Salaries Exercise -

Welcome to a quick exercise for you to practice your pandas skills! Just follow along and complete the tasks outlined in bold below. The tasks will get harder and harder as you go along.

** Import pandas as pd.**

In [2]:
```
import pandas as pd
```

** Read Salaries.csv as a dataframe called sal.**

In [5]:
```
# Load the uploaded Salaries.csv file as a pandas dataframe
sal = pd.read_csv('Salaries.csv')
```

** Check the head of the DataFrame. **

In [6]:
```
# Display the first few rows of the dataframe to verify the data
sal.head()
```

Out[6]:

	Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	T
0	1	NATHANIEL FORD	GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY	167411.18	0.00	400184.25	NaN	56
1	2	GARY JIMENEZ	CAPTAIN III (POLICE DEPARTMENT)	155966.02	245131.88	137811.38	NaN	53
2	3	ALBERT PARDINI	CAPTAIN III (POLICE DEPARTMENT)	212739.13	106088.18	16452.60	NaN	33
3	4	CHRISTOPHER CHONG	WIRE ROPE CABLE MAINTENANCE MECHANIC	77916.00	56120.71	198306.90	NaN	33
4	5	PATRICK GARDNER	DEPUTY CHIEF OF DEPARTMENT, (FIRE DEPARTMENT)	134401.60	9737.00	182234.59	NaN	32

** Use the .info() method to find out how many entries there are.**

In [7]:
```
sal.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
```

What is the average BasePay ?

In [8]:
```
average_base_pay = sal['BasePay'].mean()
print(average_base_pay)
```

```
66325.4488404877
```

** What is the highest amount of OvertimePay in the dataset ? **

In [9]:
```
highest_overtime_pay = sal['OvertimePay'].max()
print(highest_overtime_pay)
```

```
245131.88
```

** What is the job title of JOSEPH DRISCOLL ? Note: Use all caps, otherwise you may get an answer that doesn't match up (there is also a lowercase Joseph Driscoll). **

In [10]:
```
job_title = sal[sal['EmployeeName'] == 'JOSEPH DRISCOLL']['JobTitle'].values[0]
print(job_title)
```

```
CAPTAIN, FIRE SUPPRESSION
```

** How much does JOSEPH DRISCOLL make (including benefits)? **

In [11]:
```
total_pay_benefits = sal[sal['EmployeeName'] == 'JOSEPH DRISCOLL']['TotalPayBene
print(total_pay_benefits)
```

```
270324.91
```

** What is the name of highest paid person (including benefits)?**

In [14]:
```
# Find the highest paid person including benefits
highest_paid = sal.loc[sal['TotalPayBenefits'].idxmax()]

highest_paid[['EmployeeName', 'TotalPayBenefits']]
```

Out[14]:
```
EmployeeName        NATHANIEL FORD
TotalPayBenefits         567595.43
Name: 0, dtype: object
```

** What is the name of lowest paid person (including benefits)? Do you notice something strange about how much he or she is paid?**

In [15]:
```
# Find the lowest paid person including benefits
lowest_paid = sal.loc[sal['TotalPayBenefits'].idxmin()]

lowest_paid[['EmployeeName', 'TotalPayBenefits']]
```

Out[15]:
```
EmployeeName        Joe Lopez
TotalPayBenefits      -618.13
Name: 148653, dtype: object
```

** What was the average (mean) BasePay of all employees per year? (2011-2014) ? **

In [16]:
```
# Calculate the average (mean) BasePay per year for the years 2011-2014
average_basepay_per_year = sal[sal['Year'].between(2011, 2014)].groupby('Year')[

average_basepay_per_year
```

Out[16]:
```
Year
2011    63595.956517
2012    65436.406857
2013    69630.030216
2014    66564.421924
Name: BasePay, dtype: float64
```

** How many unique job titles are there? **

In [17]:
```
# Calculate the number of unique job titles
unique_job_titles = sal['JobTitle'].nunique()

unique_job_titles
```

Out[17]:
```
2159
```

** What are the top 5 most common jobs? **

In [18]:
```
# Find the top 5 most common job titles
top_5_common_jobs = sal['JobTitle'].value_counts().head(5)

top_5_common_jobs
```

Out[18]:
```
JobTitle
Transit Operator                7036
Special Nurse                   4389
Registered Nurse                3736
Public Svc Aide-Public Works    2518
Police Officer 3                2421
Name: count, dtype: int64
```

** How many Job Titles were represented by only one person in 2013? (e.g. Job Titles with only one occurence in 2013?) **

In [20]:
```
# Filter the data for the year 2013 and count job titles with only one occurrenc
job_titles_2013 = sal[sal['Year'] == 2013]['JobTitle'].value_counts()
```

In [21]:
```
# Find the number of job titles represented by only one person
job_titles_one_occurrence_2013 = (job_titles_2013 == 1).sum()

job_titles_one_occurrence_2013
```

Out[21]:
```
202
```

** How many people have the word Chief in their job title? (This is pretty tricky) **

In [22]:
```
# Count the number of people who have the word "Chief" in their job title (case-
```

In [24]:
```
chief_count = sal[sal['JobTitle'].str.contains('Chief', case=False, na=False)].s
```

In [25]:
```
chief_count
```

Out[25]:
```
627
```

** Bonus: Is there a correlation between length of the Job Title string and Salary? **

In [26]:
```
# First, calculate the length of the job title string for each employee
sal['JobTitleLength'] = sal['JobTitle'].apply(len)
```

In [27]:
```
# Check the correlation between Job Title length and TotalPay
correlation = sal[['JobTitleLength', 'TotalPay']].corr().iloc[0, 1]
correlation
```

Out[27]:
```
-0.015356226699097014
```

Great Job!