DNF

October 24, 2023

[ ]:
```
import pandas as pd
import statsmodels
import numpy as np
import matplotlib.pyplot as plt

status = pd.read_csv('status.csv',encoding = "utf-8")
drivers = pd.read_csv('drivers.csv',encoding = "utf-8")
constructors = pd.read_csv('constructors.csv',encoding = "utf-8")
circuits = pd.read_csv('circuits.csv',encoding = "utf-8")
races = pd.read_csv('races.csv',encoding = "utf-8")
results = pd.read_csv('results.csv', encoding="utf-8")
constructor_standings = pd.read_csv('constructor_standings.csv',encoding =␣
 ↪"utf-8")
print('This is the Status data')
status.head()
```

```
This is the Status data
```

[ ]:

	statusId	status
0	1	Finished
1	2	Disqualified
2	3	Accident
3	4	Collision
4	5	Engine

[ ]:
```
drivers.head()
```

[ ]:

	driverId	driverRef	number	code	forename	surname	dob	\
0	1	hamilton	44	HAM	Lewis	Hamilton	1985-01-07	
1	2	heidfeld	\N	HEI	Nick	Heidfeld	1977-05-10	
2	3	rosberg	6	ROS	Nico	Rosberg	1985-06-27	
3	4	alonso	14	ALO	Fernando	Alonso	1981-07-29	
4	5	kovalainen	\N	KOV	Heikki	Kovalainen	1981-10-19	

	nationality	url
0	British	http://en.wikipedia.org/wiki/Lewis_Hamilton
1	German	http://en.wikipedia.org/wiki/Nick_Heidfeld
2	German	http://en.wikipedia.org/wiki/Nico_Rosberg

```
3     Spanish    http://en.wikipedia.org/wiki/Fernando_Alonso
4     Finnish  http://en.wikipedia.org/wiki/Heikki_Kovalainen
```

[ ]:
```
drivers['driver_name'] = drivers['forename'] + ' ' + drivers['surname']
col = ['driverId', 'driver_name', 'nationality']
my_drivers = drivers[col]
my_drivers.rename(columns = {'nationality': 'driver_nationality' }, inplace =␣
 ↪True)
my_drivers.head()
```

```
/var/folders/zj/rx7n4ybx1ml9886cdbwv429m0000gn/T/ipykernel_84467/3935489618.py:4
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  my_drivers.rename(columns = {'nationality': 'driver_nationality' }, inplace =
True)
```

[ ]:
```
   driverId        driver_name driver_nationality
0         1     Lewis Hamilton            British
1         2      Nick Heidfeld             German
2         3       Nico Rosberg             German
3         4    Fernando Alonso            Spanish
4         5  Heikki Kovalainen            Finnish
```

[ ]:
```
constructors.head()
```

[ ]:
```
   constructorId constructorRef        name nationality  \
0              1        mclaren     McLaren     British
1              2     bmw_sauber  BMW Sauber      German
2              3       williams    Williams     British
3              4        renault     Renault      French
4              5     toro_rosso  Toro Rosso     Italian

                                          url
0              http://en.wikipedia.org/wiki/McLaren
1           http://en.wikipedia.org/wiki/BMW_Sauber
2  http://en.wikipedia.org/wiki/Williams_Grand_Pr…
3  http://en.wikipedia.org/wiki/Renault_in_Formul…
4   http://en.wikipedia.org/wiki/Scuderia_Toro_Rosso
```

[ ]:
```
cols = ['constructorId', 'name', 'nationality']
my_constructors = constructors[cols]
my_constructors.rename(columns = {'nationality': 'team_nationality' }, inplace␣
 ↪= True)
my_constructors.rename(columns = {'name': 'constructor' }, inplace = True)
my_constructors.head()
```

```
/var/folders/zj/rx7n4ybx1ml9886cdbwv429m0000gn/T/ipykernel_84467/1242970380.py:3
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  my_constructors.rename(columns = {'nationality': 'team_nationality' }, inplace
= True)
/var/folders/zj/rx7n4ybx1ml9886cdbwv429m0000gn/T/ipykernel_84467/1242970380.py:4
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  my_constructors.rename(columns = {'name': 'constructor' }, inplace = True)
```

[ ]:

	constructorId	constructor	team_nationality
0	1	McLaren	British
1	2	BMW Sauber	German
2	3	Williams	British
3	4	Renault	French
4	5	Toro Rosso	Italian

[ ]:
```
circuits.head()
```

[ ]:

	circuitId	circuitRef	name	location	\
0	1	albert_park	Albert Park Grand Prix Circuit	Melbourne	
1	2	sepang	Sepang International Circuit	Kuala Lumpur	
2	3	bahrain	Bahrain International Circuit	Sakhir	
3	4	catalunya	Circuit de Barcelona-Catalunya	Montmeló	
4	5	istanbul	Istanbul Park	Istanbul	

	country	lat	lng	alt	\
0	Australia	-37.84970	144.96800	10	
1	Malaysia	2.76083	101.73800	18	
2	Bahrain	26.03250	50.51060	7	
3	Spain	41.57000	2.26111	109	
4	Turkey	40.95170	29.40500	130	

	url
0	http://en.wikipedia.org/wiki/Melbourne_Grand_P…
1	http://en.wikipedia.org/wiki/Sepang_Internatio…
2	http://en.wikipedia.org/wiki/Bahrain_Internati…
3	http://en.wikipedia.org/wiki/Circuit_de_Barcel…
4	http://en.wikipedia.org/wiki/Istanbul_Park

[ ]:
```
colss = ['circuitId','circuitRef', 'name', 'location', 'country']
my_circuits = circuits[colss]
my_circuits.rename(columns = {'name': 'circuit' }, inplace = True)
my_circuits.head()
```

```
/var/folders/zj/rx7n4ybx1ml9886cdbwv429m0000gn/T/ipykernel_84467/823439227.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  my_circuits.rename(columns = {'name': 'circuit' }, inplace = True)
```

[ ]:

	circuitId	circuitRef	circuit	location	country
0	1	albert_park	Albert Park Grand Prix Circuit	Melbourne	Australia
1	2	sepang	Sepang International Circuit	Kuala Lumpur	Malaysia
2	3	bahrain	Bahrain International Circuit	Sakhir	Bahrain
3	4	catalunya	Circuit de Barcelona-Catalunya	Montmeló	Spain
4	5	istanbul	Istanbul Park	Istanbul	Turkey

[ ]:
```
constructor_standings.head()
```

[ ]:

	constructorStandingsId	raceId	constructorId	points	position	positionText	wins
0	1	18	1	14.0	1	1	1
1	2	18	2	8.0	3	3	0
2	3	18	3	9.0	2	2	0
3	4	18	4	5.0	4	4	0
4	5	18	5	2.0	5	5	0

[ ]:
```
colss4 = ['constructorStandingsId',        'raceId', 'constructorId', 'points']
my_constructor_standings = constructor_standings[colss4]
my_constructor_standings.head()
```

[ ]:

	constructorStandingsId	raceId	constructorId	points
0	1	18	1	14.0
1	2	18	2	8.0
2	3	18	3	9.0
3	4	18	4	5.0
4	5	18	5	2.0

[ ]:
```
races.head()
```

[ ]:

	raceId	year	round	circuitId	name	date	time	url	fp1_date	fp1_time	fp2_date	fp2_time	fp3_date	fp3_time	quali_date	quali_time	sprint_date	sprint_time
0	1	2009	1	1	Australian Grand Prix	2009-03-29	06:00:00	http://en.wikipedia.org/wiki/2009_Australian_G…	\N	\N	\N	\N	\N	\N	\N	\N	\N	\N
1	2	2009	2	2	Malaysian Grand Prix	2009-04-05	09:00:00	http://en.wikipedia.org/wiki/2009_Malaysian_Gr…	\N	\N	\N	\N	\N	\N	\N	\N	\N	\N
2	3	2009	3	17	Chinese Grand Prix	2009-04-19	07:00:00	http://en.wikipedia.org/wiki/2009_Chinese_Gran…	\N	\N	\N	\N	\N	\N	\N	\N	\N	\N
3	4	2009	4	3	Bahrain Grand Prix	2009-04-26	12:00:00	http://en.wikipedia.org/wiki/2009_Bahrain_Gran…	\N	\N	\N	\N	\N	\N	\N	\N	\N	\N
4	5	2009	5	4	Spanish Grand Prix	2009-05-10	12:00:00	http://en.wikipedia.org/wiki/2009_Spanish_Gran…	\N	\N	\N	\N	\N	\N	\N	\N	\N	\N

[ ]:
```
colsss = ['raceId', 'year', 'round', 'circuitId', 'name']
my_races = races[colsss]
my_races.rename(columns = {'name': 'prix' }, inplace = True)
my_races.head()
```

```
/var/folders/zj/rx7n4ybx1ml9886cdbwv429m0000gn/T/ipykernel_84467/1917903944.py:3
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
my_races.rename(columns = {'name': 'prix' }, inplace = True)
```

[ ]:
```
   raceId  year  round  circuitId                  prix
0       1  2009      1          1  Australian Grand Prix
1       2  2009      2          2   Malaysian Grand Prix
2       3  2009      3         17     Chinese Grand Prix
3       4  2009      4          3     Bahrain Grand Prix
4       5  2009      5          4     Spanish Grand Prix
```

[ ]:
```
results.head()
```

[ ]:
```
   resultId  raceId  driverId  constructorId number  grid position  \
0         1      18         1              1     22     1        1
1         2      18         2              2      3     5        2
2         3      18         3              3      7     7        3
3         4      18         4              4      5    11        4
4         5      18         5              1     23     3        5

  positionText  positionOrder  points  laps         time milliseconds  \
0            1              1    10.0    58  1:34:50.616      5690616
1            2              2     8.0    58       +5.478      5696094
2            3              3     6.0    58       +8.163      5698779
3            4              4     5.0    58      +17.181      5707797
4            5              5     4.0    58      +18.014      5708630

  fastestLap rank fastestLapTime fastestLapSpeed  statusId
0         39    2       1:27.452         218.300         1
1         41    3       1:27.739         217.586         1
2         41    5       1:28.090         216.719         1
3         58    7       1:28.603         215.464         1
4         43    1       1:27.418         218.385         1
```

[ ]:
```
col2 = ['resultId', 'raceId', 'driverId', 'constructorId', 'points', 'laps',␣
 ↪'statusId']
my_results = results[col2]
my_results.head()
```

[ ]:
```
   resultId  raceId  driverId  constructorId  points  laps  statusId
0         1      18         1              1    10.0    58         1
1         2      18         2              2     8.0    58         1
2         3      18         3              3     6.0    58         1
3         4      18         4              4     5.0    58         1
4         5      18         5              1     4.0    58         1
```

[ ]:
```
# make one dataframe
my_data = pd.merge(my_results, my_races, on ='raceId', how ='left')
my_data = pd.merge(my_data, my_circuits, on ='circuitId', how ='left')
my_data = pd.merge(my_data, my_drivers, on ='driverId', how ='left')
```

```
my_data = pd.merge(my_data, status, on ='statusId', how ='left')
my_data = pd.merge(my_data, my_constructors, on ='constructorId', how ='left')
my_data = pd.merge(my_data, my_constructor_standings, on =['constructorId',␣
 ↪'raceId'], how ='left')
my_data.head()
```

[ ]:

	resultId	raceId	driverId	constructorId	points_x	laps	statusId	year	\
0	1	18	1	1	10.0	58	1	2008	
1	2	18	2	2	8.0	58	1	2008	
2	3	18	3	3	6.0	58	1	2008	
3	4	18	4	4	5.0	58	1	2008	
4	5	18	5	1	4.0	58	1	2008	

	round	circuitId	…	circuit	location	\
0	1	1	…	Albert Park Grand Prix Circuit	Melbourne	
1	1	1	…	Albert Park Grand Prix Circuit	Melbourne	
2	1	1	…	Albert Park Grand Prix Circuit	Melbourne	
3	1	1	…	Albert Park Grand Prix Circuit	Melbourne	
4	1	1	…	Albert Park Grand Prix Circuit	Melbourne	

	country	driver_name	driver_nationality	status	constructor	\
0	Australia	Lewis Hamilton	British	Finished	McLaren	
1	Australia	Nick Heidfeld	German	Finished	BMW Sauber	
2	Australia	Nico Rosberg	German	Finished	Williams	
3	Australia	Fernando Alonso	Spanish	Finished	Renault	
4	Australia	Heikki Kovalainen	Finnish	Finished	McLaren	

	team_nationality	constructorStandingsId	points_y
0	British	1.0	14.0
1	German	2.0	8.0
2	British	3.0	9.0
3	French	4.0	5.0
4	British	1.0	14.0

[5 rows x 22 columns]

```
# Dataframe of all drivers who finished a Grand Prix
finished_race = my_data.loc[my_data['status'].isin(['Finished', '+1 Lap', '+2␣
 ↪Laps', '+3 Laps', '+4 Laps', '+5 Laps', '+6 Laps', '+7 Laps', '+8 Laps', '+9␣
 ↪Laps',
                                                     '+10 Laps', '+11 Laps',␣
 ↪'+14 Laps'])] # included the overlapped status

# Percentage of drivers finishing a race in all of F1 history
finish = round((finished_race.shape[0] / my_data.shape[0]) * 100, 3)
```

```
print(f"In all of F1 history, {finish}% drivers have finished a Grand Prix that they started")
```

```
In all of F1 history, 55.691% drivers have finished a Grand Prix that they started
```

[ ]:
```
my_data.describe()
```

[ ]:
```
           resultId        raceId      driverId  constructorId      points_x  \
count  26180.000000  26180.000000  26180.000000   26180.000000  26180.000000
mean   13091.388426    538.896982    268.148739      49.237051      1.918795
std     7558.893818    304.537441    274.174971      60.420088      4.240048
min        1.000000      1.000000      1.000000       1.000000      0.000000
25%     6545.750000    295.000000     57.000000       6.000000      0.000000
50%    13090.500000    521.000000    164.000000      25.000000      0.000000
75%    19635.250000    794.000000    373.000000      59.000000      2.000000
max    26185.000000   1115.000000    859.000000     214.000000     50.000000

                laps      statusId          year         round     circuitId  \
count  26180.000000  26180.000000  26180.000000  26180.000000  26180.000000
mean      46.101795     17.459358   1990.677082      8.393965     23.579908
std       29.691532     26.161846     19.573972      4.957933     18.782538
min        0.000000      1.000000   1950.000000      1.000000      1.000000
25%       22.000000      1.000000   1976.000000      4.000000      9.000000
50%       53.000000     10.000000   1991.000000      8.000000     18.000000
75%       66.000000     14.000000   2008.000000     12.000000     34.000000
max      200.000000    141.000000   2023.000000     22.000000     79.000000

       constructorStandingsId      points_y
count            24313.000000  24313.000000
mean             16719.382224     38.125756
std               8916.962800     80.463991
min                  1.000000      0.000000
25%               8685.000000      1.000000
50%              20138.000000     10.000000
75%              25045.000000     36.000000
max              28632.000000    765.000000
```

[ ]:
```
# List of statusId values to exclude - those who finished
exclude_ids = [1, 11, 12, 13, 14, 15, 16, 17, 18, 19, 45, 50, 128, 53, 55, 58, 88, 111, 112,
               113, 114, 115, 116, 117, 118, 119, 120, 122, 123, 124, 125, 127, 133, 134]

# Filter data to exclude specific statusId values and include all the rest
crashes = my_data.loc[~my_data['statusId'].isin(exclude_ids)]
```

```
# Describe the resulting data
crashes.describe()
#11524
```

[ ]:

	resultId	raceId	driverId	constructorId	points_x	\
count	11524.000000	11524.000000	11524.000000	11524.000000	11524.000000	
mean	11865.126605	494.742798	238.241843	51.008330	0.013016	
std	6353.839707	249.597802	219.073648	55.302585	0.187255	
min	7.000000	1.000000	1.000000	1.000000	0.000000	
25%	6700.750000	308.000000	88.000000	15.000000	0.000000	
50%	11589.500000	477.000000	170.000000	32.000000	0.000000	
75%	16871.000000	681.000000	327.250000	60.000000	0.000000	
max	26185.000000	1115.000000	858.000000	214.000000	6.000000	

	laps	statusId	year	round	circuitId	\
count	11524.000000	11524.000000	11524.000000	11524.000000	11524.000000	
mean	24.274731	30.516314	1985.009198	7.826189	23.992103	
std	24.745933	33.581783	17.070282	4.650605	17.857834	
min	0.000000	2.000000	1950.000000	1.000000	1.000000	
25%	2.000000	5.000000	1974.000000	4.000000	10.000000	
50%	19.000000	10.000000	1986.000000	7.000000	19.000000	
75%	40.000000	54.000000	1996.000000	11.000000	36.000000	
max	196.000000	141.000000	2023.000000	22.000000	79.000000	

	constructorStandingsId	points_y
count	10479.000000	10479.000000
mean	15096.260807	18.181649
std	8182.478043	44.405906
min	2.000000	0.000000
25%	8487.000000	0.000000
50%	11282.000000	4.000000
75%	22865.500000	17.000000
max	28629.000000	701.000000

[ ]:
```
# Remove duplicates based on 'raceId' and 'driverId'
crashes_unique = crashes.drop_duplicates(subset=['raceId', 'driverId'])

crashes_unique.describe()
```

[ ]:

	resultId	raceId	driverId	constructorId	points_x	\
count	11498.000000	11498.000000	11498.000000	11498.000000	11498.000000	
mean	11847.029396	494.093147	237.580275	50.890503	0.013046	
std	6349.071231	249.491276	218.839733	55.278920	0.187466	
min	7.000000	1.000000	1.000000	1.000000	0.000000	
25%	6694.250000	308.000000	88.000000	15.000000	0.000000	
50%	11563.500000	476.000000	170.000000	32.000000	0.000000	
75%	16841.750000	679.000000	326.000000	59.000000	0.000000	

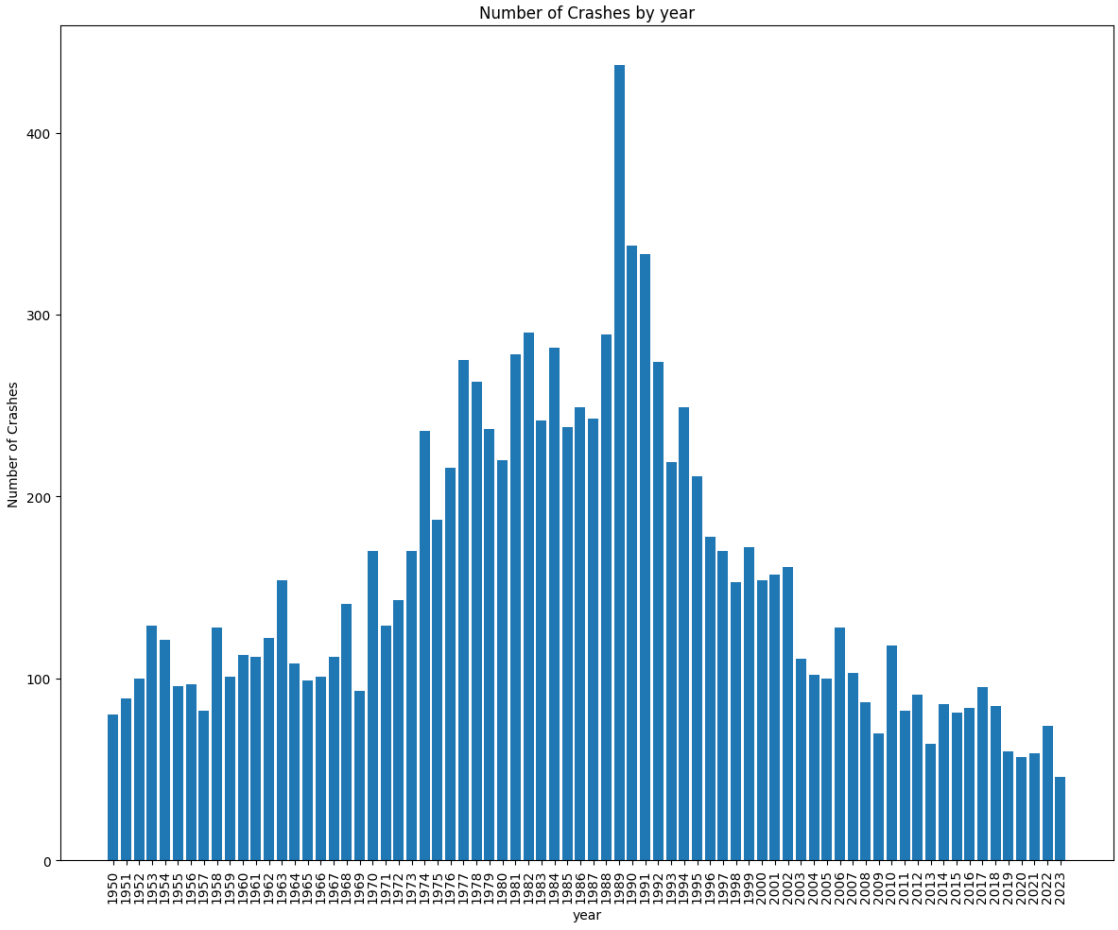
```
max    26185.000000   1115.000000    858.000000     214.000000      6.000000

              laps      statusId          year         round     circuitId  \
count  11498.000000  11498.000000  11498.000000  11498.000000  11498.000000
mean      24.143416     30.504175   1985.074709      7.835624     23.988607
std       24.425842     33.576483     17.031860      4.649396     17.858035
min        0.000000      2.000000   1950.000000      1.000000      1.000000
25%        2.000000      5.000000   1974.000000      4.000000     10.000000
50%       19.000000     10.000000   1986.000000      7.000000     19.000000
75%       40.000000     54.000000   1996.000000     11.000000     36.000000
max      196.000000    141.000000   2023.000000     22.000000     79.000000

       constructorStandingsId      points_y
count            10473.000000  10473.000000
mean             15093.612336     18.184713
std               8183.935630     44.417275
min                  2.000000      0.000000
25%               8485.000000      0.000000
50%              11280.000000      4.000000
75%              22866.000000     17.000000
max              28629.000000    701.000000
```

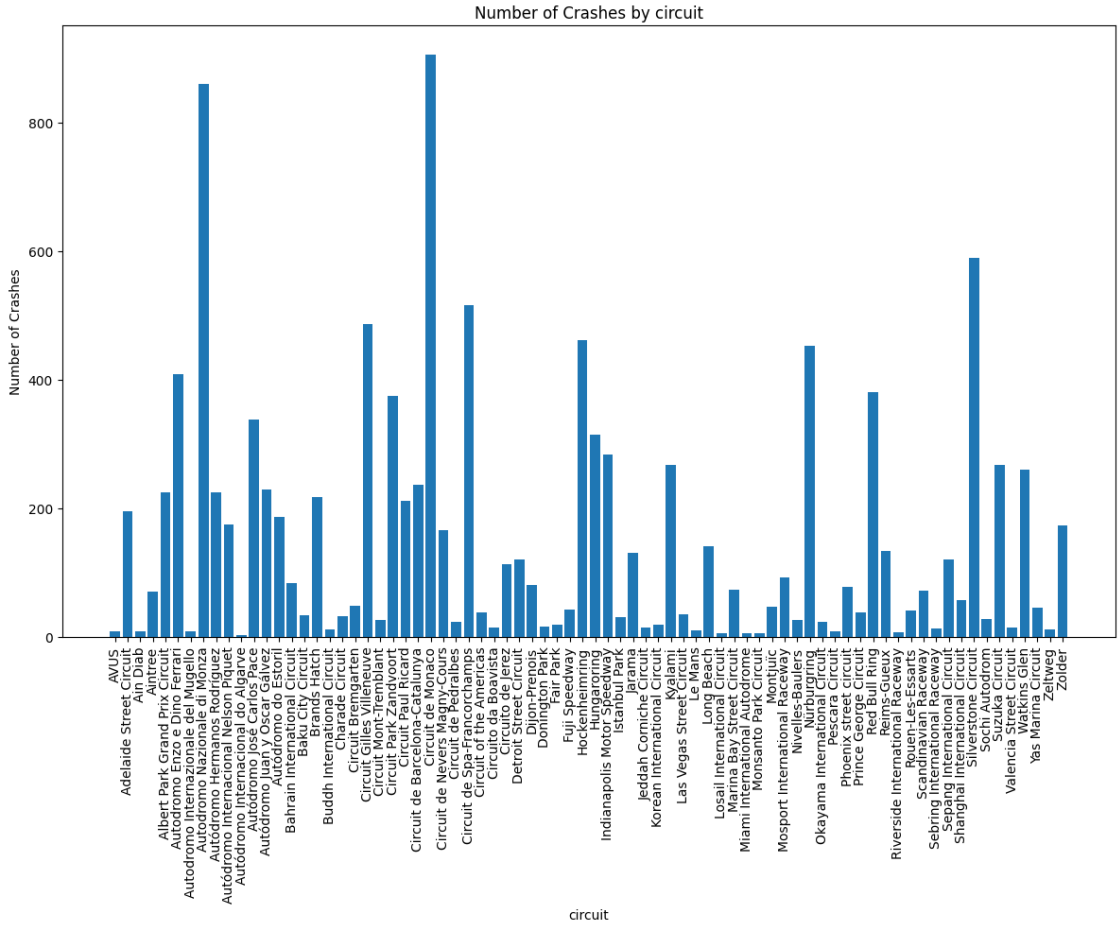
[ ]:
```
crashes.head()
```

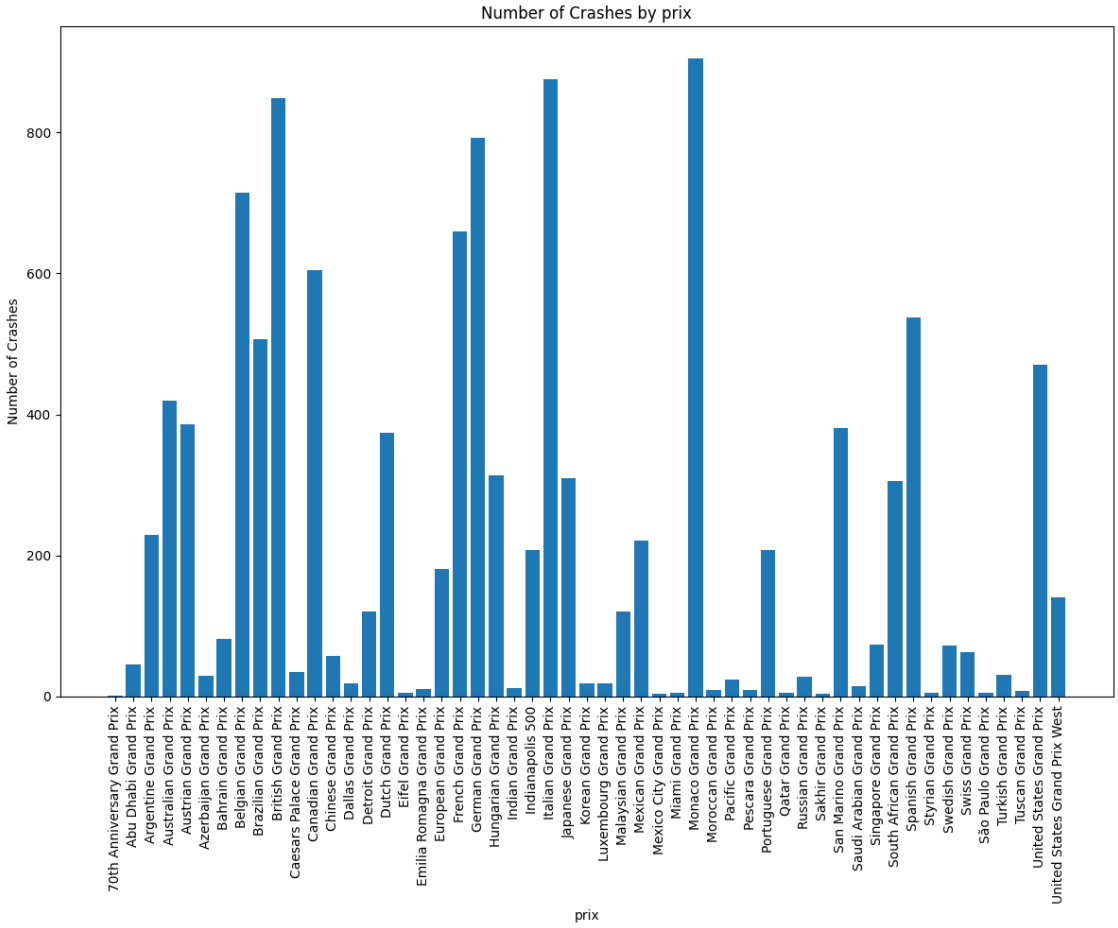
[ ]:
```
    resultId  raceId  driverId  constructorId  points_x  laps  statusId  year  \
6          7      18         7              5       2.0    55         5  2008
7          8      18         8              6       1.0    53         5  2008
8          9      18         9              2       0.0    47         4  2008
9         10      18        10              7       0.0    43         3  2008
10        11      18        11              8       0.0    32         7  2008

    round  circuitId  …                        circuit   location  \
6       1          1  …  Albert Park Grand Prix Circuit  Melbourne
7       1          1  …  Albert Park Grand Prix Circuit  Melbourne
8       1          1  …  Albert Park Grand Prix Circuit  Melbourne
9       1          1  …  Albert Park Grand Prix Circuit  Melbourne
10      1          1  …  Albert Park Grand Prix Circuit  Melbourne

      country         driver_name driver_nationality        status  \
6   Australia  Sébastien Bourdais             French        Engine
7   Australia      Kimi Räikkönen            Finnish        Engine
8   Australia       Robert Kubica             Polish     Collision
9   Australia          Timo Glock             German      Accident
10  Australia        Takuma Sato           Japanese  Transmission

    constructor team_nationality constructorStandingsId points_y
6    Toro Rosso          Italian                    5.0      2.0
```

```
7       Ferrari          Italian                    6.0      1.0
8    BMW Sauber           German                    2.0      8.0
9        Toyota         Japanese                    NaN      NaN
10  Super Aguri         Japanese                    NaN      NaN

[5 rows x 22 columns]
```

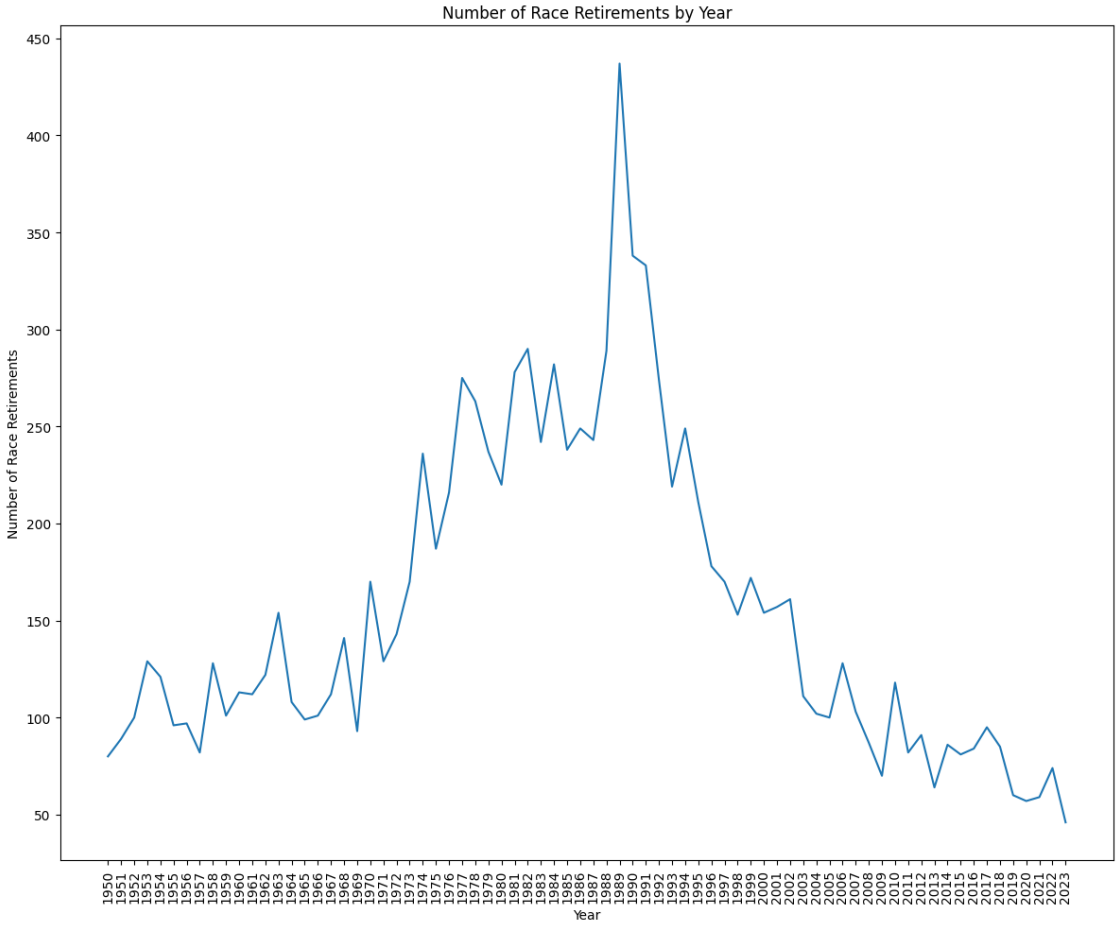
[ ]:
```python
def plot_crashes_by_columns(crashes, column_names):
    """Plot the number of crashes based on the selected columns from the␣
 ↪crashes df."""

    for column_name in column_names:
        crashes_by_column = crashes.groupby(column_name).size()

        plt.figure(figsize=(12, 10))
        plt.bar(crashes_by_column.index, crashes_by_column.values)
        plt.title(f'Number of Crashes by {column_name}')
        plt.ylabel('Number of Crashes')
        plt.xlabel(column_name)
        plt.xticks(crashes_by_column.index, rotation=90)
        plt.tight_layout()
        plt.show()

# Example usage:
columns_to_plot = ['year', 'circuit', 'prix']
plot_crashes_by_columns(crashes, columns_to_plot)
```


Number of Crashes by year
Number of Crashes
0
100
200
300
400
year
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023


Number of Crashes by circuit
Number of Crashes
0
200
400
600
800
circuit
AVUS
Adelaide Street Circuit
Ain Diab
Anderstorp
Albert Park Grand Prix Circuit
Autodromo Enzo e Dino Ferrari
Autodromo Internazionale del Mugello
Autodromo Nazionale di Monza
Autódromo Hermanos Rodríguez
Autódromo Internacional Nelson Piquet
Autódromo Internacional do Algarve
Autódromo José Carlos Pace
Autódromo Juan y Oscar Gálvez
Autódromo do Estoril
Bahrain International Circuit
Baku City Circuit
Brands Hatch
Buddh International Circuit
Charade Circuit
Circuit Bremgarten
Circuit Gilles Villeneuve
Circuit Mont-Tremblant
Circuit Park Zandvoort
Circuit Paul Ricard
Circuit de Barcelona-Catalunya
Circuit de Monaco
Circuit de Nevers Magny-Cours
Circuit de Pedralbes
Circuit de Spa-Francorchamps
Circuit of the Americas
Circuito da Boavista
Circuito de Jerez
Detroit Street Circuit
Dijon-Prenois
Donington Park
Fair Park
Fuji Speedway
Hockenheimring
Hungaroring
Indianapolis Motor Speedway
Istanbul Park
Jarama
Jeddah Corniche Circuit
Korean International Circuit
Kyalami
Las Vegas Street Circuit
Le Mans
Long Beach
Losail International Circuit
Marina Bay Street Circuit
Miami International Autodrome
Monsanto Park Circuit
Montjuic
Mosport International Raceway
Nivelles-Baulers
Nürburgring
Okayama International Circuit
Pescara Circuit
Phoenix street circuit
Prince George Circuit
Red Bull Ring
Reims-Gueux
Riverside International Raceway
Rouen-Les-Essarts
Scandinavian Raceway
Sebring International Raceway
Sepang International Circuit
Shanghai International Circuit
Silverstone Circuit
Sochi Autodrom
Suzuka Circuit
Valencia Street Circuit
Watkins Glen
Yas Marina Circuit
Zeltweg
Zolder



[ ]:
```
DNF_year = crashes.groupby('year').size()

plt.figure(figsize=(12, 10))
plt.plot(DNF_year.index, DNF_year.values)
plt.title(f'Number of Race Retirements by Year')
plt.ylabel('Number of Race Retirements')
plt.xlabel('Year')
plt.xticks(DNF_year.index, rotation=90)
plt.tight_layout()
plt.show()
```



[ ]:
```
import plotly.express as px
import plotly.graph_objects as go

def plot_dnf(year):
    # Filter crashes for the given year
    crashes_in_specific_year = crashes[crashes['year'] == year]

    # For Drivers
    crashes_driver = crashes_in_specific_year.groupby('driver_name').size().
 ↪reset_index(name='counts')
    scaled_size_driver = crashes_driver['counts']**2
    fig_driver = create_bubble_chart(crashes_driver, 'driver_name',␣
 ↪scaled_size_driver, 'Driver Name', f'DNFs by Driver in {year}')
    fig_driver.show()

    # For Constructors
    crashes_team = crashes_in_specific_year.groupby('constructor').size().
 ↪reset_index(name='counts')
```

```
    scaled_size_team = crashes_team['counts']**2
    fig_team = create_bubble_chart(crashes_team, 'constructor',␣
 ↪scaled_size_team, 'Constructor', f'DNFs by Constructor in {year}')
    fig_team.show()

def create_bubble_chart(df, x_col, scaled_size, xaxis_title, chart_title):
    colors = px.colors.qualitative.Plotly

    fig = go.Figure(data=[go.Scatter(
        x=df[x_col], y=df['counts'],
        mode='markers',
        marker=dict(
            color=colors * (len(df) // len(colors)) + colors[:len(df) %␣
 ↪len(colors)],
            size=scaled_size / max(scaled_size) * 100,  # Normalize and then␣
 ↪scale to desired range
            sizemin=6  # Minimum marker size
        )
    )])

    fig.update_layout(
        height=800,
        yaxis=dict(title='Number of DNFs'),
        xaxis=dict(title=xaxis_title),
        title=chart_title
    )

    return fig
```

[ ]:
```
# Use the function
plot_dnf(2021)
```

[ ]:
```
races.head()
races_points = races[['raceId', 'year']]
races_points.head()
```

[ ]:
```
   raceId  year
0       1  2009
1       2  2009
2       3  2009
3       4  2009
4       5  2009
```

[ ]:
```
constructor_standings.head()
constructor_points = constructor_standings[['constructorStandingsId', 'raceId',␣
 ↪'constructorId', 'points']]
constructor_points.head()
```

[ ]:

	constructorStandingsId	raceId	constructorId	points
0	1	18	1	14.0
1	2	18	2	8.0
2	3	18	3	9.0
3	4	18	4	5.0
4	5	18	5	2.0

[ ]:
```
constructors.head()
cons = constructors[['constructorId', 'name']]
cons.head()
```

[ ]:

	constructorId	name
0	1	McLaren
1	2	BMW Sauber
2	3	Williams
3	4	Renault
4	5	Toro Rosso

[ ]:
```
final_df = pd.merge(constructor_points, cons, on = 'constructorId', how =␣
↪'left')
final_df = pd.merge(final_df, races_points, on = 'raceId', how = 'left')
final_df.head()
```

[ ]:

	constructorStandingsId	raceId	constructorId	points	name	year
0	1	18	1	14.0	McLaren	2008
1	2	18	2	8.0	BMW Sauber	2008
2	3	18	3	9.0	Williams	2008
3	4	18	4	5.0	Renault	2008
4	5	18	5	2.0	Toro Rosso	2008

[ ]:
```
final_df['total_points'] = final_df.groupby(['raceId',␣
↪'constructorId'])['points'].transform('sum')

# Since we have a sum, drop duplicates
last_df = final_df[['constructorId', 'points', 'name', 'year']]
last_df = last_df.drop_duplicates(subset=['constructorId', 'year'])
```

[ ]:

	constructorId	points	name	year
12316	210	1.0	Haas F1 Team	2020
12317	9	78.0	Red Bull	2020
12318	3	0.0	Williams	2020
12319	51	2.0	Alfa Romeo	2020
12320	4	32.0	Renault	2020
12321	213	13.0	AlphaTauri	2020
12322	211	42.0	Racing Point	2020
12323	1	51.0	McLaren	2020
12324	6	43.0	Ferrari	2020

```
12325            131    146.0      Mercedes  2020
```

[ ]:
```
filtered_df = last_df[last_df['year'] == 2020]
filtered_df
```