Case Study: Objectives 1. Identify the most popular programming languages among IT professionals. 2. Analyze average salaries and income statistics for IT professionals. 3. Explore the age distribution among IT professionals. 4. Provide statistical information on working hours for part-time and full-time IT professionals. 5. Examine the relationship between income and various factors such as working hours, age, education, and other variables. 6. Determine the most popular databases among IT professionals. --- Tools: Python, SQL, Postgres Libraries: numpy , pandas , seaborn , matplotlib.pyplot Dataset: Our data set is a survey works among IT professional , collected and published on Github in the link below. It has 11551 records and 84 columns. (11552, 85) Columns: 'Respondent', 'MainBranch', 'Hobbyist', 'OpenSourcer', 'OpenSource', 'Employment', 'Country', 'Student', 'EdLevel', 'UndergradMajor', 'EduOther', 'OrgSize', 'DevType', 'YearsCode', 'Age1stCode', 'YearsCodePro', 'CareerSat', 'JobSat', 'MgrIdiot', 'MgrMoney', 'MgrWant', 'JobSeek', 'LastHireDate', 'LastInt', 'FizzBuzz', 'JobFactors', 'ResumeUpdate', 'CurrencySymbol', 'CurrencyDesc', 'CompTotal', 'CompFreq', 'ConvertedComp', 'WorkWeekHrs', 'WorkPlan', 'WorkChallenge', 'WorkRemote', 'WorkLoc', 'ImpSyn', 'CodeRev', 'CodeRevHrs', 'UnitTests', 'PurchaseHow', 'PurchaseWhat', 'LanguageWorkedWith', 'LanguageDesireNextYear', 'DatabaseWorkedWith', 'DatabaseDesireNextYear', 'PlatformWorkedWith', 'PlatformDesireNextYear', 'WebFrameWorkedWith', 'WebFrameDesireNextYear', 'MiscTechWorkedWith', 'MiscTechDesireNextYear', 'DevEnviron', 'OpSys', 'Containers', 'BlockchainOrg', 'BlockchainIs', 'BetterLife', 'ITperson', 'OffOn', 'SocialMedia', 'Extraversion', 'ScreenName', 'SOVisit1st', 'SOVisitFreq', 'SOVisitTo', 'SOFindAnswer', 'SOTimeSaved', 'SOHowMuchTime', 'SOAccount', 'SOPartFreq', 'SOJobs', 'EntTeams', 'SOComm', 'WelcomeChange', 'SONewContent', 'Age', 'Gender', 'Trans', 'Sexuality', 'Ethnicity', 'Dependents', 'SurveyLength', 'SurveyEase'], dtype='object') Technical Objectives: 1. Connecting to API and import file 2. Save file into Data frame. 3. Performing EDA. 4. Cleaning the data (Nulls, outliers, rename, datatypes..) 5. Splitting languages field into new data frame. 6. Reframe the new dataset as needed NOTE: so that include one piece of information per record each. 7. Performing EDA & Clean up. 8. Visualizing & Plotting 9. Save the data frame into new table using Postgres Database.

Reshaping and pivot tables pandas provides methods for manipulating a Series and DataFrame to alter the representation of the data for further data processing or data summarization.

pivot() and pivot_table(): Group unique values within one or more discrete categories.

stack() and unstack(): Pivot a column or row level to the opposite axis respectively.

melt() and wide_to_long(): Unpivot a wide DataFrame to a long format.

get_dummies() and from_dummies(): Conversions with indicator variables.

explode(): Convert a column of list-like values to individual rows.

crosstab(): Calculate a cross-tabulation of multiple 1 dimensional factor arrays.

cut(): Transform continuous variables to discrete, categorical values

factorize(): Encode 1 dimensional variables into integer labels.

In [34]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

pd.set_option('display.max_columns', None)
#pd.set_option('display.max_rows', None)
```

In [35]:
```
df = pd.read_csv("https://cf-courses-data.s3.us.
```

In [36]:
```
df.shape
```

Out[36]:
```
(11552, 85)
```

In [42]:
```
df.head()
```

Out[42]:

	Respondent	MainBranch	Hobbyist	OpenSourcer	OpenS
0	4	I am a developer by profession	No	Never	The of O softv
1	9	I am a developer by profession	Yes	Once a month or more often	The of O softv

	Respondent	MainBranch	Hobbyist	OpenSourcer	OpenS
2	13	I am a developer by profession	Yes	Less than once a month but more than once per ...	OSS avera H qualit
3	16	I am a developer by profession	Yes	Never	The of OS softv
4	17	I am a developer by profession	Yes	Less than once a month but more than once per ...	The of OS softv

In [37]:
```
df.duplicated().sum()
```

Out[37]:
```
154
```

In [38]:
```
df=df.drop_duplicates()
```

In [39]:
```
df.duplicated().sum()
```

Out[39]:
```
0
```

In [40]:
```
df.isna().sum()
```

Out[40]:
```
Respondent        0
MainBranch        0
Hobbyist          0
OpenSourcer       0
OpenSource       81
               ...
Sexuality       542
Ethnicity       675
Dependents      140
SurveyLength     19
SurveyEase       14
Length: 85, dtype: int64
```

In [41]:
```
df['LanguageWorkedWith']
```

Out[41]:
```
0                                                  C;C++;C#;P
ython;SQL
1         Bash/Shell/PowerShell;C#;HTML/CSS;JavaSc
ript;P...
2         Bash/Shell/PowerShell;HTML/CSS;JavaScrip
t;PHP;...
3         Bash/Shell/PowerShell;C#;HTML/CSS;JavaSc
ript;T...
4         Bash/Shell/PowerShell;HTML/CSS;JavaScrip
t;Type...
                                ...
11547                   C#;F#;HTML/CSS;Java;JavaS
cript;SQL
11548                           HTML/CSS;JavaScrip
t;PHP;SQL
11549     Assembly;Bash/Shell/PowerShell;C;C++;C#;
Java;J...
11550     Bash/Shell/PowerShell;C++;C#;HTML/CSS;Ja
va;Jav...
11551     Bash/Shell/PowerShell;C;C++;Go;HTML/CSS;
PHP;Py...
Name: LanguageWorkedWith, Length: 11398, dtype: o
bject
```

What you can do in this case is: Step 1 - convert comma-separated data in one column to multiple columns (wide data format)

In [10]:
```
data = pd.concat((df, df['LanguageWorkedWith'].s
```

Now we have new table based on the fields we need from the splited dataframe (named languages), Next some data cleaning then: Step 2 - convert from wide data format (multiple columns) to long data format, where multiple columns are squeezed into one column, and their values are converted into extra rows

In [11]:
```
languages=data[[0,85,86,87,88,89,90,91,92,93,94,
```

In [12]:
```
languages.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11398 entries, 0 to 11551
Data columns (total 20 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   0       11398 non-null  int64 
 1   85      11387 non-null  object
 2   86      11063 non-null  object
 3   87      10204 non-null  object
 4   88      8850 non-null   object
 5   89      6722 non-null   object
 6   90      4571 non-null   object
 7   91      2871 non-null   object
 8   92      1737 non-null   object
 9   93      1033 non-null   object
 10  94      609 non-null    object
 11  95      334 non-null    object
 12  96      180 non-null    object
 13  97      112 non-null    object
 14  98      63 non-null     object
 15  99      30 non-null     object
 16  100     16 non-null     object
 17  101     7 non-null      object
 18  102     2 non-null      object
 19  103     1 non-null      object
dtypes: int64(1), object(19)
memory usage: 1.8+ MB
```

Rename our new table's columns:

In [15]:
```
#languages=languages.rename(columns={0:'Responde
```

In [13]:
```
languages=languages.rename(columns={0:'Responden
```

In [14]:
```
languages
```

Out[14]:

	Respondent	85	8
0	4	C	C-
1	9	Bash/Shell/PowerShell	(
2	13	Bash/Shell/PowerShell	HTML/CS
3	16	Bash/Shell/PowerShell	(
4	17	Bash/Shell/PowerShell	HTML/CS
...	...	...	
11547	25136	C#	I
11548	25137	HTML/CSS	JavaScri
11549	25138	Assembly	Bash/Shell/PowerSh
11550	25141	Bash/Shell/PowerShell	C-
11551	25142	Bash/Shell/PowerShell	

11398 rows × 20 columns

In [15]:
```
lan=pd.melt(languages, id_vars='Respondent',igno
```

In [16]:
```
lan.drop(columns='variable',inplace=True)
```

In [17]:
```
lan['value'].isna().sum()
```

Out[17]:
```
156770
```

In [18]:
```
lan.dropna(inplace=True)
```

In [19]:
```
lan[lan['Respondent']==25142]
```

Out[19]:

	Respondent	value
11551	25142	Bash/Shell/PowerShell
11551	25142	C
11551	25142	C++
11551	25142	Go
11551	25142	HTML/CSS
11551	25142	PHP
11551	25142	Python
11551	25142	R

In [20]:
```
lan['value'].value_counts()
```

Out[20]:
```
JavaScript               8687
HTML/CSS                 7830
SQL                      7106
Bash/Shell/PowerShell    4642
```

```
Python                   4542
Java                     4506
C#                       4288
TypeScript               3232
PHP                      2913
C++                      1946
C                        1578
Ruby                     1149
Go                       1114
Other(s):                 840
Kotlin                    751
Swift                     707
VBA                       628
R                         585
Objective-C               518
Scala                     492
Assembly                  437
Rust                      324
Dart                      237
Elixir                    187
Clojure                   164
F#                        158
WebAssembly               133
Erlang                     98
Name: value, dtype: int64
```

In [21]:

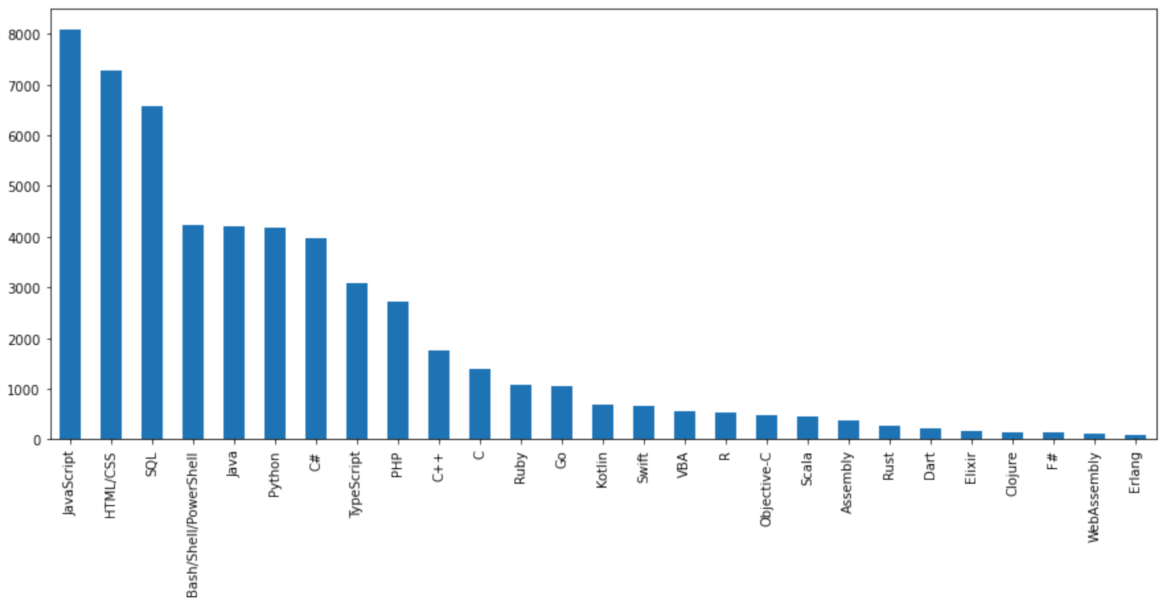
```
lan.drop(lan[lan['value'] == 'Other(s):'].index,
```

In [47]:

```
lan['value'].value_counts().plot(kind='bar',figs
```

Out[47]:

```
<AxesSubplot:>
```


8000
7000
6000
5000
4000
3000
2000
1000
0
JavaScript
HTML/CSS
SQL
Bash/Shell/PowerShell
Java
Python
C#
TypeScript
PHP
C++
C
Ruby
Go
Kotlin
Swift
VBA
R
Objective-C
Scala
Assembly
Rust
Dart
Elixir
Clojure
F#
WebAssembly
Erlang

In [ ]: