F1-DATA

Meschi Maurizio
5194399
Seasons - Teams

Naccarato Matteo
5330843
Circuits - Drivers

DATASET OVERVIEW

- All drivers
- All constructors
- All circuits and location data
- All seasons (1950 to present)
- All races and qualifying results

SEASONS

Number of GP over the years
- *line chart* -

```python
return px.line(df_count,
    x = "year",
    y = "value",
    markers = True,
    labels = labels_dict,
    color_discrete_sequence=f1db_utils.custom_colors,
    template = f1db_utils.template
).update_layout(
    f1db_utils.transparent_bg,
    title = f1db_utils.getTitleObj("Number of GP Over the Years"),
    hovermode = "x",
    margin=f1db_utils.margin
).update_traces(
    hoverlabel=f1db_utils.getHoverlabel(),
    hovertemplate="<br>".join(["<b>%{y}</b><extra></extra>"])
).update_yaxes(title_text='Number of GP')
```

Number of GP by country
- *scatter geo* -

```python
for index, value in df_count['grandPrixId'].items():
    countryId = df2.loc[df2['id'] == value, "countryId"].iloc[0]
    alpha3Code = df3.loc[df3['id'] == countryId, "alpha3Code"]
    continent = df3.loc[df3['id'] == countryId, "continentId"]
    if not alpha3Code.empty:
        if not continent.empty:
            alpha3Code = alpha3Code.iloc[0]
            continent = continent.iloc[0]
            df = df._append({
                'id': value,
                'number_gps': df_count.loc[df_count['grandPrixId'] == value, "value"].iloc[0],
                'continent': continent,
                'code': alpha3Code
            }, ignore_index=True)
```

WDC driver comparison
- *line chart* -

```python
title = "Positions" if radio_button_value == "positionNumber" else "Points"
fig = px.line(data_in_range,
    x="year",
    y=radio_button_value,
    color="driverName",
    labels = labels_dict,
    color_discrete_sequence=f1db_utils.custom_colors,
    template = f1db_utils.template,
    hover_data = { "driverName": True }
).update_layout(
    f1db_utils.transparent_bg,
    hovermode = "x",
    title = f1db_utils.getTitleObj(f"Drivers' {title} in WDCs"),
    margin=f1db_utils.margin
)
fig.update_xaxes(dtick=1, tickmode='linear')
if (radio_button_value == "positionNumber"):
    fig.update_traces(
        hoverlabel = f1db_utils.getHoverlabel(),
        hovertemplate="<br>".join(["%{customdata} <b>(%{y})</b><extra></extra>"]
```

CIRCUITS

Number of GP Held by Circuit
- bar chart -

Qualifying vs Race
- scatter plot -

Pole Lap Time Over the Years
- line chart -

```python
# UP-LEFT GRAPH
def get_gp_held(minValue):
    df = pd.read_csv(f"{f1db_utils.folder}/{f1db_utils.circuits}")
    df.sort_values(by=["totalRacesHeld", "name"], ascending=False, inplace=True)

    df_countries = pd.read_csv(f"{f1db_utils.folder}/{f1db_utils.countries}")
    df_countries.rename(columns={"id": "countryId", "name":"countryName"}, inplace=True)
    df_countries.drop(columns=df_countries.columns.difference(["countryId", "countryName", "alpha3Code"]), inplace=True)
    df = pd.merge(df, df_countries, on="countryId", how="left")

    df = df[df["totalRacesHeld"] >= minValue]
    df.reset_index(inplace=True)
    df.drop(columns=["index"], inplace=True)

    return df
```

```python
df['positionRace'] = df['positionRace'].replace(not_a_number_replace)
df['positionRace'] = pd.to_numeric(df['positionRace'], errors='coerce')
df["positionRace"] = df["positionRace"].fillna(f1db_utils.QUALI_FILL_NA)
df['positionRace'] = df['positionRace'].astype(int)
df = df[(df["positionRace"] > 0) & (df["positionRace"] < f1db_utils.INFINITE_RESULT - 1)]

df['positionQualifying'] = df['positionQualifying'].replace(not_a_number_replace)
df['positionQualifying'] = df["positionQualifying"].fillna(f1db_utils.INFINITE_RESULT)
df['positionQualifying'] = df['positionQualifying'].astype(int)
max_quali_value = df[df["positionQualifying"] != f1db_utils.INFINITE_RESULT]["positionQualifying"].max()
# Replace qualifying NaN (previously replaced with INFINITE) with max real qualifying result + 1
df["positionQualifying"] = df["positionQualifying"].replace(f1db_utils.INFINITE_RESULT, max_quali_value + 1)
df = df[df["positionQualifying"] > 0]
```

```python
df = pd.merge(df, df_races, on="raceId", how="left")
df = pd.merge(df, df_drivers_info, on="driverId", how="left")
df = pd.merge(df, df_circuits, on="circuitId", how="left")

# Filter by Pole and Selected Circuits
df = df[f1db_utils.get_p1_mask(df, f1db_utils.PerformanceType.POLES.value)]
selected_circuits_mask = df["circuitId"].isin(selected_circuits)
df = df[selected_circuits_mask]
# Fillna required to merge different types of Qualifying format
df["time"] = df["time"].fillna(df["q3"])
df["timeMillis"] = df["timeMillis"].fillna(df["q3Millis"])
```

DRIVERS

```
no_test_driver_mask = df["testDriver"] == False
df_by_year = df[no_test_driver_mask].groupby(by=["year"]).count()
df_by_year.rename(columns={"driverId":"officialDriver"}, inplace=True)
df_by_year.drop(columns=["testDriver"], inplace=True)
df_by_year = df_by_year.reset_index()

df_test_drivers_by_year = df[~no_test_driver_mask].groupby(by=["year"]).count()
df_test_drivers_by_year = df_test_drivers_by_year.reset_index()
df_test_drivers_by_year.drop(columns=["driverId"], inplace=True)

return pd.merge(df_by_year, df_test_drivers_by_year, on="year", how="outer")
```

```
df_merged = pd.merge(df_drivers_entrants, df_drivers_info, on="driverId", how="left")
df_merged = pd.merge(df_merged, df_countries, on="nationalityCountryId", how="left")
df_merged = pd.merge(df_merged, df_continents, on="continentId", how="left")
df_merged = df_merged.drop_duplicates(subset=['driverId', 'year'])
df_merged["count"] = df_merged.groupby(["year","alpha3Code"])["driverId"].transform("count")
df_merged["count_display"] = df_merged["count"]

return df_merged
```

Number of Drivers over the Years
- line chart -

Spread of Drivers' Nationalities Over the Years
- geo scatter -

```
match performanceType:
    case f1db_utils.PerformanceType.WINS.value | f1db_utils.PerformanceType.PODIUMS.value:
        podium_mask = (df["positionNumber"] == 1) | (df["positionNumber"] == 2) | (df["positionNumber"] == 3)
        df = df[podium_mask]
        for place in [1,2,3]:
            df[f'count_position_{place}'] = df.groupby('driverId')['positionNumber'].transform(lambda x: (x == place*1.0).sum()) # count P1,P2,P3
        df["count_podiums"] = df.groupby('driverId')['positionNumber'].transform(lambda x: (x <= 3.0).sum()) # count podiums
        df.sort_values(by=["count_position_1"], inplace=True, ascending=False)
        df.drop_duplicates(subset=["driverId","count_position_1","count_position_2","count_position_2"], inplace=True)

    case f1db_utils.PerformanceType.WDCS.value | f1db_utils.PerformanceType.POLES.value:
        df = df[f1db_utils.get_p1_mask(df, performanceType)]
        for place in [1]:
            df[f'count_position_{place}'] = df.groupby('driverId')['positionNumber'].transform(lambda x: (x == place*1.0).sum()) # count P1
        df.sort_values(by=["count_position_1"], inplace=True, ascending=False)
        df.drop_duplicates(subset=["driverId","count_position_1"], inplace=True)
```

Most F1 WDCs / Wins / Podiums / Poles
- bar chart (Absolute) -
- line chart (by Driver) -

```
selected_drivers_mask = df["driverId"].isin(selected_drivers)
df = df[selected_drivers_mask]
if performanceType == f1db_utils.PerformanceType.WDCS.value:
    df = df[f1db_utils.currentSeasonCheckMask(df, performanceType)]
else:
    df = df[performanceType2Mask(df, performanceType)] # show only when he achieved the result as a marker

df.reset_index(inplace=True)
match performanceType:
    case f1db_utils.PerformanceType.WDCS.value:
        df.drop(columns=["index","positionDisplayOrder","positionText","points"], inplace=True)

    case f1db_utils.PerformanceType.WINS.value | f1db_utils.PerformanceType.PODIUMS.value | f1db_utils.PerformanceType.POLES.value:
        df.drop(columns=df.columns.difference(["raceId","driverId","positionNumber"]), inplace=True)
        df_races = pd.read_csv(f"{f1db_utils.folder}/{f1db_utils.races}")
        df_races.rename(columns={"id":"raceId"}, inplace=True)
        df_races.drop(columns=df_races.columns.difference(["raceId","date","grandPrixId","officialName","circuitId"]), inplace=True)
        df = pd.merge(df, df_races, on="raceId", how="left")

for driver_id in selected_drivers:
    counter[driver_id] = 0
df["progressiveCounter"] = df.apply(count_p1 if performanceType != f1db_utils.PerformanceType.PODIUMS.value else count_podiums, axis=1)
```

TEAMS

Number of teams over the years
- *line chart -*

```python
fig = px.line(df,
              x="year",
              y="value",
              markers = True,
              color_discrete_sequence=f1db_utils.custom_colors,
              template = f1db_utils.template
).update_layout(
    f1db_utils.transparent_bg,
    hovermode="x",
    title=f1db_utils.getTitleObj("Number of Teams Over the Year"),
    margin=f1db_utils.margin
)
fig.update_yaxes(title_text='Number of Teams')
fig.update_xaxes(title_text='Year')
fig.update_traces(
    hovertemplate="<br>".join([
        "<b>%{y}</b><extra></extra>",
    ]),
    hoverlabel = f1db_utils.getHoverlabel(13)
```

Spread of teams around the world
- *scatter geo -*

```python
df = pd.DataFrame(columns=['Teams', 'Country', 'Continent','code'])
for index, value in df1['fullName'].items():

    countryId = df1["countryId"][index]

    alpha3Code = df2.loc[df2['id'] == countryId, "alpha3Code"]
    continent = df2.loc[df2['id'] == countryId, "continentId"]
    if not alpha3Code.empty:
        if not continent.empty:
            continent = continent.iloc[0]
            alpha3Code = alpha3Code.iloc[0]
            df = df._append({'Teams': value, 'Country': countryId, 'Continent': continent, 'code': alpha3Code}, ignore_index=True)

df_count = df['Country'].value_counts().reset_index()
```

Teams statistics: WCCs, wins, podiums
- *bar chart (absolute) -*
- *line chart (by teams) -*

```python
match graph_info:
    case 'win':
        df2 = getExtraTeamData()
        df2 = df2[df2['constructorId'].isin(df['id'])]
        df2 =df2.loc[df2['positionNumber'] == 1].reset_index()
        df2['RowNumber'] = df2.groupby('constructorId').cumcount() + 1

        df.rename(columns={"id":"constructorId"}, inplace=True)
        df2 = pd.merge(df2, df, on="constructorId", how="left")

        df2 = f1db_utils.order_df(df2, "constructorId", teamName)
        fig = px.line(df2,
            x='year',
            y='RowNumber',
            color='fullName',
            color_discrete_sequence=f1db_utils.custom_colors,
            markers=True,
            template=f1db_utils.template,
            labels=labels_dict,
            hover_data = {
                "fullName": True
            }
```

END

Meschi Maurizio
5194399
Seasons - Teams

Naccarato Matteo
5330843
Circuits - Drivers