M4DataVisualization-lab

June 24, 2022

1 Data Visualization Lab

Estimated time needed: **45 to 60** minutes

In this assignment you will be focusing on the visualization of data.

The data set will be presented to you in the form of a RDBMS.

You will have to use SQL queries to extract the data.

1.1 Objectives

In this lab you will perform the following:

- Visualize the distribution of data.
- Visualize the relationship between two features.
- Visualize composition of data.
- Visualize comparison of data.

1.2 Demo: How to work with database

Download database file.

[2]:
```
!wget https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
↪IBM-DA0321EN-SkillsNetwork/LargeData/m4_survey_data.sqlite
```

```
--2022-06-24 17:46:12--  https://cf-courses-data.s3.us.cloud-object-
storage.appdomain.cloud/IBM-DA0321EN-
SkillsNetwork/LargeData/m4_survey_data.sqlite
Resolving cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-
courses-data.s3.us.cloud-object-storage.appdomain.cloud)… 169.63.118.104
Connecting to cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud (cf-
courses-data.s3.us.cloud-object-storage.appdomain.cloud)|169.63.118.104|:443…
connected.
HTTP request sent, awaiting response… 200 OK
Length: 36679680 (35M) [application/octet-stream]
Saving to: ‘m4_survey_data.sqlite.1’

m4_survey_data.sqli 100%[===================>]  34.98M  29.9MB/s    in 1.2s
```

```
2022-06-24 17:46:13 (29.9 MB/s) - ‘m4_survey_data.sqlite.1’ saved
[36679680/36679680]
```

Connect to the database.

[3]:
```
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import sqlite3
conn = sqlite3.connect("m4_survey_data.sqlite") # open a database connection
```

Import pandas module.

[4]:
```
import pandas as pd
```

1.3 Demo: How to run an sql query

[5]:
```
# print how many rows are there in the table named 'master'
QUERY = """
SELECT COUNT(*)
FROM master
"""

# the read_sql_query runs the sql query and returns the data as a dataframe
df = pd.read_sql_query(QUERY,conn)
df.head()
```

[5]:

	COUNT(*)
0	11398

1.4 Demo: How to list all tables

[6]:
```
# print all the tables names in the database
QUERY = """
SELECT name as Table_Name FROM
sqlite_master WHERE
type = 'table'
"""
# the read_sql_query runs the sql query and returns the data as a dataframe
pd.read_sql_query(QUERY,conn)
```

[6]:

	Table_Name
0	EduOther
1	DevType
2	LastInt
3	JobFactors

```
4                  WorkPlan
5             WorkChallenge
6        LanguageWorkedWith
7    LanguageDesireNextYear
8        DatabaseWorkedWith
9    DatabaseDesireNextYear
10       PlatformWorkedWith
11   PlatformDesireNextYear
12       WebFrameWorkedWith
13   WebFrameDesireNextYear
14       MiscTechWorkedWith
15   MiscTechDesireNextYear
16               DevEnviron
17               Containers
18                SOVisitTo
19            SONewContent
20                   Gender
21                Sexuality
22                Ethnicity
23                   master
```

1.5 Demo: How to run a group by query

[7]:
```
QUERY = """
SELECT Age,COUNT(*) as count
FROM master
group by age
order by age
"""
pd.read_sql_query(QUERY,conn)
```

[7]:

	Age	count
0	NaN	287
1	16.0	3
2	17.0	6
3	18.0	29
4	19.0	78
5	20.0	109
6	21.0	203
7	22.0	406
8	23.0	581
9	24.0	679
10	25.0	738
11	26.0	720
12	27.0	724
13	28.0	787
14	29.0	697

15	30.0	651
16	31.0	531
17	32.0	489
18	33.0	483
19	34.0	395
20	35.0	393
21	36.0	308
22	37.0	280
23	38.0	279
24	39.0	232
25	40.0	187
26	41.0	136
27	42.0	162
28	43.0	100
29	44.0	95
30	45.0	85
31	46.0	66
32	47.0	68
33	48.0	64
34	49.0	66
35	50.0	57
36	51.0	29
37	52.0	41
38	53.0	32
39	54.0	26
40	55.0	13
41	56.0	16
42	57.0	11
43	58.0	12
44	59.0	11
45	60.0	2
46	61.0	10
47	62.0	5
48	63.0	7
49	65.0	2
50	66.0	1
51	67.0	1
52	69.0	1
53	71.0	2
54	72.0	1
55	99.0	1

1.6 Demo: How to describe a table

[8]:
```
table_name = 'master'  # the table you wish to describe

QUERY = """
SELECT sql FROM sqlite_master
WHERE name= '{}'
""".format(table_name)

df = pd.read_sql_query(QUERY,conn)
print(df.iat[0,0])
```

```
CREATE TABLE "master" (
"index" INTEGER,
  "Respondent" INTEGER,
  "MainBranch" TEXT,
  "Hobbyist" TEXT,
  "OpenSourcer" TEXT,
  "OpenSource" TEXT,
  "Employment" TEXT,
  "Country" TEXT,
  "Student" TEXT,
  "EdLevel" TEXT,
  "UndergradMajor" TEXT,
  "OrgSize" TEXT,
  "YearsCode" TEXT,
  "Age1stCode" TEXT,
  "YearsCodePro" TEXT,
  "CareerSat" TEXT,
  "JobSat" TEXT,
  "MgrIdiot" TEXT,
  "MgrMoney" TEXT,
  "MgrWant" TEXT,
  "JobSeek" TEXT,
  "LastHireDate" TEXT,
  "FizzBuzz" TEXT,
  "ResumeUpdate" TEXT,
  "CurrencySymbol" TEXT,
  "CurrencyDesc" TEXT,
  "CompTotal" REAL,
  "CompFreq" TEXT,
  "ConvertedComp" REAL,
  "WorkWeekHrs" REAL,
  "WorkRemote" TEXT,
  "WorkLoc" TEXT,
  "ImpSyn" TEXT,
  "CodeRev" TEXT,
  "CodeRevHrs" REAL,
```

```
    "UnitTests" TEXT,
    "PurchaseHow" TEXT,
    "PurchaseWhat" TEXT,
    "OpSys" TEXT,
    "BlockchainOrg" TEXT,
    "BlockchainIs" TEXT,
    "BetterLife" TEXT,
    "ITperson" TEXT,
    "OffOn" TEXT,
    "SocialMedia" TEXT,
    "Extraversion" TEXT,
    "ScreenName" TEXT,
    "SOVisit1st" TEXT,
    "SOVisitFreq" TEXT,
    "SOFindAnswer" TEXT,
    "SOTimeSaved" TEXT,
    "SOHowMuchTime" TEXT,
    "SOAccount" TEXT,
    "SOPartFreq" TEXT,
    "SOJobs" TEXT,
    "EntTeams" TEXT,
    "SOComm" TEXT,
    "WelcomeChange" TEXT,
    "Age" REAL,
    "Trans" TEXT,
    "Dependents" TEXT,
    "SurveyLength" TEXT,
    "SurveyEase" TEXT
)
```

2 Hands-on Lab

2.1 Visualizing distribution of data

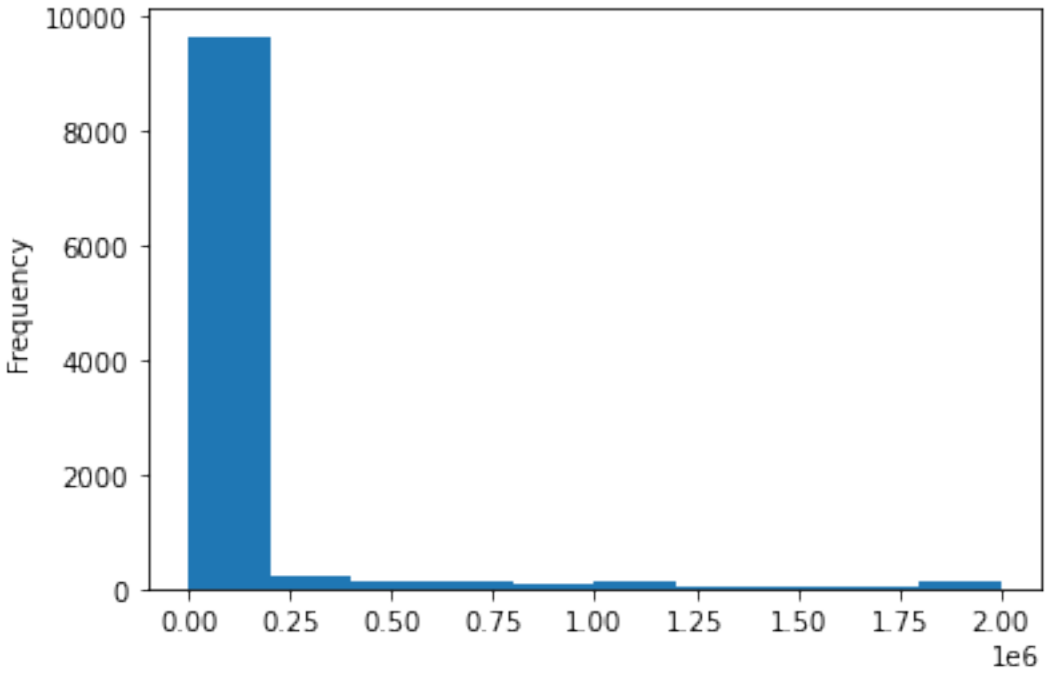
2.1.1 Histograms

Plot a histogram of `ConvertedComp`.

[9]:
```
# your code goes here
QUERY = """
SELECT *
FROM master
"""
df = pd.read_sql_query(QUERY,conn)
df.shape
```

[9]:
```
(11398, 63)
```

[10]:
```
df['ConvertedComp'].plot(kind='hist')
plt.show()
```

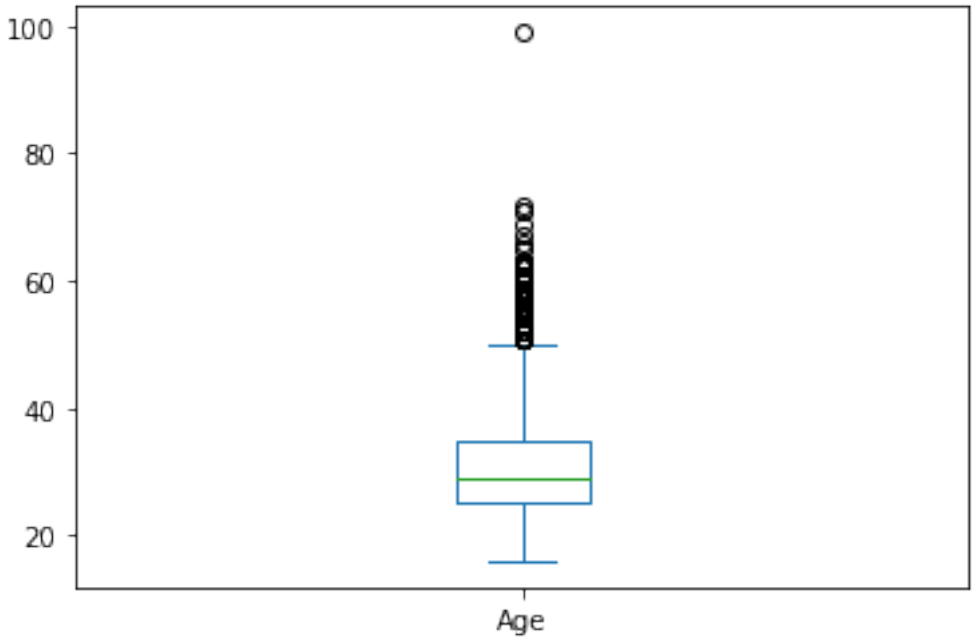


2.1.2 Box Plots

Plot a box plot of `Age`.

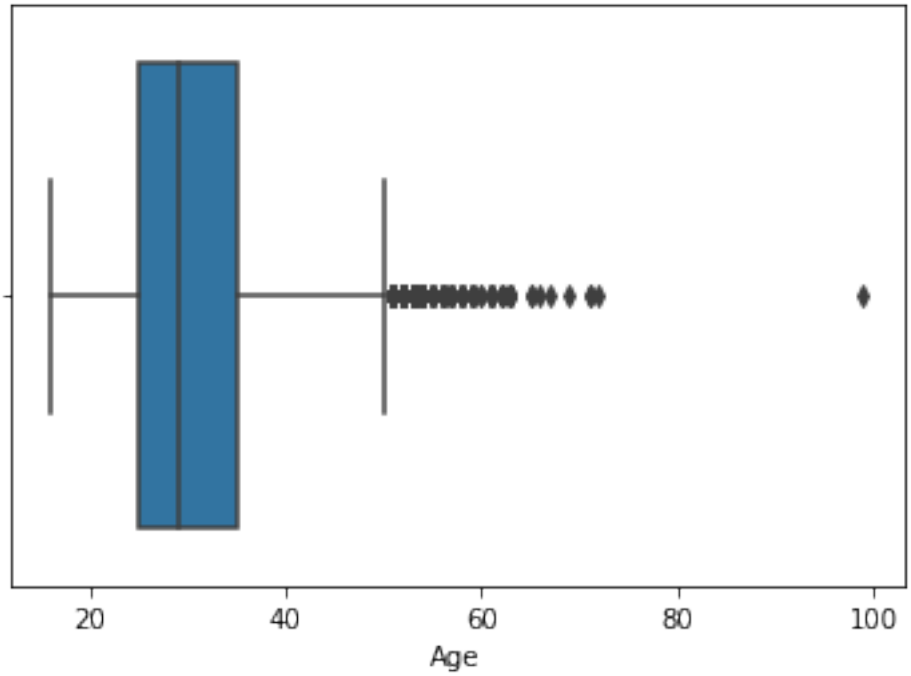
[11]:
```
# your code goes here
df['Age'].plot(kind='box')
```

[11]:
```
<AxesSubplot:>
```



[12]:
```
sns.boxplot(df['Age'])
```

[12]: <AxesSubplot:xlabel='Age'>



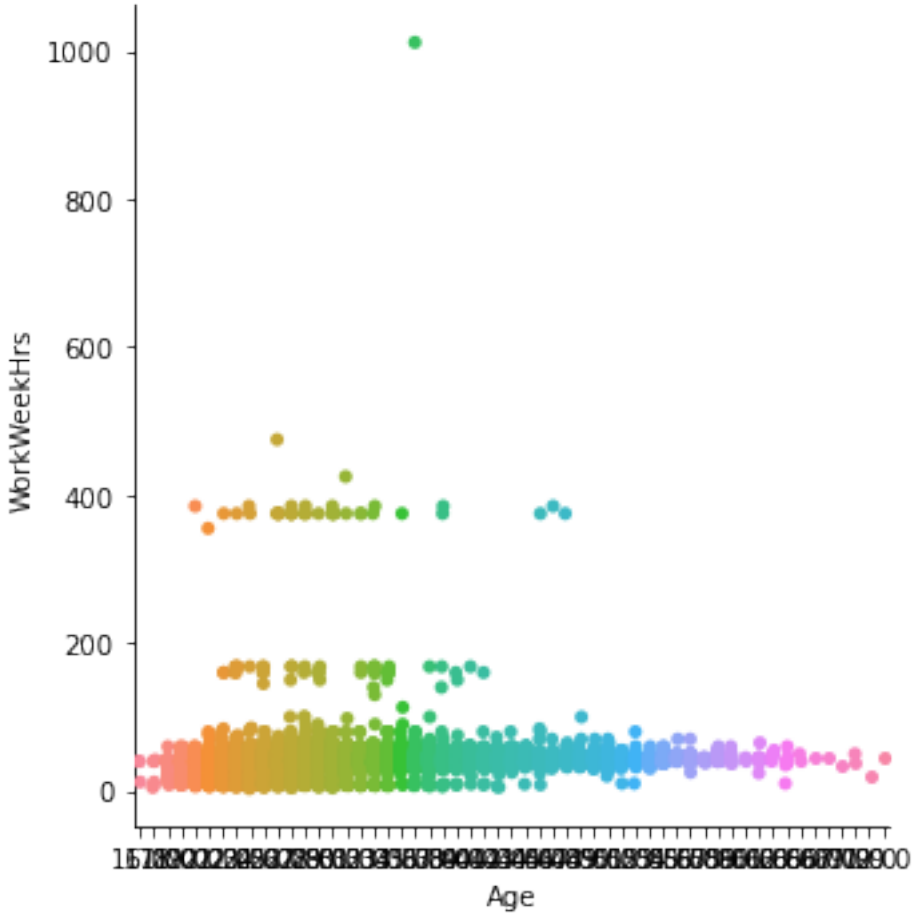
2.2 Visualizing relationships in data

2.2.1 Scatter Plots

Create a scatter plot of `Age` and `WorkWeekHrs`.

[13]:
```
# your code goes here
sns.catplot(x='Age', y='WorkWeekHrs', data=df)
```

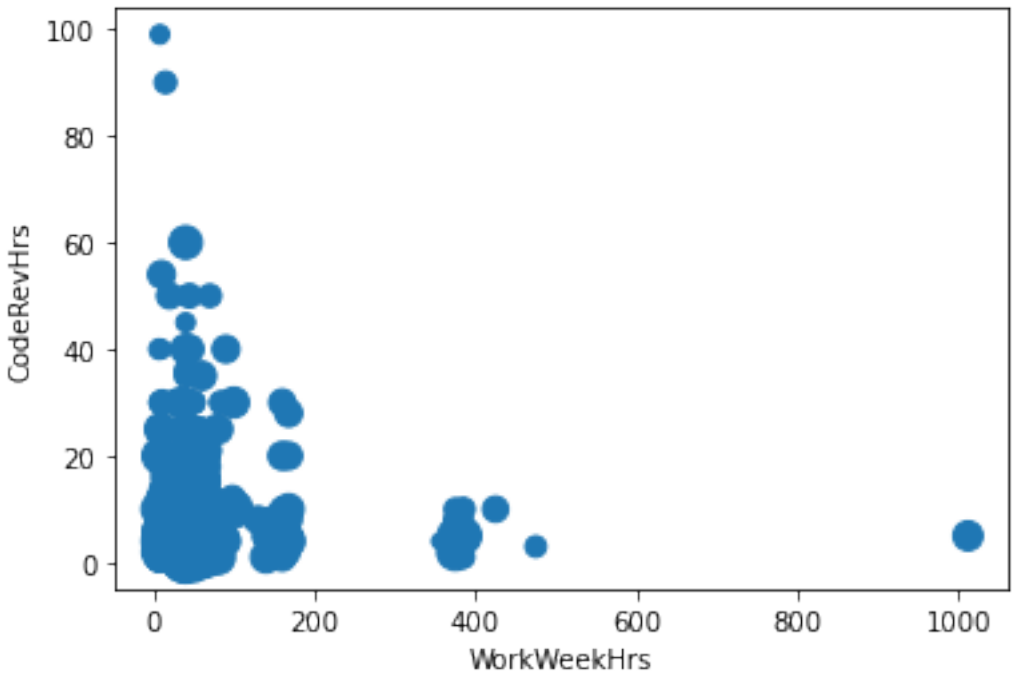
[13]:
```
<seaborn.axisgrid.FacetGrid at 0x7fabb05b3250>
```



2.2.2 Bubble Plots

Create a bubble plot of `WorkWeekHrs` and `CodeRevHrs`, use `Age` column as bubble size.

[14]:
```
# your code goes here
norm_age = (df['Age'] - df['Age'].min()) / (df['Age'].max() - df['Age'].min())
ax = df.plot(kind='scatter', x = 'WorkWeekHrs', y='CodeRevHrs', s=norm_age *␣
↪500)
```



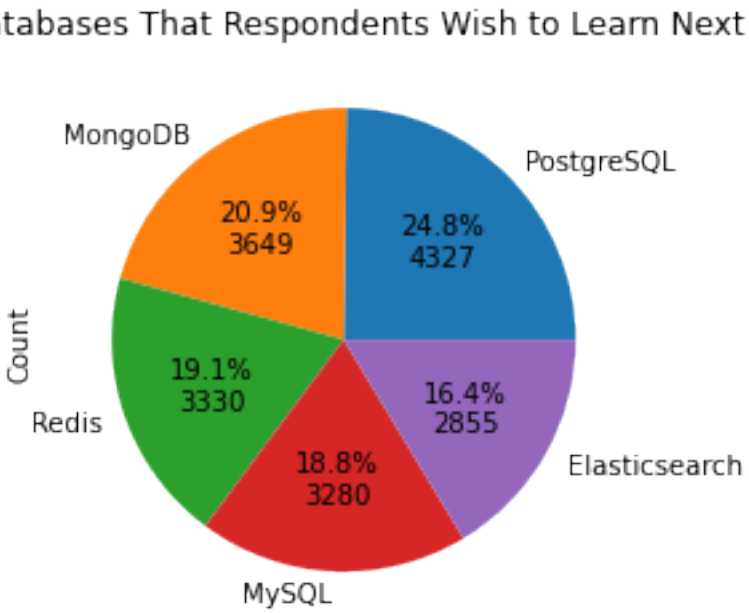
2.3 Visualizing composition of data

2.3.1 Pie Charts

Create a pie chart of the top 5 databases that respondents wish to learn next year. Label the pie chart with database names. Display percentages of each database on the pie chart.

[15]:
```
# your code goes here
QUERY2 = """
SELECT * FROM
DatabaseDesireNextYear
"""
df2 = pd.read_sql_query(QUERY2,conn)
# df2.head(15)
df3 = pd.DataFrame(df2['DatabaseDesireNextYear'].value_counts(sort=True,␣
 ↪ascending=False)[0:5])
# df3.reset_index(inplace=True)
# df3.rename(columns={'DatabaseDesireNextYear': 'Count', 'index':'Language'},␣
 ↪inplace=True)
df3.rename(columns={'DatabaseDesireNextYear': 'Count'}, inplace=True)
def func(pct, allvalues):
    absolute = int(pct / 100.*np.sum(allvalues))
    return "{:.1f}%\n{:d}".format(pct, absolute)
ax = df3['Count'].plot(kind='pie', autopct = lambda pct: func(pct, df3))
ax.set_title("Top 5 Databases That Respondents Wish to Learn Next Year")
```

[15]: Text(0.5, 1.0, 'Top 5 Databases That Respondents Wish to Learn Next Year')



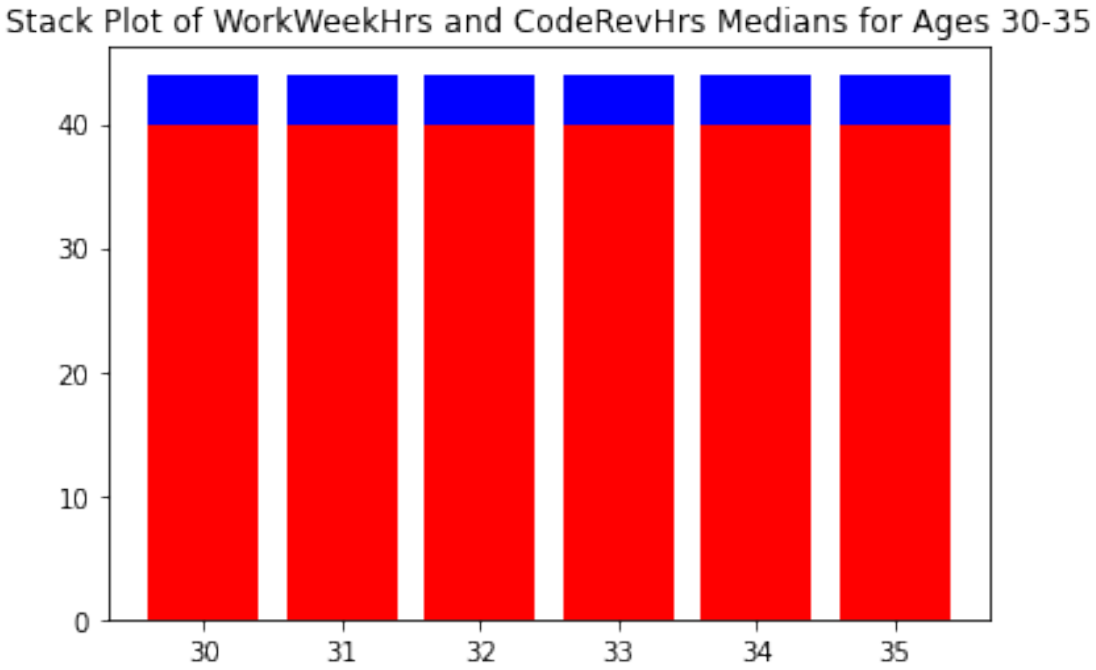
2.3.2 Stacked Charts

Create a stacked chart of median `WorkWeekHrs` and `CodeRevHrs` for the age group 30 to 35.

[16]:
```
# your code goes here
x = [30, 31, 32, 33, 34 ,35]
df_custom = df[['Age', 'WorkWeekHrs', 'CodeRevHrs']].loc[(df['Age'] >=30) &␣
 ↪(df['Age'] <=35)]
y1 = [df_custom.loc[df_custom['Age'] == 30]['WorkWeekHrs'].median(),
     df_custom.loc[df_custom['Age'] == 31]['WorkWeekHrs'].median(),
     df_custom.loc[df_custom['Age'] == 32]['WorkWeekHrs'].median(),
     df_custom.loc[df_custom['Age'] == 33]['WorkWeekHrs'].median(),
     df_custom.loc[df_custom['Age'] == 34]['WorkWeekHrs'].median(),
     df_custom.loc[df_custom['Age'] == 35]['WorkWeekHrs'].median()]

y2 = [df_custom.loc[df_custom['Age'] == 30]['CodeRevHrs'].median(),
     df_custom.loc[df_custom['Age'] == 31]['CodeRevHrs'].median(),
     df_custom.loc[df_custom['Age'] == 32]['CodeRevHrs'].median(),
     df_custom.loc[df_custom['Age'] == 33]['CodeRevHrs'].median(),
     df_custom.loc[df_custom['Age'] == 34]['CodeRevHrs'].median(),
     df_custom.loc[df_custom['Age'] == 35]['CodeRevHrs'].median()]

plt.bar(x, y1, color='r')
plt.bar(x, y2, bottom=y1, color='b')
```

```
z = "Stack Plot of WorkWeekHrs and CodeRevHrs Medians for Ages 30-35"
plt.title(z)

# ax = plt.stackplot(x, y1, y2,colors =['r', 'c'])
# ax.set_title("Stack Plot of WorkWeekHrs and CodeRevHrs Medians for Ages␣
 ↪30-35")
```

[16]:
```
Text(0.5, 1.0, 'Stack Plot of WorkWeekHrs and CodeRevHrs Medians for Ages
30-35')
```



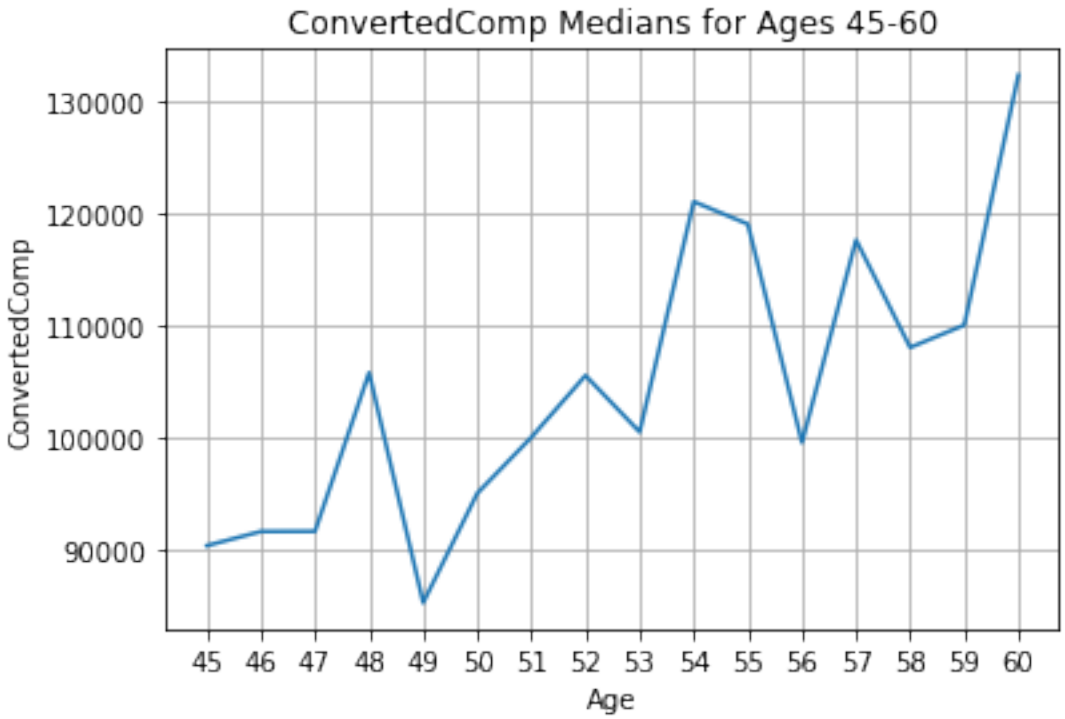
2.4 Visualizing comparison of data

2.4.1 Line Chart

Plot the median `ConvertedComp` for all ages from 45 to 60.

[39]:
```
# your code goes here
df_need = df[['Age', 'ConvertedComp']].loc[df['Age'].between(45, 60)]
df_need = df_need.groupby(['Age']).median()
df_need.reset_index(inplace=True)
df_need['Age'] = df_need['Age'].astype(int)
df_need
plt.plot(df_need['Age'], df_need['ConvertedComp'])
plt.xlabel('Age')
plt.ylabel('ConvertedComp')
```

```
plt.title('ConvertedComp Medians for Ages 45-60')
plt.xticks(df_need['Age'], df_need['Age'])
plt.grid()
# plt.grid(axis = 'y')
```

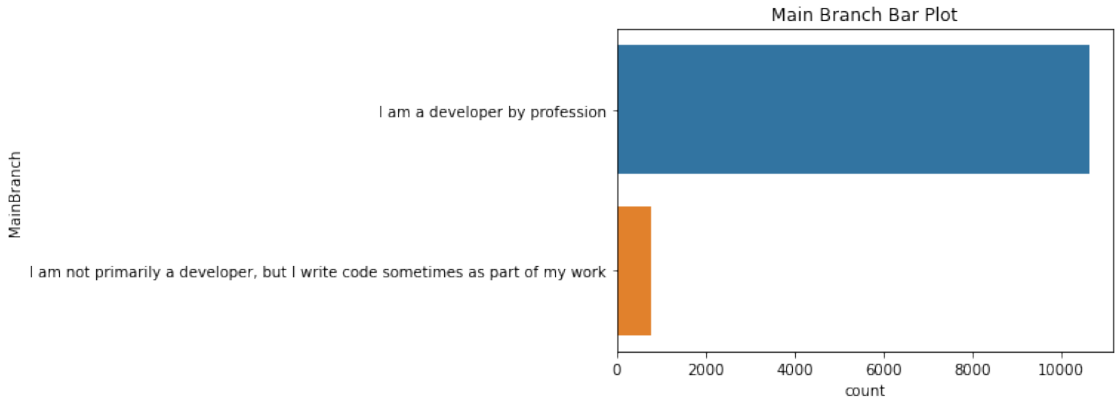


2.4.2 Bar Chart

Create a horizontal bar chart using column `MainBranch`.

[18]:
```
# your code goes here
ax = sns.countplot(y = df['MainBranch'], orient="h")
ax.set_title("Main Branch Bar Plot")
```

[18]:
```
Text(0.5, 1.0, 'Main Branch Bar Plot')
```



Close the database connection.

[19]:
```
conn.close()
```

2.5 Authors

Ramesh Sannareddy

2.5.1 Other Contributors

Rav Ahuja

2.6 Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2020-10-17	0.1	Ramesh Sannareddy	Created initial version of the lab

Copyright © 2020 IBM Corporation. This notebook and its source code are released under the terms of the MIT License.