Pokedex Exploratory Data Analysis

1. Importing Libraries and Data Set

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import os
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
os.chdir(r'C:\Javeria\Projects\Data Science Roadmap\EDA - Week 8\Pokemon dataset ED
```

In [3]:
```
df=pd.read_csv('pokemons.csv')
df.head()
```

Out[3]:

	id	name	rank	generation	evolves_from	type1	type2	hp	atk	def	spatk	spdef
0	1	bulbasaur	ordinary	generation-i	nothing	grass	poison	45	49	49	65	65
1	2	ivysaur	ordinary	generation-i	bulbasaur	grass	poison	60	62	63	80	80
2	3	venusaur	ordinary	generation-i	ivysaur	grass	poison	80	82	83	100	100
3	4	charmander	ordinary	generation-i	nothing	fire	NaN	39	52	43	60	50
4	5	charmeleon	ordinary	generation-i	charmander	fire	NaN	58	64	58	80	65

2. Exploring the Dataset

In [4]:
```
df.shape
```

Out[4]:
```
(1025, 18)
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['id', 'name', 'rank', 'generation', 'evolves_from', 'type1', 'type2',
       'hp', 'atk', 'def', 'spatk', 'spdef', 'speed', 'total', 'height',
       'weight', 'abilities', 'desc'],
      dtype='object')
```

What do the columns stand for?

id = pokedex id of the pokemon

name = name of the pokemon

rank = whether the pokemon is legendary, ordinary, mythical or baby

generation = generation of each pokemon

evolves_from = the pokemon's prior form

type1 = primary type of the pokemon

type2 = secondary type of the pokemon

hp = hp stat

atk = attack stat

def = defense stat

spatk = special attack stat

spdef = special defense stat

speed = speed stat

total = sum of all the stats

height = height in decimeters

weight = weight in hectograms

abilities = abilities of the pokemon

desc = short description about the pokemon

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 18 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   id            1025 non-null   int64
 1   name          1025 non-null   object
 2   rank          1025 non-null   object
 3   generation    1025 non-null   object
 4   evolves_from  1025 non-null   object
 5   type1         1025 non-null   object
 6   type2         526 non-null    object
 7   hp            1025 non-null   int64
 8   atk           1025 non-null   int64
 9   def           1025 non-null   int64
 10  spatk         1025 non-null   int64
 11  spdef         1025 non-null   int64
 12  speed         1025 non-null   int64
 13  total         1025 non-null   int64
 14  height        1025 non-null   int64
 15  weight        1025 non-null   int64
 16  abilities     1025 non-null   object
 17  desc          1025 non-null   object
dtypes: int64(10), object(8)
memory usage: 144.3+ KB
```

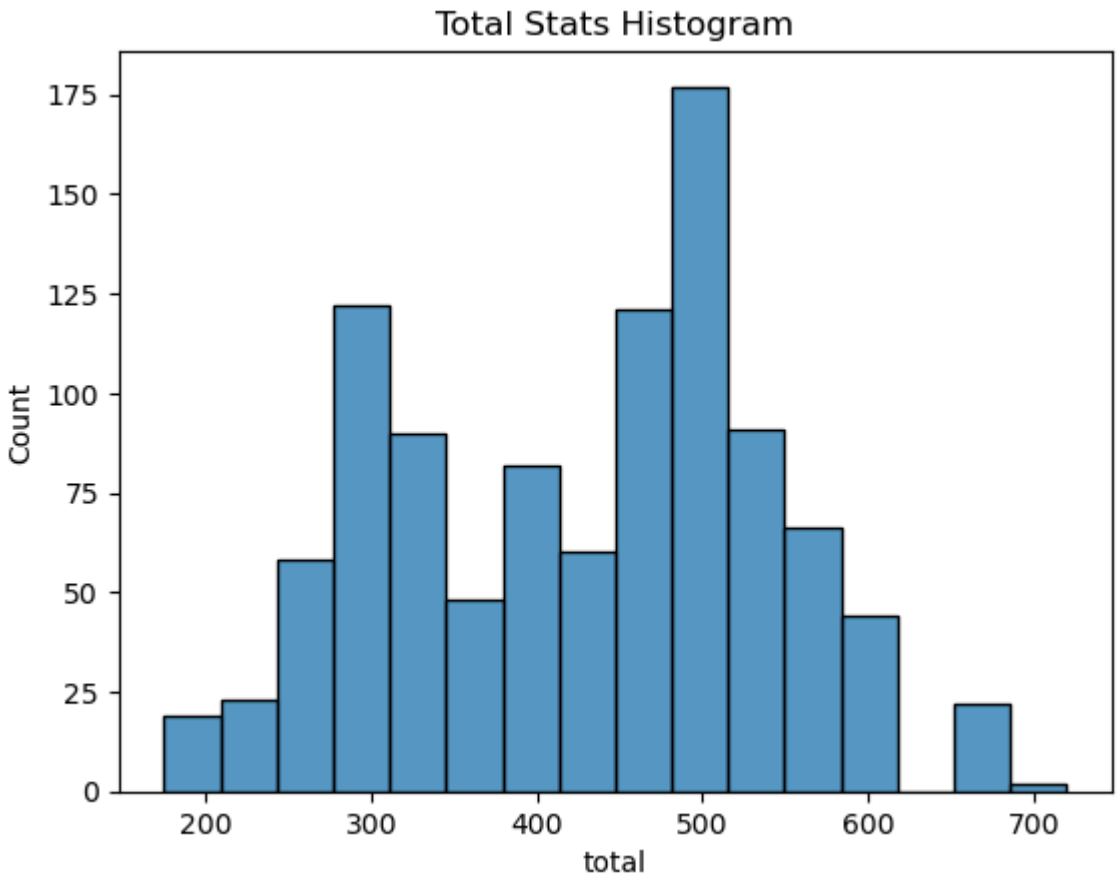
Here we realize that the only null values are in the type2 column.

In [7]:
```
df.describe()
```

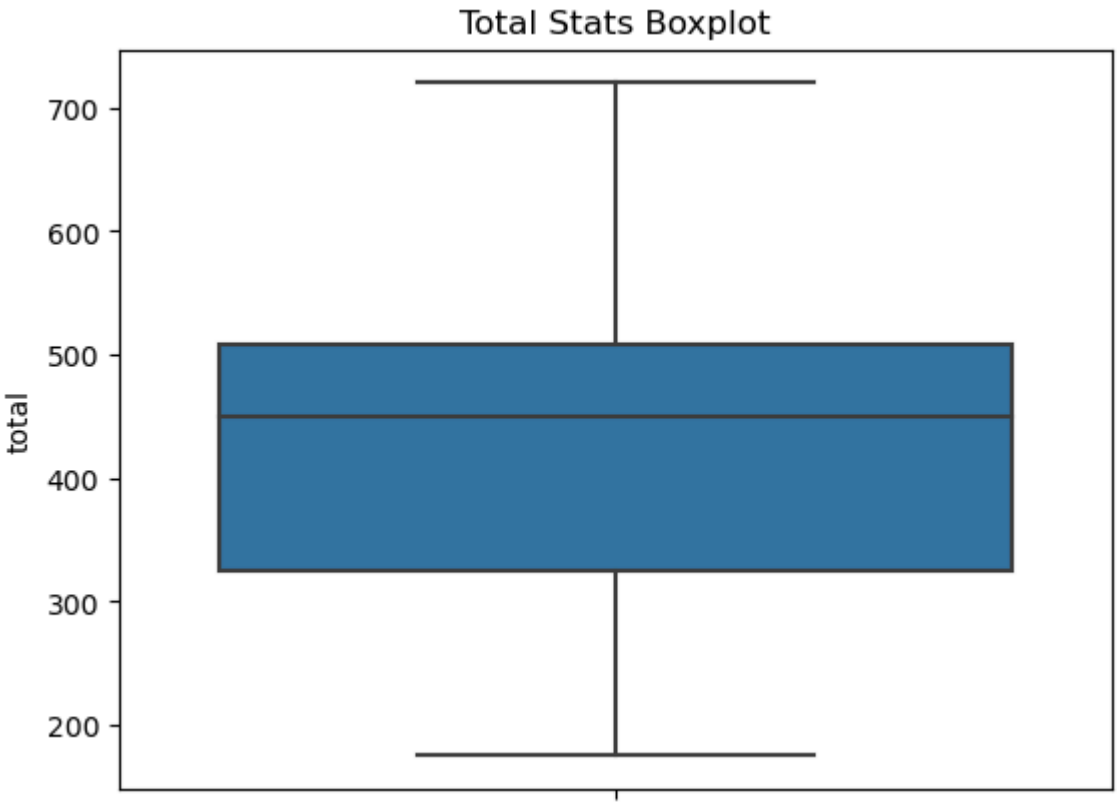
Out[7]:

	id	hp	atk	def	spatk	spdef	speed
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000
mean	513.000000	70.184390	77.521951	72.507317	70.080976	70.205854	67.186341
std	296.036315	26.631054	29.782541	29.286972	29.658378	26.639329	28.717227
min	1.000000	1.000000	5.000000	5.000000	10.000000	20.000000	5.000000
25%	257.000000	50.000000	55.000000	50.000000	47.000000	50.000000	45.000000
50%	513.000000	68.000000	75.000000	70.000000	65.000000	67.000000	65.000000
75%	769.000000	85.000000	100.000000	90.000000	90.000000	86.000000	88.000000
max	1025.000000	255.000000	181.000000	230.000000	173.000000	230.000000	200.000000

In [8]:
```
sns.histplot(data=df, x="total")
plt.title('Total Stats Histogram')
plt.show()
```



In [9]:
```
sns.boxplot(y='total', data=df)
plt.title('Total Stats Boxplot')
plt.show()
```



3. Analysis

3.1 Starter Pokemons

In [10]:
```
df_start_poke=df.iloc[:9:3,:]
```

In [11]:
```
df_start_poke
```

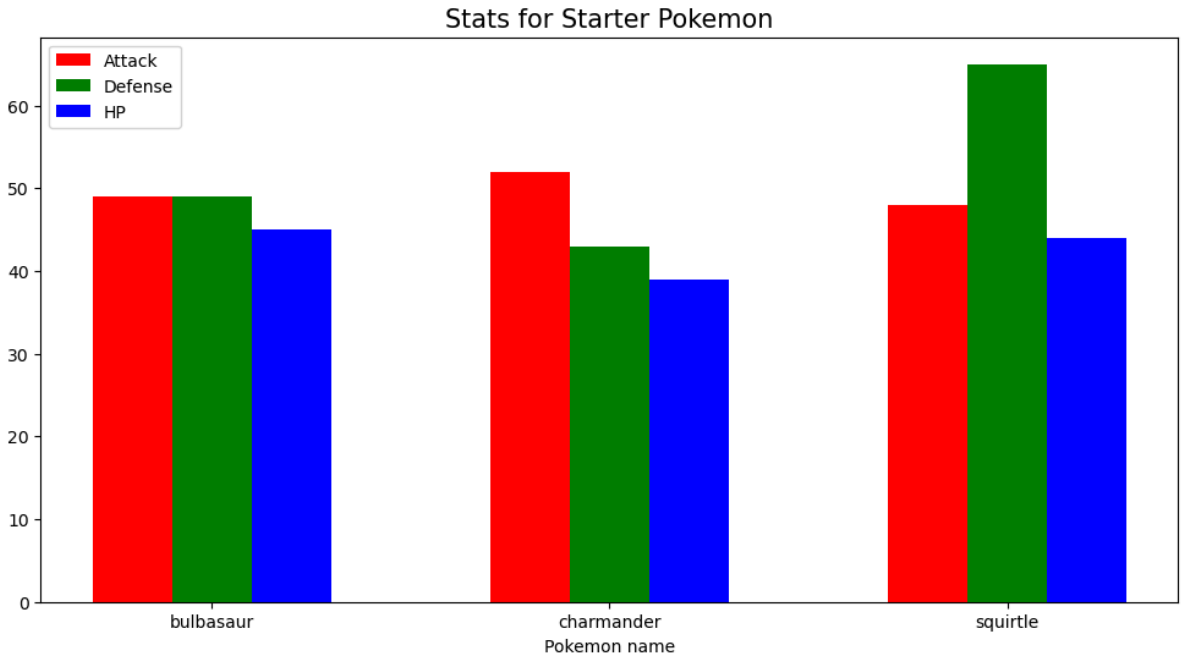
Out[11]:

	id	name	rank	generation	evolves_from	type1	type2	hp	atk	def	spatk	spdef
0	1	bulbasaur	ordinary	generation-i	nothing	grass	poison	45	49	49	65	65
3	4	charmander	ordinary	generation-i	nothing	fire	NaN	39	52	43	60	50
6	7	squirtle	ordinary	generation-i	nothing	water	NaN	44	48	65	50	64

In [12]:
```
plt.figure(figsize=(12,6))

x=np.arange(len(df_start_poke.name))
w=0.2
plt.bar(x,df_start_poke.atk, w,color='red',label='Attack')
plt.bar(x+w,df_start_poke['def'],w, color='green',label='Defense')
plt.bar(x+(2*w),df_start_poke['hp'],w, color='blue',label='HP')

plt.title('Stats for Starter Pokemon', fontsize=15)
plt.xlabel('Pokemon name')
plt.xticks(x+w, df_start_poke.name)
plt.legend()
plt.show()
```



Pokémon Selection Insight

For anyone who prioritizes Pokémon's attacking ability, **their clear choice among the three will be Charmander**.

However, within Squirtle and Bulbasaur:

- **Squirtle** should be a wiser choice given its significantly better defense system.
- The attacking ability and HP of Squirtle and Bulbasaur are almost the same.

3.2 Types of Pokemons

In [13]:
```
df.type1.unique()
```

Out[13]:
```
array(['grass', 'fire', 'water', 'bug', 'normal', 'poison', 'electric',
       'ground', 'fairy', 'fighting', 'psychic', 'rock', 'ghost', 'ice',
       'dragon', 'dark', 'steel', 'flying'], dtype=object)
```

In [14]:
```
df.type2.unique()
```

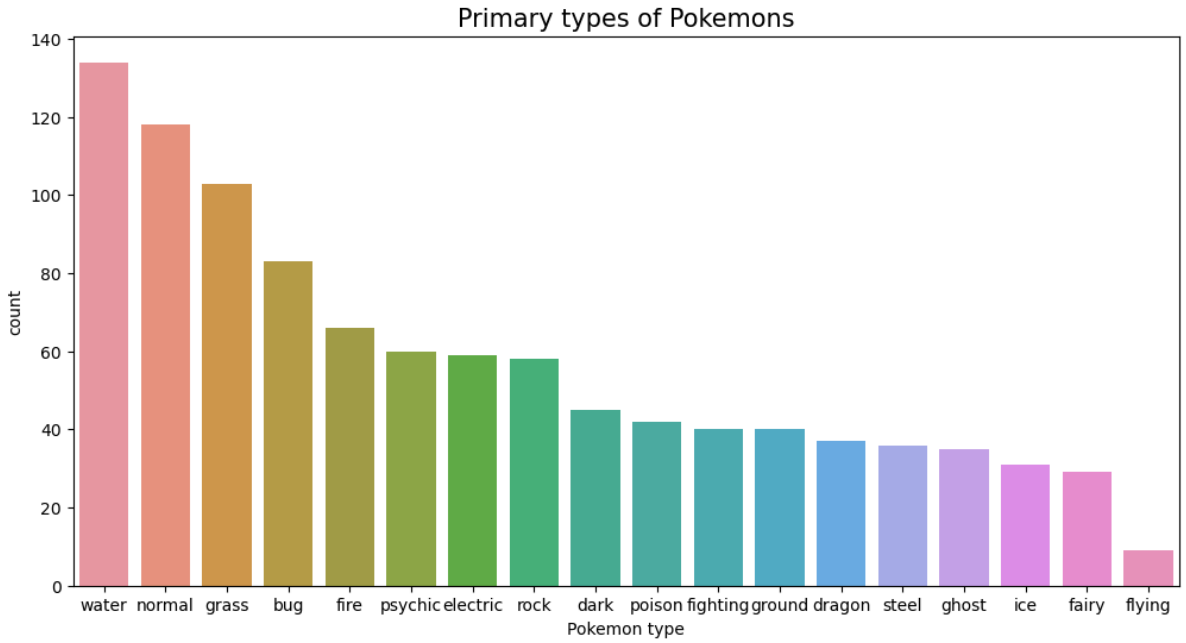
Out[14]:
```
array(['poison', nan, 'flying', 'ground', 'fairy', 'grass', 'fighting',
       'psychic', 'steel', 'ice', 'rock', 'water', 'electric', 'fire',
       'dragon', 'dark', 'ghost', 'bug', 'normal'], dtype=object)
```

3.2.1 Analysis on Primary type of Pokemon

In [15]:
```
df.type1.value_counts()
```

Out[15]:
```
type1
water       134
normal      118
grass       103
bug          83
fire         66
psychic      60
electric     59
rock         58
dark         45
poison       42
fighting     40
ground       40
dragon       37
steel        36
ghost        35
ice          31
fairy        29
flying        9
Name: count, dtype: int64
```

In [16]:
```
plt.figure(figsize=(12,6))
sns.countplot(x='type1', data=df, order=df.type1.value_counts().index)
plt.title('Primary types of Pokemons', fontsize=15)
plt.xlabel('Pokemon type');
```



In [17]:
```
#Making a df for pokemon types and their average attack, defense and total scores
columns_to_show = ['atk','def','total']
df_type_stats = df.groupby(df.type1)[columns_to_show].mean()
df_type_stats.reset_index(inplace=True)
df_type_stats
```

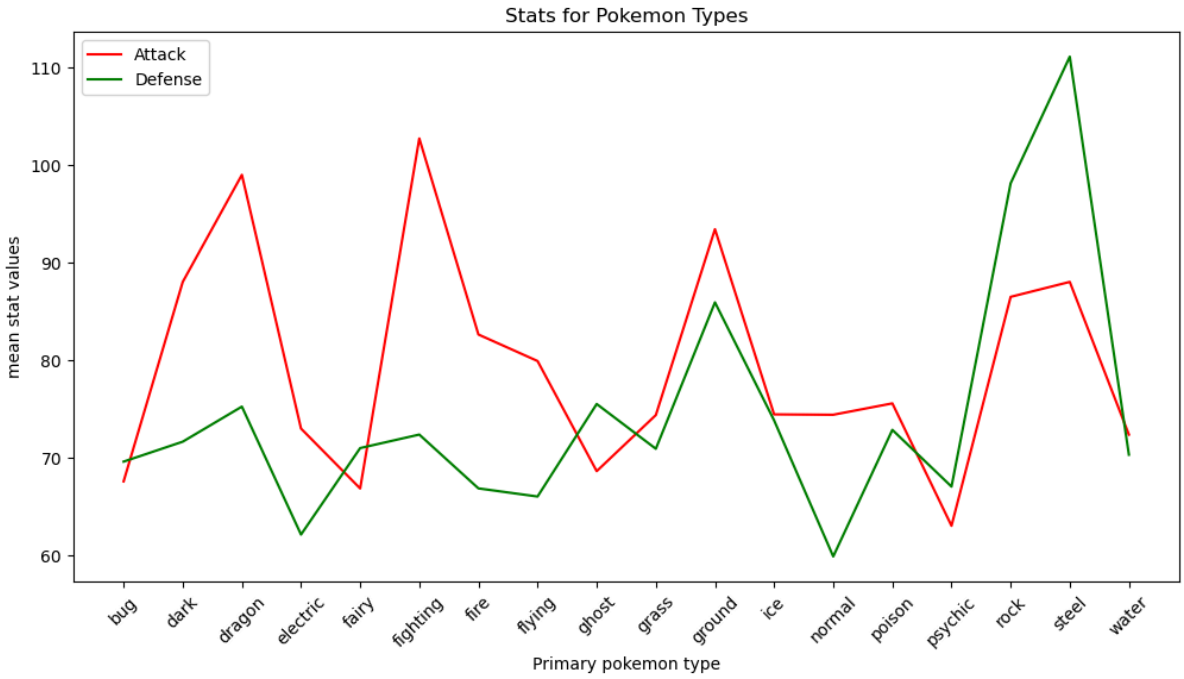
Out[17]:

	type1	atk	def	total
0	bug	67.554217	69.578313	374.638554
1	dark	88.000000	71.622222	454.733333
2	dragon	98.972973	75.216216	490.162162
3	electric	72.966102	62.101695	436.305085
4	fairy	66.827586	70.965517	436.068966
5	fighting	102.700000	72.350000	441.550000
6	fire	82.606061	66.833333	446.196970
7	flying	79.888889	66.000000	436.111111
8	ghost	68.600000	75.485714	431.171429
9	grass	74.349515	70.893204	413.116505
10	ground	93.400000	85.900000	434.575000
11	ice	74.419355	73.806452	436.387097
12	normal	74.381356	59.855932	399.838983
13	poison	75.547619	72.833333	426.333333
14	psychic	63.000000	67.016667	446.716667
15	rock	86.465517	98.103448	441.155172
16	steel	88.000000	111.083333	475.083333
17	water	72.343284	70.283582	418.865672

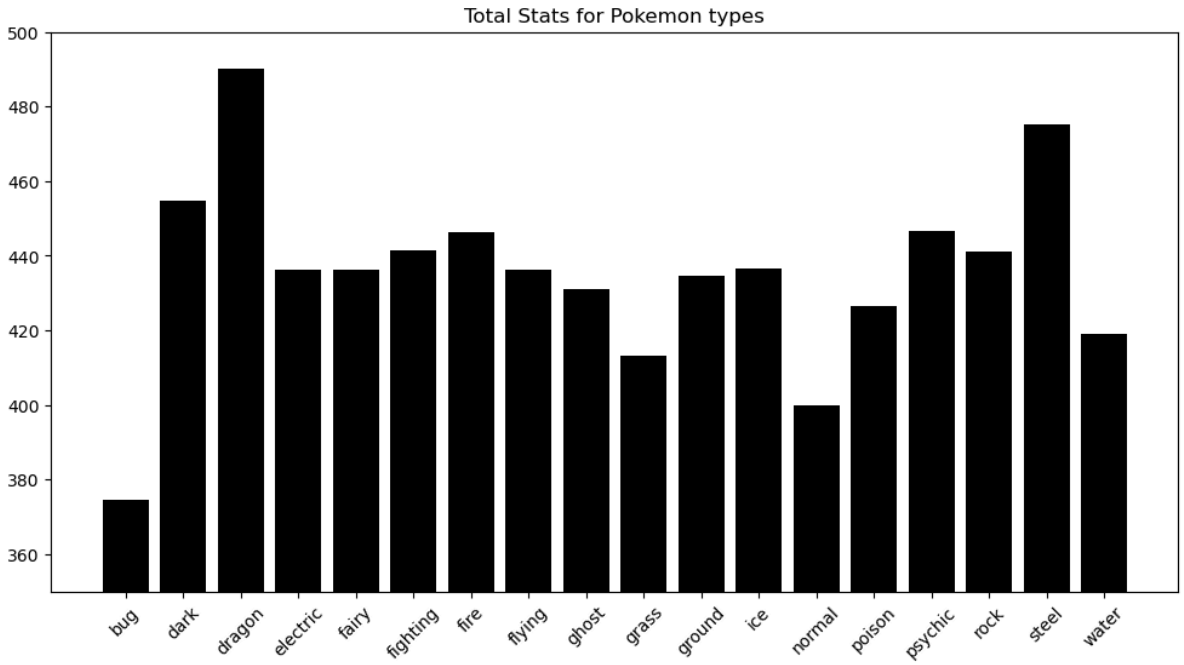
In [18]:
```
plt.figure(figsize=(12,6))

plt.plot(df_type_stats.type1,df_type_stats.atk, color='red',label='Attack')
plt.plot(df_type_stats.type1,df_type_stats['def'], color='green',label='Defense')

plt.title('Stats for Pokemon Types')
plt.ylabel('mean stat values')
plt.xlabel('Primary pokemon type')
plt.xticks(rotation=45)
plt.legend()
plt.show()
```



In [19]:
```
plt.figure(figsize=(12,6))
plt.bar(df_type_stats.type1,df_type_stats['total'], color='black',label='Total Stat
plt.title('Total Stats for Pokemon types')
plt.ylim(350,500)
plt.xticks(rotation=45)
plt.show()
```



Pokemon Type Insight

Steel and Dragon Pokémon have the highest total stats on average among the types of Pokémon.

- **Dragon Pokémon**:
  - Strength: Attacking ability
- **Steel Pokémon**:
  - Strength: Defense ability

3.2.1.1 Top Steel Pokemons

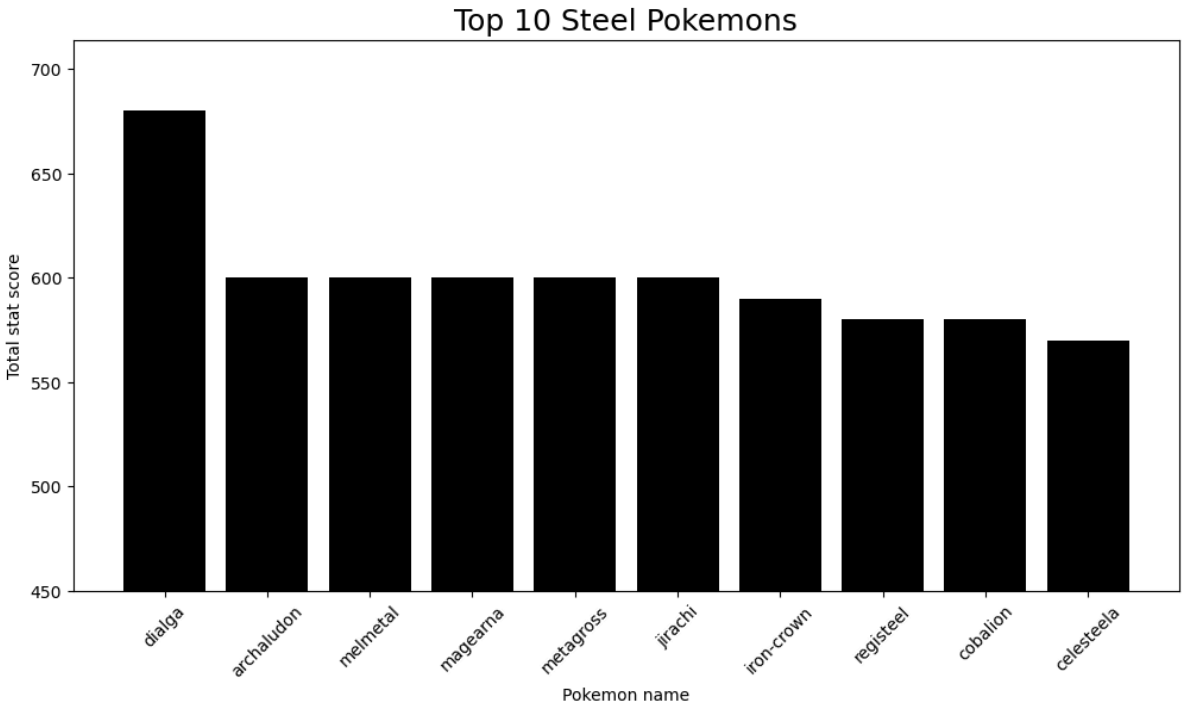
In [20]:
```
df_steel=df[df.type1=='steel']
df_steel10=df_steel.sort_values(by='total', ascending=False).head(10)
df_steel10.head(3)
```

Out[20]:

	id	name	rank	generation	evolves_from	type1	type2	hp	atk	def	spatk
482	483	dialga	legendary	generation-iv	nothing	steel	dragon	100	120	120	150
1017	1018	archaludon	ordinary	generation-ix	duraludon	steel	dragon	90	105	130	125
808	809	melmetal	mythical	generation-vii	meltan	steel	NaN	135	143	143	80

In [21]:
```
plt.figure(figsize=(12,6))
plt.bar(df_steel10.name,df_steel10['total'], color='black')

plt.xticks(rotation=45)
plt.title('Top 10 Steel Pokemons', fontsize=18)
plt.xlabel('Pokemon name')
plt.ylabel('Total stat score')
plt.ylim(450,)
plt.show()
```



3.2.1.2 Top Dragon Pokemons

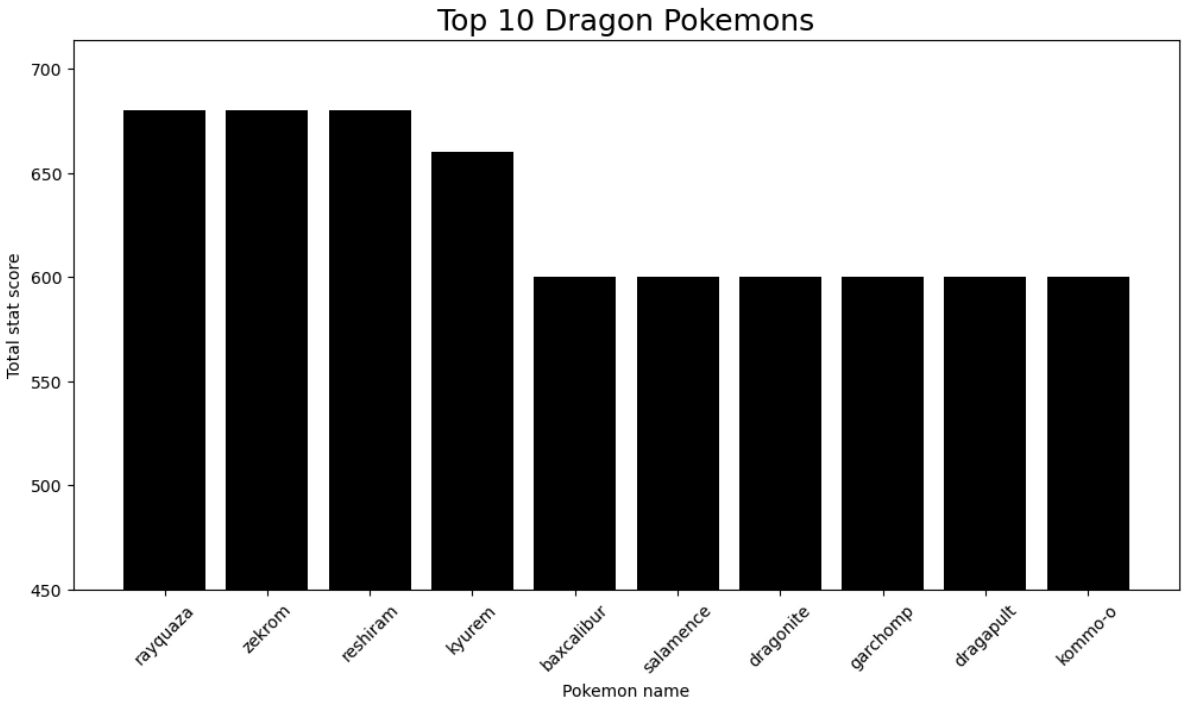
In [22]:
```
df_drag=df[df.type1=='dragon']
df_drag10=df_drag.sort_values(by=['total','atk'], ascending=False).head(10)
df_drag10.head(3)
```

Out[22]:

	id	name	rank	generation	evolves_from	type1	type2	hp	atk	def	spatk	s
383	384	rayquaza	legendary	generation-iii	nothing	dragon	flying	105	150	90	150	
643	644	zekrom	legendary	generation-v	nothing	dragon	electric	100	150	120	120	
642	643	reshiram	legendary	generation-v	nothing	dragon	fire	100	120	100	150	

In [23]:
```
plt.figure(figsize=(12,6))
plt.bar(df_drag10.name,df_drag10['total'], color='black')

plt.xticks(rotation=45)
plt.title('Top 10 Dragon Pokemons', fontsize=18)
plt.xlabel('Pokemon name')
plt.ylabel('Total stat score')
plt.ylim(450,)
plt.show()
```

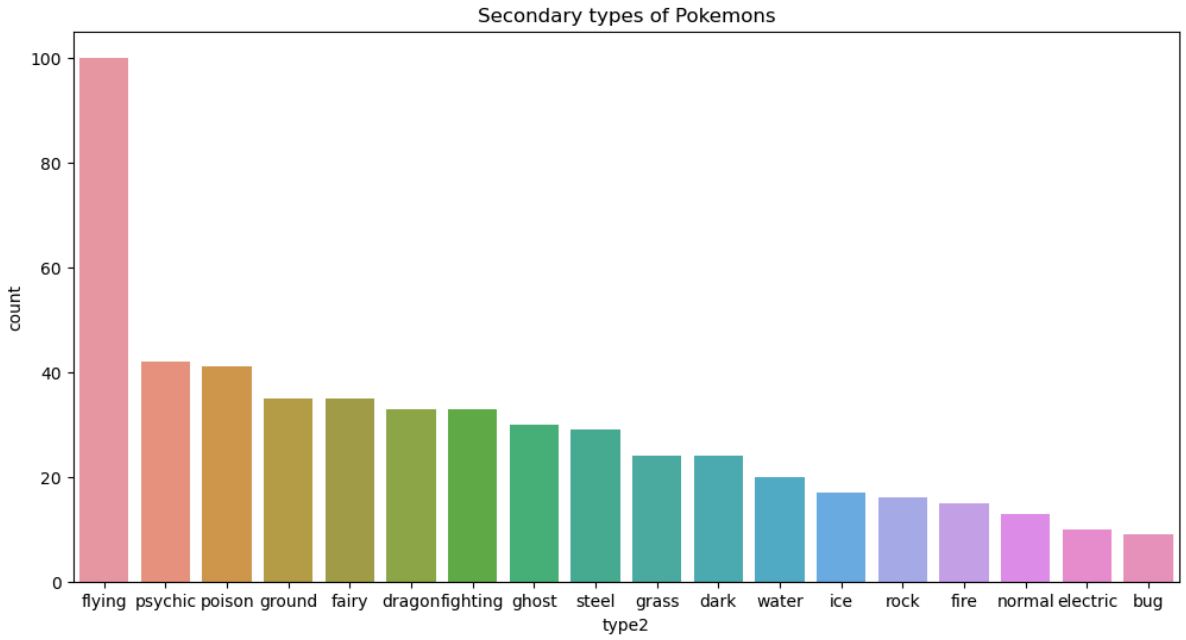


3.2.2 Analysis on secondary or combined type of Pokemon

In [24]:
```
df.type2.value_counts()
```

Out[24]:
```
type2
flying      100
psychic      42
poison       41
ground       35
fairy        35
dragon       33
fighting     33
ghost        30
steel        29
grass        24
dark         24
water        20
ice          17
rock         16
fire         15
normal       13
electric     10
bug           9
Name: count, dtype: int64
```

In [25]:
```
plt.figure(figsize=(12,6))
sns.countplot(x='type2',data=df, order=df.type2.value_counts().index)
plt.title('Secondary types of Pokemons');
```

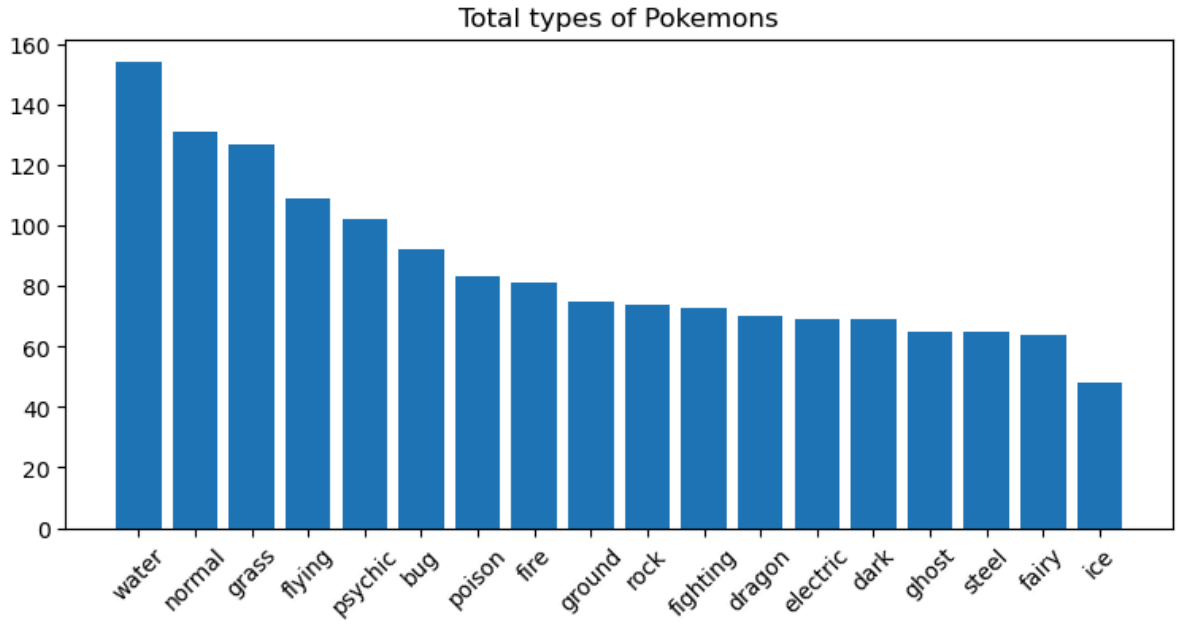


In [26]:
```
total_type_counts=pd.DataFrame(df.type2.value_counts()+df.type1.value_counts())
tt_counts=total_type_counts.sort_values('count',ascending=False)
tt_counts.reset_index(inplace=True)
tt_counts.columns = ['type', 'count']
tt_counts
```

Out[26]:

	type	count
0	water	154
1	normal	131
2	grass	127
3	flying	109
4	psychic	102
5	bug	92
6	poison	83
7	fire	81
8	ground	75
9	rock	74
10	fighting	73
11	dragon	70
12	electric	69
13	dark	69
14	ghost	65
15	steel	65
16	fairy	64
17	ice	48

In [27]:
```
plt.figure(figsize=(9,4))
plt.bar(tt_counts['type'],tt_counts['count'])
plt.title('Total types of Pokemons')
plt.xticks(rotation=45)
plt.show()
```



Most common and rare pokemon Insight

Given the types of Pokémon, regardless of their primary or secondary type, **the most common Pokémon types are Water, Normal, and Grass**.

However, **the rarest types of Pokémon are Ice, Fairy, and Steel**.

Hence, if one wants an Ice Pokémon in their Pokémon portfolio and comes across it, **they shouldn't let it go**.

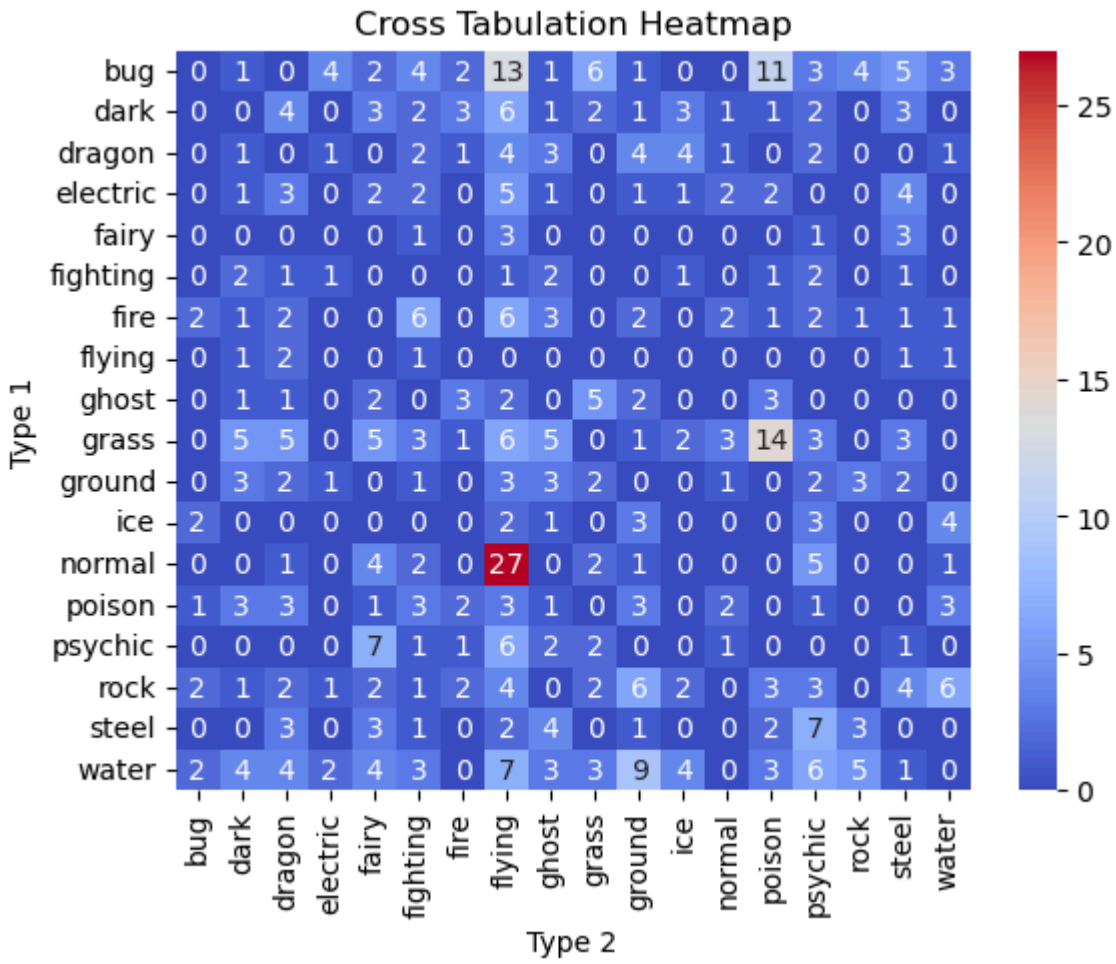
In [28]:
```
#Finding out the most common types of the pokemon that have both primary and secona
crosstab=pd.crosstab(df.type1,df.type2)
crosstab
```

Out[28]:

type1 \ type2	bug	dark	dragon	electric	fairy	fighting	fire	flying	ghost	grass	ground	ice	nor
bug	0	1	0	4	2	4	2	13	1	6	1	0	
dark	0	0	4	0	3	2	3	6	1	2	1	3	
dragon	0	1	0	1	0	2	1	4	3	0	4	4	
electric	0	1	3	0	2	2	0	5	1	0	1	1	
fairy	0	0	0	0	0	1	0	3	0	0	0	0	
fighting	0	2	1	1	0	0	0	1	2	0	0	1	
fire	2	1	2	0	0	6	0	6	3	0	2	0	
flying	0	1	2	0	0	1	0	0	0	0	0	0	
ghost	0	1	1	0	2	0	3	2	0	5	2	0	
grass	0	5	5	0	5	3	1	6	5	0	1	2	
ground	0	3	2	1	0	1	0	3	3	2	0	0	
ice	2	0	0	0	0	0	0	2	1	0	3	0	
normal	0	0	1	0	4	2	0	27	0	2	1	0	
poison	1	3	3	0	1	3	2	3	1	0	3	0	
psychic	0	0	0	0	7	1	1	6	2	2	0	0	
rock	2	1	2	1	2	1	2	4	0	2	6	2	
steel	0	0	3	0	3	1	0	2	4	0	1	0	
water	2	4	4	2	4	3	0	7	3	3	9	4	

In [29]:
```
sns.heatmap(crosstab, annot=True, cmap='coolwarm', fmt='d')
plt.xlabel('Type 2')
plt.ylabel('Type 1')
plt.title('Cross Tabulation Heatmap')
```

Out[29]:
```
Text(0.5, 1.0, 'Cross Tabulation Heatmap')
```



Among the double type pokemons, the **highest amount of pokemons belong to the normal flying category** followed by the grass poison type

In [30]:
```
highest_value = crosstab.max().max()
highest_value
```

Out[30]:
```
27
```

In [31]:
```
highest_indices = crosstab.stack()[crosstab.stack() == highest_value].index.to_list
highest_indices
```

Out[31]:
```
[('normal', 'flying')]
```

In [32]:
```
df[(df.type1 == 'normal') & (df.type2 == 'flying')].head()
```

Out[32]:

	id	name	rank	generation	evolves_from	type1	type2	hp	atk	def	spatk	spdef
15	16	pidgey	ordinary	generation-i	nothing	normal	flying	40	45	40	35	35
16	17	pidgeotto	ordinary	generation-i	pidgey	normal	flying	63	60	55	50	50
17	18	pidgeot	ordinary	generation-i	pidgeotto	normal	flying	83	80	75	70	70
20	21	spearow	ordinary	generation-i	nothing	normal	flying	40	60	30	31	31
21	22	fearow	ordinary	generation-i	spearow	normal	flying	65	90	65	61	61

3.3 Top 10 pokemons

In [33]:
```
df_top10 = df.sort_values('total', ascending=False).head(10)
df_top10.head()
```

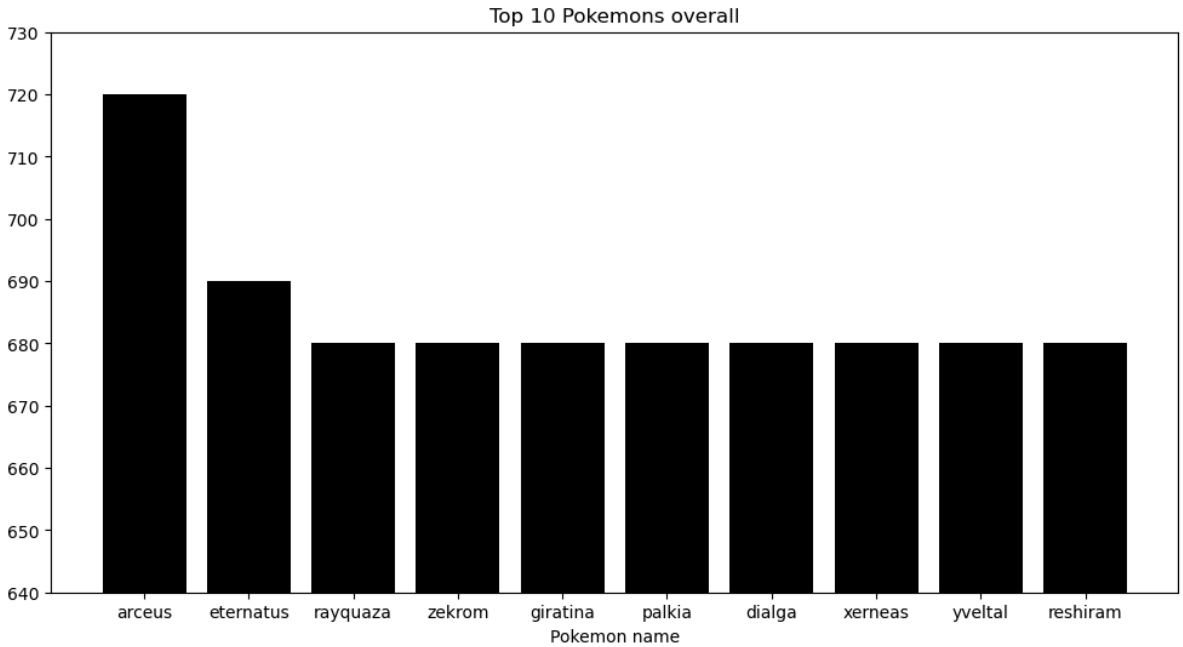
Out[33]:

	id	name	rank	generation	evolves_from	type1	type2	hp	atk	def	spatk	s
492	493	arceus	mythical	generation-iv	nothing	normal	NaN	120	120	120	120	
889	890	eternatus	legendary	generation-viii	nothing	poison	dragon	140	85	95	145	
383	384	rayquaza	legendary	generation-iii	nothing	dragon	flying	105	150	90	150	
643	644	zekrom	legendary	generation-v	nothing	dragon	electric	100	150	120	120	
486	487	giratina	legendary	generation-iv	nothing	ghost	dragon	150	100	120	100	

In [34]:

```
plt.figure(figsize=(12,6))
plt.bar(df_top10.name,df_top10['total'], color='black')

plt.title('Top 10 Pokemons overall')
plt.xlabel('Pokemon name')
plt.ylim(640, df_top10.total.max()+10)
plt.show()
```



In [35]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 18 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   id            1025 non-null   int64 
 1   name          1025 non-null   object
 2   rank          1025 non-null   object
 3   generation    1025 non-null   object
 4   evolves_from  1025 non-null   object
 5   type1         1025 non-null   object
 6   type2         526 non-null    object
 7   hp            1025 non-null   int64 
 8   atk           1025 non-null   int64 
 9   def           1025 non-null   int64 
 10  spatk         1025 non-null   int64 
 11  spdef         1025 non-null   int64 
 12  speed         1025 non-null   int64 
 13  total         1025 non-null   int64 
 14  height        1025 non-null   int64 
 15  weight        1025 non-null   int64 
 16  abilities     1025 non-null   object
 17  desc          1025 non-null   object
dtypes: int64(10), object(8)
memory usage: 144.3+ KB
```

3.4 Legendary Pokemon Analysis

In [36]:
```
df.rename(columns={'rank': 'pokerank'}, inplace=True)
```

In [37]:
```
df.columns
```

Out[37]:
```
Index(['id', 'name', 'pokerank', 'generation', 'evolves_from', 'type1',
       'type2', 'hp', 'atk', 'def', 'spatk', 'spdef', 'speed', 'total',
       'height', 'weight', 'abilities', 'desc'],
      dtype='object')
```

In [38]:
```
df.pokerank.unique()
```

Out[38]:
```
array(['ordinary', 'legendary', 'mythical', 'baby'], dtype=object)
```

In [39]:
```
df.pokerank.value_counts()
```

Out[39]:
```
pokerank
ordinary     913
legendary     70
mythical      23
baby          19
Name: count, dtype: int64
```

Grouping by and checking stats for these pokemons

In [40]:
```
df_rank=df.groupby(df.pokerank)[columns_to_show].mean()
df_rank
```

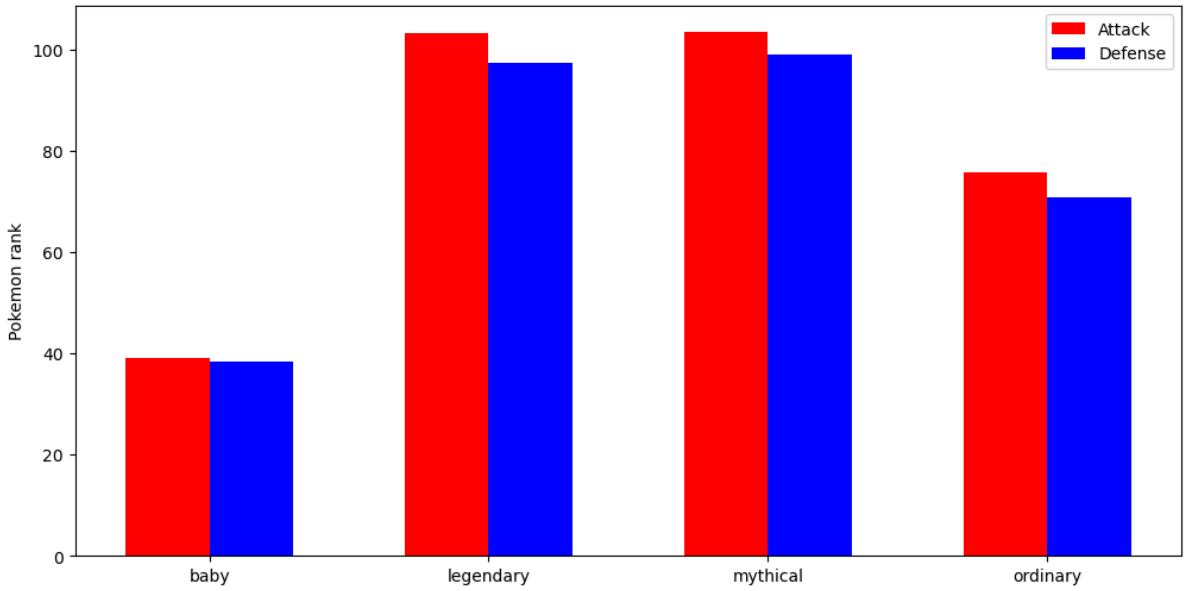
Out[40]:

pokerank	atk	def	total
baby	39.157895	38.421053	274.473684
legendary	103.214286	97.342857	594.714286
mythical	103.347826	98.913043	586.086957
ordinary	75.699890	70.647317	414.078861

In [41]:
```
plt.figure(figsize=(12,6))

w=0.3
x=np.arange(len(df_rank.index))
plt.bar(x, df_rank.atk, w, color='red', label='Attack')
plt.bar(x+w, df_rank['def'], w, color='blue', label='Defense')

plt.xticks(x+(w/2), df_rank.index)
plt.ylabel('Pokemon rank')
plt.legend()
plt.show;
```

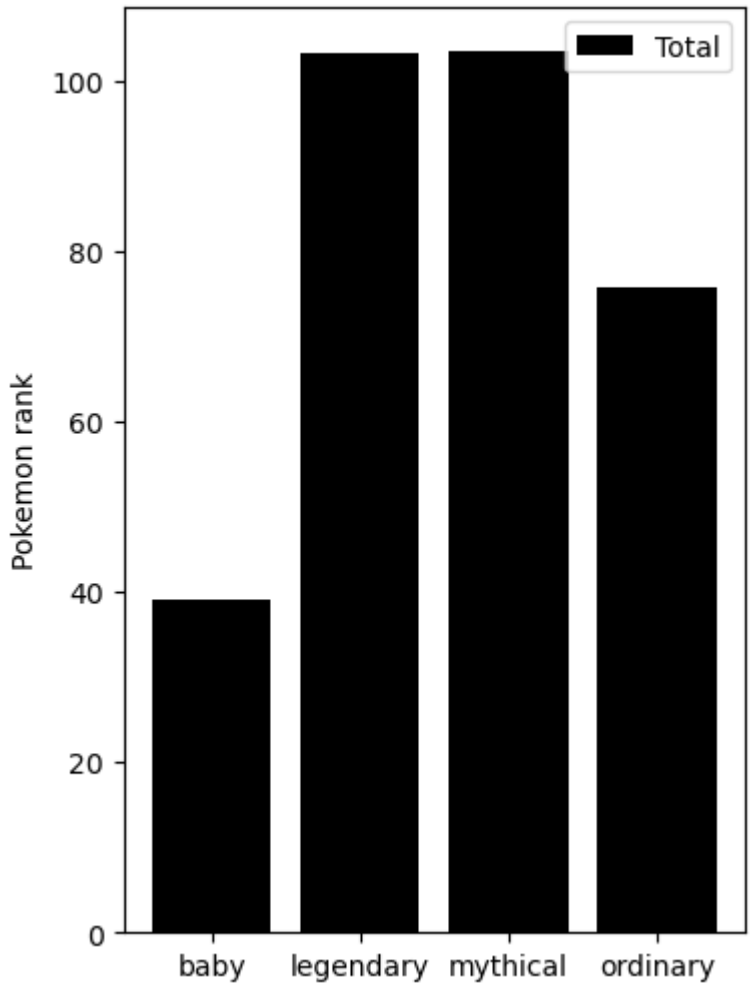


In [42]:
```
plt.figure(figsize=(4,6))

plt.bar(df_rank.index,df_rank.atk, color='black', label='Total')


plt.ylabel('Pokemon rank')
plt.legend()
plt.show
```

Out[42]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [43]:
```
ratio = df_rank.loc['legendary', 'total'] / df_rank.loc['ordinary', 'total']
ratio
```

Out[43]:
```
1.4362343550316694
```

Analysis Focus: Legendary Pokémon

For the focus of our analysis, we will be analyzing the **Legendary Pokémon**.

- **Legendary Pokémon** are unique, meaning there is only one of each kind.
- Legendary Pokémon have very high attack and defense stats.

As we see above, legendary Pokémon have **1.4x more total stats on average** compared to ordinary Pokémon.

Hence, every Pokémon player should own at least one Legendary Pokémon.

3.4.1 Top 10 Legendary Pokemon

In [44]:
```
df_leg=df[df.pokerank == 'legendary']
df_leg.head()
```

Out[44]:

	id	name	pokerank	generation	evolves_from	type1	type2	hp	atk	def	spatk	sp
143	144	articuno	legendary	generation-i	nothing	ice	flying	90	85	100	95	
144	145	zapdos	legendary	generation-i	nothing	electric	flying	90	90	85	125	
145	146	moltres	legendary	generation-i	nothing	fire	flying	90	100	90	125	
149	150	mewtwo	legendary	generation-i	nothing	psychic	NaN	106	110	90	154	
242	243	raikou	legendary	generation-ii	nothing	electric	NaN	90	85	75	115	

In [45]:
```
df_leg10 = df_leg.sort_values('total', ascending=False).head(10)
df_leg10.head()
```

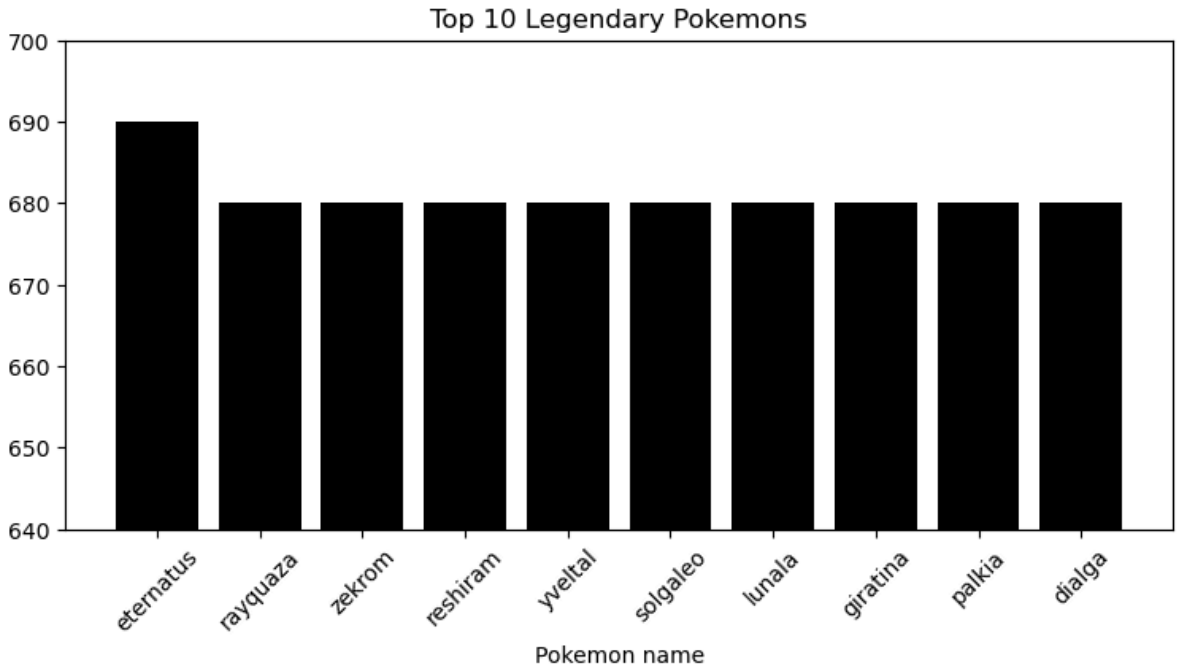
Out[45]:

	id	name	pokerank	generation	evolves_from	type1	type2	hp	atk	def	spatk	s
889	890	eternatus	legendary	generation-viii	nothing	poison	dragon	140	85	95	145	
383	384	rayquaza	legendary	generation-iii	nothing	dragon	flying	105	150	90	150	
643	644	zekrom	legendary	generation-v	nothing	dragon	electric	100	150	120	120	
642	643	reshiram	legendary	generation-v	nothing	dragon	fire	100	120	100	150	
716	717	yveltal	legendary	generation-vi	nothing	dark	flying	126	131	95	131	

In [46]:

```
plt.figure(figsize=(9,4))
plt.bar(df_leg10.name,df_leg10['total'], color='black')

plt.title('Top 10 Legendary Pokemons')
plt.xlabel('Pokemon name')
plt.ylim(640, df_leg10.total.max()+10)
plt.xticks(rotation=45)
plt.show()
```



3.4.2 Legendary Pokemon per type

In [47]:
```
df_leg.sort_values(by=['atk','total'], ascending=False).head()
```

Out[47]:

	id	name	pokerank	generation	evolves_from	type1	type2	hp	atk	def	spatk	s
485	486	regigigas	legendary	generation-iv	nothing	normal	NaN	110	160	110	80	
383	384	rayquaza	legendary	generation-iii	nothing	dragon	flying	105	150	90	150	
643	644	zekrom	legendary	generation-v	nothing	dragon	electric	100	150	120	120	
382	383	groudon	legendary	generation-iii	nothing	ground	NaN	100	150	140	100	
895	896	glastrier	legendary	generation-viii	nothing	ice	NaN	100	145	130	65	

In [48]:
```
legen=df_leg.type1.value_counts()
legen
```

Out[48]:
```
type1
psychic     13
dragon       8
electric     6
dark         5
poison       4
fire         4
water        4
fighting     4
grass        3
fairy        3
ice          3
normal       3
steel        3
ghost        2
rock         2
ground       2
flying       1
Name: count, dtype: int64
```

In [49]:
```
df_leg_type=pd.DataFrame(legen)
```

In [50]:
```
df_leg_type
```

Out[50]:

type1	count
psychic	13
dragon	8
electric	6
dark	5
poison	4
fire	4
water	4
fighting	4
grass	3
fairy	3
ice	3
normal	3
steel	3
ghost	2
rock	2
ground	2
flying	1

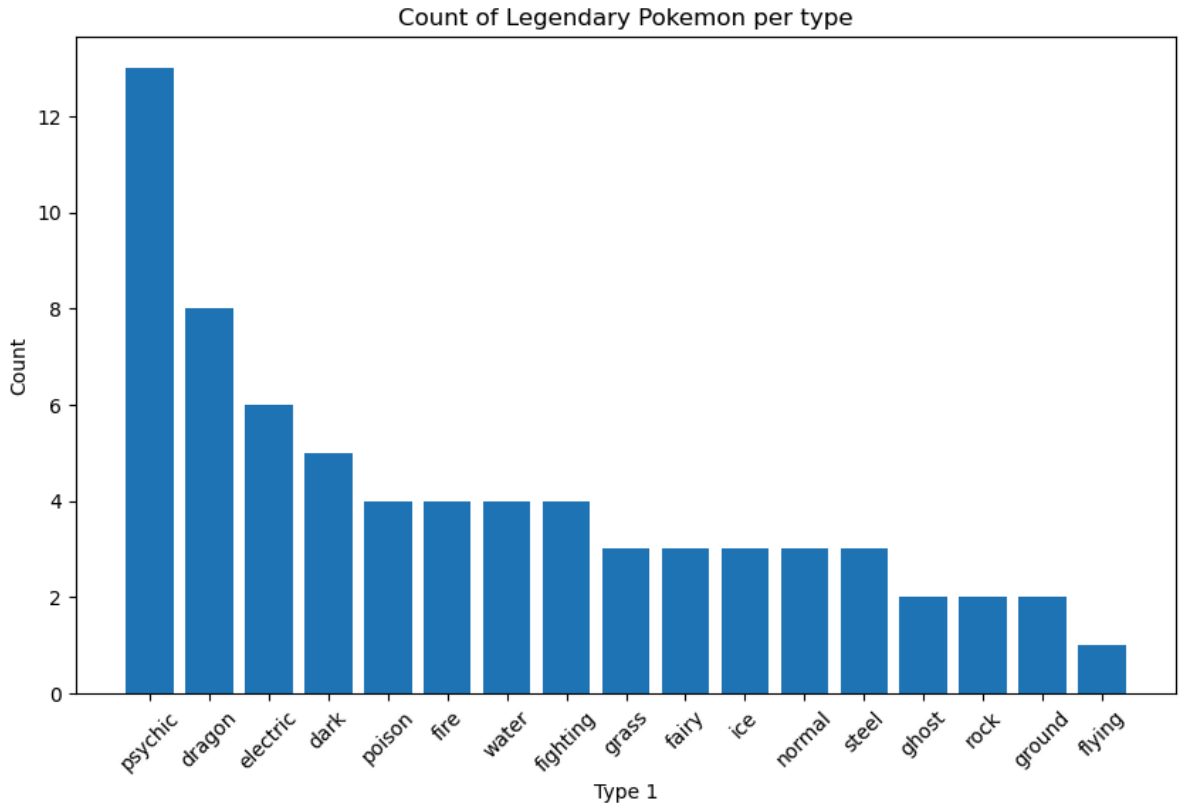
In [51]:
```
df_leg_type.reset_index(inplace=True)
```

In [52]:
```
df_leg_type.columns
```

Out[52]:
```
Index(['type1', 'count'], dtype='object')
```

In [53]:
```
plt.figure(figsize=(10,6))
plt.bar(df_leg_type.type1,df_leg_type['count'])

plt.xlabel('Type 1')
plt.ylabel('Count')
plt.title('Count of Legendary Pokemon per type')
plt.xticks(rotation=45);
```



In [54]:
```
#Legendary pokemon types by stats
df_leg.groupby(df_leg.type1)[columns_to_show].mean().sort_values(by=['total','atk',
```

Out[54]:

type1	atk	def	total
fairy	122.0	93.333333	640.0
ground	137.5	115.000000	635.0
dragon	115.0	92.625000	635.0
ghost	82.5	90.000000	630.0
water	92.5	105.000000	625.0