Women Representation in City Property of San Francisco, an Exploratoration

Welcome! This notebook will follow the process of initially exploring a dataset regarding the gender distribution of monuments in San Francisco. Let's start by loading in the packages we will need.

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

Now let's load in our data and get it's dimensions.

In [2]:
```
df = pd.read_csv("C:/Users/IB/Desktop/website/projects/sf_women_statues/WomenRepresentaionInCityPropert
y-SanFrancisco.csv")
df.shape
```

Out[2]:
```
(82, 11)
```

Let's take a look at the important characteristics of each column.

In [3]:
```
col_chr = pd.DataFrame({"Data Type":df.dtypes,
                        "NA's":df.isnull().sum(),
                        "Unique Values":df.nunique()})
col_chr
```

Out[3]:

	Data Type	NA's	Unique Values
Department/Source	object	0	8
Name	object	0	82
Person	object	1	69
Gender	object	0	5
Reference	object	28	21
Comments	object	50	13
Current Police Districts	int64	0	1
Current Supervisor Districts	int64	0	1
Analysis Neighborhoods	int64	0	1
Neighborhoods	int64	0	1
SF Find Neighborhoods	int64	0	1

Note that the final 5 columns consist of only a single possible observation, with no NA values. Let's print the single observations for each of these variables.

In [4]:
```
df[["Current Police Districts","Current Supervisor Districts",
   "Analysis Neighborhoods","Neighborhoods","SF Find Neighborhoods"]].loc[1]
```

Out[4]:
```
Current Police Districts         4
Current Supervisor Districts    10
Analysis Neighborhoods          36
Neighborhoods                   21
SF Find Neighborhoods           21
Name: 1, dtype: int64
```

Now let's rename our columns and drop the uninteresting variables.

In [5]:
```
df_final = df.drop(df.columns[6:11],axis=1).rename(
    columns={"Department/Source":"dep_source",
             "Name":"name","Person":"person",
             "Gender":"gender","Reference":"ref",
             "Comments":"comments"}) \
             .replace({"gender":{"M & F":"F & M"}})
df_final.head()
```

Out[5]:

	dep_source	name	person	gender	ref	comments
0	Administrator	MOSCONE CENTER (South)	George R. Moscone	M	City Administrator	NaN
1	Administrator	Maxine Hall Health Center	NaN	F	Public Health	NaN
2	REC AND PARKS	Moscone Recreation Center	George R. Moscone	M	NaN	park
3	REC AND PARKS	Helen Crocker Russell Library of Horticulture,...	Helen Crocker	F	NaN	facilities and other amenities
4	REC AND PARKS	Sharon Building, Golden Gate Park	Sharon Building	M	NaN	facilities and other amenities

Let's look at the columns in the order they are presented here. So, we will start with the Department column. Note that since this dataset focuses on gender, we can include a breakdown of each column with the associated genders they represent.

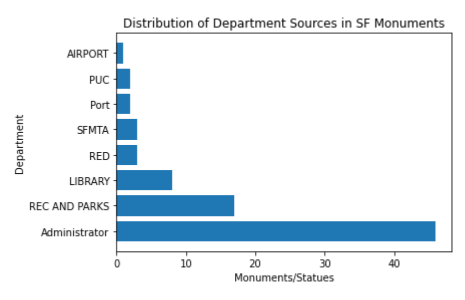
In [6]:
```
col_chr.loc["Department/Source"]
```

Out[6]:
```
Data Type         object
NA's                   0
Unique Values          8
Name: Department/Source, dtype: object
```

With 8 unique values and no NA's, we can see that the important question is how these observations are distributed.

In [7]:
```
dep = df_final.dep_source.value_counts()
dep = dep.to_frame()
plt.ylabel('Department')
plt.xlabel('Monuments/Statues')
plt.title('Distribution of Department Sources in SF Monuments')
plt.barh(dep.index,dep.dep_source)
```

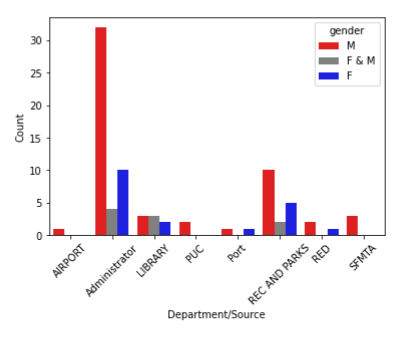
Out[7]:
```
<BarContainer object of 8 artists>
```



Now let's check out the gender breakdown for the Department/Source column.

In [8]:
```
dep_gender2=df_final.replace({"gender":{"M & M":"M"}}) \
                    .groupby(["dep_source","gender"]).size() \
                    .to_frame().reset_index() \
                    .rename(columns={0:"Count","dep_source":"Department/Source"})
plt.xticks(rotation=45)
sns.barplot(x="Department/Source",y="Count",hue="gender",
            data=dep_gender2,
            hue_order=["M","F & M","F"],
           palette=["Red","grey","blue"])
```

Out[8]:
```
<AxesSubplot:xlabel='Department/Source', ylabel='Count'>
```



Now let's look at the "name" column, which describes the location of each monument.

In [9]:
```
col_chr.loc["Name"]
```

Out[9]:
```
Data Type         object
NA's                   0
Unique Values         82
Name: Name, dtype: object
```

With 82 unique values, each observation is a unique string. This column would perhaps be useful with a locational dataset to merge with, but on it's own doesn't give us much information.

As such, let's continue on to the next column; the "person" column, which denotes who the statue is of.

In [10]:
```
col_chr.loc["Person"]
```

Out[10]:
```
Data Type         object
NA's                   1
Unique Values         69
Name: Person, dtype: object
```

Let's take a look at how many times various individuals are honored with a statue. The "person" column denotes who the subject is, so we will create a frequency histogram of said column. However, note that there are some statues of multiple people, so it is important to code a solution with this in mind. Here we will begin by revealing all the multi-person statues, and the individual statues that also honor those individuals.

In [11]:
```
multiples = df_final.loc[(df_final['gender']=="F & M")|
                         (df_final['gender']=="M & F")|
                         (df_final['gender']=="M & M")] \
                    .person.str.split(" and | & ",expand=True)
multiples=multiples.reset_index()
multiples=multiples[[0,1]]
multiples.columns = ["person_1","person_2"]
multiples.iloc[1].person_1 = "Minnie Ward"
multiples.iloc[2].person_1 = "Charlotte Shultz"
multiples.iloc[3].person_1 = "Walter Hass"
multiples.iloc[4].person_2 = "Syncip Family"
multiples.iloc[5].person_2 = "Koret (Family?)"
multiples.iloc[6].person_1 = "Herman Herbst"
multiples.iloc[7].person_2 = "Fulton (family?)"
multiples.iloc[8].person_1 = "Dianne Taube"
multiples.iloc[9].person_1 = "Thelma Doelger"
multiples=multiples.person_1.append(multiples.person_2).reset_index(drop=True)
multiples="|".join(multiples)
df_final[df_final['person'].str.contains(multiples,na=False)]
```

```
c:\users\ib\appdata\local\programs\python\python38\lib\site-packages\pandas\core\strings.py:1954: Use
rWarning: This pattern has match groups. To actually get the groups, use str.extract.
  return func(self, *args, **kwargs)
```

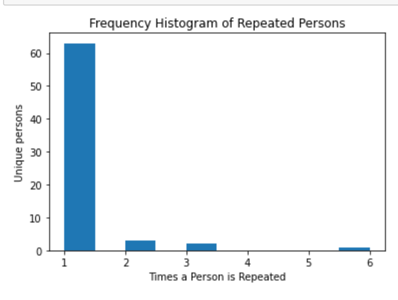
Out[11]:

	dep_source	name	person	gender	ref	comments
5	Administrator	Priscilla Chan and Mark Zuckerberg San Francis...	Priscilla Chan and Mark Zuckerberg	F & M	Public Health	NaN
7	REC AND PARKS	Minnie & Lovie Ward Recreation Center	Minnie & Lovie Ward	F & M	NaN	park
26	Administrator	Charlotte and George Shultz Horseshoe Drive	Charlotte and George Shultz	F & M	War Memorial Opera House	NaN
30	Administrator	Walter and Elise Haas Grand Lounge	Walter and Elise Hass	F & M	Davies Symphony Hall	NaN
34	LIBRARY	Syncip Family Conference Room	Syncip Family	F & M	NaN	4th floor
58	Administrator	Veterans Building - Herbst Theater	Herman and Maurice Herbst	M & M	Veterans Building	NaN
65	Administrator	Dianne and Tab Taube Atrium Theatre	Dianne and Tab Taube	F & M	Veterans Building	NaN
66	REC AND PARKS	Thelma and Henry Doelger Primate Discovery Cen...	Thelma and Henry Doelger	F & M	NaN	facilities and other amenities

We have looked through every name in the person column, and it seems that no person that is the subject of a multi-individual statue is also the subject of a single subject statue. This makes things much easier, but we still have to decide whether to treat these multi-person statues as two separate observations or as a single. Since many of these people multi-person statues are of two partners, or of an unknown number of people under the denotion of a family, it is a wise choice to treat each multi-statue monument as a single individual instead of portioning into each unique individual.

So, let us plow ahead and look at the distribution of repeated individuals in San Francisco Monuments.

In [12]:
```
person_ftable = df_final.person.value_counts().to_frame()
plt.hist(person_ftable.person, density=False)
plt.xlabel('Times a Person is Repeated')
plt.ylabel('Unique persons')
plt.title('Frequency Histogram of Repeated Persons')
plt.show()
```



One person was embodied 6 times in various monuments. Let's see who it was.

In [13]:
```
person_ftable.loc[person_ftable["person"]==6].index[0]
```

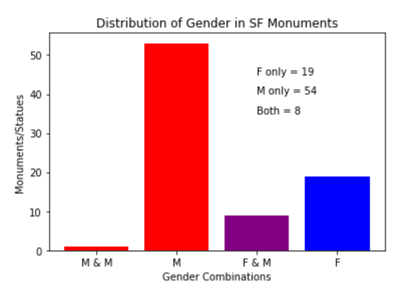
Out[13]:
```
'George R. Moscone'
```

George Moscone was the mayor of San Francisco from 1976 until 1978 when he was assassinated. It makes sense that he would be honored more than the average individual.

Let's look at the distribution of genders in monuments. Note that there is a subset of statues that include more than one person, specifically those with two males or one of each male and female subjects.

In [14]:
```
gender_ftable = df_final.gender.value_counts().to_frame().reset_index().replace({"index":{"M & F":"F &
 M"}}).groupby("index",sort = False).sum()
x_axis_order = ["M & M", "M", "F & M", "F"]
plt.xlabel('Gender Combinations')
plt.ylabel('Monuments/Statues')
plt.title('Distribution of Gender in SF Monuments')
plt.text("F & M",40,"M only = 54",fontsize=10)
plt.text("F & M",45,"F only = 19")
plt.text("F & M",35,"Both = 8")
plt.bar(gender_ftable.loc[x_axis_order].index,
        gender_ftable.gender.loc[x_axis_order],
        color = ['red', 'red', 'purple', 'blue'])
```

Out[14]:
```
<BarContainer object of 4 artists>
```



Here we can see that there are more statues of men than there are of women.

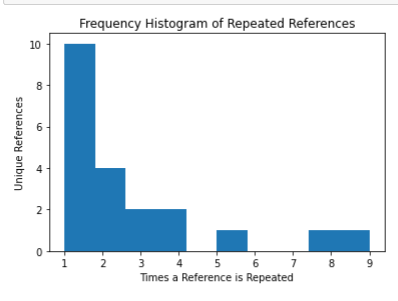
Let's move on to the reference column.

In [15]:
```
col_chr.loc["Reference"]
```

Out[15]:
```
Data Type         object
NA's                  28
Unique Values         21
Name: Reference, dtype: object
```

The reference column includes a collection of unique strings, but some will be repeated. As such, we can check do a similar analysis as with the person column, where we check how many times various references are repeated.

In [16]:
```
ref_ftable = df_final.ref.value_counts().to_frame()
plt.hist(ref_ftable.ref, density=False)
plt.xlabel('Times a Reference is Repeated')
plt.ylabel('Unique References')
plt.title('Frequency Histogram of Repeated References')
plt.show()
```



So, we can see a right-skewed histogram which is more spread out than the person column. There seem to be 3 references repeated 5 times or more. Let's take a look at each:

In [17]:
```
ref_ftable[ref_ftable["ref"]>=5]
```

Out[17]:

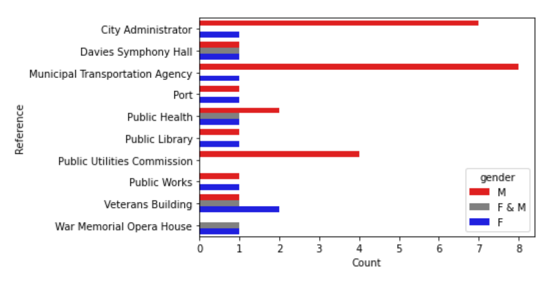
	ref
Municipal Transportation Agency	9
City Administrator	8
Veterans Building	5

Now let's conduct a gender breakdown of the reference column. There are a lot of them, so lets include only references associated with more than 1 statue.

In [18]:
```
ref_gender2=df_final.groupby(["ref","gender"]).size() \
                    .to_frame().reset_index() \
                    .rename(columns={0:"Count","ref":"Reference"})
ref_freq=ref_gender2.groupby("Reference").sum()
ref_considered="|".join(ref_freq[ref_freq["Count"]>=2].index.tolist())
ref_gender_final=ref_gender2[
    ref_gender2["Reference"].str.contains(ref_considered)]
sns.barplot(y="Reference",x="Count",hue="gender",
            data=ref_gender_final,
            hue_order=["M","F & M","F"],
            palette=["Red","grey","blue"],orient="h")
```

```
c:\users\ib\appdata\local\programs\python\python38\lib\site-packages\pandas\core\strings.py:1954: Use
rWarning: This pattern has match groups. To actually get the groups, use str.extract.
  return func(self, *args, **kwargs)
```

Out[18]:
```
<AxesSubplot:xlabel='Count', ylabel='Reference'>
```



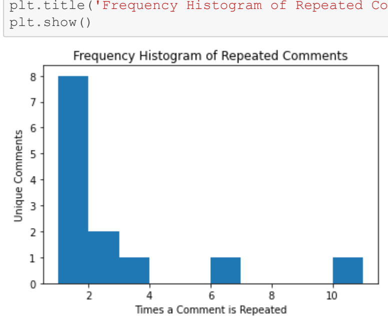
Now we can move on to the comments column:

In [19]:
```
col_chr.loc["Comments"]
```

Out[19]:
```
Data Type         object
NA's                  50
Unique Values         13
Name: Comments, dtype: object
```

Again, we will plot a distribution of how often various comments are repeated.

In [20]:
```
com_ftable = df_final.comments.value_counts().to_frame()
plt.hist(com_ftable.comments, density=False)
plt.xlabel('Times a Comment is Repeated')
plt.ylabel('Unique Comments')
plt.title('Frequency Histogram of Repeated Comments')
plt.show()
```



Here there are 3 comments repeated 3 or more times. Let's look at what they were.

In [21]:
```
com_ftable[com_ftable["comments"]>=3]
```

Out[21]:

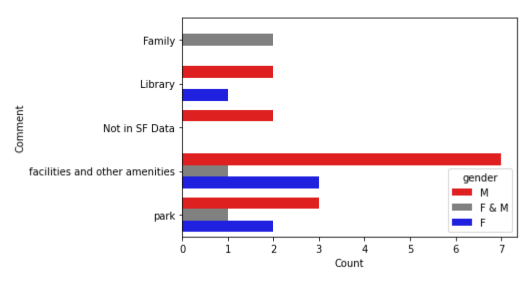
	comments
facilities and other amenities	11
park	6
Library	3

It seems that this information might denote the environment of a monument.

Let's do a gender breakdown of the comments column, again only considering comments that occur more than once.

In [22]:
```
com_gender2=df_final.groupby(["comments","gender"]).size() \
                    .to_frame().reset_index() \
                    .rename(columns={0:"Count","comments":"Comment"})
com_freq=com_gender2.groupby("Comment").sum()
com_considered="|".join(com_freq[com_freq["Count"]>=2].index.tolist())
com_gender_final=com_gender2[
    com_gender2["Comment"].str.contains(com_considered)]
sns.barplot(y="Comment",x="Count",hue="gender",
            data=com_gender_final,
            hue_order=["M","F & M","F"],
            palette=["Red","grey","blue"],orient="h")
```

Out[22]:
```
<AxesSubplot:xlabel='Count', ylabel='Comment'>
```



That concludes a case by case examination of each individual column. There is a lot of possibilities in terms of merging with other datasets, such as perhaps locational data.

Thanks to Ashwani Rathee for submitting the data to kaggle.

Original Data Source: “Representation of Women in City Property - City Administrator's List.” Data.gov, Publisher Data.sfgov.org, 7 Oct. 2020, catalog.data.gov/dataset/representation-of-women-in-city-property-city-administrators-list.