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 analysis and pre-processing**, which and important phases of data mining.

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

Task: This notebook is a part of your assessment. This Notebook demonstrates various processes of data analysis step-by-step. You are required to perform these processes in a new Notebook on Adult dataset. Complete all the sections of 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 Try-it-yourself and 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 (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, summarising data using 'groupby'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')
```

Setting path to dataset: Note that for this notebook, 'adult.csv' is placed in the working folder i.e., same folder where this jupyter notebook is placed. If you downloaded and placed the dataset in a different folder such as Downloads, 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')
```

In [3]:

```
data.head()
```

Out[3]:

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

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., answers/comments/description.

In [4]:

```
data.shape
```

Out[4]:

```
(32561, 15)
```

The DataFrame.shape property tells you the dimensionality of the dataset (Pandas DataFrame) in the form of number of rows and columns. In this case, there are 32561 rows and 15 columns.

Generating your unique dataset for this task

Please follow these instructions carefully.

For this task, you are required to generate your own version of dataset. To achieve this, replace 48 in ramdom_state with the last two digits of your student number. i.e.'48' → 'last two digits of your student number'. If the first number starts with '0', replace '0' with '2'. E.g., if the last two digits of your student number is '05', use '25' instead. Failing to do so may result in '0' or reduced grades for this task.

In [5]:

```
data = data.sample(n=30000, random_state = 48)
```

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	38.601733	1.897456e+05	10.078533	1072.036100	86.703967	40.409433
std	13.645910	1.058005e+05	2.571715	7387.730029	401.942288	12.349540
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.177670e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783090e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.368742e+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     9678
10    6724
13    4907
14    1600
11    1271
7     1091
12     985
6      869
4      589
15     526
5      471
8      401
16     381
3      301
2      157
1       49
Name: education-num, dtype: int64
```

In [9]:

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

Out[9]:

```
 HS-grad         9678
 Some-college    6724
 Bachelors       4907
 Masters         1600
 Assoc-voc       1271
 11th            1091
 Assoc-acdm       985
 10th             869
 7th-8th          589
 Prof-school      526
 9th              471
 12th             401
 Doctorate        381
 5th-6th          301
 1st-4th          157
 Preschool         49
Name: education, dtype: int64
```

In [10]:

```
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 [11]:

```
data.shape
```

Out[11]:

```
(30000, 14)
```

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

In [12]:

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

Out[12]:

	age	workclass	education	education-num	marital-status	occupation	relationship	
count	30000.000000	30000	30000	30000.000000	30000	30000	30000	3
unique	NaN	9	16	NaN	7	15	6	
top	NaN	Private	HS-grad	NaN	Married-civ-spouse	Prof-specialty	Husband	\
freq	NaN	20897	9678	NaN	13807	3834	12170	2
mean	38.601733	NaN	NaN	10.078533	NaN	NaN	NaN	
std	13.645910	NaN	NaN	2.571715	NaN	NaN	NaN	
min	17.000000	NaN	NaN	1.000000	NaN	NaN	NaN	
25%	28.000000	NaN	NaN	9.000000	NaN	NaN	NaN	
50%	37.000000	NaN	NaN	10.000000	NaN	NaN	NaN	
75%	48.000000	NaN	NaN	12.000000	NaN	NaN	NaN	
max	90.000000	NaN	NaN	16.000000	NaN	NaN	NaN	

In [13]:

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

Out[13]:

```
 HS-grad         9678
 Some-college    6724
 Bachelors       4907
 Masters         1600
 Assoc-voc       1271
 11th            1091
 Assoc-acdm       985
 10th             869
 7th-8th          589
 Prof-school      526
 9th              471
 12th             401
 Doctorate        381
 5th-6th          301
 1st-4th          157
 Preschool         49
Name: education, dtype: int64
```

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

In [14]:

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

Out[14]:

```
16
```

In [15]:

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

Out[15]:

```
36    832
34    830
31    815
35    809
33    808
     ...
83      6
85      3
88      2
87      1
86      1
Name: age, Length: 73, dtype: int64
```

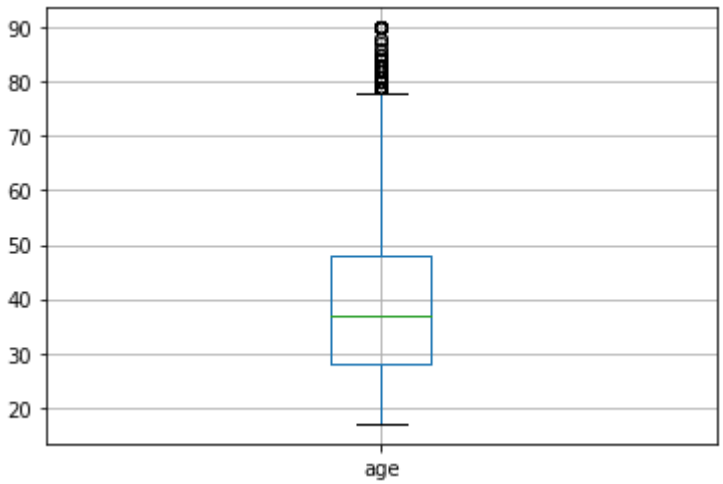
You may agree that using value_counts() for ‘age’ is not a great example as the frequency table is too lengthy to be analysed. This is due to 'age' being continuous value and frequency of each value is displayed. Let's visualise 'age' through graphs instead to make observations.

In [16]:

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

Out[16]:

```
<AxesSubplot:>
```

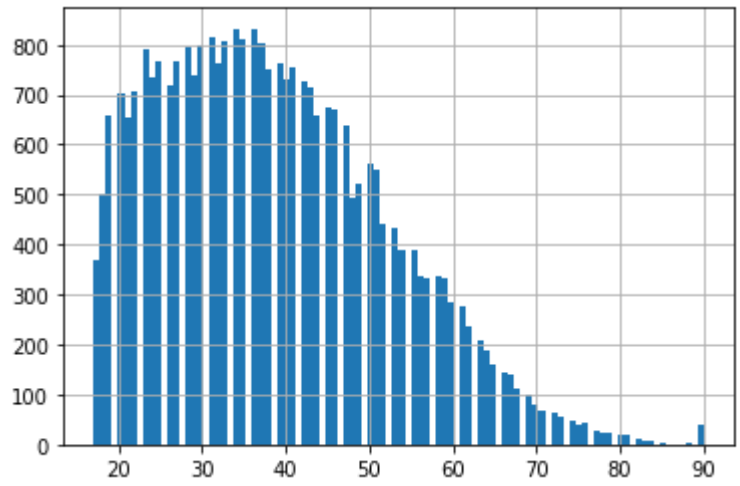


In [17]:

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

Out[17]:

<AxesSubplot:>

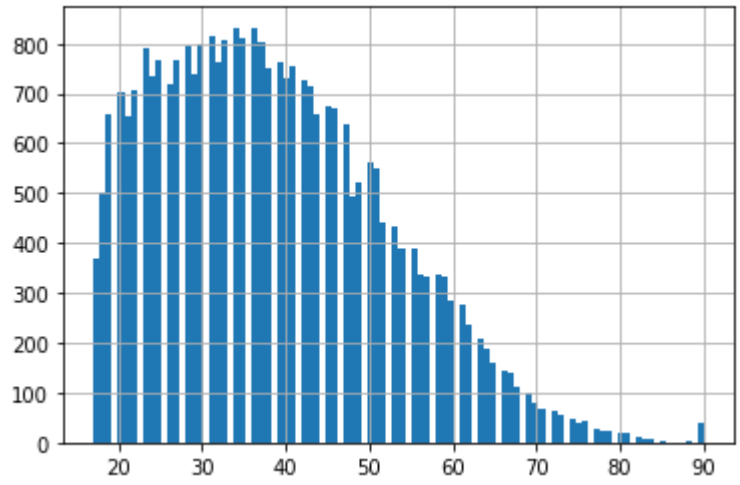


In [18]:

```
data.age.hist(bins=100)
```

Out[18]:

<AxesSubplot:>



In [19]:

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

Out[19]:

```
 Male      20079
 Female     9921
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     2339
 Local-gov            1931
 ?                    1693
 State-gov            1204
 Self-emp-inc         1031
 Federal-gov           885
 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    36.908477
 Male      39.438368
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 [24]:

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

Out[24]:

```
sex      education
 Female   10th            34.946429
          11th            30.591584
          12th            29.894737
          1st-4th         49.125000
          5th-6th         43.881579
          7th-8th         50.609589
          9th             41.731343
          Assoc-acdm      36.444444
          Assoc-voc       37.810870
          Bachelors       35.722410
          Doctorate       45.573171
          HS-grad         38.780519
          Masters         43.202840
          Preschool       42.071429
          Prof-school     40.204819
          Some-college    33.775762
 Male     10th            38.512733
          11th            33.621543
          12th            33.421642
          1st-4th         45.666667
          5th-6th         42.351111
          7th-8th         47.733634
          9th             40.652819
          Assoc-acdm      37.856187
          Assoc-voc       39.000000
          Bachelors       40.412536
          Doctorate       48.183946
          HS-grad         39.165930
          Masters         44.288166
          Preschool       43.228571
          Prof-school     45.343115
          Some-college    36.946286
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/ (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 [25]:

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

Out[25]:

```
 White                 25608
 Black                  2894
 Asian-Pac-Islander      962
 Amer-Indian-Eskimo      281
 Other                   255
Name: race, dtype: int64
```

In [26]:

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

Out[26]:

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

Q5. Are minimum and maximum age by sex same?

In [ ]:

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

In [ ]:

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

Write your answer in a new 'Markdown' cell.

In [ ]:

```

```

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 [27]:

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

In [28]:

```
data.describe()
```

Out[28]:

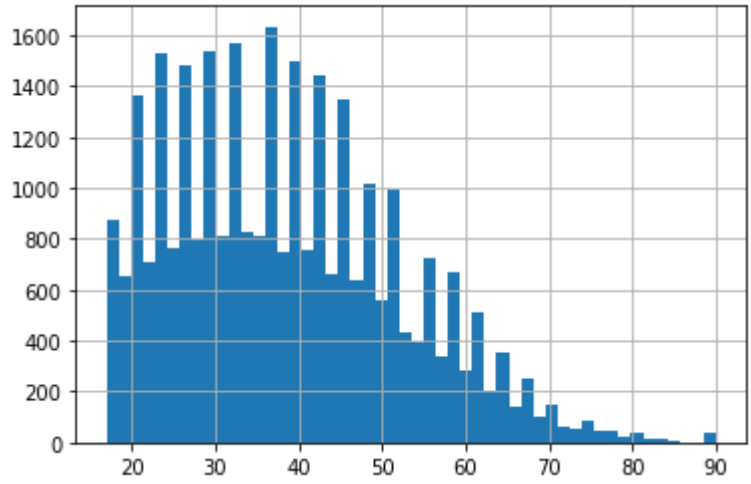
	age	education-num	capital-gain	capital-loss	hours-per-week
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000
mean	38.601733	10.078533	1072.036100	86.703967	40.409433
std	13.645910	2.571715	7387.730029	401.942288	12.349540
min	17.000000	1.000000	0.000000	0.000000	1.000000
25%	28.000000	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 [29]:

```
data['age'].hist(bins=50)
```

Out[29]:

```
<AxesSubplot:>
```



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.

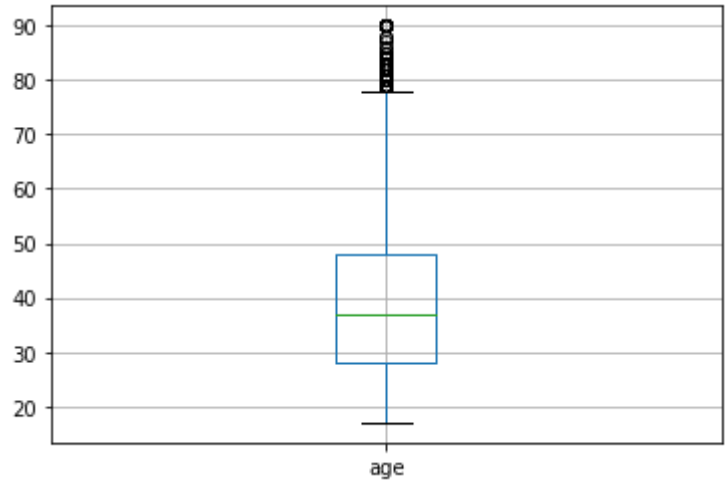
In the above figure, histogram with bins = 50 is used to show number of peolpe belongs to different age-groups. Here, x-axis represents 'age' and y-axis represents the 'count'. ***Try-it-yourself:*** change bins = 100 and run the cell to observe the difference for your own understanding.

In [30]:

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

Out[30]:

<AxesSubplot:>



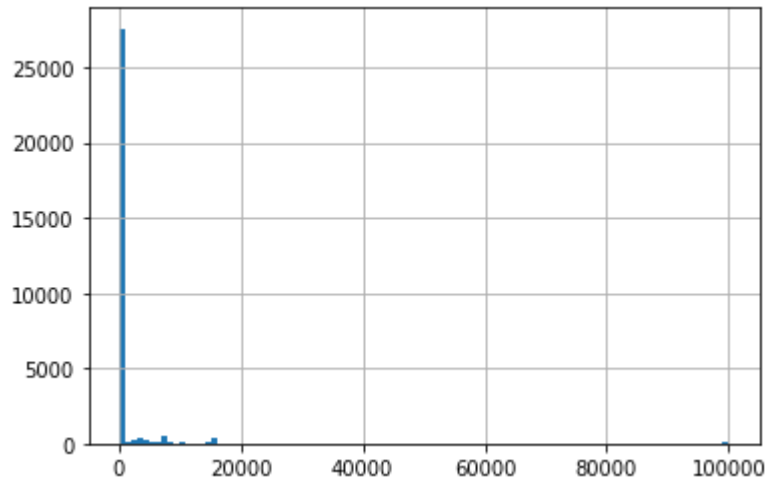
In the above figure, boxplot is used to find the average number of people belongs to which age-range group.

In [31]:

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

Out[31]:

<AxesSubplot:>

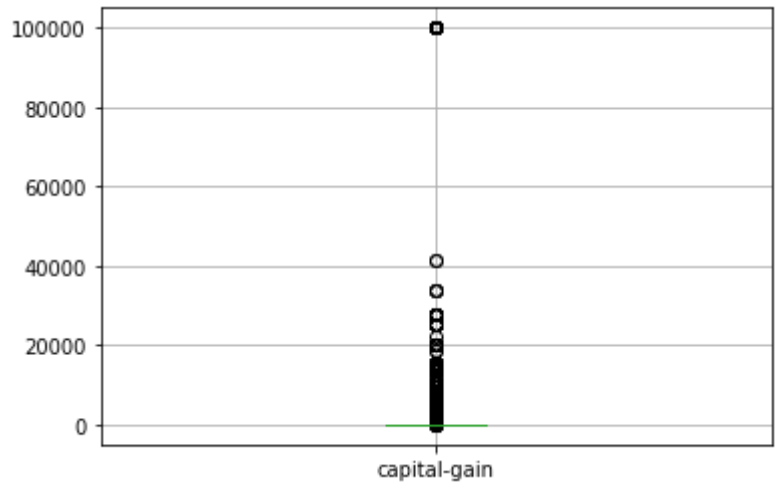


In [32]:

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

Out[32]:

<AxesSubplot:>



In [33]:

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

Out[33]:

<AxesSubplot:title={'center':'age'}, xlabel='education'>

In [34]:

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

Out[34]:

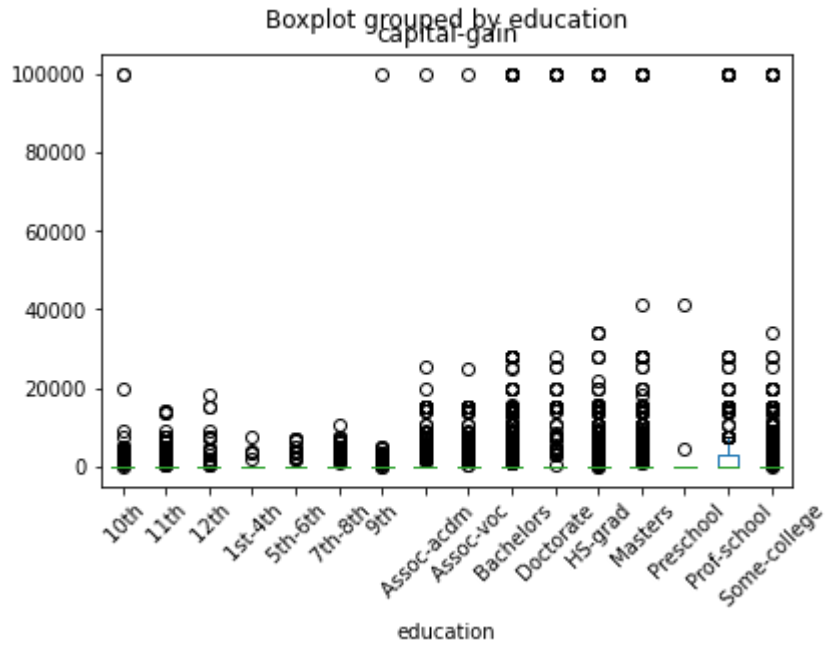
```
 HS-grad         9678
 Some-college    6724
 Bachelors       4907
 Masters         1600
 Assoc-voc       1271
 11th            1091
 Assoc-acdm       985
 10th             869
 7th-8th          589
 Prof-school      526
 9th              471
 12th             401
 Doctorate        381
 5th-6th          301
 1st-4th          157
 Preschool         49
Name: education, dtype: int64
```

In [35]:

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

Out[35]:

```
<AxesSubplot:title={'center':'capital-gain'}, xlabel='education'>
```



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 [36]:

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

Out[36]:

```
 Married-civ-spouse       13807
 Never-married             9818
 Divorced                  4093
 Separated                  957
 Widowed                    924
 Married-spouse-absent      381
 Married-AF-spouse           20
Name: marital-status, dtype: int64
```

Checking NULL values in the dataset

In [37]:

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

Out[37]:

```
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 [38]:

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

In [39]:

```
data.head()
```

Out[39]:

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex
22453	24	Private	Some-college	10	Never-married	Adm-clerical	Not-in-family	White	Female
14244	41	Private	HS-grad	9	Married-civ-spouse	Craft-repair	Husband	White	Male
6229	36	Private	HS-grad	9	Never-married	Adm-clerical	Not-in-family	Black	Female
31328	31	Private	HS-grad	9	Married-civ-spouse	Other-service	Husband	White	Male
21040	30	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	White	Male

In [40]:

```
data.dtypes
```

Out[40]:

```
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 [41]:

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

In [42]:

```
columns
```

Out[42]:

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

In [43]:

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

Out[43]:

```
 <=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	ca
22453	24	4	15	10	4	1	1	4	0	
14244	41	4	11	9	2	3	0	4	1	
6229	36	4	11	9	4	1	1	2	0	
31328	31	4	11	9	2	8	0	4	1	
21040	30	4	9	13	2	10	0	4	1	

In [46]:

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

Out[46]:

```
4    20897
6     2339
2     1931
0     1693
7     1204
5     1031
1      885
8       13
3        7
Name: workclass, dtype: int64
```

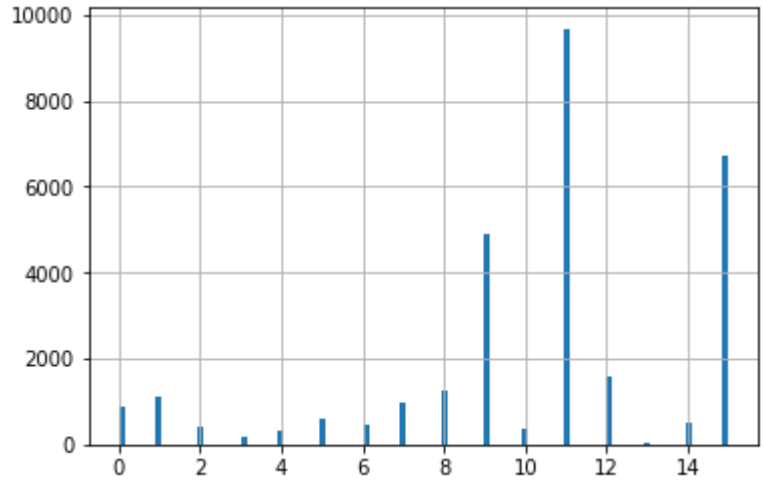
You will notice that all the values are now numeric. Now, more computations and analysis can be performed on the dataset.

In [47]:

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

Out[47]:

<AxesSubplot:>



In [48]:

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

Out[48]:

	age	workclass	education	education-num	marital-status	occupation	r
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30
mean	38.601733	3.869233	10.296467	10.078533	2.612600	6.577933	
std	13.645910	1.457213	3.877260	2.571715	1.507602	4.231957	
min	17.000000	0.000000	0.000000	1.000000	0.000000	0.000000	
25%	28.000000	4.000000	9.000000	9.000000	2.000000	3.000000	
50%	37.000000	4.000000	11.000000	10.000000	2.000000	7.000000	
75%	48.000000	4.000000	12.000000	12.000000	4.000000	10.000000	
max	90.000000	8.000000	15.000000	16.000000	6.000000	14.000000	

Report

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 or Pandas.

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.

In [ ]: