Visualization of Complex Data

DATS 6401 Reza Jafari,PHD

Final Project(San Francisco Payrolls)

Mahtab Barkhordarian

02/05/2022

Table of Contents

Abstract:

The purpose of this project is to observe the peoples 'income who live in San Francisco with different job titles . The reason for choosing this dataset is, the most high tech companies are on those area and I wanted to know are the base pay and others pay is related to job title or even other variables or not.

Introduction:

In this project, I covered the whole things from beginning of the semester. First , I started with cleaning the data. I removed outliers, then PCA to see how many features are needed. Then I've done the normality test to see if the data is normal or not, and then data transformation to normal. I have done heatmap and Pearson correlation. Some statistics and finally at the end visualize the data using Seaborn package and dashboard.

Description of the Dataset:

This dataset contains the California public employee salaries from years 2014 to 2019. It has 10 variables which 3 of them are categorical and 7 are numerical. The length of the dataset is 205906 before preprocessing and after that is 54647.

```
>>> df.head(5)
        Employee Name              Job Title  ...  Year Status
7    Douglas R Murray             Firefighter  ...  2019     FT
8     Clint I Pereyra  Patient Care Assistant  ...  2019     FT
11          Brent Lee           Special Nurse  ...  2019     PT
12    Jabari L Albert        Transit Operator  ...  2019     FT
15  Porfirio O Magana        Transit Operator  ...  2019     FT

[5 rows x 10 columns]
>>> df.columns
Index(['Employee Name', 'Job Title', 'Base Pay', 'Overtime Pay', 'Other Pay',
       'Benefits', 'Total Pay', 'Total Pay & Benefits', 'Year', 'Status'],
      dtype='object')
```

Here is the head of dataset and the column names to get more familiar with.

All the students are trying to find the job after their graduation and this dataset can help to see which kind of jobs with which status has the most impact on the base or over time salary. Which title can help them with more benefits. Or they can understand working full time is going to work better in their situation or part time.

Pre Processing Dataset:

I did some preprocessing such as remove the nans. Removed some 'Not-provided' observations from the dataset. Some numeric data which the types were object were converted to integer.

The number of the job titles were too much for plotting then, I kept the most repeated ones which are :

```
>>> df['Job Title'].unique()
array(['Firefighter', 'Patient Care Assistant', 'Special Nurse',
       'Transit Operator', 'Registered Nurse', 'Police Officer 3',
       'Custodian', 'Public Service Trainee', 'Recreation Leader',
       'Public Svc Aide-Public Works'], dtype=object)
```

The first few rows after cleaning:

```
this is the head of the dataset:
         Employee Name              Job Title  ...  Year  Status
7     Douglas R Murray            Firefighter  ...  2019      FT
8      Clint I Pereyra  Patient Care Assistant  ...  2019      FT
11           Brent Lee          Special Nurse  ...  2019      PT
12     Jabari L Albert       Transit Operator  ...  2019      FT
15   Porfirio O Magana       Transit Operator  ...  2019      FT

[5 rows x 10 columns]
```

And this picture shows how the data is clean:

```
Employee Name         0
Job Title             0
Base Pay              0
Overtime Pay          0
Other Pay             0
Benefits              0
Total Pay             0
Total Pay & Benefits  0
Year                  0
Status                0
dtype: int64
2.772357128479148
Employee Name         0
Job Title             0
Base Pay              0
Overtime Pay          0
Other Pay             0
Benefits              0
Total Pay             0
Total Pay & Benefits  0
Year                  0
Status                0
```

Outlier detection & removal:

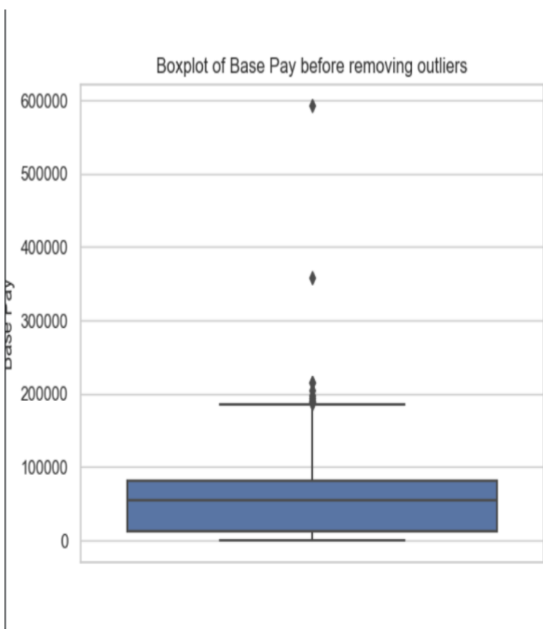
I used IQR method here. There are some information from this website:

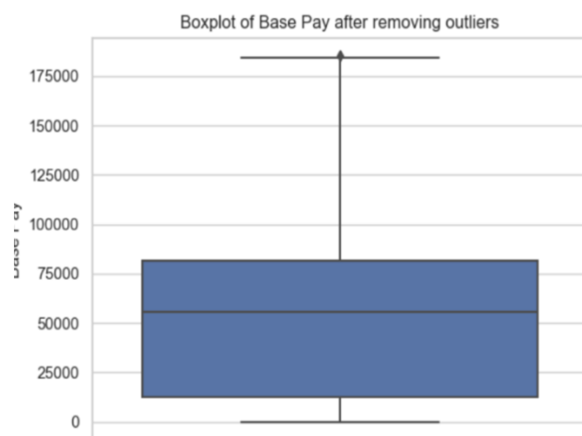
Another robust method for labeling outliers is the IQR (interquartile range) method of outlier detection developed by John Tukey, the pioneer of exploratory data analysis. This was in the days of calculation and plotting by hand, so the datasets involved were typically small, and the emphasis was on understanding the story the data told. If you’ve seen a box-and-whisker plot (also a Tukey contribution), you’ve seen this method in action.

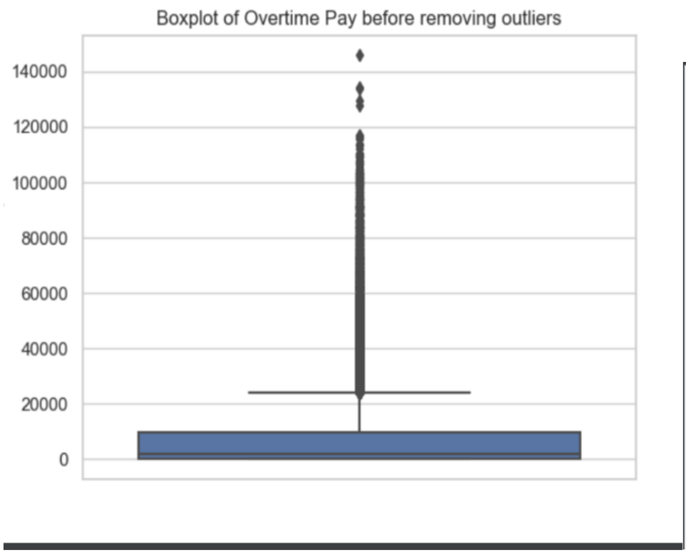
A box-and-whisker plot uses quartiles (points that divide the data into four groups of equal size) to plot the shape of the data. The box represents the 1st and 3rd quartiles, which are equal to the 25th and 75th percentiles. The line inside the box represents the 2nd quartile, which is the median.

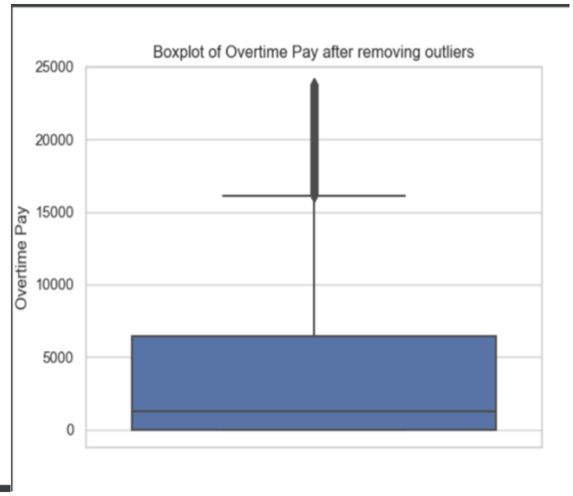
The interquartile range, which gives this method of outlier detection its name, is the range between the first and the third quartiles (the edges of the box). Tukey considered any data point that fell outside of either 1.5 times the IQR below the first – or 1.5 times the IQR above the third – quartile to be “outside” or “far out”. In a classic box-and-whisker plot, the ‘whiskers’ extend up to the last data point that is not “outside”.

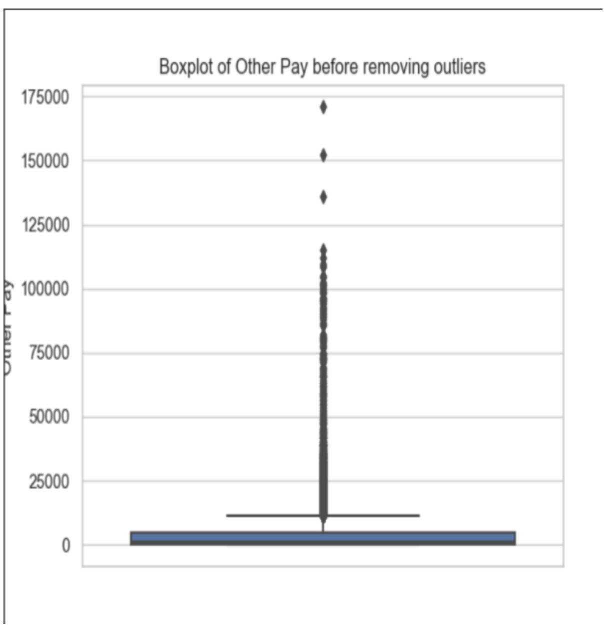
I plotted the variables before removing the outliers and after removing the outliers.

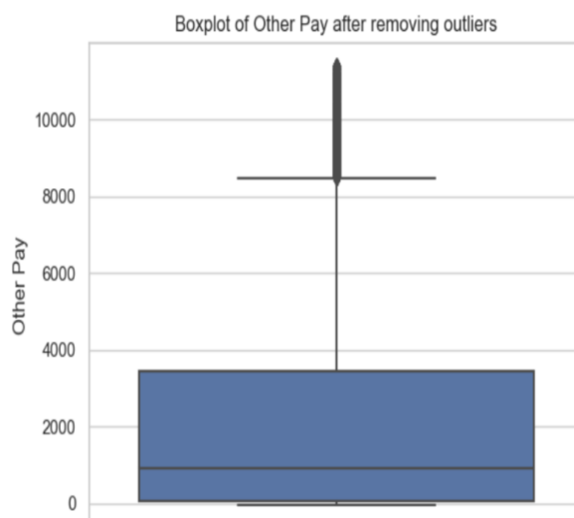


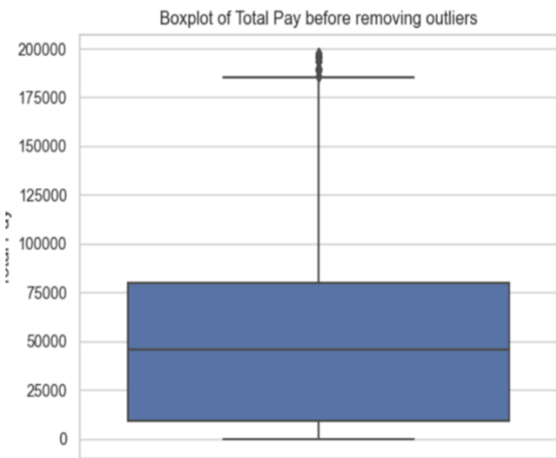



Boxplot of Overtime Pay before removing outliers
140000
120000
100000
80000
60000
40000
20000
0

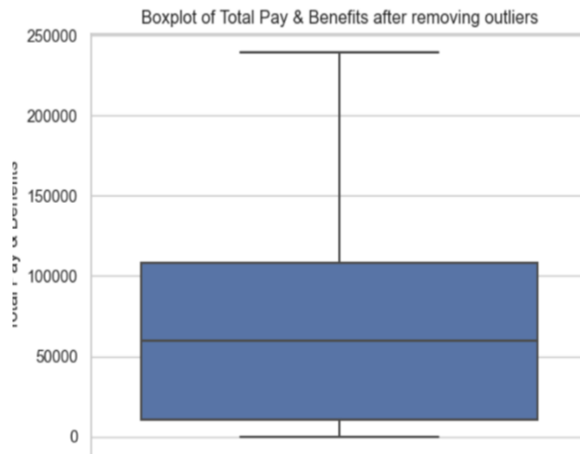

Boxplot of Overtime Pay after removing outliers
Overtime Pay
25000
20000
15000
10000
5000
0


Boxplot of Other Pay before removing outliers
175000
150000
125000
100000
75000
50000
25000
0

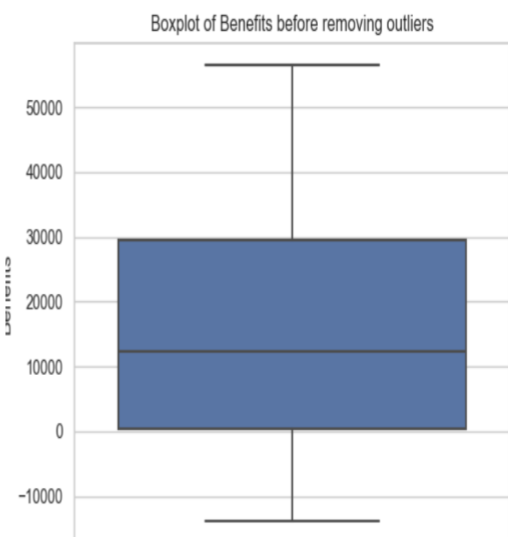

Boxplot of Other Pay after removing outliers
Other Pay
10000
8000
6000
4000
2000
0



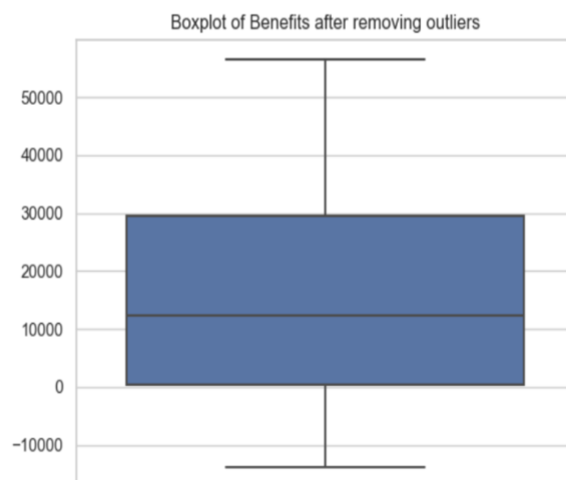
Boxplot of Total Pay before removing outliers
200000
175000
150000
125000
100000
75000
50000
25000
0



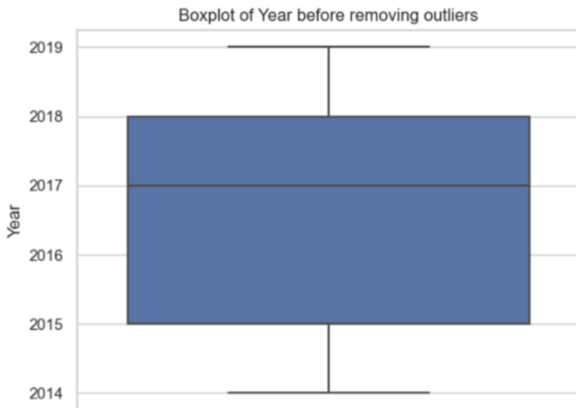
Boxplot of Total Pay & Benefits after removing outliers
250000
200000
150000
100000
50000
0

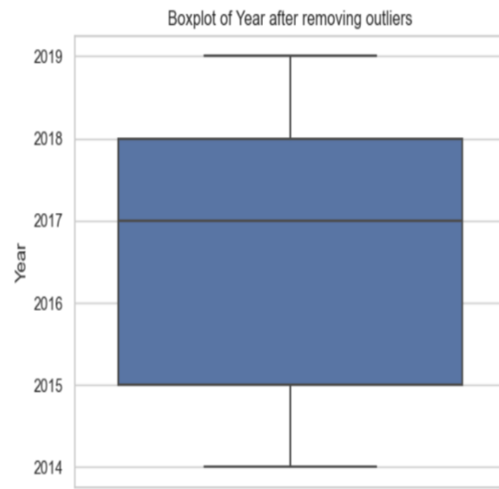


Boxplot of Benefits before removing outliers
50000
40000
30000
20000
10000
0
−10000



Boxplot of Benefits after removing outliers
50000
40000
30000
20000
10000
0
−10000



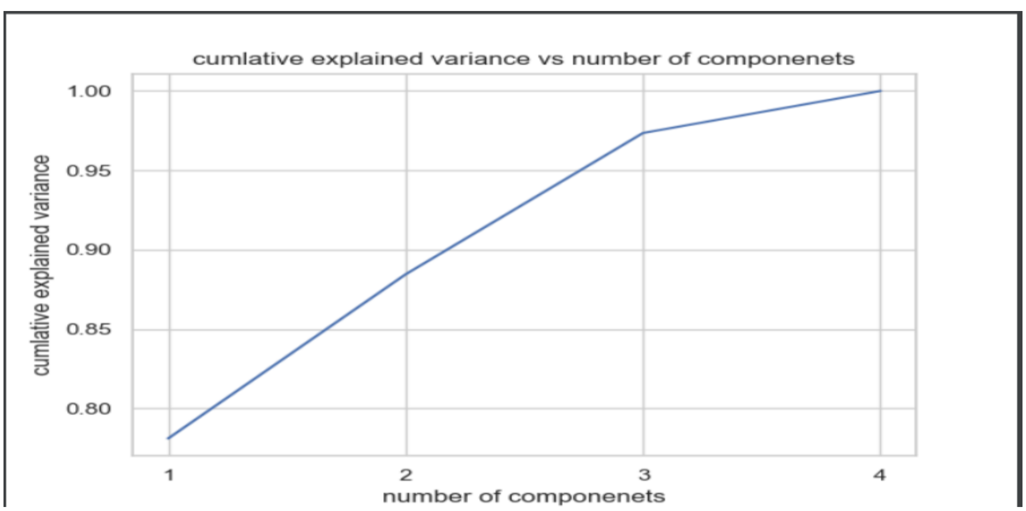


Principal Component Analysis (PCA):

The PCA shows, how many features we need to keep and how many we can remove. from this PCA we can realize that we can have the explained variance ratio more that 95 percent with only just 4 features.

```
original data: singular values [7.48220024e+14 3.68264134e+12 1.29054091e+12 2.30177640e+11
 1.23509317e-01 8.25066623e-02]
original data: condition number 3.3689207776897948e+16
Original Dim (44377, 6)
Transform Data (44377, 4)
explained variance ratio[0.78099956 0.10343326 0.08906817 0.02649901]

>>>
```



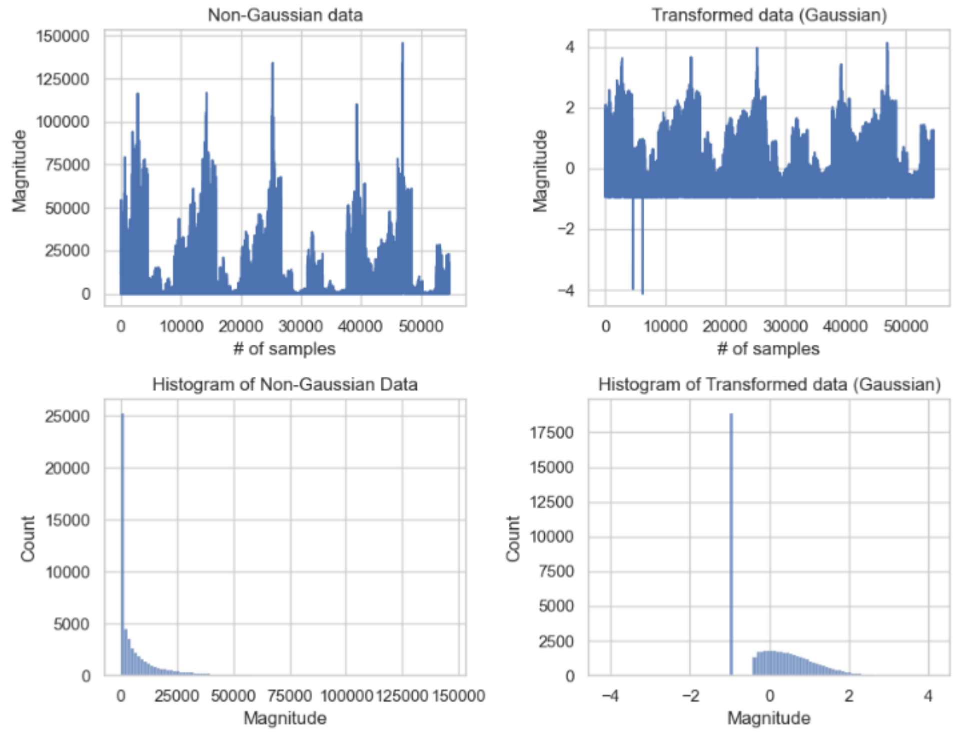
Normality test :

I did the KDTEST to see if the data distributed normal or not. For that I plot them before getting normal and also used the Gaussian method to transform the data to normal if it is not by itself.

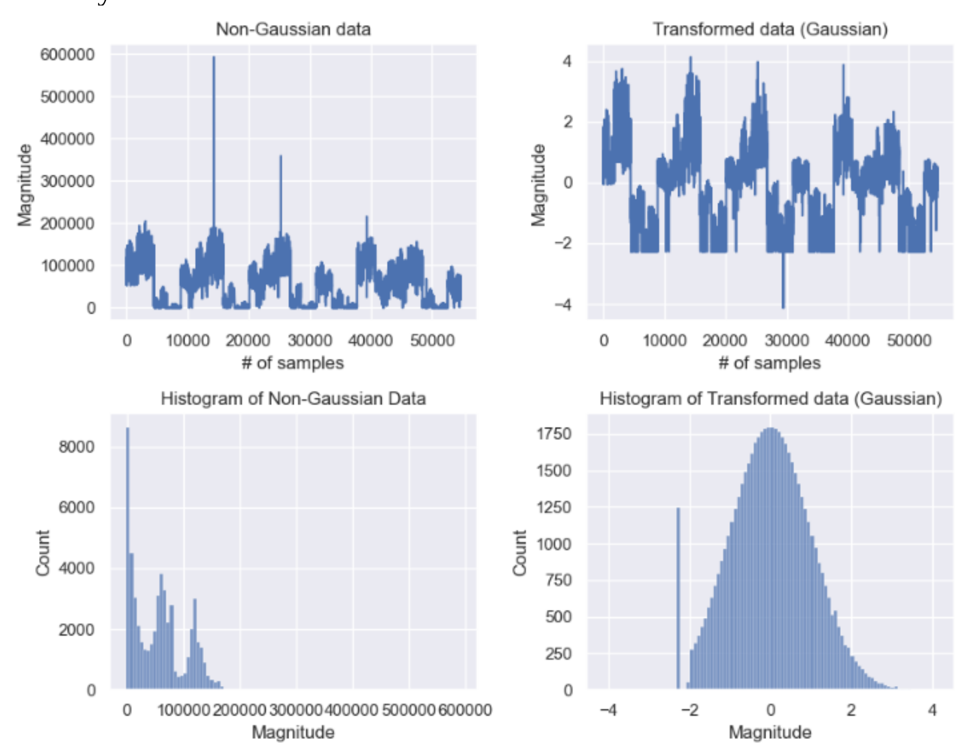
```
K-S test: statistics=0.65246, p-value=0.00000
K-S test: Overtime-pay column looks Non-Normal
K-S test: statistics=0.97713, p-value=0.00000
K-S test: Base-pay column looks Non-Normal
K-S test: statistics=0.86278, p-value=0.00000
K-S test: other-pay column looks Non-Normal
K-S test: statistics=0.97132, p-value=0.00000
K-S test: Benefit column looks Non-Normal
K-S test: statistics=0.99792, p-value=0.00000
K-S test: Total Pay column looks Non-Normal
K-S test: statistics=0.99862, p-value=0.00000
K-S test: Total Pay & Benefits column looks Non-Normal
K-S test: statistics=1.00000, p-value=0.00000
K-S test: Year column looks Non-Normal
```

KS TEST shows none of the column were normal and in the few next screenshots, you can see that I converted them to normal.

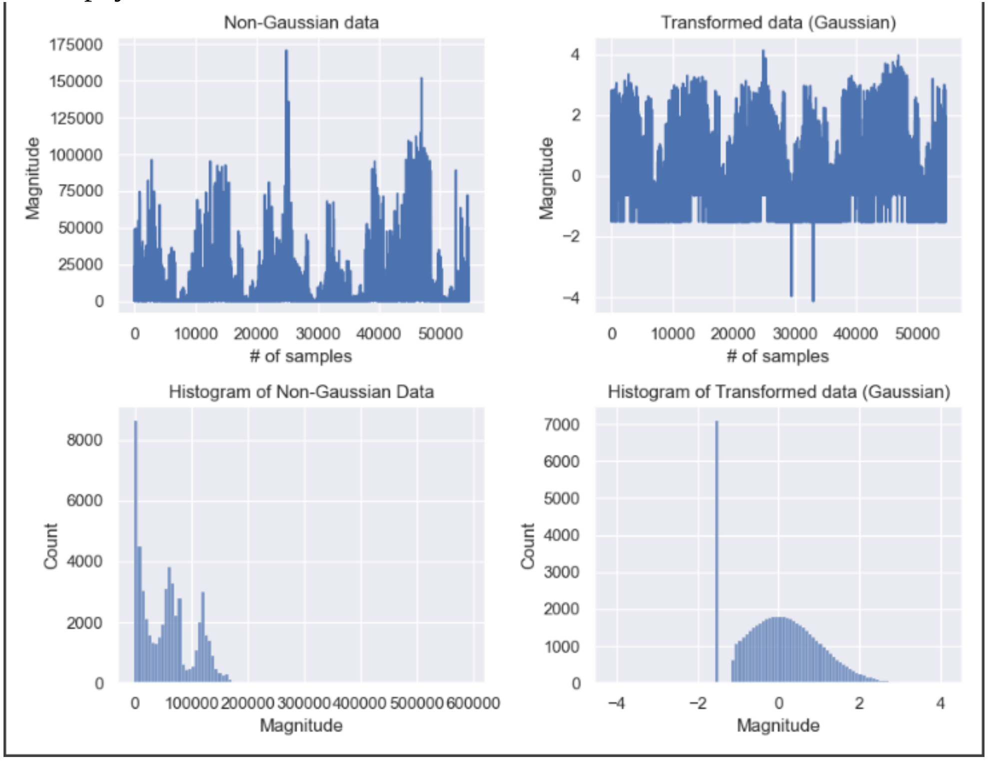
Overtime Pay:



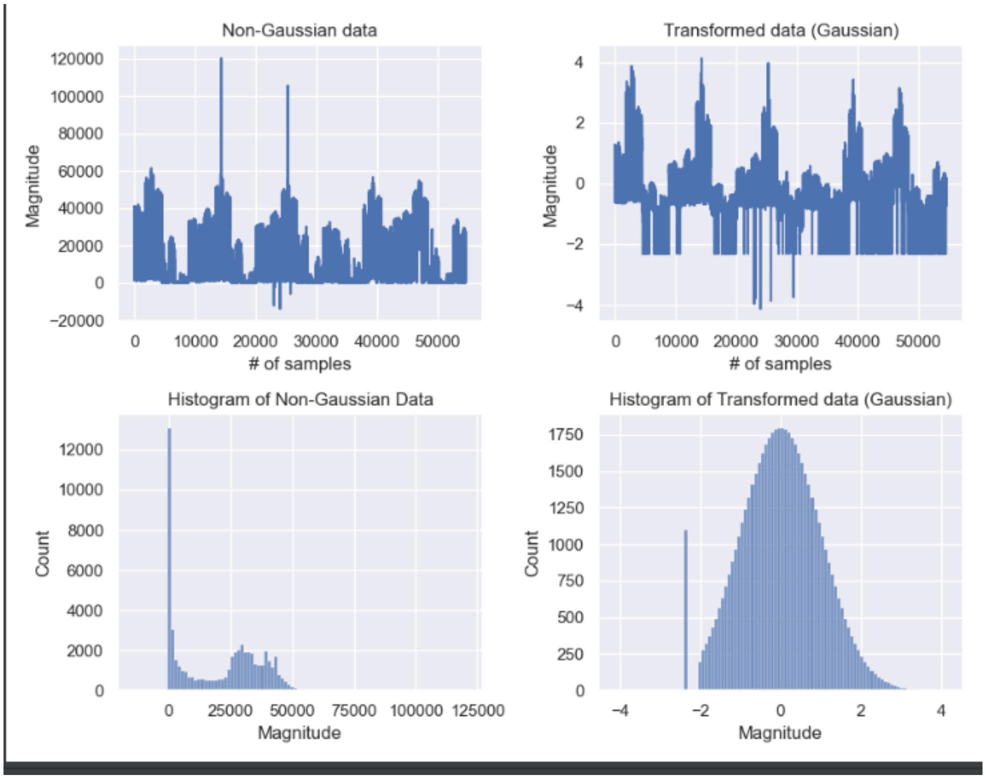
Base Pay



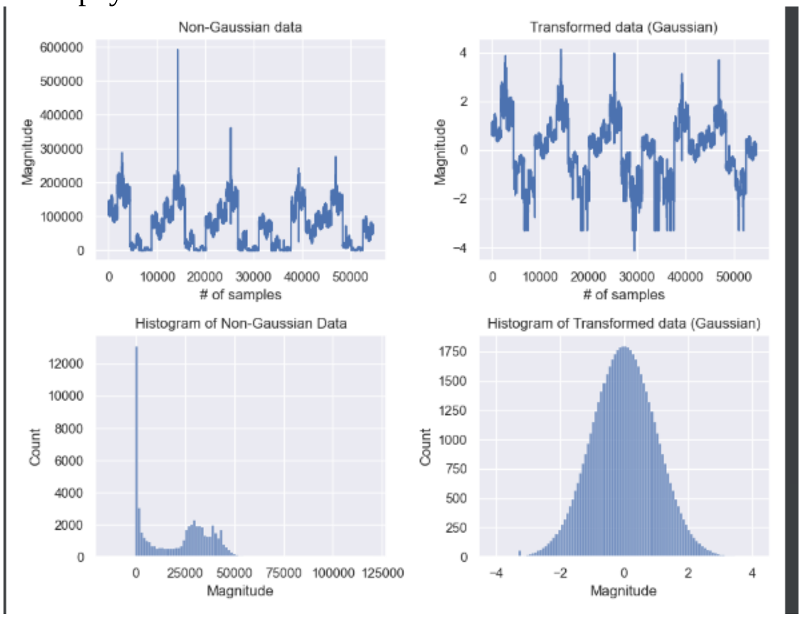
Other pay:



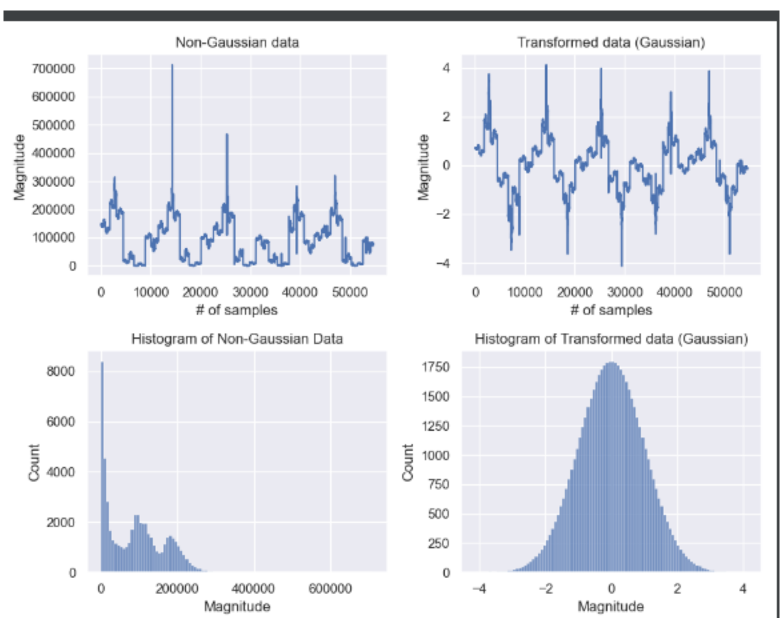
Benefits:



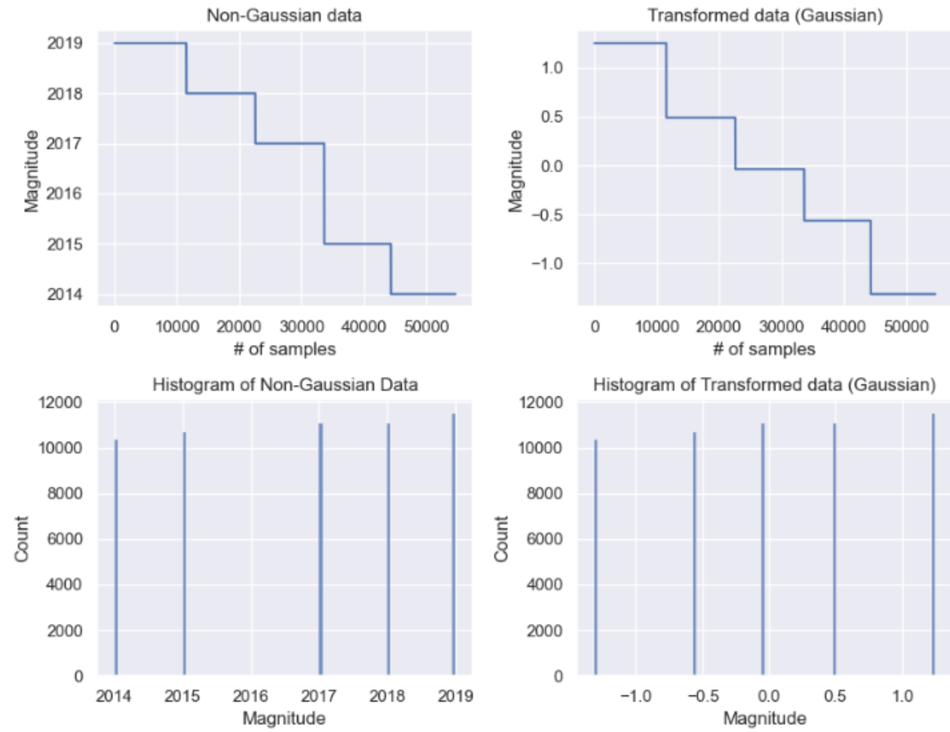
Total pay:



Total pay and benefits:



Year:



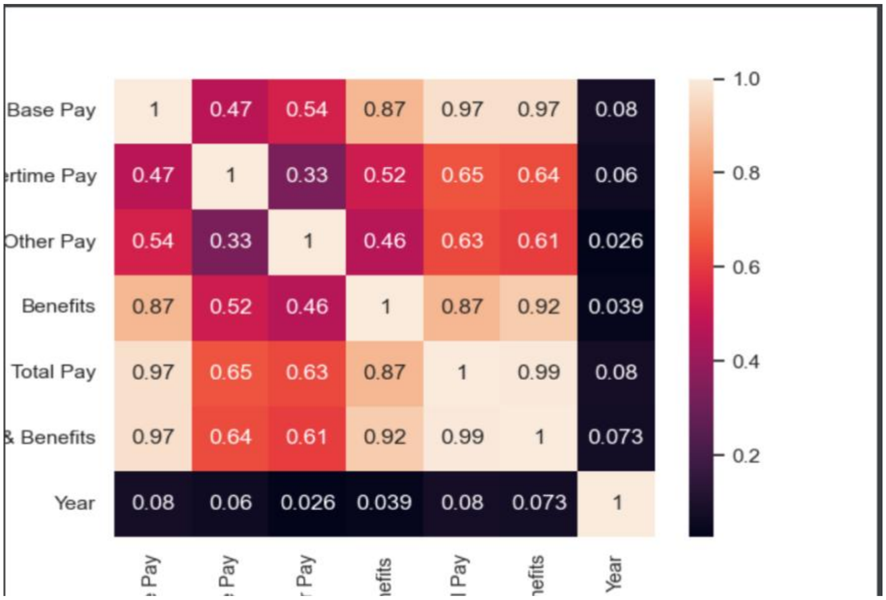
Statistics:

I have done some statistics to see what is the average and median of the base pay, total pay, benefits and other features on the San Francisco area.

```
the median of the base pay is: 55508.52
the mean of the base pay is: 56672.736849232344
the median of the Overtime Pay is: 2036.99
the mean of the Overtime Pay is: 7843.5960804801725
the median of the Other Pay is: 1622.78
the mean of the Other Pay is: 4486.5198140794555
the median of the Benefits is: 23647.91
the mean of the Benefits is: 19726.023964170035
the median of the Total Pay is: 62840.33
the mean of the Total Pay is: 69002.85274379197
the median of the Total Pay & Benefits is: 87107.83
the mean of the Total Pay & Benefits is: 88728.87670796202
```

Heatmap & Pearson correlation coefficient matrix :

we mostly use the heatmap to see what is the correlation between features, for example some time if you see your model still is going to work with 4 features and have the explained expected variance of over 90% then you may can remove one of the features which has the high correlation with another one.



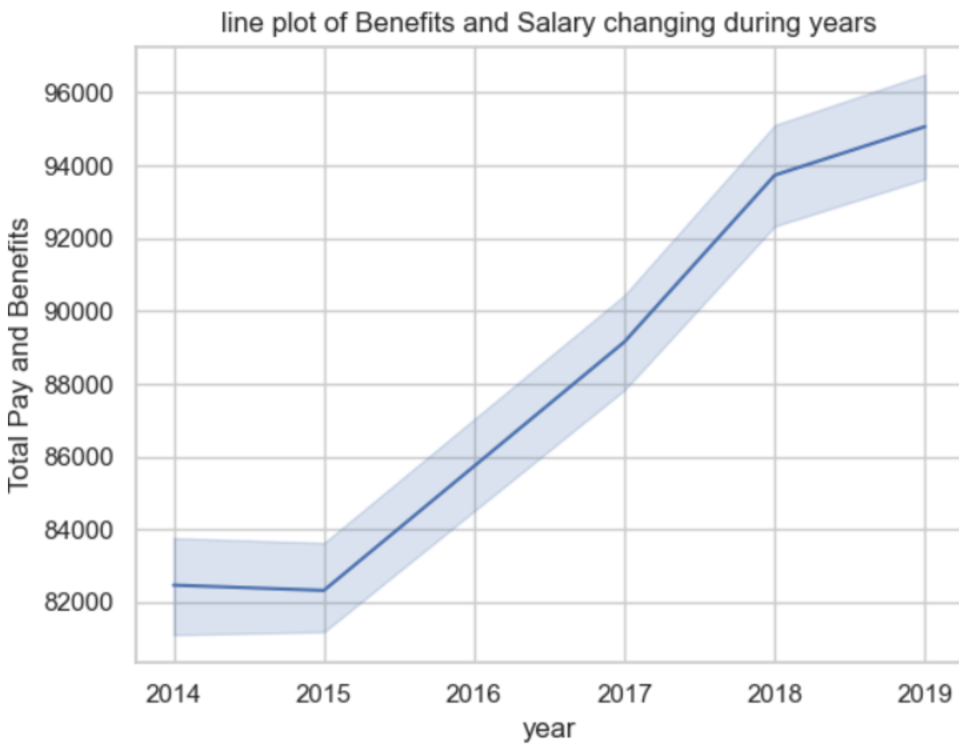
The columns from lefts are :Base pay, overtime pay, other pay, Benefits, Total Pay, Total pay and benefits and year.
As you can see Base pay and total pay and total pay & Benefits have some high correlation.
Total pay and total pay & Benefits have the high correlation which makes sense.
Also Base pay and Benefits and Base pay and total pay have the high correlation together.

Data Visualization(Seaborn):

I used seaborn to visualize my data with different plots here.

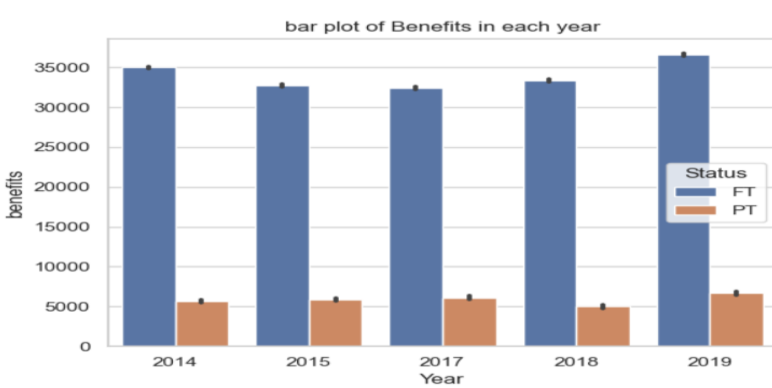
Line plot:

I used line plot here to see if the total pay and benefits is going to increase during the years in San Francisco Area. Which as you see it is highly increasing which means if someone move there for the listed job, can have the big progress.



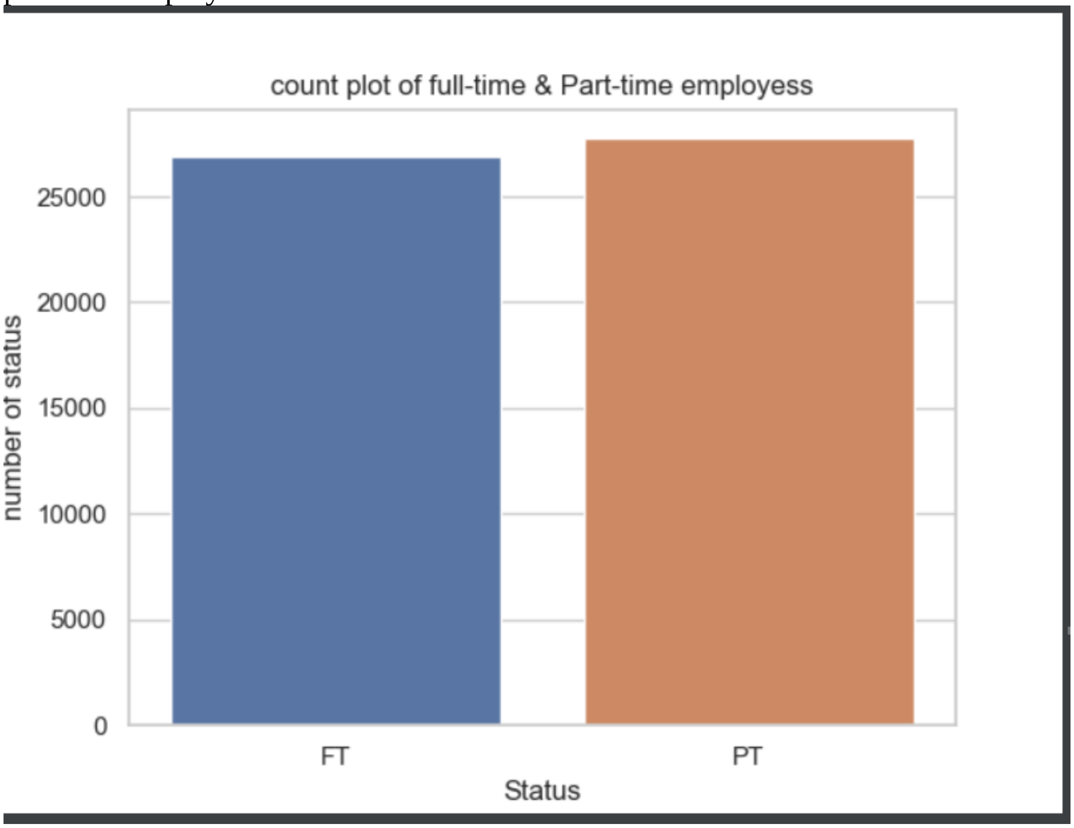
Bar plot:

From this bar plot we can observe that full time employees have much more benefuts compare to Part times.



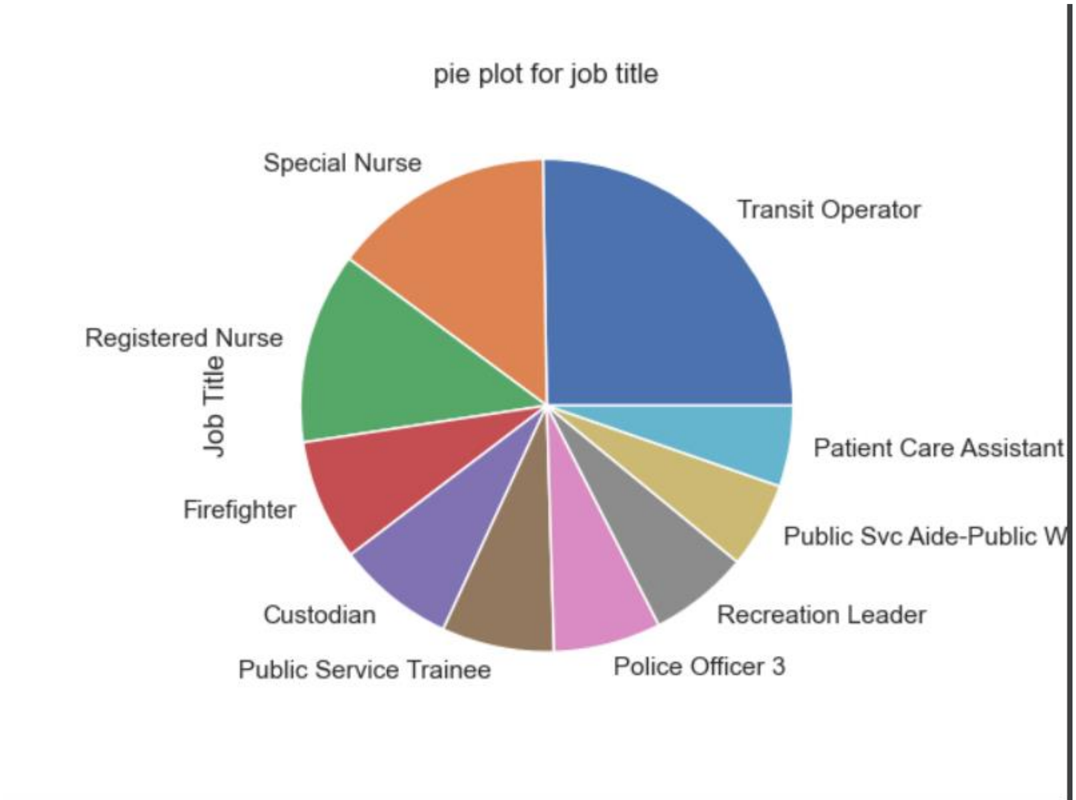
Count plot:

From this count plot , I wanted to realize that the full time employees number are almost equal to part time employess.

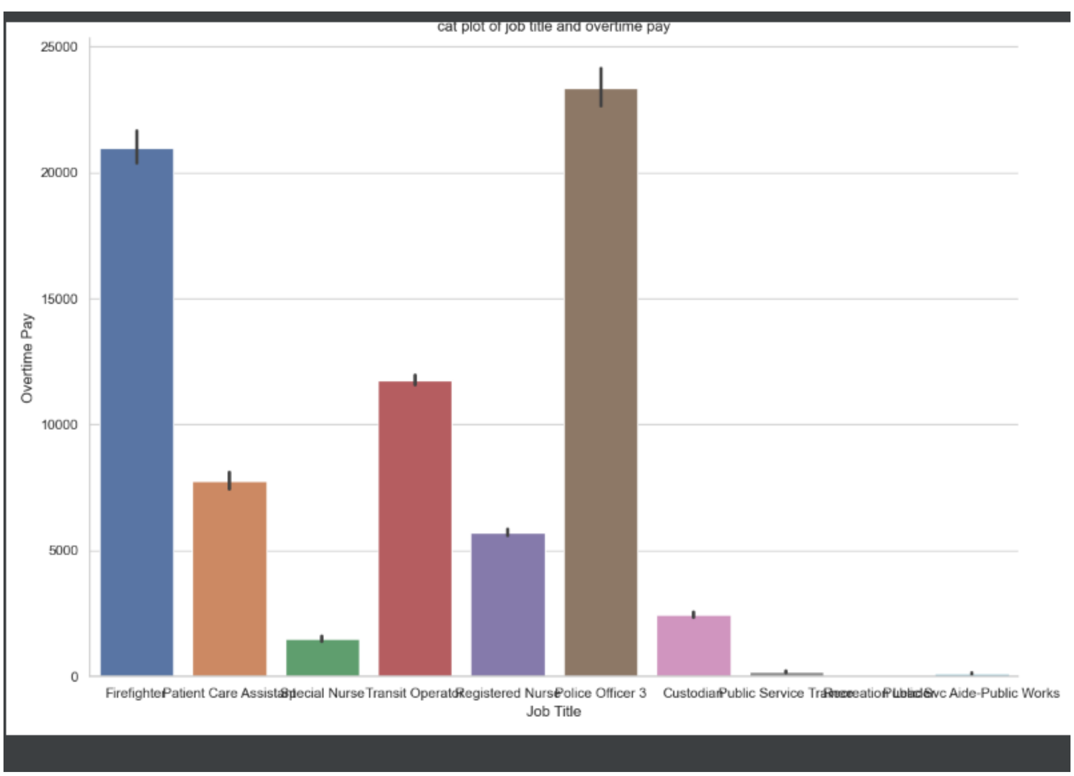


Pie Chart:

These are all the job titles which I used in this dataset. I sliced the dataset with these job titles since they had the most population between others. As you see Transit operator, special and registered nurses have the most population in the dataset.

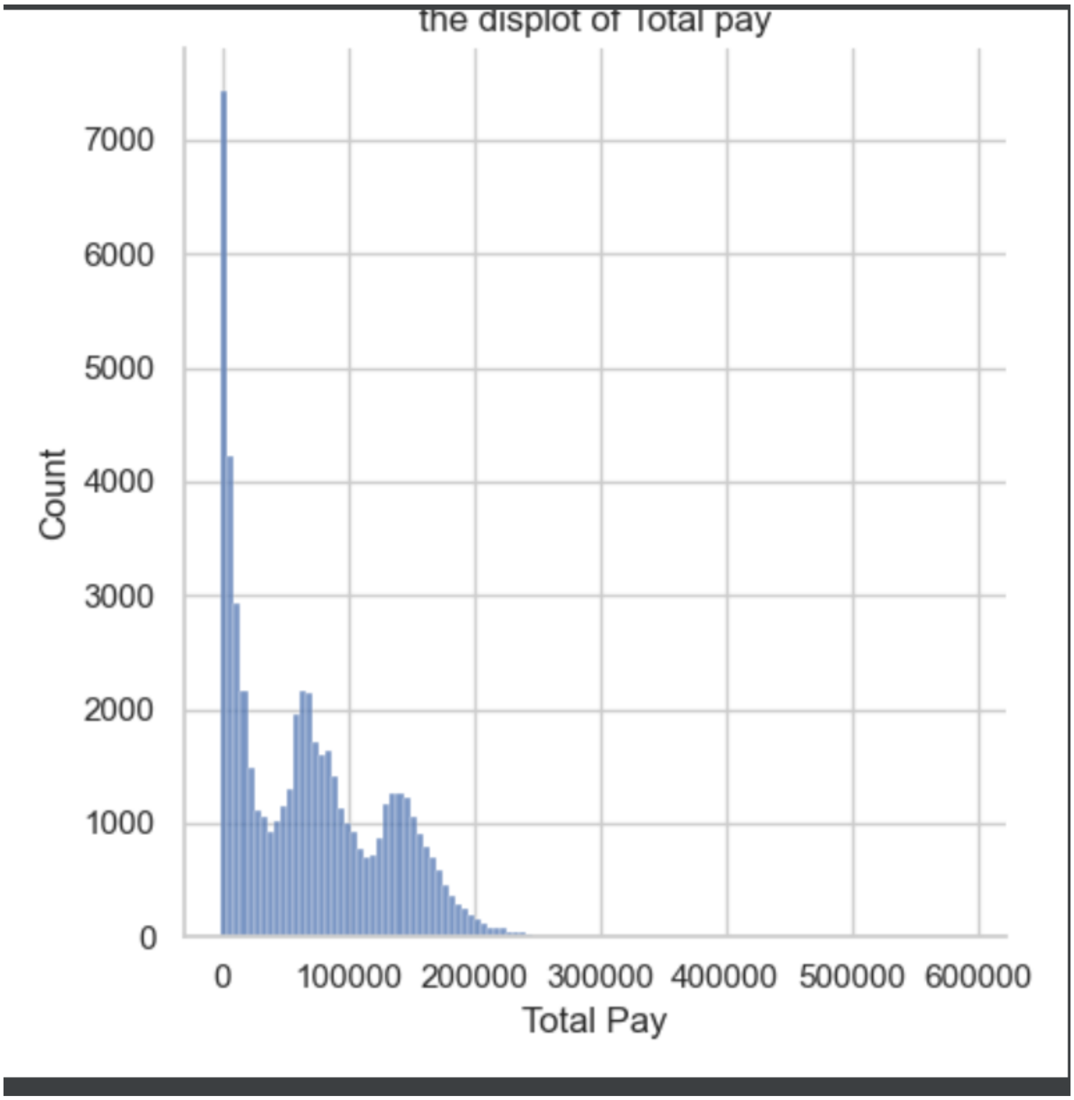

pie plot for job title
Special Nurse
Transit Operator
Registered Nurse
Job Title
Patient Care Assistant
Firefighter
Public Svc Aide-Public W
Custodian
Recreation Leader
Public Service Trainee
Police Officer 3

Cat plot:


cat plot of job title and overtime pay
25000
20000
15000
10000
5000
0
Overtime Pay
Job Title

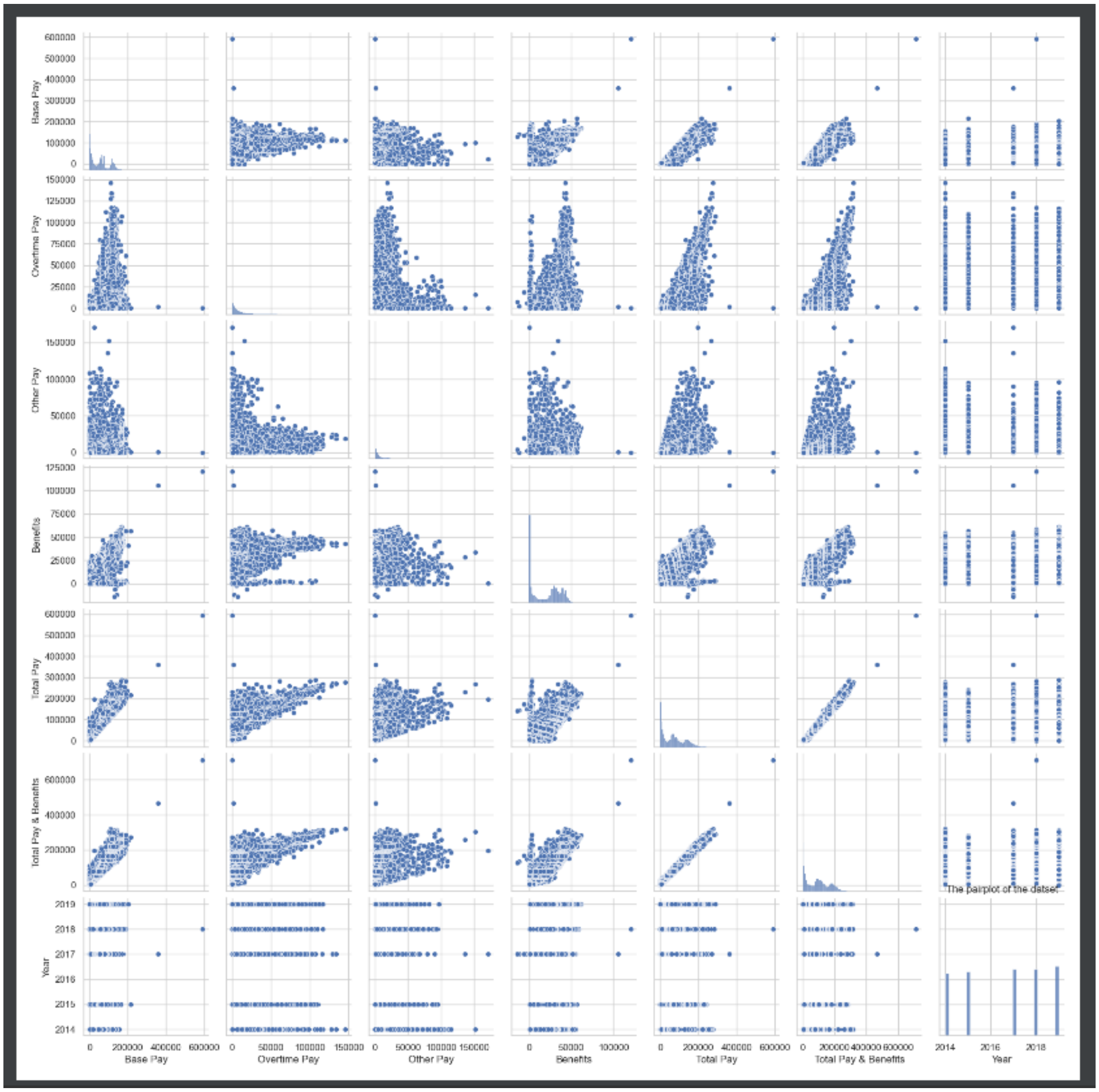
From this Cat plot , I can realize that the jobs with the most overtime pay. Police officers have the most overtime pay and then Fire fighters . The least overtime pay is for patient care assistance and public workers.

Displot:



From this displot we can see the annual income for the San Francisco area is up to $250000 annually. And most of the income is around 100000 to 200000 .

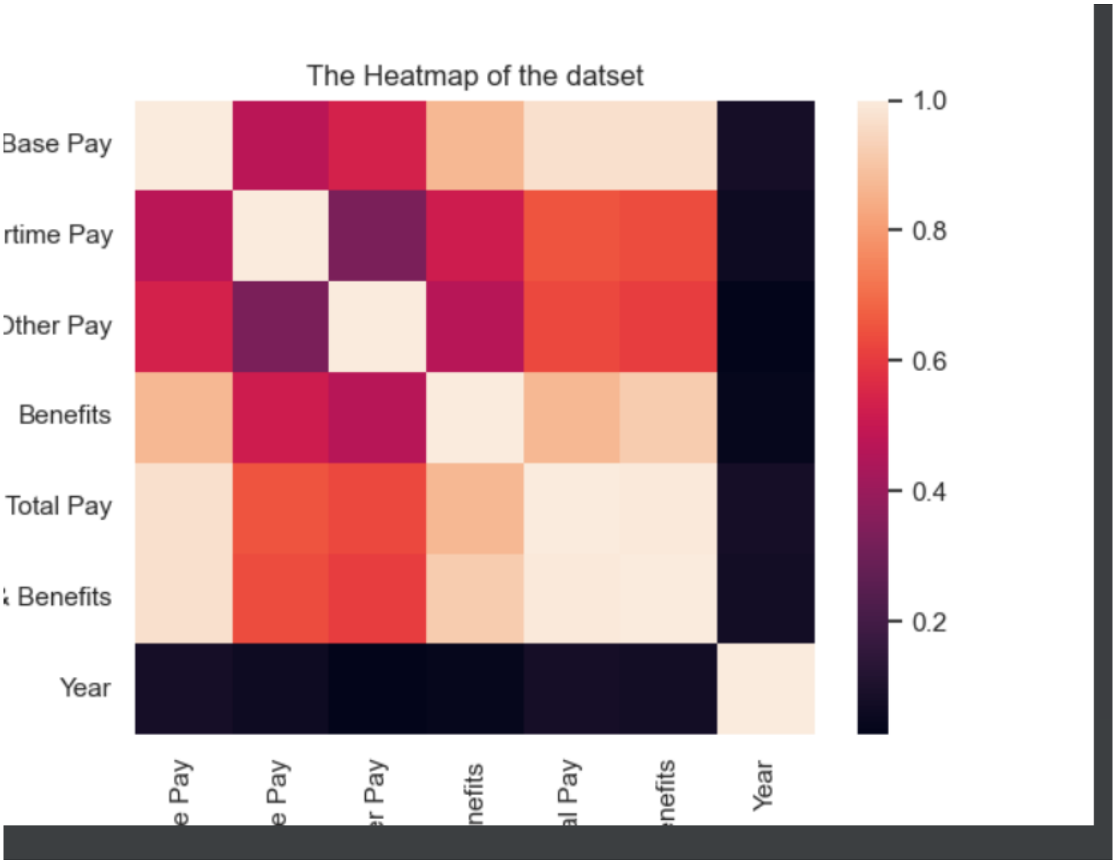
Pair plot:



The pair plot of each dataset shows the relationship of 2 by 2 datasets together.

Heatmap:

Heatmap shows the correlation of variables together.



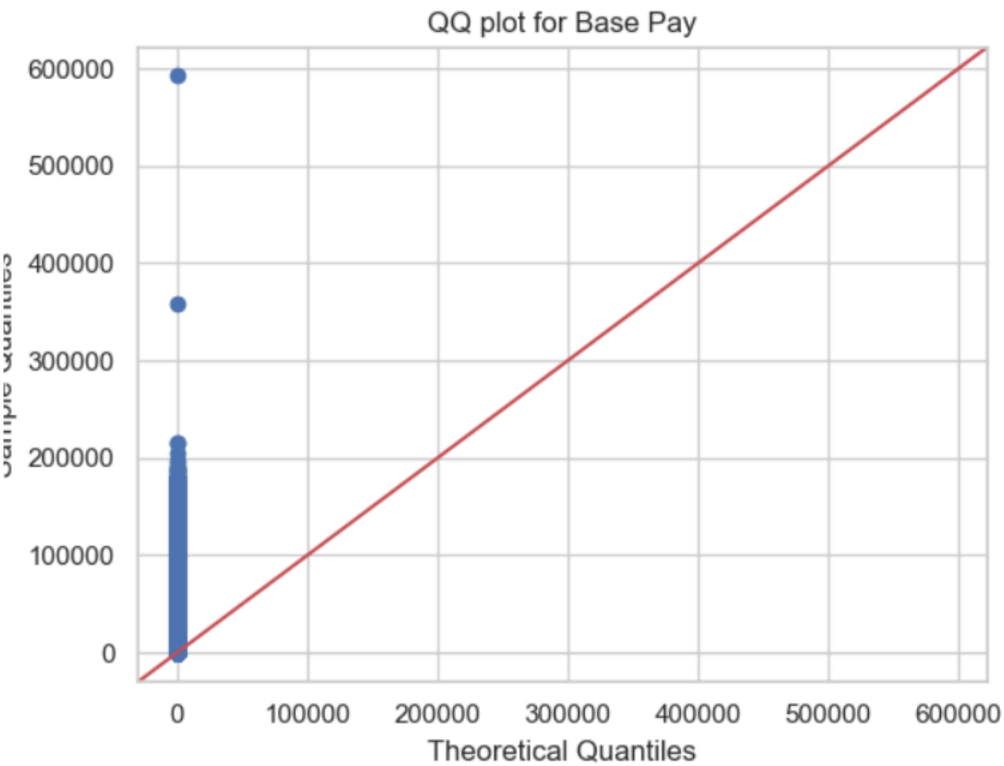
The columns from lefts are :Base pay, overtime pay, other pay, Benefits, Total Pay, Total pay and benefits and year.
As you can see Base pay and total pay and total pay & Benefits have some high correlation.
Total pay and total pay & Benefits have the high correlation which makes sense.
Also Base pay and Benefits and Base pay and total pay have the high correlation together.

Hist Plot:

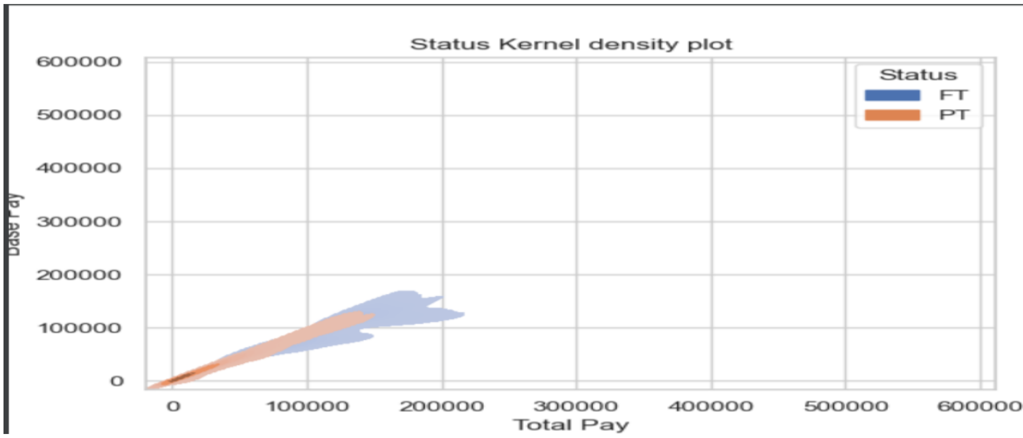
The histplot compares full time employees' total pay with part time employees and it shows that the total pay for the full time employees is almost normal which is not from par time employees.

QQ plot:

The Q-Q plot, or quantile-quantile plot, is a graphical tool to help us assess if a set of data plausibly came from some theoretical distribution such as a Normal or exponential.



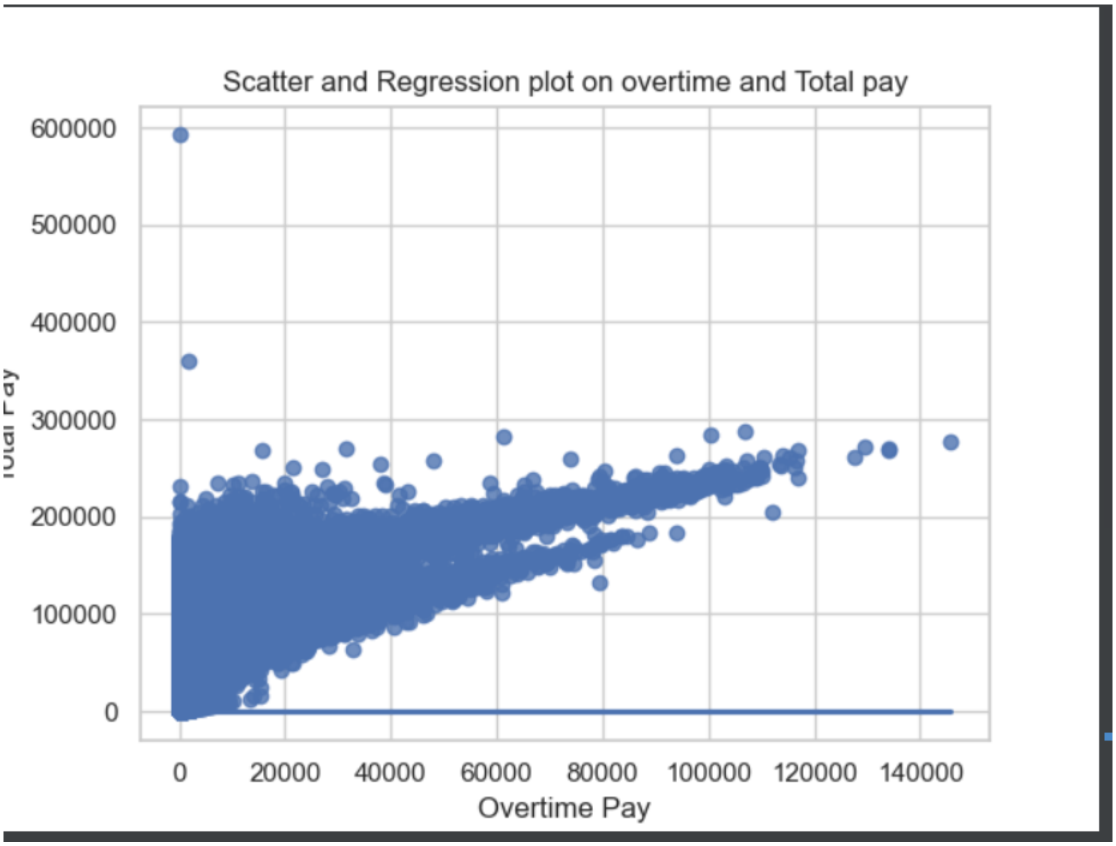
Kernel density estimate:



A kernel density estimate (KDE) plot is a method for visualizing the distribution of observations in a dataset, analogous to a histogram. KDE represents the data using a continuous probability density curve in one or more dimensions. The approach is explained further in the user guide.

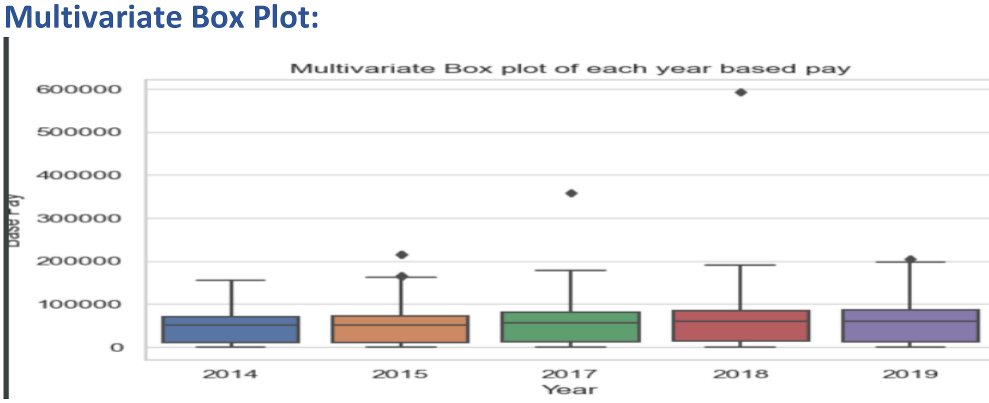
From this plot we can see the distribution of observation is more in full time base on base pay and total compare to part time.

Scatter Plot and regression line:



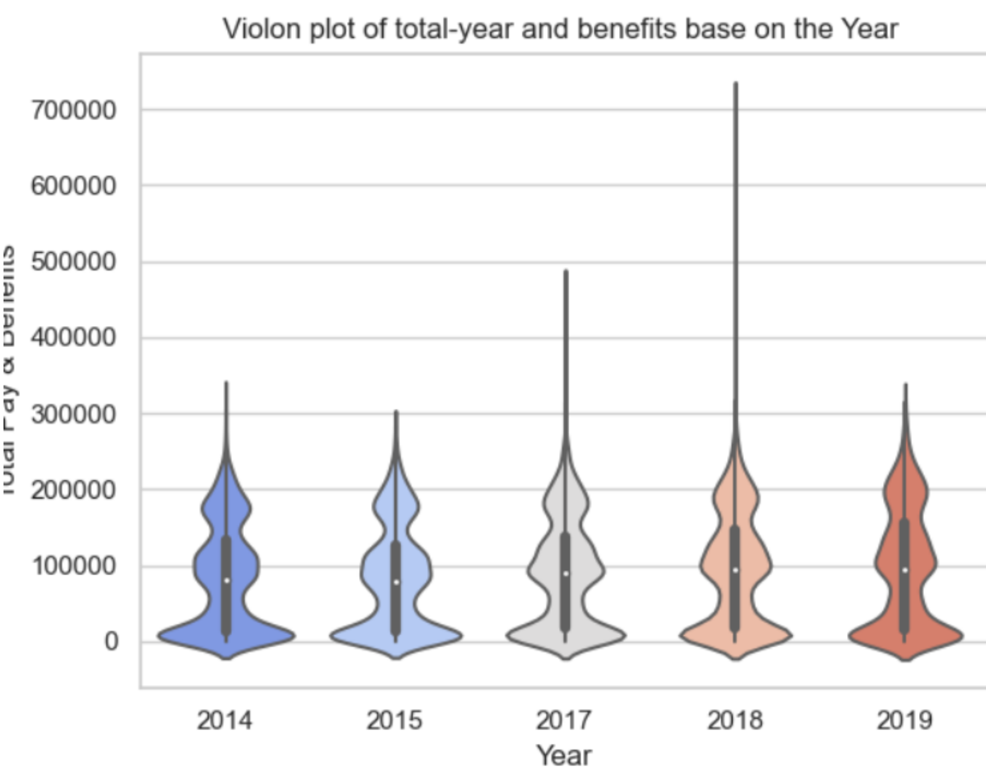
There is no linear relation between total pay and overtime pay.

Multivariate Box Plot:



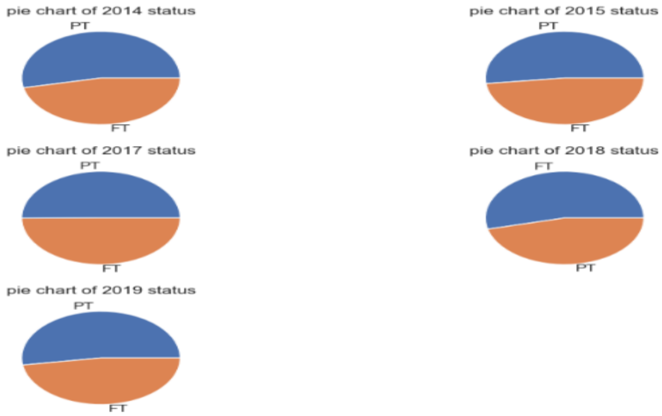
We can observe that the base pay is increasing sligtly from year 2014 till 2019.

Violin plot:



A violin plot is a method of plotting numeric data. It is similar to a box plot, with the addition of a rotated kernel density plot on each side. Violin plots are similar to box plots, except that they also show the probability density of the data at different values, usually smoothed by a kernel density estimator.

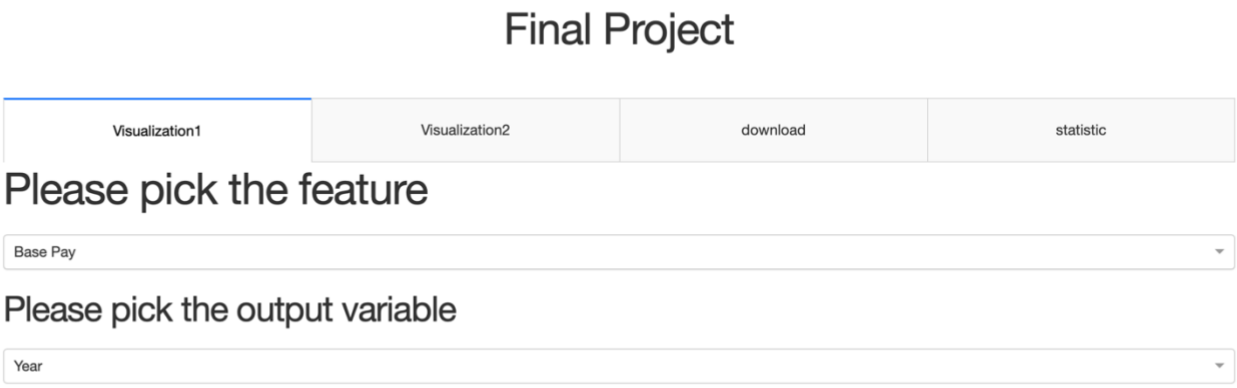
Sub plots:



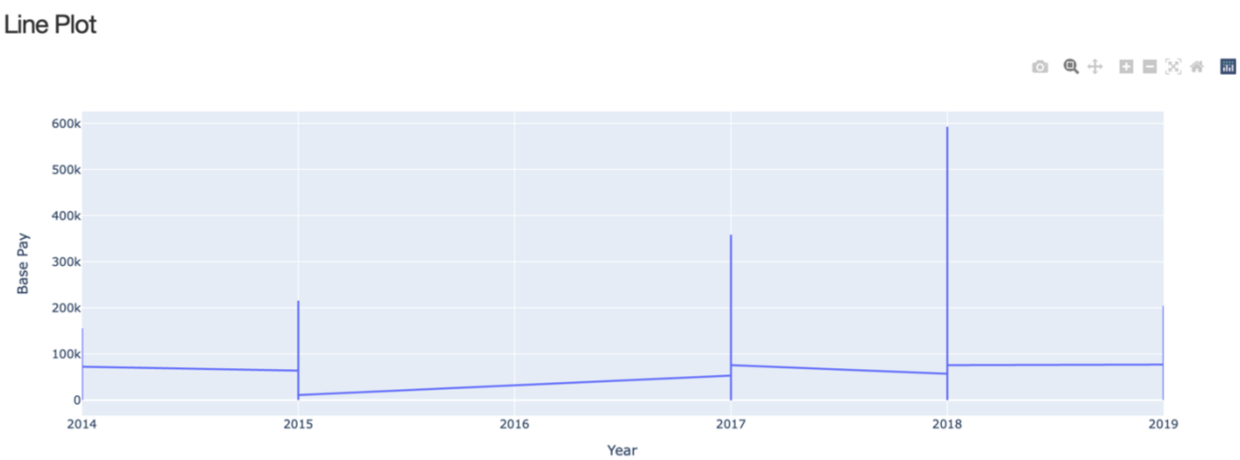
This subplot shows the comparison of the population of full time and part time employees per year. Year 2017 the part time employees were as many as full time employee, but as we can see all other years the population of part time employees are more than full time employees.

Dashboard:

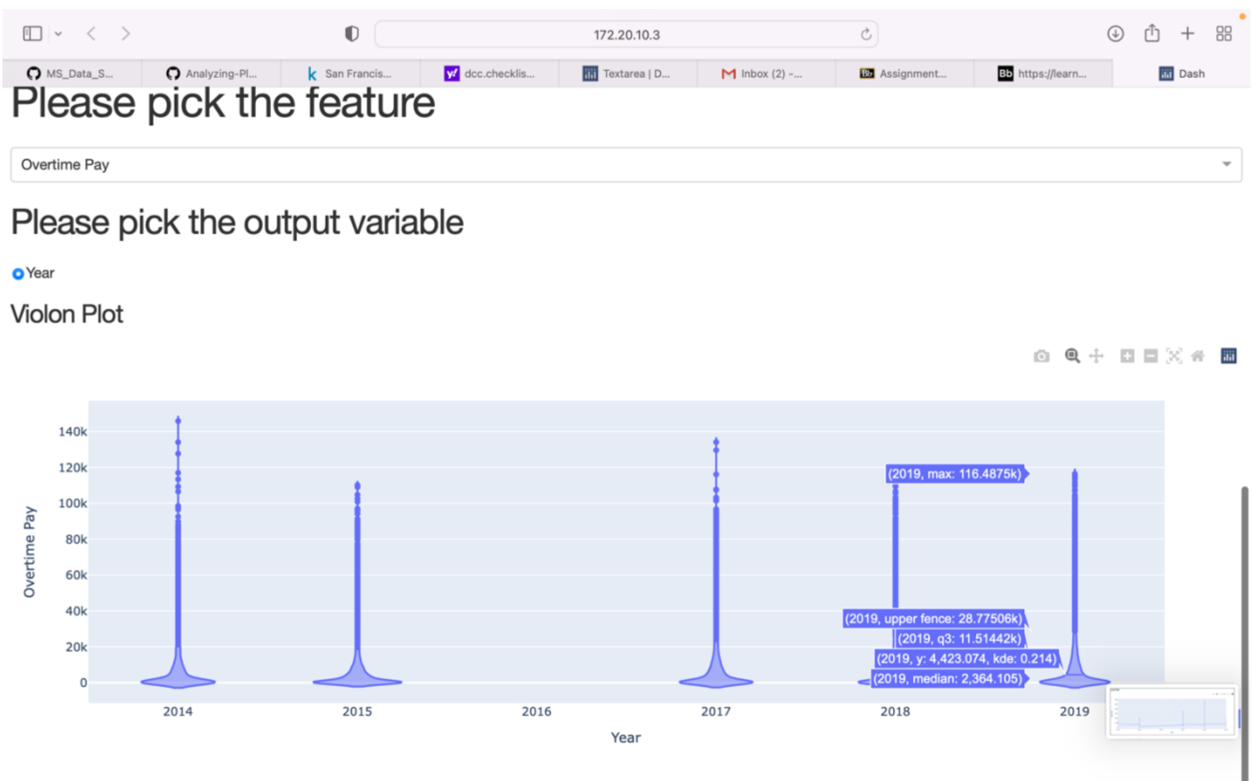
At the end of the project I moved to Dashboard because it's so use full to give the view of data to some none technical people who wants to observe the statistical change in the data.



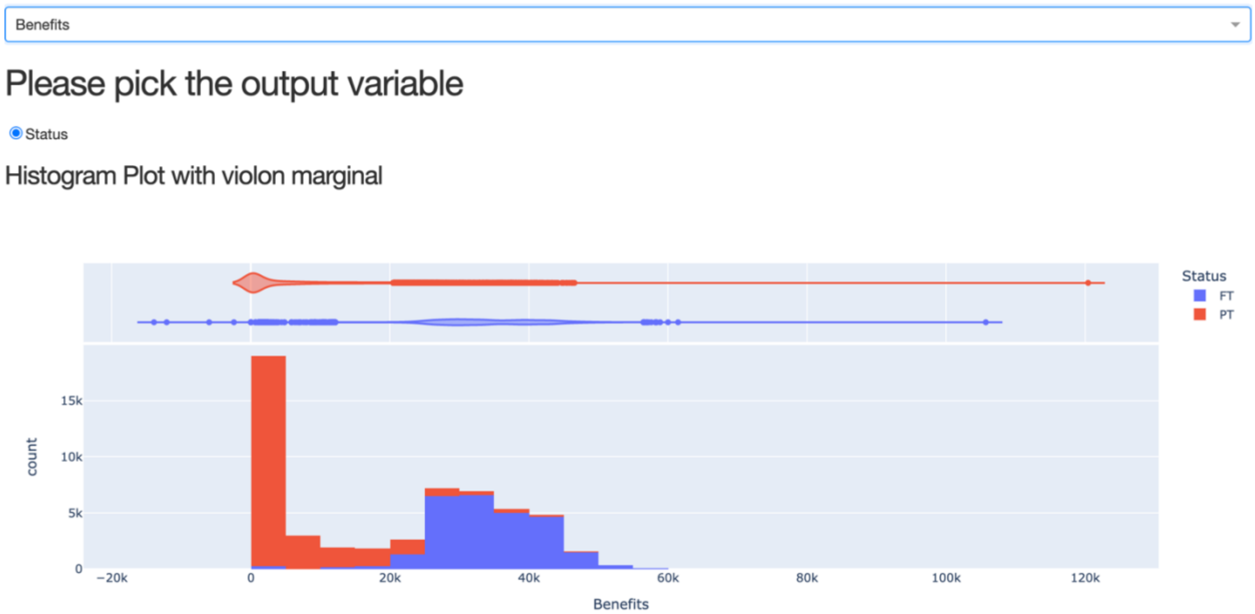
My Dash has 4 tabs, 2 for visualization(I made it in two tabs to not be that busy) and 2 others , one is for download and one for statistical calculation.



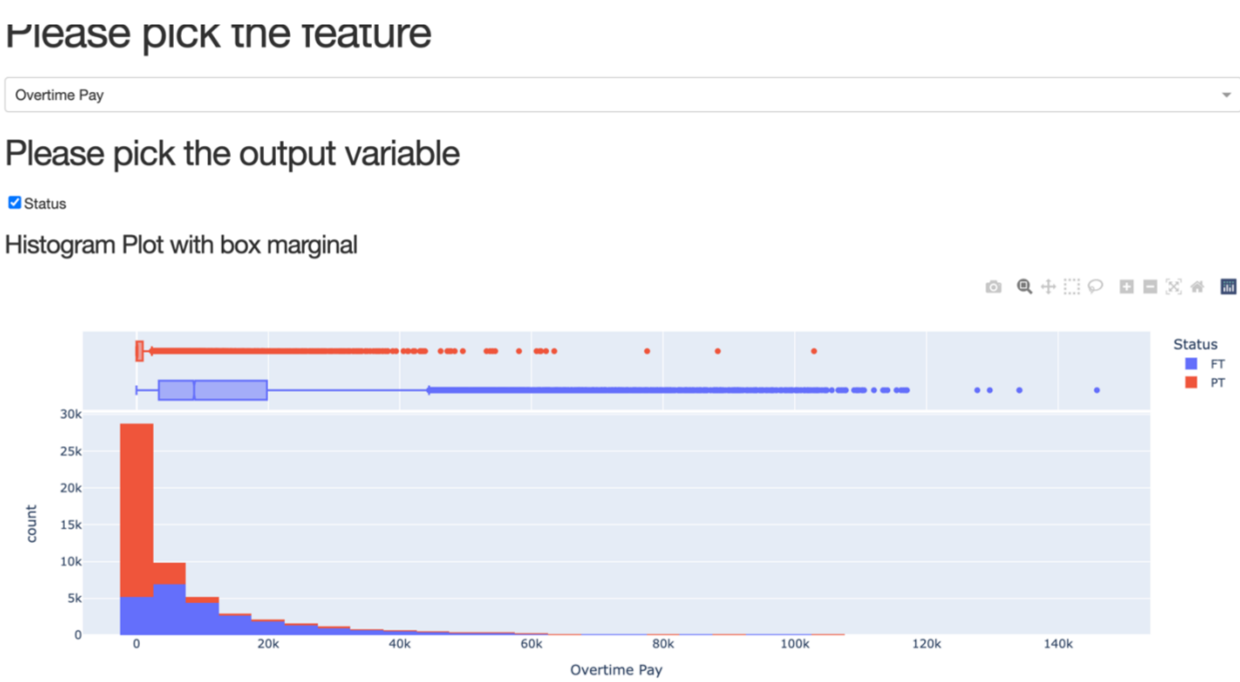
This is one of the line plot graphs I used for Dash, we can choose the type of pay and see which year has the most total pay and which year has the least.



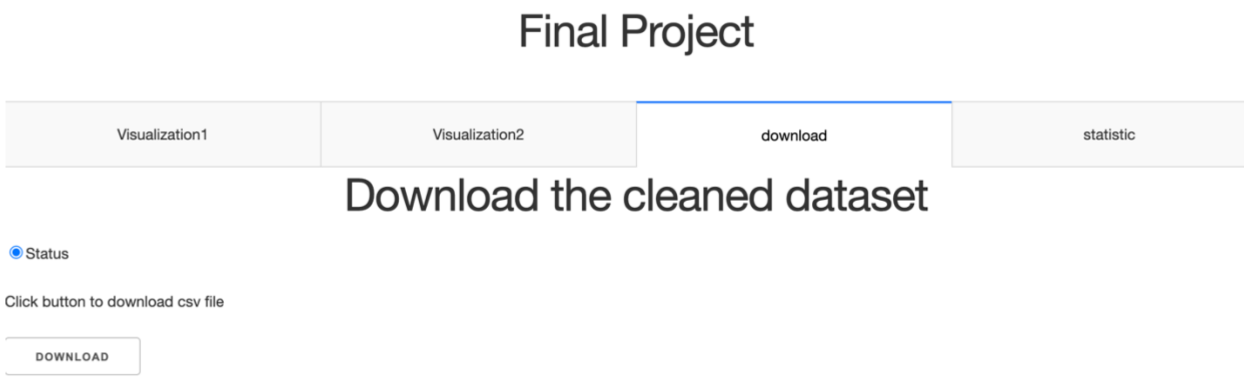
This is another violin plot that shows here. The point I want to add for these plots is for the year 2016 we didn't have the data, but I didn't want to remove it from the visualization. I wanted to give the viewer the info that we don't have any info here.



The second tab has the histogram plot with violin marginal to show the various pays for full time and part time employees.



This is another plot just with box marginal.



This is Tab 3 which give the viewer the possibility to download the raw dataset.

Radiobox to select Variable

- ○ Overtime Pay
- ○ Base Pay
- ○ Other Pay
- ◉ Benefits
- ○ Total Pay
- ○ Total Pay & Benefits

This Variable is Numeric

mean : 19726.023964170035

median : 23647.91

standard deviation : 16440.618470722166

And this is the last lab which with radio box, you can choose the variable and it can calculate some basic statistics here.

Recommendations:

during this project, I've learned how different plots can be helpful in different situations and when and why use them. The dashboards make the work easier for visualization and for presenting the data. Creating the app is so functional, not only for the visualization but also for some statistics and downloads and uploads.

Appendix for the first part(Seaborn) :

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import norm
from sklearn.preprocessing import StandardScaler
import plotly.express as px
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from numpy import linalg as LA
import numpy as np
import scipy.stats as st
import statsmodels.api as sm
import dash as dash
from dash import dcc
from dash import html
from dash.dependencies import Input,Output
import plotly.express as px
import numpy as np
from scipy.fft import fft
from dash.exceptions import PreventUpdate
import statistics
import warnings
warnings.filterwarnings('ignore')





#read the data
df=pd.read_csv('san-francisco-payroll_2011-2019.csv')


#Preproccessing

print(df.isnull().sum())
df=df.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
status_percent_null=151501/len(df['Status'])
print(status_percent_null)

print(df.isnull().sum())
print(len(df))
df=df[df['Base Pay'] != 'Not Provided']
df=df[df['Benefits'] != 'Not Provided']
print(len(df))

job_title=['Transit Operator', 'Special Nurse', 'Registered Nurse',
'Firefighter', 'Custodian', 'Police Officer 3', 'Public Service Trainee',
'Recreation Leader', 'Public Svc Aide-Public Works', 'Patient Care
Assistant']
df = df.loc[df['Job Title'].isin(job_title)]
#df['Job Title']=df.loc[df['Job Title'].isin([job_title])]


df['Overtime Pay'] = pd.to_numeric(df['Overtime Pay'])
```

```
df['Base Pay'] = pd.to_numeric(df['Base Pay'])
df['Other Pay'] = pd.to_numeric(df['Other Pay'])
df['Benefits'] = pd.to_numeric(df['Benefits'])
df['Total Pay'] = pd.to_numeric(df['Total Pay'])
df['Total Pay & Benefits'] = pd.to_numeric(df['Total Pay & Benefits'])
df['Year'] = pd.to_numeric(df['Year'])
print('this is the head of the dataset:\n',df.head())

#heatmap
corr=df.corr()
ax =sns.heatmap(corr,annot=True)
plt.show()

#outlier detection
df1=df.copy()
cols_names = ['Base Pay','Overtime Pay','Other Pay','Benefits','Total
Pay','Total Pay & Benefits','Year']

for i in cols_names:
    q1_h, q2_h, q3_h = df1[i].quantile([0.25,0.5,0.75])

    IQR_h = q3_h - q1_h
    lower1 = q1_h - 1.5*IQR_h
    upper1 = q3_h + 1.5*IQR_h

    print(f'Q1 and Q3 of the {i} is {q1_h:.2f}  & {q3_h:.2f} ')
    print(f'IQR for the {i} is {IQR_h:.2f} ')
    print(f'Any {i} < {lower1:.2f}  and {i} > {upper1:.2f}  is an outlier')

    sns.boxplot(y=df1[i])
    plt.title(f'Boxplot of {i} before removing outliers')
    plt.show()


    df1 = df1[(df1[i] > lower1) & (df1[i] < upper1)]

    sns.boxplot(y=df1[i])
    plt.title(f'Boxplot of {i} after removing outliers')
    plt.show()

#PCA
X=df1[df1._get_numeric_data().columns.to_list()[:-1]]

H=np.matmul(X.T,X)
_,d,_=np.linalg.svd(H)
print(f'original data: singular values {d}')
print(f'original data: condition number {LA.cond(X)}')
sns.heatmap(corr,annot=True)
plt.title('co-eff co-rell between original features')
plt.show()

X=X.values
X=StandardScaler().fit_transform(X)
pca=PCA(n_components='mle',svd_solver='full')
pca.fit(X)
X_PCA=pca.transform(X)
```

```python
print('Original Dim',X.shape)
print('Transform Data',X_PCA.shape)
print(f'explained variance ratio{pca.explained_variance_ratio_}')

#we can remove 2 features out of 6 features.
#we can keep 4 of the features to get the explained variance ratio more than
95.

plt.figure()
x=np.arange(1,len(np.cumsum(pca.explained_variance_ratio_))+1)
plt.xticks(x)
plt.plot(x,np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('number of componenets')
plt.ylabel('cumlative explained variance')
plt.title('cumlative explained variance vs number of componenets')
plt.show()

a,b=X_PCA.shape
print(a,b)

column=[]
for i in range(b):
    column.append(f'Principal Col{i}')
df_PCA=pd.DataFrame(data=X_PCA,columns=column)

total_df_redused=pd.DataFrame(df_PCA).corr()
sns.heatmap(total_df_redused,annot=True)
plt.title('heatmap for redused features')
plt.show()

print(df_PCA.head(5))

#########Normality test
kstest_Overtime_Pay = st.kstest(df['Overtime Pay'],'norm')
print(f"K-S test: statistics={kstest_Overtime_Pay[0]:.5f}, p-
value={kstest_Overtime_Pay[1]:.5f}")
print(f"K-S test: Overtime-pay column looks {'Normal' if
kstest_Overtime_Pay[1] > 0.01 else 'Non-Normal'}")

kstest_Base_Pay = st.kstest(df['Base Pay'],'norm')
print(f"K-S test: statistics={kstest_Base_Pay[0]:.5f}, p-
value={kstest_Base_Pay[1]:.5f}")
print(f"K-S test: Base-pay column looks {'Normal' if kstest_Base_Pay[1] >
0.01 else 'Non-Normal'}")

kstest_Other_Pay = st.kstest(df['Other Pay'],'norm')
print(f"K-S test: statistics={kstest_Other_Pay[0]:.5f}, p-
value={kstest_Other_Pay[1]:.5f}")
print(f"K-S test: other-pay column looks {'Normal' if kstest_Other_Pay[1] >
0.01 else 'Non-Normal'}")


kstest_Benefits = st.kstest(df['Benefits'],'norm')
print(f"K-S test: statistics={kstest_Benefits[0]:.5f}, p-
value={kstest_Benefits[1]:.5f}")
print(f"K-S test: Benefit column looks {'Normal' if kstest_Benefits[1] > 0.01
else 'Non-Normal'}")
```

```
kstest_Total_Pay = st.kstest(df['Total Pay'],'norm')
print(f"K-S test: statistics={kstest_Total_Pay[0]:.5f}, p-
value={kstest_Total_Pay[1]:.5f}")
print(f"K-S test: Total Pay column looks {'Normal' if kstest_Total_Pay[1] >
0.01 else 'Non-Normal'}")

kstest_Total_Pay_benefits = st.kstest(df['Total Pay & Benefits'],'norm')
print(f"K-S test: statistics={kstest_Total_Pay_benefits[0]:.5f}, p-
value={kstest_Total_Pay_benefits[1]:.5f}")
print(f"K-S test: Total Pay & Benefits column looks {'Normal' if
kstest_Total_Pay_benefits[1] > 0.01 else 'Non-Normal'}")

kstest_Year = st.kstest(df['Year'],'norm')
print(f"K-S test: statistics={kstest_Year[0]:.5f}, p-
value={kstest_Year[1]:.5f}")
print(f"K-S test: Year column looks {'Normal' if kstest_Year[1] > 0.01 else
'Non-Normal'}")

#Normalize the overtime_pay
new_overtime_pay = st.norm.ppf(st.rankdata(df['Overtime
Pay'])/(len(df['Overtime Pay']) + 1))
fig, axes = plt.subplots(2,2,figsize=(9,7))
sns.set_style('darkgrid')
sns.lineplot(x=np.linspace(1,54647,54647),y=df['Overtime
Pay'],ax=axes[0,0]).set(xlabel= '# of samples',ylabel='Magnitude')
axes[0,0].set_title('Non-Gaussian data')

sns.set_style('darkgrid')
sns.lineplot(x=np.linspace(1,54647,54647),y=new_overtime_pay,ax=axes[0,1]).se
t(xlabel= '# of samples',ylabel='Magnitude')
axes[0,1].set_title('Transformed data (Gaussian)')

sns.set_style('darkgrid')
sns.histplot(x=df['Overtime
Pay'],bins=100,ax=axes[1,0]).set(xlabel='Magnitude')
axes[1,0].set_title('Histogram of Non-Gaussian Data')

sns.set_style('darkgrid')
sns.histplot(x=new_overtime_pay,bins=100,ax=axes[1,1]).set(xlabel='Magnitude'
)
axes[1,1].set_title('Histogram of Transformed data (Gaussian)')
plt.tight_layout()
plt.show()

#Normalize the Base_pay
new_Base_pay = st.norm.ppf(st.rankdata(df['Base Pay'])/(len(df['Base Pay']) +
1))
fig, axes = plt.subplots(2,2,figsize=(9,7))
sns.set_style('darkgrid')
sns.lineplot(x=np.linspace(1,54647,54647),y=df['Base
Pay'],ax=axes[0,0]).set(xlabel= '# of samples',ylabel='Magnitude')
axes[0,0].set_title('Non-Gaussian data')

sns.set_style('darkgrid')
sns.lineplot(x=np.linspace(1,54647,54647),y=new_Base_pay,ax=axes[0,1]).set(xl
abel= '# of samples',ylabel='Magnitude')
```

```
axes[0,1].set_title('Transformed data (Gaussian)')

sns.set_style('darkgrid')
sns.histplot(x=df['Base Pay'],bins=100,ax=axes[1,0]).set(xlabel='Magnitude')
axes[1,0].set_title('Histogram of Non-Gaussian Data')

sns.set_style('darkgrid')
sns.histplot(x=new_Base_pay,bins=100,ax=axes[1,1]).set(xlabel='Magnitude')
axes[1,1].set_title('Histogram of Transformed data (Gaussian)')
plt.tight_layout()
plt.show()

#Normalize the Other Pay
new_other_pay = st.norm.ppf(st.rankdata(df['Other Pay'])/(len(df['Other
Pay']) + 1))
fig, axes = plt.subplots(2,2,figsize=(9,7))
sns.set_style('darkgrid')
sns.lineplot(x=np.linspace(1,54647,54647),y=df['Other
Pay'],ax=axes[0,0]).set(xlabel= '# of samples',ylabel='Magnitude')
axes[0,0].set_title('Non-Gaussian data')

sns.set_style('darkgrid')
sns.lineplot(x=np.linspace(1,54647,54647),y=new_other_pay,ax=axes[0,1]).set(x
label= '# of samples',ylabel='Magnitude')
axes[0,1].set_title('Transformed data (Gaussian)')

sns.set_style('darkgrid')
sns.histplot(x=df['Base Pay'],bins=100,ax=axes[1,0]).set(xlabel='Magnitude')
axes[1,0].set_title('Histogram of Non-Gaussian Data')

sns.set_style('darkgrid')
sns.histplot(x=new_other_pay,bins=100,ax=axes[1,1]).set(xlabel='Magnitude')
axes[1,1].set_title('Histogram of Transformed data (Gaussian)')
plt.tight_layout()
plt.show()

#Normalize the Benefits
new_Benefits = st.norm.ppf(st.rankdata(df['Benefits'])/(len(df['Benefits']) +
1))
fig, axes = plt.subplots(2,2,figsize=(9,7))
sns.set_style('darkgrid')
sns.lineplot(x=np.linspace(1,54647,54647),y=df['Benefits'],ax=axes[0,0]).set(
xlabel= '# of samples',ylabel='Magnitude')
axes[0,0].set_title('Non-Gaussian data')

sns.set_style('darkgrid')
sns.lineplot(x=np.linspace(1,54647,54647),y=new_Benefits,ax=axes[0,1]).set(xl
abel= '# of samples',ylabel='Magnitude')
axes[0,1].set_title('Transformed data (Gaussian)')

sns.set_style('darkgrid')
sns.histplot(x=df['Benefits'],bins=100,ax=axes[1,0]).set(xlabel='Magnitude')
axes[1,0].set_title('Histogram of Non-Gaussian Data')

sns.set_style('darkgrid')
sns.histplot(x=new_Benefits,bins=100,ax=axes[1,1]).set(xlabel='Magnitude')
axes[1,1].set_title('Histogram of Transformed data (Gaussian)')
```

```
plt.tight_layout()
plt.show()

#Normalize the Total Pay
new_Total_pay = st.norm.ppf(st.rankdata(df['Total Pay'])/(len(df['Total
Pay']) + 1))
fig, axes = plt.subplots(2,2,figsize=(9,7))
sns.set_style('darkgrid')
sns.lineplot(x=np.linspace(1,54647,54647),y=df['Total
Pay'],ax=axes[0,0]).set(xlabel= '# of samples',ylabel='Magnitude')
axes[0,0].set_title('Non-Gaussian data')

sns.set_style('darkgrid')
sns.lineplot(x=np.linspace(1,54647,54647),y=new_Total_pay,ax=axes[0,1]).set(x
label= '# of samples',ylabel='Magnitude')
axes[0,1].set_title('Transformed data (Gaussian)')

sns.set_style('darkgrid')
sns.histplot(x=df['Benefits'],bins=100,ax=axes[1,0]).set(xlabel='Magnitude')
axes[1,0].set_title('Histogram of Non-Gaussian Data')

sns.set_style('darkgrid')
sns.histplot(x=new_Total_pay,bins=100,ax=axes[1,1]).set(xlabel='Magnitude')
axes[1,1].set_title('Histogram of Transformed data (Gaussian)')
plt.tight_layout()
plt.show()

#Normalize the Total Pay & Benefits
new_Total_pay_benefits = st.norm.ppf(st.rankdata(df['Total Pay &
Benefits'])/(len(df['Total Pay & Benefits']) + 1))
fig, axes = plt.subplots(2,2,figsize=(9,7))
sns.set_style('darkgrid')
sns.lineplot(x=np.linspace(1,54647,54647),y=df['Total Pay &
Benefits'],ax=axes[0,0]).set(xlabel= '# of samples',ylabel='Magnitude')
axes[0,0].set_title('Non-Gaussian data')

sns.set_style('darkgrid')
sns.lineplot(x=np.linspace(1,54647,54647),y=new_Total_pay_benefits,ax=axes[0,
1]).set(xlabel= '# of samples',ylabel='Magnitude')
axes[0,1].set_title('Transformed data (Gaussian)')

sns.set_style('darkgrid')
sns.histplot(x=df['Total Pay &
Benefits'],bins=100,ax=axes[1,0]).set(xlabel='Magnitude')
axes[1,0].set_title('Histogram of Non-Gaussian Data')

sns.set_style('darkgrid')
sns.histplot(x=new_Total_pay_benefits,bins=100,ax=axes[1,1]).set(xlabel='Magn
itude')
axes[1,1].set_title('Histogram of Transformed data (Gaussian)')
plt.tight_layout()
plt.show()

#Normalize the Year
new_year = st.norm.ppf(st.rankdata(df['Year'])/(len(df['Year']) + 1))
fig, axes = plt.subplots(2,2,figsize=(9,7))
sns.set_style('darkgrid')
```

```python
sns.lineplot(x=np.linspace(1,54647,54647),y=df['Year'],ax=axes[0,0]).set(xlab
el= '# of samples',ylabel='Magnitude')
axes[0,0].set_title('Non-Gaussian data')

sns.set_style('darkgrid')
sns.lineplot(x=np.linspace(1,54647,54647),y=new_year,ax=axes[0,1]).set(xlabel
= '# of samples',ylabel='Magnitude')
axes[0,1].set_title('Transformed data (Gaussian)')

sns.set_style('darkgrid')
sns.histplot(x=df['Year'],bins=100,ax=axes[1,0]).set(xlabel='Magnitude')
axes[1,0].set_title('Histogram of Non-Gaussian Data')

sns.set_style('darkgrid')
sns.histplot(x=new_year,bins=100,ax=axes[1,1]).set(xlabel='Magnitude')
axes[1,1].set_title('Histogram of Transformed data (Gaussian)')
plt.tight_layout()
plt.show()

#Statistic
print('the median of the base pay is:',statistics.median(df['Base Pay']))
print('the mean of the base pay is:',statistics.mean(df['Base Pay']))

print('the median of the Overtime Pay is:',statistics.median(df['Overtime
Pay']))
print('the mean of the Overtime Pay is:',statistics.mean(df['Overtime Pay']))

print('the median of the Other Pay is:',statistics.median(df['Other Pay']))
print('the mean of the Other Pay is:',statistics.mean(df['Other Pay']))

print('the median of the Benefits is:',statistics.median(df['Benefits']))
print('the mean of the Benefits is:',statistics.mean(df['Benefits']))

print('the median of the Total Pay is:',statistics.median(df['Total Pay']))
print('the mean of the Total Pay is:',statistics.mean(df['Total Pay']))

print('the median of the Total Pay & Benefits
is:',statistics.median(df['Total Pay & Benefits']))
print('the mean of the Total Pay & Benefits is:',statistics.mean(df['Total
Pay & Benefits']))


#Visualization
#line_plot
sns.lineplot(data='df',x=df['Year'],y=df['Total Pay & Benefits'])
plt.xlabel('year')
plt.ylabel('Total Pay and Benefits')
plt.title('line plot of Benefits and Salary changing during years')
plt.show()
#bar plot
ax = sns.barplot(x=df['Year'], y=df['Benefits'], hue="Status", data=df)
plt.xlabel('Year')
plt.ylabel('benefits')
plt.title('bar plot of Benefits in each year')
plt.show()
#count plot
ax = sns.countplot(x="Status", data=df)
```

```
plt.xlabel('Status')
plt.ylabel('number of status')
plt.title('count plot of full-time & Part-time employess')
plt.show()
#cat plot

sns.catplot(x = 'Job Title',y = 'Overtime Pay',data=df,
            kind ='bar',
            height = 8 , aspect= 1.5)
plt.title('cat plot of job title and overtime pay')
plt.show()

#pie plot for job title
df['Job Title'].value_counts().plot(kind='pie')
plt.title('pie plot for job title')
plt.show()

df_2014=df[df['Year'] == 2014]
df_2015=df[df['Year'] == 2015]
#df_2016=df[df['Year'] == 2016]
df_2017=df[df['Year'] == 2017]
df_2018=df[df['Year'] == 2018]
df_2019=df[df['Year'] == 2019]

s14=df_2014['Status'].value_counts().to_dict()
s15=df_2015['Status'].value_counts().to_dict()
#s16=df_2016['Status'].value_counts().to_dict()
s17=df_2017['Status'].value_counts().to_dict()
s18=df_2018['Status'].value_counts().to_dict()
s19=df_2019['Status'].value_counts().to_dict()

#sub plot and pie plot

fig=plt.figure(figsize=(10,8))
plt.subplot(3,2,1)
plt.pie(s14.values(), labels=s14.keys())
plt.title("pie chart of 2014 status")
plt.subplot(3,2,2)
plt.pie(s15.values(), labels=s15.keys())
plt.title("pie chart of 2015 status")

plt.subplot(3,2,3)
plt.pie(s17.values(), labels=s17.keys())
plt.title("pie chart of 2017 status")
plt.subplot(3,2,4)
plt.pie(s18.values(), labels=s18.keys())
plt.title("pie chart of 2018 status")
plt.subplot(3,2,5)
plt.pie(s19.values(), labels=s19.keys())
plt.title("pie chart of 2019 status")
plt.show()
#displot
sns.displot(x=df['Total Pay'],kde=False)
plt.title('the displot of Total pay')
plt.show()
#total pay is aroun 100k to 200K for most of the titles
#pairplot
```

```
sns.pairplot(df)
plt.title('The pairplot of the datset')
plt.show()
#heatmap
sns.heatmap(corr)
plt.title('The Heatmap of the datset')
plt.show()
#hist-plot
sns.histplot(data=df, x=df['Total Pay'], hue=df['Status'],element="step",
             stat="density", common_norm=False)
plt.title('the histplot of Total pay ')
plt.show()

#QQ plot
fig = sm.qqplot(df['Base Pay'],line ='45')
plt.title('QQ plot for Base Pay')
plt.show()

#Kernal density estimate
sns.kdeplot(data=df,
            x='Total Pay',
            y='Base Pay',
            hue='Status',
            fill=True)

plt.title('Status Kernel density plot')
plt.show()

#Scatter plot and regression line using sklearn
sns.regplot(x='Overtime Pay', y='Total Pay', data=df, logistic=True, ci=None)
plt.title('Scatter and Regression plot on overtime and Total pay')
plt.show()
# # #Multivariate Box plot
sns.boxplot(x='Year',y='Base Pay',data=df)
plt.title('Multivariate Box plot of each year based pay')
plt.show()
#violon plot
sns.set_theme(style="whitegrid")
Ax = sns.violinplot(x="Year", y="Total Pay & Benefits", data=df,
palette="coolwarm")
plt.title('Violon plot of total-year and benefits base on the Year ')
plt.show()
```

Appendix for dashboard:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import norm
from sklearn.preprocessing import StandardScaler
import plotly.express as px
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from numpy import linalg as LA
import numpy as np
import scipy.stats as st
```

```
import statsmodels.api as sm
import dash as dash
from dash import dcc
from dash import html
from dash.dependencies import Input,Output
import plotly.express as px
import numpy as np
from scipy.fft import fft
from dash.exceptions import PreventUpdate


#read the data
df=pd.read_csv('san-francisco-payroll_2011-2019.csv')


#Preproccessing

print(df.isnull().sum())
df=df.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
status_percent_null=151501/len(df['Status'])
print(status_percent_null)

print(df.isnull().sum())
print(len(df))
df=df[df['Base Pay'] != 'Not Provided']
df=df[df['Benefits'] != 'Not Provided']
print(len(df))

job_title=['Transit Operator', 'Special Nurse', 'Registered Nurse',
'Firefighter', 'Custodian', 'Police Officer 3', 'Public Service Trainee',
'Recreation Leader', 'Public Svc Aide-Public Works', 'Patient Care
Assistant']
df = df.loc[df['Job Title'].isin(job_title)]
#df['Job Title']=df.loc[df['Job Title'].isin([job_title])]


df['Overtime Pay'] = pd.to_numeric(df['Overtime Pay'])
df['Base Pay'] = pd.to_numeric(df['Base Pay'])
df['Other Pay'] = pd.to_numeric(df['Other Pay'])
df['Benefits'] = pd.to_numeric(df['Benefits'])
df['Total Pay'] = pd.to_numeric(df['Total Pay'])
df['Total Pay & Benefits'] = pd.to_numeric(df['Total Pay & Benefits'])
df['Year'] = pd.to_numeric(df['Year'])
print('this is the head of the dataset:\n',df.head())

#Dashbord
external_stylesheets=['https://codepen.io/chriddyp/pen/bWLwgP.css']
my_app = dash.Dash('My app',external_stylesheets=external_stylesheets)

my_app.layout=html.Div([
    html.H1('Final Project',style={'textAlign':'center'}),
    html.Br(),
    dcc.Tabs(id='hw-questions',
             children=[
                 dcc.Tab(label='Visualization1',value='Visualization1'),
                 dcc.Tab(label='Visualization2',value='Visualization2'),
                 dcc.Tab(label = 'download', value = 'download'),
```

```
                 dcc.Tab(label = 'statistic', value = 'statistic')
             ]),
    html.Div(id='layout')
])

Visualization_layout=html.Div([
    html.H1('Please pick the feature'),
    dcc.Dropdown(id = 'Status',
                 options=[
                     {'label':'Base Pay', 'value':'Base Pay'},
                     {'label':'Overtime Pay', 'value':'Overtime Pay'},

                 ], value='Base Pay', clearable=False),
    html.H2('Please pick the output variable'),
    dcc.Dropdown(id = 'Year',
                 options=[
                     {'label':'Year', 'value':'Year'},
                     {'label':'Status', 'value':'Status'},

                 ], value='Year', clearable=False),
    html.H4('Line Plot'),
    dcc.Graph(id='my-graph'),
    html.Br(),
    html.H1('Please pick the feature'),
    dcc.Dropdown(id='Benefits',options=[
        {'label':'Overtime Pay', 'value':'Overtime Pay'},
        {'label':'Base Pay', 'value':'Base Pay'},
        {'label':'Other Pay', 'value':'Other Pay'},
        {'label':'Benefits', 'value':'Benefits'},
        {'label':'Total Pay', 'value':'Total Pay'},
        {'label':'Total Pay & Benefits', 'value':'Total Pay & Benefits'}

    ], value='Overtime Pay', clearable=False),
    html.H2('Please pick the output variable'),
    dcc.RadioItems(['Year'],value='Year',id='date'),
    html.H4('Violin Plot'),
    dcc.Graph(id='my-graph2')

])
Visualization2_layout=html.Div([
    html.H1('Please pick the feature'),
    dcc.Dropdown(id='Benefits',options=[
        {'label':'Overtime Pay', 'value':'Overtime Pay'},
        {'label':'Base Pay', 'value':'Base Pay'},
        {'label':'Other Pay', 'value':'Other Pay'},
        {'label':'Benefits', 'value':'Benefits'},
        {'label':'Total Pay', 'value':'Total Pay'},
        {'label':'Total Pay & Benefits', 'value':'Total Pay & Benefits'}

         ], value='Overtime Pay', clearable=False),
    html.H2('Please pick the output variable'),
    dcc.RadioItems(['Status'],value='Status',id='Status'),
    html.H4('Histogram Plot with violin marginal'),
    dcc.Graph(id='my-graph3'),
    html.Br(),
    html.Br(),
    html.H1('Please pick the feature'),
```

```
    dcc.Dropdown(id='Benefits2',options=[
        {'label':'Overtime Pay', 'value':'Overtime Pay'},
        {'label':'Base Pay', 'value':'Base Pay'},
        {'label':'Other Pay', 'value':'Other Pay'},
        {'label':'Benefits', 'value':'Benefits'},
        {'label':'Total Pay', 'value':'Total Pay'},
        {'label':'Total Pay & Benefits', 'value':'Total Pay & Benefits'}

    ], value='Overtime Pay', clearable=False),
    html.H2('Please pick the output variable'),
    dcc.Checklist(['Status'],value='Status',id='Status'),
    html.H4('Histogram Plot with box marginal'),
    dcc.Graph(id='my-graph4'),
])

download_layout=html.Div([
    html.H1('Download the cleaned dataset', style={'textAlign':'center'}),
    html.Br(),
    html.Label('Click button to download csv file'),
    html.Br(),
    html.Button(id='download1', children='Download'),
    dcc.Download(id='download2')
])
#call back for download
@my_app.callback(Output(component_id="download2", component_property="data"),
              Input(component_id="download1", component_property="n_clicks"),
              prevent_initial_call=True)

def displaydown_layout(sell):
    return dcc.send_data_frame(df.to_csv, "SF_payroll.csv")

statistic_layout=html.Div([
    html.H1('Basic statistics', style={'textAlign':'center'}),
    html.Br(),
    html.Br(),
    html.P('Radiobox to select Variable'),
    html.Br(),
    dcc.RadioItems(
        id='radio',
        options=[
            {'label':'Overtime Pay', 'value':'Overtime Pay'},
            {'label':'Base Pay', 'value':'Base Pay'},
            {'label':'Other Pay', 'value':'Other Pay'},
            {'label':'Benefits', 'value':'Benefits'},
            {'label':'Total Pay', 'value':'Total Pay'},
            {'label':'Total Pay & Benefits', 'value':'Total Pay & Benefits'},
        ],value=['Overtime Pay'],
    ),
    html.Br(),
    html.Br(),
    html.H2(id='output1'),
    html.Br(),
    html.H2(id='output2'),
    html.Br(),
    html.H2(id='output3'),
    html.Br(),
    html.H2(id='output4'),
```

```python
    html.Br(),
    html.H2(id='output5')
])

@my_app.callback(
    [Output(component_id='output1', component_property='children'),
     Output(component_id='output2', component_property='children'),
     Output(component_id='output3', component_property='children'),
     Output(component_id='output4', component_property='children'),
     Output(component_id='output5', component_property='children'),],
    [Input(component_id='radio',component_property='value'),]
)

def display_color(sel1):

        sent = f' This Variable is Numeric'
        mean_df = f'mean : {np.mean(df[sel1])}'
        median_df = f'median : {np.median(df[sel1])}'
        std_df = f'standard deviation : {np.std(df[sel1])}'
        var_df = f'Variance : {np.var(df[sel1])}'

        return sent, mean_df, median_df, std_df, var_df

#first tab1-graph lay out
@my_app.callback(
    Output(component_id='my-graph', component_property='figure'),
    [Input(component_id='Status', component_property='value'),
     Input(component_id='Year', component_property='value'),]
)
def undate_n(Status, Year):
    fig=px.line(df,x=Year,y=Status)
    return fig
#second graph call back
@my_app.callback(
    Output(component_id='my-graph2', component_property='figure'),
    [Input(component_id='Benefits', component_property='value'),
     Input(component_id='date', component_property='value')]
)
def undate_n2(Benefits, date):

    #fig = px.pie(df,values=Year, names=Status)
    fig=px.violin(df,x=date,y=Benefits)
    return fig


#graph3 in tab2
@my_app.callback(
    Output(component_id='my-graph3', component_property='figure'),
    [Input(component_id='Status', component_property='value'),
     Input(component_id='Benefits', component_property='value'),]
)
def undate_3(Status, Benefits):

    fig = px.histogram(df, x=Benefits,color=Status,
                       nbins=50, marginal = 'violin')
    return fig
```

```
#graph4 in tab2
@my_app.callback(
    Output(component_id='my-graph4', component_property='figure'),
    [Input(component_id='Status', component_property='value'),
     Input(component_id='Benefits2', component_property='value'),]
)
def undate_4(Status, Benefits):

    fig = px.histogram(df, x=Benefits,color=Status,
                       nbins=50, marginal = 'box')
    return fig


#tabs
@my_app.callback(Output(component_id='layout',component_property='children'),

                 [Input(component_id='hw-
questions',component_property='value'),
                  ])

def update_layout(ques):
    if ques=='Visualization1':
        return Visualization_layout
    elif ques=='Visualization2':
        return Visualization2_layout
    elif ques=='download':
        return download_layout
    elif ques=='statistic':
        return statistic_layout

my_app.run_server(port=8093,
                  host='0.0.0.0')
```

References:

- Class codes and lectures and recording
- https://www.kaggle.com/datasets/tomtillo/san-francisco-city-payrollsalary-data-20112019
- https://dash.plotly.com/dash-core-components/checklist
- https://dash.plotly.com
- https://github.com/rjafari979
- https://data.library.virginia.edu/understanding-q-q-plots/
- https://github.com
- https://r.search.yahoo.com/_ylt=AwrEeGH_u3FibGgAYQAPxQt.;_ylu=Y29sbwNiZjEEcG9zAzEEdnRpZAMEc2VjA3Nj/RV=2/RE=1651649664/RO=10/RU=https%3A%2F%2Fseaborn.pydata.org%2Fgenerated%2Fseaborn.kdeplot.html%23%3A~%3Atext%3DA%2520kernel%2520density%2520estimate%2520%2528KDE%2529%2520plot%2520is%2520a%2Capproach%2520is%2520explained%2520further%2520in%2520the%2520user%2520guide./RK=2/RS=CanTGUmIWM2ALAtqvkUceWU9WPI-

- https://r.search.yahoo.com/_ylt=AwrEeBksw3FitFEAqwQPxQt.;_ylu=Y29sbwNiZjEEcG9zAzEEdnRpZAMEc2VjA3Nj/RV=2/RE=1651651501/RO=10/RU=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FViolin_plot/RK=2/RS=FbF3cUqzbIhOrir5n5aJiUVyZGA-
- https://dash.plotly.com/dash-core-components/download
- https://r.search.yahoo.com/_ylt=AwrEeBksw3FitFEAqwQPxQt.;_ylu=Y29sbwNiZjEEcG9zAzEEdnRpZAMEc2VjA3Nj/RV=2/RE=1651651501/RO=10/RU=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FViolin_plot/RK=2/RS=FbF3cUqzbIhOrir5n5aJiUVyZGA-
- https://www.geeksforgeeks.org/detect-and-remove-the-outliers-using-python/#:~:text=IQR%20%28Inter%20Quartile%20Range%29%20Inter%20Quartile%20Range%20approach,%3D%20np.percentile%20%28df_boston%20%5B%27DIS%27%5D%2C%2075%2C%20interpolation%20%3D%20%27midpoint%27%29