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.

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.

Demo: How to work with database

Download database file.

[1]:

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

```
--2024-03-18 05:47:57--  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, 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'

m4_survey_data.sqli 100%[===================>]  34.98M  33.1MB/s    in 1.1s

2024-03-18 05:47:59 (33.1 MB/s) - 'm4_survey_data.sqlite' saved [36679680/36679680]
```

Connect to the database.

[2]:

```
import sqlite3
conn = sqlite3.connect("m4_survey_data.sqlite") # open a database connection
```

Import pandas module.

[3]:

```
import pandas as pd
```

Demo: How to run an sql query

[4]:

```
# 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()
```

[4]:

	COUNT(*)
0	11398

Demo: How to list all tables

[5]:

```
# 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)
```

[5]:

	Table_Name
0	EduOther
1	DevType
2	LastInt
3	JobFactors
4	WorkPlan
5	WorkChallenge
6	LanguageWorkedWith
7	LanguageDesireNextYear
8	DatabaseWorkedWith
9	DatabaseDesireNextYear

	Table_Name
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

Demo: How to run a group by query

[6]:

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

[6]:

	Age	count
0	NaN	287
1	16.0	3
2	17.0	6
3	18.0	29

	Age	count
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

	Age	count
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

	Age	count
54	72.0	1
55	99.0	1

Demo: How to describe a table

[7]:

```
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
)
```

Hands-on Lab

Visualizing distribution of data

Histograms

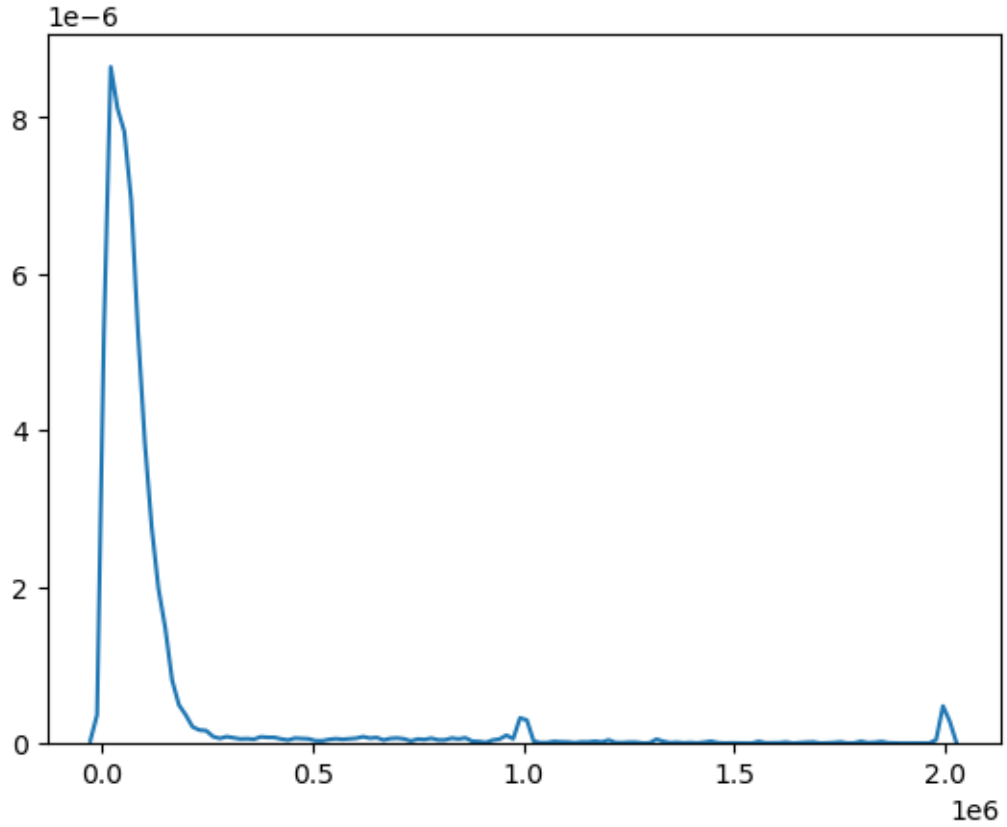
Plot a histogram of ConvertedComp.

[8]:

```
# your code goes here
#df['ConvertedComp'].plot.hist(bins=12, alpha=0.5)
#df = pd.DataFrame(np.random.rand(10, 1), columns=['ConvertedComp'])
#df['ConvertedComp']

import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

[9]:

```
QUERY = """
SELECT ConvertedComp
FROM master
"""
df = pd.read_sql_query(QUERY,conn)
df.head()
sns.distplot(df, 'ConvertedComp', hist = False, kde = True)

plt.show()
```

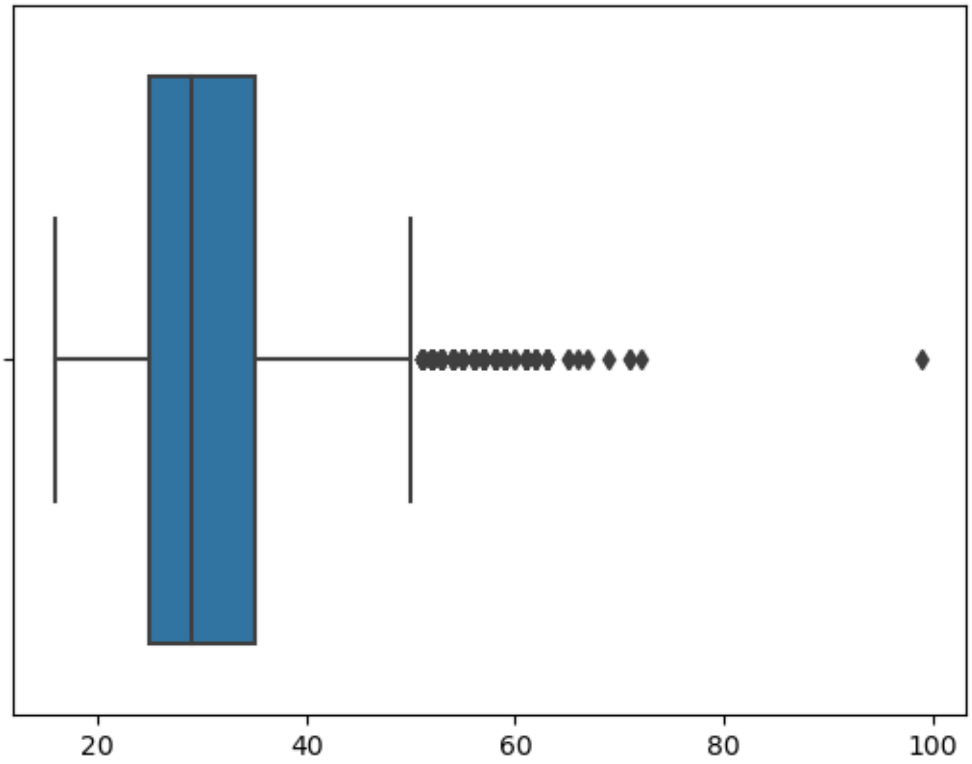


Box Plots

Plot a box plot of Age.

[10]:

```
# your code goes here
QUERY = """
SELECT Age
FROM master
"""
df = pd.read_sql_query(QUERY,conn)
df.head()
sns.boxplot(df,)

plt.show()
```



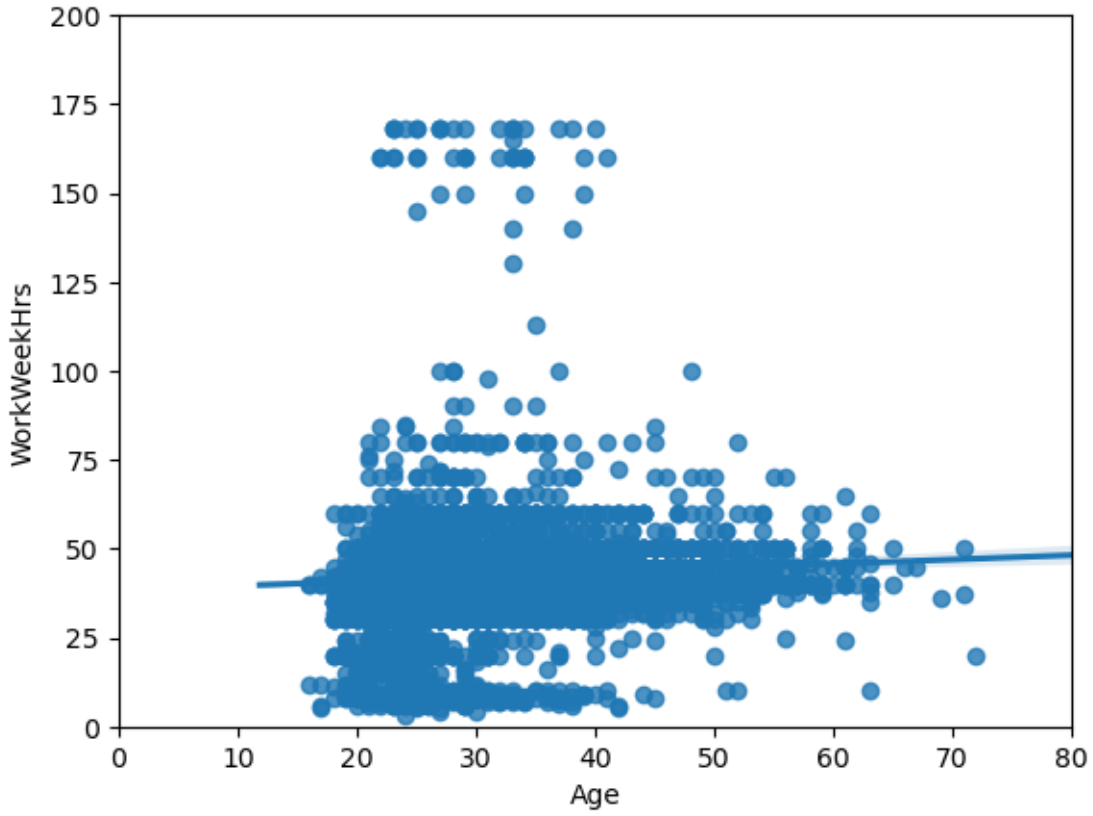
Visualizing relationships in data

Scatter Plots

Create a scatter plot of Age and WorkWeekHrs.

[11]:

```
# your code goes here

QUERY = """
SELECT Age, WorkWeekHrs
FROM master
"""
df = pd.read_sql_query(QUERY,conn)
df.head()

# Age as potential predictor variable of WorkWeekHrs
sns.regplot(x="Age", y="WorkWeekHrs", data=df)
plt.ylim(0,200)
plt.xlim(0,80)

plt.show()
```



Bubble Plots

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

[12]:

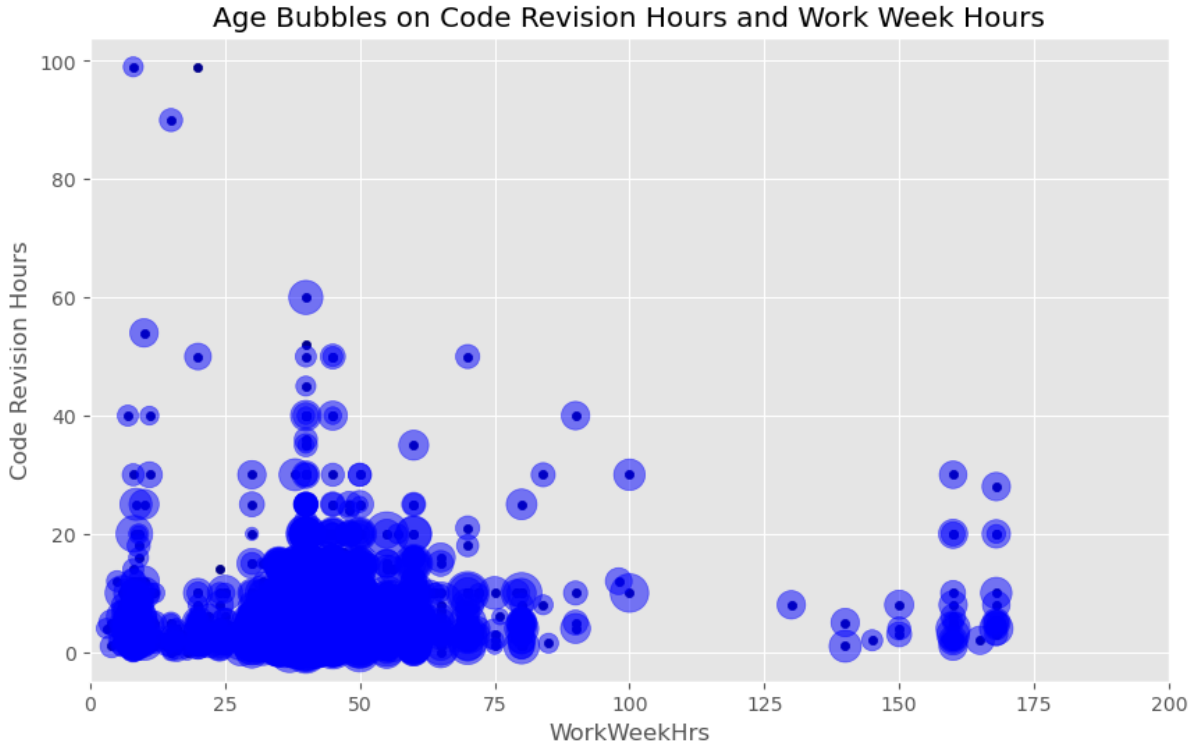
```
# your code goes here
%matplotlib inline

import matplotlib as mpl
import matplotlib.pyplot as plt

mpl.style.use('ggplot') # optional: for ggplot-like style

# check for latest version of Matplotlib
print('Matplotlib version: ', mpl.__version__) # >= 2.0.0
```

```
Matplotlib version:  3.5.3
```

[13]:

```
# your code goes here
QUERY = """
SELECT Age, WorkWeekHrs, CodeRevHrs
FROM master
"""
df = pd.read_sql_query(QUERY,conn)
df.head()

ax = df.plot(kind='scatter', x='WorkWeekHrs', y='CodeRevHrs', figsize=(10, 6), color='darkblue')

plt.title('Age Bubbles on Code Revision Hours and Work Week Hours')
plt.xlabel('WorkWeekHrs')
plt.ylabel('CodeRevHrs')
```

```
plt.xlim(0,200)
#plt.ylim(0,70)

#sns.regplot(x="WorkWeekHrs", y="CodeRevHrs", data=df)



# normalize Age data
norm_age = (df['Age'] - df['Age'].min()) / (df['Age'].max() - df['Age'].min())

# Age
ax = df.plot(kind='scatter',
             x='WorkWeekHrs',
             y='CodeRevHrs',
             alpha=0.5,
             color="blue",
             s=norm_age * 1000 + 10,
             ax = ax
            )

ax.set_ylabel('Code Revision Hours')
ax.set_title('Age Bubbles on Code Revision Hours and Work Week Hours')

plt.show()
```



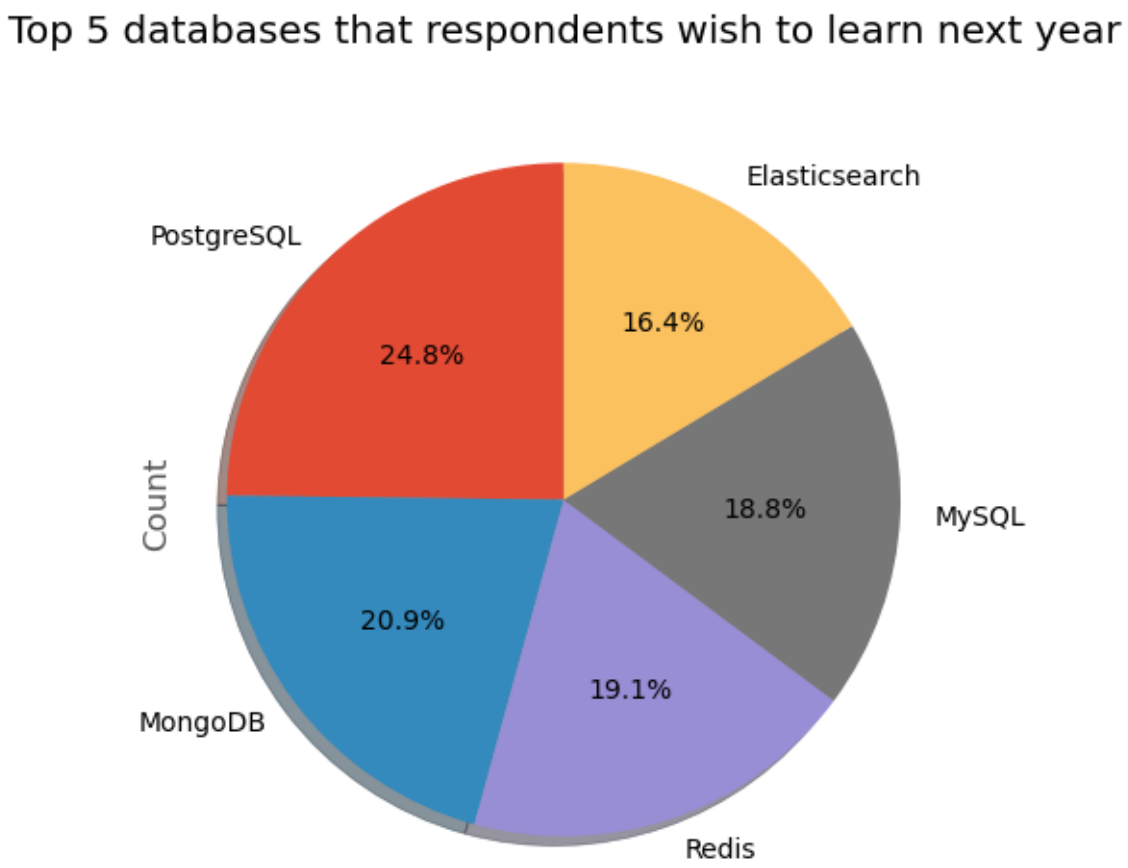
Visualizing composition of data

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.

[14]:

```
# your code goes here

table_name = 'DatabaseDesireNextYear'  # 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])

QUERY = """
SELECT DatabaseDesireNextYear, count(DatabaseDesireNextYear) as Count
FROM DatabaseDesireNextYear
GROUP BY DatabaseDesireNextYear
"""
df = pd.read_sql_query(QUERY,conn)
df.sort_values('Count', ascending=False,inplace=True)

df = df.head(5)
total = df.sum(0)[1]

df['percent'] = 100 * df['Count']/total

df.set_index('DatabaseDesireNextYear')
print(df)


# autopct create %, start angle represent starting point
df['Count'].plot(kind='pie',
                 figsize=(5, 6),
                 autopct='%1.1f%%', # add in percentages
                 startangle=90,     # start angle 90° (Africa)
                 shadow=True,       # add shadow
                 labels=df['DatabaseDesireNextYear'])

plt.title('Top 5 databases that respondents wish to learn next year')
plt.axis('equal') # Sets the pie chart to look like a circle.

plt.show()
CREATE TABLE "DatabaseDesireNextYear" (
"Respondent" INTEGER,
  "DatabaseDesireNextYear" TEXT
)
   DatabaseDesireNextYear  Count    percent
11           PostgreSQL   4328  24.809401
7             MongoDB   3649  20.917168
12              Redis   3331  19.094296
8              MySQL   3281  18.807681
3         Elasticsearch   2856  16.371453
```



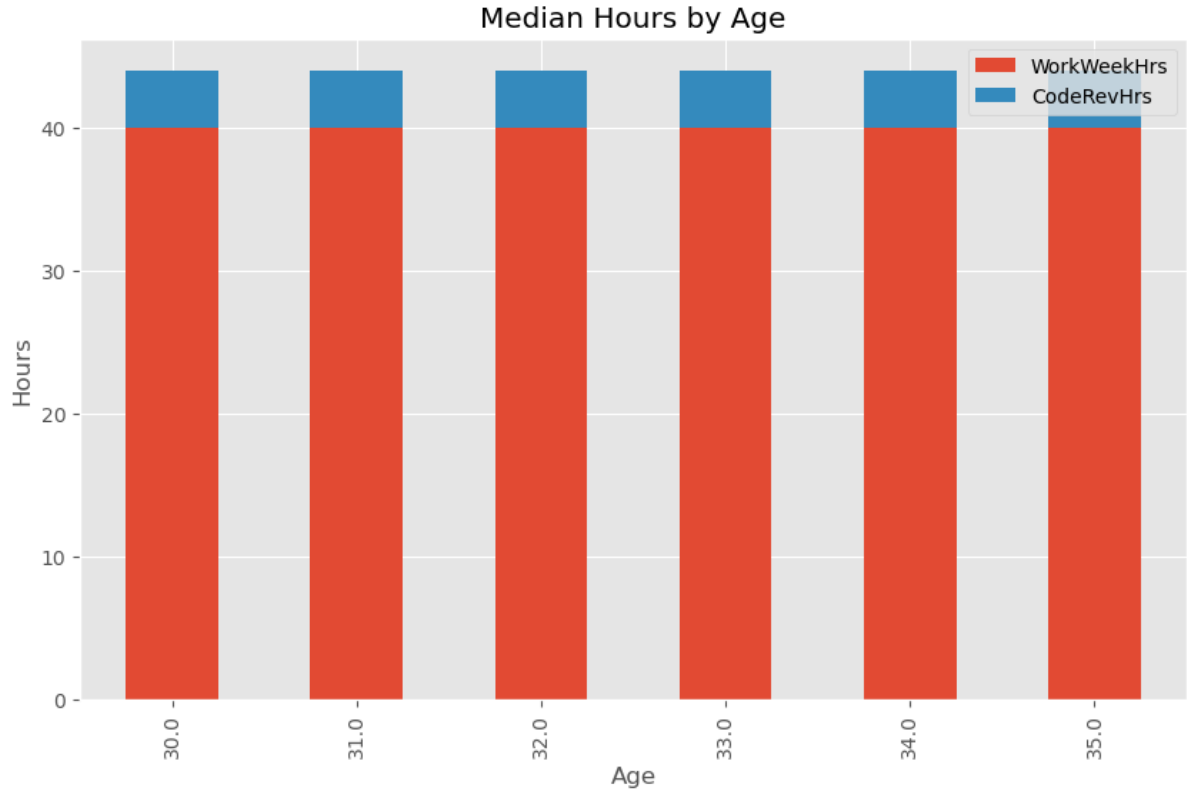
Stacked Charts

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

[15]:

```
# your code goes here

#step 1: get the data needed
QUERY = """
SELECT WorkWeekHrs, CodeRevHrs, Age
FROM master
"""
df_age = pd.read_sql_query(QUERY,conn)
#print(df_age.shape)

# group respondents by age and apply median() function
df_age = df_age.groupby('Age', axis=0).median()
#df_age = df_age[30:35]
#print(df_age.shape)

# step 2: plot data
df_age[30:35].plot(kind='bar', figsize=(10, 6), stacked=True)

plt.xlabel('Age') # add to x-label to the plot
plt.ylabel('Hours') # add y-label to the plot
plt.title('Median Hours by Age') # add title to the plot
```

plt.show()



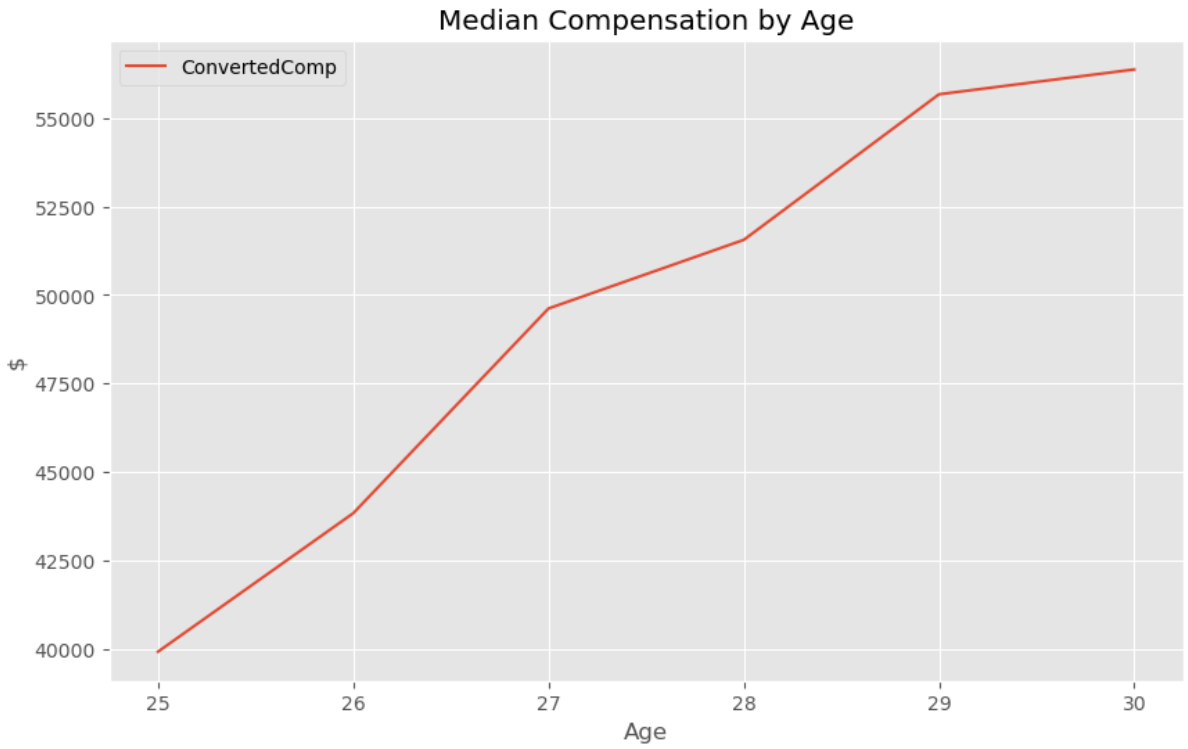
Visualizing comparison of data

Line Chart

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

[16]:

```
# your code goes here
#step 1: get the data needed
QUERY = """
SELECT ConvertedComp, Age
FROM master
"""
df_comp = pd.read_sql_query(QUERY,conn)

# group respondents by age and apply median() function
df_comp = df_comp.groupby('Age', axis=0).median()


# step 2: plot data
df_comp[25:30].plot(kind='line', figsize=(10, 6), stacked=True)

plt.xlabel('Age') # add to x-label to the plot
plt.ylabel('$') # add y-label to the plot
plt.title('Median Compensation by Age') # add title to the plot

plt.show()
```

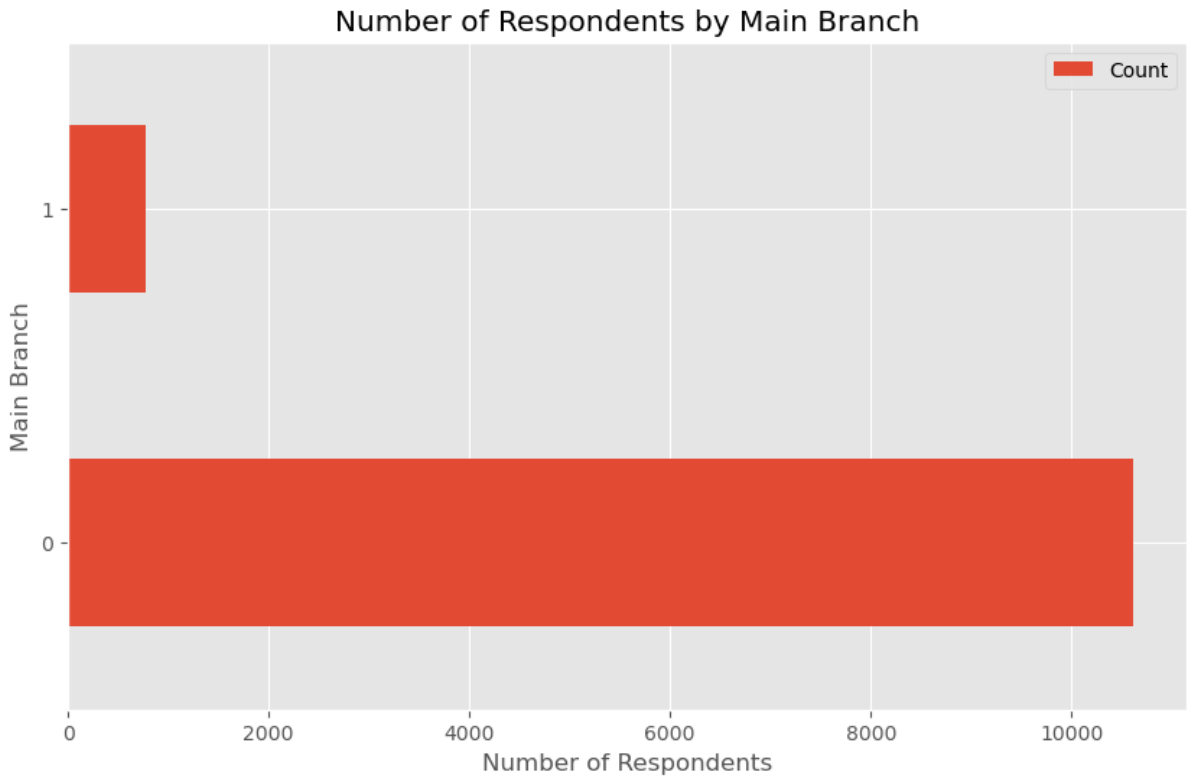


Bar Chart

Create a horizontal bar chart using column MainBranch.

[17]:

```
# your code goes here

#step 1: get the data needed
QUERY = """
SELECT MainBranch, count(MainBranch) as Count
FROM master
GROUP BY MainBranch
"""
df_main = pd.read_sql_query(QUERY,conn)
df_main.head()

# group respondents by age and apply median() function
#df_main = df_main.groupby('Age', axis=0).median()


# step 2: plot data
df_main.plot(kind='barh', figsize=(10, 6))

plt.xlabel('Number of Respondents') # add to x-label to the plot
plt.ylabel('Main Branch') # add y-label to the plot
plt.title('Number of Respondents by Main Branch') # add title to the plot
plt.show()
```



[19]:

```
# your code goes here
table_name = 'DevType'  # 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])

#step 1: get the data needed
QUERY = """
SELECT DevType, count(DevType) as Count
FROM DevType
GROUP BY DevType
"""
df_type = pd.read_sql_query(QUERY,conn)
df_type = df_type.sort_values('Count', ascending=False)
df_type.head()
```

```
---------------------------------------------------------------------------
ProgrammingError                          Traceback (most recent call last)
/tmp/ipykernel_493/476445222.py in <module>
      7 """.format(table_name)
      8
----> 9 df = pd.read_sql_query(QUERY,conn)
     10 print(df.iat[0,0])
     11

~/conda/envs/python/lib/python3.7/site-packages/pandas/io/sql.py in read_sql_query(sql, con, index_
col, coerce_float, params, parse_dates, chunksize, dtype)
    441         parse_dates=parse_dates,
    442         chunksize=chunksize,
```

```
--> 443         dtype=dtype,
  444     )
  445
```

```
~/conda/envs/python/lib/python3.7/site-packages/pandas/io/sql.py in read_query(self, sql, index_col,
coerce_float, params, parse_dates, chunksize, dtype)
 2114
 2115         args = _convert_params(sql, params)
-> 2116         cursor = self.execute(*args)
 2117         columns = [col_desc[0] for col_desc in cursor.description]
 2118
```

```
~/conda/envs/python/lib/python3.7/site-packages/pandas/io/sql.py in execute(self, *args, **kwargs)
 2052             cur = self.con
 2053         else:
-> 2054             cur = self.con.cursor()
 2055         try:
 2056             cur.execute(*args, **kwargs)
```

```
ProgrammingError: Cannot operate on a closed database.
```

Close the database connection.

[18]:

```
conn.close()
```