97 2,490
98 4,600
99 16,250
```

In [208…
```
zeromass=set(payloadmass.keys()[[2,5,28]])
rangemass=set(payloadmass.keys()[[27]])
```

In [208…
```
data['PayloadMass(kg)']=data['PayloadMass(kg)'].apply(lambda x: 0 if (x in zeromass) else
```

In [208…
```
data['PayloadMass(kg)']=data['PayloadMass(kg)'].str.replace(',','')
```

In [208…
```
data['PayloadMass(kg)']=data['PayloadMass(kg)'].fillna(0)
```

In [209…
```
data['PayloadMass(kg)']=data['PayloadMass(kg)'].astype('int')
```

Orbit

Group "Polar" and "Polar orbit" in one group

In [209…
```
data['Orbit'].value_counts()
```

Out[209…
```
LEO             86
GTO             34
SSO             12
Polar            7
MEO              4
Polar orbit      1
HEO              1
Sub-orbital      1
Heliocentric     1
Name: Orbit, dtype: int64
```

In [209…
```
data['Orbit']=data['Orbit'].apply(lambda x: 'Polar' if (x == 'Polar orbit') else x)
```

Customer

In [209…

```
customer=data['Customer'].value_counts()
customer
```

Out[209…

```
SpaceX                                          37
NASA (CRS)                                      24
Iridium Communications                           7
SES                                              5
NRO                                              4
Various                                          4
NASA (CTS)[497]                                  3
NASA (LSP)                                       3
SKY Perfect JSAT Group                           2
Telesat                                          2
SpaceXPlanet Labs                                2
USAF                                             2
Sirius XM                                        2
Türksat                                          2
ABS\nEutelsat                                    2
AsiaSat                                          2
Thaicom                                          2
Orbcomm                                          2
SpaceX Capella Space and Tyvak                   1
U.S. Space Force[530]                            1
Spacecom                                         1
Sky Perfect JSATKacific 1                        1
ASI                                              1
SpaceXSpaceflight, Inc. (BlackSky Global)        1
NASA (CCDev)                                     1
SpaceXSpaceflight Industries (BlackSky)          1
Republic of Korea Army                           1
USSF[530]                                        1
CONAEPlanetIQTyvak                               1
USSF                                             1
NASA (CCP)[497]                                  1
NASA / NOAA / ESA / EUMETSAT                     1
Jared Isaacman[note 1][703][704]                 1
NASA (CCD)                                       1
Canadian Space Agency (CSA)                      1
Iridium Communications\nGFZ • NASA               1
PSN\nSpaceIL / IAI\nAir Force Research           1
NSPO                                             1
NASA (COTS)                                      1
MDA                                              1
USAF\nNASA\nNOAA                                 1
Turkmenistan NationalSpace Agency[90]            1
NASA (LSP)\nNOAA\nCNES                           1
EchoStar                                         1
Inmarsat                                         1
Bulsatcom                                        1
Intelsat                                         1
SES S.A.\nEchoStar                               1
Spaceflight Industries                           1
KT Corporation                                   1
Northrop Grumman [f]                             1
Hisdesat\nexactEarth\nSpaceX                     1
Hispasat[277]\nNovaWurks                         1
Thales-Alenia / BTRC                             1
NASA (COTS)\nNRO                                 1
Telkom Indonesia                                 1
CONAE                                            1
Es'hailSat                                       1
Axiom Space                                      1
Name: Customer, dtype: int64
```

Replace the special character '\n' and '/' in columns Customer with space

In [209…

```
data['Customer']= data['Customer'].str.replace('\n',' ')
data['Customer']= data['Customer'].str.replace('/',' ')
```

In [209…

```
for i,key in enumerate(customer.keys()):
    print(i,key)
```

```
0 SpaceX
1 NASA (CRS)
2 Iridium Communications
3 SES
4 NRO
5 Various
6 NASA (CTS)[497]
7 NASA (LSP)
8 SKY Perfect JSAT Group
9 Telesat
10 SpaceXPlanet Labs
11 USAF
12 Sirius XM
13 Türksat
14 ABS
Eutelsat
15 AsiaSat
16 Thaicom
17 Orbcomm
18 SpaceX Capella Space and Tyvak
19 U.S. Space Force[530]
20 Spacecom
21 Sky Perfect JSATKacific 1
22 ASI
23 SpaceXSpaceflight, Inc. (BlackSky Global)
24 NASA (CCDev)
25 SpaceXSpaceflight Industries (BlackSky)
26 Republic of Korea Army
27 USSF[530]
28 CONAEPlanetIQTyvak
29 USSF
30 NASA (CCP)[497]
31 NASA / NOAA / ESA / EUMETSAT
32 Jared Isaacman[note 1][703][704]
33 NASA (CCD)
34 Canadian Space Agency (CSA)
35 Iridium Communications
GFZ • NASA
36 PSN
SpaceIL / IAI
Air Force Research
37 NSPO
38 NASA (COTS)
39 MDA
40 USAF
NASA
NOAA
41 Turkmenistan NationalSpace Agency[90]
42 NASA (LSP)
NOAA
CNES
43 EchoStar
44 Inmarsat
45 Bulsatcom
46 Intelsat
```

```
47 SES S.A.
EchoStar
48 Spaceflight Industries
49 KT Corporation
50 Northrop Grumman [f]
51 Hisdesat
exactEarth
SpaceX
52 Hispasat[277]
NovaWurks
53 Thales-Alenia / BTRC
54 NASA (COTS)
NRO
55 Telkom Indonesia
56 CONAE
57 Es'hailSat
58 Axiom Space
```

Create group of Customer from SpaceX, Nasa, IridiumCommunications and Other

In [209…
```
SpaceX=set(customer.keys()[[0,10,18,23,25,51]])
NASA=set(customer.keys()[[1,6,7,24,30,31,33,38,40,42,54]])
IridiumCommunications=set(customer.keys()[[2,35]])
```

In [209…
```
data['GroupCustomer']=data['Customer'].apply(lambda x: 'SpaceX' if (x in SpaceX) else 'NAS
```

VersionBooster

In columns VersionBooster, split values into 3 columns by using space between value

In [209…
```
data['VersionBooster'].value_counts()
```

Out[209…
```
F9 B5            40
F9 FT            17
F9 v1.1          14
F9 B5 ♺          10
F9 B4             7
                 ..
F9 B5 B1048.3     1
F9 B5 [268]       1
F9 B5 B1049.3     1
F9 v1.0 B0004     1
F9 B5 B1062.5     1
Name: VersionBooster, Length: 65, dtype: int64
```

In [209…
```
data[["a","b","c"]]=data['VersionBooster'].str.split('\s',2,expand=True)
```

Replace the special character '♺' with reused in columns c and other value in columns c replace with no values

In [210…
```
data['c']=data['c'].replace('♺','reused')
```

In [210…
```
version=data['c'].value_counts()
```

In [210…
```
for i,key in enumerate(version.keys()):
    print(i,key)
```

```
0 reused
1 B0003
2 B1063.2
3 B1049.6
4 B1060.2
5 B1058.3
6 B1051.6
7 B1051.8
8 B1058.5
9 B1060.6
10 B1061.2
11 B1060.7
12 B1049.9
13 B1051.10
14 B1058.8
15 B1067.1
16 B1056.3
17 B1062.2
18 B1049.10
19 B1062.3
20 B1067.2
21 B1058.9
22 B1063.3
23 B1060.9
24 B1063.4
25 B1060.11
26 B1052.4
27 B1051.12
28 B1061.7
29 B1058.2
30 B1047.3
31 B0004
32 B1046.1
33 B0005
34 B0006
35 B0007
36 B1003
37 B1029.2
38 B1031.2
39 B1035.2
40 B1036.2
41 B1032.2
42 B1038.2
43 B1041.2
44 B1039.2
45 B1043.2
46 B1056.2
47 B1040.2
48 B1045.2
49 B1048
50 B1046.2
51 B1048.2
52 B1047.2
53 B1046.3
54 B1049.2
55 B1048.3
56 [268]
57 B1049.3
58 B1051.2
59 B1062.5
```

In [210…

```
reused=set(version.keys()[[0]])
```

Group values in columns c to 'reused' and other to empty

In [210…
```
data['c']=data['c'].apply(lambda x: 'reused' if (x in reused) else '')
```

In [210…
```
data['c'].value_counts()
```

Out[210…
```
          137
reused     11
Name: c, dtype: int64
```

In [210…
```
data['b'].value_counts()
```

Out[210…
```
B5       92
FT       24
v1.1     15
B4       12
v1.0      5
Name: b, dtype: int64
```

replace VersionBooster columns by combline the columns b and c and remove columns a,b,c after finish

In [210…
```
data['VersionBooster']=data['b']+' '+data['c']
data.drop(['a','c','b'],axis=1,inplace=True)
```

In [210…
```
group_reused=data['VersionBooster'].value_counts()
```

In [210…
```
for i,key in enumerate(group_reused.keys()):
    print(i,key)
```

```
0 B5
1 FT
2 v1.1
3 B4
4 B5 reused
5 v1.0
6 FT reused
```

In [211…
```
group_reused=set(group_reused.keys()[[4,6]])
```

Add new columns ReusedBooster by indicate the values reused and not reused from VersionBooster

In [211…
```
data['ReusedBooster']=data['VersionBooster'].apply(lambda x: "Yes" if (x in group_reused)
```

In [211…
```
data['ReusedBooster'].value_counts()
```

Out[211…
```
No      137
Yes      11
Name: ReusedBooster, dtype: int64
```

Payload

for spaceX mission can separate into 2 group

- crewed flights
- cargo delivery for all value that have word Crew in values will transform to crewd flights and other will be cargo delivery

In [211…
```
payload=data['Payload'].value_counts()
payload
```

Out[211…
```
Starlink         41
Iridium NEXT      8
GPS III           4
Dragon            2
Orbcomm-OG2       2
                 ..
NROL-76           1
SES-10            1
EchoStar 23       1
SpaceX CRS-10     1
NROL-85           1
Name: Payload, Length: 92, dtype: int64
```

In [211…
```
for i,key in enumerate(payload.keys()):
    print(i,key)
```

```
0 Starlink
1 Iridium NEXT
2 GPS III
3 Dragon
4 Orbcomm-OG2
5 Crew Dragon Demo-1
6 Crew Dragon in-flight abort test
7 JCSat-18
8 SpaceX CRS-19
9 AMOS-17
10 SpaceX CRS-18
11 RADARSAT Constellation
12 SpaceX CRS-17
13 Nusantara Satu
14 Crew Dragon Demo-2
15 SpaceX CRS-16
16 SSO-A
17 Es'hail 2
18 SAOCOM 1A
19 Telstar 18V
20 Merah Putih
21 Telstar 19V
22 SpaceX CRS-15
23 SpaceX CRS-20
24 Dragon Spacecraft Qualification Unit
25 SES-12
26 SAOCOM 1B
27 Axiom-1
28 NROL-87
29 CSG-2
30 SpaceX CRS-24
31 Türksat 5B
32 Imaging X-ray Polarimetry Explorer
33 Double Asteroid Redirection Test (DART)
34 Crew-3
35 Inspiration4
36 SpaceX CRS-23
37 SXM-8
38 SpaceX CRS-22
39 Crew-2
40 Türksat 5A
41 NROL-108
42 SXM-7
43 SpaceX CRS-21
44 Sentinel-6 Michael Freilich (Jason-CS A)
45 Crew-1
```

```
46 ANASIS-II
47 Transiting Exoplanet Survey Satellite
48 Bangabandhu-1
49 SpaceX CRS-14
50 JCSAT-14
51 SpaceX CRS-8
52 SES-9
53 Jason-3
54 SpaceX CRS-7
55 TürkmenÄlem 52°E / MonacoSAT
56 SpaceX CRS-6
57 ABS-3A
58 DSCOVR
59 SpaceX CRS-5
60 SpaceX CRS-4
61 AsiaSat 6
62 AsiaSat 8
63 SpaceX CRS-3
64 Thaicom 6
65 SES-8
66 CASSIOPE
67 SpaceX CRS-2
68 SpaceX CRS-1
69 Thaicom 8
70 ABS-2A
71 SpaceX CRS-9
72 Formosat-5
73 Hispasat 30W-6
74 Paz
75 GovSat-1
76 Zuma
77 SpaceX CRS-13
78 Koreasat 5A
79 SES-11
80 Boeing X-37B
81 SpaceX CRS-12
82 JCSAT-16
83 Intelsat 35e
84 BulgariaSat-1
85 SpaceX CRS-11
86 Inmarsat-5 F4
87 NROL-76
88 SES-10
89 EchoStar 23
90 SpaceX CRS-10
91 NROL-85
```

In [211…

```
data['TypicalMissions']=data['Payload'].str.replace('^Crew.*', 'Crew')
payload=data['TypicalMissions'].value_counts()
payload
```

Out[211…

```
Starlink         41
Iridium NEXT      8
Crew              6
GPS III           4
Dragon            2
                 ..
EchoStar 23       1
SpaceX CRS-10     1
JCSAT-16          1
SpaceX CRS-9      1
NROL-85           1
Name: TypicalMissions, Length: 87, dtype: int64
```

In [211…

```
for i,key in enumerate(payload.keys()):
    print(i,key)
```

```
0 Starlink
1 Iridium NEXT
2 Crew
3 GPS III
4 Dragon
5 Orbcomm-OG2
6 Dragon Spacecraft Qualification Unit
7 SpaceX CRS-16
8 SpaceX CRS-18
9 RADARSAT Constellation
10 SpaceX CRS-17
11 Nusantara Satu
12 Es'hail 2
13 SSO-A
14 SpaceX CRS-19
15 SAOCOM 1A
16 Telstar 18V
17 Merah Putih
18 Telstar 19V
19 SpaceX CRS-15
20 SES-12
21 Bangabandhu-1
22 AMOS-17
23 JCSat-18
24 SpaceX CRS-14
25 SpaceX CRS-23
26 Axiom-1
27 NROL-87
28 CSG-2
29 SpaceX CRS-24
30 Türksat 5B
31 Imaging X-ray Polarimetry Explorer
32 Double Asteroid Redirection Test (DART)
33 Inspiration4
34 SXM-8
35 SpaceX CRS-20
36 SpaceX CRS-22
37 Türksat 5A
38 NROL-108
39 SXM-7
40 SpaceX CRS-21
41 Sentinel-6 Michael Freilich (Jason-CS A)
42 SAOCOM 1B
43 ANASIS-II
44 Transiting Exoplanet Survey Satellite
45 Paz
46 Hispasat 30W-6
47 JCSAT-14
48 SES-9
49 Jason-3
50 SpaceX CRS-7
51 TürkmenÄlem 52°E / MonacoSAT
52 SpaceX CRS-6
53 ABS-3A
54 DSCOVR
55 SpaceX CRS-5
56 SpaceX CRS-4
57 AsiaSat 6
58 AsiaSat 8
59 SpaceX CRS-3
60 Thaicom 6
61 SES-8
```

```
62 CASSIOPE
63 SpaceX CRS-2
64 SpaceX CRS-1
65 SpaceX CRS-8
66 Thaicom 8
67 GovSat-1
68 ABS-2A
69 Zuma
70 SpaceX CRS-13
71 Koreasat 5A
72 SES-11
73 Boeing X-37B
74 Formosat-5
75 SpaceX CRS-12
76 Intelsat 35e
77 BulgariaSat-1
78 SpaceX CRS-11
79 Inmarsat-5 F4
80 NROL-76
81 SES-10
82 EchoStar 23
83 SpaceX CRS-10
84 JCSAT-16
85 SpaceX CRS-9
86 NROL-85
```

In [211…

```
crew=set(payload.keys()[[2]])
```

In [211…

```
data['TypicalMissions']=data['TypicalMissions'].apply(lambda x: 'crewed flights' if (x in
```

In [211…

```
data
```

Out[211…

	FlightNo	LaunchSite	Payload	PayloadMass(kg)	Orbit	Customer	LaunchOutcome	VersionBooster
0	1	CCSFS SLC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	v1.0
1	2	CCSFS SLC-40	Dragon	0	LEO	NASA (COTS) NRO	Success	v1.0
2	3	CCSFS SLC-40	Dragon	525	LEO	NASA (COTS)	Success	v1.0
3	4	CCSFS SLC-40	SpaceX CRS-1	4700	LEO	NASA (CRS)	Success	v1.0
4	5	CCSFS SLC-40	SpaceX CRS-2	4877	LEO	NASA (CRS)	Success	v1.0
...	...	...	...	...	...	...	...	...
143	144	CCSFS SLC-40	Starlink	14160	LEO	SpaceX	Success	B5
144	145	CCSFS SLC-40	Starlink	16250	LEO	SpaceX	Success	B5
145	146	CCSFS SLC-40	NaN	0	SSO	Various	Success	B5

	FlightNo	LaunchSite	Payload	PayloadMass(kg)	Orbit	Customer	LaunchOutcome	VersionBooster
146	147	KSC LC-39A	Axiom-1	13000	LEO	Axiom Space	Success	B5
147	148	VSFB SLC-4E	NROL-85	0	LEO	NRO	Success	B5

148 rows × 16 columns

In [212…
```
data['GroupCustomer'].value_counts()
```

Out[212…
```
Other                       65
SpaceX                      42
NASA                        34
Iridium Communications       7
Name: GroupCustomer, dtype: int64
```

Create outcome columne from BoosterLanding columns

create 1 for success landing and 0 for otherwise for columns Outcome

In [212…
```
data['BoosterLanding'].value_counts()
```

Out[212…
```
Success          107
No attempt        22
Failure           11
Controlled         5
Uncontrolled       2
Precluded          1
Name: BoosterLanding, dtype: int64
```

In [212…
```
outcome=data['BoosterLanding'].value_counts()
for i,key in enumerate(outcome.keys()):
    print(i,key)
```

```
0 Success
1 No attempt
2 Failure
3 Controlled
4 Uncontrolled
5 Precluded
```

In [212…
```
success=set(outcome.keys()[[0,3]])
```

In [212…
```
data['Outcome']=data['BoosterLanding'].apply(lambda x: 1 if (x in success) else 0)
```

In [212…
```
data['Outcome'].value_counts()
```

Out[212…
```
1    112
0     36
Name: Outcome, dtype: int64
```

Basic Visualization

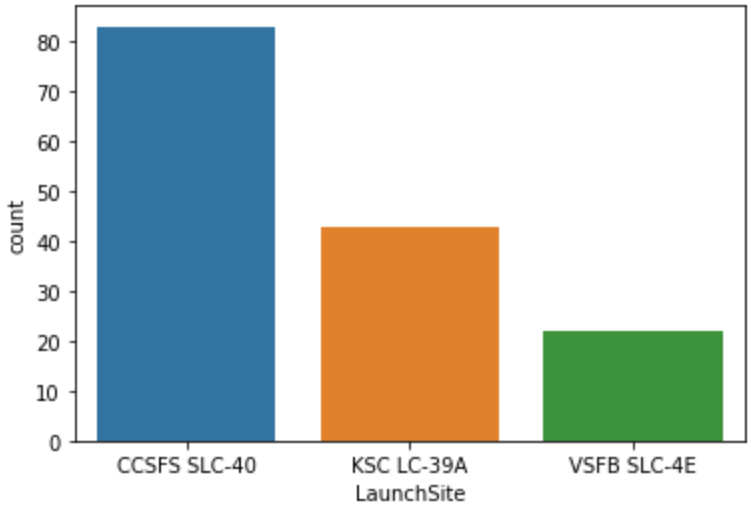
Visual check each values in the LaunchSite columns shown where the most used for launch is CCSFS SLC-

In [212…
```
data['LaunchSite'].value_counts()
```

Out[212…
```
CCSFS SLC-40    83
KSC LC-39A      43
VSFB SLC-4E     22
Name: LaunchSite, dtype: int64
```

In [212…
```
sns.countplot(data['LaunchSite'],order=data['LaunchSite'].value_counts().index)
```

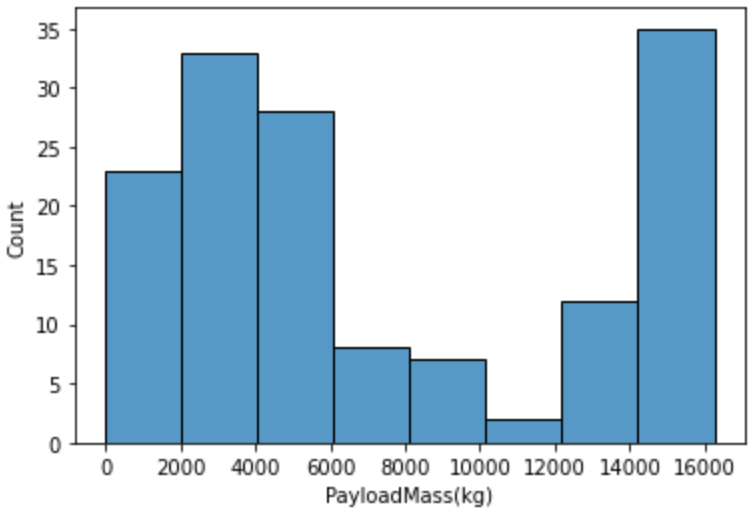
Out[212…
```
<AxesSubplot:xlabel='LaunchSite', ylabel='count'>
```



Visual check the distribution of payloadmass columns shown the most mass that load in the low mass

In [212…
```
sns.histplot(data['PayloadMass(kg)'], bins=8)
```

Out[212…
```
<AxesSubplot:xlabel='PayloadMass(kg)', ylabel='Count'>
```



In [212…
```
data['Orbit'].value_counts()
```

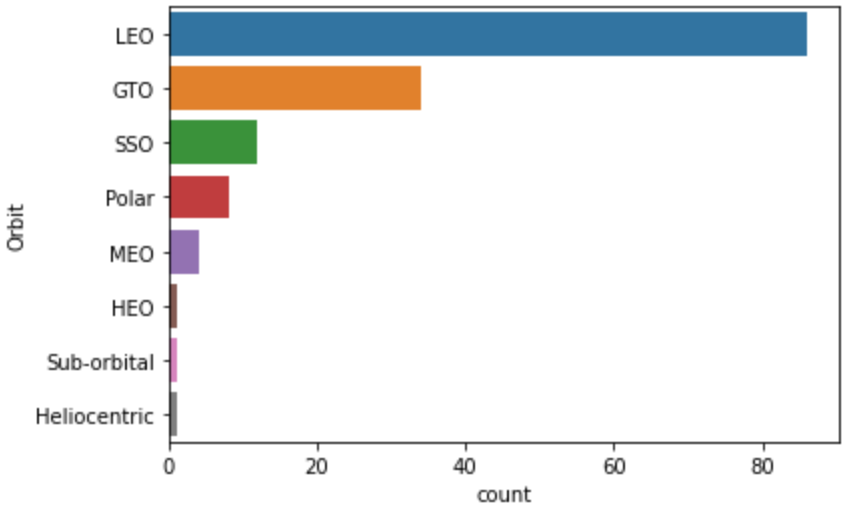
Out[212…
```
LEO             86
GTO             34
SSO             12
Polar            8
MEO              4
HEO              1
Sub-orbital      1
```

```
Heliocentric     1
Name: Orbit, dtype: int64
```

the rockets most launch to LEO or Low Earth orbit

In [213…
```
sns.countplot(y=data['Orbit'],order=data['Orbit'].value_counts().index)
```

Out[213…
```
<AxesSubplot:xlabel='count', ylabel='Orbit'>
```



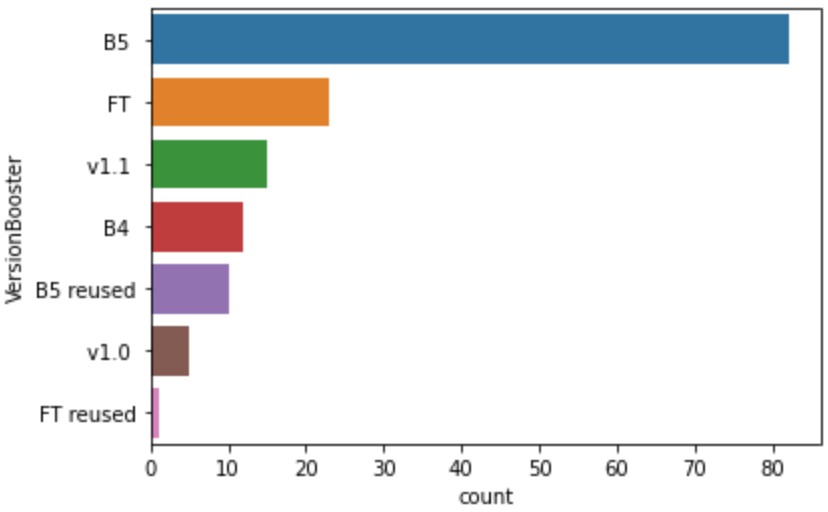
In [213…
```
data['VersionBooster'].value_counts()
```

Out[213…
```
B5            82
FT            23
v1.1          15
B4            12
B5 reused     10
v1.0           5
FT reused      1
Name: VersionBooster, dtype: int64
```

Most Booster version using for launch is B5 with no reused

In [213…
```
sns.countplot(y=data['VersionBooster'],order=data['VersionBooster'].value_counts().index)
```

Out[213…
```
<AxesSubplot:xlabel='count', ylabel='VersionBooster'>
```



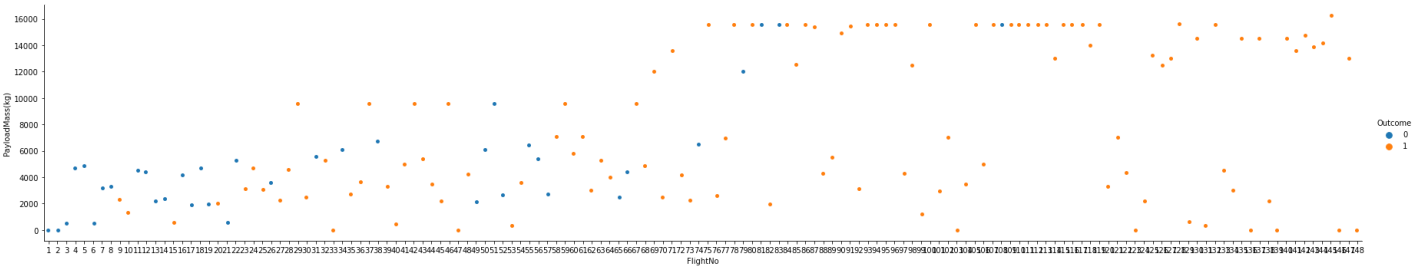
In [213…
```
data['BoosterLanding'].value_counts()
```

Out[213…
```
Success          107
No attempt        22
Failure           11
Controlled         5
Uncontrolled       2
Precluded          1
Name: BoosterLanding, dtype: int64
```

the chart shown the outcome that unsuccess was occure in low of flight number and with low flight number no high mass payload is apply to rockets

In [213…
```
sns.catplot(y="PayloadMass(kg)", x="FlightNo", hue="Outcome", data=data, aspect = 5)
```

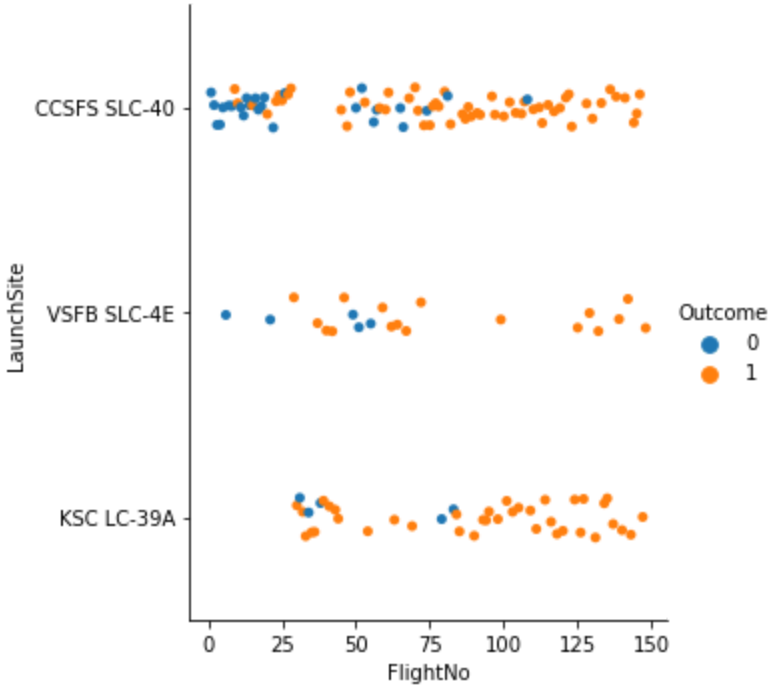
Out[213…
```
<seaborn.axisgrid.FacetGrid at 0x7ff57d7828b0>
```



as the chart below shown that all launchsite location, the unsuccess outcome most occure in the low flight number

In [213…
```
sns.catplot(x='FlightNo',y='LaunchSite',hue='Outcome',data=data,)
```

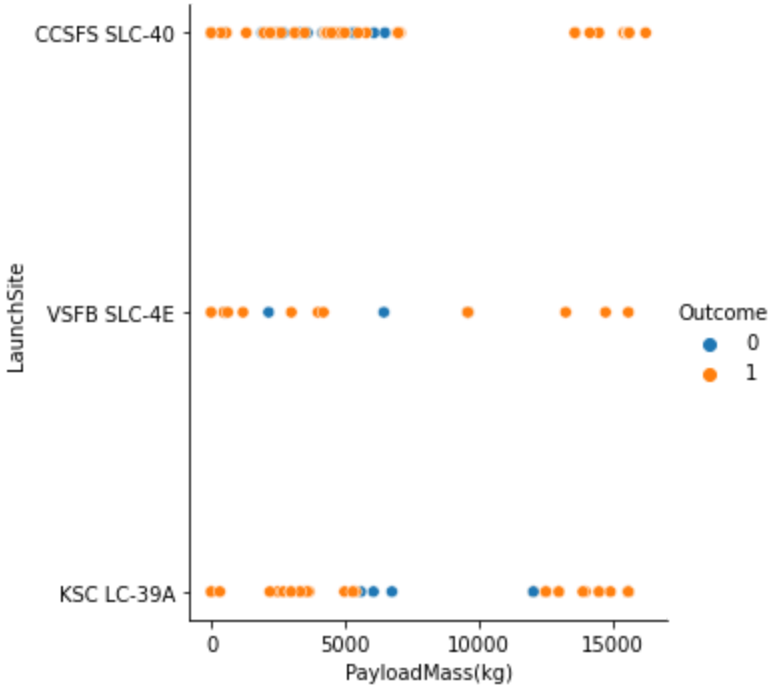
Out[213…
```
<seaborn.axisgrid.FacetGrid at 0x7ff5285a0df0>
```



As the low mass payload applie in the beginning period launch (low flight number) and also the most unsuccess occur in this period, the most unsuccess rockets payload is low as the chart below

In [213…
```
sns.relplot(x='PayloadMass(kg)',y='LaunchSite',data=data,kind='scatter',hue='Outcome')
```

Out[213…
```
<seaborn.axisgrid.FacetGrid at 0x7ff57d7864c0>
```

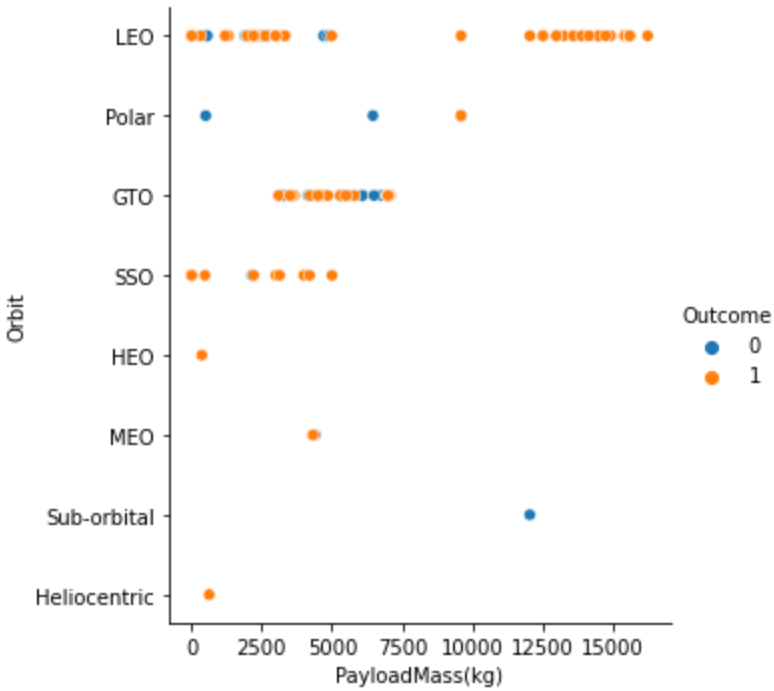


In [213…

```
sns.relplot(x='PayloadMass(kg)',y='Orbit',data=data,kind='scatter',hue='Outcome')
```

Out[213…

```
<seaborn.axisgrid.FacetGrid at 0x7ff57e7af100>
```



In [213…

```
sns.relplot(x='FlightNo',y='Orbit',data=data,kind='scatter',hue='Outcome')
```

Out[213…

```
<seaborn.axisgrid.FacetGrid at 0x7ff57d764070>
```

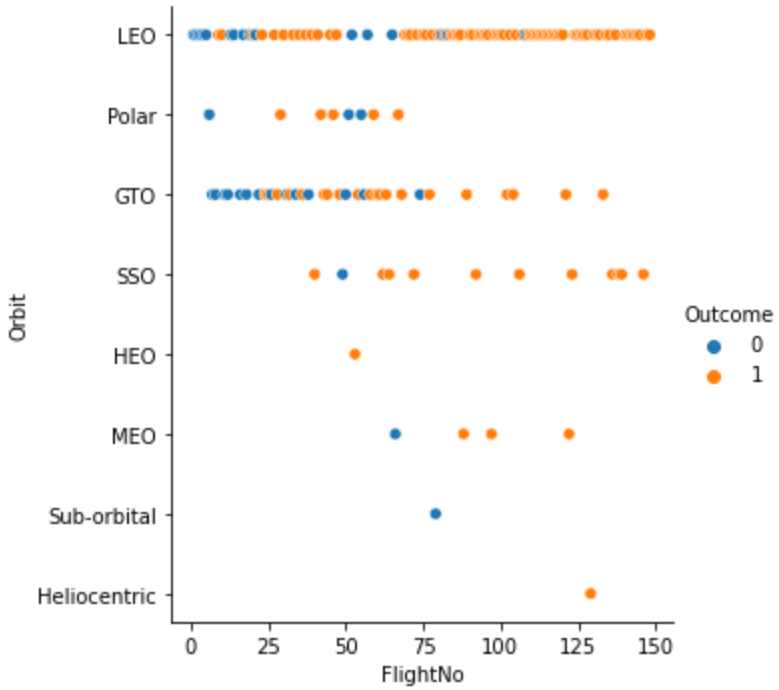
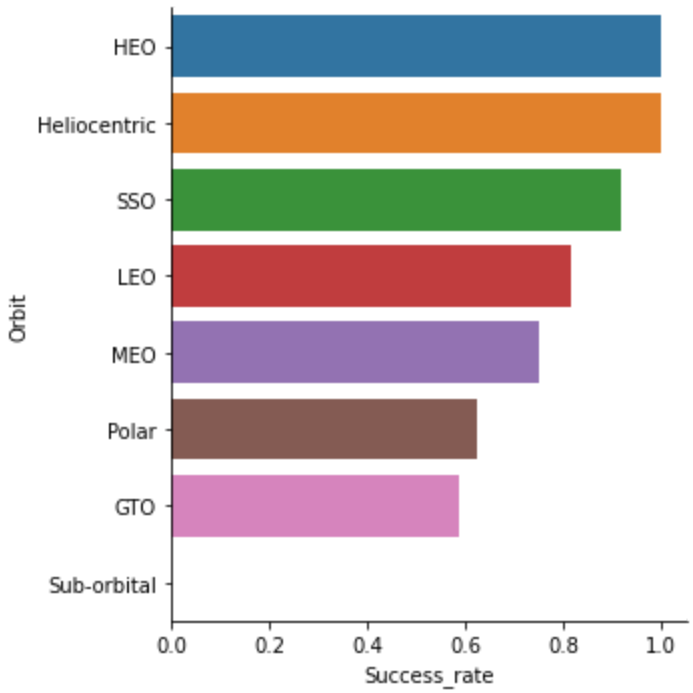


chart below shown the success rate compare the launch to orbit target. as HEO and Heliocentric orbit has only one flight, that why shown the high score. For SSO with high score due to the launch to this orbit start when has more flight number

In [213…

```
dataplot=data.groupby('Orbit')['Outcome'].mean().reset_index()
dataplot.sort_values('Outcome',ascending=False, inplace=True)
dataplot.rename(columns={'Outcome':'Success_rate'},inplace=True)
sns.catplot(data=dataplot,kind='bar',y='Orbit',x='Success_rate');
```



as the success rate for each launchsite also come from this launchsite was use after has more flight number

In [214…

```
dataplot=data.groupby('LaunchSite')['Outcome'].mean().reset_index()
dataplot.sort_values('Outcome',ascending=False, inplace=True)
dataplot.rename(columns={'Outcome':'Success_rate'},inplace=True)
sns.catplot(data=dataplot,kind='bar',y='LaunchSite',x='Success_rate');
```

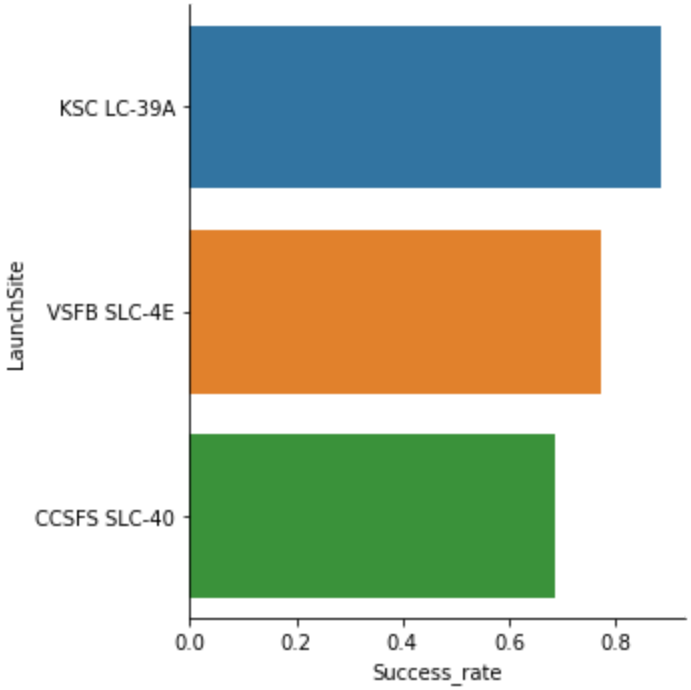


Chart below shown the success rate compare in booster version

In [214…

```
dataplot=data.groupby('VersionBooster')['Outcome'].mean().reset_index()
dataplot.sort_values('Outcome',ascending=False, inplace=True)
dataplot.rename(columns={'Outcome':'Success_rate'},inplace=True)
sns.catplot(data=dataplot,kind='bar',y='VersionBooster',x='Success_rate');
```

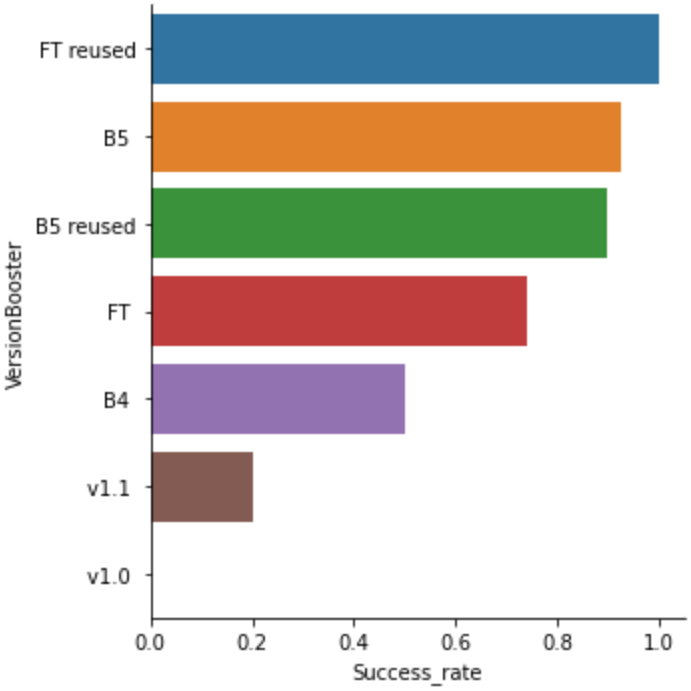


Chart below shown the success rate compart the mission

In [214…

```
dataplot=data.groupby('TypicalMissions')['Outcome'].mean().reset_index()
dataplot.sort_values('Outcome',ascending=False, inplace=True)
dataplot.rename(columns={'Outcome':'Success_rate'},inplace=True)
sns.catplot(data=dataplot,kind='bar',y='TypicalMissions',x='Success_rate');
```

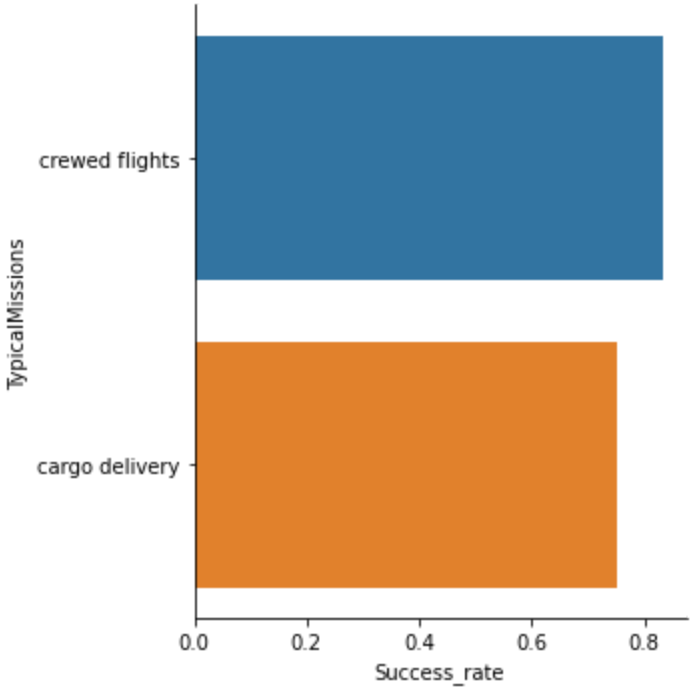
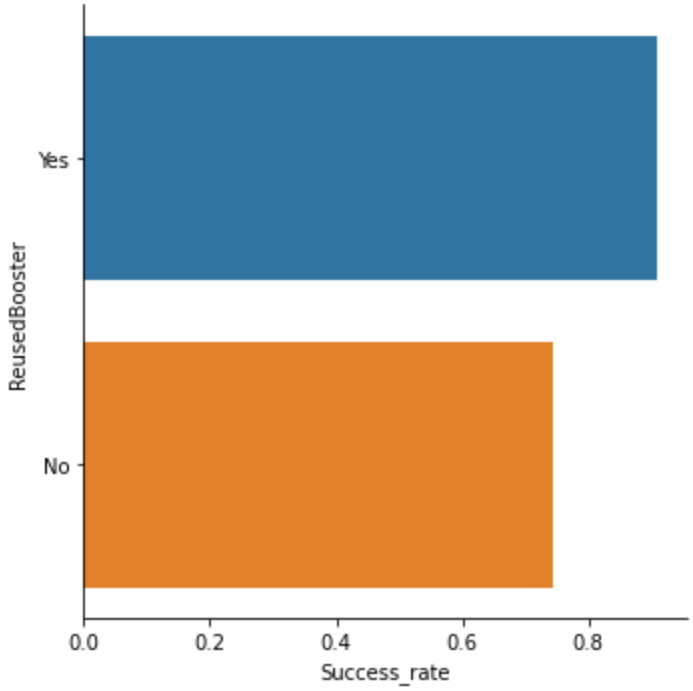


chart below shown the success rate compare reused and not reused booster

In [214…

```
dataplot=data.groupby('ReusedBooster')['Outcome'].mean().reset_index()
dataplot.sort_values('Outcome',ascending=False, inplace=True)
dataplot.rename(columns={'Outcome':'Success_rate'},inplace=True)
sns.catplot(data=dataplot,kind='bar',y='ReusedBooster',x='Success_rate');
```



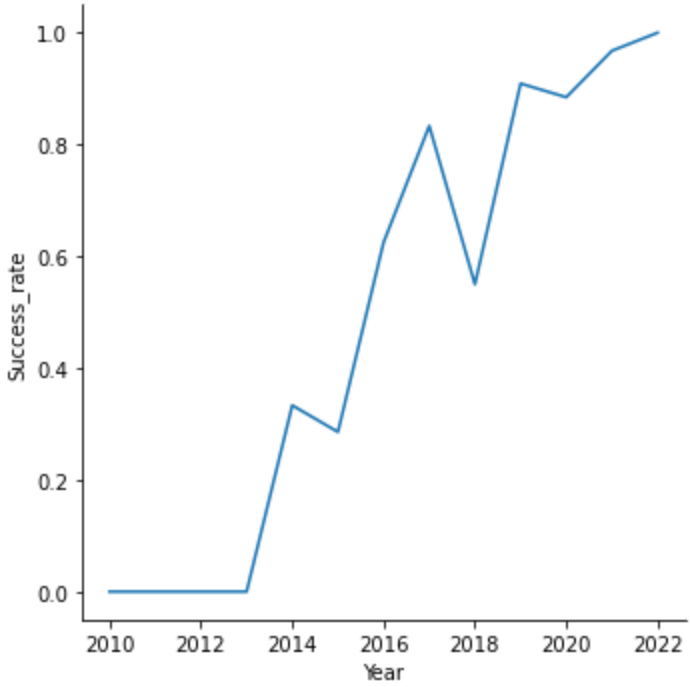
as the chart below shown that the success rate rais by current year

In [214…

```
data_plot=data.groupby('Year')['Outcome'].mean().reset_index()
data_plot.rename(columns={'Outcome':'Success_rate'},inplace=True)
sns.relplot(kind='line',x='Year',y='Success_rate',data=data_plot)
```

Out[214…

```
<seaborn.axisgrid.FacetGrid at 0x7ff5454d5610>
```



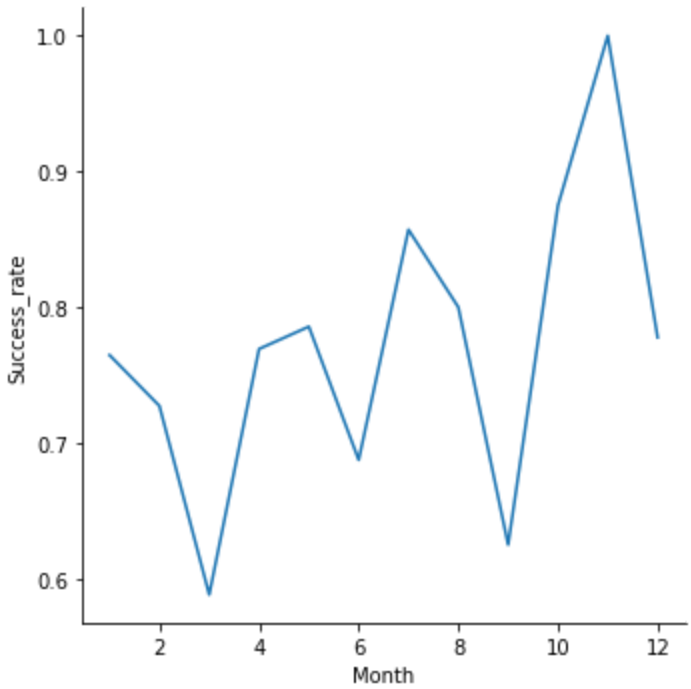
as the chart below shown some season effect to the success rate

In [214…

```
data_plot=data.groupby('Month')['Outcome'].mean().reset_index()
data_plot.rename(columns={'Outcome':'Success_rate'},inplace=True)
sns.relplot(kind='line',x='Month',y='Success_rate',data=data_plot)
```

Out[214…

```
<seaborn.axisgrid.FacetGrid at 0x7ff5186920d0>
```



conclusion

as the visualization, we notice that as the fisrt period of flight no, there are less success rate and can observe that the sucess rate since 2013 kept increasing till 2020. that mean with more flight no we can get the more success rate

In [ ]: