Data Cleaning

Because data is often taken from multiple sources which are normally not too reliable and that too in different formats, more than half our time is consumed in dealing with data quality issues when working on a machine learning problem. It is simply unrealistic to expect that the data will be perfect. There may be problems due to human error, limitations of measuring devices, or flaws in the data collection process. Some of them are as follows:-

- **Missing values:** It is very much usual to have missing values in your dataset. It may have happened during data collection, or maybe due to some data validation rule, but regardless missing values must be taken into consideration. You can eliminate rows with missing data or estimate the missing values.
- **Inconsistent values:** Data can contain inconsistent values. Most probably we have already faced this issue at some point. For instance, the Address field contains the Phone Number. It may be due to human error or maybe the information was misread while being scanned from a handwritten form. It is therefore always advised to perform data assessment like knowing what the data type of the features should be and whether it is the same for all the data objects.
- **Duplicate values:** A dataset may include data objects which are duplicates of one another. It may happen when say the same person submits a form more than once. The term deduplication is often used to refer to the process of dealing with duplicates. In most cases, the duplicates are removed so as to not give that particular data object an advantage or bias, when running machine learning algorithms.

To learn more about data cleaning, you can visit this link.

Imports

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

Read CSV File

In [2]:
```
df = pd.read_csv('Salaries.csv')
```

In [3]:
```
df.head() #shows first five rows
```

Out[3]:

	Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayB
0	1	NATHANIEL FORD	GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY	167411.18	0.00	400184.25	NaN	567595.43	567
1	2	GARY JIMENEZ	CAPTAIN III (POLICE DEPARTMENT)	155966.02	245131.88	137811.38	NaN	538909.28	538

	Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayB
2	3	ALBERT PARDINI	CAPTAIN III (POLICE DEPARTMENT)	212739.13	106088.18	16452.60	NaN	335279.91	33!
3	4	CHRISTOPHER CHONG	WIRE ROPE CABLE MAINTENANCE MECHANIC	77916.00	56120.71	198306.90	NaN	332343.61	33:
4	5	PATRICK GARDNER	DEPUTY CHIEF OF DEPARTMENT, (FIRE DEPARTMENT)	134401.60	9737.00	182234.59	NaN	326373.19	32(

In [4]:

```
df.describe()
```

Out[4]:

	Id	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPa
count	148654.000000	148045.000000	148650.000000	148650.000000	112491.000000	148654.000000	1486
mean	74327.500000	66325.448841	5066.059886	3648.767297	25007.893151	74768.321972	936
std	42912.857795	42764.635495	11454.380559	8056.601866	15402.215858	50517.005274	627
min	1.000000	-166.010000	-0.010000	-7058.590000	-33.890000	-618.130000	-6
25%	37164.250000	33588.200000	0.000000	0.000000	11535.395000	36168.995000	440
50%	74327.500000	65007.450000	0.000000	811.270000	28628.620000	71426.610000	924
75%	111490.750000	94691.050000	4658.175000	4236.065000	35566.855000	105839.135000	1328
max	148654.000000	319275.010000	245131.880000	400184.250000	96570.660000	567595.430000	5675

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 148654 entries, 0 to 148653
Data columns (total 13 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   Id                148654 non-null  int64  
 1   EmployeeName      148654 non-null  object 
 2   JobTitle          148654 non-null  object 
 3   BasePay           148045 non-null  float64
 4   OvertimePay       148650 non-null  float64
 5   OtherPay          148650 non-null  float64
 6   Benefits          112491 non-null  float64
 7   TotalPay          148654 non-null  float64
 8   TotalPayBenefits  148654 non-null  float64
 9   Year              148654 non-null  int64  
 10  Notes             0 non-null       float64
 11  Agency            148654 non-null  object 
 12  Status            0 non-null       float64
dtypes: float64(8), int64(2), object(3)
memory usage: 14.7+ MB
```

Here we can see that some of the columns contain null values.

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

Out[6]:

	Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayB
0	1	NATHANIEL FORD	GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY	167411.18	0.00	400184.25	NaN	567595.43	567
1	2	GARY JIMENEZ	CAPTAIN III (POLICE DEPARTMENT)	155966.02	245131.88	137811.38	NaN	538909.28	538
2	3	ALBERT PARDINI	CAPTAIN III (POLICE DEPARTMENT)	212739.13	106088.18	16452.60	NaN	335279.91	335
3	4	CHRISTOPHER CHONG	WIRE ROPE CABLE MAINTENANCE MECHANIC	77916.00	56120.71	198306.90	NaN	332343.61	332
4	5	PATRICK GARDNER	DEPUTY CHIEF OF DEPARTMENT,(FIRE DEPARTMENT)	134401.60	9737.00	182234.59	NaN	326373.19	326

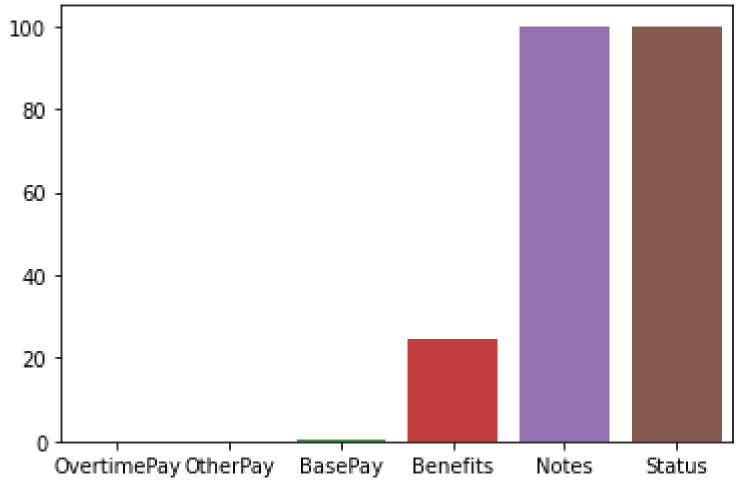
In [7]:
```
df = df.set_index('Id')
```

In [8]:
```
# How many values in each column is null
df.isnull().sum()
```

Out[8]:
```
EmployeeName             0
JobTitle                 0
BasePay                609
OvertimePay              4
OtherPay                 4
Benefits             36163
TotalPay                 0
TotalPayBenefits         0
Year                     0
Notes               148654
Agency                   0
Status              148654
dtype: int64
```

In [9]:
```
# Plot the above data
def percent_null():
    p = 100 * df.isnull().sum() / len(df) # Percentage of null values in each columns
    p = p[p > 0].sort_values() # Ignore cases where percentage is 0 and then sort
    sns.barplot(x=p.index, y=p) # Plot the data
```

In [10]:
```
percent_null()
```



We can see that the 'Notes' and 'Status' columns are completely null. So we can simply remove those columns.

In [11]:

```
df.drop(['Notes', 'Status'], axis=1, inplace=True) #dropping notes and status column
```

In [12]:

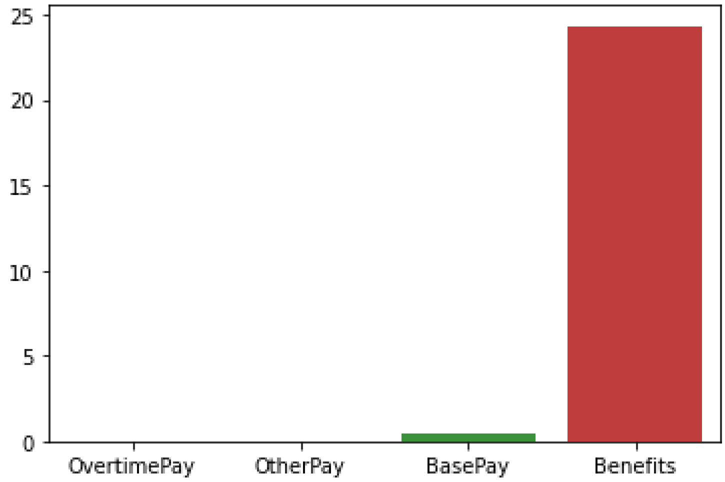
```
df.head()
```

Out[12]:

Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayBene
1	NATHANIEL FORD	GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY	167411.18	0.00	400184.25	NaN	567595.43	56759!
2	GARY JIMENEZ	CAPTAIN III (POLICE DEPARTMENT)	155966.02	245131.88	137811.38	NaN	538909.28	53890!
3	ALBERT PARDINI	CAPTAIN III (POLICE DEPARTMENT)	212739.13	106088.18	16452.60	NaN	335279.91	33527!
4	CHRISTOPHER CHONG	WIRE ROPE CABLE MAINTENANCE MECHANIC	77916.00	56120.71	198306.90	NaN	332343.61	33234:
5	PATRICK GARDNER	DEPUTY CHIEF OF DEPARTMENT,(FIRE DEPARTMENT)	134401.60	9737.00	182234.59	NaN	326373.19	32637:

In [13]:

```
percent_null()
```



Now, let's check the JobTitle and Agency columns.

In [14]:
```
df['JobTitle'].nunique() # Count of unique values in the column
```

Out[14]: 2159

In [15]:
```
df['Agency'].nunique()
```

Out[15]: 1

'EmployeeName' is unique for each employee. Moreover 'JobTitle' also contains too many unique values for One Hot Encoding. Value of 'Agency' is same for every row. So we can drop these three columns.

In [16]:
```
df.drop(['JobTitle', 'Agency', 'EmployeeName'], axis=1, inplace=True)
```

In [17]:
```
df.head()
```

Out[17]:

Id	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayBenefits	Year
1	167411.18	0.00	400184.25	NaN	567595.43	567595.43	2011
2	155966.02	245131.88	137811.38	NaN	538909.28	538909.28	2011
3	212739.13	106088.18	16452.60	NaN	335279.91	335279.91	2011
4	77916.00	56120.71	198306.90	NaN	332343.61	332343.61	2011
5	134401.60	9737.00	182234.59	NaN	326373.19	326373.19	2011

Let's explore the BasePay column

In [18]:
```
df.sort_values('BasePay') # Sorting dataset by base pay
```

Out[18]:

Id	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayBenefits	Year
72833	-166.01	249.02	0.0	6.56	83.01	89.57	2012

Id	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayBenefits	Year
72866	-121.63	182.70	0.0	5.44	61.07	66.51	2012
72873	-109.22	163.83	0.0	4.32	54.61	58.93	2012
72875	-106.60	159.90	0.0	4.66	53.30	57.96	2012
72879	-101.88	153.08	0.0	4.55	51.20	55.75	2012
...	...	...	...	...	...	...	...
110531	NaN	0.00	0.0	-33.89	0.00	-33.89	2013
148647	NaN	NaN	NaN	NaN	0.00	0.00	2014
148651	NaN	NaN	NaN	NaN	0.00	0.00	2014
148652	NaN	NaN	NaN	NaN	0.00	0.00	2014
148653	NaN	NaN	NaN	NaN	0.00	0.00	2014

148654 rows × 7 columns

Negative 'BasePay' doen't make much sense. So we can remove the rows where 'BasePay' is negative or null.

In [19]:
```
df = df[df['BasePay'] > 0] # Remove all rows where BasePay <= 0 or null
```

In [20]:
```
df.sort_values('BasePay') # Sorting dataset by base pay
```

Out[20]:

Id	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayBenefits	Year
148620	6.04	0.00	10.05	2.30	16.09	18.39	2014
36088	14.25	0.00	56.14	NaN	70.39	70.39	2011
148621	15.50	0.00	0.00	0.16	15.50	15.66	2014
110520	15.83	0.00	0.00	0.16	15.83	15.99	2013
110521	15.83	0.00	0.00	0.16	15.83	15.99	2013
...	...	...	...	...	...	...	...
72932	313312.52	0.00	0.00	82319.51	313312.52	395632.03	2013
72927	313686.01	0.00	23236.00	85431.39	336922.01	422353.40	2013
72930	315572.01	0.00	0.00	82849.66	315572.01	398421.67	2013
110533	318835.49	10712.95	60563.54	89540.23	390111.98	479652.21	2014
72926	319275.01	0.00	20007.06	86533.21	339282.07	425815.28	2013

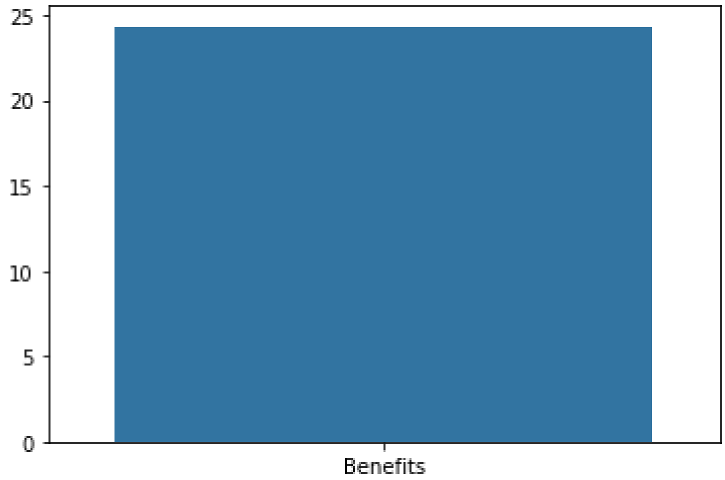
146736 rows × 7 columns

In [21]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
Int64Index: 146736 entries, 1 to 148621
Data columns (total 7 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   BasePay           146736 non-null  float64
 1   OvertimePay       146736 non-null  float64
 2   OtherPay          146736 non-null  float64
 3   Benefits          111029 non-null  float64
 4   TotalPay          146736 non-null  float64
 5   TotalPayBenefits  146736 non-null  float64
 6   Year              146736 non-null  int64  
dtypes: float64(6), int64(1)
memory usage: 9.0 MB
```

In [22]:
```
percent_null()
```



We can see that other than 'Benefits' all the missing data has been solved. We can replace missing Benefits with 0.

In [23]:
```
df[df['Benefits'].isna()].head()
```

Out[23]:

Id	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayBenefits	Year
1	167411.18	0.00	400184.25	NaN	567595.43	567595.43	2011
2	155966.02	245131.88	137811.38	NaN	538909.28	538909.28	2011
3	212739.13	106088.18	16452.60	NaN	335279.91	335279.91	2011
4	77916.00	56120.71	198306.90	NaN	332343.61	332343.61	2011
5	134401.60	9737.00	182234.59	NaN	326373.19	326373.19	2011

In [24]:
```
df['Benefits'].fillna(0, inplace=True)
# We have seen in the above cases that Benefits being null actually means 0 Benefits
```

In [25]:
```
df.head()
```

Out[25]:

Id	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayBenefits	Year
1	167411.18	0.00	400184.25	0.0	567595.43	567595.43	2011

	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayBenefits	Year
Id							
2	155966.02	245131.88	137811.38	0.0	538909.28	538909.28	2011
3	212739.13	106088.18	16452.60	0.0	335279.91	335279.91	2011
4	77916.00	56120.71	198306.90	0.0	332343.61	332343.61	2011
5	134401.60	9737.00	182234.59	0.0	326373.19	326373.19	2011

In [26]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 146736 entries, 1 to 148621
Data columns (total 7 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   BasePay           146736 non-null  float64
 1   OvertimePay       146736 non-null  float64
 2   OtherPay          146736 non-null  float64
 3   Benefits          146736 non-null  float64
 4   TotalPay          146736 non-null  float64
 5   TotalPayBenefits  146736 non-null  float64
 6   Year              146736 non-null  int64  
dtypes: float64(6), int64(1)
memory usage: 9.0 MB
```

We can remove the rows where the values don't add up properly.

In [27]:

```
condition = df['TotalPay'] == df['BasePay'] + df['OvertimePay'] + df['OtherPay']
df = df[condition]
```

In [28]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 119650 entries, 2 to 148621
Data columns (total 7 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   BasePay           119650 non-null  float64
 1   OvertimePay       119650 non-null  float64
 2   OtherPay          119650 non-null  float64
 3   Benefits          119650 non-null  float64
 4   TotalPay          119650 non-null  float64
 5   TotalPayBenefits  119650 non-null  float64
 6   Year              119650 non-null  int64  
dtypes: float64(6), int64(1)
memory usage: 7.3 MB
```

In [29]:

```
condition = df['TotalPayBenefits'] == df['TotalPay'] + df['Benefits']
df = df[condition]
```

In [30]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 101716 entries, 2 to 148621
Data columns (total 7 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   BasePay           101716 non-null  float64
```

```
 1   OvertimePay       101716 non-null  float64
 2   OtherPay          101716 non-null  float64
 3   Benefits          101716 non-null  float64
 4   TotalPay          101716 non-null  float64
 5   TotalPayBenefits  101716 non-null  float64
 6   Year              101716 non-null  int64
dtypes: float64(6), int64(1)
memory usage: 6.2 MB
```

In [31]:
```
df.head()
```

Out[31]:

Id	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayBenefits	Year
2	155966.02	245131.88	137811.38	0.0	538909.28	538909.28	2011
3	212739.13	106088.18	16452.60	0.0	335279.91	335279.91	2011
4	77916.00	56120.71	198306.90	0.0	332343.61	332343.61	2011
5	134401.60	9737.00	182234.59	0.0	326373.19	326373.19	2011
6	118602.00	8601.00	189082.74	0.0	316285.74	316285.74	2011

Now we can remove 'TotalPay' and 'TotalPayBenefits' as they are sum of other features.

In [32]:
```
df.drop(['TotalPay', 'TotalPayBenefits'], axis=1, inplace=True)
```

In [33]:
```
df.head()
```

Out[33]:

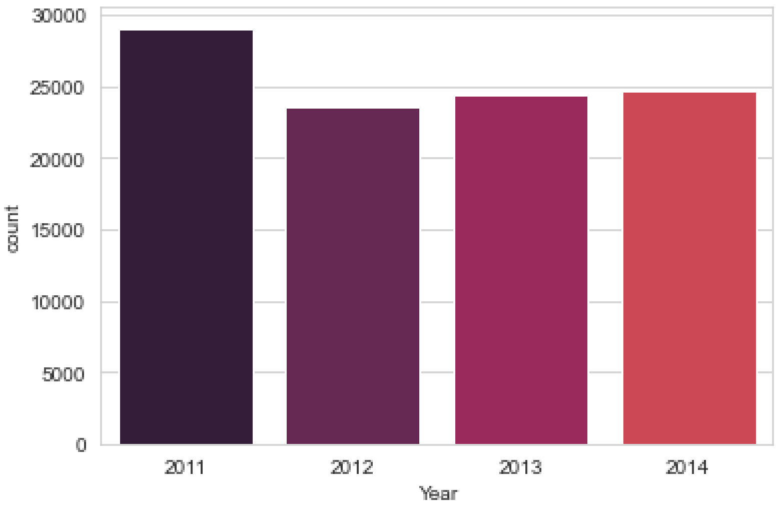
Id	BasePay	OvertimePay	OtherPay	Benefits	Year
2	155966.02	245131.88	137811.38	0.0	2011
3	212739.13	106088.18	16452.60	0.0	2011
4	77916.00	56120.71	198306.90	0.0	2011
5	134401.60	9737.00	182234.59	0.0	2011
6	118602.00	8601.00	189082.74	0.0	2011

Exploratory Data Analysis

In [34]:
```
sns.set_style('whitegrid')
sns.set_palette('rocket')
```

In [35]:
```
sns.countplot(x='Year', data=df)
```

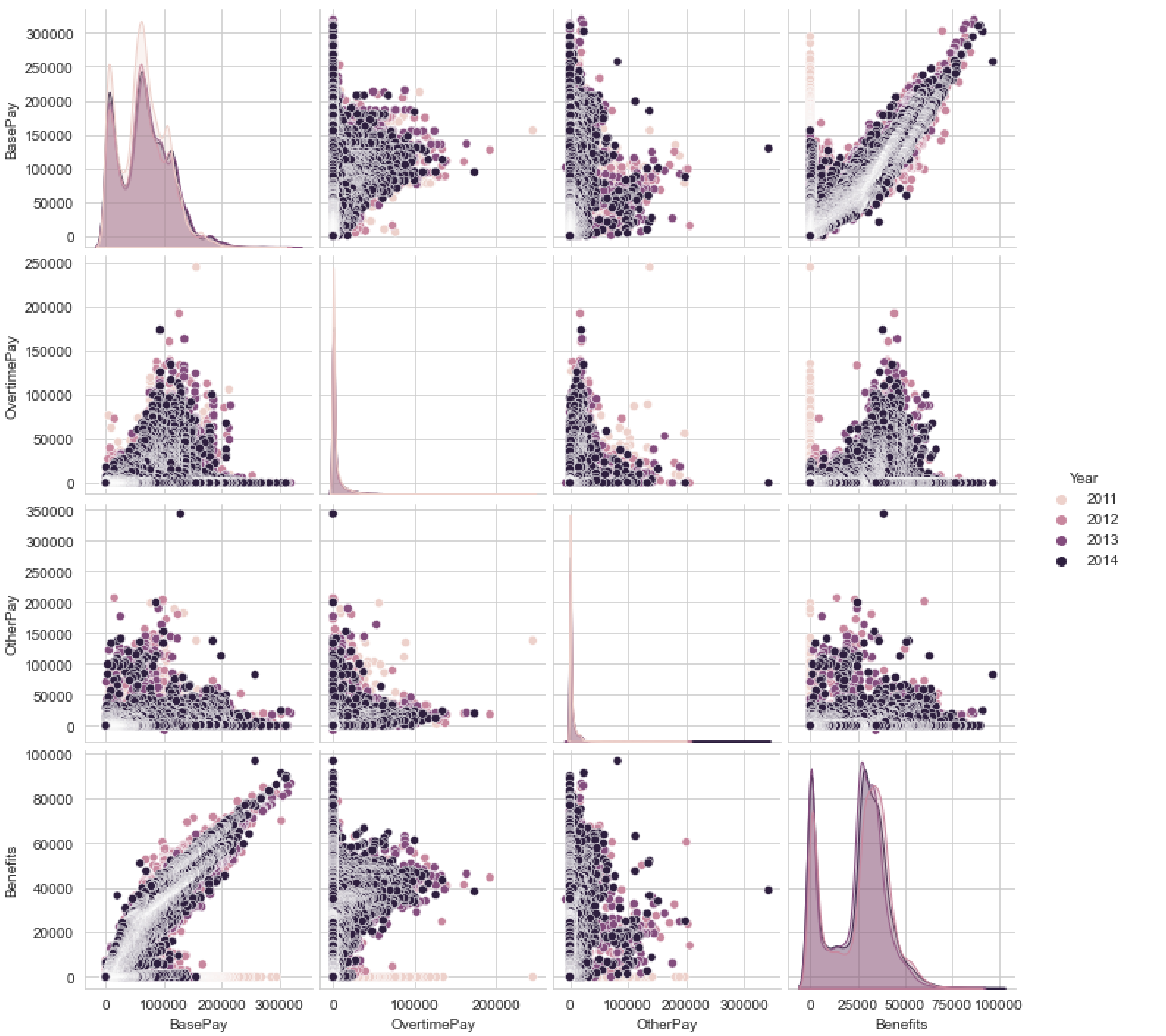
Out[35]: <AxesSubplot:xlabel='Year', ylabel='count'>



In [36]:
```
sns.pairplot(df, hue='Year')
```

```
C:\Users\msoum\anaconda3\lib\site-packages\seaborn\distributions.py:306: UserWarning: Dat
aset has 0 variance; skipping density estimate.
  warnings.warn(msg, UserWarning)
```

Out[36]: <seaborn.axisgrid.PairGrid at 0x22e4b2e7dc0>



Now, let's save the cleaned DataFrame to a csv file so that we can use it in the next file.

In [38]:
```
df.to_csv('Salaries_Cleaned.csv')
```