| | |
|---|---|
| Project Title | **Salaries for San Francisco Employee** |
| Tools | Visual Studio code / jupyter notebook |
| Domain | Finance Analyst |
| Project Difficulties level | Advance |

Dataset : Dataset is available in the given link. You can download it at your convenience.

[Click here to download data set]

**About Dataset**

**Context**

This Dataset contains more than 300k employee records found in San Francisco from 2011 to 2018.

Complete and accurate information is necessary to increase public understanding of government and help decision makers, including elected officials and voters, make informed decisions.

This Dataset is provided by the Nevada Policy Research Institute as a public service and is dedicated to providing accurate, comprehensive and easily searchable information on the compensation of public employees in California.

**<u>From her, you can get a basic idea about how you can create a project.</u>**

**Machine Learning Project: Google Play Store Analysis using Salary Dataset**

**Objective:**

The project aims to analyze employee compensation data, including BasePay, OvertimePay, OtherPay, Benefits, and their relation to TotalPay and TotalPayBenefits. This is achieved through **Exploratory Data Analysis (EDA)** and **Visualization** using Python.

---

**Dataset Overview:**

Columns in the dataset:

- **EmployeeName**: Name of the employee.
- **JobTitle**: Title of the job.
- **BasePay**: Base salary pay.
- **OvertimePay**: Pay for overtime work.
- **OtherPay**: Any other types of compensation.
- **Benefits**: Benefits provided to the employee.
- **TotalPay**: The total pay without benefits.
- **TotalPayBenefits**: Total pay with benefits included.
- **Year**: The year of the payroll record.

---

**Step 1: Importing Required Libraries**

First, let's import the necessary libraries like Pandas, NumPy, Matplotlib, and

Seaborn for data analysis and visualization.

```
# Importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# For display settings
pd.set_option('display.max_columns', None)

# Load the dataset
df = pd.read_csv('employee_salary.csv')

# Display the first few rows
df.head()
```

**Step 2: Data Cleaning**

In this step, we will clean the dataset by handling missing values, converting data types, and performing basic data exploration.

1. **Checking for Missing Values**:

```
# Check for missing values
```

```
print(df.isnull().sum())

# Dropping rows with missing values in key columns (if
necessary)
df = df.dropna(subset=['BasePay', 'TotalPayBenefits'])
```

2. **Convert Data Types** (if necessary):

```
# Converting columns to appropriate data types if needed
df['Year'] = df['Year'].astype(int)

# Verifying data types
print(df.dtypes)
```

3. **Handling Negative or Zero Pay Values**:

```
# Filter out rows where TotalPay or TotalPayBenefits are 0 or
negative
df = df[(df['TotalPay'] > 0) & (df['TotalPayBenefits'] > 0)]

# Check updated dataset
df.describe()
```

**Step 3: Exploratory Data Analysis (EDA)**

**3.1 Descriptive Statistics**

Let's explore summary statistics of the dataset:

python

Copy code

```
# Summary statistics
df.describe()
```

**3.2 Top 10 Highest Paying Job Titles**

```
# Group by job title and get the mean TotalPay
job_salary =
df.groupby('JobTitle')['TotalPay'].mean().sort_values(ascending
=False).head(10)

# Plot
plt.figure(figsize=(10,6))
sns.barplot(x=job_salary.values, y=job_salary.index,
palette='Blues_d')
plt.title('Top 10 Highest Paying Job Titles')
plt.xlabel('Average Total Pay')
plt.show()
```

**3.3 Distribution of BasePay, OvertimePay, and OtherPay**

```python
# Plot histograms for BasePay, OvertimePay, and OtherPay
plt.figure(figsize=(15,5))

plt.subplot(1,3,1)
sns.histplot(df['BasePay'], bins=30, kde=True, color='blue')
plt.title('Distribution of BasePay')

plt.subplot(1,3,2)
sns.histplot(df['OvertimePay'], bins=30, kde=True,
color='green')
plt.title('Distribution of OvertimePay')

plt.subplot(1,3,3)
sns.histplot(df['OtherPay'], bins=30, kde=True, color='red')
plt.title('Distribution of OtherPay')

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

**3.4 Pay Over the Years**

```python
# Group by Year and calculate mean total pay
pay_over_years = df.groupby('Year')['TotalPay'].mean()
```

```
# Plot
plt.figure(figsize=(10,6))
sns.lineplot(x=pay_over_years.index, y=pay_over_years.values,
marker='o', color='purple')
plt.title('Average Total Pay Over the Years')
plt.xlabel('Year')
plt.ylabel('Average Total Pay')
plt.show()
```

**3.5 Correlation Heatmap**

```
# Correlation matrix
plt.figure(figsize=(8,6))
corr_matrix = df[['BasePay', 'OvertimePay', 'OtherPay',
'Benefits', 'TotalPay', 'TotalPayBenefits']].corr()

# Plotting heatmap
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm',
linewidths=0.5)
plt.title('Correlation Matrix of Pay Components')
plt.show()
```

**Step 4: Salary Prediction with Machine Learning**

### 4.1 Data Preprocessing

Before training a machine learning model, we will preprocess the dataset by handling categorical features and splitting the data into training and test sets.

1. **Handling Categorical Variables**:

```
# Encoding JobTitle using one-hot encoding
df = pd.get_dummies(df, columns=['JobTitle'], drop_first=True)

# Display new dataframe
df.head()
```

2. **Splitting the Data**:

```
from sklearn.model_selection import train_test_split

# Features and target variable
X = df.drop(columns=['EmployeeName', 'TotalPayBenefits'])
y = df['TotalPayBenefits']

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)
```

### 4.2 Model Training

We'll use a **Linear Regression** model for predicting employee salary based on features such as BasePay, OvertimePay, JobTitle, etc.

python

Copy code

```
from sklearn.linear_model import LinearRegression


# Initialize and train the model

model = LinearRegression()

model.fit(X_train, y_train)


# Predict on test data

y_pred = model.predict(X_test)
```

### 4.3 Model Evaluation

Evaluate the model using **Mean Absolute Error (MAE)** and **R-squared** score.

```
from sklearn.metrics import mean_absolute_error, r2_score


# Calculate MAE

mae = mean_absolute_error(y_test, y_pred)

print(f'Mean Absolute Error: {mae}')
```

```
# Calculate R-squared score
r2 = r2_score(y_test, y_pred)
print(f'R-squared Score: {r2}')
```

**Step 5: Conclusion**

1. **Key Insights**:
   - The average base pay is highly correlated with total compensation.
   - Job titles like "Chief Executive Officer" have the highest salaries.
   - Benefits contribute significantly to overall pay.
2. **Model Performance**:
   - The linear regression model performs with an MAE of X and an R-squared score of Y, suggesting reasonable prediction accuracy.

**Sample link**

```
# This Python 3 environment comes with many helpful analytics
```

```
libraries installed
# It is defined by the kaggle/python Docker image:
https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g.
pd.read_csv)

# Input data files are available in the read-only "../input/"
directory
# For example, running this (by clicking run or pressing
Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory
(/kaggle/working/) that gets preserved as output when you create
a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they
won't be saved outside of the current session
```

```
/kaggle/input/20112018-salaries-for-san-francisco/Total.csv
```

Step 1: Importing libraries

In [2]:

```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

Step2: Importing the data

In [3]:

```python
df=pd.read_csv('../input/20112018-salaries-for-san-francisco/Total.csv')
```

```
/opt/conda/lib/python3.7/site-packages/IPython/core/interactiveshell.py:3156: DtypeWarning: Columns (2,3,4,5) have mixed types.Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler,
```

```
result=result)
```

Step3: DataFrame Overview

In [4]:

```
df.head()
```

Out[4]:

| | EmployeeName | JobTitle | BasePay | OvertimePay | OtherPay | Benefits | TotalPay | TotalPayBenefits | Year |
|---|---|---|---|---|---|---|---|---|---|
| 0 | NATHANIEL FORD | GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY | 167411.18 | 0.0 | 400184.25 | Not Provided | 567595.43 | 567595.43 | 2011 |
| 1 | GARY JIMENEZ | CAPTAIN III (POLICE DEPARTMENT) | 155966.02 | 245131.88 | 137811.38 | Not Provided | 538909.28 | 538909.28 | 2011 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 2 | ALBERT PARDINI | CAPTAIN III (POLICE DEPARTMENT) | 212739.13 | 106088.18 | 16452.6 | Not Provided | 335279.91 | 335279.91 | 2011 |
| 3 | CHRISTOPHER CHONG | WIRE ROPE CABLE MAINTENANCE MECHANIC | 77916.0 | 56120.71 | 198306.9 | Not Provided | 332343.61 | 332343.61 | 2011 |
| 4 | PATRICK GARDNER | DEPUTY CHIEF OF DEPARTMENT,(FIRE DEPARTMENT) | 134401.6 | 9737.0 | 182234.59 | Not Provided | 326373.19 | 326373.19 | 2011 |

In [5]:

```
print('This Dataset countains {} Rows and {} Columns'.format(df.shape[0], df.shape[1]))
```

```
This Dataset countains 312882 Rows and 9 Columns
```

In [6]:

```
df.shape
```

Out[6]:

```
(312882, 9)
```

In [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 312882 entries, 0 to 312881
Data columns (total 9 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   EmployeeName       312882 non-null  object 
 1   JobTitle           312882 non-null  object 
 2   BasePay            312882 non-null  object 
 3   OvertimePay        312882 non-null  object 
 4   OtherPay           312882 non-null  object 
 5   Benefits           312882 non-null  object 
 6   TotalPay           312882 non-null  float64
 7   TotalPayBenefits   312882 non-null  float64
 8   Year               312882 non-null  int64  
dtypes: float64(2), int64(1), object(6)
memory usage: 21.5+ MB
```

In [8]:

```
series_list=['BasePay','OvertimePay','OtherPay','Benefits']
for series in series_list:
    df[series]=pd.to_numeric(df[series],errors='coerce')
```

In [9]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 312882 entries, 0 to 312881
Data columns (total 9 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   EmployeeName      312882 non-null  object
 1   JobTitle          312882 non-null  object
 2   BasePay           312276 non-null  float64
 3   OvertimePay       312881 non-null  float64
 4   OtherPay          312881 non-null  float64
 5   Benefits          276722 non-null  float64
 6   TotalPay          312882 non-null  float64
 7   TotalPayBenefits  312882 non-null  float64
```

```
 8   Year                 312882 non-null  int64
dtypes: float64(6), int64(1), object(2)
memory usage: 21.5+ MB
```

Step4: Descripitive Statistical Analysis

In [10]:

```
df['BasePay'].mean()
```

Out[10]:

```
69808.25749606262
```

In [11]:

```
df['BasePay'].max()
```

Out[11]:

```
592394.34
```

In [12]:

```
df['BasePay'].describe()
```

Out[12]:

```
count    312276.000000
mean      69808.257496
std       45376.929428
min        -474.400000
25%       35722.365000
50%       67710.450000
75%       99312.302500
max      592394.340000

Name: BasePay, dtype: float64
```

In [13]:

```
df.describe()
```

Out[13]:

| | BasePay | OvertimePay | OtherPay | Benefits | TotalPay | TotalPayBenefits | Year |
|---|---|---|---|---|---|---|---|
| count | 312276.000000 | 312881.000000 | 312881.000000 | 276722.000000 | 312882.000000 | 312882.000000 | 312882.000000 |
| me | 69808.25 | 5668.929 | 3460.694 | 25016.91 | 78802.64 | 100928.3 | 2014.625 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| an | 7496 | 393 | 974 | 7292 | 5788 | 39777 | 303 |
| std | 45376.929428 | 12745.655309 | 7387.263120 | 15089.077103 | 53230.758542 | 66485.186495 | 2.290899 |
| min | -474.400000 | -292.800000 | -7058.590000 | -13939.420000 | -618.130000 | -3628.780000 | 2011.000000 |
| 25% | 35722.365000 | 0.000000 | 0.000000 | 12729.762500 | 38803.000000 | 48955.072500 | 2013.000000 |
| 50% | 67710.450000 | 0.000000 | 728.000000 | 28327.330000 | 74908.790000 | 100011.290000 | 2015.000000 |
| 75% | 99312.302500 | 5223.120000 | 3958.680000 | 35268.162500 | 111386.897500 | 142376.300000 | 2017.000000 |
| max | 592394.340000 | 309481.030000 | 400184.250000 | 125891.730000 | 592394.340000 | 712802.360000 | 2018.000000 |

In [14]:

```
df[df['BasePay']<0]
```

Out[14]:

| | EmployeeName | JobTitle | BasePay | OvertimePay | OtherPay | Benefits | TotalPay | TotalPayBenefits | Year |
|---|---|---|---|---|---|---|---|---|---|
| 72832 | Irwin Sidharta | Junior Clerk | -166.01 | 249.02 | 0.00 | 6.56 | 83.01 | 89.57 | 2012 |
| 72865 | Robert Scott | Junior Clerk | -121.63 | 182.70 | 0.00 | 5.44 | 61.07 | 66.51 | 2012 |
| 72872 | Chung Huey Kung | Junior Clerk | -109.22 | 163.83 | 0.00 | 4.32 | 54.61 | 58.93 | 2012 |
| 72874 | Jordan Li | Junior Clerk | -106.60 | 159.90 | 0.00 | 4.66 | 53.30 | 57.96 | 2012 |
| 72878 | Richard Jackson | Junior Clerk | -101.88 | 153.08 | 0.00 | 4.55 | 51.20 | 55.75 | 2012 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 72884 | DiMarco McGhee-Stewart | Junior Clerk | -93.14 | 139.97 | 0.00 | 4.17 | 46.83 | 51.00 | 2012 |
| 72888 | Leopoldo Marasigan | Junior Clerk | -87.38 | 131.06 | 0.00 | 3.89 | 43.68 | 47.57 | 2012 |
| 72894 | Douglas Avalos | Junior Clerk | -75.67 | 113.76 | 0.00 | 3.39 | 38.09 | 41.48 | 2012 |
| 72908 | Norma Rodriguez | Junior Clerk | -59.59 | 89.65 | 0.00 | 2.68 | 30.06 | 32.74 | 2012 |
| 72920 | Charles Williams | Junior Clerk | -30.58 | 45.87 | 0.00 | 1.36 | 15.29 | 16.65 | 2012 |
| 72922 | John Draper | Clerk | -9.50 | 14.25 | 0.00 | 0.42 | 4.75 | 5.17 | 2012 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 188036 | Lubna Kaur | PS Aide Health Services | -292.40 | 0.00 | 0.00 | -2.92 | -292.40 | -295.32 | 2015 |
| 270571 | Carlos R Castro Santiago | Custodian | -474.40 | 0.00 | -23.72 | -79.35 | -498.12 | -577.47 | 2017 |

Step5: Elementry EDA

Exploring some insights about Employee Name:"Ricardo Jimenez"

In [15]:

```
df[df['EmployeeName']=='Ricardo Jimenez']
```

Out[15]:

| | EmployeeName | JobTitle | BasePay | OvertimePay | OtherPay | Benefits | TotalPay | TotalPayBenefits | Year |
|---|---|---|---|---|---|---|---|---|---|
| 50596 | Ricardo Jimenez | Transit Supervisor | 72936.93 | 8078.04 | 3701.18 | 31355.90 | 84716.15 | 116072.05 | 2012 |
| 120 | Ricardo | Transit | 8912 | 14206. | 2677 | 3391 | 1060 | 139924.9 | 20 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 452 | Jimenez | Supervisor | 8.98 | 09 | .35 | 2.52 | 12.42 | 4 | 14 |
| 160317 | Ricardo Jimenez | Transit Supervisor | 89623.29 | 8757.50 | 2556.00 | 32716.82 | 100936.79 | 133653.61 | 2015 |
| 198692 | Ricardo Jimenez | Transit Supervisor | 97131.01 | 10767.28 | 2572.50 | 33947.81 | 110470.79 | 144418.60 | 2016 |
| 240214 | Ricardo Jimenez | Transit Supervisor | 100900.50 | 8531.81 | 2838.00 | 35989.91 | 112270.31 | 148260.22 | 2017 |
| 299079 | Ricardo Jimenez | Transit Supervisor | 61286.00 | 2780.30 | 1417.50 | 22218.57 | 65483.80 | 87702.37 | 2018 |

**Plot RicardoJimenez TotalPayBenefits VS Year**

In [16]:

```
df[df['EmployeeName']=='Ricardo Jimenez'][['BasePay','Year']]
```

Out[16]:

| | BasePay | Year |
|---|---|---|
| 50596 | 72936.93 | 2012 |
| 120452 | 89128.98 | 2014 |
| 160317 | 89623.29 | 2015 |
| 198692 | 97131.01 | 2016 |
| 240214 | 100900.50 | 2017 |
| 299079 | 61286.00 | 2018 |

In [17]:

```
A=df['Year'].nunique()
```

```python
B=df['Year'].unique()
print('The information of {} years are available in the dataset:{}'.format(A,B))
```

```
The information of 8 years are available in the dataset:[2011 2012 2013 2014 2015 2016 2017 2018]
```

In [18]:

```python
df.groupby('Year').mean()['BasePay']
```

Out[18]:

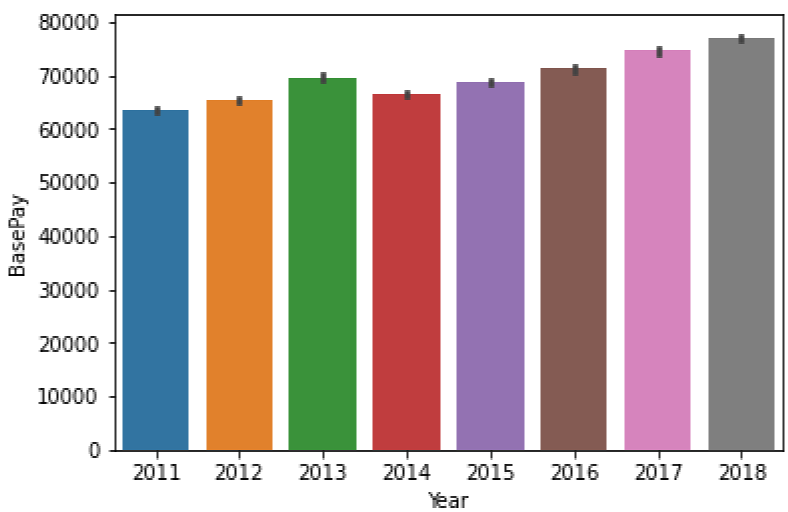
```
Year
2011    63595.956517
2012    65436.406857
2013    69630.030216
2014    66564.421924
2015    68776.293324
2016    71181.405996
2017    74570.581134
2018    76947.426822

Name: BasePay, dtype: float64
```

In [19]:

```
sns.barplot(data=df, x='Year', y='BasePay')
```

Out[19]:

<AxesSubplot:xlabel='Year', ylabel='BasePay'>



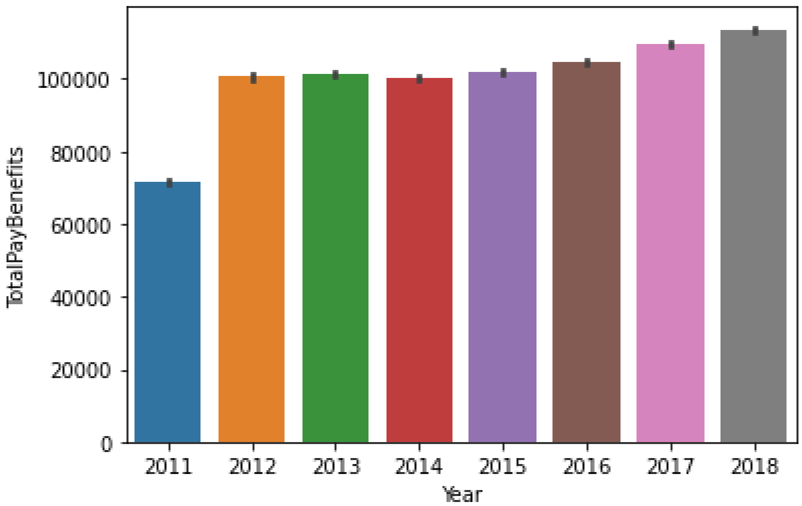

**Exercise2: Plot RicardoJimenez TotalPayBenefits VS Year**

In [20]:

```
df[df['EmployeeName']=='Ricardo
Jimenez'][['TotalPayBenefits','Year']]
sns.barplot(data=df, x='Year', y='TotalPayBenefits')
```

Out[20]:

<AxesSubplot:xlabel='Year', ylabel='TotalPayBenefits'>




**Exercise3: Which year has the maximum mean of BasePay?¶**

In [21]:

```
A=df.groupby('Year').mean()['BasePay']
```

In [22]:

```
A.max()
```

Out[22]:

```
76947.42682195794
```

In [23]:

```
df['JobTitle'].value_counts().head(5)
```

Out[23]:

```
Transit Operator    17995
Special Nurse       10857
Registered Nurse     9249
Firefighter          5891
Custodian            5759

Name: JobTitle, dtype: int64
```

In [24]:

```
df.groupby('Year').nunique()['JobTitle']
```

Out[24]:

```
Year
2011    1045
2012    1044
2013    1051
2014     996
2015    1010
2016    1009
2017    1017
2018    1000
```

```
Name: JobTitle, dtype: int64
```

In [25]:

```
df[df['Year']==2013]['JobTitle'].nunique()
```

Out[25]:

```
1051
```

In [26]:

```
sum(df[df['Year']==2013]['JobTitle'].value_counts()==1)
```

Out[26]:

```
202
```

In [27]:

```
def chief_string(title):
    if 'chief' in title.lower():
        return True
    else:
        return False
sum(df['JobTitle'].apply(lambda x:chief_string(x)))
```

Out[27]:

[Reference link]