

Data Analysis and Visualization

REPORT DEVELOPER ANALYSIS

Made By-
Akshita Gupta

SEMESTER-V

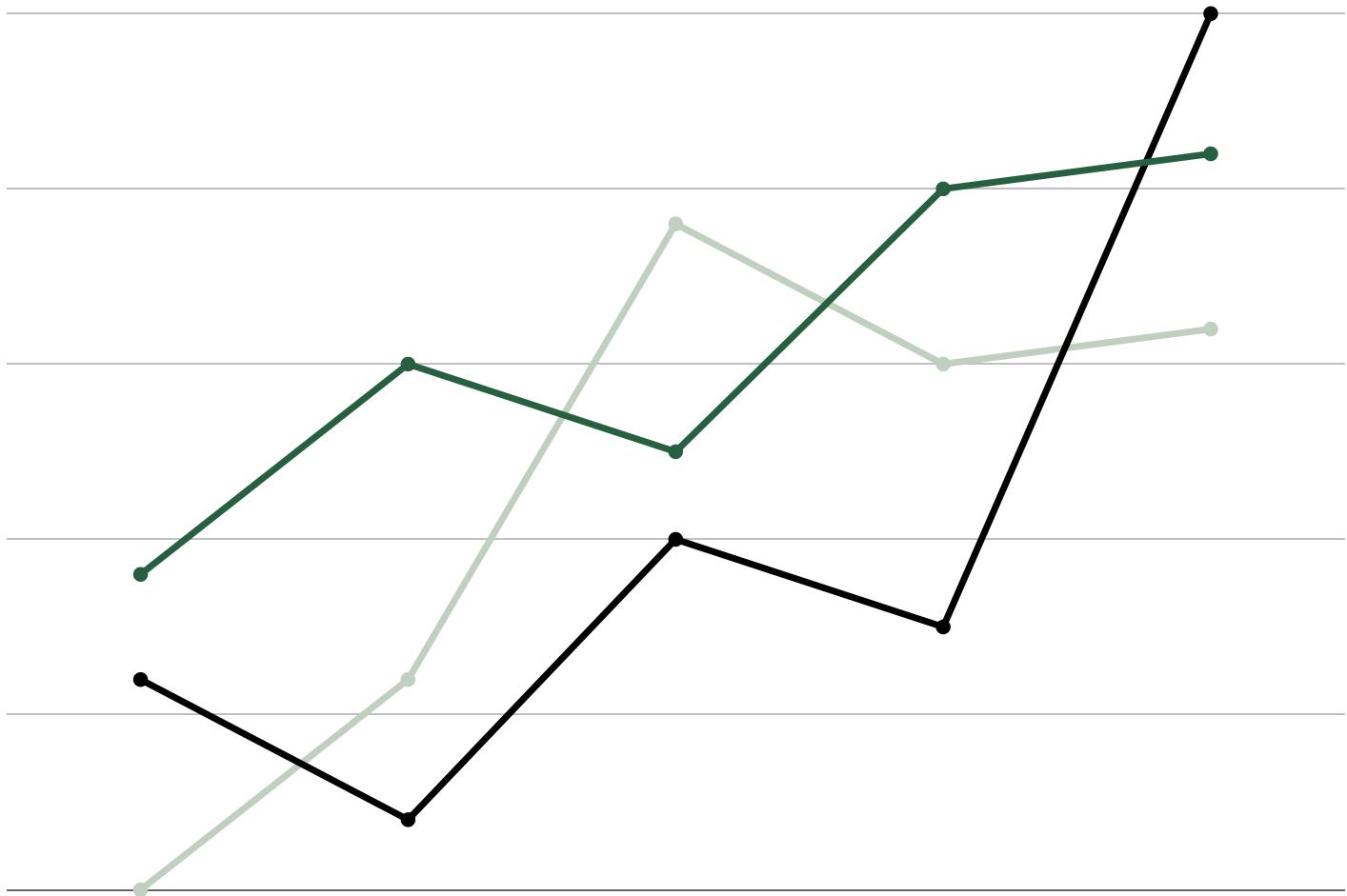
INDEX

Acknowledgement.....	03
Preface.....	04
Overview.....	05
Problem Statement & Vision.....	06
Block Diagram.....	06
Insights.....	07
Columns.....	08
Glance at the dataset.....	09
Things to analyze.....	10
Importing Libraries.....	11
ANALYSIS.....	12
CONCLUSIONS.....	63
FUTURE WORK.....	64
Bibliography.....	65

ACKNOWLEDGEMENT

We would like to express our special thanks to **Dr. Roli Bansal** for their able guidance and support in completing our Project titled- Data Analysis on Developers.

We would also like to extend our gratitude to our College Principal **Dr. Madhu Pruthi** wholeheartedly, for providing us the facility that was required.



Preface

This report has been prepared as part our Data Analysis and Visualization project, part of BSc. Computer Science Hons Semester V. This report is prepared with the view to include all the details regarding the project that we have carried out.

The initial portion is the overview of the Project. The dataset taken in consideration is about the Developers on Stackoverflow. This report includes the analysis of different columns. Analysis includes , 'How Years Coded Job affects Developers' Salary, Job Satisfaction, Career Satisfaction etc. Also it includes which language is the most popular one worldwide as well as in India.

In this project, We have plotted different types of graphs using multiple libraries in Python for ease of understanding. Thus, as you go ahead, the report will reveal minute details of the work that we have done in our project.

Overview

The Developer Analysis Dataset originally named as "Stack Overflow Survey" has been taken from Kaggle. The dataset informs us about developers' qualifications, the country in which they reside, their specialization, in their satisfaction in careers and jobs and salaries in their respective fields. The dataset also describes the developers' favorite databases, programming languages, and IDE.

This dataset gives us insight into whether the country they belong to plays a significant role in their satisfaction of jobs and opportunities. The dataset informs us about how important are skills such as adapting to newer technology, problem-solving and proper education for propelling into a career of a developer.



Dimensions

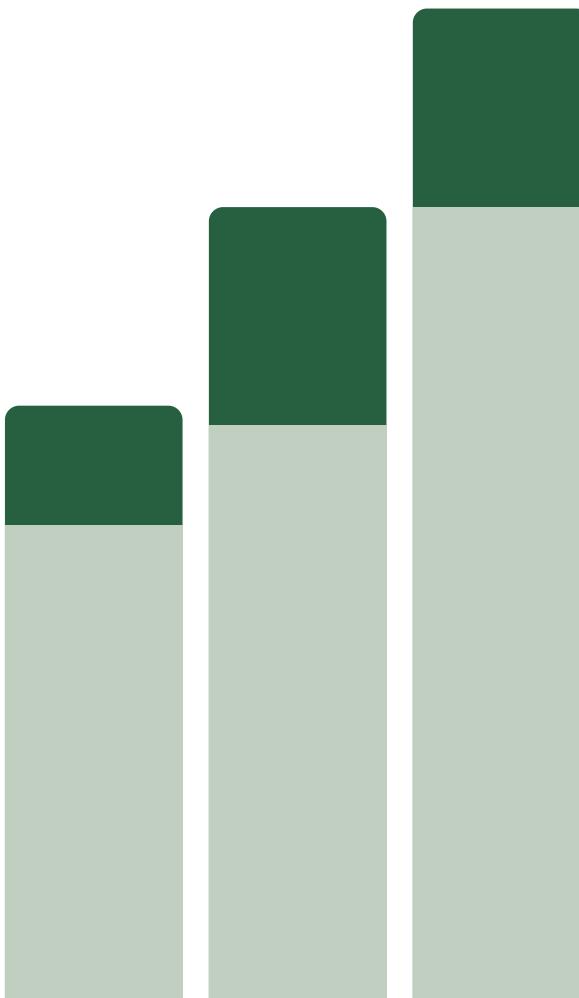
51392 Rows X 54 Columns

Problem statement

To find the average salaries for developers, based on different programming languages, databases etc. that they have studied and form a relationship between their qualifications, specializations and current jobs with respect to their current salaries

vision

To create a tool for people to help them find the most relatable programming language based on the languages already studied by them



Making a New Dataframe from the useful columns

Data Cleaning Process (deletion,imputation, customlization)

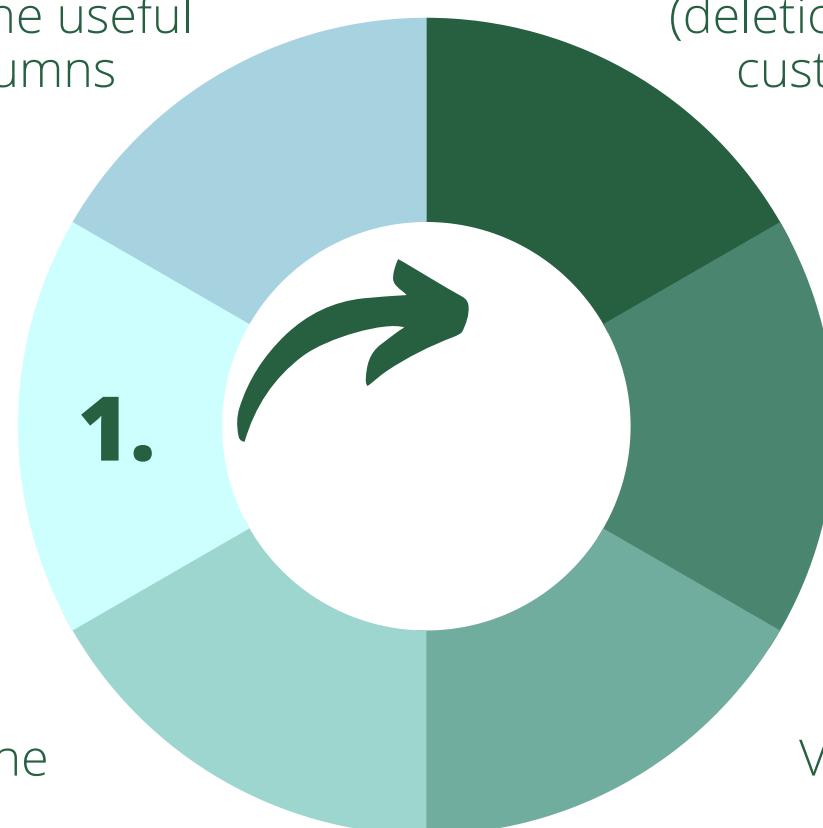
Taking into account the columns to be analyzed

1.

Performing analysis on and within columns

Concluding the analysis

Visualizing the output



Insights.

Taken From	Kaggle
Language	Python
Software	Jupyter Notebook
Libraries	<ol style="list-style-type: none">1. Numpy2. Pandas3. Matplotlib4. Seaborn5. Plotly

Columns

Respondent	LastNewJob	
Professional	Assess_Scoring	
ProgramHobby	ImportantBenefits	
Country	JobProfile	
University	LearnedHiring	
EmploymentStatus	Overpaid	
FormalEducation	EducationImportant	
MajorUndergrad	EducationTypes	
YearsCodedJob	WorkStart	
DeveloperType	HaveWorkedLanguage	
NonDeveloperType	WantWorkLanguage	
CareerSatisfaction	HaveWorkedDatabase	
JobSatisfaction	WantWorkDatabase	
ProblemSolving	HaveWorkedPlatform	
LearningNewTech	WantWorkPlatform	
JobSecurity	IDE	
DiversityImportant	Gender	
SeriousWork	Auditory Environment	
WorkPayCare	HighestEducationParents	
ChallengeMyself	Salary	
	ExpectedSalary	
		AssessJobIndustry
		AssessJobRole
		AssessJobExp
		AssessJobDept
		AssessJobTech
		AssessJobProjects
		AssessJobCompensation
		AssessJobOffice
		AssessJobCommute
		AssessJobRemote
		AssessJobProfDevel
		AssessJobDiversity
		AssessJobProduct
		AssessJobFinances

A Glance at the Dataset

Demographic Information										
Respondent	Professional	ProgramHobby	Country	University	EmploymentStatus	FormalEducation	MajorUndergrad	YearsCodedJob	DeveloperType	NonDeveloperType
	1	Student	Yes, both	United States	No	Not employed, and not looking for work	Secondary school	NaN	NaN	NaN
	2	Student	Yes, both	United Kingdom	Yes, full-time	Employed part-time	Some college/university study without earning ...	Computer science or software engineering	NaN	NaN
	3	Professional developer	Yes, both	United Kingdom	No	Employed full-time	Bachelor's degree	Computer science or software engineering	20 or more years	Other
Work Satisfaction and Job Satisfaction										
CareerSatisfaction	JobSatisfaction	ProblemSolving	LearningNewTech	JobSecurity	DiversityImportant	SeriousWork	WorkPayCare	ChallengeMyself	LastNewJob	
NaN	NaN	Strongly agree	Agree	Strongly agree	Agree	Strongly agree	Strongly disagree	Agree	Not applicable/never	
NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
8.0	9.0	Strongly agree	Strongly agree	Agree	Strongly agree	Agree	Disagree	Agree	Agree	NaN
Skills and Education										
ImportantBenefits	JobProfile	LearnedHiring	Overpaid	EducationImportant	EducationTypes	WorkStart	HaveWorkedLanguage	WantWorkLanguage	HaveWorkedDatabase	
Stock options; Vacation/days off; Remote options	Other	NaN	NaN	NaN	Online course; Open source contributions	6:00 AM	Swift	Swift	Swift	NaN
NaN	Other	Some other way	NaN	NaN	Online course; Self-taught; Hackathon; Open so...	10:00 AM	JavaScript; Python; Ruby; SQL	Java; Python; Ruby; SQL	MySQL; SQLite	
NaN	NaN	Neither underpaid nor overpaid	NaN	Not very important	Self-taught; Coding competition; Hackathon; Op...	9:00 AM	Java; PHP; Python	C; Python; Rust	MySQL	
Work Environment and Preferences										
WantWorkDatabase	HaveWorkedPlatform	WantWorkPlatform	IDE	AuditoryEnvironment	Gender	HighestEducationParents	Salary	ExpectedSalary	Access_Score	
NaN	iOS	iOS	Atom; Xcode	Turn on some music	Male	High school	NaN	NaN	30.	
MySQL; SQLite	Amazon Web Services (AWS)	Linux Desktop; Raspberry Pi; Amazon Web Servic...	Atom; Notepad++; Vim; PyCharm; RubyMine; Visua...	Put on some ambient sounds (e.g. whale songs, ...)	Male	A master's degree	NaN	37500.0	0.	
NaN	NaN	NaN	Sublime Text; Vim; IntelliJ	Turn on some music	Male	A professional degree	113750.000000	NaN	NaN	0.

Things to Analyze :

1. Different kinds of developers that exist
2. Developer Count from Different Countries
3. Developer Gender Ratio
4. Developers' Career Satisfaction vs Salary
5. Preferred Programming Language
6. Preferred Database
7. Preferred Platform to work on
8. Time to Start Work in different Countries
9. Auditory Environment's role in coding
10. Work Pay Care in Developers
11. Developers' satisfaction of Job

and much more....

Importing Libraries

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

```
import chart_studio.plotly as py
from plotly.offline import download_plotlyjs, init_notebook_mode, plot ,iplot
init_notebook_mode(connected=True)
import plotly.graph_objs as go
```

PANDAS

Pandas is built on top of two core Python libraries—matplotlib for data visualization and NumPy for mathematical operations.

- This library offers data manipulation and data operations for numerical tables and time series.
- Easily handles missing data
- It uses Series for one-dimensional data structure and Data Frame for multi-dimensional data structure
- It provides an efficient way to slice the data
- It provides a flexible way to merge, concatenate or reshape the data
- It includes a powerful time series tool to work with

NUMPY

- NumPy is a Python library used to perform mathematical and statistical operations with arrays.
- It also has functions for working in domain of linear algebra, Fourier transform, and matrices like isnan() , arange() , reshape() ,and many more...

MATPLOTLIB

- Explicitly create figures and axes, and call methods on them .
- Rely on pyplot to automatically create and manage the figures and axes, and use pyplot functions for plotting.
- Plots like Bar Graph, Histogram, Scatterplot, Area plot, Pie plot etc. are used.

SEABORN

- It is used for data visualization and exploratory data analysis.
- Seaborn works easily with dataframes and the Pandas library. The graphs created can also be customized easily.
- Plots like Heatmap , Histogram , Bar Plot , Factor Plot , Density Plots,etc are used.

PLOTLY

- It s a graphing library designed especially for data scientists, engineers, and programmers to easily visualize trends in their data using Python, R, Matlab, or Javascript.
- It enables Python users to create beautiful interactive web-based visualizations that can be displayed in Jupyter notebooks, saved to standalone HTML files, or served as part of pure Python-built web applications using Dash.

Salary vs Years Coded

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

In [2]: df = pd.read_csv("Analysis_file.csv", low_memory = False, index_col='Respondent')
df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 51392 entries, 1 to 51392
Data columns (total 53 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
0   Professional     51392 non-null    object  
1   ProgramHobby     51392 non-null    object  
2   Country          51392 non-null    object  
3   University       51392 non-null    object  
4   EmploymentStatus 51392 non-null    object  
5   FormalEducation  51392 non-null    object  
6   MajorUndergrad   42841 non-null    object  
7   YearsCodedJob   40891 non-null    object  
8   DeveloperType    36125 non-null    object  
9   NonDeveloperType 4891 non-null    object  
10  CareerSatisfaction 42695 non-null    float64 
11  JobSatisfaction  40376 non-null    float64 
12  ProblemSolving  31293 non-null    object  
13  LearningNewTech 31304 non-null    object  
14  JobSecurity      31240 non-null    object  
15  DiversityImportant 30903 non-null    object  
16  SeriousWork      31014 non-null    object  
17  WorkPayCare      30965 non-null    object  
18  ChallengeMyself  30971 non-null    object  
19  LastNewJob       32710 non-null    object  
20  AccessToIndustry  22130 non-null    object
```

Imputing the Salary Column

```
In [4]: df['Salary'].count()

Out[4]: 12891

In [5]: df['YearsCodedJob'].dropna().unique()

Out[5]: array(['20 or more years', '9 to 10 years', '10 to 11 years',
   '8 to 9 years', '7 to 8 years', '11 to 12 years', '17 to 18 years',
   '15 to 16 years', '1 to 2 years', '3 to 4 years', '12 to 13 years',
   'Less than a year', '5 to 6 years', '16 to 17 years',
   '13 to 14 years', '6 to 7 years', '2 to 3 years', '4 to 5 years',
   '14 to 15 years', '18 to 19 years', '19 to 20 years'], dtype=object)
```

```
In [6]: avg_salary = dict()
for x in df['YearsCodedJob'].dropna().unique():
    avg_salary[x] = df[df['YearsCodedJob']==x]['Salary'].mean()
```

```
In [7]: avg_salary

Out[7]: {'20 or more years': 98128.09747715948,
 '9 to 10 years': 63120.608545548384,
 '10 to 11 years': 67455.78263092662,
 '8 to 9 years': 66454.53546474078,
 '7 to 8 years': 58952.900695846074,
 '11 to 12 years': 71432.17525788138,
 '17 to 18 years': 87996.92497548716,
 '15 to 16 years': 79883.65565306017,
 '1 to 2 years': 34152.737761066004,
 '3 to 4 years': 44545.60036265642,
 '12 to 13 years': 82845.42494629917,
 'Less than a year': 33298.75204458008,
 '5 to 6 years': 52440.94312624933,
 '16 to 17 years': 77528.61148028531,
 '13 to 14 years': 75496.373677403,
 '6 to 7 years': 57454.15413633074}
```

```
In [8]: def impute_Salary(cols):
    Salary = cols[0]
    ycj = cols[1]

    if(pd.isnull(Salary)):
        if(pd.notnull(ycj)):
            return avg_salary[ycj]
    return Salary

In [9]: df['Salary'] = df[['Salary','YearsCodedJob']].apply(impute_Salary, axis=1)

In [10]: df['Salary'].count()

Out[10]: 40948

In [11]: df[['YearsCodedJob','Salary']].head(10)

Out[11]:
   YearsCodedJob      Salary
Respondent
1           NaN       NaN
2           NaN       NaN
3  20 or more years  113750.000000
4     9 to 10 years   63120.608546
5    10 to 11 years   67455.782631
6           NaN       NaN
7     8 to 9 years    66454.535465
8     7 to 8 years    58952.900696
9     7 to 8 years    58952.900696
```

Imputing 'YearsCodedJob' Column

```
In [12]: df['YearsCodedJob'].unique()

Out[12]: array([nan, '20 or more years', '9 to 10 years', '10 to 11 years',
   '8 to 9 years', '7 to 8 years', '11 to 12 years', '17 to 18 years',
   '15 to 16 years', '1 to 2 years', '3 to 4 years', '12 to 13 years',
   'Less than a year', '5 to 6 years', '16 to 17 years',
   '13 to 14 years', '6 to 7 years', '2 to 3 years', '4 to 5 years',
   '14 to 15 years', '18 to 19 years', '19 to 20 years'], dtype=object)

In [13]: def decrease_ycj_cat(x):
    if(x == '1 to 2 years' or x == '2 to 3 years' or x == '3 to 4 years' or x == '4 to 5 years'):
        return '1 to 5 years'

    elif(x == '5 to 6 years' or x == '6 to 7 years' or x == '7 to 8 years' or x == '8 to 9 years' or x == '9 to 10 years'):
        return '5 to 10 years'

    elif(x == '10 to 11 years' or x == '11 to 12 years' or x == '12 to 13 years' or x == '13 to 14 years' or x == '14 to 15 years'):
        return '10 to 15 years'

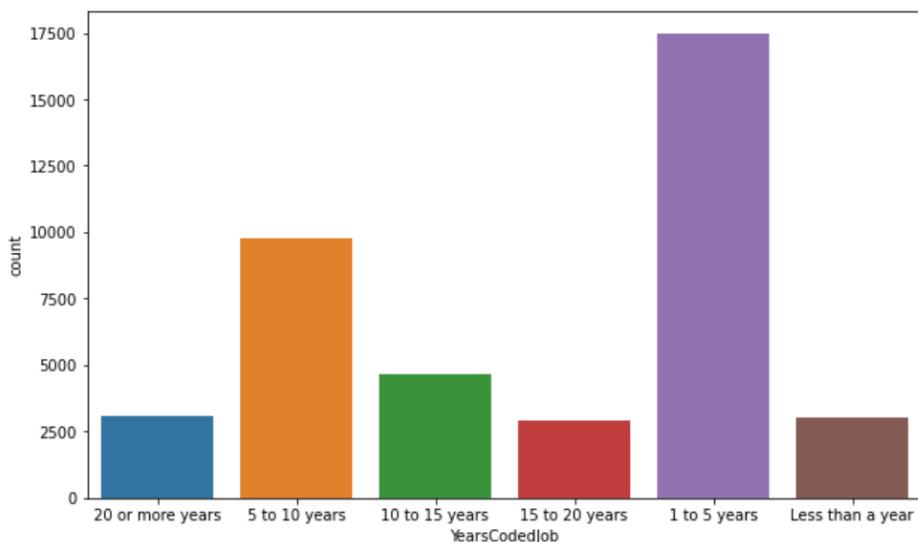
    elif(x == '15 to 16 years' or x == '16 to 17 years' or x == '17 to 18 years' or x == '18 to 19 years' or x == '19 to 20 years'):
        return '15 to 20 years'

    else:
        return x

In [14]: df["YearsCodedJob"] = df["YearsCodedJob"].apply(decrease_ycj_cat)

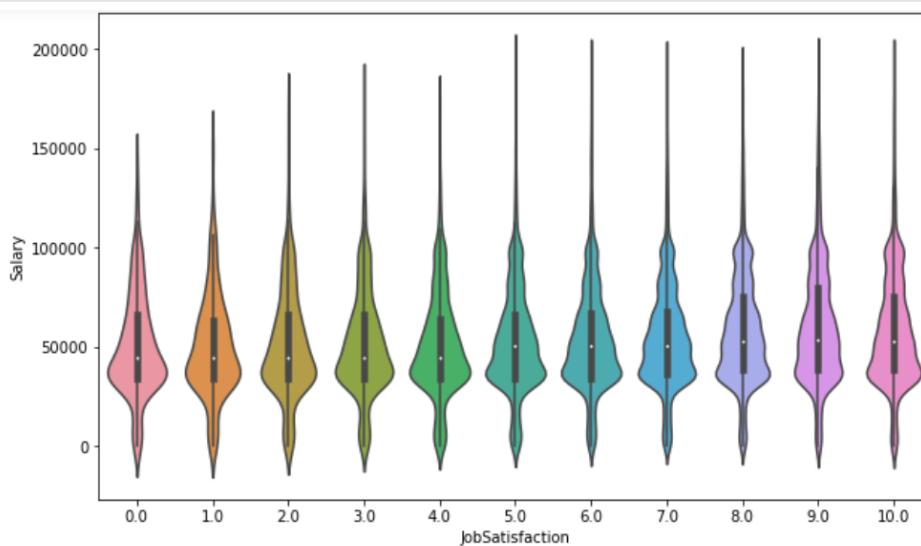
In [15]: plt.figure(figsize=(10,6))
sns.countplot(df["YearsCodedJob"])

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x9fbbeef8>
```



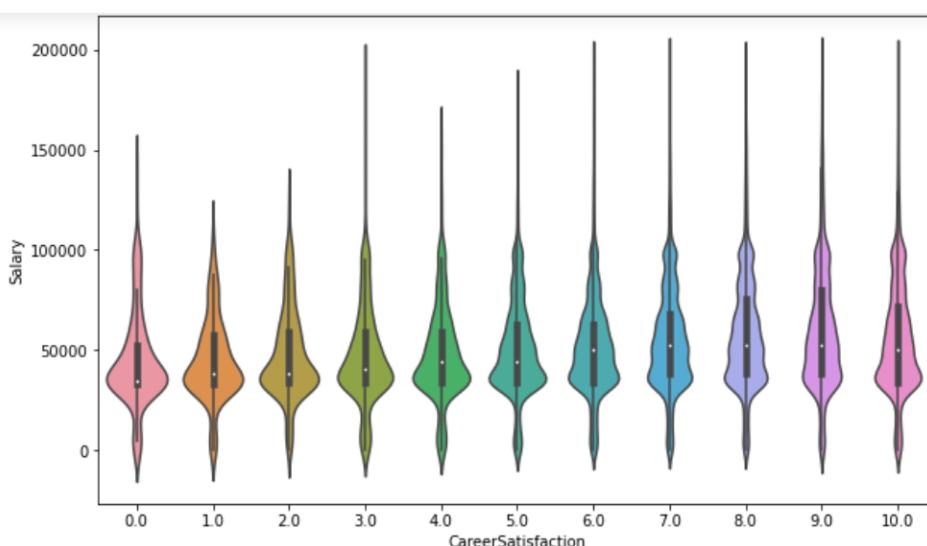
```
In [16]: #df.to_csv('updatedDatasets/Update1.csv')
```

```
In [17]: plt.figure(figsize=(10,6))
sns.violinplot(data=df,y='Salary',x='JobSatisfaction')
```



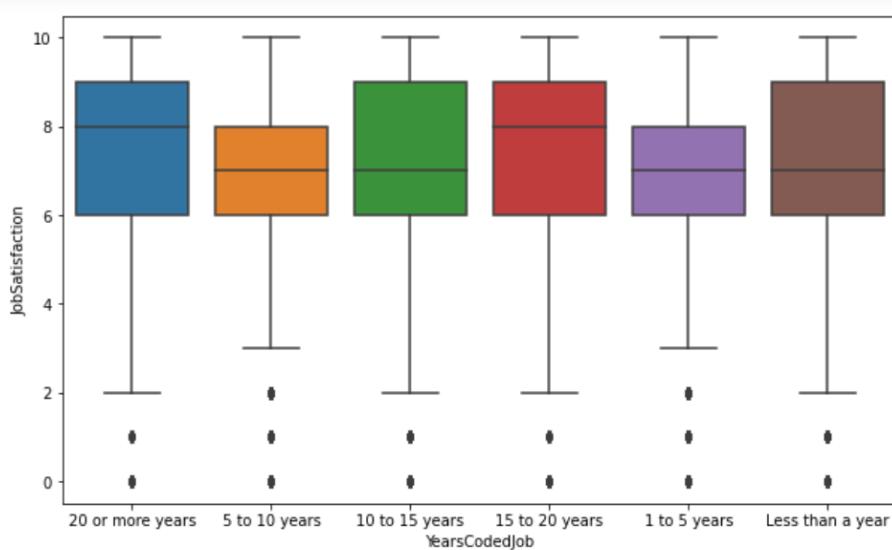
```
In [18]: plt.figure(figsize=(10,6))
sns.violinplot(data=df,y='Salary',x='CareerSatisfaction')
```

```
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0xa849d60>
```



```
In [19]: plt.figure(figsize=(10,6))
sns.boxplot(data=df,x='YearsCodedJob',y='JobSatisfaction')
```

```
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0xc35b598>
```



```
In [20]: plt.figure(figsize =(10,6))
sns.boxplot(data=df,x='YearsCodedJob',y='CareerSatisfaction')
```

```
Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0xcb75a18>
```

Assess_Score

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

```
In [2]: df = pd.read_csv('updatedDatasets/Update1.csv')
```

```
In [3]: df['AssessJobIndustry'].unique()
```

```
Out[3]: array(['Very important', 'nan', 'Somewhat important', 'Important',
       'Not very important', 'Not at all important'], dtype=object)
```

```
In [4]: df.iloc[:,21:35].head()
```

```
Out[4]: AssessJobIndustry AssessJobRole AssessJobExp AssessJobDept AssessJobTech AssessJobProjects AssessJobCompensation AssessJobOffice AssessJobCommute AssessJobRemote AssessJobProfDevel AssessJobDiversity AssessJobProduct AssessJobFinances
0 Very important Very important Important Very important Very important Very important Very important Important Very important Very important
1 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
2 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
3 Somewhat important Somewhat important Somewhat important Important Important Very important Important Very important
4 NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
```

```
In [5]: pd.DataFrame(df['AssessJobIndustry'].value_counts())
```

```
Out[5]: AssessJobIndustry
Important    7391
Somewhat important    6637
Not very important    3721
Very important    3521
Not at all important    860
```

```
In [6]: #When you're assessing potential jobs to apply to, how important are each of the following to you?
#AssessJobIndustry : The industry that I'd be working in
#AssessJobRole : The specific role or job title I'd be applying for
#AssessJobExp : The experience level called for in the job description
#AssessJobDept : The specific department or team I'd be working on
#AssessJobTech : The languages, frameworks, and other technologies I'd be working with
#AssessJobProjects : How projects are managed at the company or organization
#AssessJobCompensation : The compensation and benefits offered
#AssessJobOffice : The office environment I'd be working in
#AssessJobCommute : The amount of time I'd have to spend commuting
#AssessJobRemote : The opportunity to work from home/remote
#AssessJobProfDevel : Opportunities for professional development
#AssessJobDiversity : The diversity of the company or organization
#AssessJobProduct : How widely used or impactful the product or service I'd be working on is
#AssessJobFinances : The financial performance or funding status of the company or organization
```

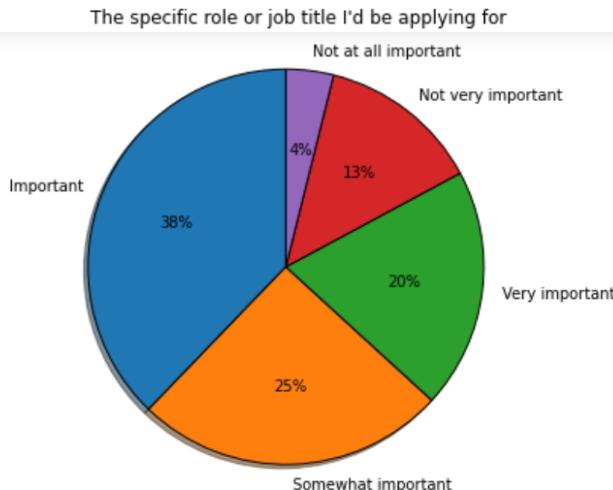
```
In [7]: plt.figure(figsize=(10,6))
plt.pie(df['AssessJobIndustry'].value_counts(), labels = df['AssessJobIndustry'].value_counts().index,
        startangle=90, wedgeprops={'edgecolor': 'black'}, autopct='%.1f%%', shadow=True)
plt.title("The industry that I'd be working in")
```

```
Out[7]: Text(0.5, 1.0, "The industry that I'd be working in")
```

Importance Level	Percentage
Important	33%
Very important	16%
Not very important	17%
Not at all important	30%
Somewhat important	4%

```
In [8]: plt.figure(figsize=(10,6))
plt.pie(df['AssessJobRole'].value_counts(), labels = df['AssessJobRole'].value_counts().index,
        startangle=90, wedgeprops={'edgecolor': 'black'}, autopct='%1.f%%', shadow=True)
plt.title("The specific role or job title I'd be applying for")
```

```
Out[8]: Text(0.5, 1.0, "The specific role or job title I'd be applying for")
```



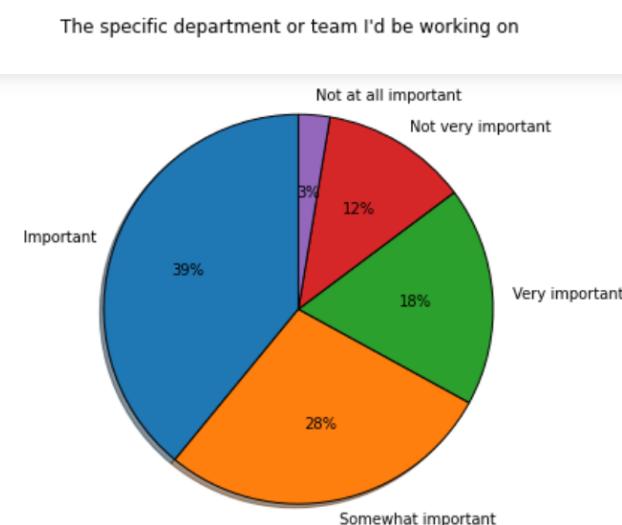
```
In [9]: plt.figure(figsize=(10,6))
plt.pie(df['AssessJobExp'].value_counts(), labels = df['AssessJobExp'].value_counts().index,
        startangle=90, wedgeprops={'edgecolor': 'black'}, autopct='%1.f%%', shadow=True)
plt.title("The experience level called for in the job description")
```

```
Out[9]: Text(0.5, 1.0, 'The experience level called for in the job description')
```



```
In [10]: plt.figure(figsize=(10,6))
plt.pie(df['AssessJobDept'].value_counts(), labels = df['AssessJobDept'].value_counts().index,
        startangle=90, wedgeprops={'edgecolor': 'black'}, autopct='%1.f%%', shadow=True)
plt.title("The specific department or team I'd be working on")
```

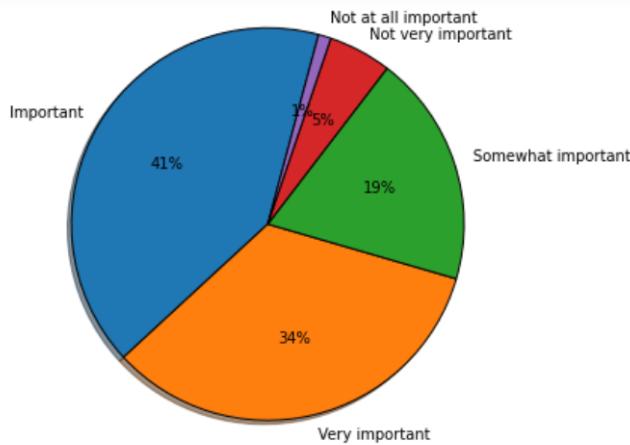
```
Out[10]: Text(0.5, 1.0, "The specific department or team I'd be working on")
```



```
In [11]: plt.figure(figsize=(10,6))
plt.pie(df['AssessJobTech'].value_counts(), labels = df['AssessJobTech'].value_counts().index,
        startangle=75, wedgeprops={'edgecolor': 'black'}, autopct='%1.f%%', shadow=True)
plt.title("The languages, frameworks, and other technologies I'd be working with")
```

```
Out[11]: Text(0.5, 1.0, "The languages, frameworks, and other technologies I'd be working with")
```

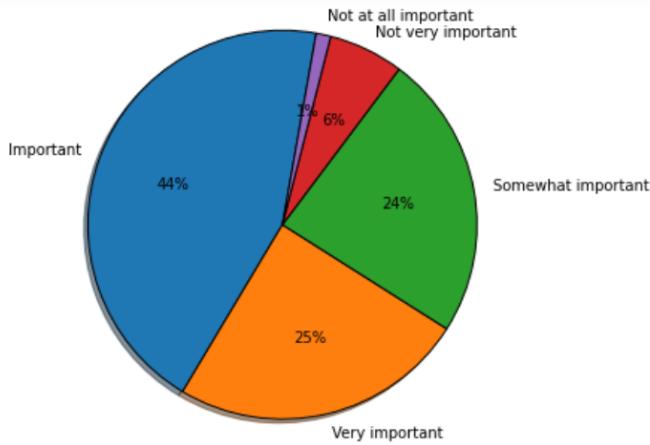
The languages, frameworks, and other technologies I'd be working with



```
In [12]: plt.figure(figsize=(10,6))
plt.pie(df['AssessJobProjects'].value_counts(), labels = df['AssessJobProjects'].value_counts().index,
        startangle=80, wedgeprops={'edgecolor': 'black'}, autopct='%1.f%%', shadow=True)
plt.title("How projects are managed at the company or organization")
```

```
Out[12]: Text(0.5, 1.0, 'How projects are managed at the company or organization')
```

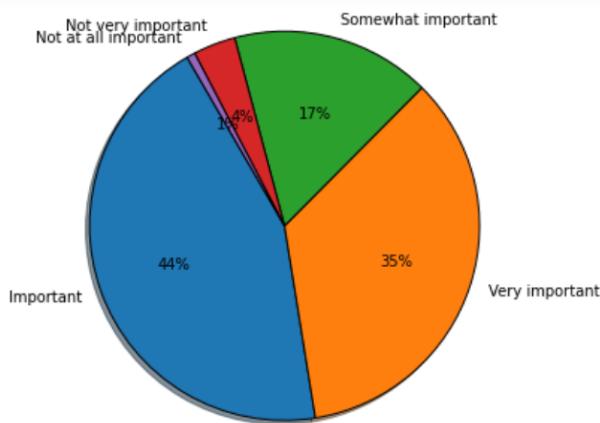
How projects are managed at the company or organization



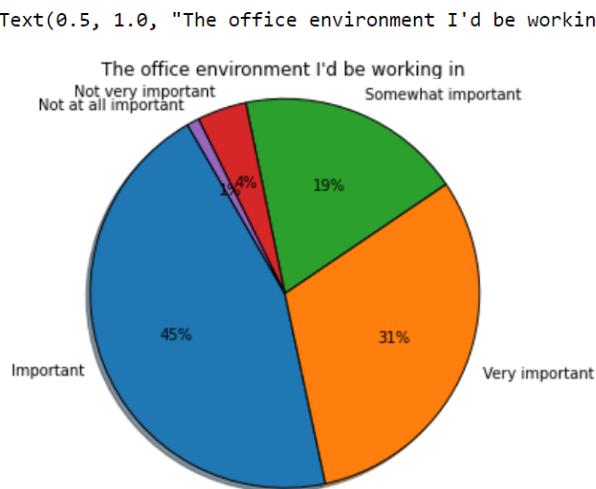
```
In [13]: plt.figure(figsize=(10,6))
plt.pie(df['AssessJobCompensation'].value_counts(), labels = df['AssessJobCompensation'].value_counts().index,
        startangle=120, wedgeprops={'edgecolor': 'black'}, autopct='%1.f%%', shadow=True)
plt.title("The compensation and benefits offered")
```

```
Out[13]: Text(0.5, 1.0, 'The compensation and benefits offered')
```

The compensation and benefits offered

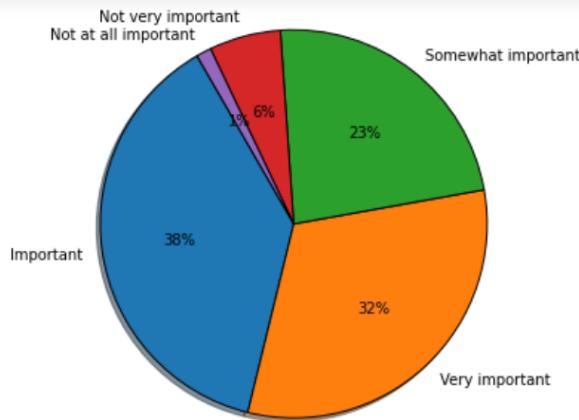


```
In [14]: plt.figure(figsize=(10,6))
plt.pie(df['AssessJobOffice'].value_counts(), labels = df['AssessJobOffice'].value_counts().index,
        startangle=120, wedgeprops={'edgecolor': 'black'}, autopct='%1.f%%', shadow=True)
plt.title("The office environment I'd be working in")
```



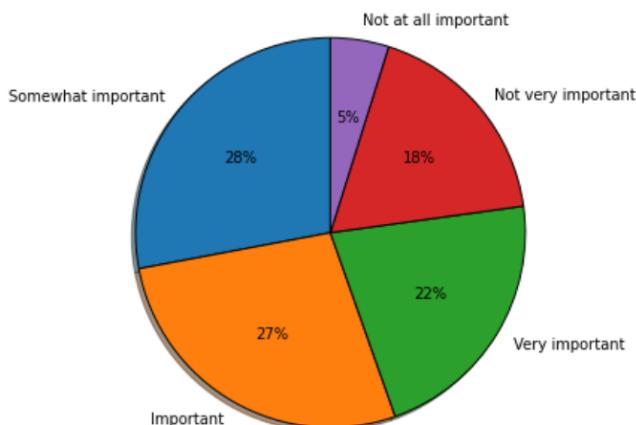
```
In [15]: plt.figure(figsize=(10,6))
plt.pie(df['AssessJobCommute'].value_counts(), labels = df['AssessJobCommute'].value_counts().index,
        startangle=120, wedgeprops={'edgecolor': 'black'}, autopct='%1.f%%', shadow=True)
plt.title("The amount of time I'd have to spend commuting")
```

Out[15]: Text(0.5, 1.0, "The amount of time I'd have to spend commuting")



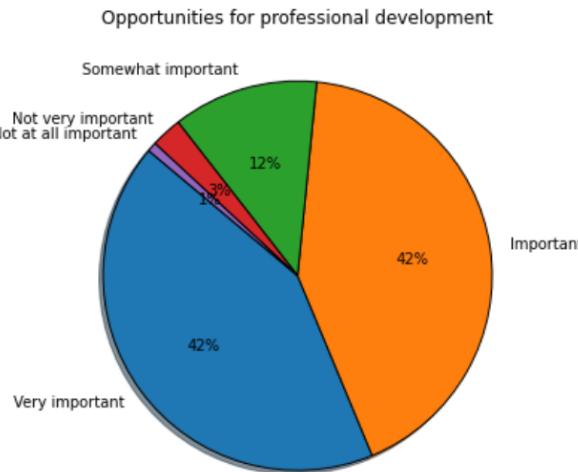
```
In [16]: plt.figure(figsize=(10,6))
plt.pie(df['AssessJobRemote'].value_counts(), labels = df['AssessJobRemote'].value_counts().index,
        startangle=90, wedgeprops={'edgecolor': 'black'}, autopct='%1.f%%', shadow=True)
plt.title("The opportunity to work from home/remotely")
```

Out[16]: Text(0.5, 1.0, 'The opportunity to work from home/remotely')



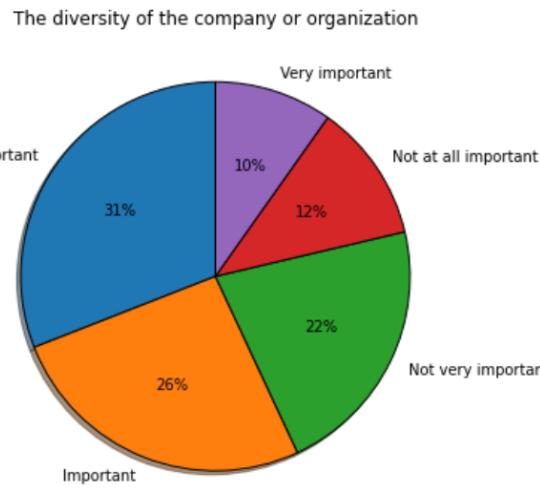
```
In [17]: plt.figure(figsize=(10,6))
plt.pie(df['AssessJobProfDevel'].value_counts(), labels = df['AssessJobProfDevel'].value_counts().index,
        startangle=140, wedgeprops={'edgecolor': 'black'}, autopct='%1.f%%', shadow=True)
plt.title("Opportunities for professional development")
```

```
Out[17]: Text(0.5, 1.0, 'Opportunities for professional development')
```



```
In [18]: plt.figure(figsize=(10,6))
plt.pie(df['AssessJobDiversity'].value_counts(), labels = df['AssessJobDiversity'].value_counts().index,
        startangle=90, wedgeprops={'edgecolor': 'black'}, autopct='%1.f%%', shadow=True)
plt.title("The diversity of the company or organization")
```

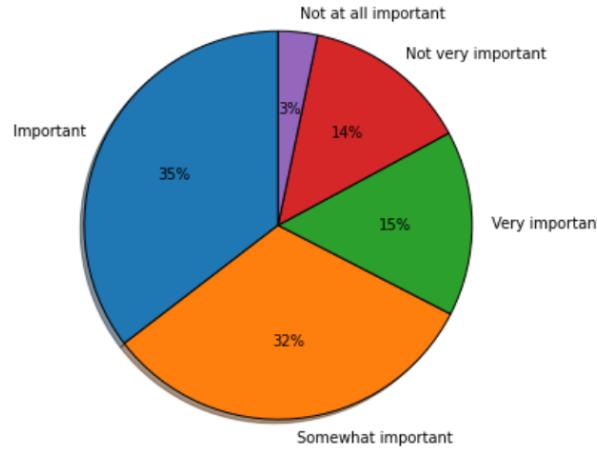
```
Out[18]: Text(0.5, 1.0, 'The diversity of the company or organization')
```



```
In [19]: plt.figure(figsize=(10,6))
plt.pie(df['AssessJobProduct'].value_counts(), labels = df['AssessJobProduct'].value_counts().index,
        startangle=90, wedgeprops={'edgecolor': 'black'}, autopct='%1.f%%', shadow=True)
plt.title("How widely used or impactful the product or service I'd be working on is")
```

```
Out[19]: Text(0.5, 1.0, "How widely used or impactful the product or service I'd be working on is")
```

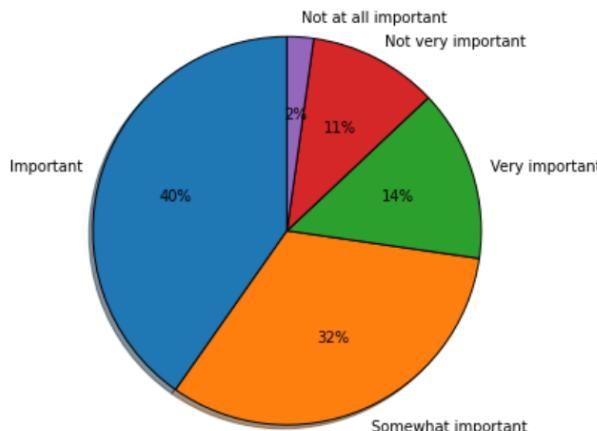
How widely used or impactful the product or service I'd be working on is



```
In [20]: plt.figure(figsize=(10,6))
plt.pie(df['AssessJobFinances'].value_counts(), labels = df['AssessJobFinances'].value_counts().index,
        startangle=90, wedgeprops={'edgecolor': 'black'}, autopct='%.1f%%', shadow=True)
plt.title("The financial performance or funding status of the company or organization")
```

```
Out[20]: Text(0.5, 1.0, 'The financial performance or funding status of the company or organization')
```

The financial performance or funding status of the company or organization



```
In [21]: def assess_score(cols):
    score = 0

    for x in cols:
        if pd.isnull(x):
            score += 0
        elif x == 'Not at all important':
            score += 0.5
        elif x == 'Not very important':
            score += 1
        elif x == 'Somewhat important':
            score += 1.5
        elif x == 'Important':
            score += 2
        else:
            score += 2.5

    return score
```

```
In [22]: temp = df.iloc[:,21:35].apply(assess_score, axis=1)
```

```
In [23]: temp.head(3)
```

```
Out[23]: 0    30.5
1     0.0
2     0.0
dtype: float64
```

```
In [24]: df['Assess_Scoring']=temp
```

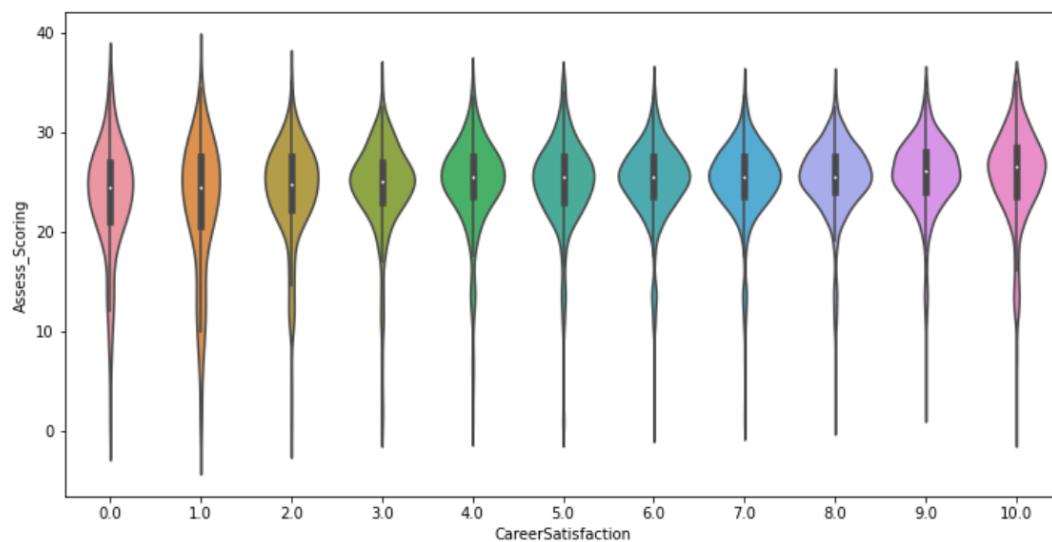
```
In [25]: temp=df.drop(df.iloc[:,21:35],axis=1)
temp['Assess_Scoring'].replace({0:np.nan}, inplace=True)
```

```
In [26]: temp[temp['Assess_Scoring']!=0]['Assess_Scoring'].count()
```

```
Out[26]: 22689
```

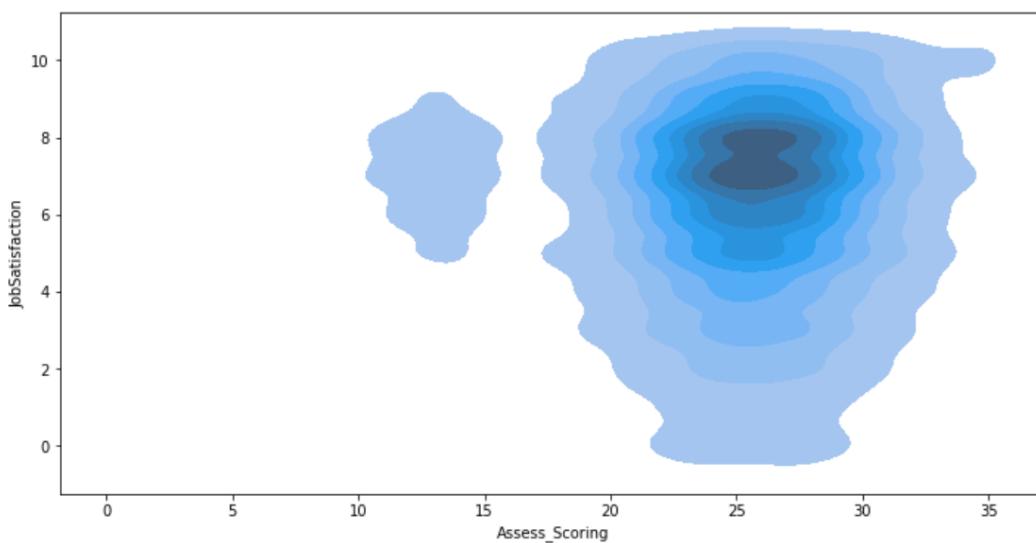
```
In [27]: plt.figure(figsize =(12,6))
sns.violinplot(data=temp,y='Assess_Scoring',x='CareerSatisfaction')
```

```
Out[27]: <AxesSubplot:xlabel='CareerSatisfaction', ylabel='Assess_Scoring'>
```



```
In [28]: plt.figure(figsize =(12,6))
sns.kdeplot(data=temp,x='Assess_Scoring',y='JobSatisfaction',fill=True, common_norm=False, palette="crest",)
```

```
Out[28]: <AxesSubplot:xlabel='Assess_Scoring', ylabel='JobSatisfaction'>
```



```
In [30]: temp.to_csv('updatedDatasets/Update2.csv',index=False)
```

```
In [21]: def assess_score(cols):
    score = 0

    for x in cols:
        if pd.isnull(x):
            score += 0
        elif x == 'Not at all important':
            score += 0.5
        elif x == 'Not very important':
            score += 1
        elif x == 'Somewhat important':
            score += 1.5
        elif x == 'Important':
            score += 2
        else:
            score += 2.5

    return score
```

```
In [23]: temp = df.iloc[:,21:35].apply(assess_score, axis=1)
```

```
In [24]: temp[:5]
```

```
Out[24]: 0    30.5
1     0.0
2     0.0
3    28.5
4     0.0
dtype: float64
```

```
In [25]: df['Assess_Scoring']=temp
```

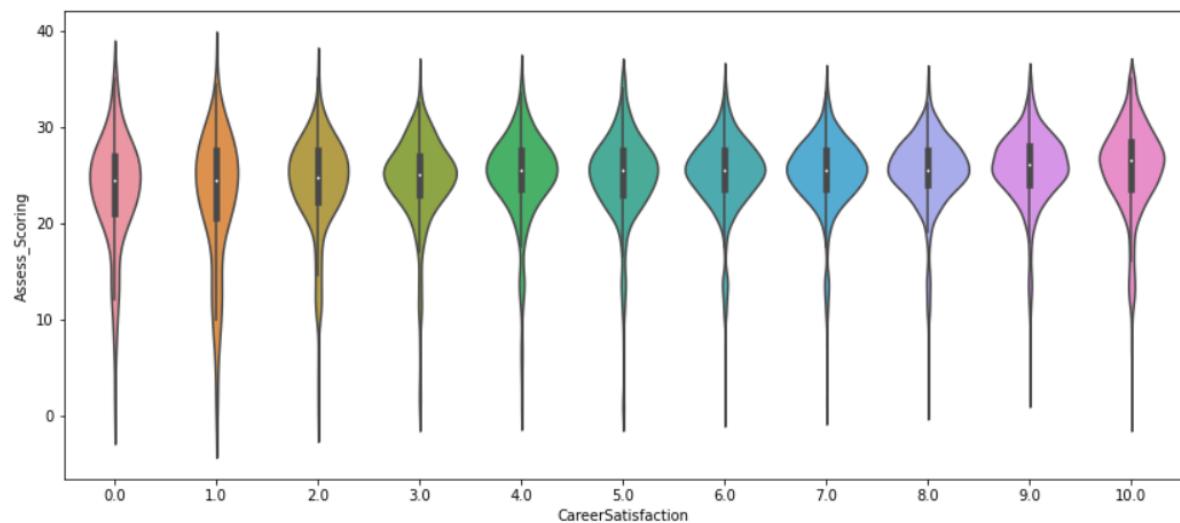
```
In [26]: temp=df.drop(df.iloc[:,21:35],axis=1)
temp['Assess_Scoring'].replace({0:np.nan}, inplace=True)
```

```
In [27]: temp[temp['Assess_Scoring']!=0]['Assess_Scoring'].count()
```

```
Out[27]: 22689
```

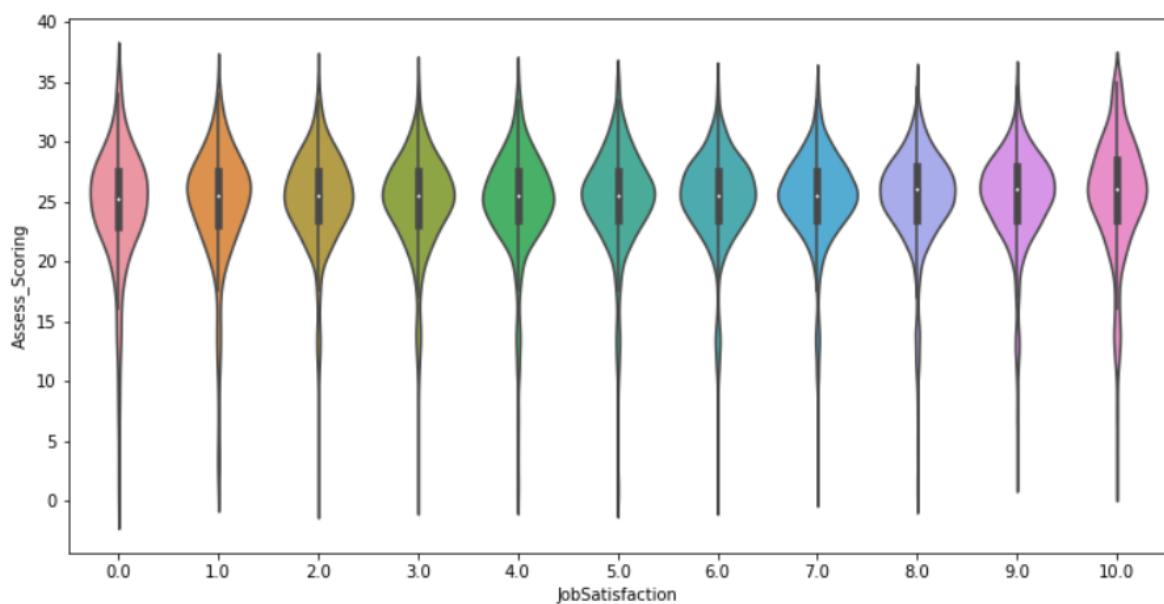
```
In [28]: plt.figure(figsize =(14,6))
sns.violinplot(data=temp,y='Assess_Scoring',x='CareerSatisfaction')
```

```
Out[28]: <AxesSubplot:xlabel='CareerSatisfaction', ylabel='Assess_Scoring'>
```



```
In [29]: plt.figure(figsize =(12,6))
sns.violinplot(data=temp,y='Assess_Scoring',x='JobSatisfaction')
```

```
Out[29]: <AxesSubplot:xlabel='JobSatisfaction', ylabel='Assess_Scoring'>
```



```
In [30]: temp.to_csv( 'updatedDatasets/Update2.csv',index=False)
```

Gender Analysis

```
In [2]: df = pd.read_csv('updatedDatasets/Update2.csv')
```

```
In [22]: df['Gender'].unique()
```

```
Out[22]: array(['Male', nan, 'Female', 'Gender non-conforming', 'Other',
       'Male; Gender non-conforming', 'Female; Transgender',
       'Male; Female', 'Male; Other', 'Transgender',
       'Transgender; Gender non-conforming',
       'Female; Gender non-conforming',
       'Male; Female; Transgender; Gender non-conforming; Other',
       'Male; Female; Transgender', 'Male; Female; Other',
       'Male; Female; Transgender; Gender non-conforming',
       'Male; Transgender', 'Female; Transgender; Gender non-conforming',
       'Gender non-conforming; Other',
       'Male; Female; Gender non-conforming', 'Female; Other',
       'Male; Transgender; Gender non-conforming', 'Transgender; Other',
       'Male; Gender non-conforming; Other',
       'Female; Gender non-conforming; Other',
       'Male; Female; Gender non-conforming; Other',
       'Female; Transgender; Other',
       'Female; Transgender; Gender non-conforming; Other',
       'Male; Transgender; Other', 'Male; Female; Transgender; Other'],
      dtype=object)
```

```
In [23]: df['Gender'].nunique()
```

```
Out[23]: 29
```

Segregation/Cleaning Gender Column

Making Following Categories : Male , Female , NaN , Other

Other contains:Transgender, Gender non-conforming

```
In [24]: def gender_sortner(x):
    if(x == 'Male' or x== 'Female' or pd.isna(x)):
        return x
    else:
        return 'Other'
df["Gender"] = df["Gender"].apply(gender_sortner)
```

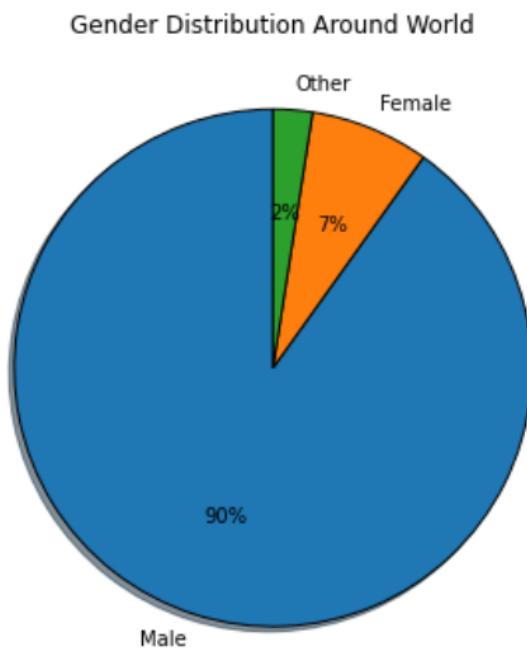
```
In [25]: df['Gender'].unique()
```

```
Out[25]: array(['Male', nan, 'Female', 'Other'], dtype=object)
```

```
In [26]: gender_groupby = df.groupby('Gender')
```

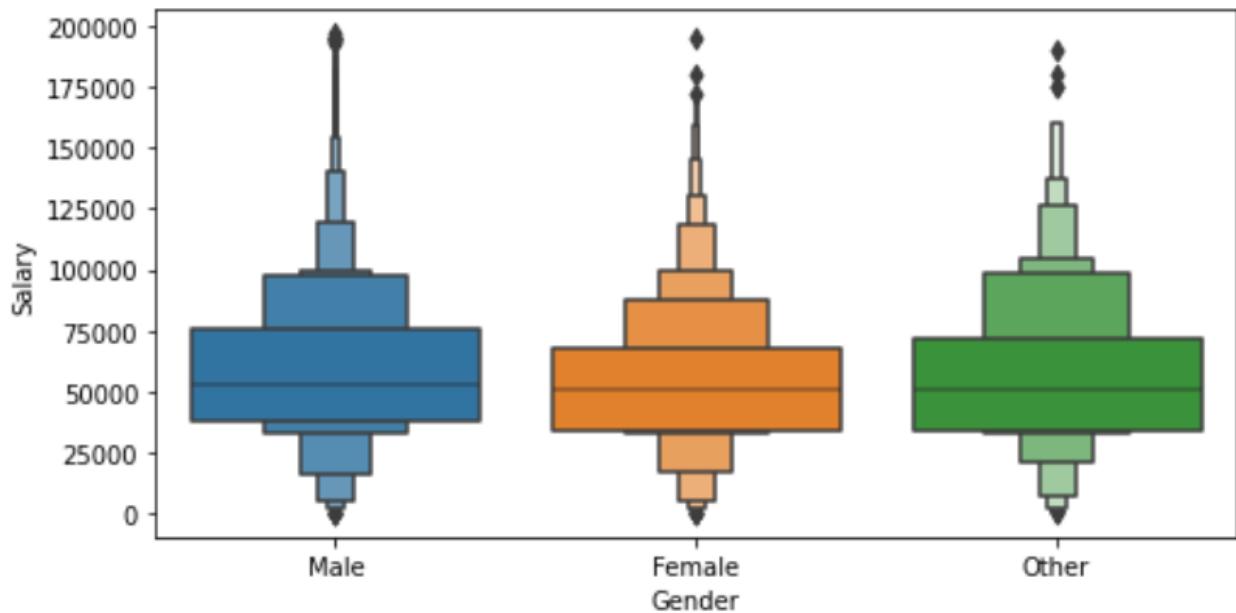
```
In [8]: plt.figure(figsize=(10,6))
plt.pie(df['Gender'].value_counts(), labels = df['Gender'].value_counts().index,
        startangle=90, wedgeprops={'edgecolor': 'black'}, autopct='%.1f%%', shadow=True)
plt.title("Gender Distribution Around World")
```

```
Out[8]: Text(0.5, 1.0, 'Gender Distribution Around World')
```



```
In [9]: plt.figure(figsize=(8,4))
sns.boxenplot(x='Gender',y='Salary',data=df)
```

```
Out[9]: <AxesSubplot:xlabel='Gender', ylabel='Salary'>
```



```
In [10]: gender_groupby['Salary'].mean()
```

```
Out[10]: Gender
Female      54346.519630
Male       57497.391317
Other      56838.943817
Name: Salary, dtype: float64
```

Worldwide Analysis

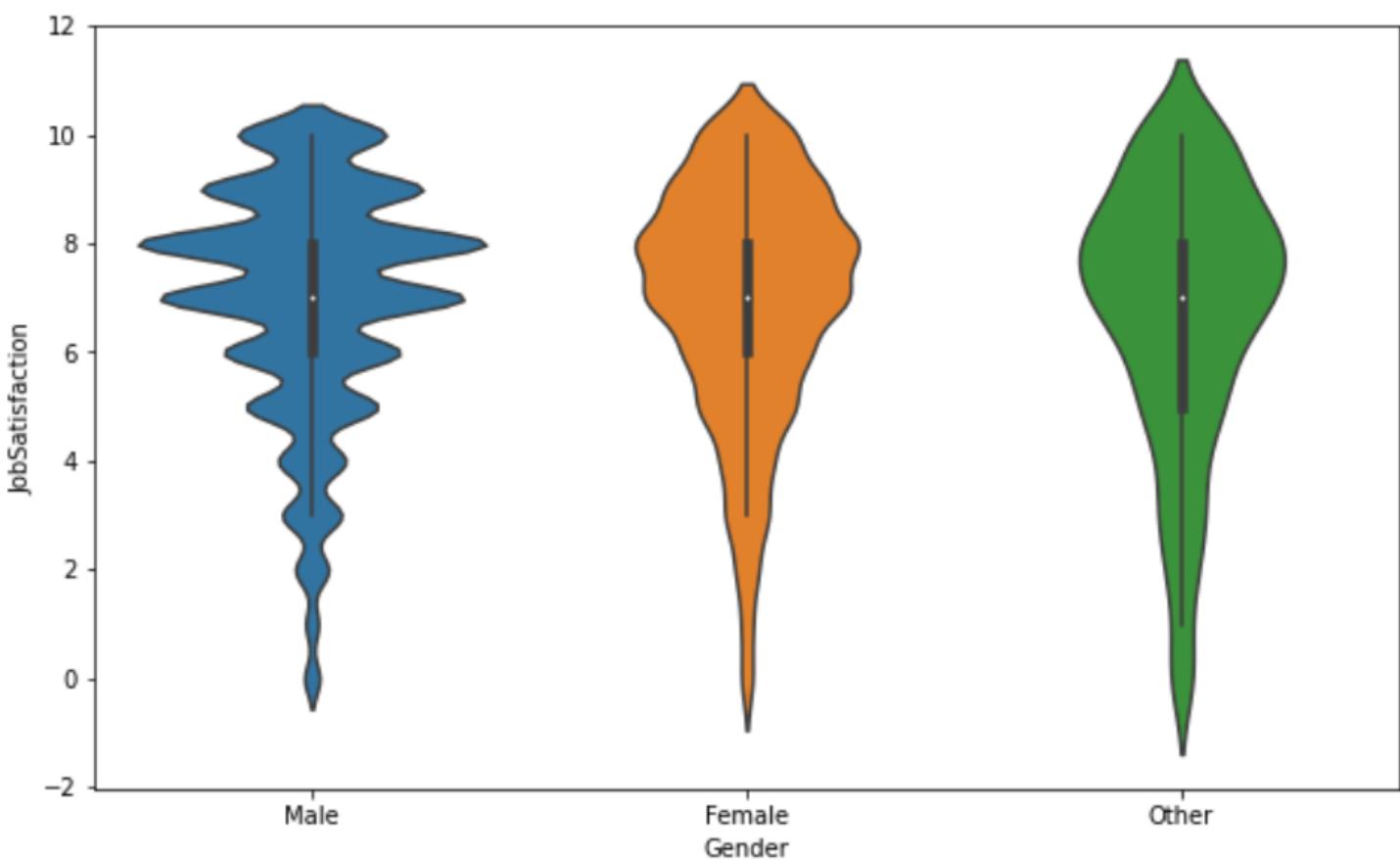
```
In [17]: gender_groupby['JobSatisfaction'].mean()
```

```
Out[17]: Gender
Female    6.936620
Male      7.004372
Other     6.735751
Name: JobSatisfaction, dtype: float64
```

```
In [18]: gender_groupby['JobSatisfaction'].count()
```

```
Out[18]: Gender
Female    2130
Male      26306
Other     579
Name: JobSatisfaction, dtype: int64
```

```
In [19]: plt.figure(figsize=(15,8))
ax=sns.countplot(data=df,x='JobSatisfaction',hue='Gender')
plt.xticks(rotation=45)
for p in ax.patches:
    ax.annotate(format(p.get_height()),(p.get_x(),p.get_height()+0.6))
plt.title(" WORLD WIDE JOB SATISFACTION ")
```



Worldwide Career Satisfaction

```
In [14]: gender_groupby['CareerSatisfaction'].count()
```

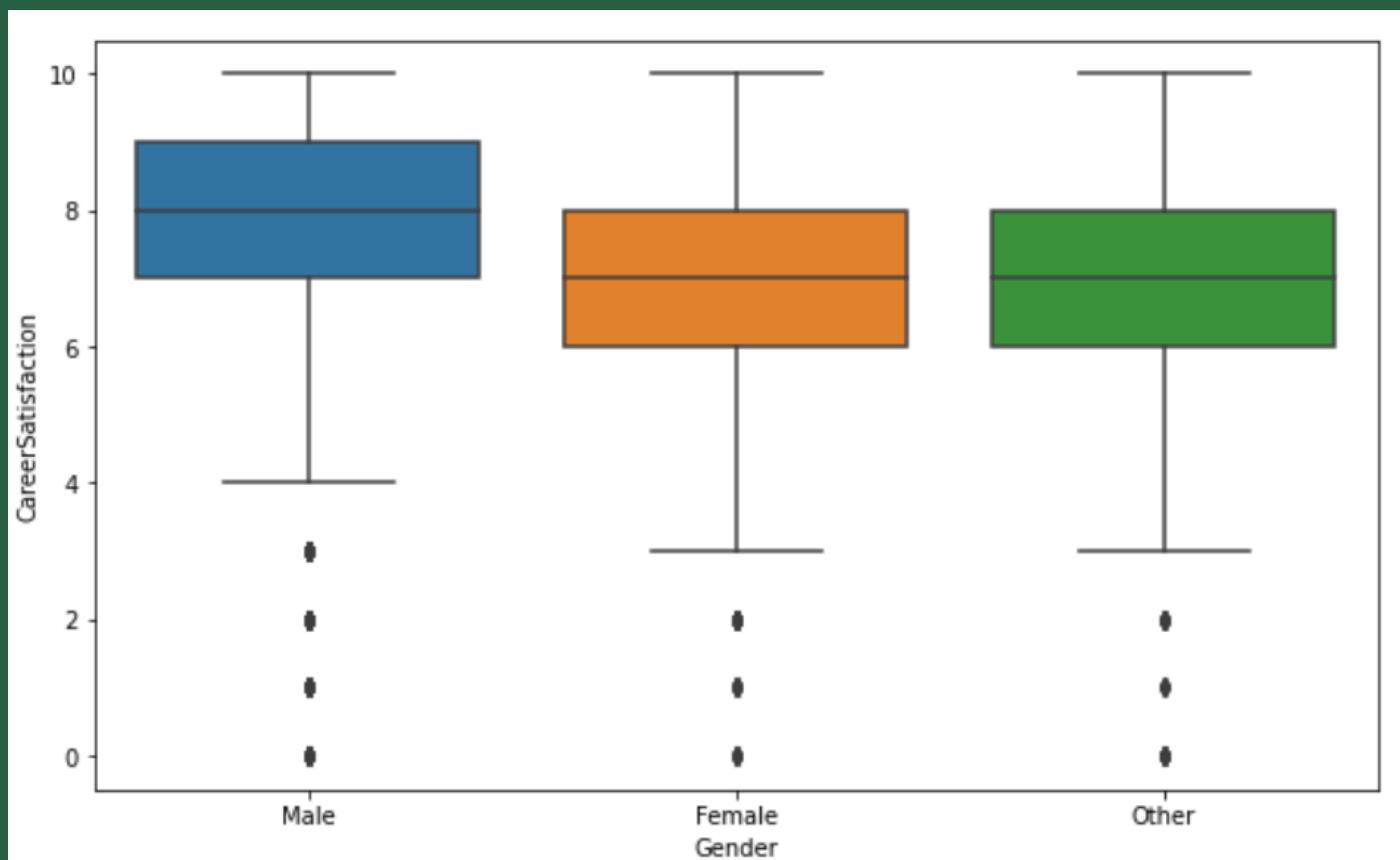
```
Out[14]: Gender
Female    2237
Male     27582
Other      660
Name: CareerSatisfaction, dtype: int64
```

```
In [38]: gender_groupby['CareerSatisfaction'].mean()
```

```
Out[38]: Gender
Female    7.185069
Male     7.394061
Other     6.924242
Name: CareerSatisfaction, dtype: float64
```

```
In [16]: plt.figure(figsize=(10,6))
sns.boxplot(data=df,x='Gender',y='CareerSatisfaction')
```

```
Out[16]: <AxesSubplot:xlabel='Gender', ylabel='CareerSatisfaction'>
```



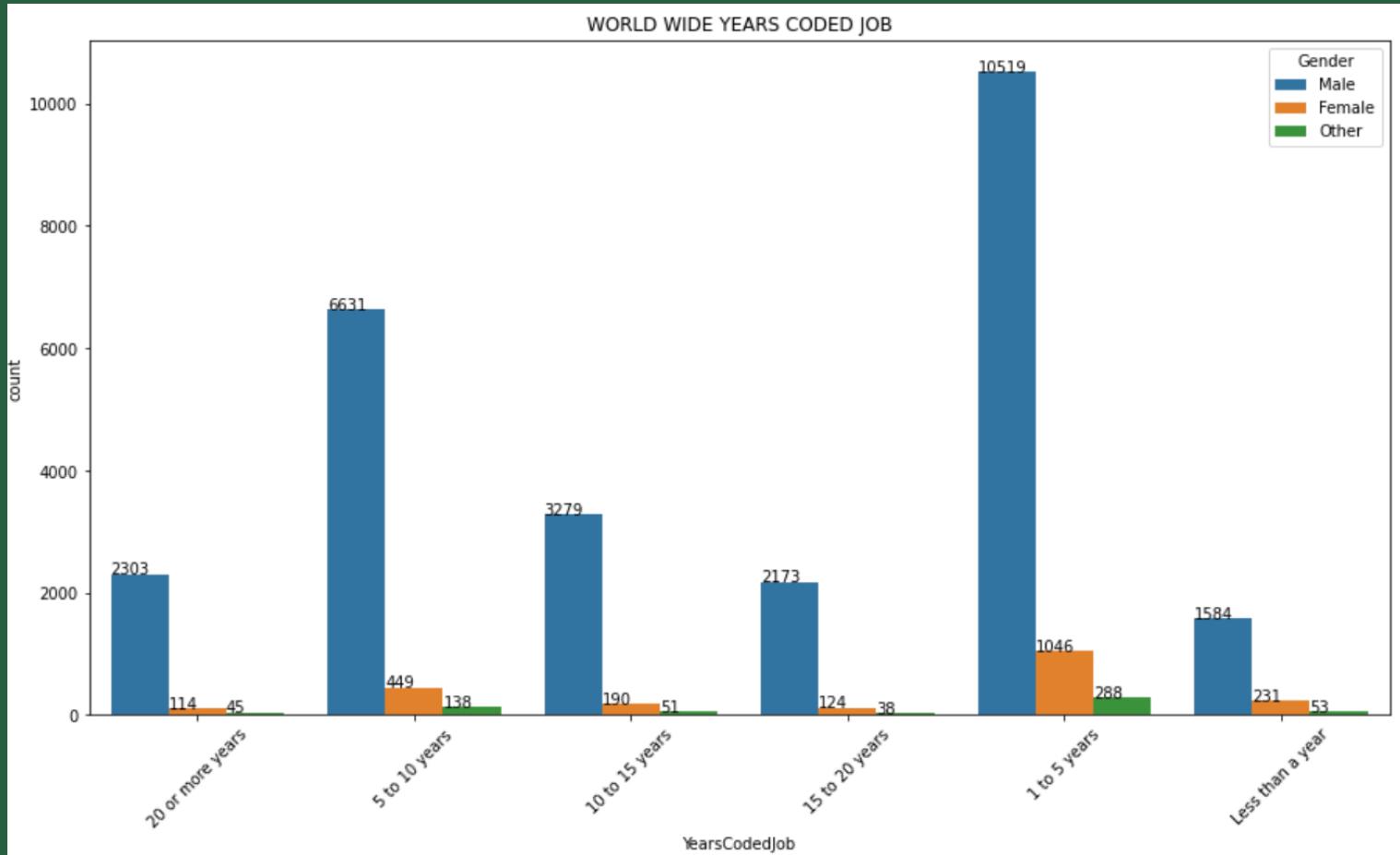
Worldwide Years Coded Job

```
In [17]: gender_groupby['YearsCodedJob'].count()
```

```
Out[17]: Gender
Female      2154
Male       26489
Other       613
Name: YearsCodedJob, dtype: int64
```

```
In [36]: plt.figure(figsize=(15,8))
ax=sns.countplot(data=df,x='YearsCodedJob',hue='Gender')
plt.xticks(rotation=45)
for p in ax.patches:
    ax.annotate(format(p.get_height()),(p.get_x(),p.get_height()+0.6))
plt.title("WORLD WIDE YEARS CODED JOB")
```

```
Out[36]: Text(0.5, 1.0, 'WORLD WIDE YEARS CODED JOB')
```

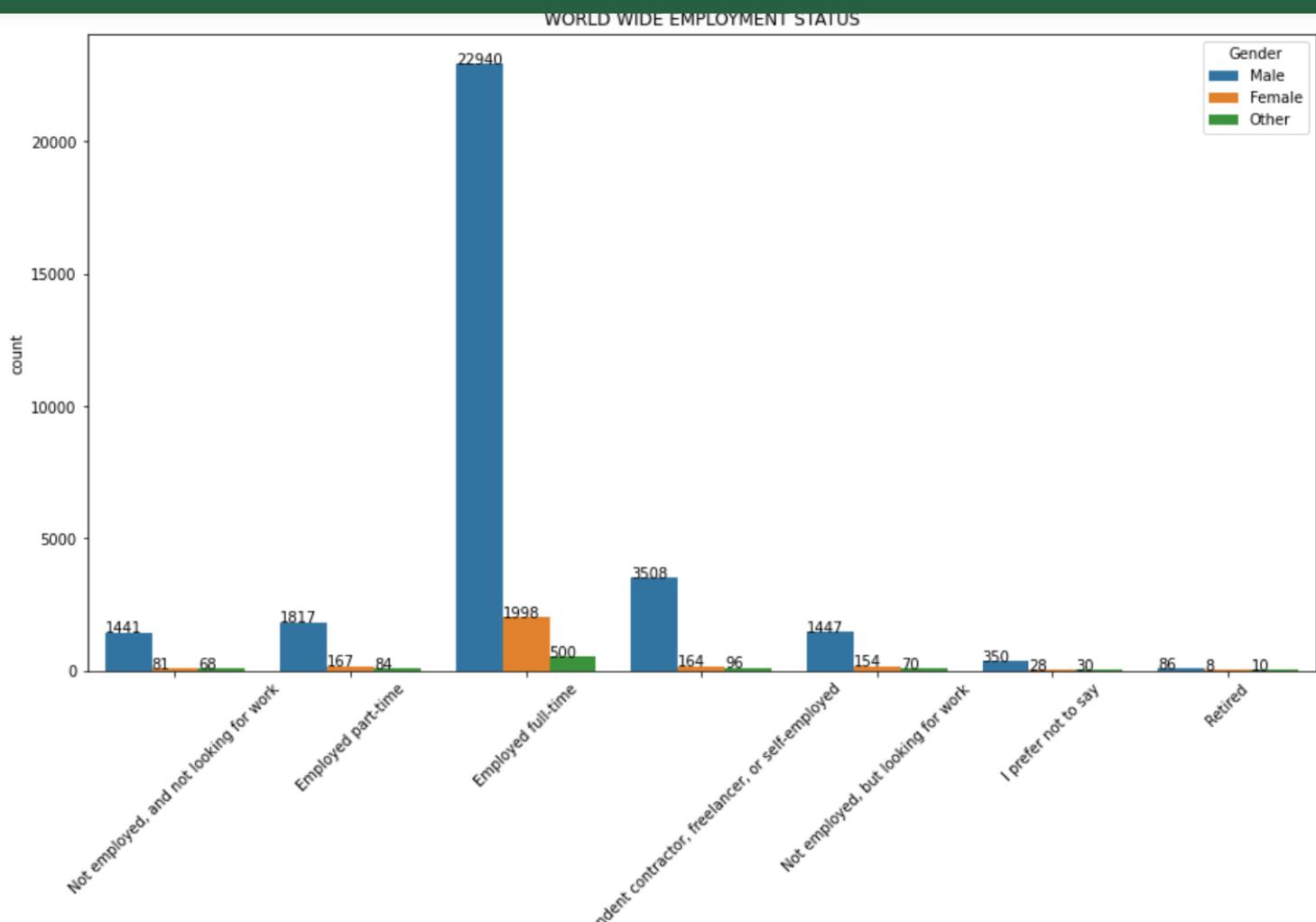


Worldwide Employment Status

```
In [19]: gender_groupby['EmploymentStatus'].count()
```

```
Out[19]: Gender
Female      2600
Male       31589
Other       858
Name: EmploymentStatus, dtype: int64
```

```
In [20]: plt.figure(figsize=(15,8))
ax=sns.countplot(data=df,x='EmploymentStatus',hue='Gender')
plt.xticks(rotation=45)
for p in ax.patches:
    ax.annotate(format(p.get_height()),(p.get_x(),p.get_height()+0.6))
plt.title("WORLD WIDE EMPLOYMENT STATUS")
```



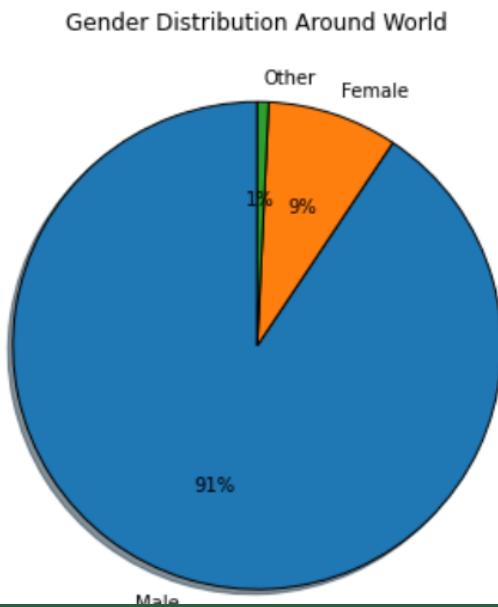
Gender Analysis India

```
In [21]: df_india=df[df['Country']=='India']
gender_groupby_India = df_india.groupby('Gender')
gender_groupby_India['Salary'].mean()
```

```
Out[21]: Gender
Female    33297.340313
Male      33225.401283
Other     33158.461985
Name: Salary, dtype: float64
```

```
In [22]: plt.figure(figsize=(10,6))
plt.pie(df_india['Gender'].value_counts(), labels = df_india['Gender'].value_counts().index,
         startangle=90, wedgeprops={'edgecolor': 'black'}, autopct='%.1f%%', shadow=True)
plt.title("Gender Distribution Around World")
```

```
Out[22]: Text(0.5, 1.0, 'Gender Distribution Around World')
```

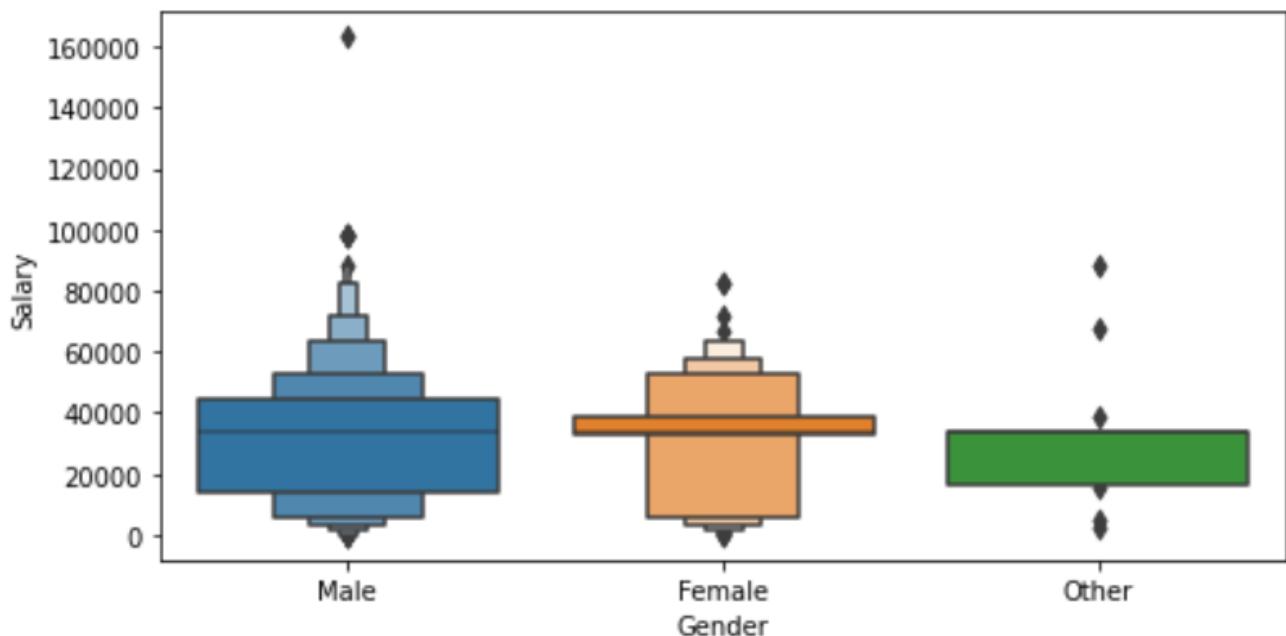


```
In [23]: gender_groupby_India['Salary'].mean()
```

```
Out[23]: Gender  
Female    33297.340313  
Male      33225.401283  
Other     33158.461985  
Name: Salary, dtype: float64
```

```
In [24]: plt.figure(figsize=(8,4))  
sns.boxenplot(x='Gender',y='Salary',data=df_india)
```

```
Out[24]: <AxesSubplot:xlabel='Gender', ylabel='Salary'>
```

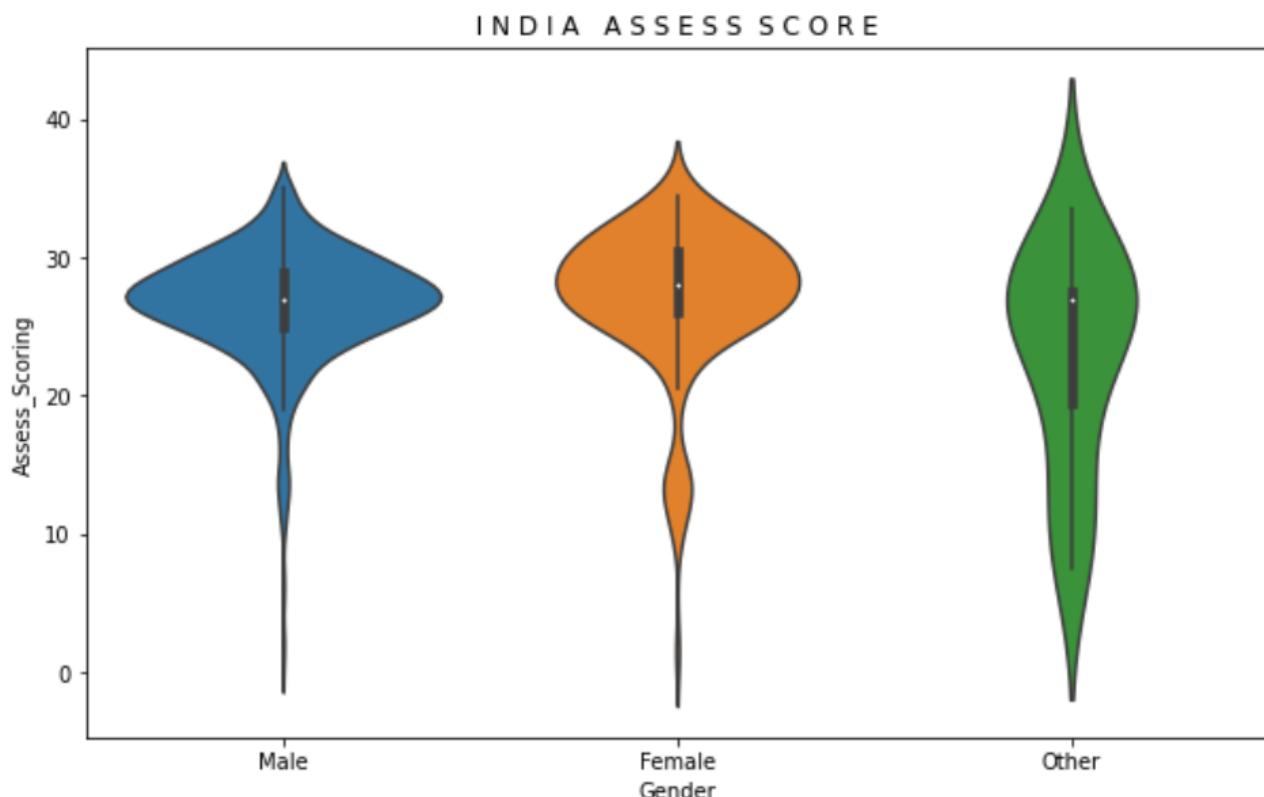


```
In [26]: gender_groupby_India['Assess_Scoring'].mean()
```

```
Out[26]: Gender
Female    27.200000
Male      26.681507
Other     22.923077
Name: Assess_Scoring, dtype: float64
```

```
In [27]: plt.figure(figsize=(10,6))
sns.violinplot(data=df_india,x='Gender',y='Assess_Scoring')
plt.title(" INDIA ASSESS SCORE ")
```

```
Out[27]: Text(0.5, 1.0, ' INDIA ASSESS SCORE ')
```

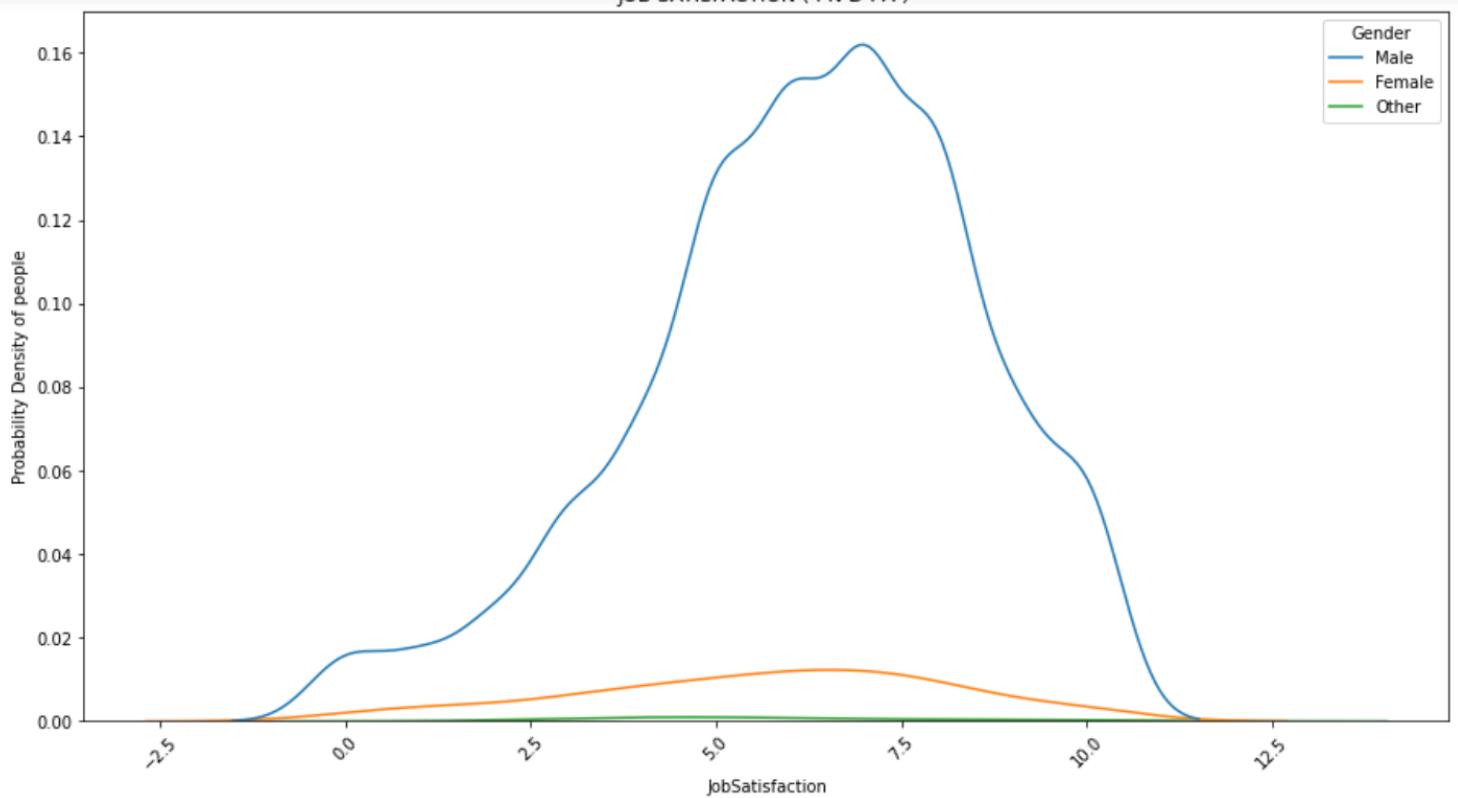


```
In [28]: gender_groupby_India['JobSatisfaction'].mean()
```

```
Out[28]: Gender
Female    5.648485
Male      6.222639
Other     5.750000
Name: JobSatisfaction, dtype: float64
```

```
In [29]: plt.figure(figsize=(15,8))
ax=sns.kdeplot(data=df_india,x='JobSatisfaction',hue='Gender')
plt.xticks(rotation=45)
for p in ax.patches:
    ax.annotate(format(p.get_height()),(p.get_x(),p.get_height()+0.6))
plt.title("JOB SATISFACTION ( INDIA ) ")
plt.ylabel("Probability Density of people")
```

```
Out[29]: Text(0, 0.5, 'Probability Density of people')
```

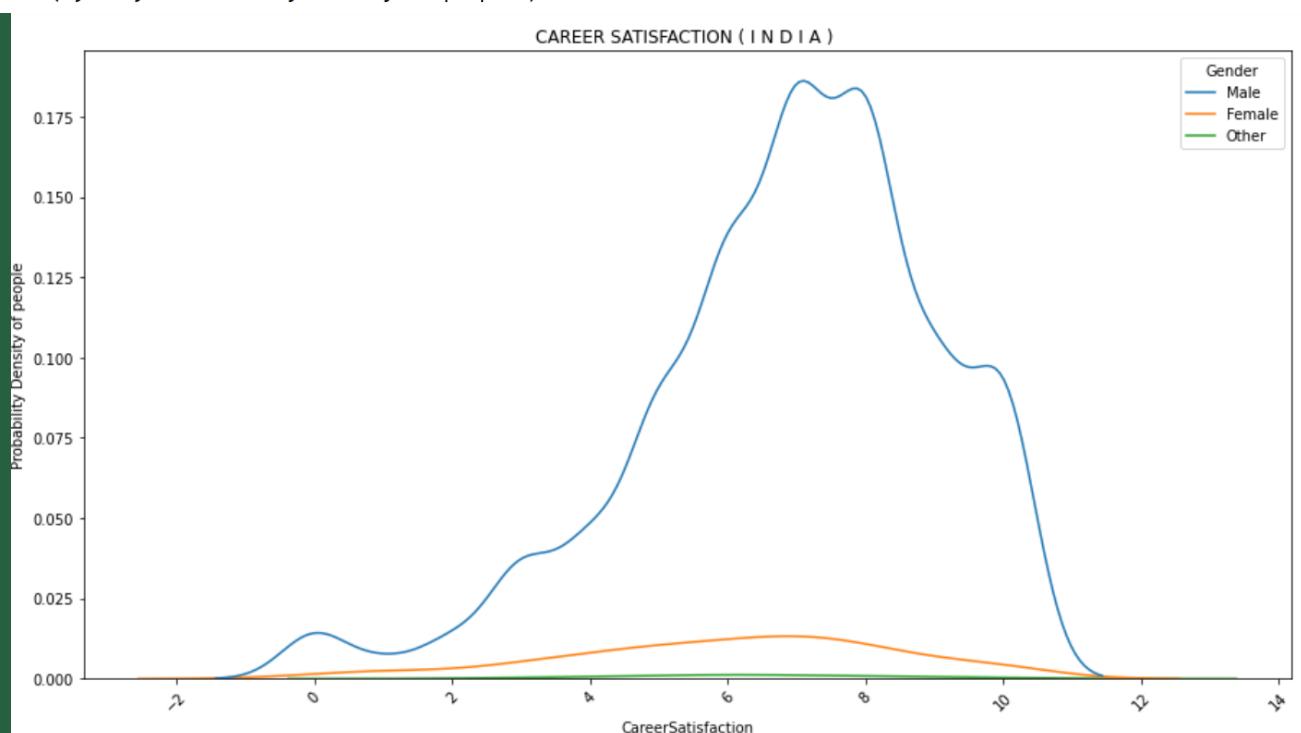


```
[30]: gender_groupby_India['CareerSatisfaction'].mean()
```

```
[30]: Gender
Female    6.028736
Male      6.856557
Other     6.428571
```

```
In [31]: plt.figure(figsize=(15,8))
ax=sns.kdeplot(data=df_india,x='CareerSatisfaction',hue='Gender')
plt.xticks(rotation=45)
for p in ax.patches:
    ax.annotate(format(p.get_height()),(p.get_x(),p.get_height()+0.6))
plt.title("CAREER SATISFACTION ( I N D I A ) ")
plt.ylabel("Probability Density of people")
```

```
Out[31]: Text(0, 0.5, 'Probability Density of people')
```



```
In [32]: df.to_csv('updatedDatasets/Update3.csv',index=False)
df_india.to_csv('updatedDatasets/UpdatedIndia.csv',index=False)
```

Country

```
In [2]: df = pd.read_csv('updatedDatasets/Update3.csv',index_col='Respondent')
```

```
In [3]: df['Country'].nunique()
```

Out[3]: 201

```
In [4]: country_groupby = df.groupby('Country')
```

```
In [5]: country_avaerage_salary = pd.DataFrame(country.groupby['Salary'].mean())
country_avaerage_salary.sort_values('Salary', ascending=False).head()
```

out[5]:	Salary
Country	
Bermuda	150000.000000
Anguilla	100000.000000
American Samoa	98128.097477
Burundi	98128.097477
Libya	87996.924975

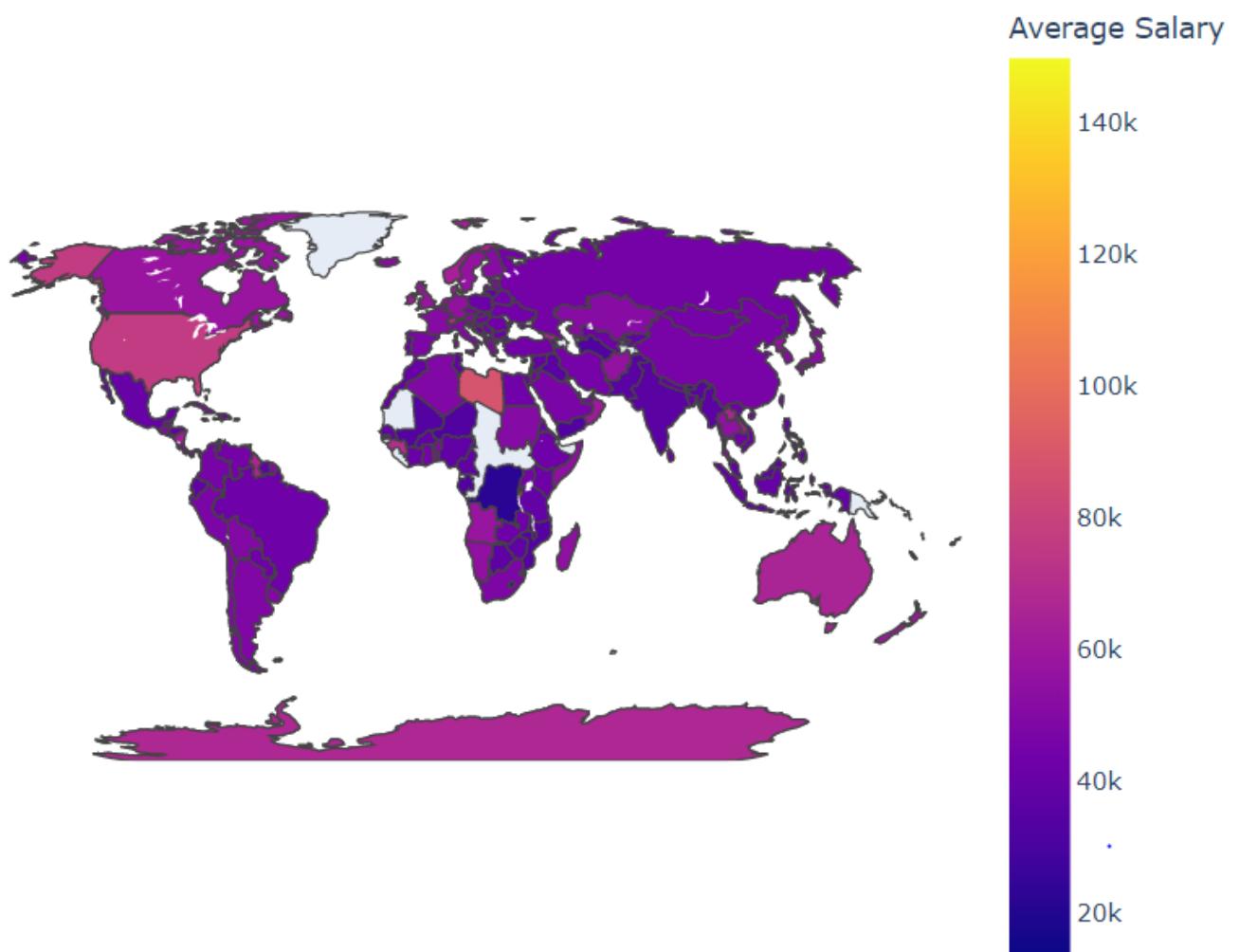
```
In [6]: import chart_studio.plotly as py  
from plotly.offline import download_plotlyjs, init_notebook_mode, plot ,iplot  
init_notebook_mode(connected=True)  
import plotly.graph_objs as go
```

```
In [7]: data = dict(type='choropleth',
                  locations=country_avaerage_salary.index,
                  locationmode='country names',
                  z = country_avaerage_salary['Salary'],
                  colorbar = {'title':'Average Salary'}))
```

```
In [9]: choromap = go.Figure(data= [data],layout=layout)
```

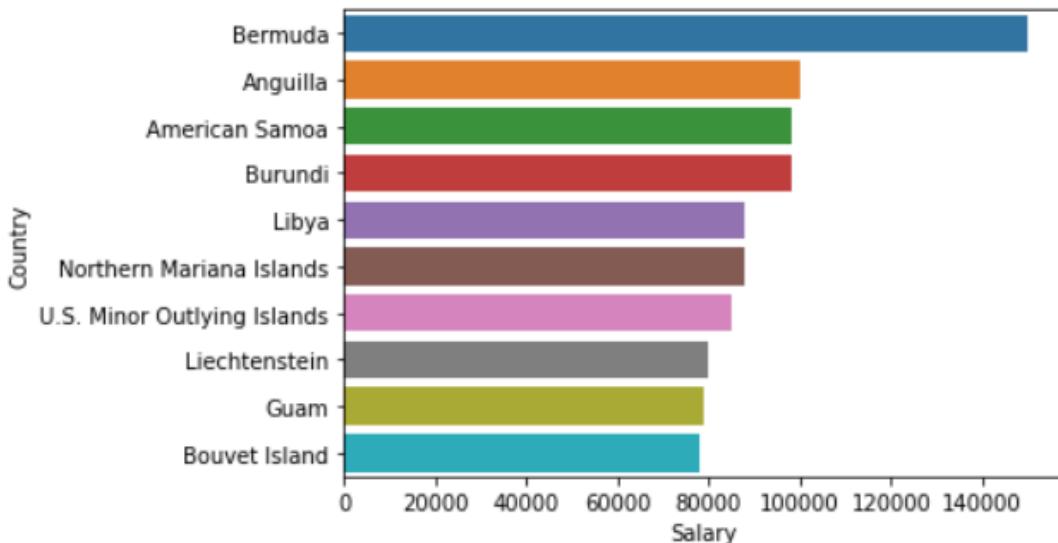
```
In [10]: iplot(choromap)
```

Average Salary per country



```
In [11]: temp = country_average_salary.sort_values('Salary', ascending=False).head(10)  
sns.barplot(x=temp['Salary'], y=temp.index)
```

```
Out[11]: <AxesSubplot:xlabel='Salary', ylabel='Country'>
```



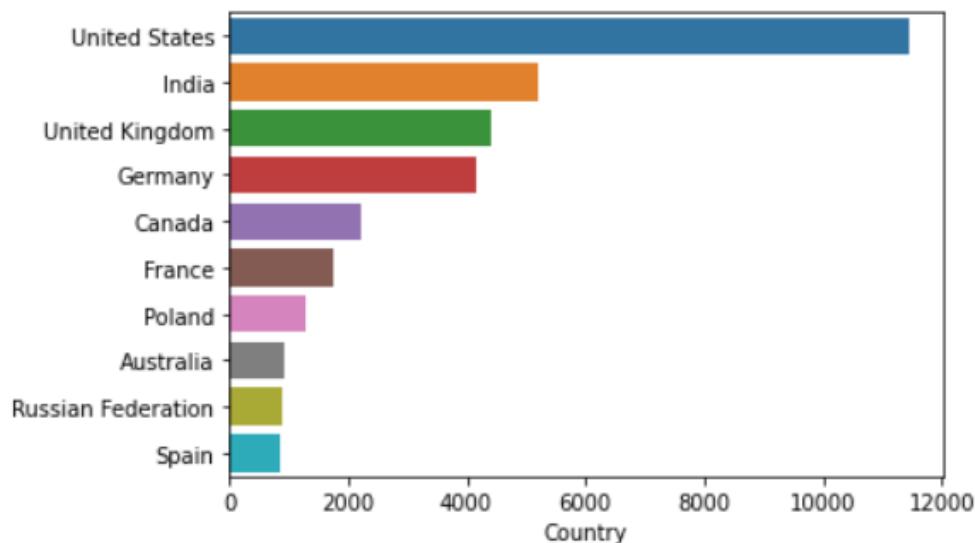
```
In [12]: country = pd.DataFrame(df.Country.value_counts())  
country.sort_values('Country', ascending=False).head()
```

```
Out[12]:
```

	Country
United States	11455
India	5197
United Kingdom	4395
Germany	4143
Canada	2233

```
In [13]: temp = country.sort_values('Country', ascending=False).head(10)  
sns.barplot(x=temp['Country'], y=temp.index)
```

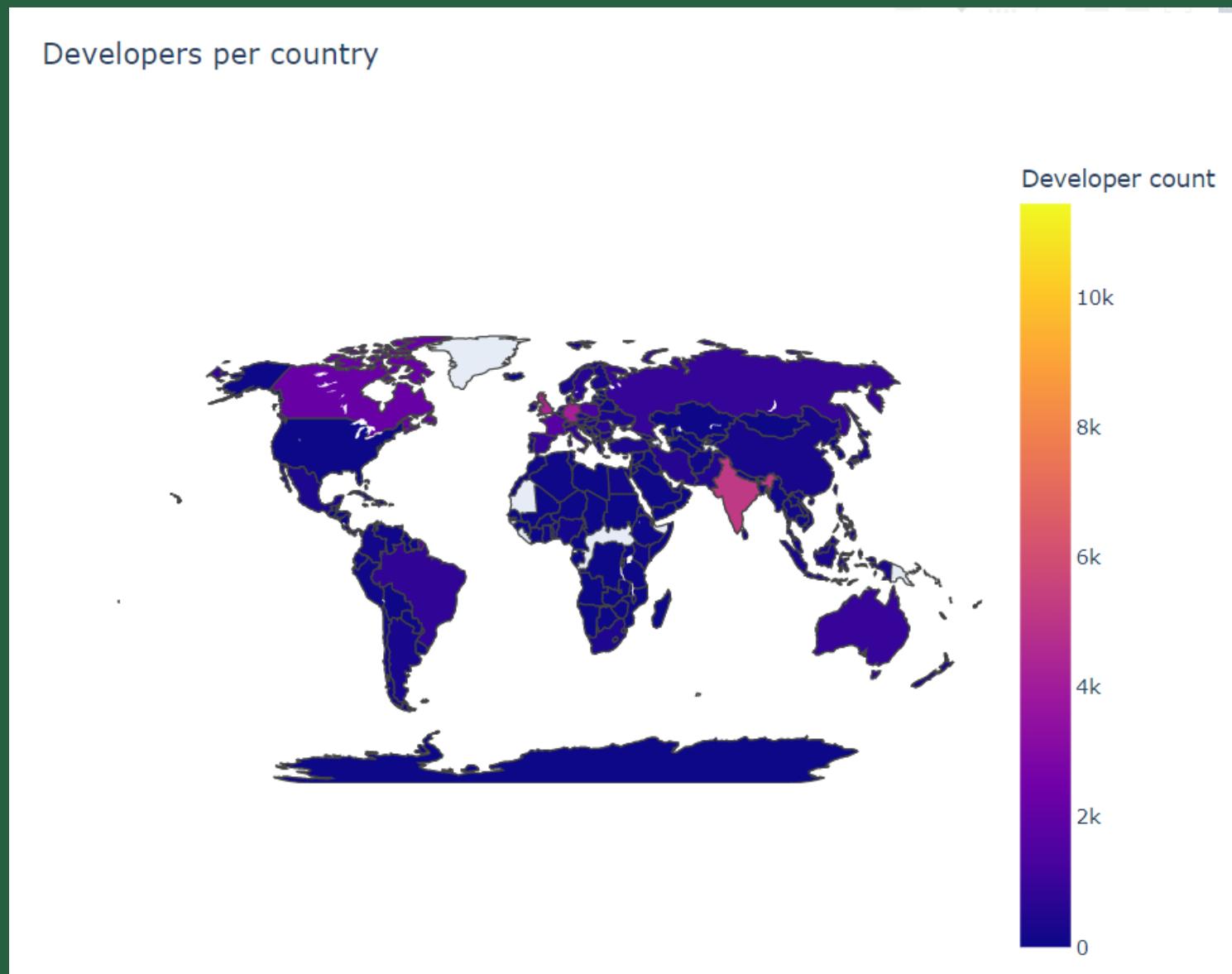
```
Out[13]: <AxesSubplot:xlabel='Country'>
```



```
In [14]: data = dict(type='choropleth',
    locations=country.index,
    locationmode='country names',
    z = country['Country'],
    text = country.index,
    colorbar = {'title':'Developer count'})
```

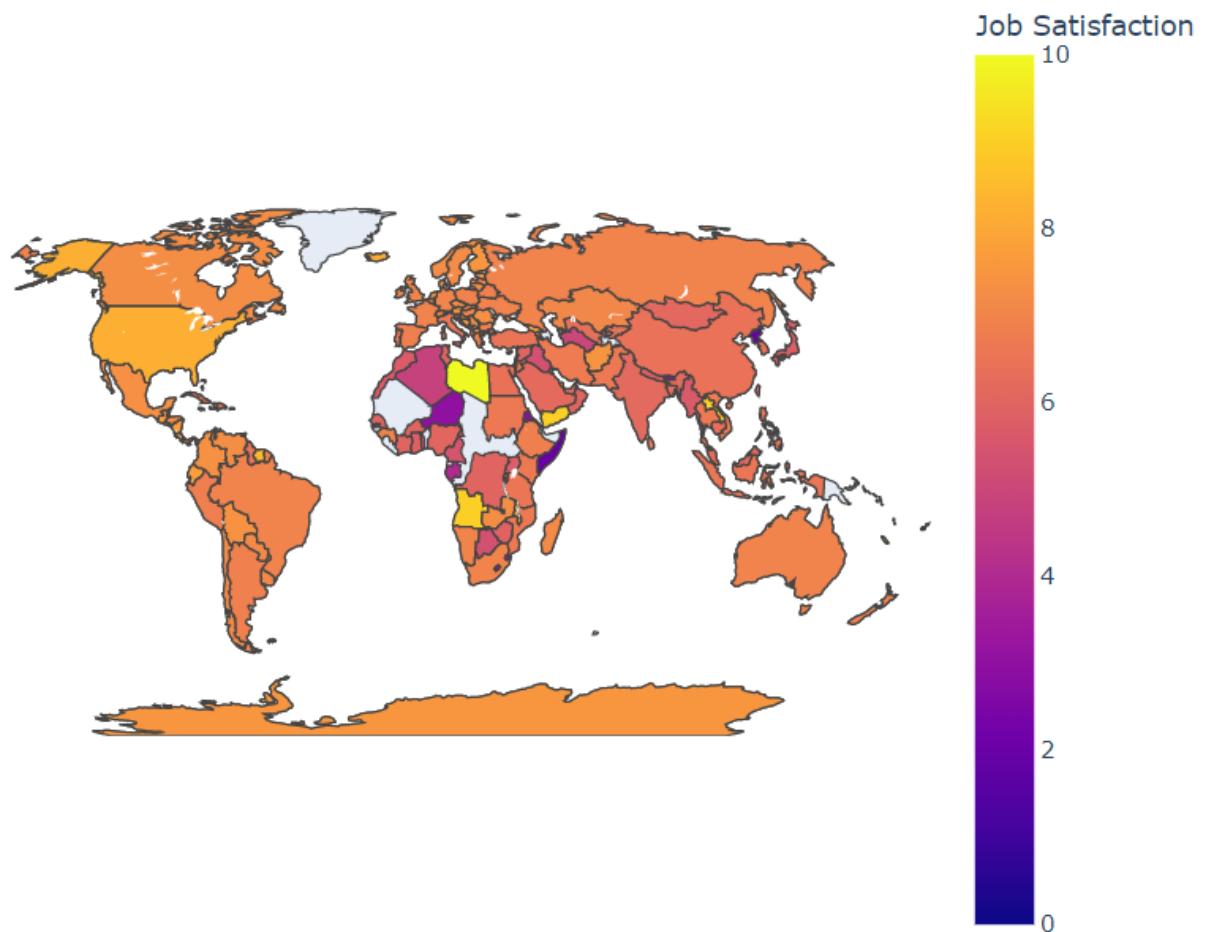
```
In [15]: layout = dict(title='Developers per country',
    geo = dict(showframe=False,
        projection={'type':'natural earth'}))
```

```
In [16]: choromap2 = go.Figure(data= [data],layout=layout)
iplot(choromap2)
```



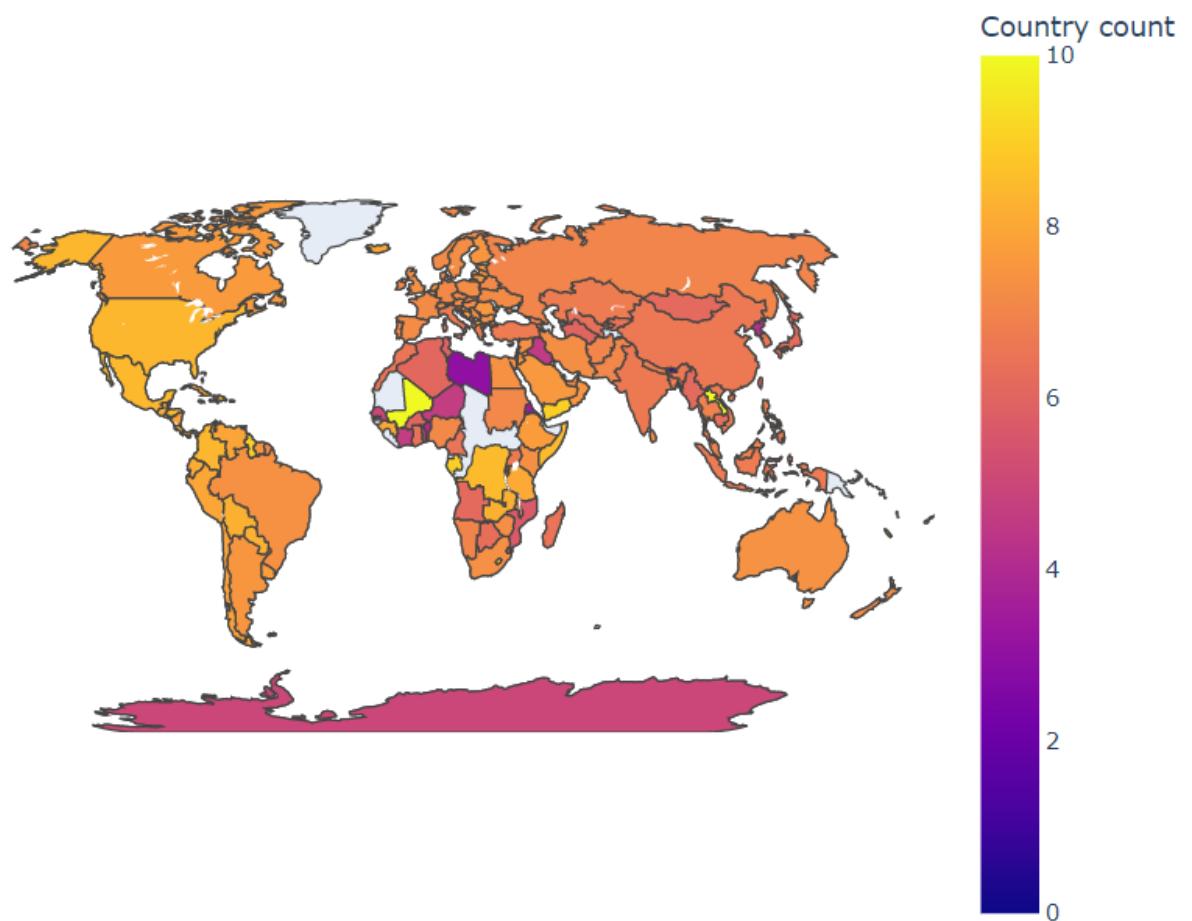
```
In [17]: data = dict(type='choropleth',
    locations=country_groupby['Salary'].mean().index,
    locationmode='country names',
    z = country_groupby['JobSatisfaction'].mean(),
    colorbar = {'title':'Job Satisfaction'})
layout = dict(title='Average Job Satisfaction per Country',
    geo = dict(showframe=False,
        projection={'type':'natural earth'}))
choromap3 = go.Figure(data= [data],layout=layout)
iplot(choromap3)
```

Average Job Satisfaction per Country



```
In [18]: data = dict(type='choropleth',
    locations=country_groupby[ 'Salary'].mean().index,
    locationmode='country names',
    z = country_groupby[ 'CareerSatisfaction'].mean(),
    colorbar = {'title':'Country count'})
layout = dict(title='Average Career Satisfaction per Country',
    geo = dict(showframe=False,
               projection={'type':'natural earth'}))
choromap3 = go.Figure(data= [data],layout=layout)
iplot(choromap3)
```

Average Career Satisfaction per Country



Top IDEs Preferred

```
In [2]: data = pd.read_csv("updatedDatasets/Update3.csv")
```

```
In [3]: dataT = data[data['IDE'].notna()]
```

```
In [4]: dataT.head(1)
```

```
Out[4]:
```

	Respondent	Professional	ProgramHobby	Country	University	EmploymentStatus	FormalEducation	MajorUndergrad	YearsCodedJob	DeveloperType	...	Wa
0	1	Student	Yes, both	United States	No	Not employed, and not looking for work	Secondary school	NaN	NaN	NaN	NaN	...

1 rows × 41 columns

[IDE] Segregation, Cleaning and Analysis

```
In [5]: A = {}
J = list()
for i in dataT['IDE']:
    x = i
    x = str(x)
    L = x.split(';')

    for k in L:
        J.append(k.strip())
    for v in J:
        if v not in A:
            A[v]=1
        else:
            A[v]=A[v]+1;
J.clear()
```

```
In [7]: D1=pd.Series(A,name='IDE Users')
```

```
In [8]: D1 =D1.to_frame()
```

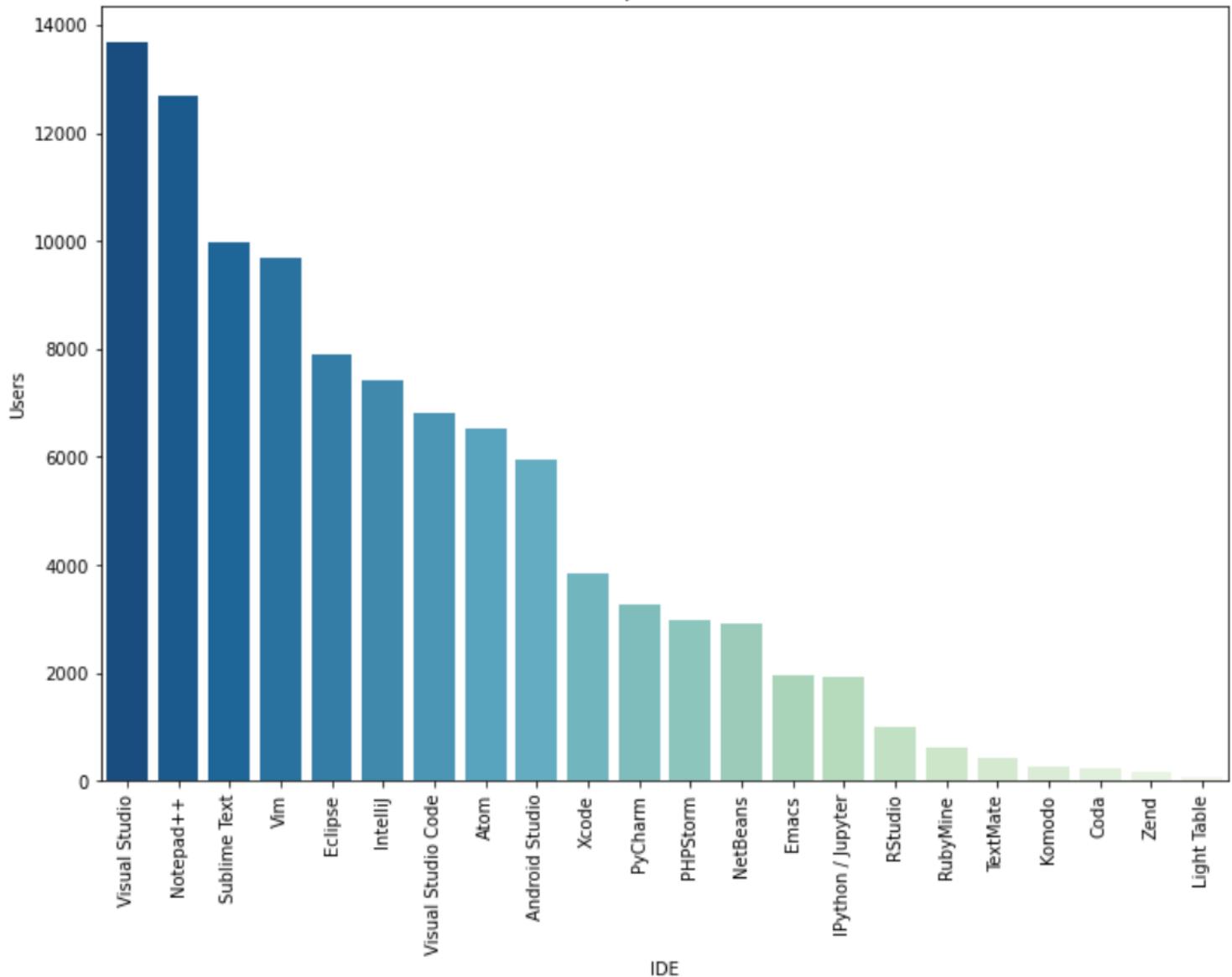
```
In [9]: D1=D1.reset_index(level=0)
```

```
In [10]: D1.columns=['IDE','Users']
```

```
In [11]: D1 = D1.sort_values(by='Users',ascending=False)
```

```
In [12]: fig, ax = plt.subplots()
fig.set_size_inches(11.7, 8.27)
sns.barplot(data=D1,x='IDE',y='Users',order=D1['IDE'],palette='GnBu_r')
plt.title('Popular IDEs')
plt.xticks(rotation=90)
plt.show()
```

Popular IDEs



Top Programming Languages

```
In [30]: dataT=data[data['HaveWorkedLanguage'].notna()]
```

```
In [31]: A = {}
J = list()
for i in dataT['HaveWorkedLanguage']:
    x = i
    x = str(x)
    L = x.split(';')

    for k in L:
        J.append(k.strip())
    for v in J:
        if v not in A:
            A[v]=1
        else:
            A[v]=A[v]+1;
J.clear()
```

```
In [16]: D2 = pd.Series(A, name = 'Languages Already Worked')
```

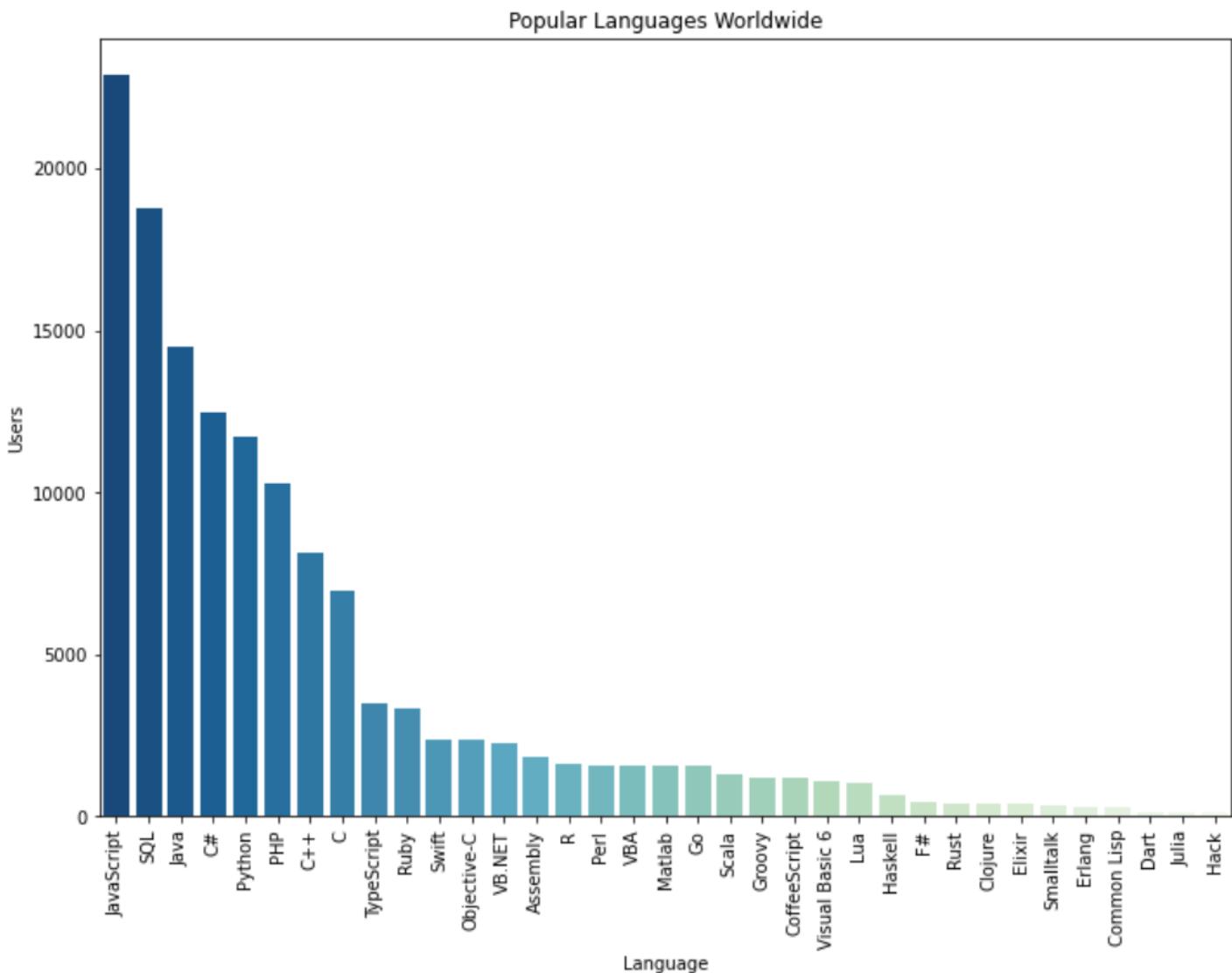
```
In [32]: D2 = D2.to_frame()
```

```
In [33]: D2 = D2.reset_index(level=0)
```

```
In [34]: D2.columns=['Language', 'Users']
```

```
In [35]: D2 = D2.sort_values(by='Users', ascending=False)
```

```
In [36]: fig, ax = plt.subplots()
fig.set_size_inches(11.7, 8.27)
sns.barplot(data=D2,x='Language',y='Users',palette='GnBu_r')
plt.xticks(rotation=90)
plt.title('Popular Languages Worldwide')
plt.show()
```



Language vs Salary

```
In [ ]: D=dict()
J=list()
for i in range(len(data)):
    x=data.iloc[i]['HaveWorkedLanguage']
    J=str(x).split('; ')
    if pd.isna(data.iloc[i]['Salary']):
        continue
    for v in J:
        if v not in D:
            D[v]=[1,data.iloc[i]['Salary']]
        else:
            D[v][0]=D[v][0]+1
            D[v][1]=D[v][1]+data.iloc[i]['Salary']
J.clear()
```

```
In [ ]: D.pop('nan')
D
```

```
In [ ]: temp = pd.DataFrame(D).transpose()
temp.columns = ['Count', 'TotalSalary']
temp.head()
```

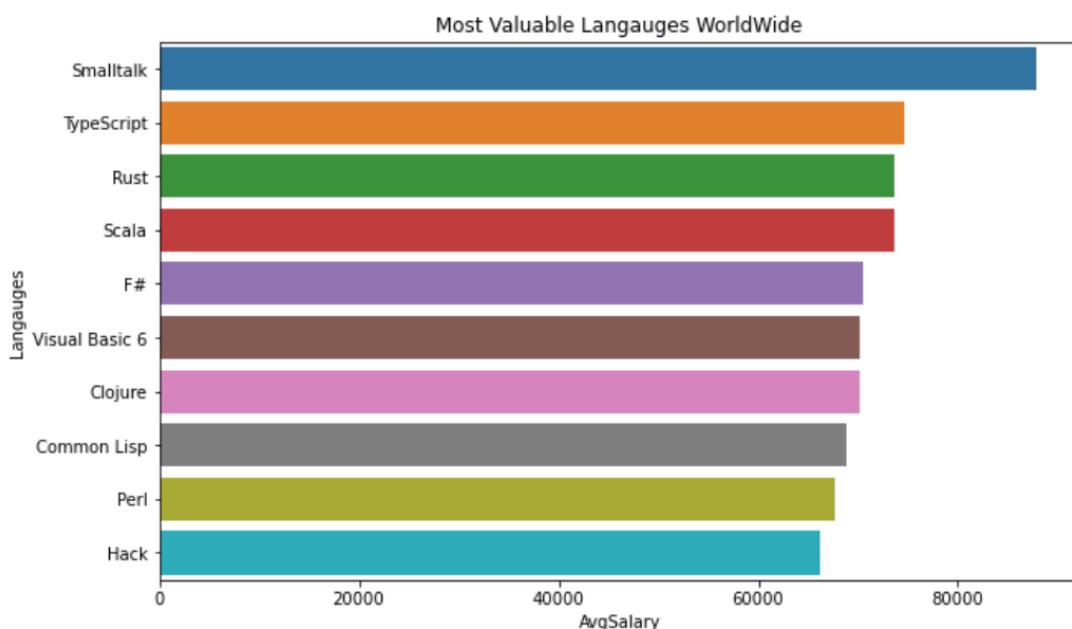
```
In [ ]: temp['AvgSalary']=temp['TotalSalary']/temp['Count']
temp.drop('TotalSalary',axis=1,inplace=True)
temp.head()
```

```
In [ ]: temp.reset_index(inplace=True)
temp.sort_values('AvgSalary',ascending=False).head()
```

```
In [42]: temp.reset_index(inplace=True)
temp.sort_values('AvgSalary',ascending=False).head()
```

```
Out[42]:
   index  Count      AvgSalary
29  Smalltalk  62.0  87855.410630
34  TypeScript  7.0   74606.885120
23     Rust    4.0   73693.728036
19     Scala   46.0   73582.858016
32       F#   23.0   70469.940540
```

```
In [43]: plt.figure(figsize=(10,6))
s.barplot(y='index',x='AvgSalary',data=temp.sort_values('AvgSalary',ascending=False).head(10))
plt.title('Most Valuable Languages Worldwide')
plt.ylabel('Languages')
plt.show()
```



Popular Databases

```
In [49]: dataT = data[data['HaveWorkedDatabase'].notna()]
```

```
In [50]: A = {}
J = list()
for i in dataT['HaveWorkedDatabase']:
    x = i
    x = str(x)
    L = x.split(';')

    for k in L:
        J.append(k.strip())
    for v in J:
        if v not in A:
            A[v]=1
        else:
            A[v]=A[v]+1;
    J.clear()
```

```
In [51]: A
```

```
Out[51]: {'MySQL': 16375,
 'SQLite': 7838,
 'MongoDB': 6192,
 'Redis': 4143,
 'SQL Server': 11358,
 'PostgreSQL': 7815,
 'Oracle': 4874,
 'Cassandra': 906}
```

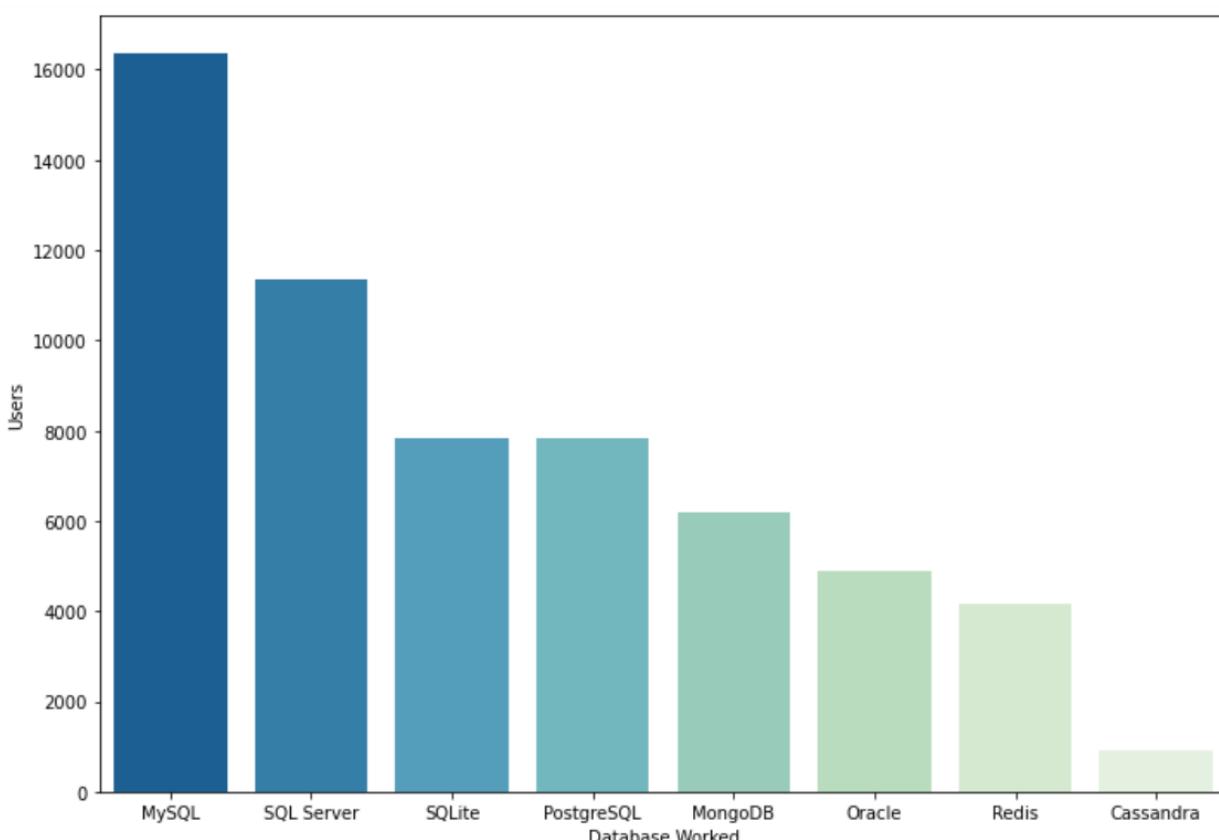
```
In [52]: D4= pd.Series(A,name="Have Worked Database")
```

```
In [54]: D4 = D4.to_frame()
```

```
In [55]: D4 =D4.reset_index(level=0)
```

```
In [56]: D4.columns=['Database Worked','Users']
```

```
In [57]: D4 = D4.sort_values(by='Users',ascending=False)
fig,ax = plt.subplots()
fig.set_size_inches(11.7, 8.27)
s.barplot(data=D4,x='Database Worked',y ='Users',palette='GnBu_r')
```



Popular Platforms

```
In [58]: dataT = data[data['HaveWorkedPlatform'].notna()]
```

```
In [59]: A = {}
J = list()
for i in dataT['HaveWorkedPlatform']:
    x = i
    x = str(x)
    L = x.split(';')

    for k in L:
        J.append(k.strip())
for v in J:
    if v not in A:
        A[v]=1
    else:
        A[v]=A[v]+1;
J.clear()
```

```
In [63]: D6 = pd.Series(A,name ='Have Worked Platform')
```

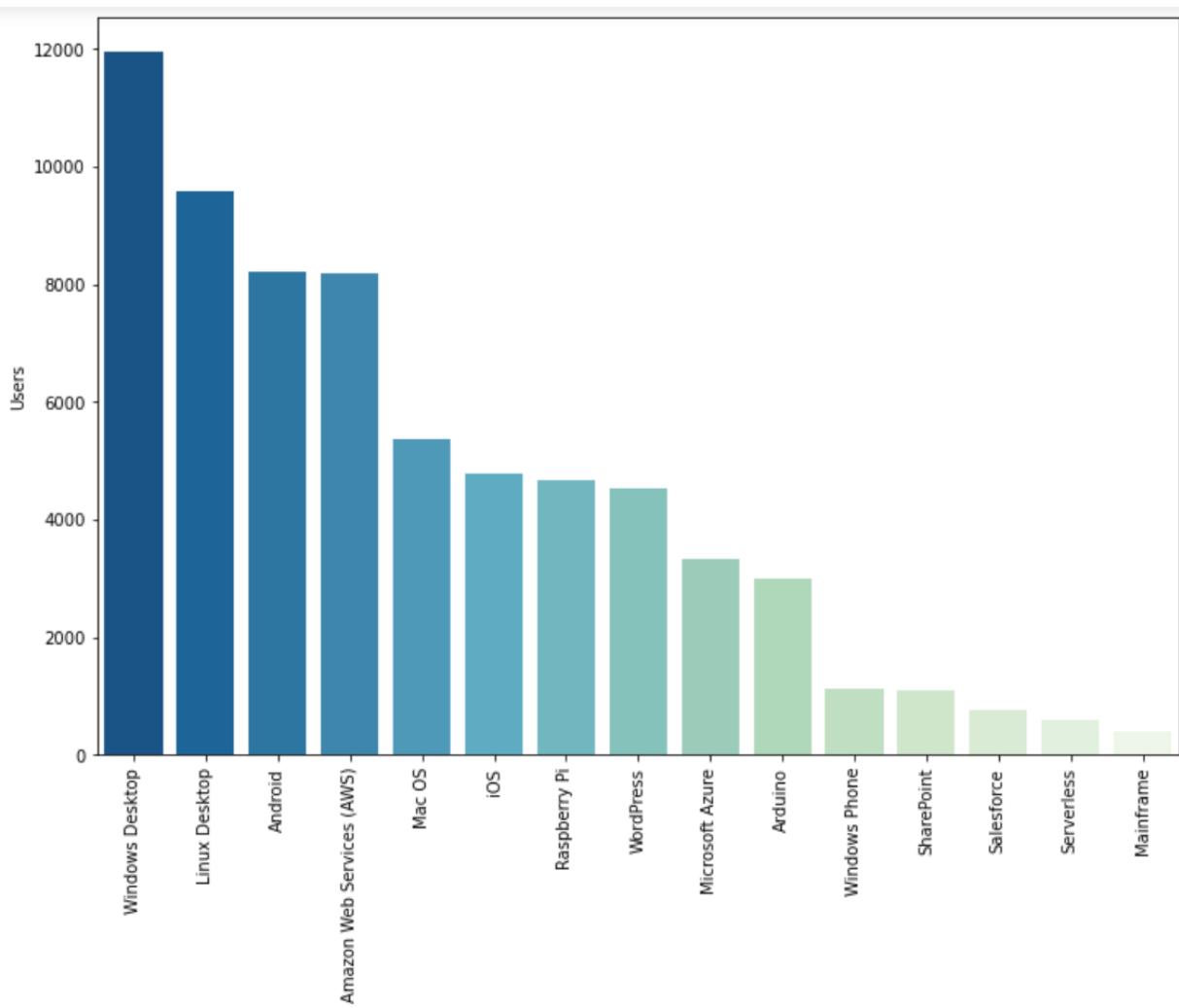
```
In [65]: D6=D6.to_frame()
```

```
In [66]: D6=D6.reset_index(level=0)
```

```
In [67]: D6.columns=['Platform Worked','Users']
```

```
In [68]: D6=D6.sort_values(by='Users',ascending=False)
```

```
In [69]: fig,ax = plt.subplots()
fig.set_size_inches(11.8,8.27)
s.barplot(data=D6,x='Platform Worked',y='Users',palette='GnBu_r')
plt.xticks(rotation=90)
```



Important Benefits

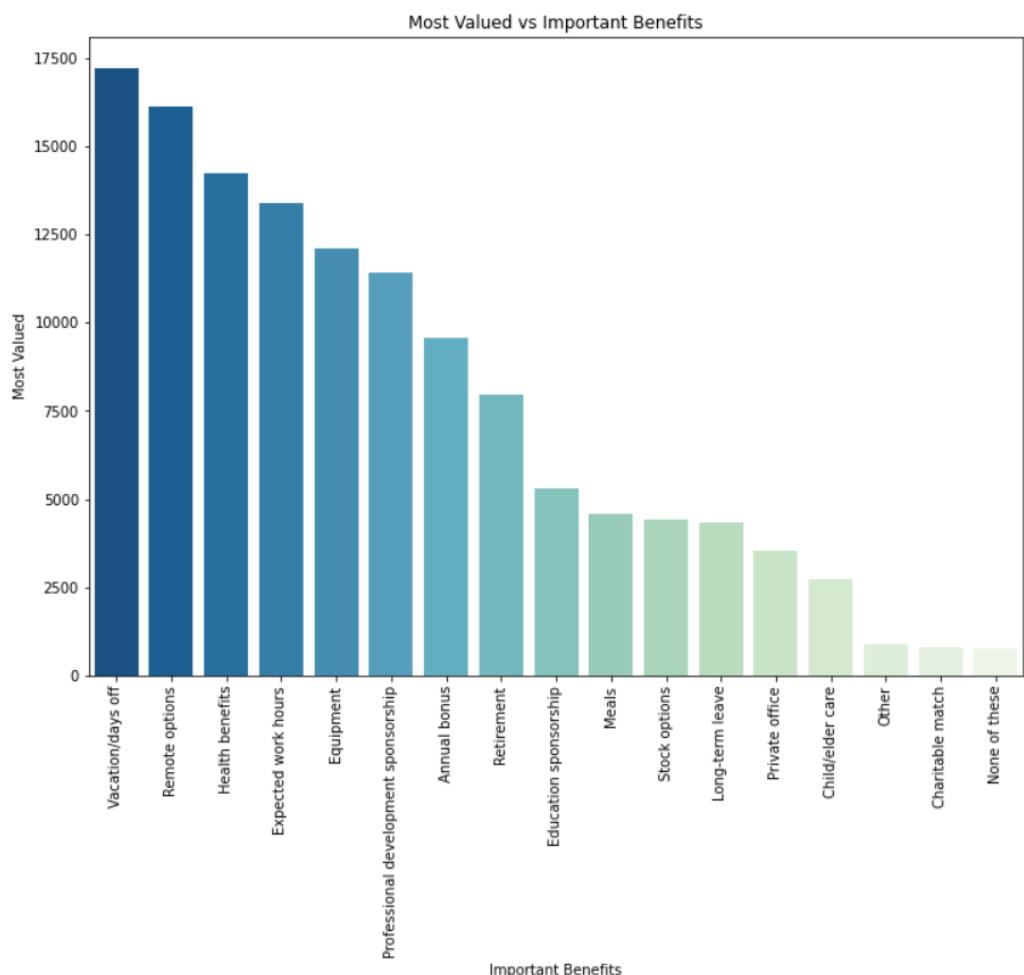
```
In [2]: data = pd.read_csv("updatedDatasets/Update3.csv")
```

ImportantBenefits Analysis

```
In [3]: A=dict()
J=list()
for i in data['ImportantBenefits']:
    x=i
    x=str(x)
    J=x.split('; ')
    for v in J:
        if v not in A:
            A[v]=1
        else:
            A[v]=A[v]+1
J.clear()
```

```
In [4]: D8 = pd.Series(A,name="Important Benefits")
D8 = D8.drop('nan')
D8 = D8.to_frame()
D8 = D8.reset_index(level=0)
D8.columns=['Important Benefits','Most Valued']
D8 = D8.sort_values(by='Most Valued',ascending=False)
```

```
In [5]: fig,ax = plt.subplots()
fig.set_size_inches(11.8,8.27)
s.barplot(data=D8,x='Important Benefits',y='Most Valued',palette='GnBu_r')
plt.xticks(rotation=90)
plt.title('Most Valued vs Important Benefits')
```

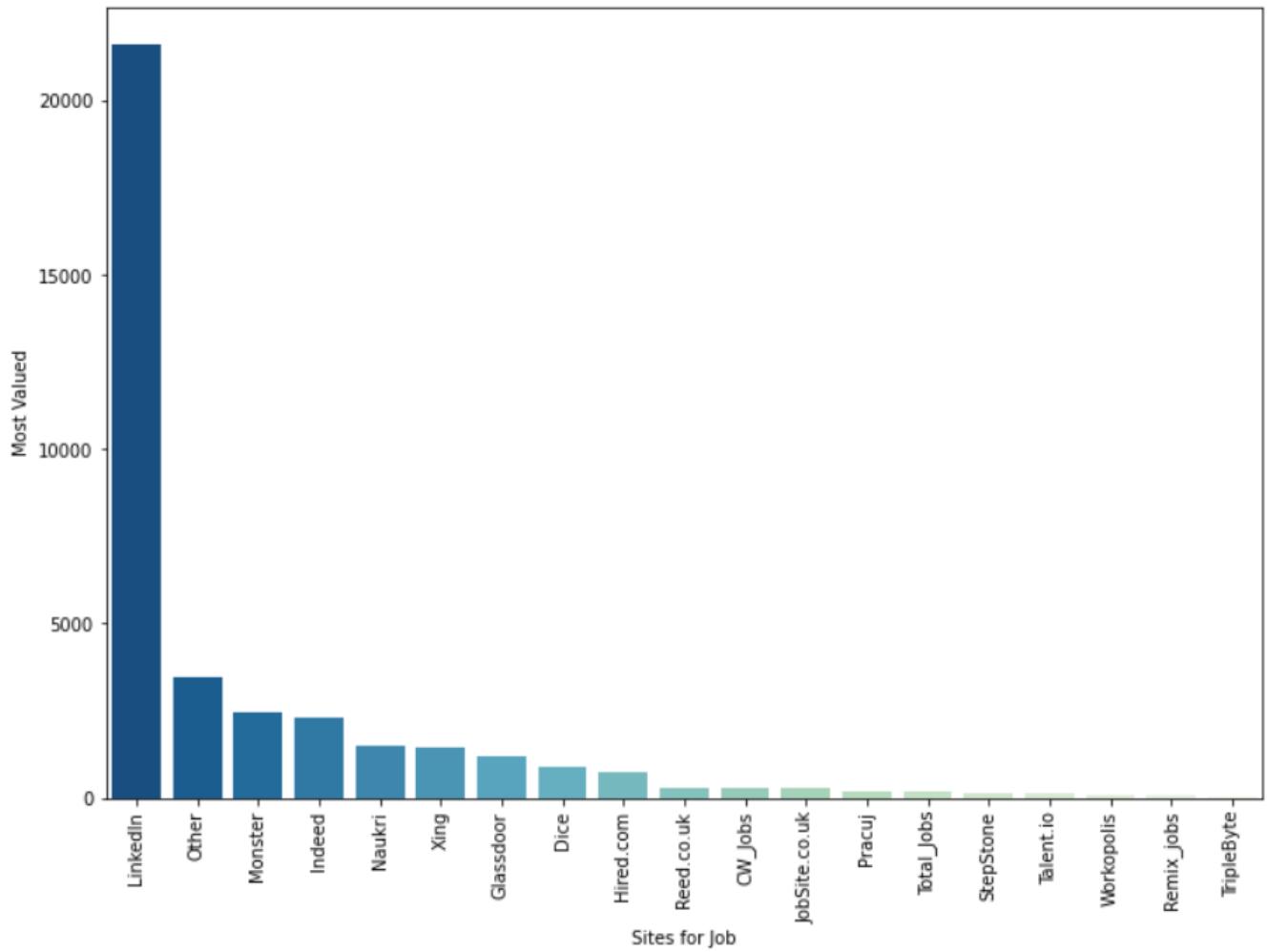


Job Profile

```
In [6]: A=dict()
J=list()
for i in data['JobProfile']:
    x=i
    x=str(x)
    J=x.split('; ')
    for v in J:
        if v not in A:
            A[v]=1
        else:
            A[v]=A[v]+1
J.clear()
```

```
In [7]: D9 = pd.Series(A,name="Job Profile Sites")
D9 = D9.drop('nan')
D9 = D9.to_frame()
D9 = D9.reset_index(level=0)
D9.columns=['Sites for Job','Most Valued']
D9 = D9.sort_values(by='Most Valued',ascending=False)
```

```
In [8]: fig,ax = plt.subplots()
fig.set_size_inches(11.8,8.27)
s.barplot(data=D9,x='Sites for Job',y='Most Valued',palette='GnBu_r')
plt.xticks(rotation=90)
plt.title('Sites For Job vs Most valued')
```

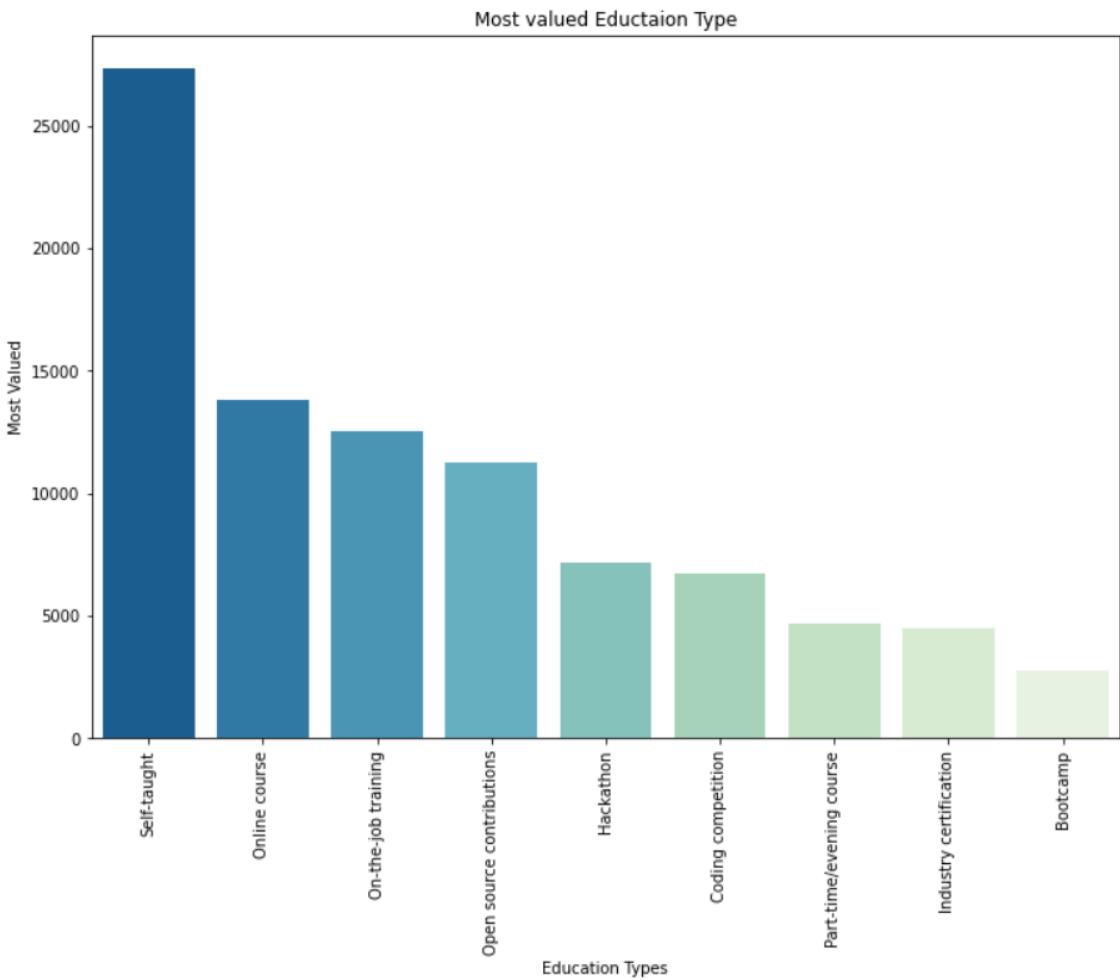


Education Types

```
In [9]: A=dict()
J=list()
for i in data['EducationTypes']:
    x=i
    x=str(x)
    J=x.split('; ')
    for v in J:
        if v not in A:
            A[v]=1
        else:
            A[v]=A[v]+1
J.clear()
```

```
In [10]: D10 = pd.Series(A,name="Education Types")
D10 = D10.drop('nan')
D10 = D10.to_frame()
D10 = D10.reset_index(level=0)
D10.columns=['Education Types','Most Valued']
D10 = D10.sort_values(by='Most Valued',ascending=False)
```

```
In [11]: fig,ax = plt.subplots()
fig.set_size_inches(11.8,8.27)
s.barplot(data=D10, x='Education Types' ,y='Most Valued' ,palette='GnBu_r')
plt.xticks(rotation=90)
plt.title('Most valued Education Type')
```

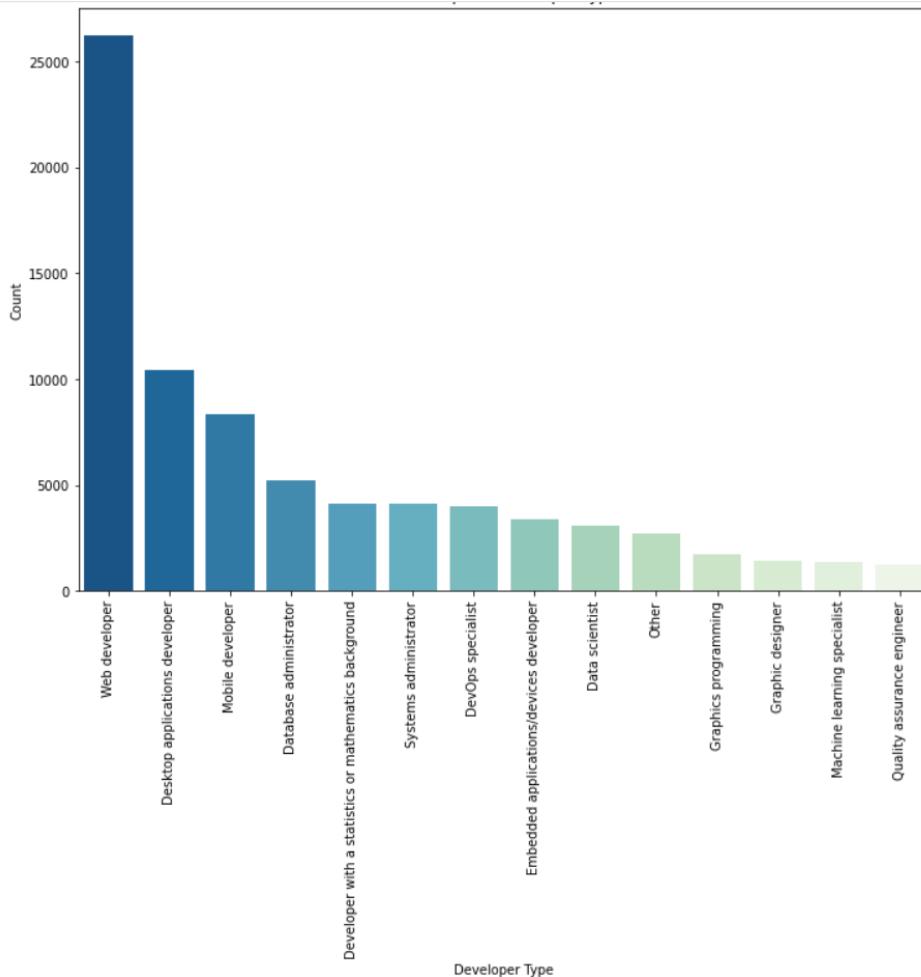


Developer Type

```
In [12]: A=dict()
J=list()
for i in data['DeveloperType']:
    x=i
    x=str(x)
    J=x.split('; ')
    for v in J:
        if v not in A:
            A[v]=1
        else:
            A[v]=A[v]+1
J.clear()
```

```
In [13]: D11=pd.Series(A,name='Developer Type')
D11=D11.drop('nan')
D11=D11.to_frame()
D11=D11.reset_index(level=0)
D11.columns=['Developer Type','Count']
D11=D11.sort_values(by='Count',ascending=False)
```

```
In [14]: fig,ax = plt.subplots()
fig.set_size_inches(11.8,8.27)
s.barplot(data=D11, x='Developer Type' ,y='Count' ,palette='GnBu_r')
plt.xticks(rotation=90)
plt.title('Most Popular Developer Type')
```



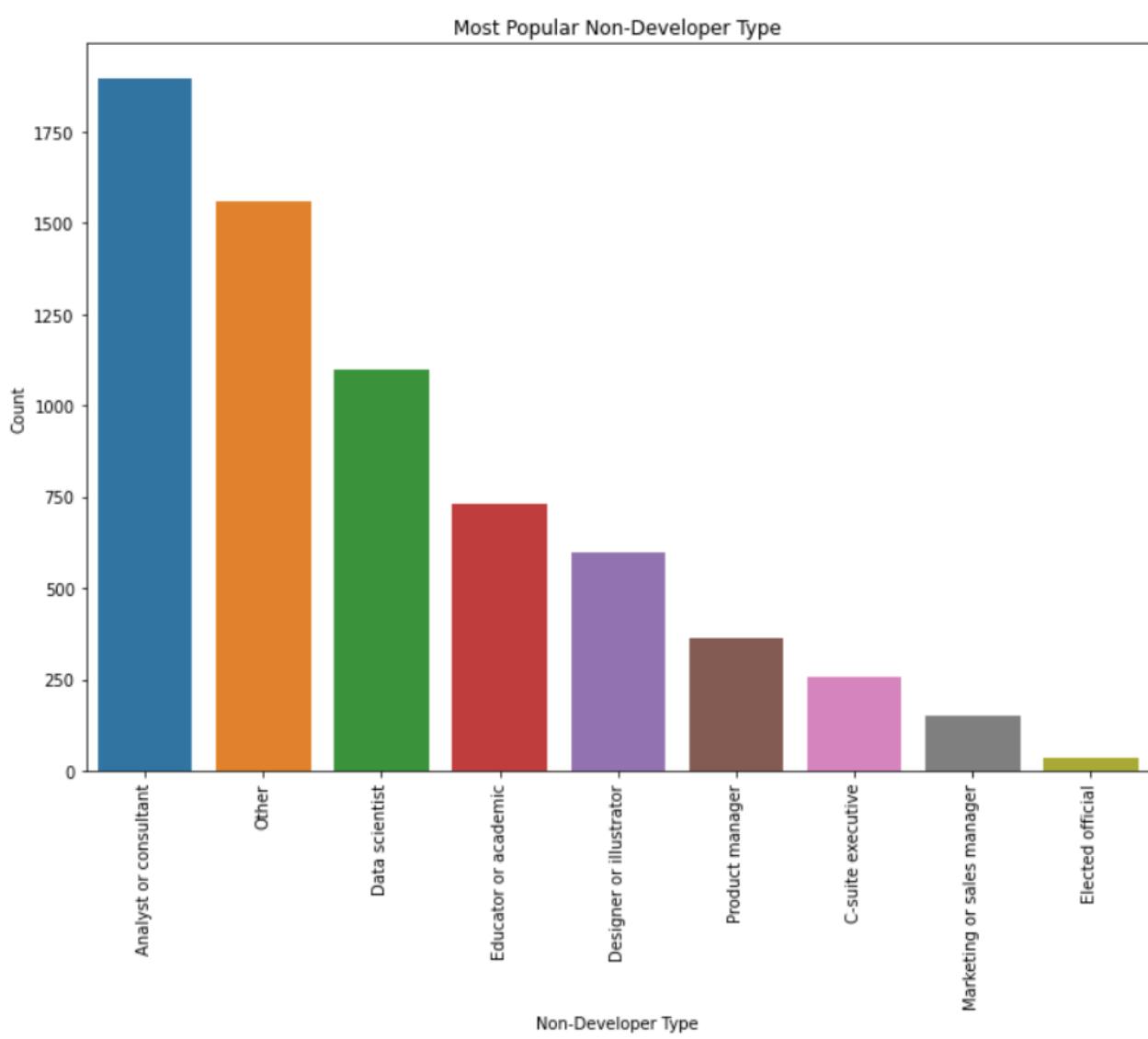
Non-Developer

```
In [15]: A=dict()
J=list()
for i in data[ 'NonDeveloperType' ]:
    x=i
    x=str(x)
    J=x.split('; ')
    for v in J:
        if v not in A:
            A[v]=1
        else:
            A[v]=A[v]+1
    J.clear()
A
```

```
Out[15]: {'.': 1,
           'nan': 46501,
           'Data scientist': 1100,
           'Other': 1558,
           'C-suite executive': 257,
           'Product manager': 365,
           'Educator or academic': 732,
           'Analyst or consultant': 1897,
           'Marketing or sales manager': 152,
           'Designer or illustrator': 600,
           'Elected official': 34}
```

```
In [16]: D12=pd.Series(A,name='Non-Developer Type')
D12=D12.drop('nan')
D12=D12.drop('.')
D12=D12.to_frame()
D12=D12.reset_index(level=0)
D12.columns=['Non-Developer Type','Count']
D12=D12.sort_values(by='Count',ascending=False)
```

```
In [17]: fig,ax=plt.subplots()
fig.set_size_inches(11.8,8.27)
s.barplot(data=D12,x='Non-Developer Type',y='Count')
D12=D12.sort_values(by='Count',ascending=False)
plt.xticks(rotation=90)
plt.title('Most Popular Non-Developer Type')
```



Language Recommendation

```
In [3]: A = {}
J = list()
for i in df['HaveWorkedLanguage']:
    x = i
    x = str(x)
    L = x.split(';')

    for k in L:
        J.append(k.strip())
    for v in J:
        if v not in A:
            A[v]=1
        else:
            A[v]=A[v]+1
    J.clear()
```

```
In [4]: A.pop('nan')
A
```

```
Out[4]: {'Swift': 2368,
 'JavaScript': 22875,
 'Python': 11704,
 'Ruby': 3324,
 'SQL': 18754,
 'Java': 14524,
```

```
In [5]: temp = []
for x in df['HaveWorkedLanguage']:
    if pd.isna(x)== False:
        temp1 = []
        J = []
        L = x.split(';')
        for k in L:
            J.append(k.strip())
        for y in list(A.keys()):
            if (y in J):
                temp1.append(1)
            else:
                temp1.append(0)
        temp.append(temp1)
```

```
In [6]: ab = pd.DataFrame(data=temp,columns=list(A.keys()))
ab.head()
```

```
Out[6]: Swift JavaScript Python Ruby SQL Java PHP Matlab R Rust ... Lua VBA Groovy Go Smalltalk Visual Basic 6 Common Lisp Dart Julia Hack
0 1 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 1 1 1 1 0 0 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2 0 0 1 0 0 1 1 0 0 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3 0 0 1 0 1 0 0 1 1 0 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
4 0 1 0 0 0 0 1 0 0 1 ... 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

5 rows × 35 columns

```
In [7]: python = ab['Python']
python.head()
```

```
Out[7]: 0 0
1 1
2 1
3 1
4 0
Name: Python, dtype: int64
```

```
In [8]: similar_to_python = ab.corrwith(python)
corr_python = pd.DataFrame(similar_to_python,columns=['correlation'])
corr_python.dropna(inplace=True)
corr_python.sort_values('correlation',ascending=False).head(10)
```

```
Out[8]: correlation
```

Python	1.000000
C	0.184977
C++	0.169447
R	0.142893
Matlab	0.136947
Assembly	0.109696
Go	0.108960
Lua	0.100857
Perl	0.090786
Haskell	0.078823

```
In [9]: ab2 = pd.DataFrame(list(A.items()),columns=['Language','Count'])
ab2.head()
```

```
Out[9]: Language Count
```

0	Swift	2368
1	JavaScript	22875
2	Python	11704
3	Ruby	3324
4	SQL	18754

```
In [10]: corr_python = pd.merge(corr_python,ab2, left_index=True,right_on='Language')
corr_python.head()
```

```
Out[10]: correlation Language Count
```

0	0.017209	Swift	2368
1	-0.041113	JavaScript	22875
2	1.000000	Python	11704
3	0.069663	Ruby	3324
4	-0.025430	SQL	18754

```
In [11]: corr_python.set_index('Language').head()
```

```
Out[11]: correlation Count
```

Language	correlation	Count
Swift	0.017209	2368
JavaScript	-0.041113	22875
Python	1.000000	11704
Ruby	0.069663	3324
SQL	-0.025430	18754

```
In [12]: corr_python[corr_python['Count']>1000].sort_values('correlation',ascending=False).head(10)
```

```
Out[12]: correlation Language Count
```

2	1.000000	Python	11704
17	0.184977	C	6974
18	0.169447	C++	8155
8	0.142893	R	1634
7	0.136947	Matlab	1569
19	0.109696	Assembly	1823

Database Recommendation

```
In [13]: A = {}
J = list()
for i in df['HaveWorkedDatabase']:
    x = i
    x = str(x)
    L = x.split(';')

    for k in L:
        J.append(k.strip())
    for v in J:
        if v not in A:
            A[v]=1
        else:
            A[v]=A[v]+1;
    J.clear()
```

```
In [14]: A.pop('nan')
A
```

```
Out[14]: {'MySQL': 16375,
 'SQLite': 7838,
 'MongoDB': 6192,
 'Redis': 4143,
 'SQL Server': 11358,
 'PostgreSQL': 7815,
 'Oracle': 4874}
```

```
In [15]: temp = []
for x in df['HaveWorkedDatabase']:
    if pd.isna(x)== False:
        temp1 = []
        J = []
        L = x.split(';')
        for k in L:
            J.append(k.strip())
        for y in list(A.keys()):
            if (y in J):
                temp1.append(1)
            else:
                temp1.append(0)
        temp.append(temp1)
```

```
In [16]: ab = pd.DataFrame(data=temp,columns=list(A.keys()))
ab.head()
```

```
Out[16]: MySQL  SQLite  MongoDB  Redis  SQL Server  PostgreSQL  Oracle  Cassandra
0         1         1         0         0         0         0         0         0
1         1         0         0         0         0         0         0         0
2         1         1         1         1         1         0         0         0
3         1         0         0         0         0         0         0         0
4         0         1         0         0         0         0         0         0
```

```
In [17]: mysql = ab['MySQL']
mysql.head()
```



```
Out[17]: 0    1
1    1
2    1
3    1
4    0
Name: MySQL, dtype: int64
```

```
In [18]: similar_to_mysql = ab.corrwith(mysql)
corr_mysql = pd.DataFrame(similar_to_mysql,columns=['correlation'])
corr_mysql.dropna(inplace=True)
corr_mysql.sort_values('correlation',ascending=False).head(3)
```

```
Out[18]:      correlation
MySQL    1.000000
SQLite   0.075484
Redis    0.040005
```

```
In [19]: ab2 = pd.DataFrame(list(A.items()),columns=['Databases','Count'])
ab2.head()
```

```
Out[19]:
```

	Databases	Count
0	MySQL	16375
1	SQLite	7838
2	MongoDB	6192
3	Redis	4143
4	SQL Server	11358

```
In [20]: corr_mysql = pd.merge(corr_mysql,ab2, left_index=True,right_on='Databases')
corr_mysql.head()
```

```
Out[20]:
```

	correlation	Databases	Count
0	1.000000	MySQL	16375
1	0.075484	SQLite	7838
2	0.038457	MongoDB	6192
3	0.040005	Redis	4143
4	-0.205809	SQL Server	11358

```
In [21]: corr_mysql.set_index('Databases').head()
```

```
Out[21]:
```

Databases	correlation	Count
MySQL	1.000000	16375
SQLite	0.075484	7838
MongoDB	0.038457	6192
Redis	0.040005	4143
SQL Server	-0.205809	11358

```
In [22]: corr_mysql[corr_mysql['Count']>2000].sort_values('correlation',ascending=False).head(3)
```

```
Out[22]:
```

	correlation	Databases	Count
0	1.000000	MySQL	16375
1	0.075484	SQLite	7838
3	0.040005	Redis	4143

Auditory Environment

```
In [3]: df['AuditoryEnvironment'].unique()
```

```
Out[3]: array(['Turn on some music',
   'Put on some ambient sounds (e.g. whale songs, forest sounds)',
   'nan', 'Keep the room absolutely quiet', 'Something else',
   'Put on a movie or TV show', 'Turn on the news or talk radio'],
  dtype=object)
```

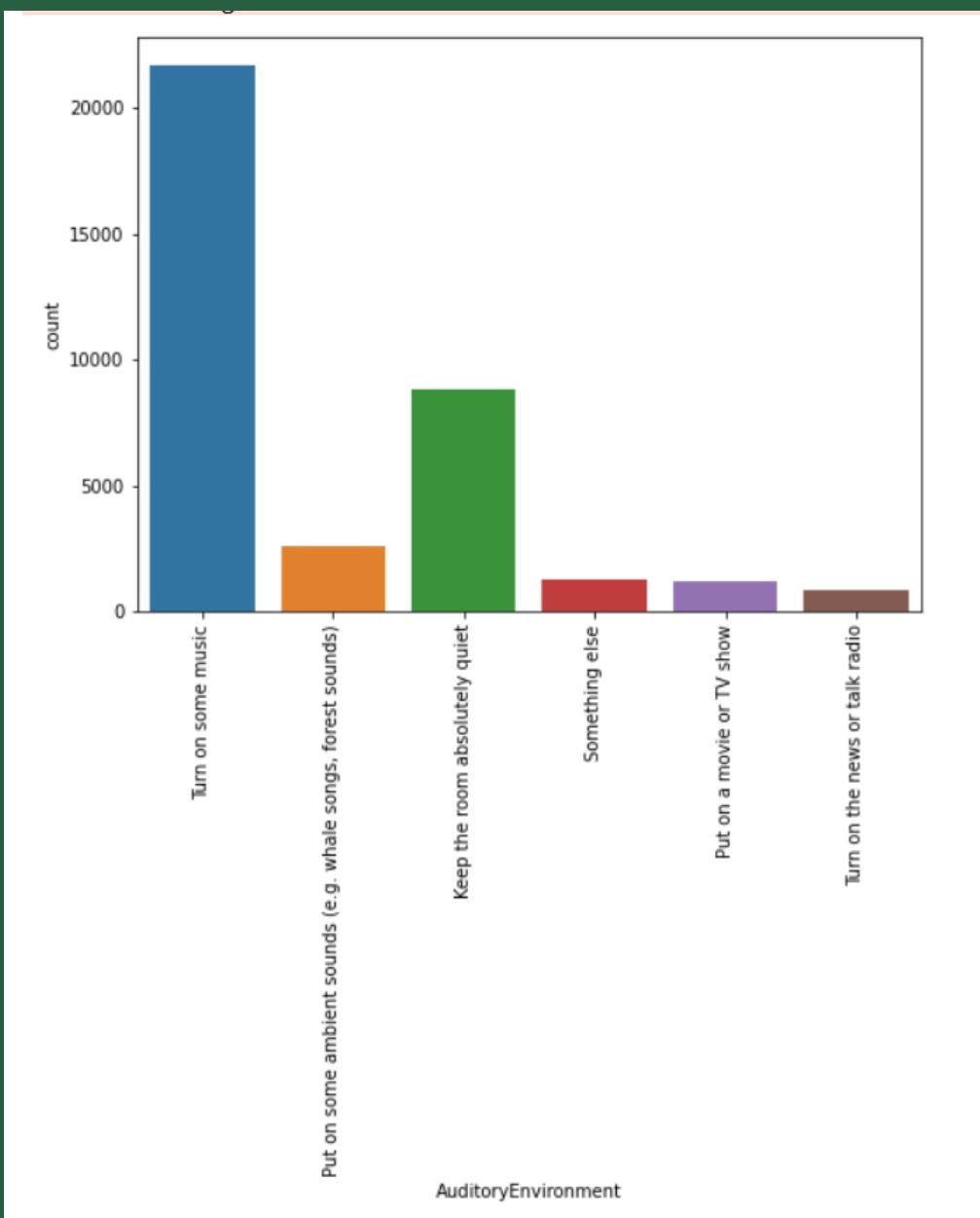
```
In [4]: df['AuditoryEnvironment'].nunique()
```

```
Out[4]: 6
```

```
In [5]: df['AuditoryEnvironment'].count()
```

```
Out[5]: 36457
```

```
In [6]: plt.figure(figsize = (8,6))
sns.countplot(df['AuditoryEnvironment'] )
plt.xticks(rotation=90)
plt.show()
```



```
In [7]: df.iloc[:,13:20].head()
```

```
Out[7]:
```

	ProblemSolving	LearningNewTech	JobSecurity	DiversityImportant	SeriousWork	WorkPayCare	ChallengeMyself
0	Strongly agree	Agree	Strongly agree	Agree	Strongly agree	Strongly disagree	Agree
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	Strongly agree	Strongly agree	Agree	Strongly agree	Agree	Disagree	Agree
3	Strongly agree	Strongly agree	Somewhat agree	Agree	Strongly agree	Disagree	Strongly agree
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
In [8]: #ProblemSolving: I Love solving problems
```

```
#LearningNewTech: Learning new technologies is fun
```

```
#JobSecurity: Job security is important to me
```

```
#DiversityImportant: Diversity in the workplace is important
```

```
#SeriousWork:I take my work very seriously
```

```
#WorkPayCare: I don't really care what I work on, so long as I'm paid well
```

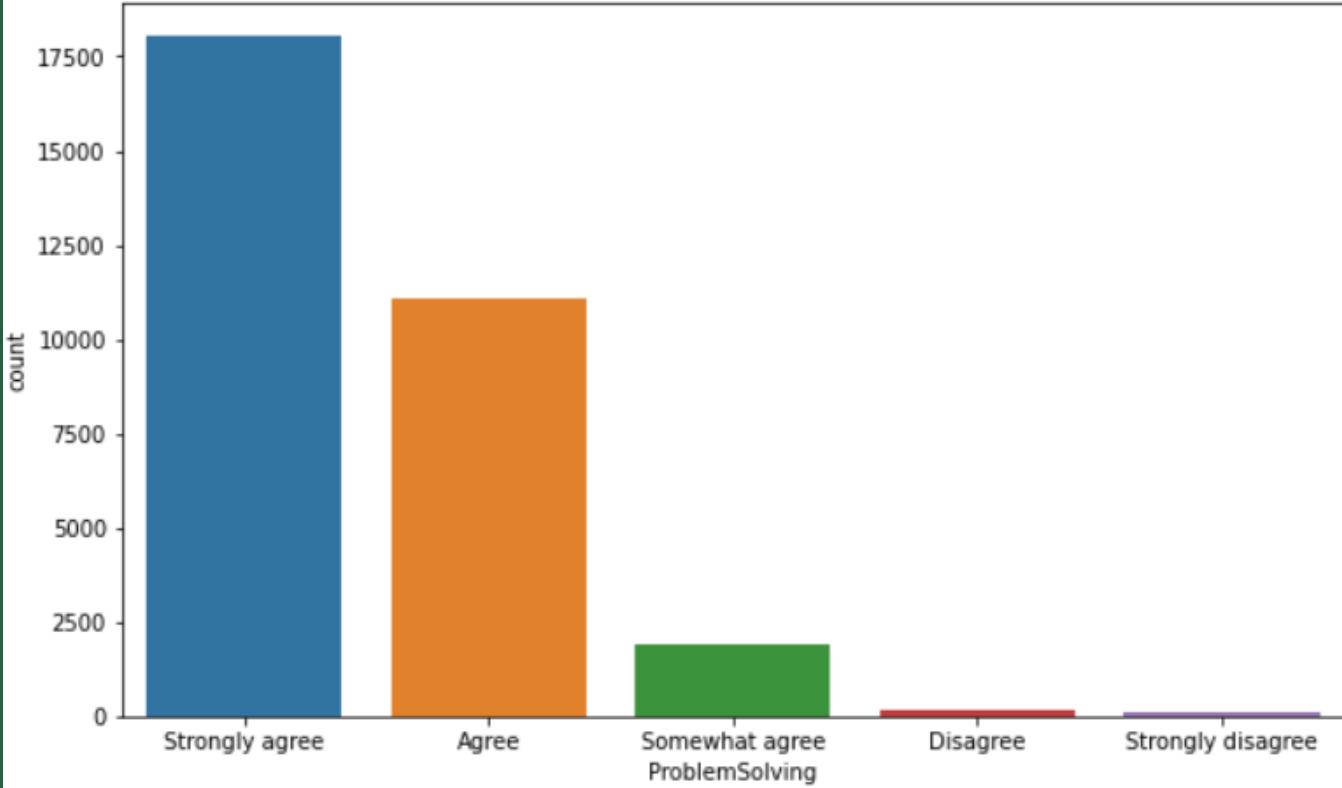
```
#ChallengeMyself: I like to challenge myself
```

```
In [9]: plt.figure(figsize=(10,6))
```

```
sns.countplot(x='ProblemSolving',data =df)
```

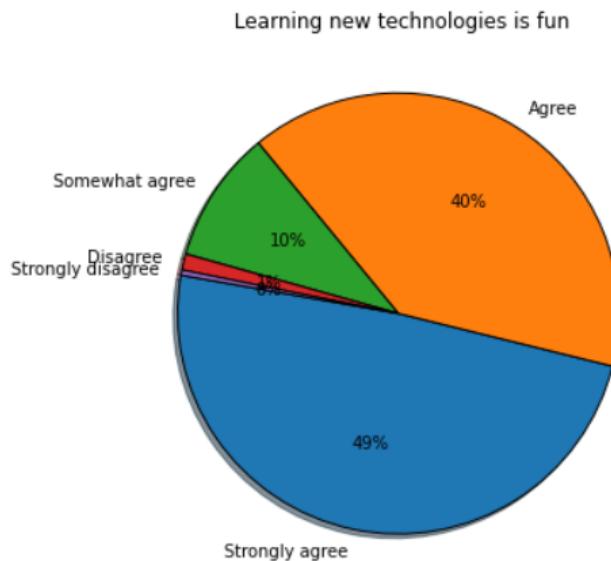
```
plt.title('I love solving problems')
```

I love solving problems



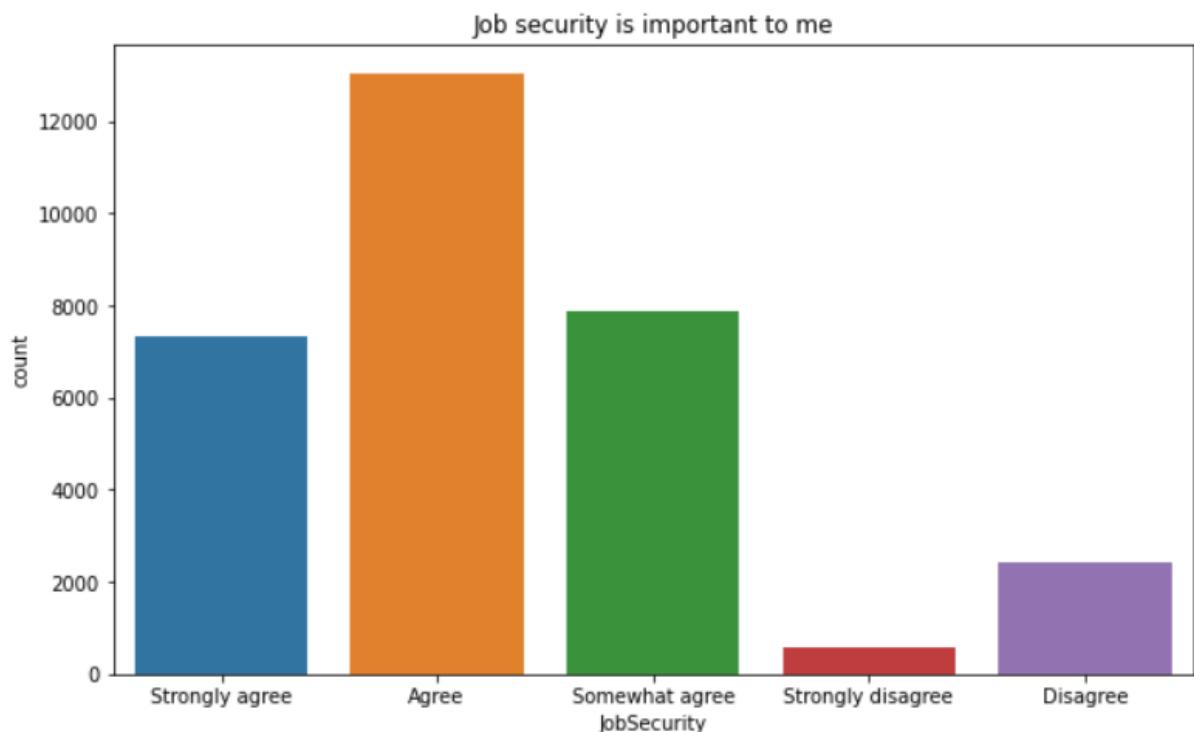
```
In [10]: plt.figure(figsize=(10,6))
plt.pie(df['LearningNewTech'].value_counts(), labels= df['LearningNewTech'].value_counts().index,
        startangle=170, wedgeprops={'edgecolor': 'black'}, autopct='%.1f%%', shadow=True)
plt.title(" Learning new technologies is fun")
```

```
Out[10]: Text(0.5, 1.0, ' Learning new technologies is fun')
```



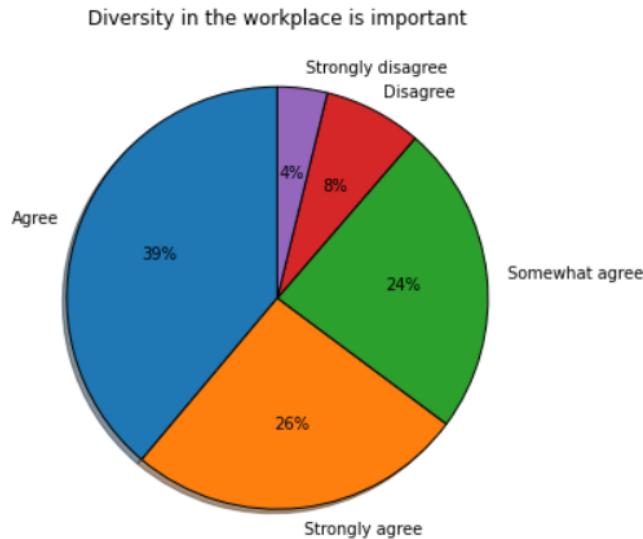
```
In [11]: plt.figure(figsize=(10,6))
sns.countplot(x='JobSecurity',data =df)
plt.title('Job security is important to me')
```

```
Out[11]: Text(0.5, 1.0, 'Job security is important to me')
```



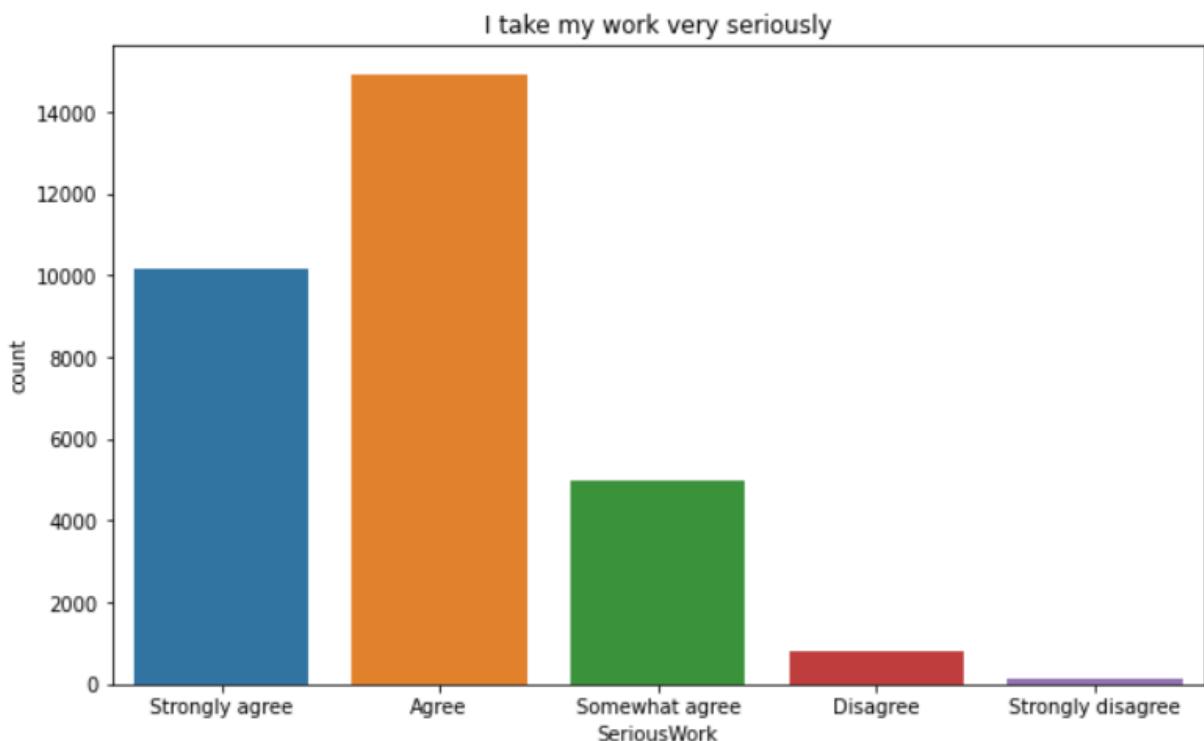
```
In [12]: plt.figure(figsize=(10,6))
plt.pie(df['DiversityImportant'].value_counts(), labels= df['DiversityImportant'].value_counts().index,
        startangle=90, wedgeprops={'edgecolor': 'black'}, autopct='%.1f%%', shadow=True)
plt.title("Diversity in the workplace is important")
```

```
Out[12]: Text(0.5, 1.0, 'Diversity in the workplace is important')
```



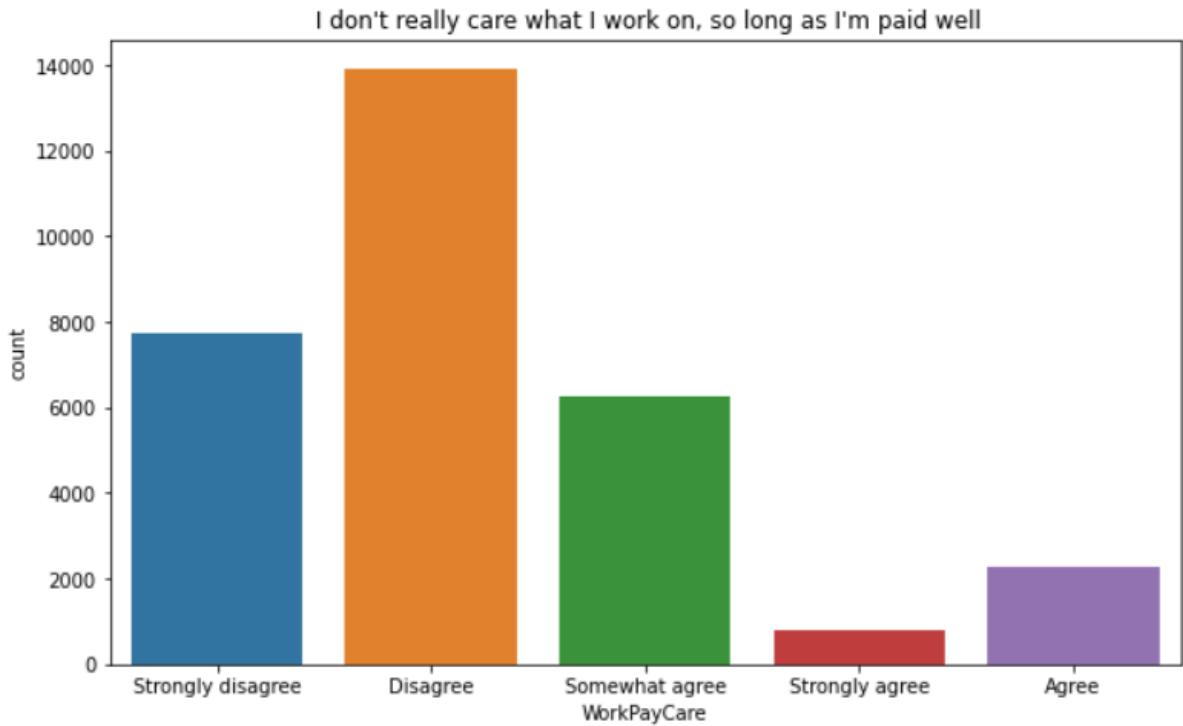
```
In [13]: plt.figure(figsize=(10,6))
sns.countplot(x='SeriousWork',data =df)
plt.title('I take my work very seriously')
```

```
Out[13]: Text(0.5, 1.0, 'I take my work very seriously')
```



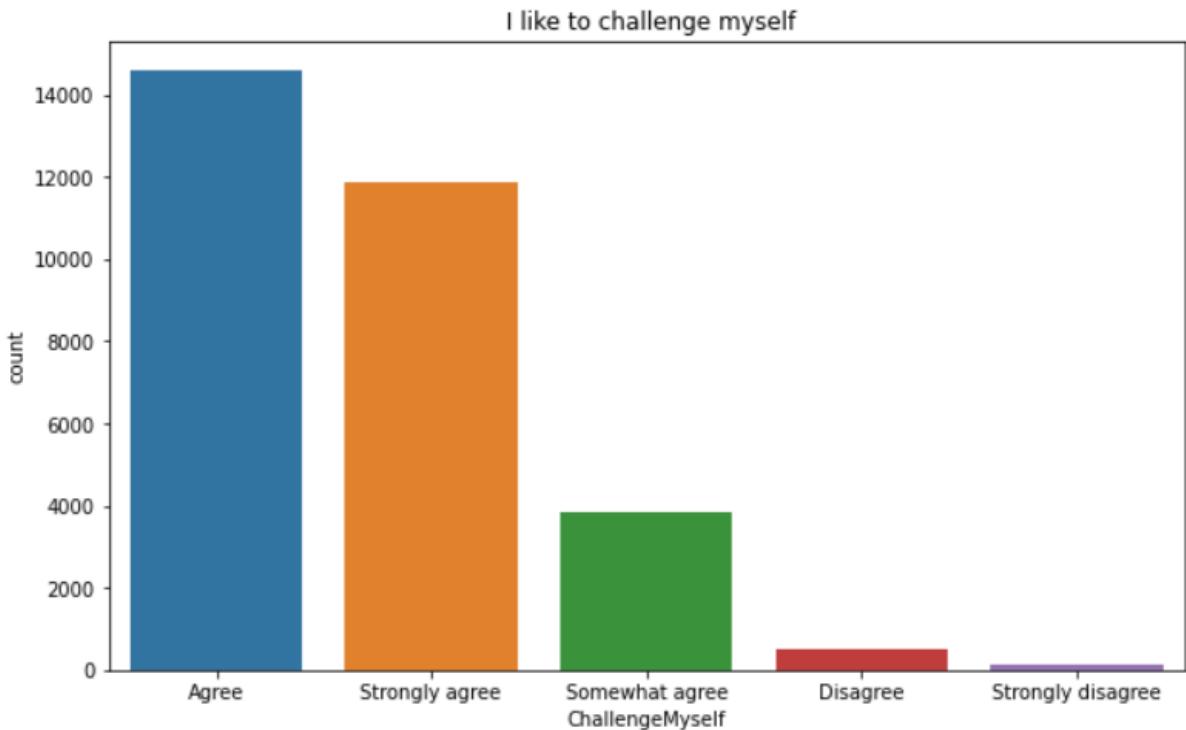
```
In [14]: plt.figure(figsize=(10,6))
sns.countplot(x='WorkPayCare',data =df)
plt.title(" I don't really care what I work on, so long as I'm paid well")
```

```
Out[14]: Text(0.5, 1.0, " I don't really care what I work on, so long as I'm paid well")
```



```
In [15]: #ChallengeMyself: I like to challenge myself
plt.figure(figsize=(10,6))
sns.countplot(x='ChallengeMyself',data =df)
plt.title(" I like to challenge myself")
```

```
Out[15]: Text(0.5, 1.0, ' I like to challenge myself')
```



Last New Job

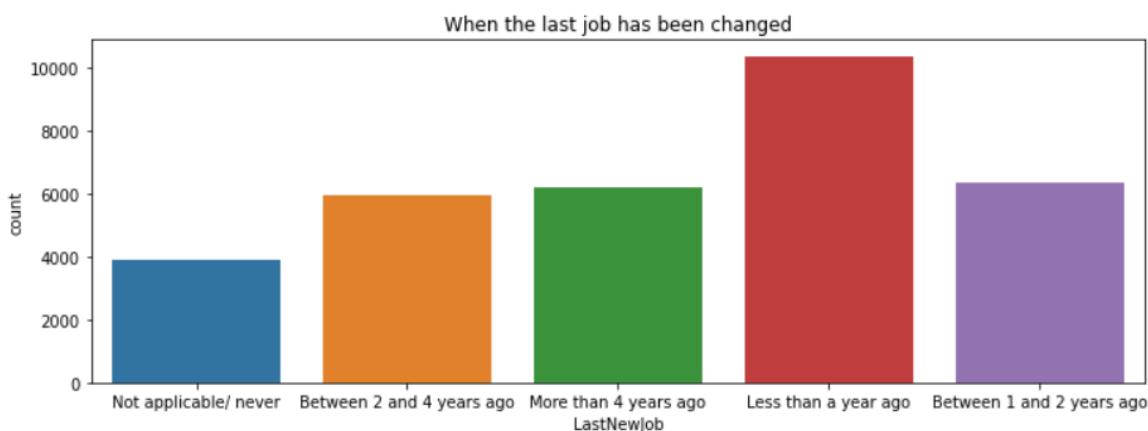
```
In [16]: df['LastNewJob'].unique()  
Out[16]: array(['Not applicable/ never', nan, 'Between 2 and 4 years ago',  
   'More than 4 years ago', 'Less than a year ago',  
   'Between 1 and 2 years ago'], dtype=object)
```

```
In [17]: df['LastNewJob'].nunique()  
Out[17]: 5
```

```
In [18]: df['LastNewJob'].count()  
Out[18]: 32710
```

```
In [19]: plt.figure(figsize = (12,4))  
sns.countplot(x='LastNewJob',data = df)  
plt.title('When the last job has been changed')
```

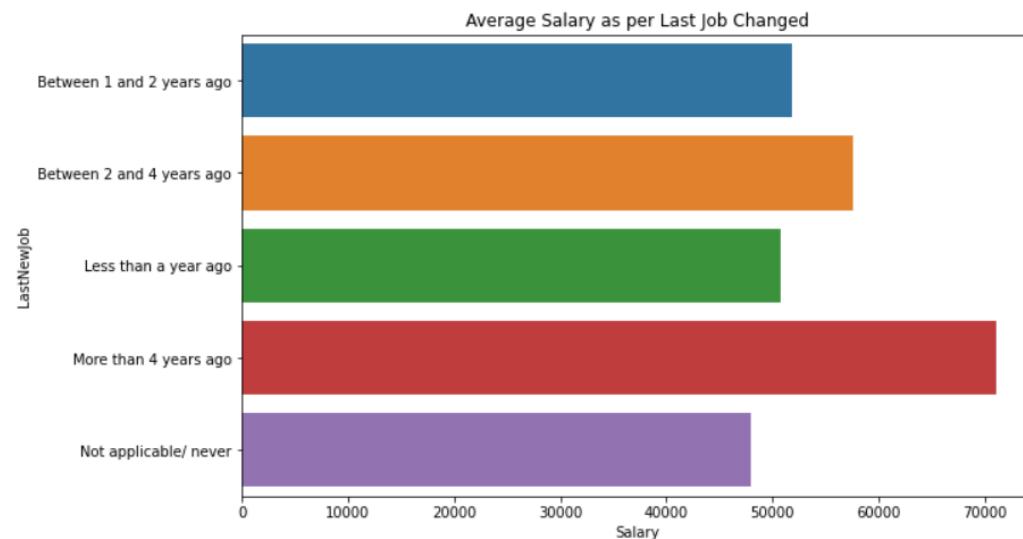
```
Out[19]: Text(0.5, 1.0, 'When the last job has been changed')
```



```
In [20]: df.groupby('LastNewJob')['Salary'].mean()
```

```
Out[20]: LastNewJob  
Between 1 and 2 years ago    51876.225155  
Between 2 and 4 years ago    57633.631064  
Less than a year ago        50825.185580  
More than 4 years ago       71009.633631  
Not applicable/ never       48016.855979  
Name: Salary, dtype: float64
```

```
In [21]: plt.figure(figsize=(10,6))  
sns.barplot(x=df.groupby('LastNewJob')['Salary'].mean(),y=df.groupby('LastNewJob')['Salary'].mean().index)  
plt.title('Average Salary as per Last Job Changed')  
plt.show()
```



CONCLUSIONS

- Imputing Salary as per the years coded job and finding out that job satisfaction and career satisfaction both are directly related to salary and years coded job.
- Cleaning Asses columns to make Assess score for further references in ML and Deep learning.
- Analyzing and cleaning gender gives a view that men are leading the count in the development field and also gender doesn't decide how much salary a developer would get.
- The average salary is maximum in Bermuda followed by Angilla, American Samoa and so on.
- Mostly Developers on StackOverFlow are residents of United States, India and United Kingdom.
- Visual Studio is most popular IDE followed by Notepad++, Sublime Text and Vim.
- JavaScript , SQL and Java are the most popular languages worldwide as well as in India.
- SmallTalk, TypeScirpt, Rust lead the race of average salary worldwide.
- Also MySql and Windows is the most used Database and Platform for Development respectively.
- Auditory Environment is somehow a plus for many developers.
- Most Information Technology companies have a start time of morning.

Future Works

- We have much more columns than we have analyzed in this project, our aim is to enhance our study on this dataset.
- We are having a look on some algorithms to provide us proper Languages, Platform, IDE and Database for better Job Opportunities.

BIBLIOGRAPHY

- McKinney, W. (2017). Python for Data Analysis: Data Wrangling with Pandasz, NumPy and IPython. 2nd edition. O'Reilly Media.
- <<https://docs.python.org/3/>>
- <<https://numpy.org/>>
- <<https://pandas.pydata.org/>>
- <<https://matplotlib.org/>>
- <<https://seaborn.pydata.org/>>
- <<https://www.kaggle.com/>>