F1-DATA

Meschi Maurizio
5194399
Seasons - Teams

Naccarato Matteo
5330843
Circuits - Drivers

f1-data
https://github.com/f1db/f1db

(project)
(dataset)

Genova, 21/06/2024

Università di Genova

DATASET OVERVIEW

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

f1db-csv > f1db-races-race-results.csv > data

```
"raceId","year","round","positionDisplayOrder","positionNumber","positionText","driverNumber","driverId","constructorId",
1,1950,1,1,"1","1","2","nino-farina","alfa-romeo","alfa-romeo","pirelli",false,70,"2:13:23.600",8003600,,,,,,,,9,1,"1",0,t
1,1950,1,2,2,"2","3","luigi-fagioli","alfa-romeo","alfa-romeo","pirelli",false,70,"2:13:26.200",8006200,,,"+2.600",2600,,
1,1950,1,3,3,"3","4","reg-parnell","alfa-romeo","alfa-romeo","pirelli",false,70,"2:14:15.600",8055600,,,"+52.000",52000,,
1,1950,1,4,4,"4","14","yves-giraud-cabantous","talbot-lago","talbot-lago","dunlop",false,68,,,,,"+2 laps",,2,,,,3,6,"6",2
1,1950,1,5,5,"5","15","louis-rosier","talbot-lago","talbot-lago","dunlop",false,68,,,,,"+2 laps",,2,,,,2,9,"9",4,false,,,
1,1950,1,6,6,"6","12","bob-gerard","era","era","dunlop",false,67,,,,,"+3 laps",,3,,,,,13,"13",7,false,,,false
1,1950,1,7,7,"7","11","cuth-harrison","era","era","dunlop",false,67,,,,,"+3 laps",,3,,,,,15,"15",8,false,,,false
1,1950,1,8,8,"8","16","philippe-etancelin","talbot-lago","talbot-lago","dunlop",false,65,,,,,"+5 laps",,5,,,,,14,"14",6,f
1,1950,1,9,9,"9","6","david-hampshire","maserati","maserati","dunlop",false,64,,,,,"+6 laps",,6,,,,,16,"16",7,false,,,fal
1,1950,1,10,10,"10","10","joe-fry","maserati","maserati","dunlop",false,64,,,,,"+6 laps",,6,,,,,20,"20",10,false,,,false
1,1950,1,11,10,"10","10","brian-shawe-taylor","maserati","maserati","dunlop",true,,,,,,,,,,,,,,,false,,,false
1,1950,1,12,11,"11","18","johnny-claes","talbot-lago","talbot-lago","dunlop",false,64,,,,,"+6 laps",,6,,,,,21,"21",10,fal
1,1950,1,13,,"DNF","1","juan-manuel-fangio","alfa-romeo","alfa-romeo","pirelli",false,62,,,,,,,,,"Oil pipe",,3,"3",,fals
1,1950,1,14,,"NC","23","joe-kelly","alta","alta","dunlop",false,57,,,,,"+13 laps",,13,,,,,19,"19",,false,,,false
1,1950,1,15,,"DNF","21","birabongse-bhanudej","maserati","maserati","pirelli",false,49,,,,,,,,,"Out of fuel",,5,"5",,fal
1,1950,1,16,,"DNF","5","david-murray","maserati","maserati","dunlop",false,44,,,,,,,,,"Engine",,18,"18",,false,,,false
1,1950,1,17,,"DNF","24","geoffrey-crossley","alta","alta","dunlop",false,43,,,,,,,,,"Transmission",,17,"17",,false,,,fal
1,1950,1,18,,"DNF","20","emmanuel-de-graffenried","maserati","maserati","pirelli",false,36,,,,,,,,,"Engine",,8,"8",,fals
1,1950,1,19,,"DNF","19","louis-chiron","maserati","maserati","pirelli",false,24,,,,,,,,,"Clutch",,11,"11",,false,,,false
1,1950,1,20,,"DNF","17","eugene-martin","talbot-lago","talbot-lago","dunlop",false,8,,,,,,,,,"Oil pressure",,7,"7",,fals
1,1950,1,21,,"DNF","9","peter-walker","era","era","dunlop",false,5,,,,,,,,,"Gearbox",,10,"10",,false,,,false
1,1950,1,22,,"DNF","9","tony-rolt","era","era","dunlop",true,,,,,,,,,,,,,,,false,,,false
1,1950,1,23,,"DNF","8","leslie-johnson","era","era","dunlop",false,2,,,,,,,,,"Compressor",,12,"12",,false,,,false
```

f1db-csv > f1db-circuits.csv > data

```
"id","name","fullName","previousNames","type","placeName","countryId","latitude","longitude","totalRacesHeld"
"adelaide","Adelaide","Adelaide Street Circuit",,"STREET","Adelaide","australia",-34.927222,138.617222,11
"aida","Aida","Okayama International Circuit","TI Circuit Aida","RACE","Aida","japan",34.915,134.221111,2
"ain-diab","Ain-Diab","Ain-Diab Circuit",,"ROAD","Casablanca","morocco",33.578611,-7.6875,1
"aintree","Aintree","Aintree Motor Racing Circuit",,"ROAD","Aintree","united-kingdom",53.476944,-2.940556,5
"anderstorp","Anderstorp Raceway","Anderstorp Raceway","Scandinavian Raceway","RACE","Anderstorp","sweden",57.264167,13.601389,6
"austin","Americas","Circuit of the Americas",,"RACE","Austin","united-states-of-america",30.132778,-97.641111,11
"avus","AVUS","AVUS",,"ROAD","Berlin","germany",52.480556,13.251389,1
"bahrain","Bahrain","Bahrain International Circuit",,"RACE","Sakhir","bahrain",26.0325,50.510556,21
"baku","Baku","Baku City Circuit",,"STREET","Baku","azerbaijan",40.3725,49.853333,7
"brands-hatch","Brands Hatch","Brands Hatch",,"RACE","Fawkham","united-kingdom",51.356667,0.2625,14
"bremgarten","Bremgarten","Circuit Bremgarten",,"ROAD","Bern","switzerland",46.95,7.410833,5
"buddh","Buddh","Buddh International Circuit",,"RACE","Greater Noida","india",28.350556,77.535,3
```

```python
# Download and extract from {url} to {folder}
def download(url, last_version):
    print(f"f1-data > Downloading\t({last_version})")
    r = requests.get(url)
    zf = zipfile.ZipFile(io.BytesIO(r.content))
    zf.extractall(folder)
    with open(last_version_file, 'w') as file:
        file.write(last_version)
        print(f"f1-data > Successfully Downloaded\t({last_version})")

# Get Data only if a new version is available
def get_data():
    response = requests.get(gh_latest_release)
    last_version = response.json()["name"]
    print(f"f1-data > Fetching Version\t({last_version})")
    url = f'https://github.com/f1db/f1db/releases/download/{last_version}/f1db-csv.zip'

    try:
        with open(last_version_file, 'r') as file:
            content = file.read().strip()

        if content == last_version:
            print(f"f1-data > Already up to date\t({last_version})")
        else:
            download(url, last_version)

    except FileNotFoundError:
        download(url, last_version)
```

```
f1-data > Fetching Version      (v2024.9.0)
f1-data > Already up to date    (v2024.9.0)
```

SEASONS

Number of GP over the years
- line chart -

```
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 -

```
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 -

```
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 -

```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
```

Qualifying vs Race
- scatter plot -

```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]
```

Pole Lap Time Over the Years
- line chart -

```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

Number of Drivers over the Years
- line chart -

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

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

```
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
```

```
match performanceType:
    # 1st, 2nd, 3rd places only
    case 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)

    # 1st place only
    case f1db_utils.PerformanceType.WINS.value | 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)
```

```
# Drivers filter by {selected_drivers}
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 the marker only when the driver achieved the result

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

    # Get races info
    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")

# Progressive counter of drivers' achievements
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 -

```
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 -

```
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) -

```
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

f1-data
https://github.com/f1db/f1db

(project)
(dataset)

Genova, 21/06/2024

Università di Genova