Workshop 1 - Data analysis with Pandas

By completing this notebook, you will be able to:

- **Program in Python using jupyter notebook.**
- **Perform data analysis using Pandas.**
- **Practice data pre-processing methods.**
- **Analyze and summarise dataset by finding facts from the data.**

In this notebook, you will be using Pandas to read Adult dataset and to perform some basic analysis to improve your understand of the dataset by completing the notebook and answering questions provided in the notebook. You will also be using matplotlib for data visualisation.

The notebook will also introduce you to data pre-processing, which is an important phase of data mining. We will be using sklearn for using data mining (and machine learning) algorithms.

To run the notebook, restart the Kernal by selecting Restart & Clear Output. Than run each cell one at a time.

Notebook submission: **This notebook is a part of your assessment, please complete the notebook by writing and running all the code provided in this notebook; and by writing appropriate codes and description to answer questions provided throughout the notebook including the Report section of the notebook. Save and submit the completed notebook in a readable pdf format.**

Dataset: Adult - https://archive.ics.uci.edu/ml/datasets/Adult

- **Please read Adult webpage carefully including Attribute Information section to familiarise yourself with the data and the data structure.**
- **To download data, click 'Data Folder' and select 'adult.data'. Save the data file as .csv file.**
- **Attributes: You will also need to see 'adult.names' for the attribute names. Insert a row at the top of the dataset and add attribute names to the respective columns. You will notice that the last column has no name. Name the last column as 'class-label'.**

Dr. Vinita Nahar, University of Wolverhampton, UK.

Exploratory analysis: Loading and exploring the dataset

We'll start this workshop, by reading the dataset into a dataframe and performing basic analysis to gain first-hand understanding of the dataset we are working on such as what are the minimum and maximum values in the dataset, whether there are NULL values exist in the dataset, etc.

Before we build the models or form any hypothesis, it is important that we gain these useful insights of the dataset. As it will give us a direction of what type of stories could be discovered for the given dataset.

Pandas library: **Pandas is a python package, which comes very handy while working with data files. It is used for data analysis such as data operation, pre-processing, manipulation and munging.**

In [1]:

```
#Importing necessary libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
#Loading dataset to a dataframe.
```

```
data = pd.read_csv('adult.csv')
```

In [3]:

```
data.head()
```

Out[3]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours per wee
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	4
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	1
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	4
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	4
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	4

Setting path to dataset: Note that for this notebook, 'adult.csv' is placed in the same folder where this jupyter notebook is placed. If you downloaded and placed the dataset in a different folder, path to the dataset needs to be provided then. E.g., if 'adult.csv' is placed in Downloads folder, command would look like:

```
data=pd.read_csv('C:/Users/Vinita/Downloads/adult.csv')
```

You will notice that the data.head() displayed first 5 rows of the dataframe data.

Q1. Use head(2), head(10), tail(2). Explain your observations, in no more than 2 to 3 lines.

In [ ]:

```
#Write your code here.
```

Write your observations in a new cell and select 'Markdown'. 'Markdown' is useful for writing free text e.g., comments.

In [4]:

```
data.shape
```

Out[4]:

```
(32561, 15)
```

The data.shape tells you the dimensionality of the dataset. In this case, there are 32561 rows and 15 columns.

Generating your unique dataset

Please follow these instructions carefully.

For this task, you are required to generate your own version of dataset. To achieve this, replace 448 in ramdom_state with the last three digits of your student number. i.e. '448' to 'last three digits of your student number'. Failing to do so may result in '0' or reduced grades for this task.

In [5]:

In [5]:

```
data = data.sample(n=30000, random_state = 448)
```

In [6]:

```
data.shape
```

Out[6]:

```
(30000, 15)
```

In [7]:

```
data.describe()
```

Out[7]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
count	30000.000000	3.000000e+04	30000.000000	30000.000000	30000.000000	30000.000000
mean	38.600600	1.897832e+05	10.083267	1063.808667	86.985233	40.402300
std	13.636469	1.056454e+05	2.573640	7322.555844	402.201267	12.354095
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178315e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.782635e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370545e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

In [8]:

```
data['education-num'].value_counts()
```

Out[8]:

```
9     9634
10    6726
13    4948
14    1589
11    1286
7     1082
12     987
6      866
4      599
15     529
5      479
8      392
16     380
3      303
2      151
1       49
Name: education-num, dtype: int64
```

In [9]:

```
data['education'].value_counts()
```

Out[9]:

```
 HS-grad         9634
 Some-college    6726
 Bachelors       4948
 Masters         1589
 Assoc-voc       1286
 11th            1082
 Assoc-acdm       987
 10th             866
 7th-8th          599
 Prof-school      529
 9th              479
```

```
 9th             479
 12th            392
 Doctorate       380
 5th-6th         303
 1st-4th         151
 Preschool        49
Name: education, dtype: int64
```

In [10]:

```
data.describe()
```

Out[10]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
count	30000.000000	3.000000e+04	30000.000000	30000.000000	30000.000000	30000.000000
mean	38.600600	1.897832e+05	10.083267	1063.808667	86.985233	40.402300
std	13.636469	1.056454e+05	2.573640	7322.555844	402.201267	12.354095
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178315e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.782635e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370545e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

In [11]:

```
data = data.drop(['fnlwgt'], axis=1)
```

The above cell will drop/remove 'fnlwgt' from data.

drop(): To drop a column from the dataframe, pass agruments - column name to be dropped and axis = 1. axis = 0 is to dropping row.

In [12]:

```
data.shape
```

Out[12]:

```
(30000, 14)
```

Yow will notice that there are now 14 columns instead of 15.

In [13]:

```
data.describe(include='all')
```

Out[13]:

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	c
count	30000.000000	30000	30000	30000.000000	30000	30000	30000	30000	30000	30000.000000	30
unique	NaN	9	16	NaN	7	15	6	5	2	NaN	
top	NaN	Private	HS-grad	NaN	Married-civ-spouse	Prof-specialty	Husband	White	Male	NaN	
freq	NaN	20897	9634	NaN	13835	3802	12176	25612	20080	NaN	
mean	38.600600	NaN	NaN	10.083267	NaN	NaN	NaN	NaN	NaN	1063.808667	
std	13.636469	NaN	NaN	2.573640	NaN	NaN	NaN	NaN	NaN	7322.555844	
min	17.000000	NaN	NaN	1.000000	NaN	NaN	NaN	NaN	NaN	0.000000	

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	c
25%	28.000000	NaN	NaN	9.000000	NaN	NaN	NaN	NaN	NaN	0.000000	
50%	37.000000	NaN	NaN	10.000000	NaN	NaN	NaN	NaN	NaN	0.000000	
75%	48.000000	NaN	NaN	12.000000	NaN	NaN	NaN	NaN	NaN	0.000000	
max	90.000000	NaN	NaN	16.000000	NaN	NaN	NaN	NaN	NaN	99999.000000	4

In [14]:

```
data['education'].value_counts()
```

Out[14]:

```
 HS-grad         9634
 Some-college    6726
 Bachelors       4948
 Masters         1589
 Assoc-voc       1286
 11th            1082
 Assoc-acdm       987
 10th             866
 7th-8th          599
 Prof-school      529
 9th              479
 12th             392
 Doctorate        380
 5th-6th          303
 1st-4th          151
 Preschool         49
Name: education, dtype: int64
```

value_counts() produces a frequency table, which shows occurance of each feature or attribute in a dataset.

In [15]:

```
data['education'].nunique()
```

Out[15]:

```
16
```

In [16]:

```
data['age'].value_counts()
```

Out[16]:

```
36    847
34    819
31    819
35    803
33    803
     ...
83      6
88      3
85      3
86      1
87      1
Name: age, Length: 73, dtype: int64
```

You may agree that using value_counts() for 'age' is not a great example as the frequency table is too lengthy to be analysed. This is due to 'age' being continuous value and frequency of each value is displayed. Let's visualise 'age' through graphs instead to make observations.

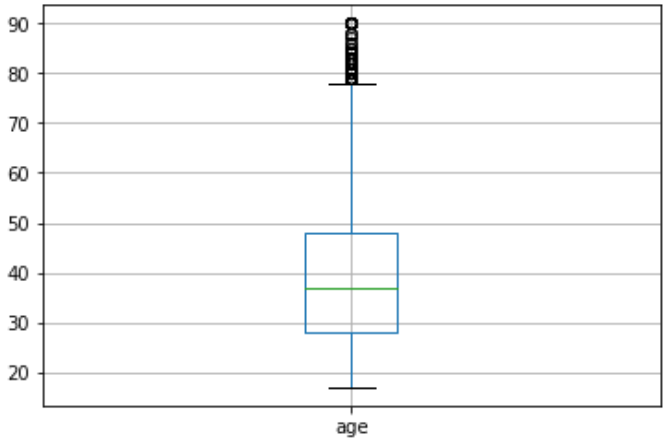
In [17]:

```
data.boxplot(column='age')
```

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x218df0a5a20>
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x218df0a5a20>
```

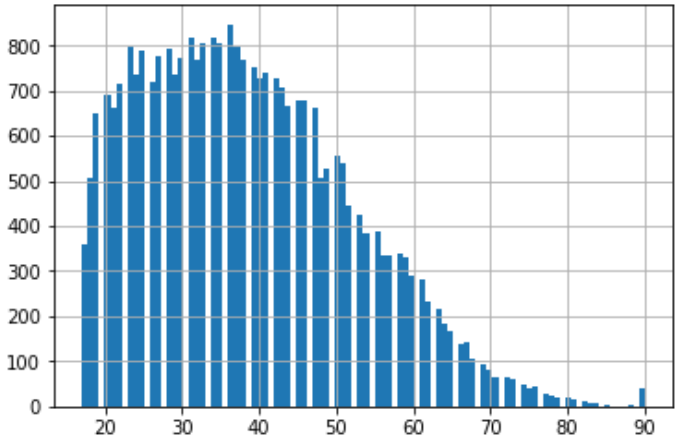


In [18]:

```
data['age'].hist(bins=100)
```

Out[18]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x218df8e49e8>
```



In [19]:

```
data['sex'].value_counts()
```

Out[19]:

```
 Male      20080
 Female     9920
Name: sex, dtype: int64
```

In [20]:

```
data.columns
```

Out[20]:

```
Index(['age', 'workclass', 'education', 'education-num', 'marital-status',
       'occupation', 'relationship', 'race', 'sex', 'capital-gain',
       'capital-loss', 'hours-per-week', 'native-country', 'class-label'],
      dtype='object')
```

In [21]:

```
data['workclass'].value_counts()
```

Out[21]:

```
 Private             20897
 Self-emp-not-inc     2351
 Local-gov            1925
 ?                    1682
 State-gov            1208
 Self-emp-inc         1033
```

```
 Self-emp-inc          1033
 Federal-gov            884
 Without-pay             13
 Never-worked             7
Name: workclass, dtype: int64
```

Q2. How many males and females exist in the dataset? In a new cell, use a correct command to answer the question and write your answer.

In [ ]:

```
```

Applying groupby functions in order to summarise the data.

Groupby functions are usually used with aggregate functions, which are useful to summarise the dataset and make observations. Some common functions are SUM, MEAN, MAX, MIN and COUNT. Using groupby, we can answer questions such as:

Question: What is the average age of each gender in the given population?

In [22]:

```
data['age'].groupby([data['sex']]).mean()
```

Out[22]:

```
sex
 Female    36.945262
 Male      39.418376
Name: age, dtype: float64
```

In the above cell, we group by 'sex' and computed the mean 'age'.

Question. What is the average age of male and female across different eduction categories?

In [23]:

```
data['age'].groupby([data['sex'],data['education']]).mean()
```

Out[23]:

```
sex      education
 Female   10th            35.952727
          11th            30.201550
          12th            30.248120
          1st-4th         48.400000
          5th-6th         44.628205
          7th-8th         49.888889
          9th             42.231343
          Assoc-acdm      36.300518
          Assoc-voc       37.942675
          Bachelors       35.736559
          Doctorate       44.578947
          HS-grad         38.692555
          Masters         43.272912
          Preschool       41.750000
          Prof-school     40.060241
          Some-college    33.885240
 Male     10th            38.485618
          11th            33.230216
          12th            32.907336
          1st-4th         45.311321
          5th-6th         41.773333
          7th-8th         47.878924
          9th             40.628986
          Assoc-acdm      37.856905
          Assoc-voc       38.883436
          Bachelors       40.294509
```

```
           Bachelors       40.294509
           Doctorate       48.125000
           HS-grad         39.130868
           Masters         44.578324
           Preschool       43.181818
           Prof-school     45.786996
           Some-college    37.033833
Name: age, dtype: float64
```

In the above code, we group by 'sex' and 'education' and computed mean 'age' in the given population.

NOTE: groupby can be applied on numeric attributes only.

For some simple examples on groupby, please refer to the link - http://www.datasciencemadesimple.com/group-dataframe-python-pandas-group-function-pandas/

Q3. What is the average contribution to capital-gain of each sex and occupation category?

In [ ]:

```
```

Q4. Identify the average capital-gain by males and females accross different marital-status.

In [ ]:

```
```

Question. What is the maximum age accross differnt races?

Let's first see what are the different races and then apply groupby.

In [24]:

```
data['race'].value_counts()
```

Out[24]:

```
 White                 25612
 Black                  2894
 Asian-Pac-Islander      956
 Amer-Indian-Eskimo      288
 Other                   250
Name: race, dtype: int64
```

In [25]:

```
data['age'].groupby([data['race']]).max()
```

Out[25]:

```
race
 Amer-Indian-Eskimo    82
 Asian-Pac-Islander    90
 Black                 90
 Other                 77
 White                 90
Name: age, dtype: int64
```

Q5. Minimum and maximum age by sex are same?

In [ ]:

```
#Minimum age by sex:
```

In [ ]:

```
#maximum age by sex:
```

```
#maximum age by sex:
```

Write your answer in a new 'Markdown' cell.

Data visualisation

Matplotlib is python library for visualising data in the form of graphs such as histograms, scatter, box plot, line plots, heat plots, etc.

In [26]:

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

In [27]:

```
data.describe()
```

Out[27]:

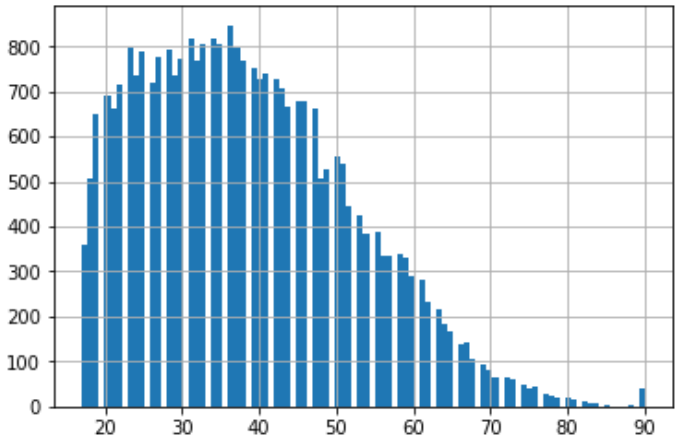
	age	education-num	capital-gain	capital-loss	hours-per-week
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000
mean	38.600600	10.083267	1063.808667	86.985233	40.402300
std	13.636469	2.573640	7322.555844	402.201267	12.354095
min	17.000000	1.000000	0.000000	0.000000	1.000000
25%	28.000000	9.000000	0.000000	0.000000	40.000000
50%	37.000000	10.000000	0.000000	0.000000	40.000000
75%	48.000000	12.000000	0.000000	0.000000	45.000000
max	90.000000	16.000000	99999.000000	4356.000000	99.000000

In [28]:

```
data['age'].hist(bins=100)
```

Out[28]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x218dfa42198>
```



Histograms is used to represent the distribution of dataset. The bars of the histograms are known as bins or "bucket" – the range of values. Bins are of same width. Width of the bins can be calculated as (max value of data – min value of data) / total number of bins. The bins are usually specified as continuous, non-overlapping intervals of a variable.

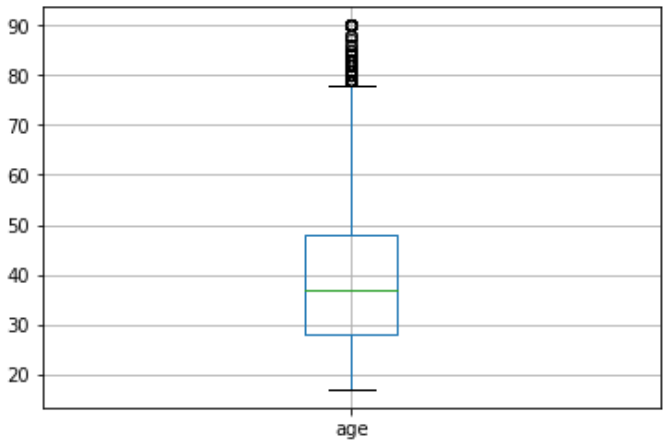
In the above figure, histogram with bins = 50 is used to show number of peolpe belongs to different age-groups. Here, x-axis represents 'age' and y-axis represents the 'count'. *Try-it-yourself:* change bins = 100 and run the cell to observe the difference for your own understanding.

In [29]:

```
data.boxplot(column='age')
```

Out[29]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x218e1b97358>
```



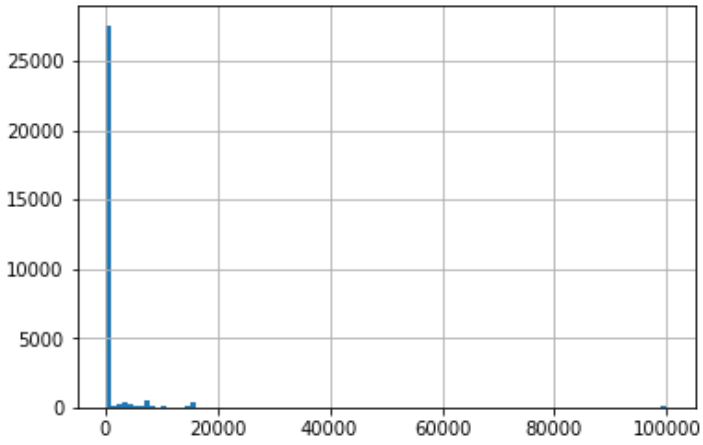
In the above figure, boxplot is used to find the average number of people belongs to which age-range group.

In [30]:

```
data['capital-gain'].hist(bins=100)
```

Out[30]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x218e2c23160>
```

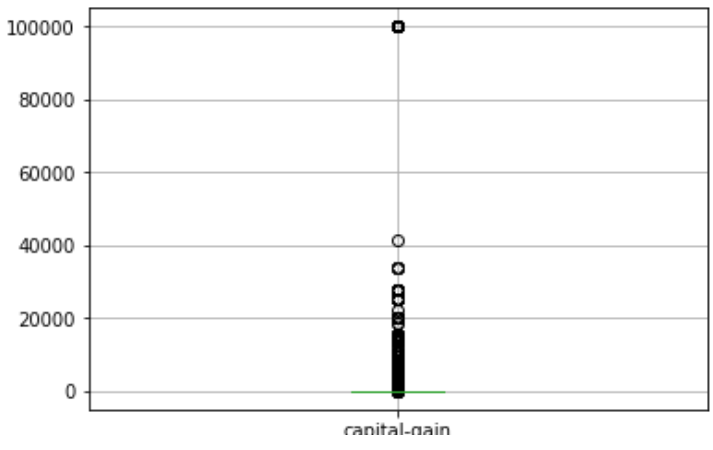


In [31]:

```
data.boxplot(column='capital-gain')
```

Out[31]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x218e2d5c400>
```



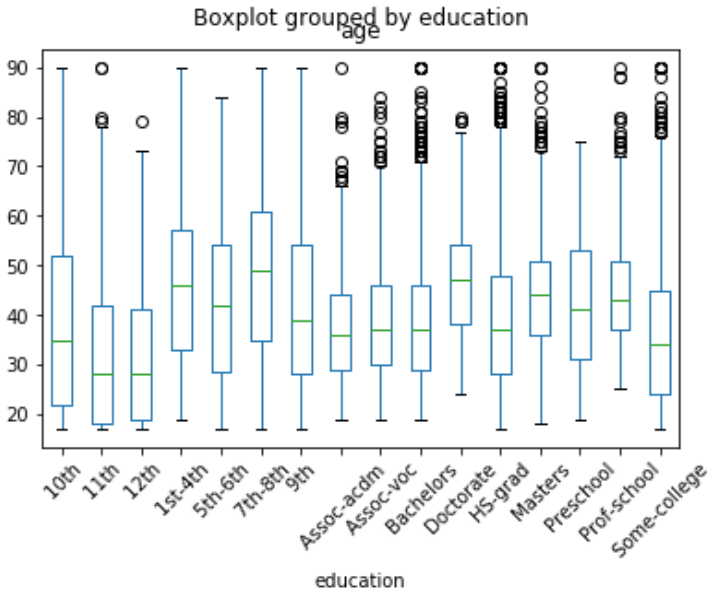
In [32]:

```
data.boxplot(column='age', by = 'education', grid=False, rot = 45, fontsize = 10)
```

```
c:\users\vinita\appdata\local\programs\python\python37\lib\site-packages\matplotlib\cbook
\__init__.py:1395: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequ
ences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or
shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when crea
ting the ndarray.
  X = np.atleast_1d(X.T if isinstance(X, np.ndarray) else np.asarray(X))
```

Out[32]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x218e2dd9668>
```



In [33]:

```
data['education'].value_counts()
```

Out[33]:

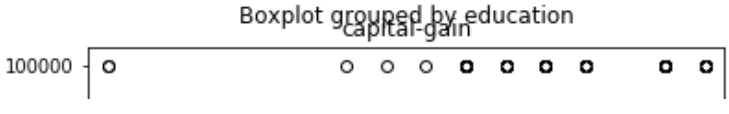
```
 HS-grad         9634
 Some-college    6726
 Bachelors       4948
 Masters         1589
 Assoc-voc       1286
 11th            1082
 Assoc-acdm       987
 10th             866
 7th-8th          599
 Prof-school      529
 9th              479
 12th             392
 Doctorate        380
 5th-6th          303
 1st-4th          151
 Preschool         49
Name: education, dtype: int64
```

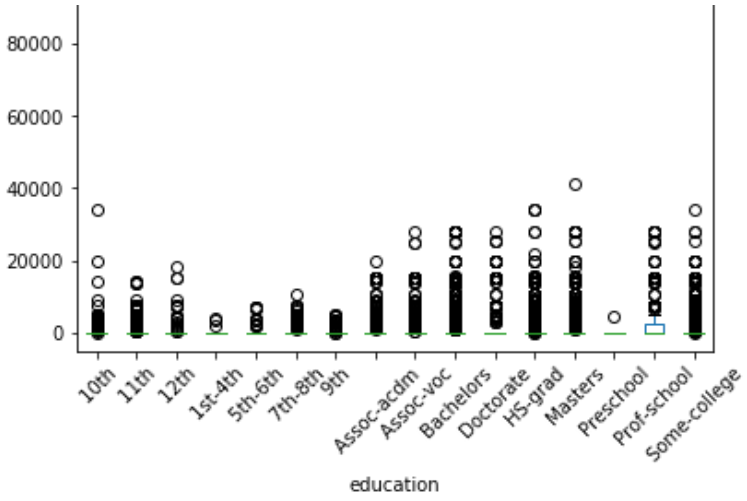
In [34]:

```
data.boxplot(column='capital-gain', by = 'education', grid=False, rot = 45, fontsize = 1
0)
```

Out[34]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x218e2f91390>
```





After performing some basic data analysis, let's look at data pre-processing to improve the quality of the dataset.

Data pre-processing is an important step in the process. Raw data can be unstructured and full of noise. Aim of this phase is to clean the raw data, reduce noise and to prepare the dataset that can be accepted by the algorithm as an input. Remember garbage in, garbage out!

In [35]:

```
data['marital-status'].value_counts()
```

Out[35]:

```
 Married-civ-spouse       13835
 Never-married             9788
 Divorced                  4131
 Separated                  928
 Widowed                    909
 Married-spouse-absent      390
 Married-AF-spouse           19
Name: marital-status, dtype: int64
```

Checking NULL values in the dataset

In [36]:

```
data.apply(lambda x: sum(x.isnull()), axis = 0)
```

Out[36]:

```
age               0
workclass         0
education         0
education-num     0
marital-status    0
occupation        0
relationship      0
race              0
sex               0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
class-label       0
dtype: int64
```

You will notice that the missing values are not picked up by this code. As the NULL or missing values are replaced by '?'. It is important that we treat NULL or missing values in our dataset, which is usually done in a data pre-processing phase of data mining. In this workshop, we ignore this step as the values are already being replaced by '?'. In the following workshops, we will see this step in the great details.

Data transformation

Label encoding:

Some attributes are categorical, therefore (statistical) analysis on those variables is not possible. We need to convert all categorical variables (string labels) into numeric by encoding the categories. Package 'sklearn' provides 'LabelEncoder' library for encoding labels between 0 to n-1 discrete values/labels, where n is the number of values/labels. E.g.:

```
Male -> 0
Female -> 1
```

In [37]:

```
from sklearn.preprocessing import LabelEncoder
```

In [38]:

```
data.head()
```

Out[38]:

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week
18418	35	Private	HS-grad	9	Married-civ-spouse	Adm-clerical	Wife	White	Female	0	1887	42
25529	45	Private	Some-college	10	Separated	Craft-repair	Not-in-family	White	Male	0	0	41
8224	29	Federal-gov	Prof-school	15	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	0	80
30612	37	Private	HS-grad	9	Never-married	Craft-repair	Own-child	White	Male	0	0	40
23960	37	Private	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	40

In [39]:

```
data.dtypes
```

Out[39]:

```
age               int64
workclass        object
education        object
education-num     int64
marital-status   object
occupation       object
relationship     object
race             object
sex              object
capital-gain      int64
capital-loss      int64
hours-per-week    int64
native-country   object
class-label      object
dtype: object
```

In [40]:

```
columns = list(data.select_dtypes(exclude=['int64']))
```

In [41]:

```
columns
```

Out[41]:

```
['workclass',
 'education',
 'marital-status',
 'occupation',
 'relationship',
 'race',
 'sex',
 'native-country',
 'class-label']
```

In [42]:

```
data['class-label'].value_counts()
```

Out[42]:

```
 <=50K    22779
 >50K      7221
Name: class-label, dtype: int64
```

In [44]:

```
le = LabelEncoder()
for i in columns:
    #print(i)
    data[i] = le.fit_transform(data[i])

data.dtypes
```

Out[44]:

```
age               int64
workclass         int32
education         int32
education-num     int64
marital-status    int32
occupation        int32
relationship      int32
race              int32
sex               int32
capital-gain      int64
capital-loss      int64
hours-per-week    int64
native-country    int32
class-label       int32
dtype: object
```

In [45]:

```
data.head()
```

Out[45]:

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country
18418	35	4	11	9	2	1	5	4	0	0	1887	42	39
25529	45	4	15	10	5	3	1	4	1	0	0	41	39
8224	29	1	14	15	2	10	0	4	1	0	0	80	39
30612	37	4	11	9	4	3	3	4	1	0	0	40	39
23960	37	4	11	9	2	5	0	4	1	0	0	40	39

In [46]:

```
data['workclass'].value_counts()
```

Out[46]:

```
4    20897
6     2351
2     1925
0     1682
7     1208
5     1033
1      884
8       13
3        7
Name: workclass, dtype: int64
```

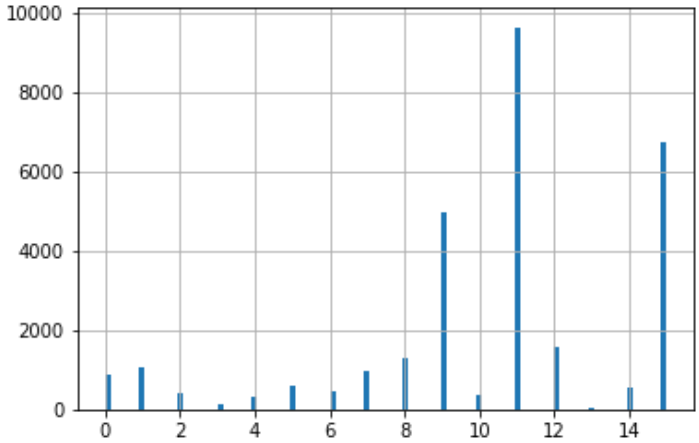
You will notice that all the values are now numeric. Now, more computation and analysis can be performed on the dataset.

In [47]:

```
data['education'].hist(bins=100)
```

Out[47]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x218ded82e48>
```



In [48]:

```
data.describe(include='all')
```

Out[48]:

	age	workclass	education	education-num	marital-status	occupation	relationship	race	
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	300
mean	38.600600	3.872467	10.296800	10.083267	2.603500	6.561233	1.447233	3.664733	
std	13.636469	1.456093	3.871556	2.573640	1.506175	4.224529	1.609524	0.849835	
min	17.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	
25%	28.000000	4.000000	9.000000	9.000000	2.000000	3.000000	0.000000	4.000000	
50%	37.000000	4.000000	11.000000	10.000000	2.000000	7.000000	1.000000	4.000000	
75%	48.000000	4.000000	12.000000	12.000000	4.000000	10.000000	3.000000	4.000000	
max	90.000000	8.000000	15.000000	16.000000	6.000000	14.000000	5.000000	4.000000	

Report

Answer the following questions. You are required to include correct code(s) to answer the questions.

Answer the following questions. You are required to include correct code(s) to answer the questions.

Q6. Write a summary of the outcome of data.describe().

Q7. What are the different data types (or attribut types) in data mining? Explain with the help of the examples from Adult dataset. **HINT:** Don't get confused with data types in Python.

Q8. Highest migrants belongs to which country?

Q9. Which occupation represents more males than females?

Q10. What is the difference between data.head() and data.tail()?

End of the Workshop 1.