Exploratory Data Analysis

Introduction

The primary goal of **Exploratory Data Analysis (EDA)** is to gain a comprehensive understanding of the data, its quality, identify any underlying trends or relationships that may influence the analysis and modeling process.

Importance:

- EDA helps detect outliers, and inconsistencies in the data, addressing these issues ensures that the analysis is based on accurate and clean data.
- Through visualizations and statistical analysis, EDA identifies which features are most relevant to the questions and should be included in the model.
- EDA reveals relationships between variables, such as correlations, and trends, which can help discard certain features or create new derived features.
- By analyzing the distribution of variables, we can identify central tendencies and variations. This helps us understand how the data is spread out and it it meets assumptions required by certain algorithms.

Techniques:

- Univariate Analysis:
  - To understand the distribution, and central tendency of a variable.
  - Visualizations: Histograms, Box Plots, kernel Density plots.
  - Statistical measures: Mean, Median, Mode, Variance, Standard Deviation.
- Bivariate Analysis:
  - To understand the relationship between two variables
  - Categorical-Categorical: Heatmaps.
  - Numerical-Categorical: Box plots, Bar Graphs.
  - Numerical-Numerical: Scatter plots, correlation coefficients.
- Multivariate Analysis:
  - To explore relationships between three or more variables simultaneously.
  - Visualizations: Heatmaps, Scatter plots.
  - Dimensionality Reduction: PCA, t-SNE.
- Data Distribution:
  - Histograms and density plots are useful to detect skewness or multimodal distributions.
- Statistical Analysis:

- Hypothesis Testing: t-tests, chi-square tests, ANOVA.
- Correlation: Pearson, Spearman correlation coefficients
- Summary Statistics: Mean, variance, Quartiles.

In this project:
There are various datasets, and mutiple independent features such as: Driver Performance, Pit stop durations, race track features, overall performance of the car, which play an important role in determining the outcome of the race. Therefore, EDA is crucial to understand ow these factors influence race results.

Code

Provide the source code used for this section of the project here.

If you're using a package for code organization, you can import it at this point. However, make sure that the **actual workflow steps**—including data processing, analysis, and other key tasks—are conducted and clearly demonstrated on this page. The goal is to show the technical flow of your project, highlighting how the code is executed to achieve your results.

If relevant, link to additional documentation or external references that explain any complex components. This section should give readers a clear view of how the project is implemented from a technical perspective.

Remember, this page is a technical narrative, NOT just a notebook with a collection of code cells, include in-line Prose, to describe what is going on.

Required Libraries

```python
# import required libraries
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
import json
import os
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.lines as mlines
from matplotlib.cm import get_cmap
import seaborn as sns
from scipy.stats import ttest_ind
```

News Data[1]

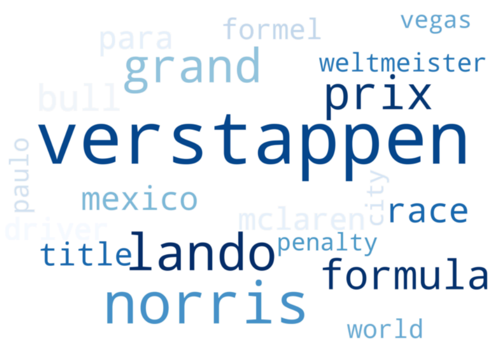
In this section, we will analyze media coverage of select F1 drivers who created significant buzz during the 2024 season. The drivers include **Max Verstappen**, **Carlos Sainz**, **Lando Norris**, **Daniel Riccardo**, and **Lewis Hamilton**. These drivers have been at the center of media discussions due to their performance, contract negotiations, and potential moves for the 2025 season.

```
# Function to generate a word cloud for a single JSON file
def generate_wordcloud_top_n(file_path, output_folder, top_n=10):
    # helper function to plot and save the word cloud
    def plot_cloud(wordcloud, output_file):
        plt.figure(figsize=(10, 8))
        plt.imshow(wordcloud, interpolation='bilinear')
        plt.axis("off")
        plt.savefig(output_file)

        plt.close()

    # read the text from the JSON file
    with open(file_path, 'r', encoding='utf-8') as file:
        data = json.load(file)

    # concatenate all text content if it's a JSON dictionary
    if isinstance(data, dict):
        my_text = ' '.join(data.values())

    # generate word cloud to extract word frequencies
    wordcloud = WordCloud(
        width=3000,
        height=2000,
        random_state=1,
        background_color="white",
        colormap="Blues_r",
        collocations=False,
        stopwords=STOPWORDS
    ).generate(my_text)

    # extract word frequencies and get the top N words
    word_frequencies = wordcloud.words_
    top_words = dict(list(word_frequencies.items())[:top_n])

    # generate a new word cloud with only the top N words
    top_wordcloud = WordCloud(
        width=3000,
        height=2000,
        random_state=1,
        background_color="white",
        colormap="Blues_r",
        collocations=False,
        stopwords=STOPWORDS
    ).generate_from_frequencies(top_words)

    # Save and display the word cloud
    file_name = os.path.basename(file_path).replace('.json', '_word
    output_file = os.path.join(output_folder, file_name)
    plot_cloud(top_wordcloud, output_file)


# function to process all JSON files in a folder
def generate_wordclouds_for_folder(input_folder, output_folder, top_
    os.makedirs(output_folder, exist_ok=True)
    # loop through all files in the input folder
    for file_name in os.listdir(input_folder):
        # process only JSON files
        if file_name.endswith('.json'):
            file_path = os.path.join(input_folder, file_name)
            try:
                generate_wordcloud_top_n(file_path, output_folder, 
            # for debugging purpose
            except Exception as e:
                print(f"Error processing file {file_path}: {e}")
```

```
input_folder = "../../data/processed-data/News_Drivers/"
output_folder = "../../data/eda/WordClouds/"
generate_wordclouds_for_folder(input_folder, output_folder, top_n=2
```

Analysis

Max Verstappen

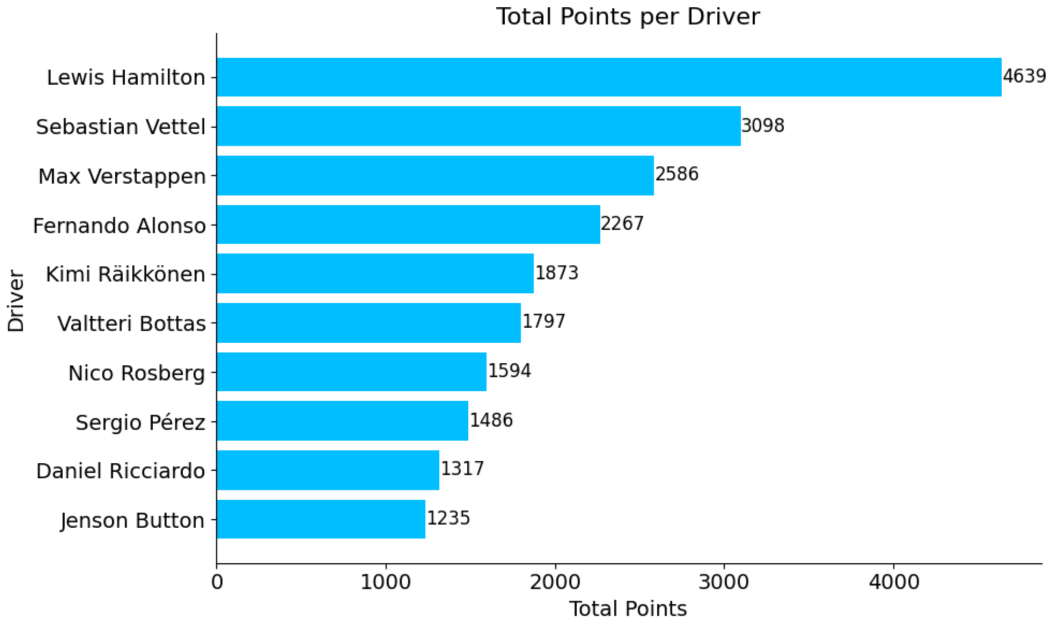


- Strong occurence of “Norris” suggests the media discussion about the rivalry and cmpetition between the top two drivers.
- The presence of “penalty” indicates that articles may have covered controversies or penalties during races that involved Verstappen. This adds to the narrative of the challenges and incidents faced during his title run.
- Words such as “title” and “world champion” reinforce the focus on Max Verstappen’s achievements, particularly his dominance throughout the season and his consecutive championship wins.

Drivers’ Performance

```
# number of points by driver
data = pd.read_csv("../../data/processed-data/driver_standings_2000_
# Calculate total points for each driver
total_points = data.groupby("driverName")["Points"].sum().reset_inde

# plot only top 10 drivers
top_10_drivers = total_points.sort_values(by="Points", ascending=Fa

# Plot the bar chart
plt.figure(figsize=(10, 6))
bars = plt.barh(top_10_drivers["driverName"], top_10_drivers["Point

for bar in bars:
    plt.text(
        bar.get_width() + 1,
        bar.get_y() + bar.get_height() / 2,
        f'{int(bar.get_width())}',
        va='center', fontsize=12, color='black'
    )
```

```
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)
plt.xlabel("Total Points", fontsize=14)
plt.ylabel("Driver", fontsize=14)
plt.title("Total Points per Driver", fontsize=16)
plt.gca().invert_yaxis()
plt.tight_layout()

plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)

plt.show()
```

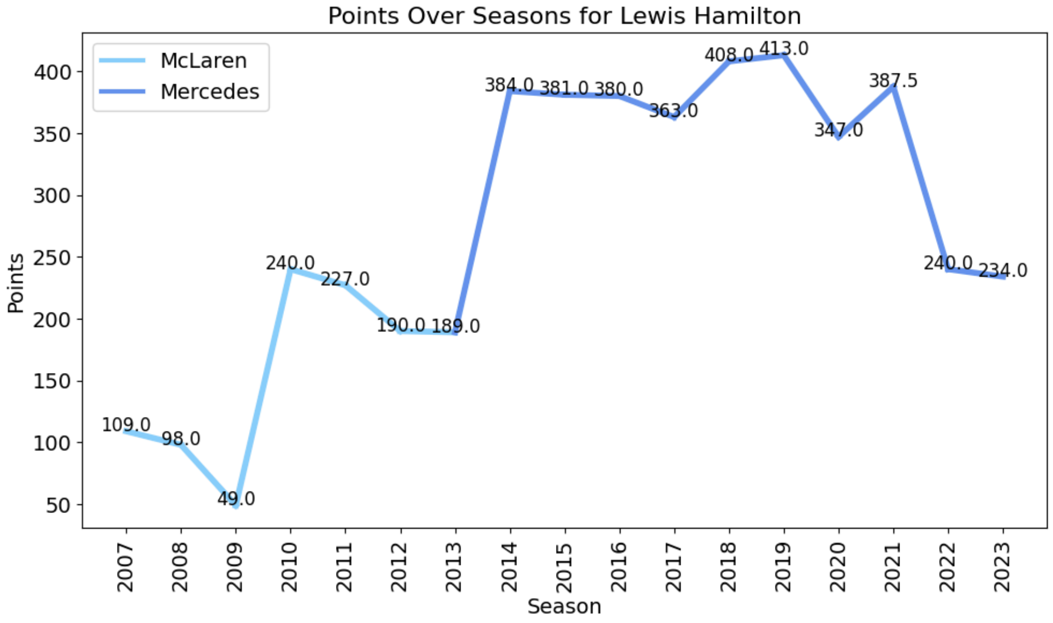


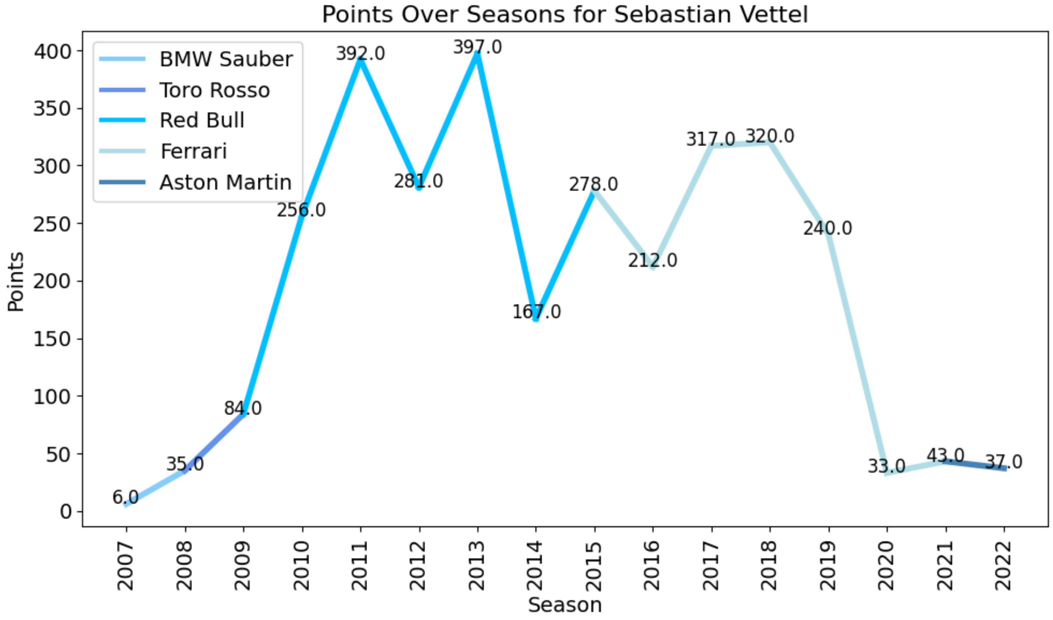
Insights:

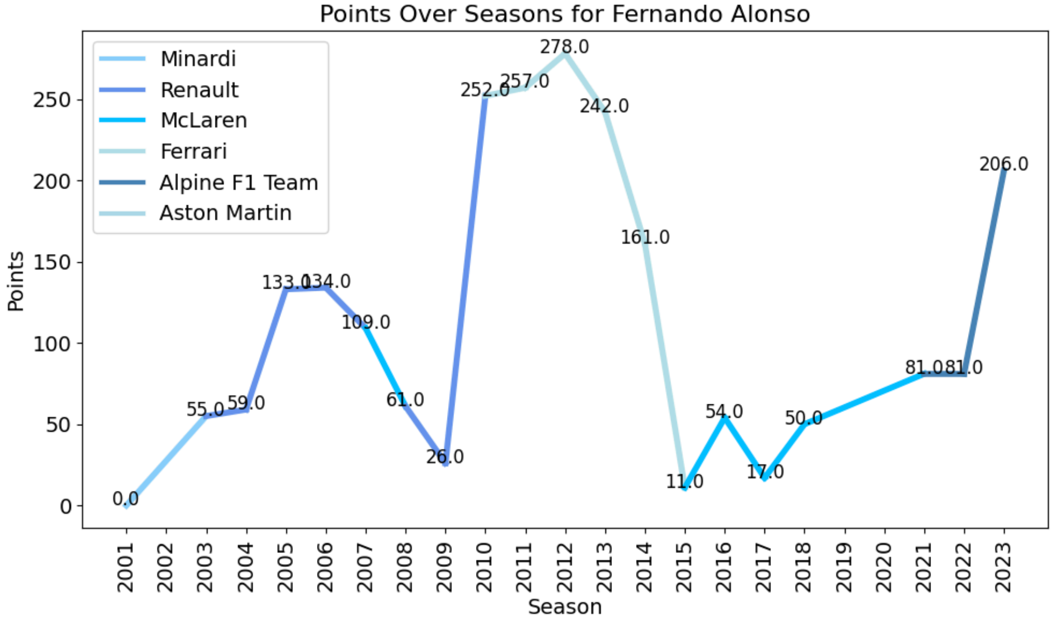
- The cumulative points reflect a driver's career longevity, performance, and overall success in the psport
- **Lewis Hamilton** stands out as the clear leader with 4,630 points, significantly ahead of other drivers. This highlights his dominance in the sport over the years, attributed to his consistent race wins, and career with top teams like McLaren and Mercedes in their eras.
- **Sebastian Vettel** holds the second spot with 3,098 points, his performance in Red Bull and Ferrari might have significant contribution to these numbers, since his early years in Sauber, and Toro Rosso, and his performance in Aston Martin were not as successful.
- **Max Verstappen** has the third highest cumulative total despite entering the sport relatively later. His rapid accumulation of points in the recent years underscores his dominance in the current era of F1.

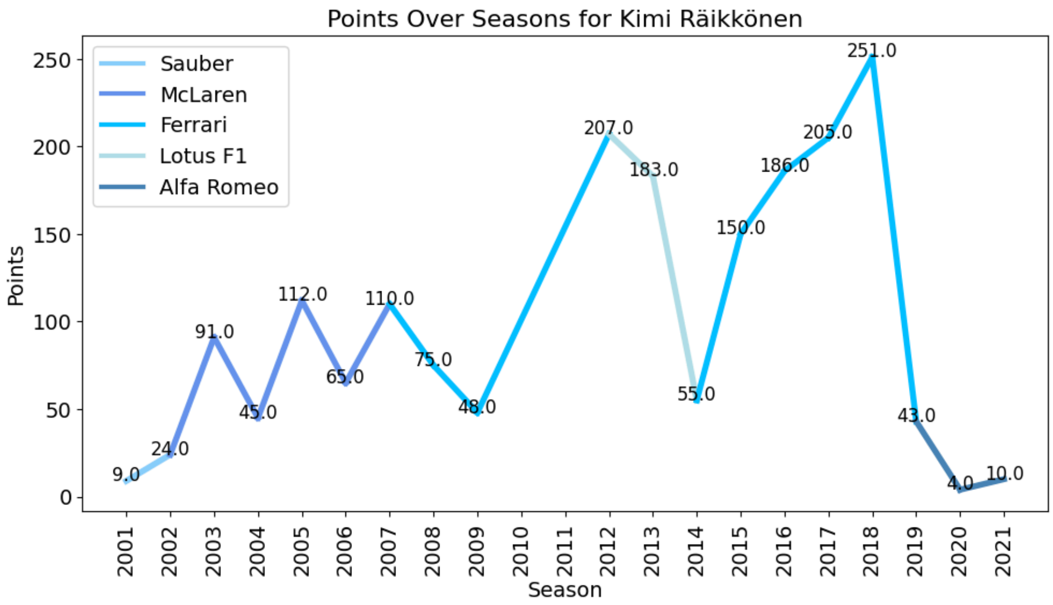
```
# points scored by driver over time with different teams
def plot_driver_points(data, driver_name, color_list=None):

    # filter the data to include only the specified driver
    # sort by season
    driver_data = data[data['driverName'] == driver_name].sort_value
    # list of unique constructors the driver raced for
    unique_constructors = driver_data["Constructor_Name"].unique()
    # assign colors to constructors
    if color_list:
        constructor_colors = {
            constructor: color_list[i % len(color_list)]
            for i, constructor in enumerate(unique_constructors)
        }
    else:
        constructor_colors = {
            constructor: plt.cm.tab10(i / len(unique_constructors))
            for i, constructor in enumerate(unique_constructors)
        }
    plt.figure(figsize=(10, 6))

    # line plot
    for i in range(len(driver_data) - 1):
        season_start = driver_data.iloc[i]['Season']
        season_end = driver_data.iloc[i + 1]['Season']
        points_start = driver_data.iloc[i]['Points']
        points_end = driver_data.iloc[i + 1]['Points']
        constructor = driver_data.iloc[i]['Constructor_Name']


        plt.plot(
            [season_start, season_end],
            [points_start, points_end],
            color=constructor_colors[constructor],
            linewidth=4
        )


    for _, row in driver_data.iterrows():
        plt.text(row['Season'], row['Points'], str(row['Points']), 


    plt.title(f"Points Over Seasons for {driver_name}", fontsize=16
    plt.xticks(fontsize = 14, rotation = 90)
    plt.yticks(fontsize = 14)
    plt.xlabel("Season", fontsize=14)
    plt.ylabel("Points", fontsize=14)
    plt.tight_layout()
    plt.xticks(range(int(driver_data["Season"].min()), int(driver_da


    legend_elements = [
        mlines.Line2D([], [], color=constructor_colors[constructor],
        for constructor in unique_constructors
    ]
    plt.legend(handles=legend_elements, loc="upper left", fontsize=


    plt.show()

color_list = ["lightskyblue", "cornflowerblue", "deepskyblue", "pow
```

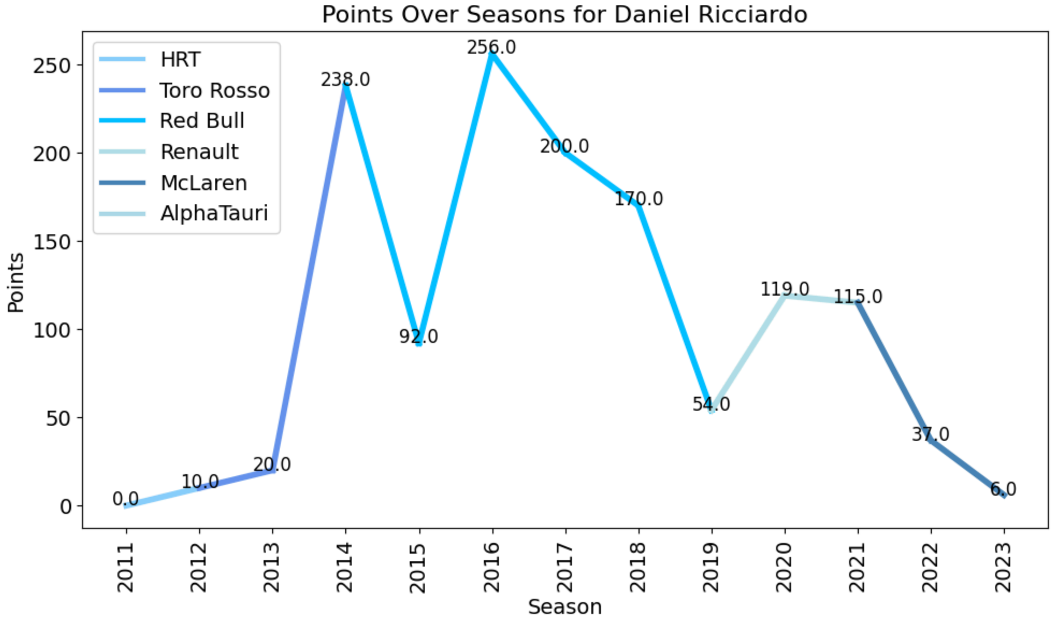
```
plot_driver_points(data, "Lewis Hamilton", color_list)
```

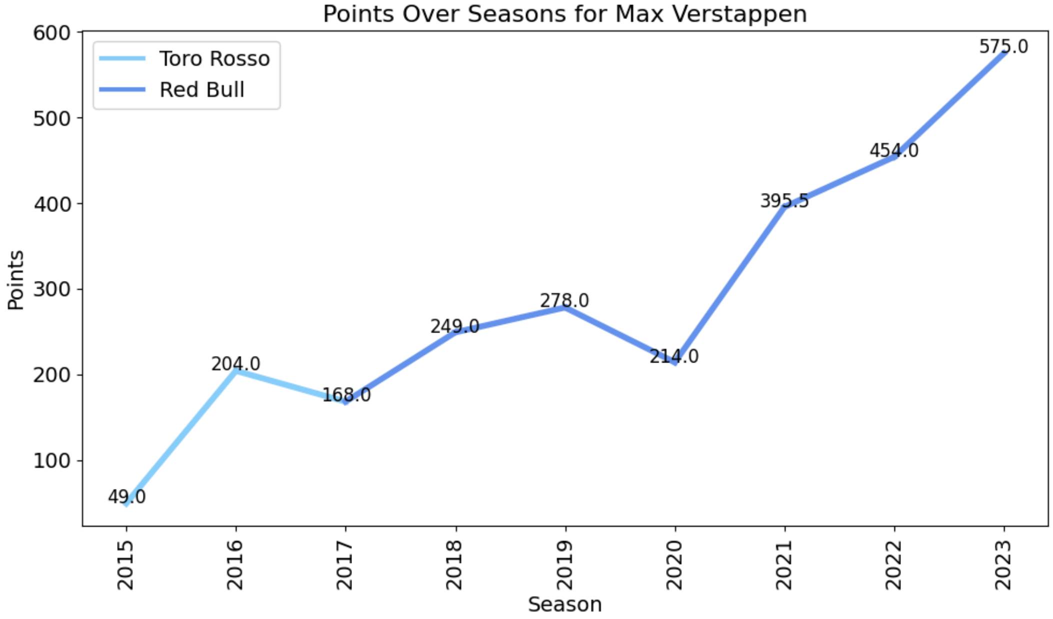












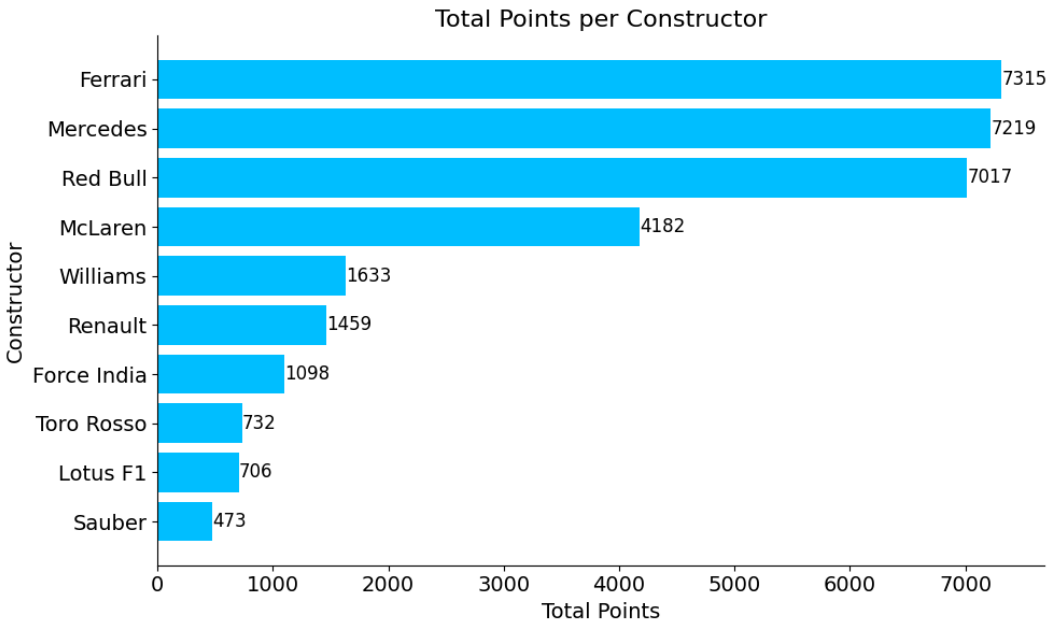
Insights:

- A significant portion of the total championship points for most drivers has been earned while driving for top-performing teams such as Ferrari and Red Bull.

- There is a clear correlation between the points scored in a season and the team a driver represented, emphasizing the importance of team performance in a driver’s success.
- Observing the career trajectories of drivers like Daniel Riccardo, Kimi Räikkönen, and Sebastian Vettel, we notice a pattern where they initially drove for top teams, followed by a move to midfield teams before retiring.
- Some drivers switched teams despite scoring a substantial number of championship points in the previous season. This highlights the strategic decisions made by teams in planning their driver line-ups and the drivers’ foresights regarding the potential improvements and competitiveness of cars across the grid.

Constructors’ Performance

```
total_points = data.groupby("Constructor_Name")["Points"].sum().rese


constructors = total_points.sort_values(by="Points", ascending=False

# Plot the bar chart
plt.figure(figsize=(10, 6))
bars = plt.barh(constructors["Constructor_Name"], constructors["Poin

for bar in bars:
    plt.text(
        bar.get_width() + 1,
        bar.get_y() + bar.get_height() / 2,
        f'{int(bar.get_width())}',
        va='center', fontsize=12, color='black'
    )
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)
plt.xlabel("Total Points", fontsize=14)
plt.ylabel("Constructor", fontsize=14)
plt.title("Total Points per Constructor", fontsize=16)
plt.gca().invert_yaxis()
plt.tight_layout()

plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)

plt.show()
```

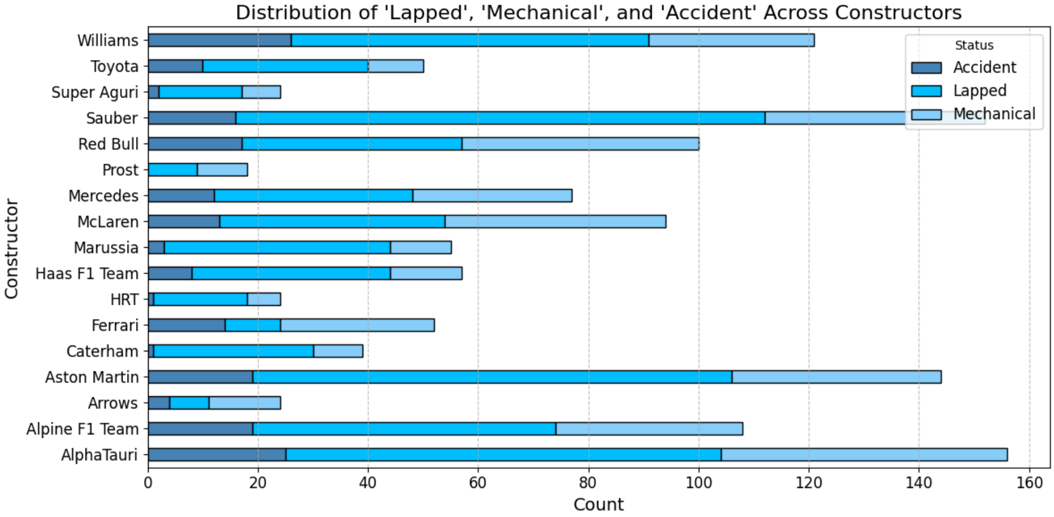


Insights:

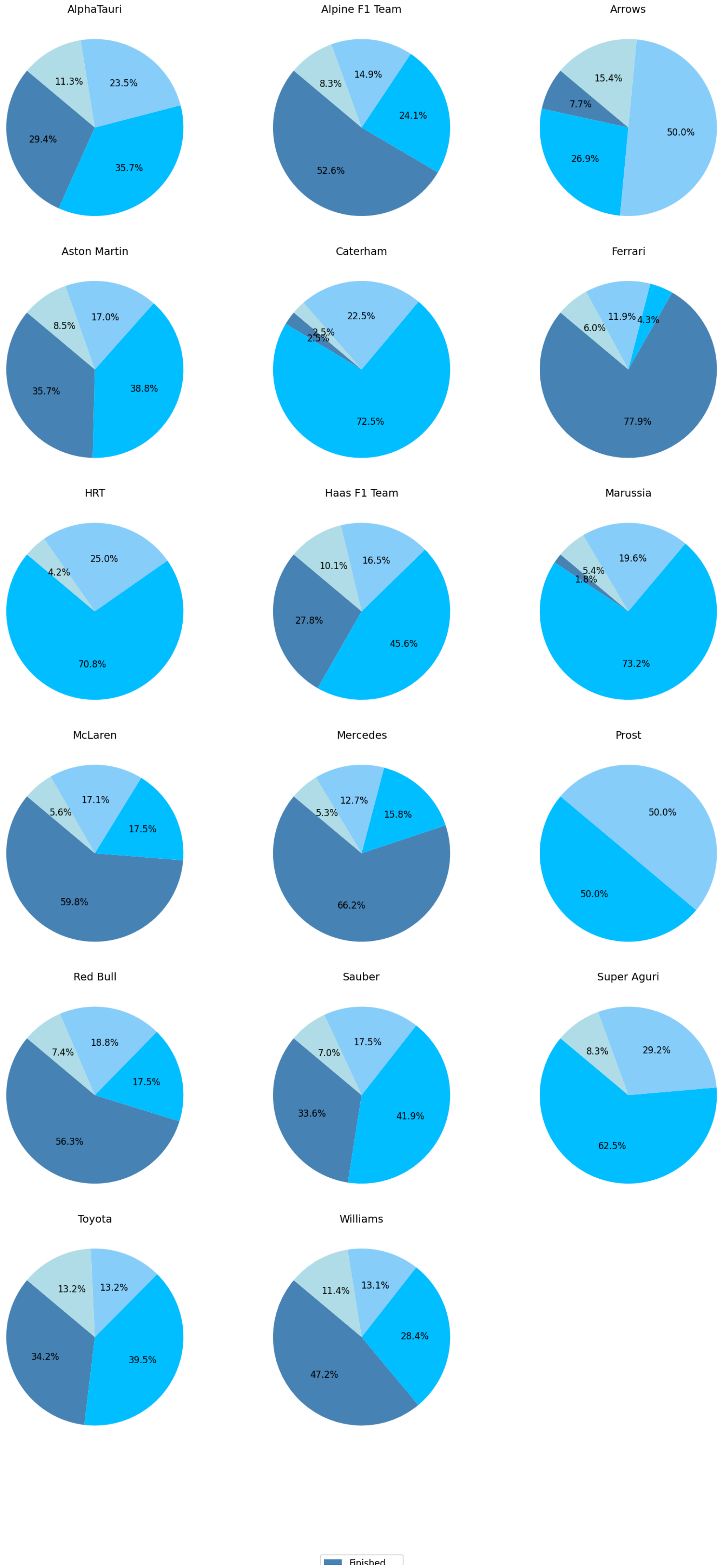
- The top 3 teams – Ferrari, Mercedes, and Red Bull – are closely positioned in the cumulative points standings.
- Mercedes' position is particularly notable because they re-entered the sports as a constructor in 2010, while Ferrari and Red Bull have accumulated points across the full 2000-2023 period. Despite the sorter timeframe, Mercedes managed to secure the second position, showcasing their unparalleled dominance 2014 onward.
- From domain knowledge, a key differentiating factor for Mercedes has been the consistent performance of their drivers, such as Lewis Hamilton, Nico Rosberg, and Valtteri Bottas, who regularly secured podium finishes during their tenure. This level of consistency has allowed Mercedes to close the gap with Ferrari and Red Bull in a relatively short period.
- For Ferrari and Red Bull, the points reflect their long-standing success and ability to remain competitive across different eras. Ferrari's consistent presence at the top, coupled with Red Bull's dominance during Sebastian Vettel's era (2010-2013) and Max Verstappen's current reign, has solidified their positions in the top three

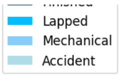
```
# distribution of "Lapped", "Mechanical", "Accident" across Construc
data = pd.read_csv("../../data/processed-data/race_info.csv")
```

```
# Filter the data for specific statuses
filtered_data = data[data["status"].isin(["Lapped", "Mechanical", "A

# Group by constructor and status to get counts
status_distribution = filtered_data.groupby(["constructorName", "sta

# Plot the distribution as a stacked horizontal bar chart
status_distribution.plot(
    kind="barh",
    figsize=(12, 6),
    color=['steelblue', 'deepskyblue', 'lightskyblue'],
    edgecolor="black",
    stacked=True
)

# Customize the plot
plt.title("Distribution of 'Lapped', 'Mechanical', and 'Accident' Ac
plt.xlabel("Count", fontsize=14)
plt.ylabel("Constructor", fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.legend(title="Status", fontsize=12, loc="upper right")
plt.grid(axis="x", linestyle="--", alpha=0.7)
plt.tight_layout()
plt.show()
```



```
# distrbution of status for each constructor
def autopct_format(pct):
    return f'{pct:.1f}%' if pct > 0 else ''

filtered_data = data[data["status"].isin(["Finished", "Lapped", "Mec
# Group by constructor and status to get counts
status_distribution = filtered_data.groupby(["constructorName", "sta
# total number of entries per constructor
status_distribution["Total"] = data.groupby("constructorName").size
# total number of constructors
num_constructors = len(status_distribution)
# for subplot
num_cols = 3
num_rows = (num_constructors + num_cols - 1) // num_cols
fig, axes = plt.subplots(num_rows, num_cols, figsize=(15, 5 * num_ro

axes = axes.flatten()

colors = ['steelblue', 'deepskyblue', 'lightskyblue', 'powderblue']
# loop through each constructor and create a pie chart
for idx, constructor in enumerate(status_distribution.index):
    # extract status counts for each constructor
    status_counts = status_distribution.loc[constructor, ["Finished"
    total_count = status_distribution.loc[constructor, "Total"]
    # calcuate percentage for each status
    percentages = (status_counts / total_count) * 100


    wedges, texts, autotexts = axes[idx].pie(
        percentages,
        labels=None,
        autopct=autopct_format,
        colors=colors,
        startangle=140,
        textprops={'fontsize': 14}
    )
    axes[idx].set_title(f"{constructor}", fontsize=14)

    for autotext in autotexts:
        autotext.set_fontsize(12)


for i in range(idx + 1, len(axes)):
    fig.delaxes(axes[i])

# Add a shared legend
fig.legend(
    labels=["Finished", "Lapped", "Mechanical", "Accident"],
    loc="lower center",
    ncol=1,
    fontsize=12
)

plt.tight_layout(rect=[0, 0.1, 1, 1])
plt.show()
```


AlphaTauri
11.3%
23.5%
29.4%
35.7%
Alpine F1 Team
8.3%
14.9%
24.1%
52.6%
Arrows
15.4%
7.7%
50.0%
26.9%
Aston Martin
8.5%
17.0%
35.7%
38.8%
Caterham
22.5%
2.5%
2.5%
72.5%
Ferrari
11.9%
6.0%
4.3%
77.9%
HRT
25.0%
4.2%
70.8%
Haas F1 Team
10.1%
16.5%
27.8%
45.6%
Marussia
19.6%
5.4%
1.8%
73.2%
McLaren
17.1%
5.6%
17.5%
59.8%
Mercedes
12.7%
5.3%
15.8%
66.2%
Prost
50.0%
50.0%
Red Bull
18.8%
7.4%
17.5%
56.3%
Sauber
17.5%
7.0%
33.6%
41.9%
Super Aguri
8.3%
29.2%
62.5%
Toyota
13.2%
13.2%
34.2%
39.5%
Williams
11.4%
13.1%
28.4%
47.2%
Finished

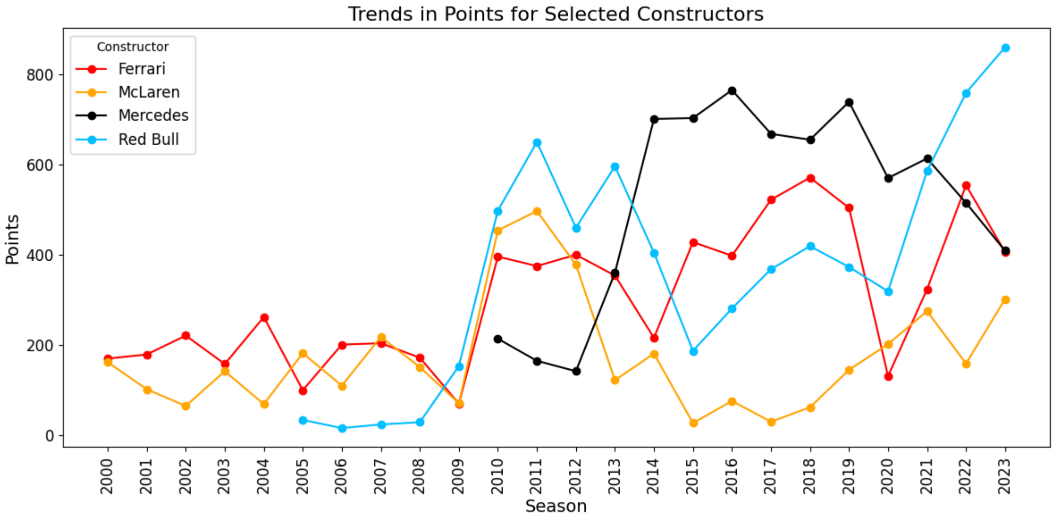


Insights:

- Ferrari stands out with an impressive 77.9% race completion rate. The low percentage of mechanical failures and lapped races further emphasize their ability to build durable and reliable cars. Mercedes also displays as a high completion rate (66.2%) and a low percentage of mechanical failures and accidents. McLaren is a close third with 59.8% finished races, although it has low percentage of accidents, it has comparatively high percentage of lapped and incomplete races due to mechanical failures out of the top 3.
- Williams, a team with a glorious past, shows 47.2% completion rate and 28.4% lapped races, highlighting the shift in their performance over the years. Having 11.4% accidents indicates that the drivers have not been performing well, 13.1% mechanical failures indicates struggles with car development, limited resources, and their current inability to compete with the top teams.
- Haas F1 Team a relatively new entrant, records 27.8% completion rate, with notable lapped races (45.6%) and mechanical failures. Their results reflect the challenges faced by newer teams with smaller budgets and sponsorships.
- Midfield and smaller teams face significant challenges in maintaining consistency due to resource limitations, budget caps, and the learning curve required to compete at the highest level. These teams often serve as development platforms for young drivers transitioning from F2 to F1 but struggle to compete with top constructors.

```
# trends in teams scoring points across seasons - for "Red Bull", "
data = pd.read_csv("../../data/processed-data/driver_standings_2000_

filtered_data = data[data["Constructor_Name"].isin(["Red Bull", "Fe

grouped_data = filtered_data.groupby(["Season", "Constructor_Name"]

color_list = ["red", "orange", "black", "deepskyblue"]

unique_constructors = grouped_data.columns
constructor_colors = {constructor: color_list[i % len(color_list)]


plt.figure(figsize=(12, 6))
for constructor in grouped_data.columns:
    plt.plot(
        grouped_data.index,
        grouped_data[constructor],
        marker='o',
        label=constructor,
        color=constructor_colors[constructor]
    )
```

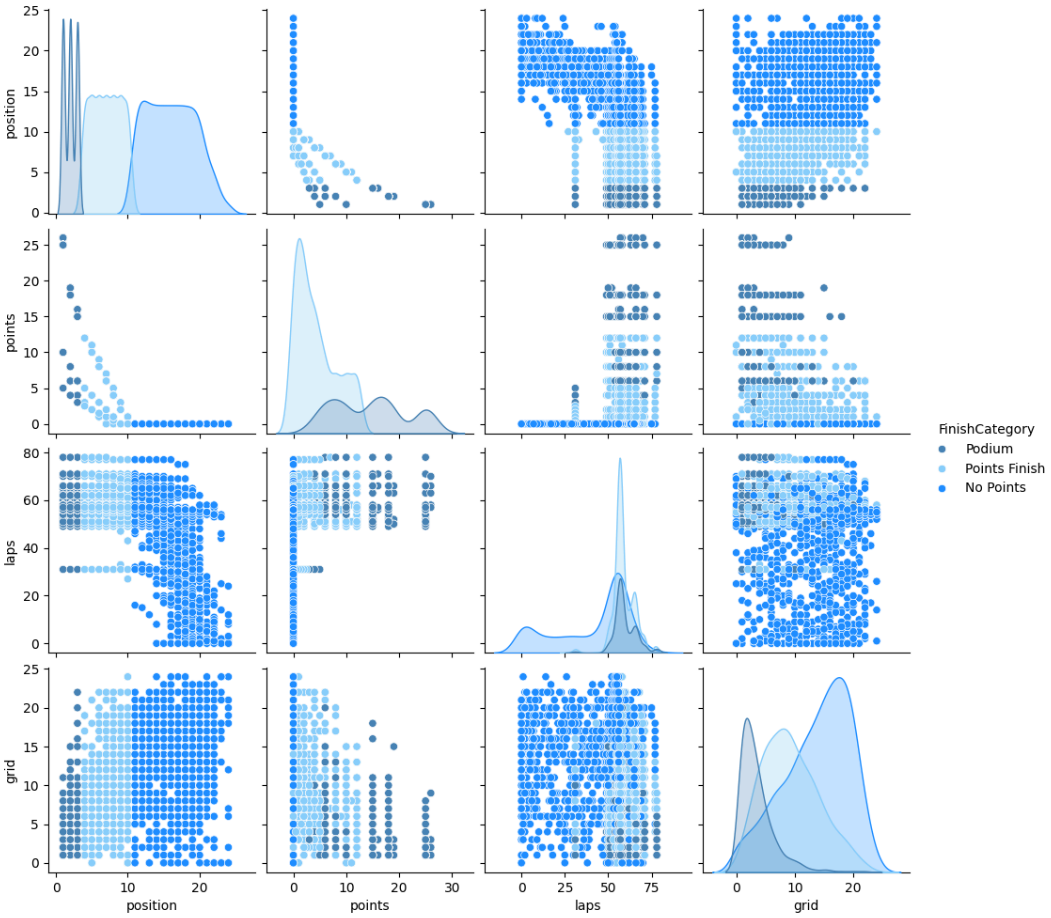
```
plt.title("Trends in Points for Selected Constructors", fontsize=16
plt.xlabel("Season", fontsize=14)
plt.ylabel("Points", fontsize=14)
plt.xticks(grouped_data.index, rotation=90, fontsize=12)
plt.yticks(fontsize=12)
plt.legend(title="Constructor", fontsize=12)
plt.tight_layout()


plt.show()
```



Insights:

- The plot highlights clear periods of highs and lows for each team, emphasizing the dynamic nature of Formula 1. Success in the sport is cyclical, with teams experiencing periods of dominance followed by rebuilding phases.
- Red Bull showed a rapid rise starting in 2009, coinciding with their dominance during the Sebastian Vettel era (2010–2013). After a slight dip between 2014–2020, they have experienced another significant upward trend, driven by Max Verstappen's dominance in recent years.
- Mercedes entered the sport as a constructor in 2010 and saw rapid progress starting in 2014 during the hybrid engine era, where they dominated for nearly a decade. Their decline starting in 2021 coincides with the resurgence of Red Bull and highlights challenges in adapting to regulation changes.
- Ferrari has experienced the most noticeable highs and lows among the teams. A steep decline is observed in 2020, likely due to car performance struggles, followed by a sharp recovery in 2022, marking their return to competitiveness.
- McLaren had strong results in the early 2000s but faced a significant decline after 2012, struggling during the hybrid era. Their gradual recovery post-2018, reflected in the upward trend, signals improvements in car performance and team structure.
- This analysis underscores that success in Formula 1 is not just about driver skill but also hinges on the ability of teams to innovate, adapt, and build competitive cars under constantly evolving technical regulations.

```
sns.pairplot(data[['position', 'points', 'laps', 'grid', 'FinishCate
```



Circuit Features

```
data = pd.read_csv("../../data/processed-data/race_track_features.cs
```

```
# Full Throttle (%) vs Number of Straights

#  grouping based on the median of 'Number of Straights'
median_straights = data['Number of Straights'].median()

# tracks with fewer straights than median
low_straights = data[data['Number of Straights'] < median_straights
# tracks with >= median straights
high_straights = data[data['Number of Straights'] >= median_straigh

# perform t-test
t_stat, p_value = ttest_ind(low_straights, high_straights, nan_polic

# resutls
print(f"T-Statistic: {t_stat:.4f}")
print(f"P-Value: {p_value:.4f}")
```

```
T-Statistic: 8.7778
P-Value: 0.0000
```

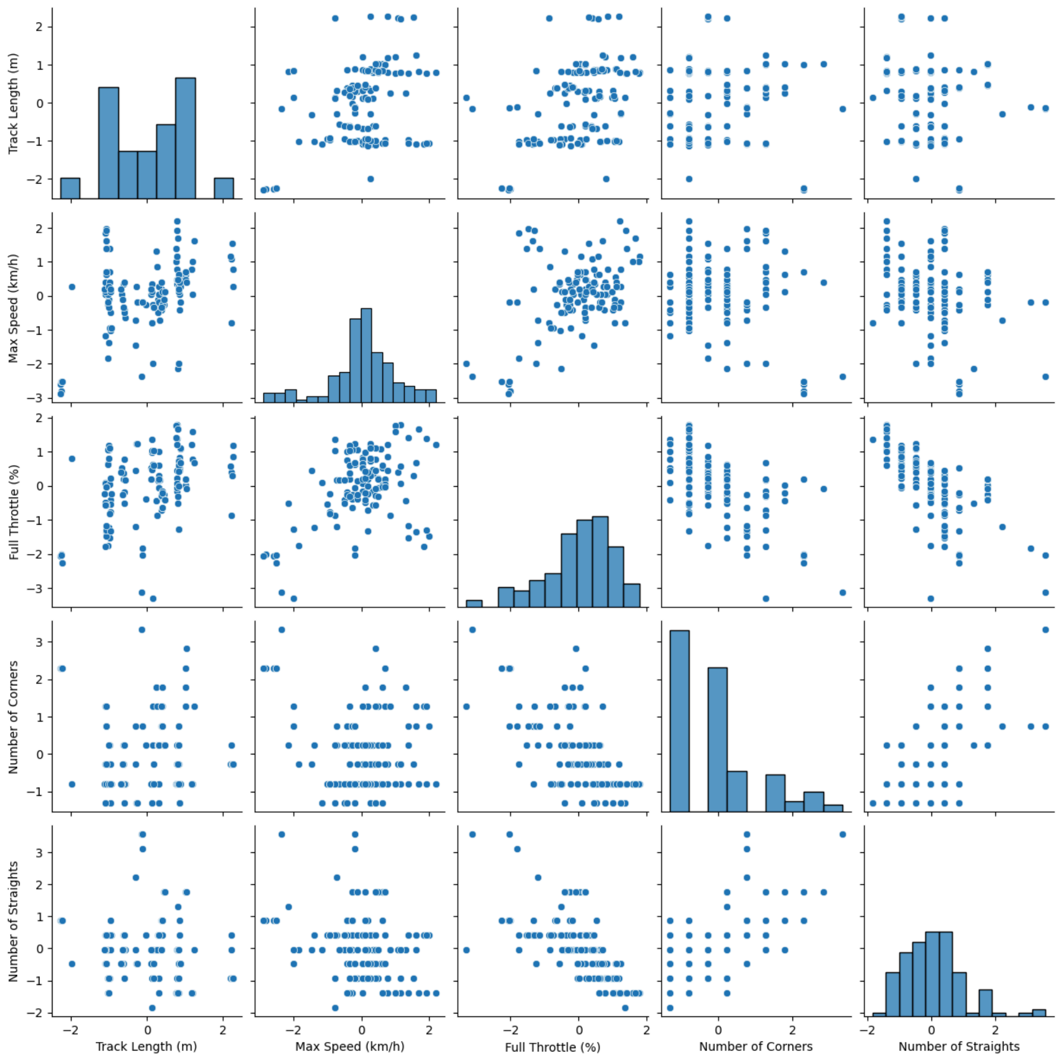
There is a significant difference in Full Throttle (%) between tracks with low and high straights.

```
# Full Throttle (%) vs Number of Corners

median_corners = data['Number of Corners'].median()

# tracks with lower number of corners than median
low_corners = data[data['Number of Corners'] < median_corners]['Ful
# tracks with >= median corners
high_corners = data[data['Number of Corners'] >= median_corners]['F

# perform t-test
t_stat, p_value = ttest_ind(low_corners, high_corners, nan_policy='

# results
print(f"T-Statistic: {t_stat:.4f}")
print(f"P-Value: {p_value:.4f}")
```

```
T-Statistic: 6.8204
P-Value: 0.0000
```

There is a significant difference in Full Throttle (%) between tracks with low and high corners.

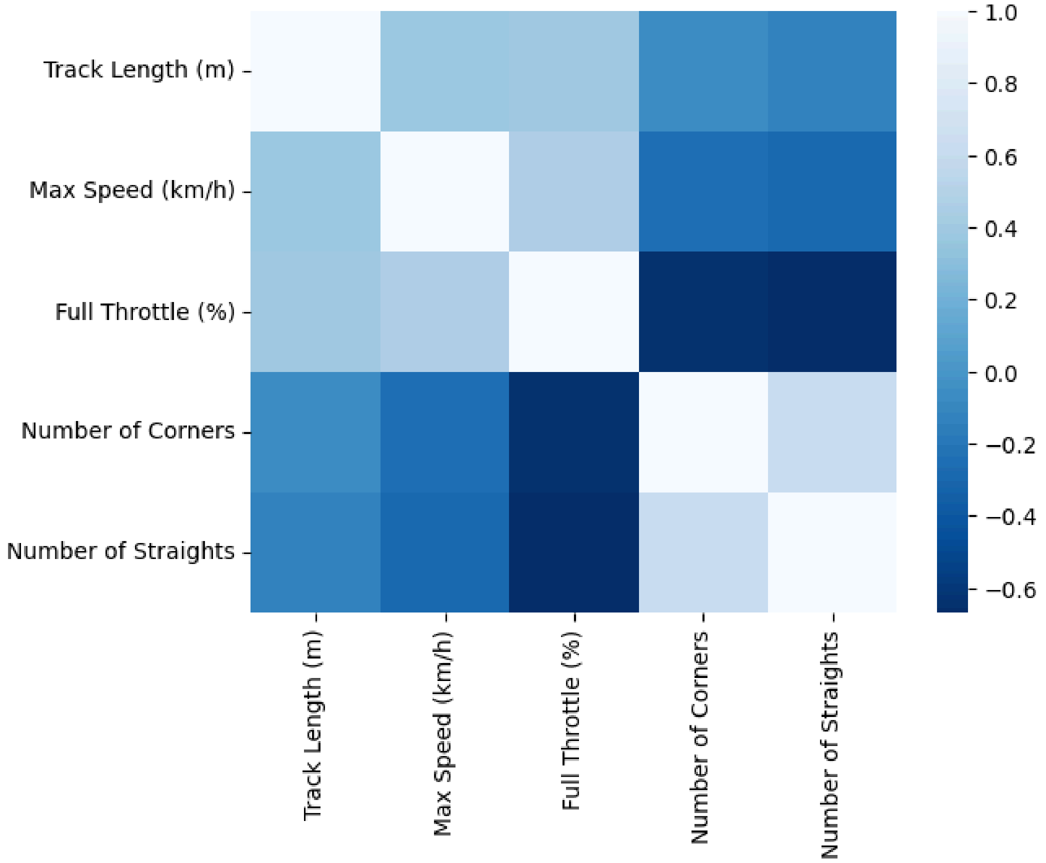
```
sns.pairplot(data[["Track Length (m)", "Max Speed (km/h)", "Full Th
plt.show()
```



```
corr_matrix = data[["Track Length (m)", "Max Speed (km/h)", "Full Th
corr_matrix
```

	Track Length (m)	Max Speed (km/h)	Full Throttle (%)	Number of Corners	Number of Straights
Track Length (m)	1.000000	0.373616	0.399017	-0.058163	-0.118008
Max Speed (km/h)	0.373616	1.000000	0.462422	-0.246056	-0.281559
Full Throttle (%)	0.399017	0.462422	1.000000	-0.654084	-0.667532
Number of Corners	-0.058163	-0.246056	-0.654084	1.000000	0.631757
Number of Straights	-0.118008	-0.281559	-0.667532	0.631757	1.000000

```
sns.heatmap(corr_matrix, cmap="Blues_r")
plt.show()
```



Pitstop Analysis

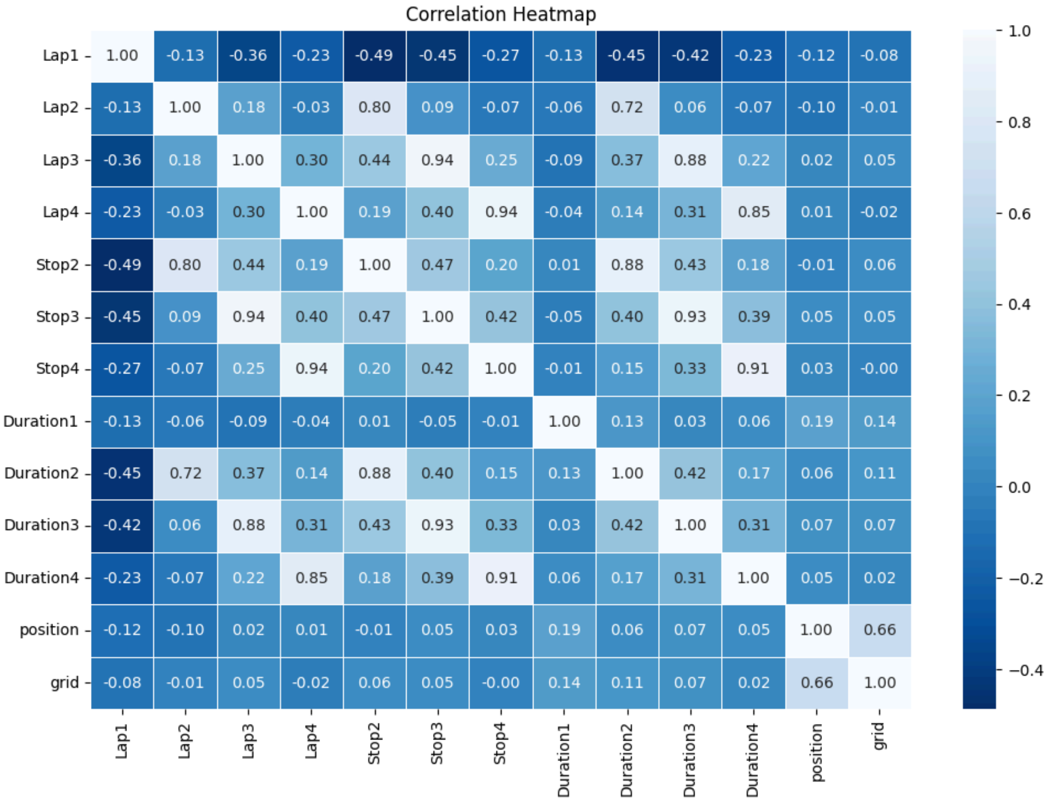
```
data = pd.read_csv("../../data/processed-data/pitstop_with_position
data.head()
```

	Unnamed: 0	Year	Round	RaceName	DriverID	Lap1	Lap2	Lap3	L
0	0	2011	1	Australian Grand Prix	alguersuari	-1.492522	-0.208201	1.339923	-
1	1	2011	1	Australian Grand Prix	alonso	-0.414215	0.355923	1.723412	-
2	2	2011	1	Australian Grand Prix	ambrosio	-0.218159	0.976458	-0.577525	-
3	3	2011	1	Australian Grand Prix	barrichello	-0.316187	0.130273	0.956433	3
4	4	2011	1	Australian Grand Prix	buemi	-0.120131	0.468747	-0.577525	-

5 rows × 36 columns

```
numerical_columns = ["Lap1", "Lap2", "Lap3", "Lap4", "Stop2", "Stop
                     "Duration1", "Duration2", "Duration3", "Duratio
correlation_data = data[numerical_columns]

# Calculate correlation matrix
correlation_matrix = correlation_data.corr()
```

```
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="Blues_r", fmt=".2
plt.title("Correlation Heatmap")
plt.show()
```



	Lap1	Lap2	Lap3	Lap4	Stop2	Stop3	Stop4	Duration1	Duration2	Duration3	Duration4	position	grid
Lap1	1.00	-0.13	-0.36	-0.23	-0.49	-0.45	-0.27	-0.13	-0.45	-0.42	-0.23	-0.12	-0.08
Lap2	-0.13	1.00	0.18	-0.03	0.80	0.09	-0.07	-0.06	0.72	0.06	-0.07	-0.10	-0.01
Lap3	-0.36	0.18	1.00	0.30	0.44	0.94	0.25	-0.09	0.37	0.88	0.22	0.02	0.05
Lap4	-0.23	-0.03	0.30	1.00	0.19	0.40	0.94	-0.04	0.14	0.31	0.85	0.01	-0.02
Stop2	-0.49	0.80	0.44	0.19	1.00	0.47	0.20	0.01	0.88	0.43	0.18	-0.01	0.06
Stop3	-0.45	0.09	0.94	0.40	0.47	1.00	0.42	-0.05	0.40	0.93	0.39	0.05	0.05
Stop4	-0.27	-0.07	0.25	0.94	0.20	0.42	1.00	-0.01	0.15	0.33	0.91	0.03	-0.00
Duration1	-0.13	-0.06	-0.09	-0.04	0.01	-0.05	-0.01	1.00	0.13	0.03	0.06	0.19	0.14
Duration2	-0.45	0.72	0.37	0.14	0.88	0.40	0.15	0.13	1.00	0.42	0.17	0.06	0.11
Duration3	-0.42	0.06	0.88	0.31	0.43	0.93	0.33	0.03	0.42	1.00	0.31	0.07	0.07
Duration4	-0.23	-0.07	0.22	0.85	0.18	0.39	0.91	0.06	0.17	0.31	1.00	0.05	0.02
position	-0.12	-0.10	0.02	0.01	-0.01	0.05	0.03	0.19	0.06	0.07	0.05	1.00	0.66
grid	-0.08	-0.01	0.05	-0.02	0.06	0.05	-0.00	0.14	0.11	0.07	0.02	0.66	1.00

Some of the cell outputs were exlcluded from the main page. They can be found here

References

1. DSAN 5000 lecture content, https://jfh.georgetown.domains/centralized-lecture-content/course-timelines/dsan-5000.html.