IDS_EXP_4

February 19, 2024

[ ]:
```
EXP:04          Exploratory Data Analysis              URK22AI1022
DATE                                                   DINESH R
```

[ ]:
```
#Aim:
#   To demonstrate the exploratory data analysis using python for data science␣
 ↪applications

#Description:
#    Exploratory Data Analysis is a crucial step before you jump to machine␣
 ↪learning or modeling of dataOnce Exploratory Data Analysis is
#    complete and insights are drawn, its feature can be used for supervised␣
 ↪and unsupervised machine learning modeling.
```

[ ]:
```
import numpy as np
#from scipy.stats import zscore as stats
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv("Salary.csv")
```

[ ]:
```
#1.Remove the columns that has null value form data_science_salaries.
#URK22AI1022
df.dropna(axis=1,inplace=False).head()
```

[ ]:
```
        job_title experience_level employment_type work_models company_size
0   Data Engineer        Mid-level       Full-time      Remote       Medium
1   Data Engineer        Mid-level       Full-time      Remote       Medium
2  Data Scientist     Senior-level       Full-time      Remote       Medium
3  Data Scientist     Senior-level       Full-time      Remote       Medium
4    BI Developer        Mid-level       Full-time     On-site       Medium
```

[ ]:
```
#2.Remove the rows between 5000 to 8000 when they have any null value.
#URK22AI1022
s=df.iloc[5000:6001,:]
s1=s.dropna()
s1.head()
```

[ ]:
```
              job_title experience_level employment_type work_models  \
5000  Applied Scientist     Senior-level       Full-time     On-site
```

```
5001  Applied Scientist     Senior-level       Full-time     On-site
5002     Data Scientist     Senior-level       Full-time      Remote
5003     Data Scientist     Senior-level       Full-time      Remote
5004      Data Engineer     Senior-level       Full-time      Remote

      work_year employee_residence    salary salary_currency  salary_in_usd  \
5000     2023.0      United States  222200.0             USD       222200.0
5001     2023.0      United States  136000.0             USD       136000.0
5002     2023.0      United States  161000.0             USD       161000.0
5003     2023.0      United States  151000.0             USD       151000.0
5004     2023.0      United States  136994.0             USD       136994.0

     company_location company_size
5000    United States        Large
5001    United States        Large
5002    United States       Medium
5003    United States       Medium
5004    United States       Medium
```

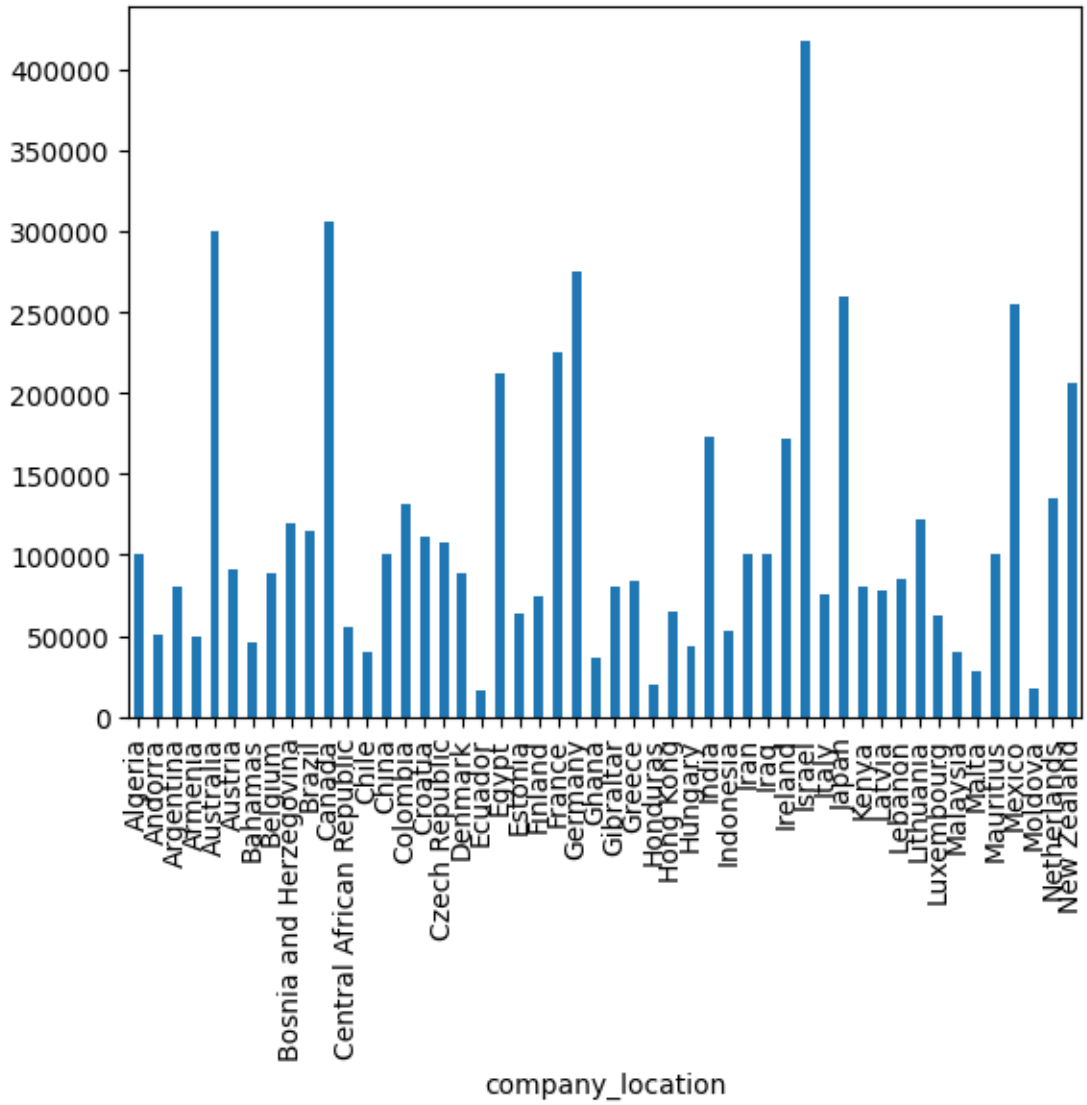
[ ]:
```
#3. Find and remove the duplicate rows.
#URK22AI1022
S1 = df.drop_duplicates()
S1.head()
```

[ ]:
```
         job_title experience_level employment_type work_models  work_year  \
0   Data Engineer        Mid-level       Full-time      Remote     2024.0
1   Data Engineer        Mid-level       Full-time      Remote     2024.0
2  Data Scientist     Senior-level       Full-time      Remote     2024.0
3  Data Scientist     Senior-level       Full-time      Remote     2024.0
4    BI Developer        Mid-level       Full-time     On-site     2024.0

  employee_residence    salary salary_currency  salary_in_usd  \
0      United States  148100.0             USD       148100.0
1      United States   98700.0             USD        98700.0
2      United States  140032.0             USD       140032.0
3      United States  100022.0             USD       100022.0
4      United States  120000.0             USD       120000.0

  company_location company_size
0    United States       Medium
1    United States       Medium
2    United States       Medium
3    United States       Medium
4    United States       Medium
```

[ ]:
```
#4.Draw the bar chart for the max 'salary_in_usd' of each country_location to␣
 ↪detect the top paid country.
```

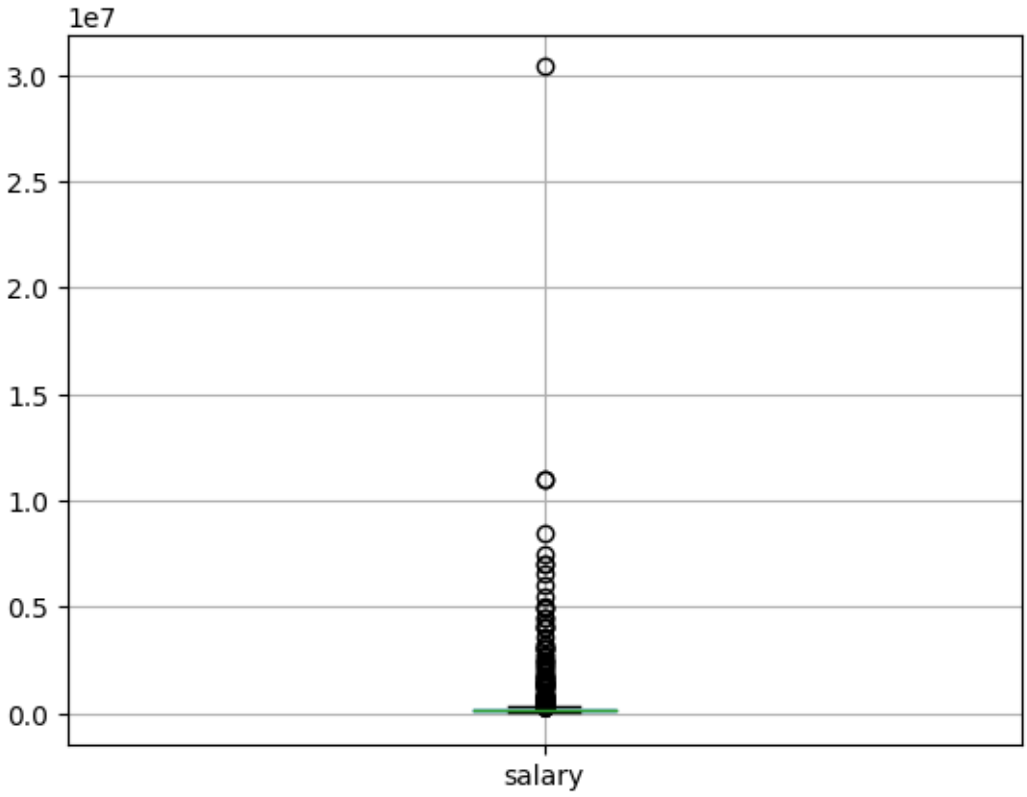
```
#URK22AI1022

s = df.groupby('company_location')['salary_in_usd'].max().head(50)
s.plot.bar()
```

[ ]: <Axes: xlabel='company_location'>



[ ]:
```
#5. Find the outliers in 'salary' column of the data_science_salaries.
#URK22AI1022

df[['salary']].boxplot()
```

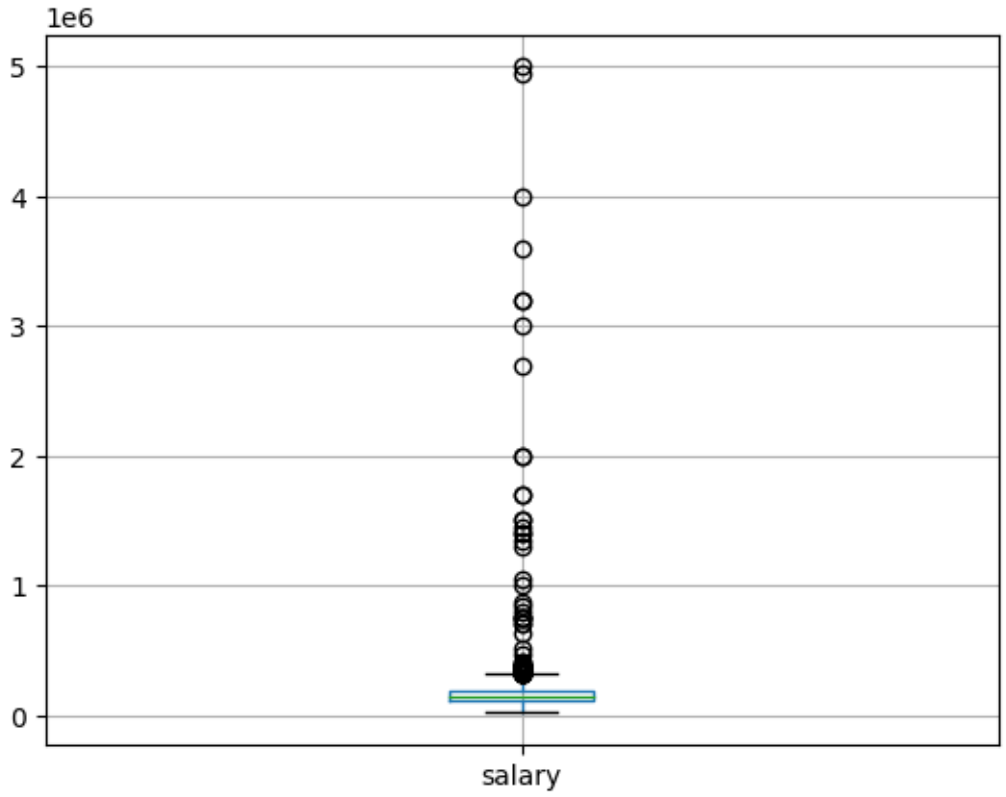
[ ]: <Axes: >



[ ]:
```
#6.Calculate the IQR using quantile and remove the outliers in work_year column␣
 ↪using IQR
#URK22AI1022


Q1 = df['work_year'].quantile(0.25)
Q3 = df['work_year'].quantile(0.75)
IQR = Q3 - Q1

df = df[~((df['work_year'] < (Q1 - 1.5 * IQR)) | (df['work_year'] > (Q3 + 1.5 *␣
 ↪IQR)))]
```

[ ]:
```
df[['salary']].boxplot()
```

[ ]:
```
<Axes: >
```



[ ]:
```
#7.Calculate the z-score of the work_year column and remove the outlier. Verify␣
 ↪using box plot
#URK22AI1022

from scipy.stats import zscore

df['work_year_zscore'] = zscore(df['work_year'])
df = df[(df['work_year_zscore'] > -3) & (df['work_year_zscore'] < 3)]
print(df.head().boxplot())
```

```
Axes(0.125,0.11;0.775x0.77)
```

[ ]:
```
#8.Insert a new_salary column by convertin usd to inr with 40% decrease.
#URK22AI1022
df['new_salary'] = df['salary_in_usd'] * 83.03
df.new_salary
```

[ ]:
```
0        12296743.00
1         8195061.00
2        11626856.96
3         8304826.66
4         9963600.00
              …
11082     3715841.59
11083     1245450.00
11084     9723145.12
11085     6155013.90
11086     7575408.11
Name: new_salary, Length: 11087, dtype: float64
```

[ ]:
```
#9.Rename the column 'work_model' into 'Work_mode'.
#URK22AI1022

df.rename(columns={'work_models': 'Work_mode'}, inplace=False)
```

[ ]:
```
Empty DataFrame
Columns: [job_title, experience_level, employment_type, Work_mode, work_year,
employee_residence, salary, salary_currency, salary_in_usd, company_location,
company_size, work_year_zscore]
Index: []
```

[ ]:
```
#10. Remove the column with index 0,1,6,7,8,9.
#URK22AI1022
p=[0, 1, 6, 7, 8, 9]
d = df.drop(p, inplace=False)
d.head()
```

[ ]:
```
                         job_title experience_level employment_type  \
2                   Data Scientist     Senior-level       Full-time
3                   Data Scientist     Senior-level       Full-time
4                     BI Developer        Mid-level       Full-time
5                     BI Developer        Mid-level       Full-time
10  Business Intelligence Developer        Mid-level       Full-time

   work_models  work_year employee_residence    salary salary_currency  \
2       Remote     2024.0      United States  140032.0             USD
3       Remote     2024.0      United States  100022.0             USD
4      On-site     2024.0      United States  120000.0             USD
5      On-site     2024.0      United States   62100.0             USD
10     On-site     2024.0      United States   87800.0             USD

    salary_in_usd company_location company_size
2        140032.0    United States       Medium
3        100022.0    United States       Medium
4        120000.0    United States       Medium
5         62100.0    United States       Medium
10        87800.0    United States       Medium
```

[ ]:
```
#11. Display 20 rows with missing values in the company_location column and␣
 ↪drop the missing values.
#URK22AI1022

print(df[df['company_location'].isnull()].head(20))
df = df.dropna(subset=['company_location'])
print(df['company_location'].isnull().head(20))
```

```
Empty DataFrame
Columns: [job_title, experience_level, employment_type, work_models, work_year,
employee_residence, salary, salary_currency, salary_in_usd, company_location,
company_size]
Index: []
0     False
1     False
```

```
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11    False
12    False
13    False
14    False
15    False
16    False
17    False
18    False
19    False
Name: company_location, dtype: bool
```

[ ]:
```
#12. Identify the missing values in the all columns and perform the following␣
 ↪operations.
#URK22AI1022

#a) Fill the missing values with '0'
df.fillna(0).head(1)
```

[ ]:
```
        job_title experience_level employment_type work_models  work_year  \
0  Data Engineer        Mid-level       Full-time      Remote     2024.0

  employee_residence    salary salary_currency  salary_in_usd  \
0      United States  148100.0             USD       148100.0

  company_location company_size
0    United States       Medium
```

[ ]:
```
#b) Fill the missing values with mean value
df.fillna(df.mean()).head(1)
```

```
<ipython-input-69-2a31522dd3f6>:2: FutureWarning: The default value of
numeric_only in DataFrame.mean is deprecated. In a future version, it will
default to False. In addition, specifying 'numeric_only=None' is deprecated.
Select only valid columns or specify the value of numeric_only to silence this
warning.
  df.fillna(df.mean()).head(1)
```

[ ]:
```
        job_title experience_level employment_type work_models  work_year  \
0  Data Engineer        Mid-level       Full-time      Remote     2024.0
```

```
  employee_residence    salary salary_currency  salary_in_usd  \
0      United States  148100.0             USD       148100.0

  company_location company_size
0    United States       Medium
```

[ ]:
```
#c) Fill the missing values with median value
df.fillna(df.median()).head(1)
```

```
<ipython-input-73-1e7ec01eab21>:2: FutureWarning: The default value of
numeric_only in DataFrame.median is deprecated. In a future version, it will
default to False. In addition, specifying 'numeric_only=None' is deprecated.
Select only valid columns or specify the value of numeric_only to silence this
warning.
  df.fillna(df.median()).head(1)
```

[ ]:
```
        job_title experience_level employment_type work_models  work_year  \
0  Data Engineer        Mid-level       Full-time      Remote     2024.0

  employee_residence    salary salary_currency  salary_in_usd  \
0      United States  148100.0             USD       148100.0

  company_location company_size
0    United States       Medium
```

[ ]:
```
#d) Fill the missing values with previous value
df.fillna(method='ffill').head(1)
```

[ ]:
```
        job_title experience_level employment_type work_models  work_year  \
0  Data Engineer        Mid-level       Full-time      Remote     2024.0

  employee_residence    salary salary_currency  salary_in_usd  \
0      United States  148100.0             USD       148100.0

  company_location company_size
0    United States       Medium
```

[ ]:
```
#e) Fill the missing values with next value
df.fillna(method='bfill').head(1)
```

[ ]:
```
        job_title experience_level employment_type work_models  work_year  \
0  Data Engineer        Mid-level       Full-time      Remote     2024.0

  employee_residence    salary salary_currency  salary_in_usd  \
0      United States  148100.0             USD       148100.0

  company_location company_size
```

```
0    United States       Medium
```

[ ]:
```
#f) Fill the missing values with linear interpolation
df.interpolate(method='linear').head(1)
```

[ ]:
```
       job_title experience_level employment_type work_models  work_year  \
0  Data Engineer        Mid-level       Full-time      Remote     2024.0

  employee_residence    salary salary_currency  salary_in_usd  \
0      United States  148100.0             USD       148100.0

  company_location company_size
0    United States       Medium
```

[ ]:
```
#13. Plot the heatmap using the correlation for employee table and titanic␣
 ↪dataset.
#URK22AI1022
import seaborn as sns

correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.show()
```

```
<ipython-input-55-178f3b0e3342>:5: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  correlation_matrix = df.corr()
```

[ ]:
```
#14. Calculate the mean, median, std deviation, variance of given dataset's␣
 ↪quantitative data.
#URK22AI1022

print(df.mean())
print(df.median())
print(df.std())
print(df.var())
```

```
work_year          2022.851296
salary           169156.023489
salary_in_usd    149656.146456
dtype: float64
work_year          2023.0
salary           142200.0
salary_in_usd    142000.0
dtype: float64
work_year             0.562761
salary           407254.591862
salary_in_usd     66689.553037
dtype: float64
work_year        3.166998e-01
```

```
salary           1.658563e+11
salary_in_usd    4.447496e+09
dtype: float64
```

```
<ipython-input-56-d31b2e25d014>:4: FutureWarning: The default value of
numeric_only in DataFrame.mean is deprecated. In a future version, it will
default to False. In addition, specifying 'numeric_only=None' is deprecated.
Select only valid columns or specify the value of numeric_only to silence this
warning.
  print(df.mean())
<ipython-input-56-d31b2e25d014>:5: FutureWarning: The default value of
numeric_only in DataFrame.median is deprecated. In a future version, it will
default to False. In addition, specifying 'numeric_only=None' is deprecated.
Select only valid columns or specify the value of numeric_only to silence this
warning.
  print(df.median())
<ipython-input-56-d31b2e25d014>:6: FutureWarning: The default value of
numeric_only in DataFrame.std is deprecated. In a future version, it will
default to False. In addition, specifying 'numeric_only=None' is deprecated.
Select only valid columns or specify the value of numeric_only to silence this
warning.
  print(df.std())
<ipython-input-56-d31b2e25d014>:7: FutureWarning: The default value of
numeric_only in DataFrame.var is deprecated. In a future version, it will
default to False. In addition, specifying 'numeric_only=None' is deprecated.
Select only valid columns or specify the value of numeric_only to silence this
warning.
  print(df.var())
```

[ ]:
```
#RESULT:
#    The exploratory data analysis using python for data science applications␣
 ↪were demonstrated.
```