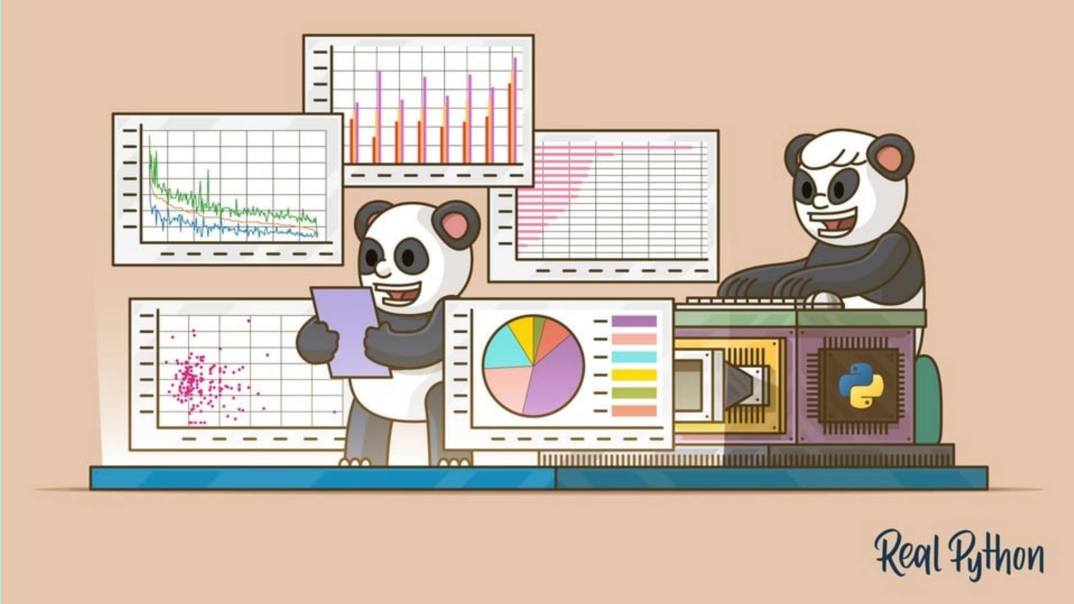


Data Analysis In The Banking Sector

Pandas Fundamentals

• Marème Dickel DIA •

The objective of this project is to master exploratory data analysis in the banking sector with Pandas librairy.

After completing this project, we will be able to:

A- Conduct an exploratory analysis of a bank dataset with Pandas library.

B- Build cross tables and pivot tables.

C- View the dataset on different graphs.

Summary

Banking Dataset Analysis

1) Project description

2) Analysis steps

- Import necessary librairies
- Exploratory Dataset Analysis
- Pivot table
- Visualization in Pandas
- Additionnal questions

---

1) Project description

The data we will use in this project are from an open-source set of bank marketing data from the UCI ML repository:

https://archive.ics.uci.edu/ml/citation_policy.html

During the work, the task of preliminary analysis of a positive response (term deposit) to a bank's direct calls is resolved. The task to be met is therefore a question of bank rating or bank scoring, that is to say that according to the characteristics of a client (potential client), his behaviour is foreseen (default of payment, desire to make a deposit, etc.).

Throughout the project, we will try to answer a set of questions that may be relevant when analyzing bank data:

1. What is the proportion of customers attracted ?
2. What are the average values of the numerical features among the attracted clients ?
3. What is the average duration of calls for attracted clients ?
4. What is the average age of attracted and unmarried clients ?
5. What is the average age and duration of calls for different types of client employment ?

In addition, we will conduct visual analysis to more effectively plan bank marketing campaigns.

Pandas is a Python library that provides many ways to analyze data. Data scientists often work with data stored in table formats like . csv, . tsv, or .xlsx. Pandas is very handy for loading, processing and analyzing these tabular data using SQL-like queries. Together with Matplotlib and Seaborn, Pandas offers a wide range of possibilities for visual analysis of tabular data.

---

2) Analysis steps

Import librairies

Download the data via the link below:

https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/EDA_Pandas_Banking_L1/bank-additional.zip
We will set a default size for graphs and ignore warnings.

In [8]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

%matplotlib inline
plt.rcParams["figure.figsize"] = (8, 6)

import warnings
warnings.filterwarnings('ignore')
```

Let's set the display to two (2) decimal digits for decimal values, eighty (80) for the number of columns and one million (1,000,000) for the number of rows.

In [18]:
```
pd.set_option('display.float_format', lambda x: '{:.2f}'.format(x))
pd.options.display.max_columns = 80
pd.options.display.max_rows = 1000000
```

Exploratory Dataset Analysis

In this section we will load our dataframe and explore it.

Let's display the first five (5) lines of our default dataframe with head() method.

In [34]:
```python
banking = pd.read_csv('bank-additional-full.csv', sep = ';')
banking.head()
```

Out[34]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	pout
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	261	1	999	0	none›
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	149	1	999	0	none›
2	37	services	married	high.school	no	yes	no	telephone	may	mon	226	1	999	0	none›
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	151	1	999	0	none›
4	56	services	married	high.school	no	no	yes	telephone	may	mon	307	1	999	0	none›

Let's display the size of our dataframe, the name of the variables that make it up and their types.

In [20]:
```python
banking.shape
```

Out[20]:
```
(41188, 21)
```

The dataframe contains 41188 values (rows), for each of the 21 variables (columns), including a target variable which is 'y'.

In [21]:
```python
banking.columns
```

Out[21]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
       'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'emp.var.rate', 'cons.price.idx',
       'cons.conf.idx', 'euribor3m', 'nr.employed', 'y'],
      dtype='object')
```

Input features (column names):

1. `age` - client's age in years (numeric)

2. `job` - type of job (categorical: `admin.`, `blue-collar`, `entrepreneur`, `housemaid`, `management`, `retired`, `self-employed`, `services`, `student`, `technician`, `unemployed`, `unknown`)

3. `marital` - marital status (categorical: `divorced`, `married`, `single`, `unknown`)

4. `education` - client's education (categorical: `basic.4y`, `basic.6y`, `basic.9y`, `high.school`, `illiterate`, `professional.course`, `university.degree`, `unknown`)

5. `default` - has credit in default? (categorical: `no`, `yes`, `unknown`)

6. `housing` - has housing loan? (categorical: `no`, `yes`, `unknown`)

7. `loan` - has personal loan? (categorical: `no`, `yes`, `unknown`)

8. `contact` - contact communication type (categorical: `cellular`, `telephone`)

9. `month` - last contact month of the year (categorical: `jan`, `feb`, `mar`, ..., `nov`, `dec`)

10. `day_of_week` - last contact day of the week (categorical: `mon`, `tue`, `wed`, `thu`, `fri`)

11. `duration` - last contact duration, in seconds (numeric).

12. `campaign` - number of contacts performed and for this client during this campaign (numeric, includes the last contact)

13. `pdays` - number of days that have passed after the client was last contacted from the previous campaign (numeric; 999 means the client has not been previously contacted)

14. `previous` - number of contacts performed for this client before this campaign (numeric)

15. `poutcome` - outcome of the previous marketing campaign (categorical: `failure`, `nonexistent`, `success`)

16. `emp.var.rate` - employment variation rate, quarterly indicator (numeric)

17. `cons.price.idx` - consumer price index, monthly indicator (numeric)

18. `cons.conf.idx` - consumer confidence index, monthly indicator (numeric)

19. `euribor3m` - euribor 3 month rate, daily indicator (numeric)

20. `nr.employed` - number of employees, quarterly indicator (numeric)

Output feature (desired target):

21. `y` - has the client subscribed a term deposit? (binary: `yes`,`no`)

Let's display the general information on all variables in our dataframe.

In [22]:
```
print(banking.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   age             41188 non-null  int64  
 1   job             41188 non-null  object 
 2   marital         41188 non-null  object 
 3   education       41188 non-null  object 
 4   default         41188 non-null  object 
 5   housing         41188 non-null  object 
 6   loan            41188 non-null  object 
 7   contact         41188 non-null  object 
 8   month           41188 non-null  object 
 9   day_of_week     41188 non-null  object 
 10  duration        41188 non-null  int64  
 11  campaign        41188 non-null  int64  
 12  pdays           41188 non-null  int64  
 13  previous        41188 non-null  int64  
 14  poutcome        41188 non-null  object 
 15  emp.var.rate    41188 non-null  float64
 16  cons.price.idx  41188 non-null  float64
 17  cons.conf.idx   41188 non-null  float64
 18  euribor3m       41188 non-null  float64
 19  nr.employed     41188 non-null  float64
 20  y               41188 non-null  object 
dtypes: float64(5), int64(5), object(11)
memory usage: 6.6+ MB
None
```

The dataframe is filled, it contains no missing values ('non-null') so there is no need to fill the gaps. On the other hand it contains 5 variables of integer type ('int64'), 5 variables of floating type ('float64') and 11 variables of categorical and binary type ('object').

The 'describe()' method shows the main statistical characters of each numerical variable in our dataset, i.e., those of the 'int64' and 'float64' types.

In [23]:
```
banking.describe()
```

Out[23]:

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
count	41188.00	41188.00	41188.00	41188.00	41188.00	41188.00	41188.00	41188.00	41188.00	41188.00
mean	40.02	258.29	2.57	962.48	0.17	0.08	93.58	-40.50	3.62	5167.04
std	10.42	259.28	2.77	186.91	0.49	1.57	0.58	4.63	1.73	72.25
min	17.00	0.00	1.00	0.00	0.00	-3.40	92.20	-50.80	0.63	4963.60
25%	32.00	102.00	1.00	999.00	0.00	-1.80	93.08	-42.70	1.34	5099.10
50%	38.00	180.00	2.00	999.00	0.00	1.10	93.75	-41.80	4.86	5191.00
75%	47.00	319.00	3.00	999.00	0.00	1.40	93.99	-36.40	4.96	5228.10
max	98.00	4918.00	56.00	999.00	7.00	1.40	94.77	-26.90	5.04	5228.10

In general, according to the data, it is impossible to say that there are outliers in the data. However, such an inspection is not enough, it is desirable to still see the charts of the target feature dependence from each input feature. We will do it later when we visualize features and dependencies.

To see the statistics of non-numerical variables, therefore categorical, you must specify the type of the variable to the 'include' parameter of the 'describe()' method. It is also possible to set 'include' to 'all' as follows 'include = all' to display statistics of all existing variables in our dataframe.

In [24]:
```
banking.describe(include = ["object"])
```

Out[24]:

	job	marital	education	default	housing	loan	contact	month	day_of_week	poutcome	y
count	41188	41188	41188	41188	41188	41188	41188	41188	41188	41188	41188
unique	12	4	8	3	3	3	2	10	5	3	2
top	admin.	married	university.degree	no	yes	no	cellular	may	thu	nonexistent	no
freq	10422	24928	12168	32588	21576	33950	26144	13769	8623	35563	36548

The result shows that the average client refers to administrative staff ('job = admin.'), is married ('marital = married') and has a university degree ('education = university.degree').

Let’s look at the distribution of our target variable 'y' using the 'value_counts()' method which is used for 'object' and 'boolean' variables.

In [25]:
```
banking["y"].value_counts()
```

Out[25]:
```
no     36548
yes     4640
Name: y, dtype: int64
```

In [26]:
```
banking["y"].value_counts(normalize = True)
```

Out[26]:
```
no    0.89
yes   0.11
Name: y, dtype: float64
```

4640 customers or 11% of 41188 have issued a term deposit and 36548 or 89% have not done it.The other analysis that could be drawn from this is that the values of our target variable are not balanced so we are talking about Unbalanced Dataset. For a prediction model that we would be asked to make, we will have to ensure that it is balanced.

Observe the distribution of the variable 'marital' by specifying the parameter 'normalize' to 'true' of the method 'value_counts' to display the result as a percentage.

In [27]:
```
banking["marital"].value_counts(normalize = True)
```

Out[27]:
```
married    0.61
single     0.28
divorced   0.11
unknown    0.00
Name: marital, dtype: float64
```

As we see 61% (0.61) of clients are married, a point to consider when planning marketing campaigns to manage deposit operations.

A dataframe can be sorted according to certain variables. In our case for example we can sort according to the variable 'duration' by specifying the parameter 'ascending' to say crescent to the value 'False' in order to sort either descending.

Sorting

In [28]:
```
banking.sort_values(by = "duration", ascending = False).head()
```

Out[28]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	prev
24091	33	technician	single	professional.course	no	yes	no	telephone	nov	mon	4918	1	999	
22192	52	blue-collar	married	basic.4y	no	no	no	telephone	aug	thu	4199	3	999	
40537	27	admin.	single	high.school	no	no	no	telephone	aug	fri	3785	1	999	
13820	31	technician	married	professional.course	no	no	no	cellular	jul	thu	3643	1	999	
7727	37	unemployed	married	professional.course	no	yes	no	telephone	may	fri	3631	2	999	

The call durations exceeding one hour are 3600s and these calls took place Monday and Thursday ('day_of_week') in November and August ('month').

Let’s sort by age and length of call increasing by age and decreasing by duration.

In [29]:
```
banking.sort_values(by = ["age", "duration"], ascending = [True, False]).head(20)
```

Out[29]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	p
38274	17	student	single	unknown	no	no	yes	cellular	oct	tue	896	1	2	2	
37579	17	student	single	basic.9y	no	unknown	unknown	cellular	aug	fri	498	2	999	1	
37140	17	student	single	unknown	no	yes	no	cellular	aug	wed	432	3	4	2	
37539	17	student	single	basic.9y	no	yes	no	cellular	aug	fri	182	2	999	2	
37558	17	student	single	basic.9y	no	yes	no	cellular	aug	fri	92	3	4	2	
37125	18	student	single	basic.9y	no	yes	no	cellular	aug	tue	642	1	999	0	n
37626	18	student	single	basic.6y	no	yes	no	cellular	aug	mon	628	1	999	0	n
41084	18	student	single	unknown	no	yes	no	cellular	nov	tue	600	2	999	3	
37955	18	student	single	unknown	no	yes	no	cellular	sep	fri	563	1	999	0	n
40379	18	student	single	unknown	no	yes	no	cellular	aug	wed	561	1	17	2	
39576	18	student	single	unknown	no	yes	no	cellular	may	tue	489	1	6	1	
39039	18	student	single	basic.9y	no	yes	no	cellular	dec	mon	446	2	999	0	n
39575	18	student	single	unknown	no	yes	no	telephone	may	tue	421	1	3	1	
39057	18	student	single	basic.9y	no	no	no	cellular	dec	thu	412	2	999	0	n
38009	18	student	single	unknown	no	no	no	telephone	sep	tue	401	2	999	0	n
41088	18	student	single	basic.4y	no	yes	no	telephone	nov	tue	394	1	13	2	
37916	18	student	single	unknown	no	no	no	cellular	sep	thu	385	1	3	1	
38597	18	student	single	basic.6y	no	no	yes	cellular	oct	fri	368	2	999	0	n
40383	18	student	single	unknown	no	yes	yes	telephone	aug	wed	297	1	999	0	n
35871	18	student	single	high.school	no	no	no	cellular	may	fri	271	1	999	1	

The youngest clients are seventeen (17) years old for the 'age' variable, and their call times are greater than three (3) minutes for only three (3) of these customers. This indicates the inefficiency of the long-term interaction with these customers, and they may be excluded from the target of marketing campaigns.

In [30]:
```
banking.apply(np.max)
```

Out[30]:
```
age                    98
job               unknown
marital           unknown
education         unknown
default               yes
housing               yes
loan                  yes
contact         telephone
month                 sep
day_of_week           wed
duration             4918
campaign               56
pdays                 999
previous                7
poutcome          success
emp.var.rate         1.40
cons.price.idx      94.77
cons.conf.idx      -26.90
euribor3m            5.04
nr.employed       5228.10
y                     yes
dtype: object
```

The oldest client is ninety-eight (98) years old ('age' = 98), the number of contacts with one of the clients reaches 56 ('campaign' = 56).

The 'map()' method can also be used to override values in a column by passing them as arguments in a dictionary: {'old_value: new_value'}.

In [35]:
```
d = {"no": 0, "yes": 1}
banking["y"] = banking["y"].map(d)
banking.head()
```

Out[35]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	pout
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	261	1	999	0	none›
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	149	1	999	0	none›
2	37	services	married	high.school	no	yes	no	telephone	may	mon	226	1	999	0	none›
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	151	1	999	0	none›
4	56	services	married	high.school	no	no	yes	telephone	may	mon	307	1	999	0	none›

Questions & Answers

1. What is the proportion of clients attracted ?

In [36]:
```
print("Proportion of attracted clients =", '{:.1%}'.format(banking["y"].mean()))
```

```
Proportion of attracted clients = 11.3%
```

11.3% is rather a very bad indicator for a bank, with such a percentage of customers attracted it could quickly go bankrupt.

2. What are the average values of the numerical features among the attracted clients ?

In [37]:
```
banking[banking["y"] == 1].mean()
```

Out[37]:
```
age                 40.91
duration           553.19
campaign             2.05
pdays              792.04
previous             0.49
emp.var.rate        -1.23
cons.price.idx      93.35
cons.conf.idx      -39.79
euribor3m            2.12
nr.employed       5095.12
y                    1.00
dtype: float64
```

Thus, the average age of clients attracted to banking services is around 40 years ('age'= 40.91) for a call duration of about 9 minutes ('duration' = 553.19) and two (2) calls were needed to attract them ('campaign'= 2.05).

3. What is the average duration of calls for attracted clients ?

In [38]:
```
acd = (banking[banking["y"] == 1]["duration"].mean())
acd_in_min = acd // 60
print("Average call duration for attracted clients=", acd_in_min, "minutes", float(acd) % 60, "seconds")
```

```
Average call duration for attracted clients= 9.0 minutes 13.191163793103442 seconds
```

So, the average time of a successful call is about 553 seconds about 9 minutes.

4. What is the average age of attracted ('y == 1') and unmarried ('marital' == 'single') clients ?

In [39]:
```
print("Average age of attracted and unmarried clients =", int(banking[(banking["y"] == 1) & (banking["marital"]
```

```
Average age of attracted and unmarried clients = 31 years
```

The average age of unmarried clients is 31 years of age another important point to consider.

Pivot tables

Now we want to see how observations in our sample are distributed in the context of two features 'y' and 'marital'. To do this, we're going to build cross tabulation by the 'crosstab()' method.

In [14]:
```
pd.crosstab(banking["y"], banking["marital"])
```

Out[14]:

marital	divorced	married	single	unknown
y				
no	4136	22396	9948	68
yes	476	2532	1620	12

The result shows that the number of attracted married clients is 2532 from the total number.

In [13]:
```
pd.crosstab(banking["y"],
            banking["marital"],
            normalize = 'index')
```

Out[13]:

marital	divorced	married	single	unknown
y				
no	0.11	0.61	0.27	0.00
yes	0.10	0.55	0.35	0.00

Values : 'age', 'duration'
Index : 'job'
Aggregate function : 'mean'
Let's find the average age and the call duration for different types of client employment 'job'.

In [27]:

```
banking.pivot_table(
    ["age", "duration"],
    ["job"],
    aggfunc = "mean",
).head(10)
```

Out[27]:

	age	duration
job		
admin.	38.19	254.31
blue-collar	39.56	264.54
entrepreneur	41.72	263.27
housemaid	45.50	250.45
management	42.36	257.06
retired	62.03	273.71
self-employed	39.95	264.14
services	37.93	258.40
student	25.89	283.68
technician	38.51	250.23

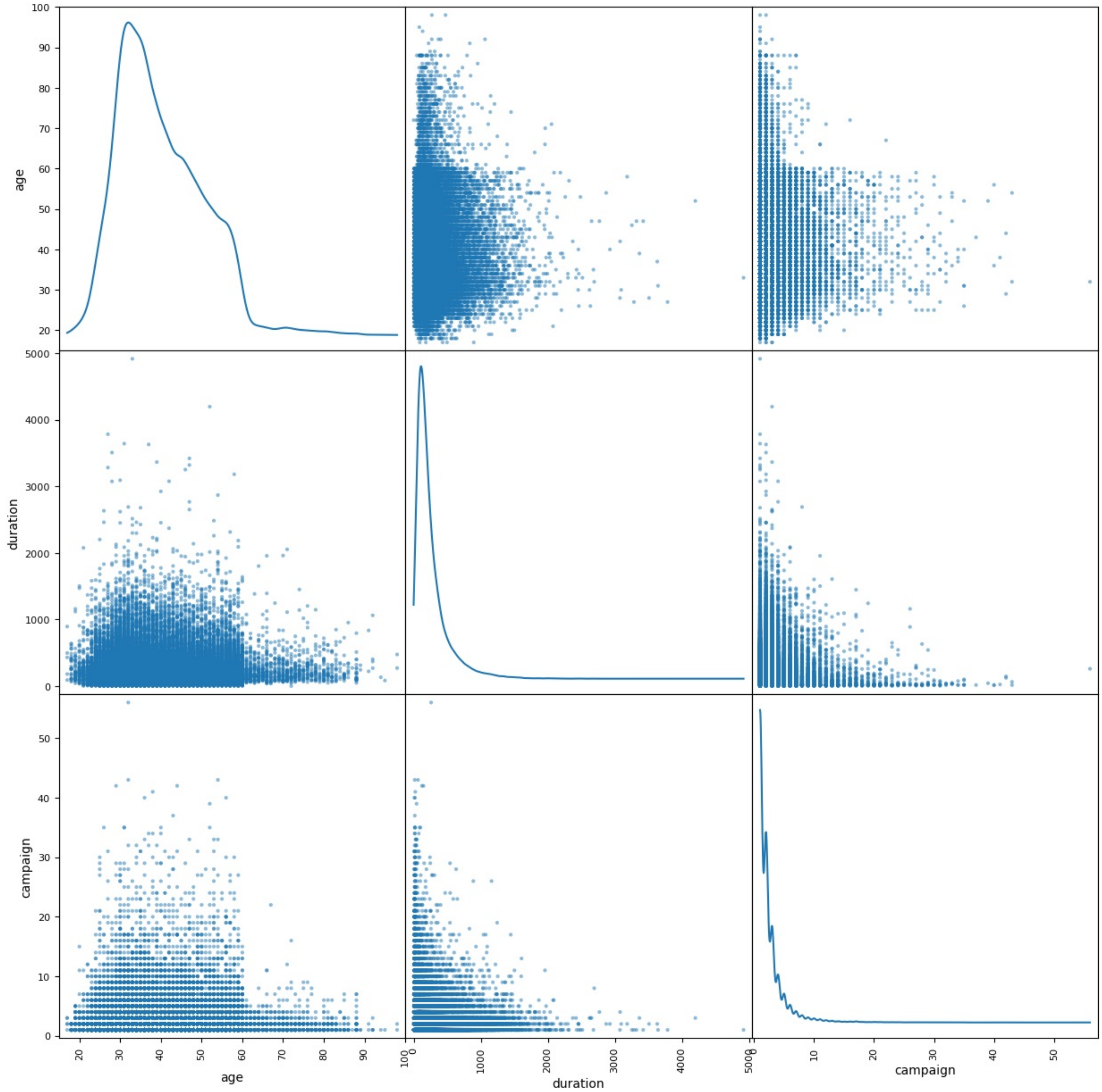
The obtained results allow you to plan marketing banking campaigns more effectively.

Visualization in Pandas

Let's use method scatter_matrix() for numerical features which allows us to visualize the pairwise dependencies between the features as well as the distribution of each feature on the diagonal.

In [40]:

```
pd.plotting.scatter_matrix(
    banking[["age", "duration", "campaign"]],
    figsize = (15, 15),
    diagonal = "kde")
plt.show()
```

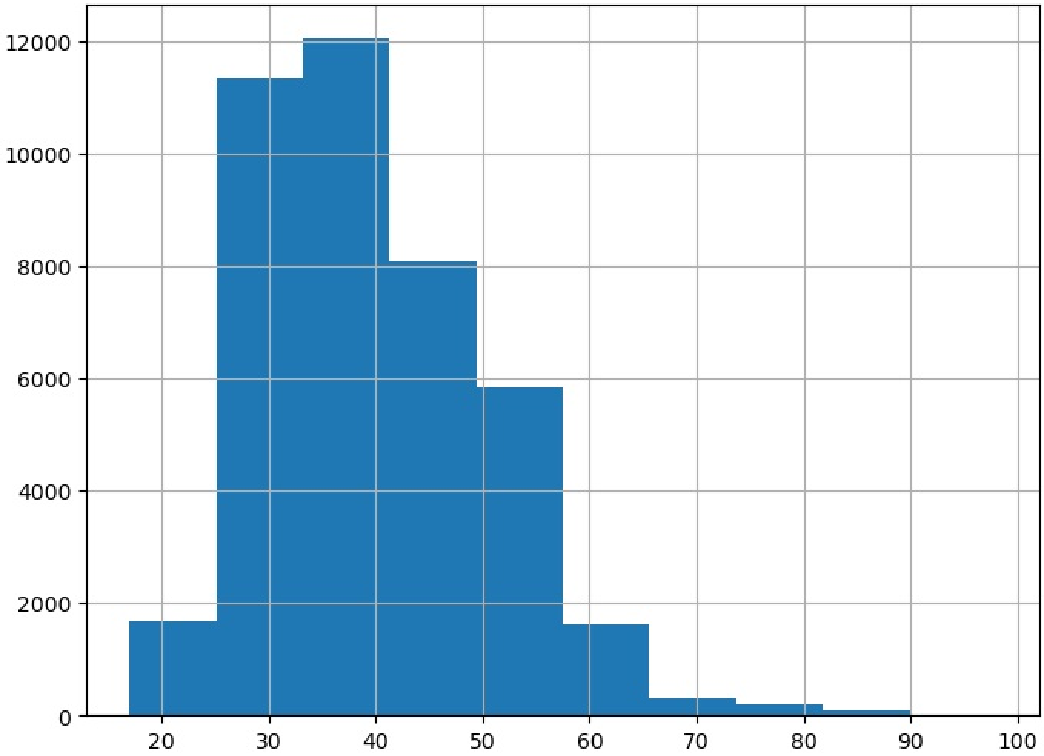


A scatter matrix (pairs plot) compactly plots all the numeric variables we have in a dataset against each other. The plots on the main diagonal allow you to visually define the type of data distribution: the distribution is similar to normal for age, and for a call duration and the number of contacts, the geometric distribution is more suitable.

Let's build a separate histogram for each feature.

In [41]:
```
banking["age"].hist()
```

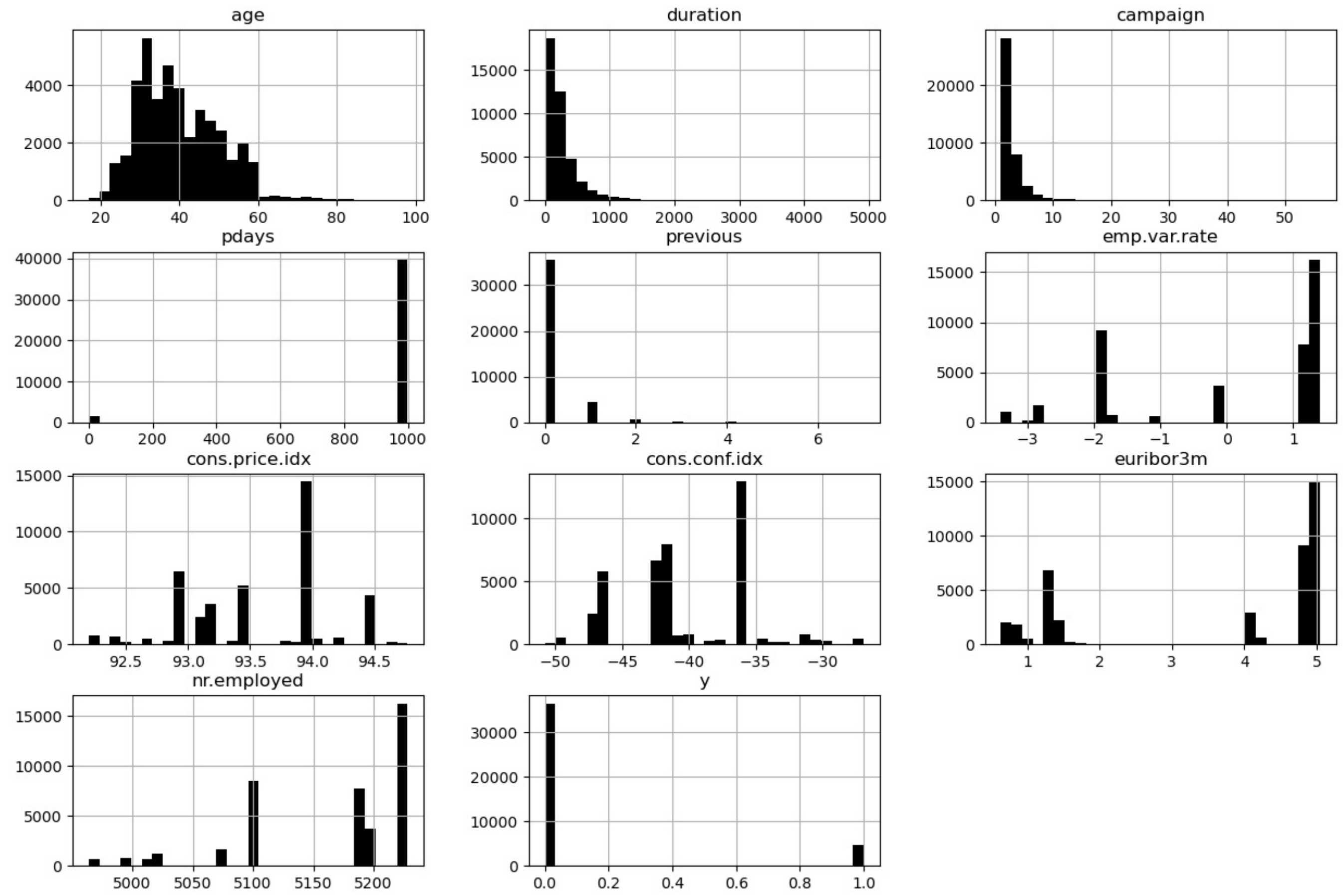
Out[41]: <AxesSubplot:>



The histogram shows that most of our clients are between the ages of 25 and 50, which corresponds to the actively working part of the population.

Now, let's build it for all together.

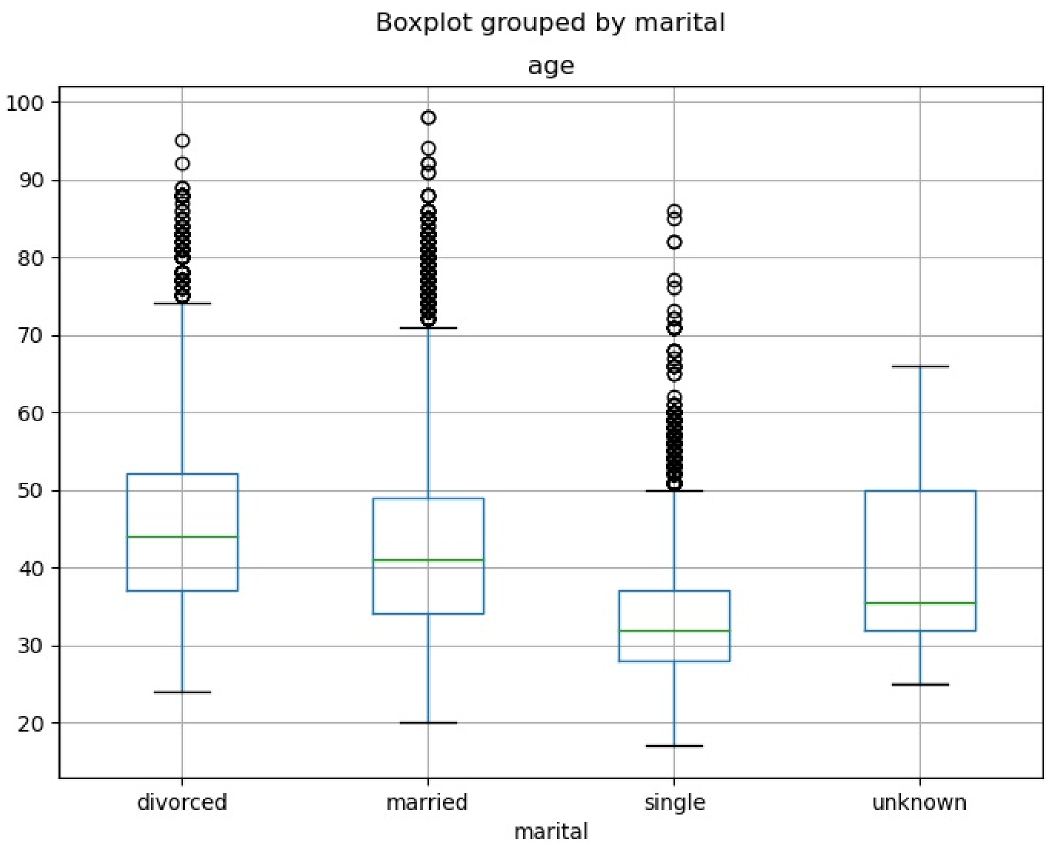
In [42]:
```
banking.hist(color = "k",
        bins = 30,
        figsize = (15, 10))
plt.show()
```



A visual analysis of the histograms presented allows us to make preliminary assumptions about the variability of the source data.

Box Plot is useful too. It allows you to compactly visualize the main characteristics of the feature distribution (the median, lower and upper quartile, minimal and maximum, outliers).

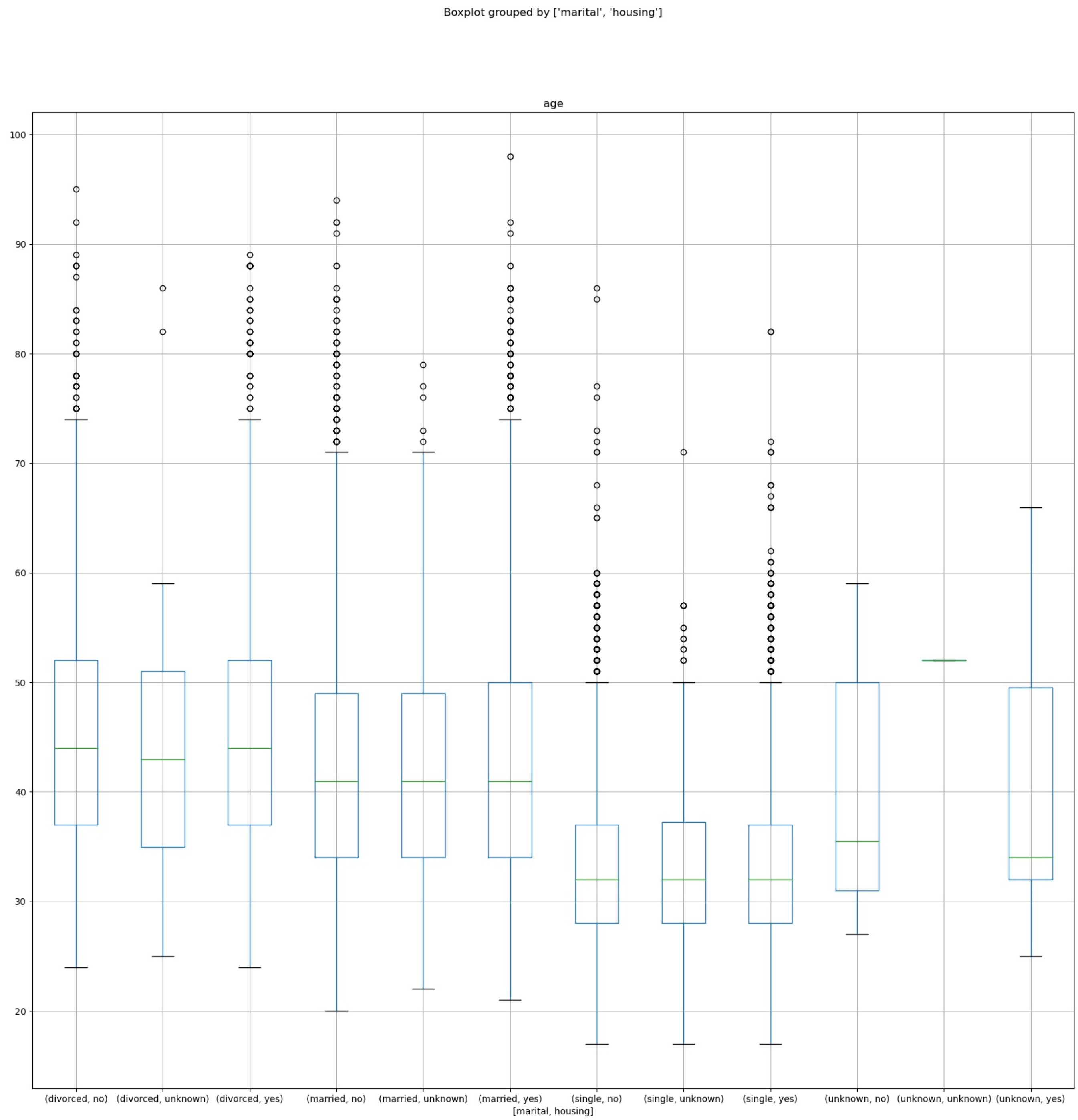
In [43]:
```
banking.boxplot(column = "age",
                by = "marital")
plt.show()
```



The plot shows that unmarried people are on average younger than divorced and married ones. For the last two groups, there is an outlier zone over 70 years old, and for unmarried - over 50.

We can do this by data grouping on any other feature

In [45]:
```
banking.boxplot(column = "age",
           by = ["marital", "housing"],
           figsize = (20, 20))
plt.show()
```



As we can see, age and marital status don't have any significant influence on having a housing loan.

Additional questions

In this section, we will solve some tasks with the source bank dataset.

Question 1

List 10 clients with the largest number of contacts.

In [47]:
```
banking.sort_values(by = "campaign", ascending = False).head(10)
```

Out[47]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays
4107	32	admin.	married	university.degree	unknown	unknown	unknown	telephone	may	mon	261	56	999
18728	54	admin.	married	university.degree	unknown	yes	no	cellular	jul	thu	65	43	999
13447	32	technician	single	university.degree	no	yes	yes	telephone	jul	wed	16	43	999
4168	29	technician	married	professional.course	no	yes	no	telephone	may	mon	124	42	999
5304	44	retired	married	basic.9y	no	yes	no	telephone	may	fri	147	42	999
11033	38	blue-collar	married	basic.4y	no	yes	no	telephone	jun	wed	25	41	999
18754	36	admin.	single	university.degree	no	no	no	cellular	jul	thu	18	40	999
11769	56	self-employed	married	professional.course	no	no	yes	telephone	jun	fri	13	40	999
4114	52	entrepreneur	married	university.degree	no	no	no	telephone	may	mon	44	39	999
11593	43	technician	married	high.school	no	yes	no	telephone	jun	fri	17	37	999

Question 2

Determine the median age and the number of contacts for different levels of client education.

In [48]:

```
banking.pivot_table(
    ["age", "campaign"],
    ["education"],
    aggfunc = ["mean", "count"],
)
```

Out[48]:

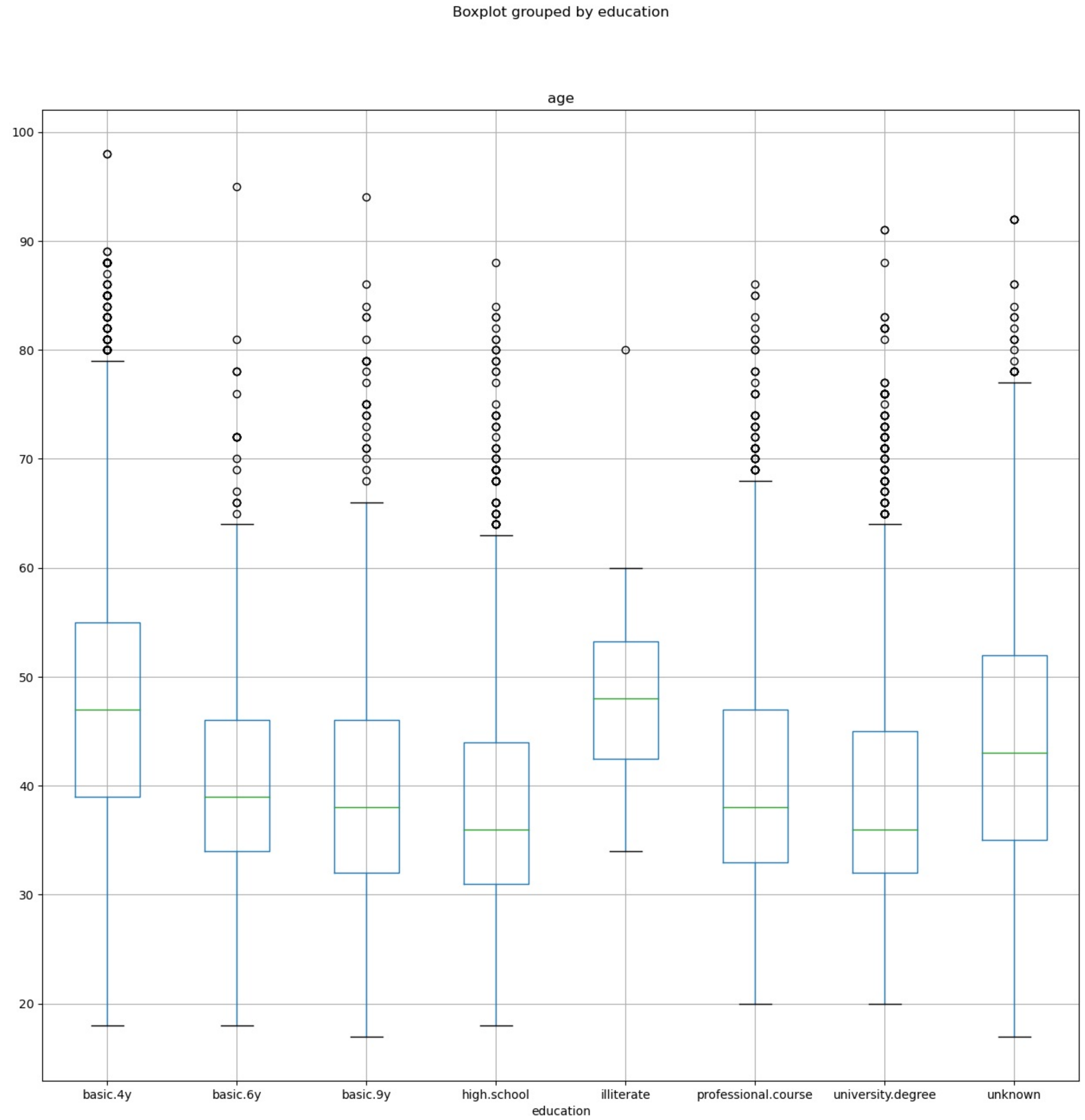
	mean		count	
	age	campaign	age	campaign
education				
basic.4y	47.60	2.60	4176	4176
basic.6y	40.45	2.56	2292	2292
basic.9y	39.06	2.53	6045	6045
high.school	38.00	2.57	9515	9515
illiterate	48.50	2.28	18	18
professional.course	40.08	2.59	5243	5243
university.degree	38.88	2.56	12168	12168
unknown	43.48	2.60	1731	1731

Question 3

Output box plot to analyze the client age distribution by their education level.

In [49]:

```
banking.boxplot(column = "age",
  by = "education",
  figsize = (15, 15))
plt.show()
```



Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js