DATA SCIENCE EDA PROJECT

Exploratory Data Analysis (EDA)

SpaceX

Launches Falcon9 Dataset

What is EDA:

Exploratory Data Analysis (EDA) is a method used to understand the dataset before applying any model or algorithm. It helps identify trends, irregularities, missing values, and key relationships within the data.
EDA provides clarity on how the data behaves and what insights can be derived from it.

What This Project Covers:

In this analysis, we explore the SpaceX Falcon 9 launch dataset. It includes details such as launch sites, payload mass, mission outcome, booster information, and landing results.
This EDA focuses on:

- Getting an overview of the dataset
- Cleaning and preparing data
- Summarizing statistics
- Visualizing important features
- Extracting insights about launch behavior and mission success

About the Dataset:

SpaceX is a major private space organization known for reusable rocket technology. This dataset contains information about Falcon 9 launches including:

- Date and location of launch
- Payload characteristics
- Orbit categories
- Rocket booster versions
- Mission results
- Landing outcomes

Falcon 9 is widely used for cargo delivery, satellite deployment, and ISS missions.

Falcon 9:

Falcon 9 is a partially reusable, medium-lift launch vehicle developed by SpaceX.
It is capable of:

- Delivering cargo and crew into Earth orbit
- Performing missions to the International Space Station (ISS)
- Executing commercial satellite launches
  Its first stage booster is designed to return and land, enabling reuse and lowering launch costs.

1. import Libraries

We use Python libraries like:

- **Pandas** → to load and manipulate data
- **NumPy** → for numerical operations
- **Matplotlib & Seaborn** → for visualization

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Load the SpaceX Dataset:

```
df = pd.read_csv("https://raw.githubusercontent.com/chuksoo/IBM-Data-Science-Capstone-SpaceX/refs/heads/main/SpaceX.csv")
```

```
1 df.head()
```

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	NaN	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0.0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)	Success	No attempt
3	2012-08-10	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	None	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	No attempt

The dataset is loaded using read_csv() to begin exploring its shape, rows, and columns.

Dataset Overview

We inspect:

- Number of entries
- Column names
- Missing values
- Datatypes

This step helps identify what needs cleaning.

Shape and column info:

```
1 print("Shape:", df.shape)
2 print("\nColumns names:\n", df.columns.tolist())

Shape: (101, 10)

Columns names:
['Date', 'Time (UTC)', 'Booster_Version', 'Launch_Site', 'Payload', 'PAYLOAD_MASS__KG_', 'Orbit', 'Customer', 'Mission_Outcome', 'Landing _Outcome']
```

info about datatypes and null values:

Displays column names, data types, and number of non-missing values.
Useful to understand which columns need cleaning

```
1 df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101 entries, 0 to 100
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Date               101 non-null    object
 1   Time (UTC)         101 non-null    object
 2   Booster_Version    101 non-null    object
 3   Launch_Site        101 non-null    object
 4   Payload            101 non-null    object
 5   PAYLOAD_MASS__KG_  97 non-null     float64
 6   Orbit              98 non-null     object
 7   Customer           101 non-null    object
 8   Mission_Outcome    98 non-null     object
 9   Landing _Outcome   98 non-null     object
dtypes: float64(1), object(9)
memory usage: 8.0+ KB
```

Summary statistics for numeric coiumns:

Gives summary statistics (count, mean, median, etc.) for numerical columns. Helps check ranges and detect unusual values.

```
1 df.describe()
```

	PAYLOAD_MASS__KG_
count	97.000000
mean	6314.164948
std	4816.464413
min	[illegible]
25%	2647.000000
50%	4696.000000
75%	9600.000000
max	15600.000000

Step 1: Count missing vaiues:

This identifies how many missing values each column contains so we can clean them later.

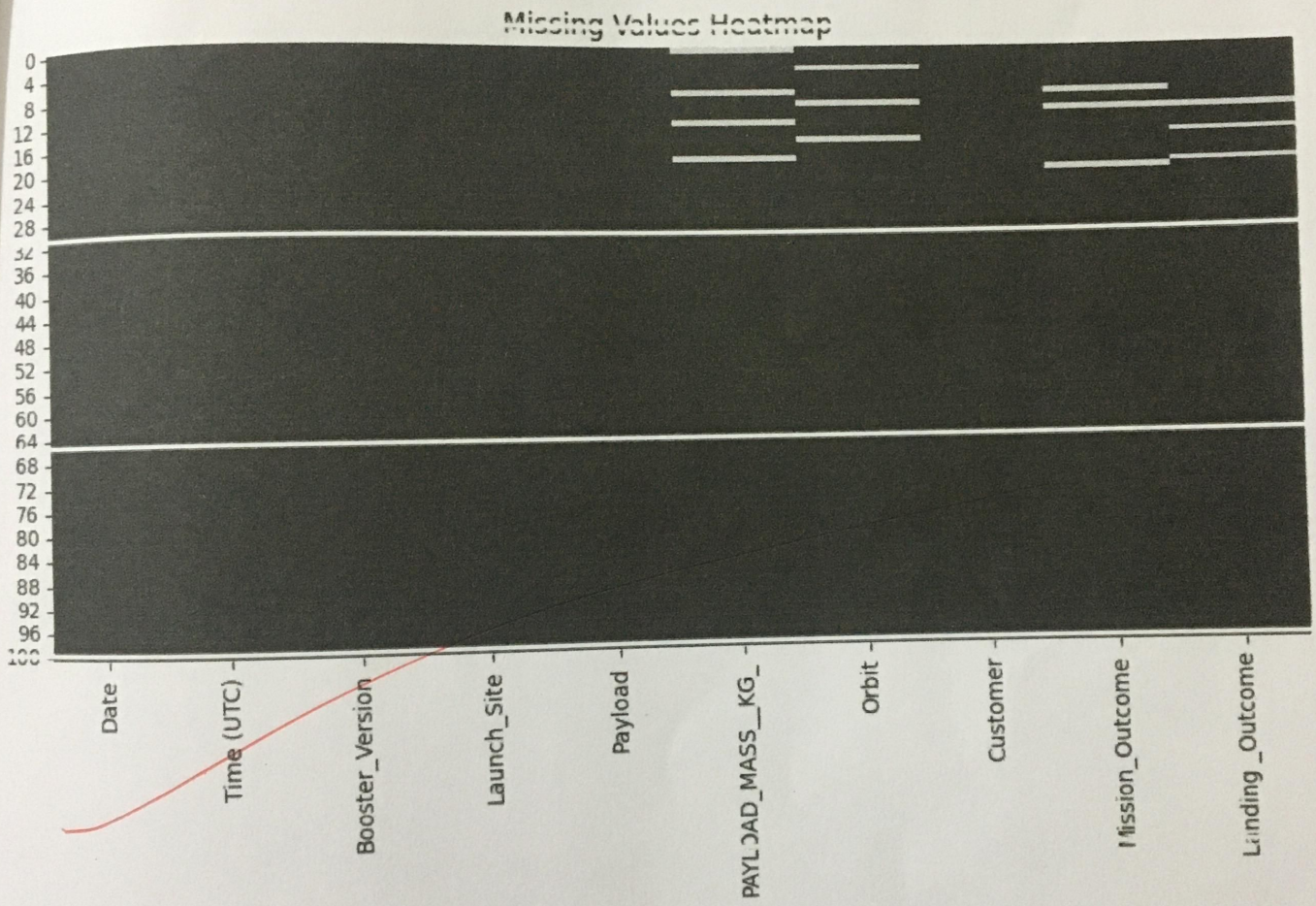
```
1 df.isnull().sum()
```

	0
Date	0
Time (UTC)	0
Booster_Version	0
Launch_Site	0
Payload	0
PAYLOAD_MASS__KG_	4
Orbit	3
Customer	0
Mission_Outcome	3
Landing _Outcome	3

dtype: int64

Step 2: View Missing Values with Heatmap:

```
plt.figure(figsize=(10,5))
sns.heatmap(df.isnull(), cbar=False, cmap='viridis')
plt.title("Missing Values Heatmap")
plt.show()
```



This heatmap shows where the dataset has missing values. The yellow lines indicate the columns that contain null entries. It helps us quickly identify which columns need cleaning before analysis.

Data Cleaning

Step 3: Handle Missing Values

- **Column: PAYLOAD_MASS__KG_**

```
df['PAYLOAD_MASS__KG_'] = df['PAYLOAD_MASS__KG_'].fillna(df['PAYLOAD_MASS__KG_'].median())
df.head()
```

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0.0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)	Success	No attempt
3	2012-08-10	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	No attempt

Step 4: Clean Categorical Missing Values

- **Column: Orbit**

```
df['Orbit'] = df['Orbit'].fillna(df['Orbit'].mode()[0])
df.head(10)
```

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0.0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)	Success	No attempt
3	2012-08-10	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	No attempt
5	2013-09-23	16:00:00	F9 v1.1 B1003	VAFB SLC-4E	CASSIOPE	500.0	Polar LEO	MDA	Success	Uncontrolled (ocean)
6	2013-03-12	22:41:00	F9 v1.1	CCAFS LC-40	SES-8	3170.0	GTO	SES	Success	No attempt
7	2014-06-01	22:06:00	F9 v1.1	CCAFS LC-40	Thaicom 6	4648.0	GTO	Thaicom	Success	No attempt
8	2014-04-18	19:25:00	F9 v1.1	CCAFS LC-40	SpaceX CRS-3	2296.0	LEO (ISS)	NASA (CRS)	Success	Controlled (ocean)
9	2014-07-14	15:15:00	F9 v1.1	CCAFS LC-40	OG2 Mission 1 6 Orbcomm-OG2 satellites	1316.0	GTO	Orbcomm	Success	Controlled (ocean)

- **Column: Mission_Outcome**

```
df['Mission_Outcome'] = df['Mission_Outcome'].fillna(df['Mission_Outcome'].mode()[0])
df.head(11)
```

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0.0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)	Success	No attempt
3	2012-08-10	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	No attempt
5	2013-09-23	16:00:00	F9 v1.1 B1003	VAFB SLC-4E	CASSIOPE	500.0	Polar LEO	MDA	Success	Uncontrolled (ocean)
6	2013-03-12	22:41:00	F9 v1.1	CCAFS LC-40	SES-8	3170.0	GTO	SES	Success	No attempt
7	2014-06-01	22:06:00	F9 v1.1	CCAFS LC-40	Thaicom 6	4648.0	GTO	Thaicom	Success	No attempt
8	2014-04-18	19:25:00	F9 v1.1	CCAFS LC-40	SpaceX CRS-3	2296.0	LEO (ISS)	NASA (CRS)	Success	Controlled (ocean)
9	2014-07-14	15:15:00	F9 v1.1	CCAFS LC-40	OG2 Mission 1 6 Orbcomm-OG2 satellites	1316.0	GTO	Orbcomm	Success	Controlled (ocean)
10	2014-05-08	8:00:00	F9 v1.1	CCAFS LC-40	AsiaSat 8	4535.0	GTO	AsiaSat	Success	No attempt

Step 5: Convert Datatypes:

- **Convert Payload Mass to integer**

```
df['PAYLOAD_MASS__KG_'] = df['PAYLOAD_MASS__KG_'].astype(int)
df.dtypes
```

	0
Date	datetime64[ns]
Time (UTC)	object
Booster_Version	object
Launch_Site	object
Payload	object
PAYLOAD_MASS__KG_	int64
Orbit	object
Customer	object
Mission_Outcome	object
Landing _Outcome	object

dtype: object

We convert the Payload Mass column into an integer datatype so that we can easily perform mathematical calculations and visual analysis without errors.

Changing the datatype ensures the values are clean, consistent, and ready for further analysis.

- **Check Again:**

```
df.isnull().sum()
```

	0
Date	0
Time (UTC)	0
Booster_Version	0
Launch_Site	0
Payload	0
PAYLOAD_MASS__KG_	0
Orbit	0
Customer	0
Mission_Outcome	0
Landing _Outcome	0

dtype: int64

This confirms that:

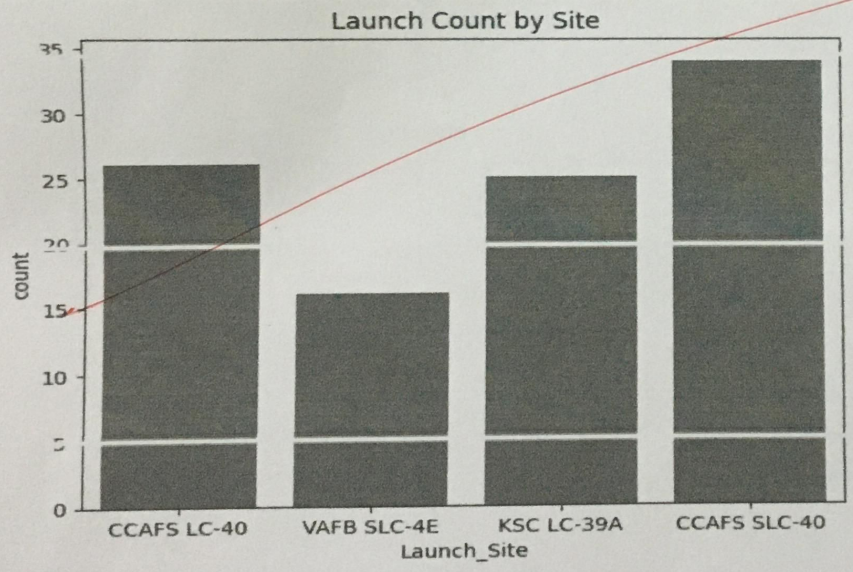
- All missing values are handled
- Dataset is clean

Univariate Data Visualization

Univariate plots show the pattern of a single column, helping us see its distribution and most common values.

- **Launch Site Count:**

```
sns.countplot(x='Launch_Site', data=df)
plt.title("Launch Count by Site")
plt.xticks()
plt.show()
```

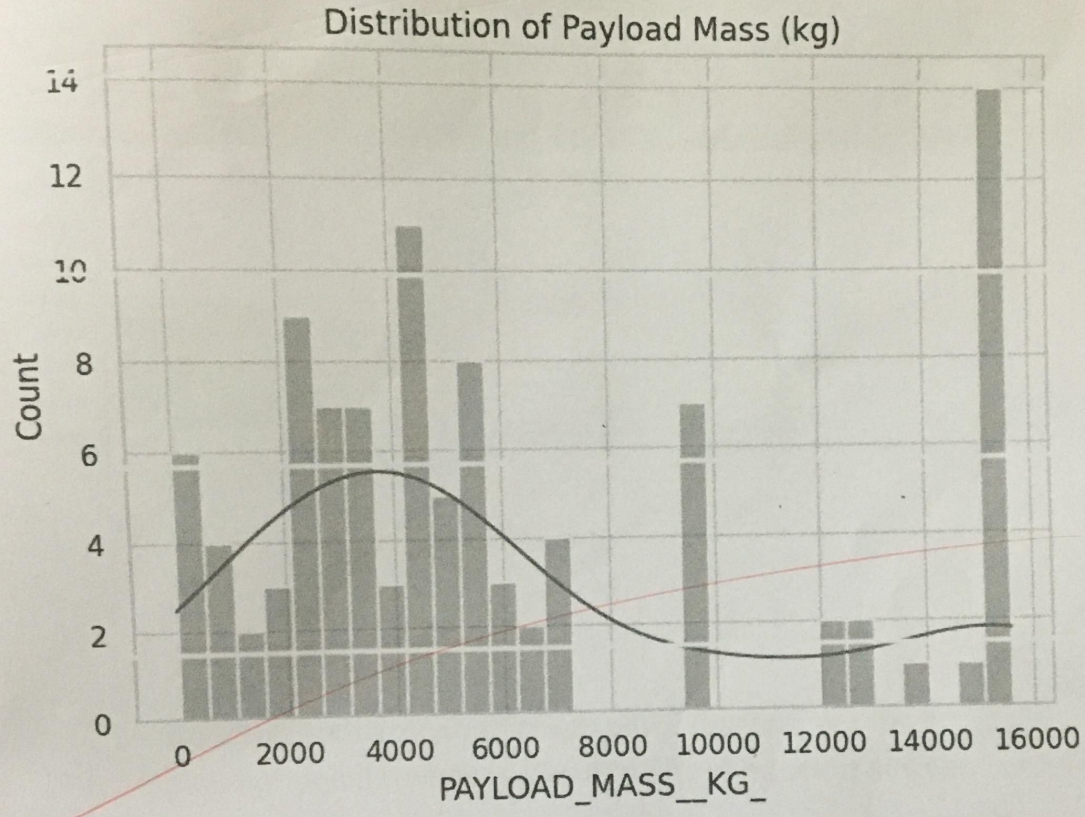


This chart shows how many launches happened at each SpaceX launch site.

It helps us see which location is used the most for Falcon 9 missions.

Payload Mass Distribution:

```
sns.set(style="whitegrid")
sns.histplot(df['PAYLOAD_MASS__KG_'], bins=30, kde=True)
plt.title("Distribution of Payload Mass (kg)")
plt.show()
```



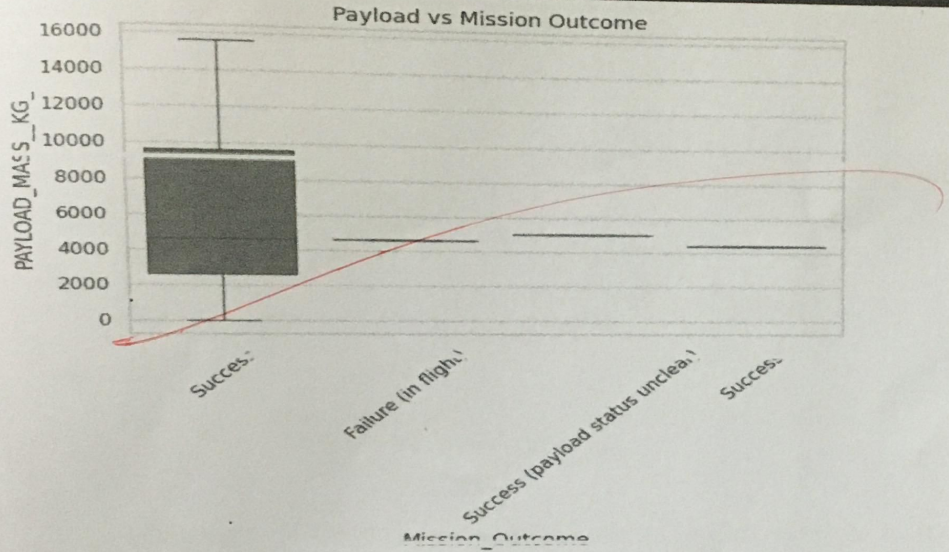
This graph shows how payload weights are spread across missions.
It helps us understand whether SpaceX launches mostly light, medium, or heavy payloads.

Bivariate Data Visualization

Bivariate plots show the relationship between two variables, helping us see how one feature affects another.

Payload vs Mission Outcome:

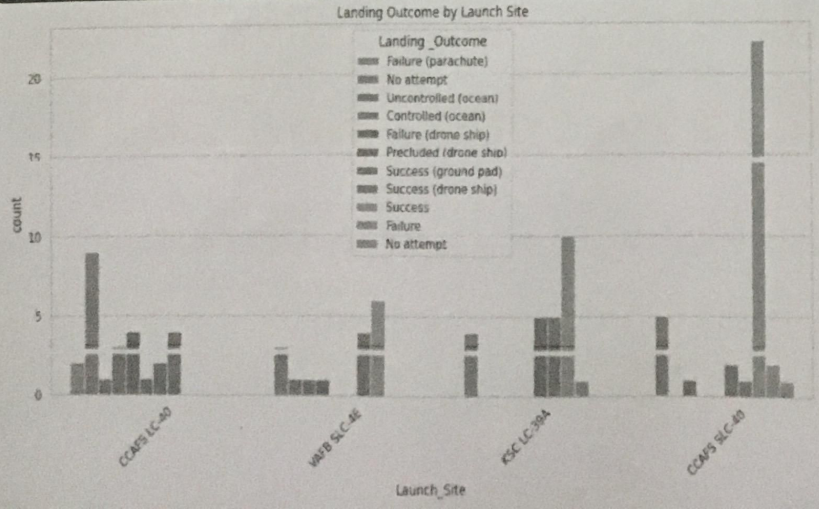
```
sns.boxplot(x='Mission_Outcome', y='PAYLOAD_MASS__KG_', data=df)
plt.title("Payload vs Mission Outcome")
plt.xticks(rotation=45)
plt.show()
```



This plot compares payload mass with mission results.
It helps us check if heavier payloads affect mission success or failure.

Launch Site vs Landing Outcome:

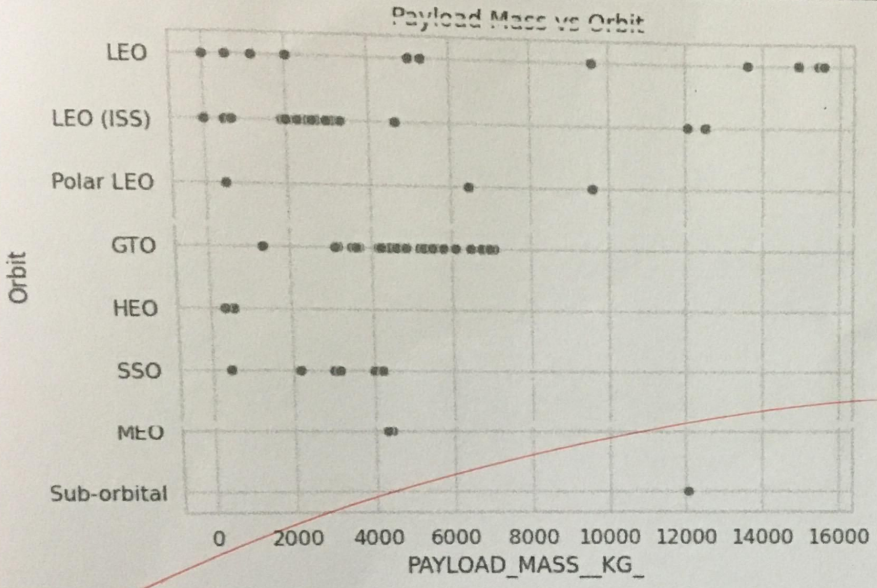
```
plt.figure(figsize=(14,6))
sns.countplot(x='Launch_Site', hue='Landing _Outcome', data=df)
plt.title("Landing Outcome by Launch Site")
plt.xticks(rotation=45)
plt.show()
```



This chart shows how landing results vary across different launch sites.
It helps identify which launch site has better landing performance.

Payload vs Orbit Type:

```
1 sns.scatterplot(x='PAYLOAD_MASS__KG_', y='Orbit', data=df)
2 plt.title("Payload Mass vs Orbit")
3 plt.show()
```



This graph shows how payload mass is distributed across different orbit types.
It tells us which orbits usually carry heavier or lighter payloads.

Select only numeric columns:

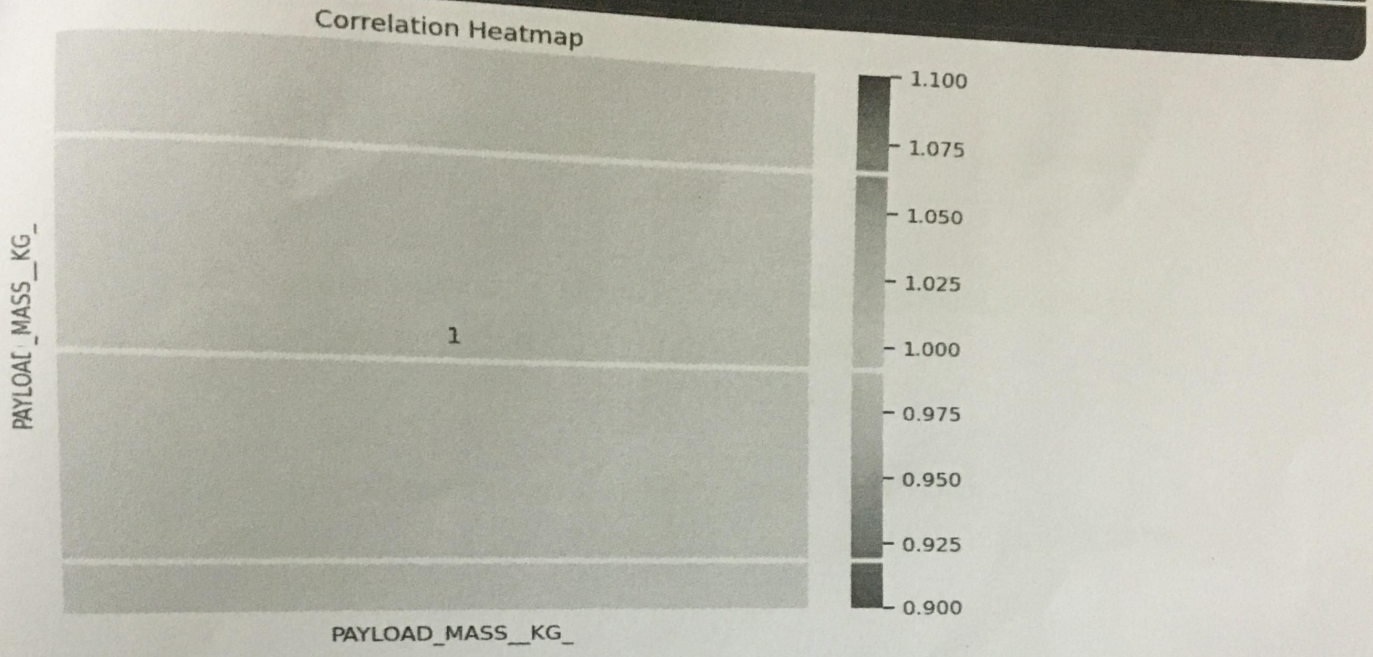
```
1 corr = df.corr(numeric_only=True)
2 corr
```

	PAYLOAD_MASS__KG_
PAYLOAD_MASS__KG_	1.0

We select only the numeric columns from the dataset to calculate correlation values.
Since the dataset has only one numeric feature (payload mass), the correlation matrix shows just one value.

Plot heatmap:

```
plt.figure(figsize=(8,6))
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```



The dataset has only **one numeric column**, which is PAYLOAD_MASS__KG_.
Because correlation works only between numeric columns, the heatmap shows just **one value: 1**.
A value of **1** appears because a column is always **100% correlated with itself**.

insights:

1. Most launches take place from a few major SpaceX launch sites.
Launch sites like KSC LC 39A and CCSFS SLC 40 appear the most, making them the primary launch locations.

2. The majority of missions in the dataset are successful.
Success outcomes clearly dominate over failures or partial failures.

3. Several missions show missing payload data, indicating incomplete recording or confidentiality.
Not every launch has a documented payload mass.

4. Payload mass varies widely across missions.
SpaceX handles everything from small satellite launches to heavy payload missions.

5. Earlier missions show more failures than recent ones.
This suggests that SpaceX has steadily improved its reliability over time.

6. Launch sites differ in their landing performance.
Certain locations show higher landing success, likely due to better conditions or recovery infrastructure.

7. Booster versions show a clear trend toward reusability.
Many booster types appear multiple times, indicating repeated use instead of single use launches.

8. Most customers belong to commercial and government sectors.
SpaceX works for global clients, not just NASA or US agencies.

9. Heavier payloads usually correspond to specific booster versions.
Advanced boosters are used for missions requiring more power.

10. Orbit types such as LEO and GTO are used frequently.
These are the preferred orbits for satellites, communication systems, and commercial payloads.

11. Landing outcomes vary by launch site and mission type.
Drone-ship landings are more common for high-payload missions, while lighter missions often return to land.

12. SpaceX launch activity increases over the years.
The dataset shows more frequent launches in recent years, reflecting the company's rapid growth.

Conclusion

This **EDA** gave us a clear understanding of the key factors related to **SpaceX** launch performance, such as **payload mass**, **orbit type**, **mission outcomes**, and landing results.

In real-world data science projects, such analysis helps identify important patterns, supports better decision-making, and guides which features should be used for future predictive models.

25/11/25