Summary

YAZAN.Y

Reading data from the CSV file

In [2]:
```
# Reading data from the CSV file
df = pd.read_csv('Salaries.csv')
```

Data Exploration:

- Checked the shape of the dataset: The dataset has [number of rows, number of columns] as its shape.
- Checked the data types of each column.

```
]: #  Data Exploration
   print(df.shape)
   print(df.dtypes)
```

Statistics:

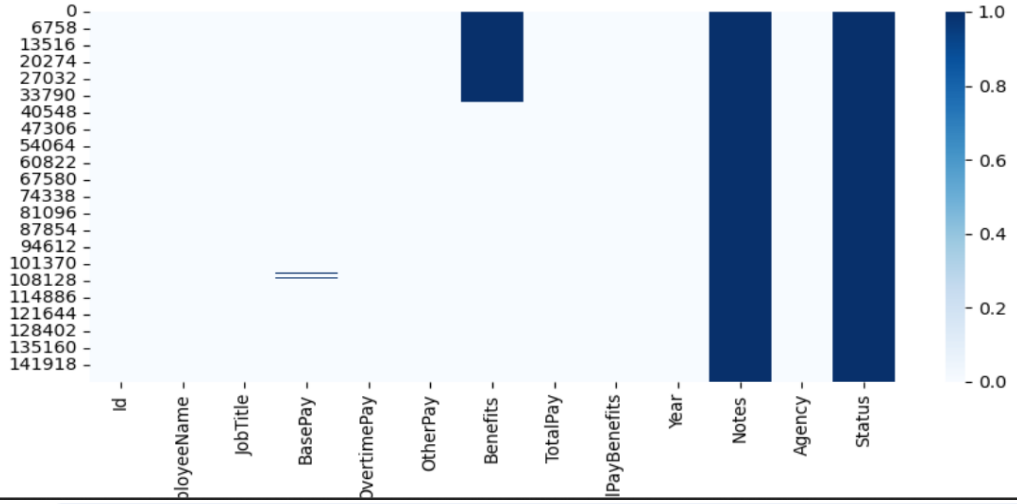
Calculated and printed basic statistics for the "TotalPay" column, including mean, median, mode, minimum, maximum, and standard deviation.

```
# Descriptive Statistics
statistics= df['TotalPay'].describe()
print(statistics)
```

missing values.

check for missing values in each column

```
plt.figure(figsize=[10,4])
sns.heatmap(df.isnull(),cmap='Blues')
```



as we see we have missing data

Handle missing data by suitable method with explain why you use it

There are several ways to handle missing data. One of the most common methods is to drop

,the column that contains the missing data. However

this is not a good decision in some cases, as it may lead to the loss of important information or to .bias in the results

Another way to handle missing data is to calculate the mean of the data that does not contain .missing values

.Then, this mean is used to replace the missing values

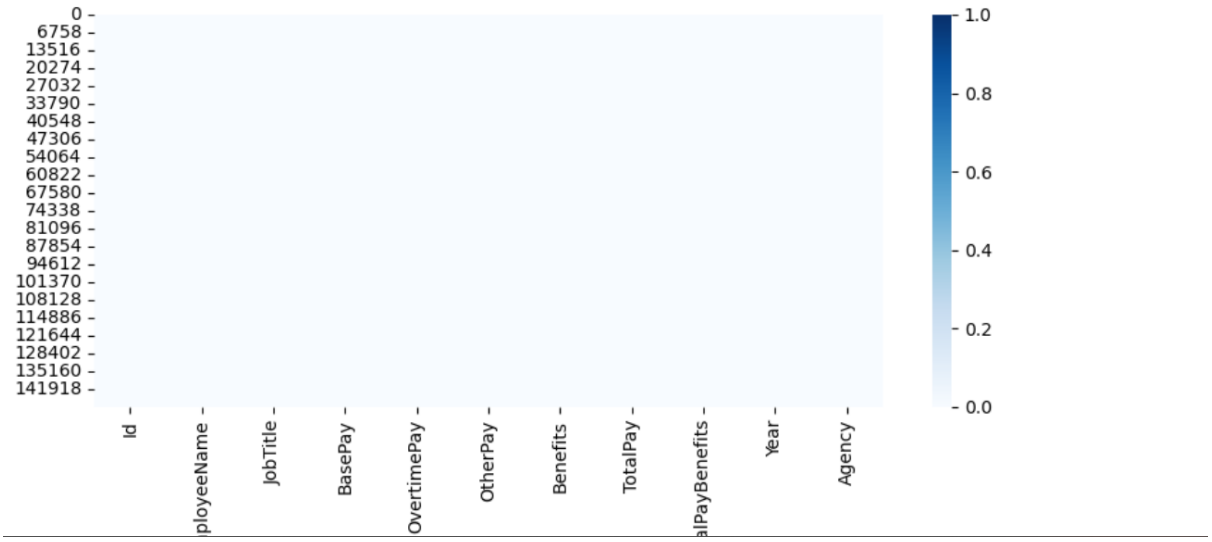
,The choice of the right method to handle missing data depends on several factors

including the percentage of missing data, the type of data, and the nature of the task being '".performed

Data Cleaning:

- **Dropped the "Notes" and "Status" columns from the dataset.**
- **Handled missing data in the "Benefits" and "BasePay" columns:**

```
df = df.drop('Notes', axis=1)
df = df.drop('Status', axis=1)
df['Benefits'] = df['Benefits'].fillna(df['Benefits'].mean())
df['BasePay'] = df['BasePay'].fillna(df['BasePay'].mean())
plt.figure(figsize=[10,4])
sns.heatmap(df.isnull(), cmap='Blues')
```

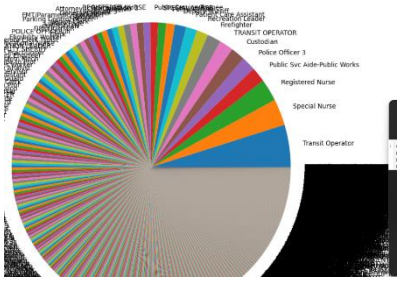


Data Visualization:

- Created a histogram to visualize the distribution of salaries in the "TotalPay" column.
- Created a pie chart to represent the proportion of employees in different departments based on the "JobTitle" column.

```
[8]: plt.figure(figsize=(10, 6))
     plt.hist(df['TotalPay'],  color='red')
     plt.title('Salary Distribution')
```


```
department_counts = df['JobTitle'].value_counts()
plt.figure(figsize=[10,10])
plt.pie(department_counts, labels=department_counts.index)
plt.show()
```



Grouped Analysis:

Grouped the data by "JobTitle" and calculated the average salary for each group.

```
: # Grouped Analysis
average = df.groupby('JobTitle')['TotalPay'].mean().sort_values(ascending=False)
print(average)
```

Correlation Analysis

- **Calculated the correlation between the "BasePay" and "TotalPay" columns.**
- **Plotted a scatter plot to visualize the relationship between "BasePay" and "TotalPay".**

```
correlation = df['BasePay'].corr(df['TotalPay'])
print(f"Correlation between BasePay and TotalPay: {correlation}")
```

```
plt.figure(figsize=(5, 3))
plt.scatter(df['BasePay'], df['TotalPay'])
plt.title('Scatter Plot: BasePay vs TotalPay')
plt.xlabel('BasePay')
plt.ylabel('TotalPay')
plt.show()
```