Virtual Internship (Data Science)

Data Intake Report

Group Name: Project Group 1

Members:

No	Name	Email	Country	College/company	Specialization
1	Preeti Verma	vermapreeti.dataanalyst@gmail.com	Canada	-	Data Science
2	Thanuja Modiboina	thanujayadav953@gmail.com	UK	-	Data Science
3	Abishek James	abishekjames1998@gmail.com	Ireland	-	Data Science

Name: Bank Marketing (Campaign)

Report date: 26-04-2023

Internship Batch: LISUM19

Data intake by:

Data intake reviewer: Data Glacier

Data storage location:

Problem Description :

ABC Bank wants to sell its term deposit product to customers and before launching the product they want to develop a model which helps to understand whether a particular customer will buy their product or not (based on the customer's past interaction with the bank or other Financial Institution). This is an application of the company's marketing data.

Business Understanding :

The goal is to build a Machine Learning model that helps in predicting the outcomes of each customer's marketing campaign and analyzing which features have an impact on the outcomes will help the company to understand how to make the campaign more effective. Additionally, categorizing the customer group that subscribed to the term deposit helps to determine who is more likely to purchase the product in the future, thereby developing more targeted marketing campaigns.

This can be accomplished by using an ML model that shortlists the customers whose possibility of purchasing the product is higher. So, marketing such as telemarketing, SMS or email marketing can concentrate only on those customers. It will save time and resources by doing this.

Project Lifecycle

Deadline (Date/week)	Plan and Deliverables
19 April 2023(Week 7)	• Problem statement • Business understanding • Dataset collection
26 April 2023(Week 8)	• Data understanding • Data analysis - finding null values, and outliers. • Data processing
2 May 2023(Week 9)	Data cleaning and transformation
9 May 2023(Week 10)	EDA and Model Recommendation
16 May 2023(Week 11)	EDA Presentation and Proposed Modeling Technique

23 May 2023(Week 12)	Model Selection and Building the Model
30 May 2023(Week 13)	Final project report and code submission

Tabular data details:

File 1: bank_additional_full.csv

Total number of observations	41189
Total number of files	2
Total number of features	21
Base format of the file	.csv
Size of the data	5.56MB

File 2: bank_additional.csv

Total number of observations	4120
Total number of files	2
Total number of features	21

Exploratory Data Analysis

1. The data covers the period from May 2008 to November 2010.

2. There are 2 datasets, the second dataset is a sample of the first dataset. So, we are not taking the second dataset.

3. There are 10 integers and 11 categorical variables.

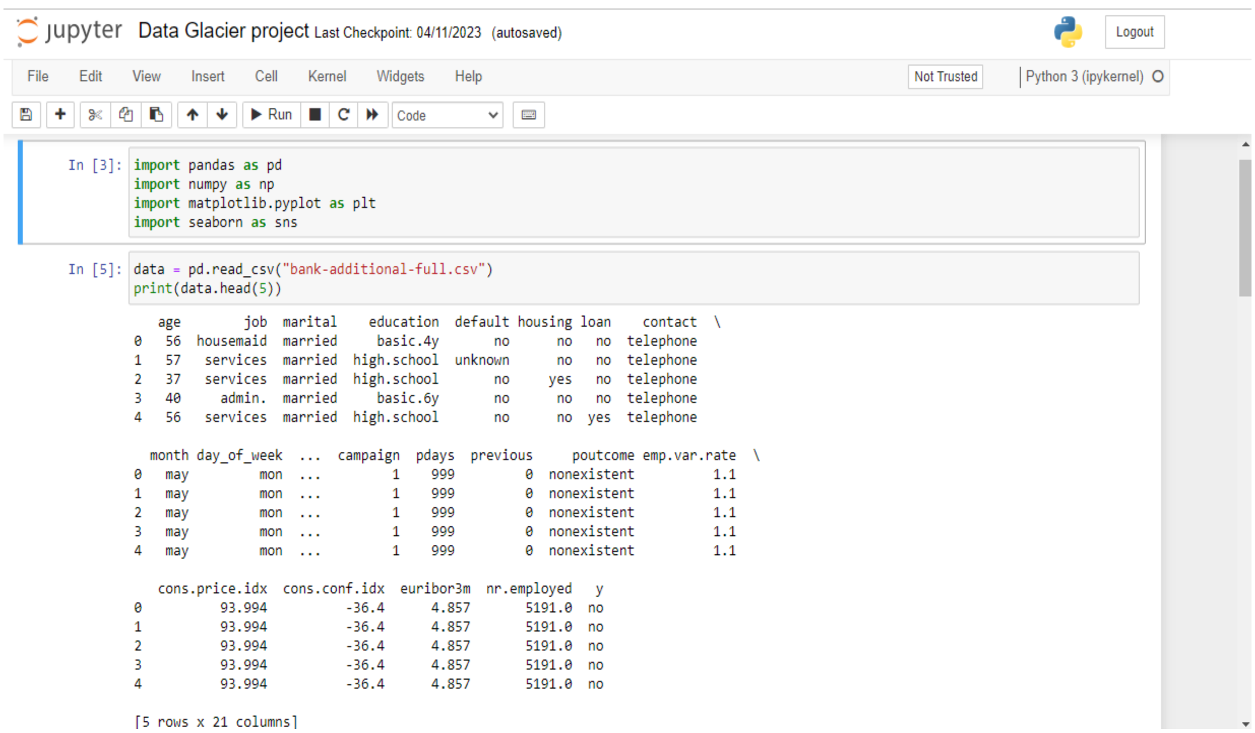
4. The missing values in the dataset are presented by an "unknown" string. We changed it to NaN.

5. There are missing values in six variables: job, marital status, education, default, housing, and loan. This will be imputed using various methods.

6. There are 12 duplicates in the first dataset and no duplicates in the sample dataset, this will be dropped since they are minimal and will not affect our analysis

Assumptions

We assume the data provided is correct and up to date.



In [3]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [5]:
```python
data = pd.read_csv("bank-additional-full.csv")
print(data.head(5))
```

```
   age        job  marital    education  default housing loan    contact  \
0   56  housemaid  married     basic.4y       no      no   no  telephone
1   57   services  married  high.school  unknown      no   no  telephone
2   37   services  married  high.school       no     yes   no  telephone
3   40     admin.  married     basic.6y       no      no   no  telephone
4   56   services  married  high.school       no      no  yes  telephone

  month day_of_week  ...  campaign  pdays  previous     poutcome emp.var.rate  \
0   may         mon  ...         1    999         0  nonexistent          1.1
1   may         mon  ...         1    999         0  nonexistent          1.1
2   may         mon  ...         1    999         0  nonexistent          1.1
3   may         mon  ...         1    999         0  nonexistent          1.1
4   may         mon  ...         1    999         0  nonexistent          1.1

   cons.price.idx  cons.conf.idx  euribor3m  nr.employed   y
0          93.994          -36.4      4.857       5191.0  no
1          93.994          -36.4      4.857       5191.0  no
2          93.994          -36.4      4.857       5191.0  no
3          93.994          -36.4      4.857       5191.0  no
4          93.994          -36.4      4.857       5191.0  no

[5 rows x 21 columns]
```

jupyter Data Glacier project Last Checkpoint: 04/11/2023 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

```
4          93.994        -36.4      4.857       5191.0  no

[5 rows x 21 columns]
```

In [6]:
```
print(data.dtypes)
```

```
age                 int64
job                object
marital            object
education          object
default            object
housing            object
loan               object
contact            object
month              object
day_of_week        object
duration            int64
campaign            int64
pdays               int64
previous            int64
poutcome           object
emp.var.rate      float64
cons.price.idx    float64
cons.conf.idx     float64
euribor3m         float64
nr.employed       float64
y                  object
dtype: object
```

In [7]:
```
data.isna().sum()
```

jupyter Data Glacier project Last Checkpoint: 04/11/2023 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

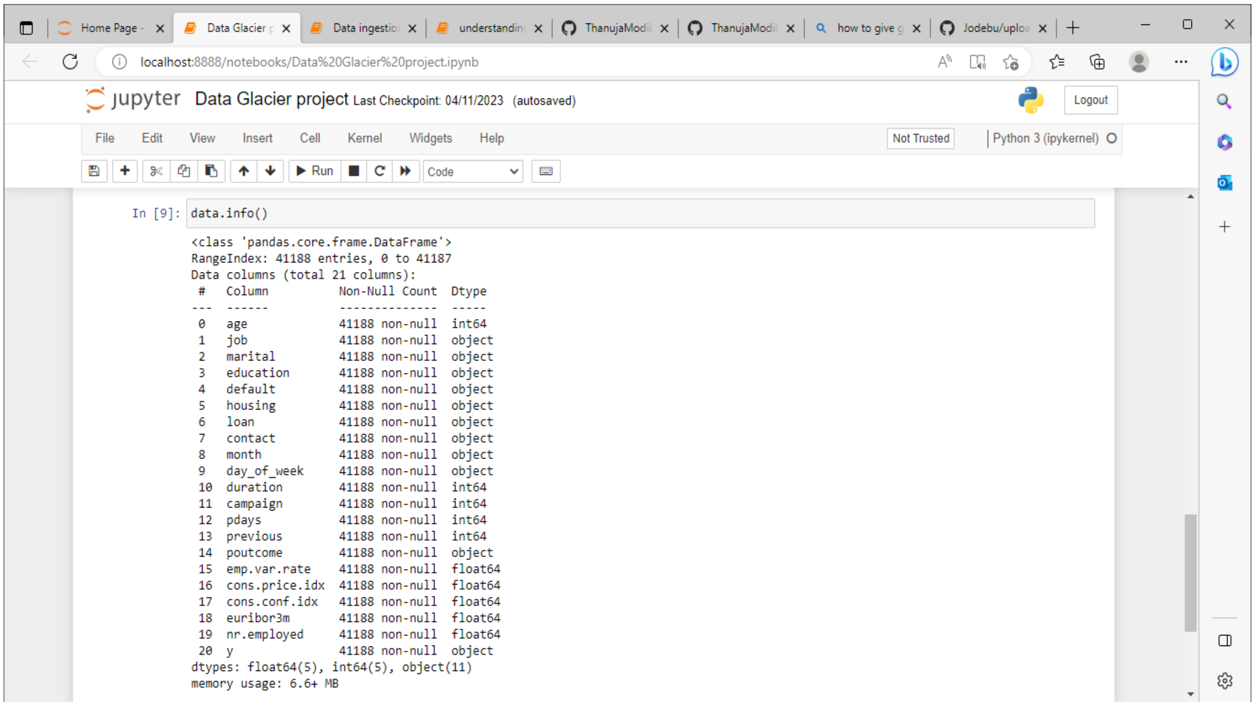
Not Trusted | Python 3 (ipykernel)

In [7]:
```
data.isna().sum()
```

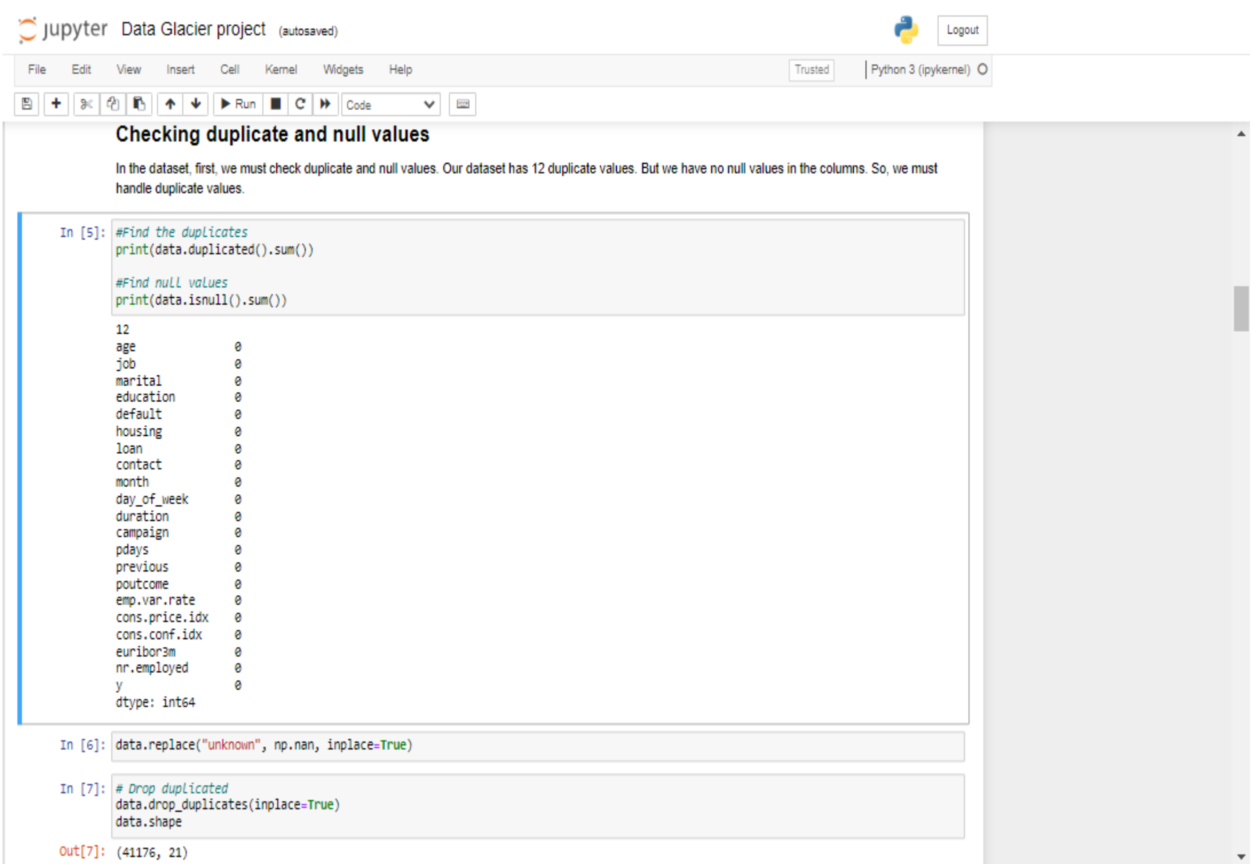
Out[7]:
```
age               0
job               0
marital           0
education         0
default           0
housing           0
loan              0
contact           0
month             0
day_of_week       0
duration          0
campaign          0
pdays             0
previous          0
poutcome          0
emp.var.rate      0
cons.price.idx    0
cons.conf.idx     0
euribor3m         0
nr.employed       0
y                 0
dtype: int64
```

In [9]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
```



Week 8 Assignment:



If we set 'Unknown' to null, then our dataset having null values. Below, we can see which feature contains null values.

In [8]:
```
# data.isna().sum()
print(data.isnull().sum())
```

```
age                 0
job               330
marital            80
education        1730
default          8596
housing           990
loan              990
contact             0
month               0
day_of_week         0
duration            0
campaign            0
pdays               0
previous            0
poutcome            0
emp.var.rate        0
cons.price.idx      0
cons.conf.idx       0
euribor3m           0
nr.employed         0
y                   0
dtype: int64
```

In [9]:
```
null_percentage = data.isnull().mean()*100
null_percentage
```

Out[9]:
```
age                0.000000
job                0.801438
marital            0.194288
education          4.201477
default           20.876239
housing            2.404313
loan               2.404313
contact            0.000000
month              0.000000
day_of_week        0.000000
duration           0.000000
```

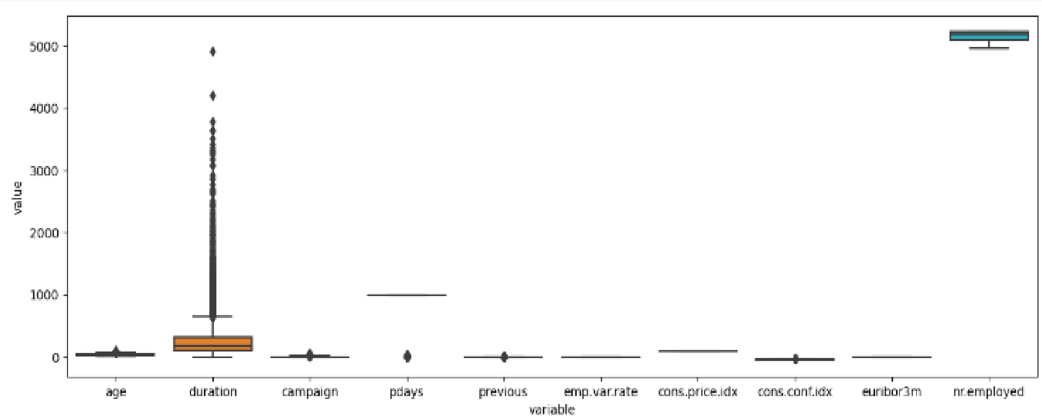
jupyter Data Glacier project (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
poutcome           0.000000
emp.var.rate       0.000000
cons.price.idx     0.000000
cons.conf.idx      0.000000
euribor3m          0.000000
nr.employed        0.000000
y                  0.000000
dtype: float64
```

Outliers in the dataset

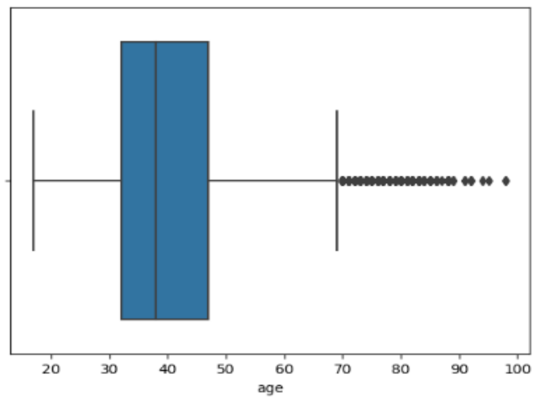
In [10]:
```
data_1 = data[['age','duration','campaign','pdays','previous','emp.var.rate','cons.price.idx','cons.conf.idx','euribor3m','nr.em
```

In [11]:
```
fig, ax = plt.subplots(figsize=(15, 5))
sns.boxplot(x="variable", y="value", data=pd.melt(data_1))
plt.figure(figsize=(20,5))
plt.show()
```




```
<Figure size 2000x500 with 0 Axes>
```

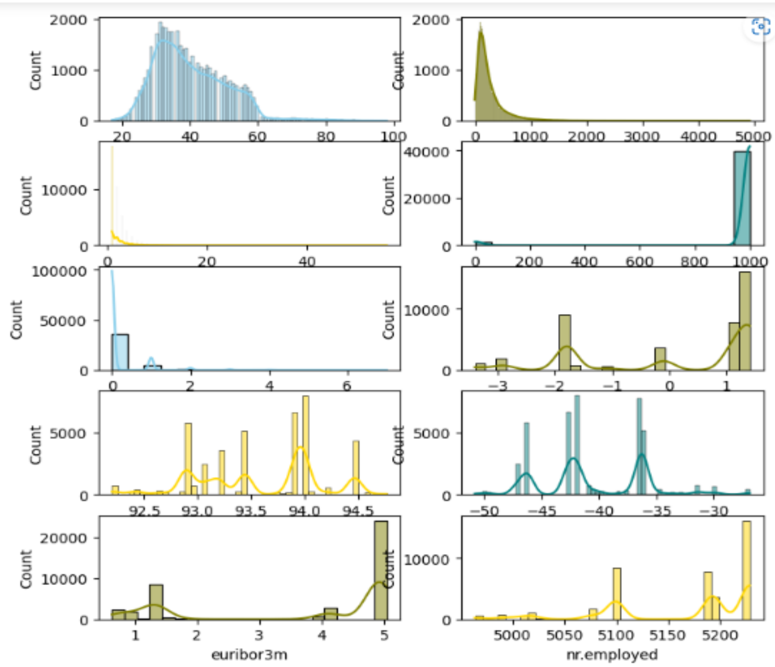
In [12]:
```
sns.boxplot(x = data_1['age'])
plt.show()
```



In [13]:
```
fig, axs = plt.subplots(5, 2, figsize=(8, 8))
sns.histplot(data=data_1, x="age", kde=True, color="skyblue",ax=axs[0, 0])
sns.histplot(data=data_1, x="duration", kde=True, color="olive",ax=axs[0, 1])
sns.histplot(data=data_1, x="campaign", kde=True, color="gold", ax=axs[1, 0])
sns.histplot(data=data_1, x="pdays", kde=True, color="teal", ax=axs[1, 1])
sns.histplot(data=data_1, x="previous", kde=True, color="skyblue", ax=axs[2, 0])
sns.histplot(data=data_1, x="emp.var.rate", kde=True, color="olive", ax=axs[2, 1])
sns.histplot(data=data_1, x="cons.price.idx", kde=True, color="gold", ax=axs[3, 0])
sns.histplot(data=data_1, x="cons.conf.idx", kde=True, color="teal", ax=axs[3, 1])
sns.histplot(data=data_1, x="euribor3m", kde=True, color="olive", ax=axs[4, 0])
sns.histplot(data=data_1, x="nr.employed", kde=True, color="gold", ax=axs[4, 1])
```

Out[13]: <Axes: xlabel='nr.employed', ylabel='Count'>

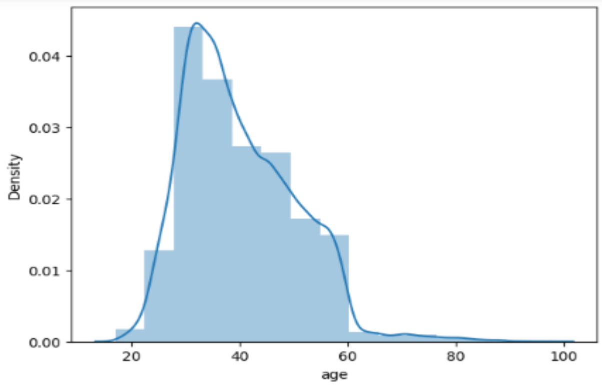
jupyter Data Glacier project (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run Code



In [14]:
```
sns.distplot(data_1['age'], bins = 15, kde = True)
plt.show()
```

jupyter Data Glacier project (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Trusted | Python 3 (ipykernel)



Handling the outliers in the Age column

The interquartile range (IQR) is a statistical measure used to determine the variability in a dataset by subtracting the 25th percentile (Q1) from the 75th percentile (Q3). Outliers are values that fall outside of a certain range, which is typically calculated as Q1−1.5IQR to Q3+1.5IQR.

In [15]:
```
data_1[['age']].describe()
```

Out[15]:

	age
count	41176.00000
mean	40.02380
std	10.42068
min	17.00000

jupyter Data Glacier project (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Trusted | Python 3 (ipykernel)

In [16]:
```
data_1['age'].quantile(0.25)
```

Out[16]: 32.0

In [17]:
```
data_1['age'].quantile(0.75)
```

Out[17]: 47.0

In [18]:
```
Q1 = data_1['age'].quantile(0.25)
Q3 = data_1['age'].quantile(0.75)
IQR = Q3-Q1
IQR
```

Out[18]: 15.0

In [19]:
```
lower_lim = Q1-1.5*IQR
upper_lim = Q1+1.5*IQR
print(lower_lim)
print(upper_lim)
```

```
9.5
54.5
```

In [20]:
```
outliers_15_low = (data_1['age']<lower_lim)
# print(outliers_15_low)
outliers_15_up = (data_1['age']>upper_lim)
# print(outliers_15_up)
```

In [21]:
```
len(data_1['age']) - (len(data_1['age'][outliers_15_low])+len(data_1['age'][outliers_15_up]))
```

Out[21]: 36948

In [22]:
```
data_1['age'][(outliers_15_low|outliers_15_up)]
```

Out[22]:
```
0        56
1        57
4        56
6        59
13       57
         ..
41178    62
41179    64
```

jupyter Data Glacier project (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

```
3        40
5        45
7        41
8        24
         ..
41180    36
41181    37
41182    29
41184    46
41186    44
Name: age, Length: 36948, dtype: int64
```

In [24]:
```
data_1 = data_1[~(outliers_15_low|outliers_15_up)]
data_1
```

Out[24]:

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
2	37	226	1	999	0	1.1	93.994	-36.4	4.857	5191.0
3	40	151	1	999	0	1.1	93.994	-36.4	4.857	5191.0
5	45	198	1	999	0	1.1	93.994	-36.4	4.857	5191.0
7	41	217	1	999	0	1.1	93.994	-36.4	4.857	5191.0
8	24	380	1	999	0	1.1	93.994	-36.4	4.857	5191.0
...	...	...	...	...	...	...	...	...	...	...
41180	36	254	2	999	0	-1.1	94.767	-50.8	1.028	4963.6
41181	37	281	1	999	0	-1.1	94.767	-50.8	1.028	4963.6
41182	29	112	1	9	1	-1.1	94.767	-50.8	1.028	4963.6
41184	46	383	1	999	0	-1.1	94.767	-50.8	1.028	4963.6
41186	44	442	1	999	0	-1.1	94.767	-50.8	1.028	4963.6

36948 rows × 10 columns

In [25]:
```
sns.boxplot(x = data_1['age'])
plt.show()
```

jupyter Data Glacier project (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

Distribution on the basis of skewness value:

Skewness = 0: Then normally distributed.

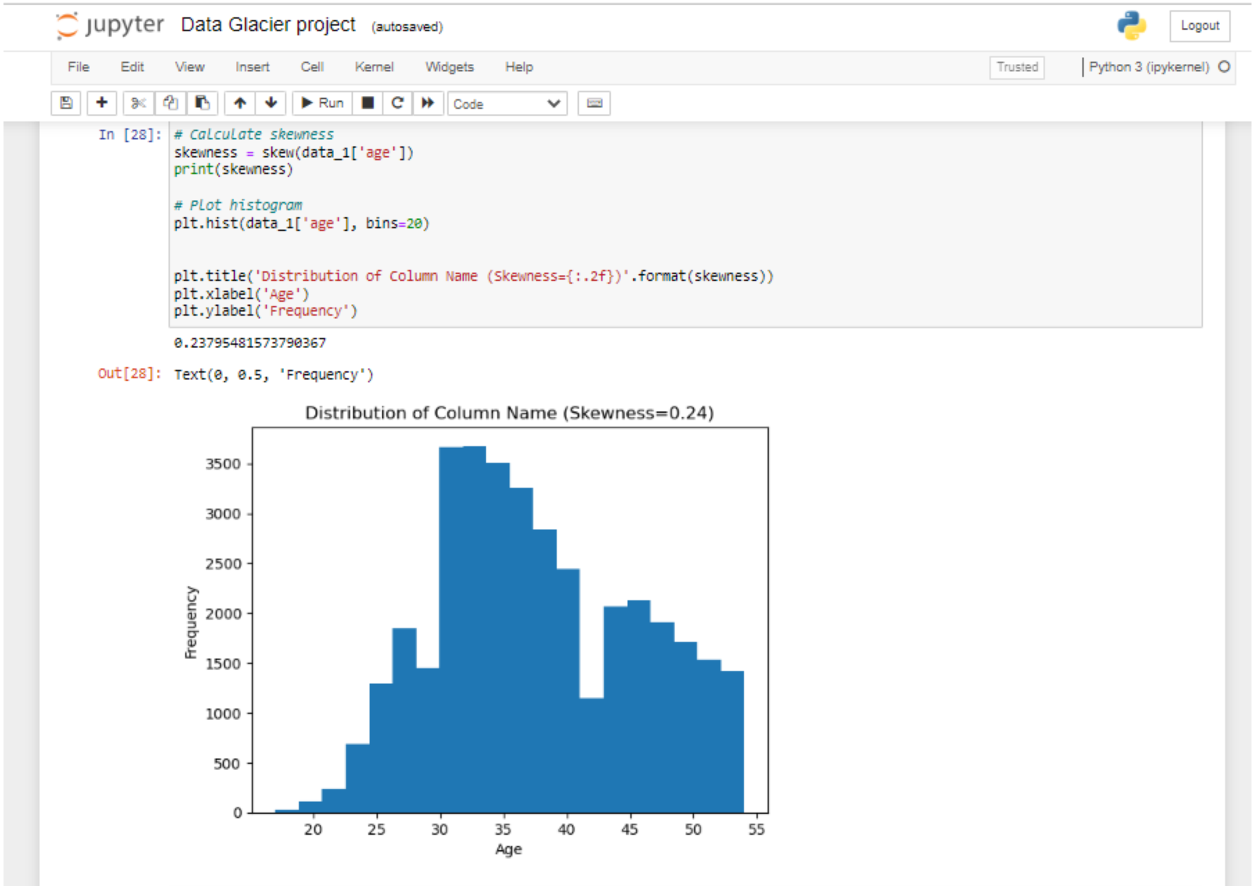
Skewness > 0: Then more weight in the left tail of the distribution.

Skewness < 0: Then more weight in the right tail of the distribution.

In [27]:
```
from scipy.stats import skew
# Calculate Pearson's skewness coefficient
skewness1 = skew(data_1)

# Calculate moment coefficient of skewness
skewness2 = np.mean((data_1 - np.mean(data_1)) ** 3) / (np.std(data_1) ** 3)

print("Pearson's skewness coefficient: ", skewness1)
print("Moment coefficient of skewness:\n", skewness2)
```

```
Pearson's skewness coefficient:  [ 0.23795482  3.3107468   4.79008024 -5.28224096  3.90406112 -0.74646063
 -0.19884943  0.27784703 -0.75398964 -1.08125485]
Moment coefficient of skewness:
 age              0.237955
duration         3.310747
campaign         4.790080
pdays           -5.282241
previous         3.904061
emp.var.rate    -0.746461
cons.price.idx  -0.198849
cons.conf.idx    0.277847
euribor3m       -0.753990
nr.employed     -1.081255
dtype: float64
```

```
C:\ProgramData\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:3430: FutureWarning: In a future version, DataFrame.mean(a
xis=None) will return a scalar mean over the entire DataFrame. To retain the old behavior, use 'frame.mean(axis=0)' or just 'fr
ame.mean()'
  return mean(axis=axis, dtype=dtype, out=out, **kwargs)
C:\ProgramData\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:3430: FutureWarning: In a future version, DataFrame.mean(a
xis=None) will return a scalar mean over the entire DataFrame. To retain the old behavior, use 'frame.mean(axis=0)' or just 'fr
ame.mean()'
  return mean(axis=axis, dtype=dtype, out=out, **kwargs)
```



Week 9 Assignment:

Handling missing values is an essential step in data cleaning and preprocessing. There are several methods to handle missing values, and choosing the right method depends on the nature of the data and the missing values.

Filling missing values using value_counts method

value_counts(): This method is useful when the missing values are represented as NaN. The value_counts() method returns a series containing counts of unique values, excluding NaN values. It can be used to determine the most frequent value in a column and impute the missing values with that value.

This column('marital') has 80 missing values. So, we can use the value_counts method to fill these values.

Forward or backward fill:

This method involves filling missing values with the previous (forward fill) or next (backward fill) known value. This method is useful when dealing with time series data where the missing value is expected to be like the previous or next value.

Mode imputation:
The mode is the most frequent value in a column. Mode imputation involves replacing the missing values with the mode of the column. This method is useful when the missing values are few compared to the total number of observations and the mode is a reasonable representation of the missing values.
Interpolation:
Interpolation is a statistical technique that involves estimating missing values based on the pattern observed in the data. Linear interpolation is commonly used to estimate missing values. This method is useful when dealing with continuous data and the missing values are not frequent.

```
In [29]: data_1['age'].fillna(data_1['age'].mean(), inplace=True)

In [30]: data_1['age'].describe()

Out[30]: count    36948.000000
         mean        37.731731
         std          8.032715
         min         17.000000
         25%         31.000000
         50%         37.000000
         75%         44.000000
         max         54.000000
         Name: age, dtype: float64

In [31]: data_1.info()

         <class 'pandas.core.frame.DataFrame'>
         Int64Index: 36948 entries, 2 to 41186
         Data columns (total 10 columns):
          #   Column         Non-Null Count  Dtype
         ---  ------         --------------  -----
          0   age            36948 non-null  int64
          1   duration       36948 non-null  int64
          2   campaign       36948 non-null  int64
          3   pdays          36948 non-null  int64
          4   previous       36948 non-null  int64
          5   emp.var.rate   36948 non-null  float64
          8   euribor3m      36948 non-null  float64
          9   nr.employed    36948 non-null  float64
         dtypes: float64(5), int64(5)
         memory usage: 3.1 MB

In [32]: # data[['age','duration','campaign','pdays','previous','emp.var.rate','cons.price.idx','cons.conf.idx',
```

Handling missing values in the dataset

Handling missing values is an essential step in data cleaning and preprocessing. There are several methods to handle missing values, and choosing the right method depends on the nature of the data and the missing values.

Filling missing values using value_counts method

value_counts(): This method is useful when the missing values are represented as NaN. The value_counts() method returns a series containing counts of unique values, excluding NaN values. It can be used to determine the most frequent value in a column and impute the missing values with that value.

This column('marital') has 80 missing values. So, we can use the value_counts method to fill these values.

```
In [33]: data['marital'].fillna(data['marital'].value_counts().index[0], inplace = True)
```

Fill missing values with forward or backward fill:

Forward or backward fill: This method involves filling missing values with the previous (forward fill) or next (backward fill) known value. This method is useful when dealing with time series data where the missing value is expected to be similar to the previous or next value.

In [34]:
```
data['housing'] = data['housing'].fillna(method='ffill')
data['default'] = data['default'].fillna(method='bfill')
```

Fill missing values with mode:

Mode imputation: The mode is the most frequent value in a column. Mode imputation involves replacing the missing values with the mode of the column. This method is useful when the missing values are few compared to the total number of observations and the mode is a reasonable representation of the missing values.

In [35]:
```
data['loan'] = data['loan'].fillna(data['loan'].mode().iloc[0])
```

Interpolation:

Interpolation: Interpolation is a statistical technique that involves estimating missing values based on the pattern observed in the data. Linear interpolation is commonly used to estimate missing values. This method is useful when dealing with continuous data and the missing values are not frequent.

In [36]:
```
data['job'] = data['job'].interpolate()
```

In [37]:
```
data.isna().sum()
```

Out[37]:
```
age                  0
job                330
marital              0
education         1730
default              0
housing              0
loan                 0
contact              0
month                0
day_of_week          0
duration             0
campaign             0
pdays                0
previous             0
poutcome             0
emp.var.rate         0
cons.price.idx       0
cons.conf.idx        0
euribor3m            0
nr.employed          0
y                    0
dtype: int64
```

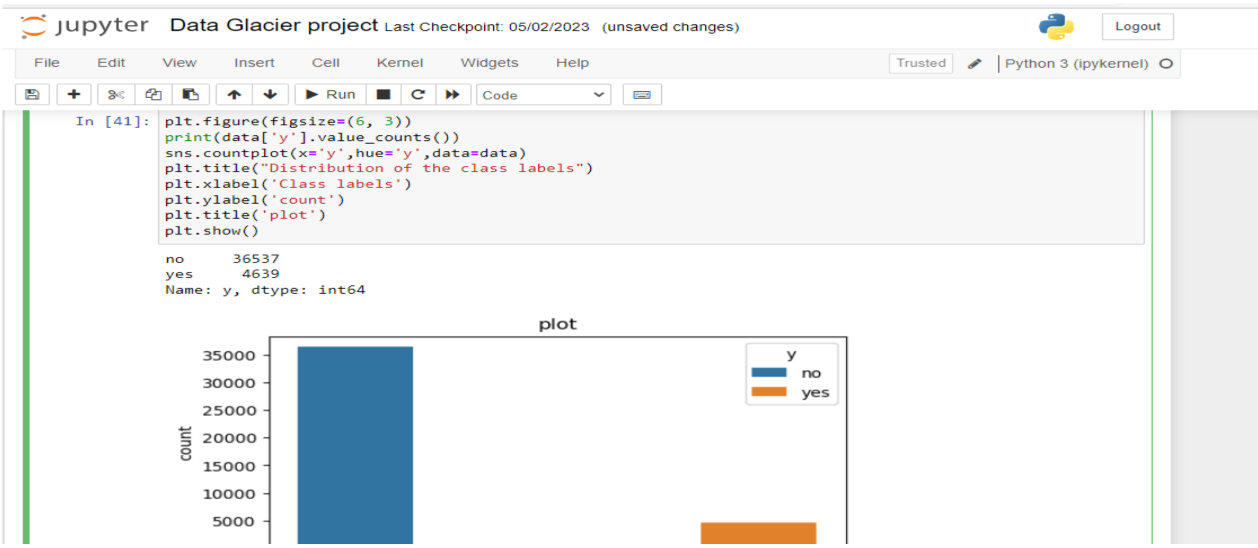
Week 10 Assignment:

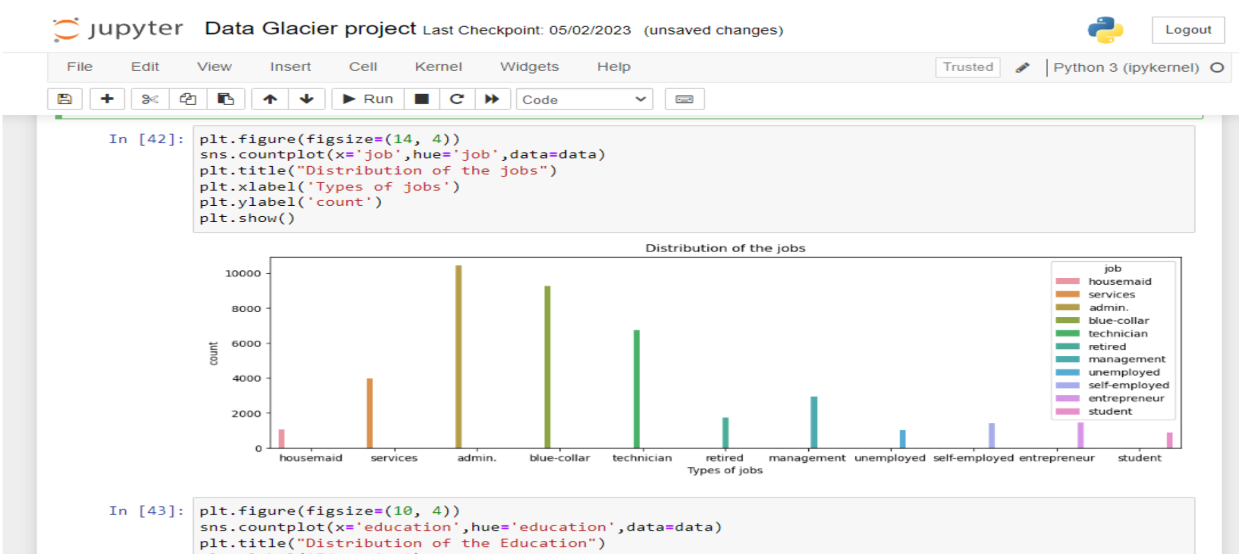
Exploratory Data Analysis (EDA) process:

Exploratory Data Analysis (EDA) is an important step in data science that involves analyzing and understanding the structure and characteristics of the data before applying any modeling techniques. The main objective of EDA is to identify patterns and relationships in the data that can inform the modeling process and ultimately lead to better insights and more accurate predictions.

Plotting features is an important part of Exploratory Data Analysis (EDA) in data science because it allows us to visually explore the distribution of the data and identify patterns and relationships between variables. By plotting the data, we can gain insights that may not be apparent from simple summary statistics or tabular data.

Plotting features can help to communicate findings to stakeholders clearly and concisely, using visualizations that are easy to understand. This can help to build trust in the analysis and improve the overall effectiveness of data-driven decision-making.
Below is plotted the distribution of a few features such as job, education, marital, housing, loan, contact, default, and month.





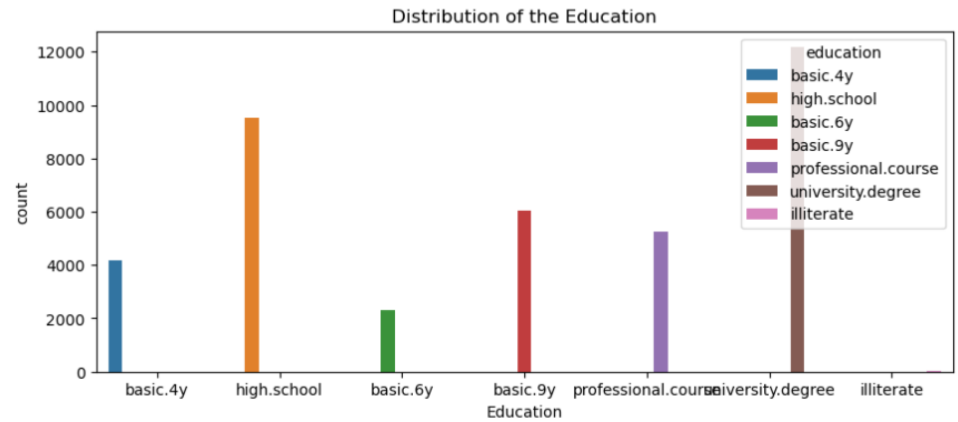
jupyter Data Glacier project Last Checkpoint: 05/02/2023 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

```
plt.ylabel('count')
plt.show()
```



In [44]:
```
plt.figure(figsize=(6, 3))
sns.countplot(x='marital',hue='marital',data=data)
plt.title("Distribution of the marital")
```

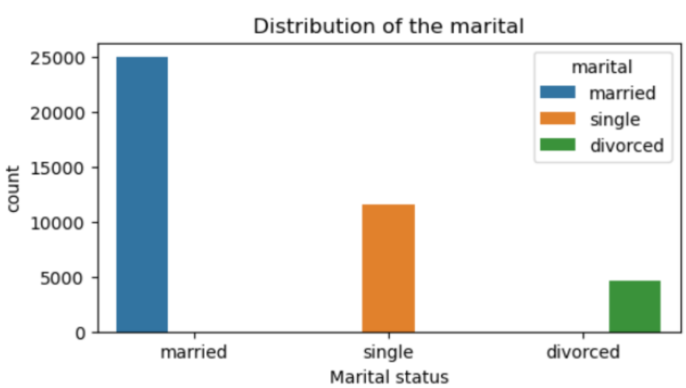
jupyter Data Glacier project Last Checkpoint: 05/02/2023 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

```
plt.title("Distribution of the marital")
plt.xlabel('Marital status')
plt.ylabel('count')
plt.show()
```



In [45]:
```
plt.figure(figsize=(6, 3))
sns.countplot(x='housing',hue='housing',data=data)
plt.title("Distribution of the housing")
```

jupyter Data Glacier project Last Checkpoint: 05/02/2023 (unsaved changes)

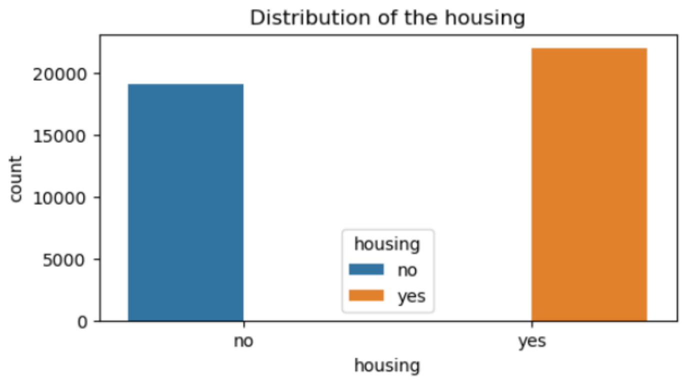
Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code

```
plt.xlabel('housing')
plt.ylabel('count')
plt.show()
```



In [46]:
```
plt.figure(figsize=(6, 3))
sns.countplot(x='loan',hue='loan',data=data)
plt.title("Distribution of the loan")
plt.xlabel('loan')
```


jupyter Data Glacier project Last Checkpoint: 05/02/2023 (unsaved changes)

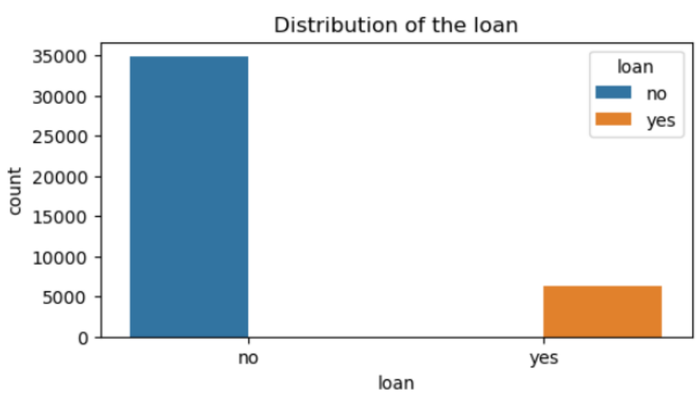
Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Code


```
plt.xlabel('loan')
plt.ylabel('count')
plt.show()
```

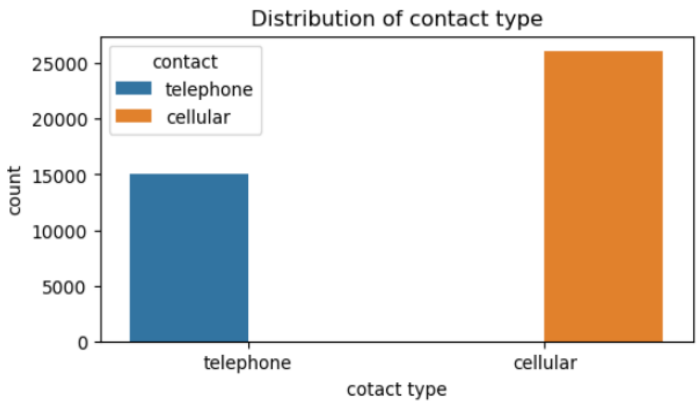


In [47]:
```
plt.figure(figsize=(6, 3))
sns.countplot(x='contact',hue='contact',data=data)
plt.title("Distribution of contact type")
plt.xlabel('cotact type')
plt.ylabel('count')
```

jupyter Data Glacier project Last Checkpoint: 05/02/2023 (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help | Trusted | Python 3 (ipykernel)

```
plt.xlabel('cotact type')
plt.ylabel('count')
plt.show()
```

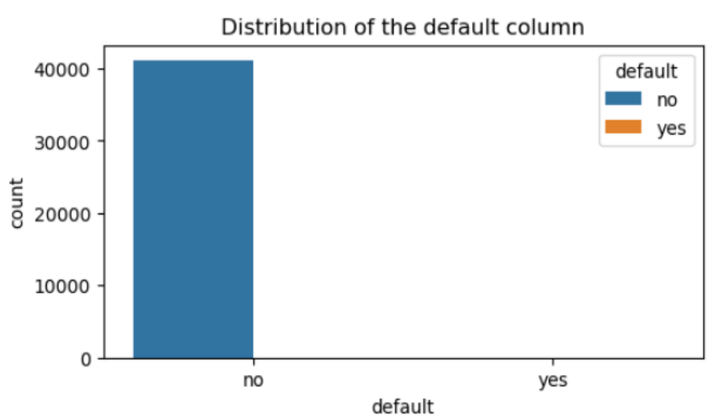


In [48]:
```
plt.figure(figsize=(6, 3))
sns.countplot(x='default',hue='default',data=data)
plt.title("Distribution of the default column")
plt.xlabel('default')
plt.ylabel('count')
```

jupyter Data Glacier project Last Checkpoint: 05/02/2023 (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help | Trusted | Python 3 (ipykernel)

```
plt.xlabel('default')
plt.ylabel('count')
plt.show()
```

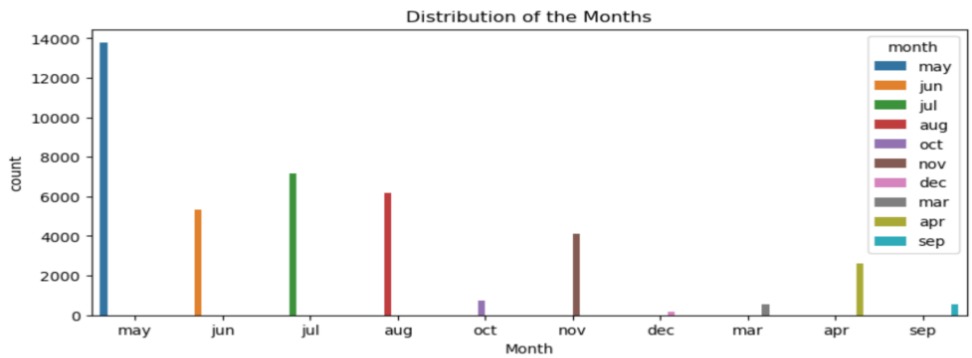


In [51]:
```
plt.figure(figsize=(10, 4))
sns.countplot(x='month',hue='month',data=data)
plt.title("Distribution of the Months")
plt.xlabel('Month')
plt.ylabel('count')
```

jupyter Data Glacier project Last Checkpoint: 05/02/2023 (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [51]:
```
plt.figure(figsize=(10, 4))
sns.countplot(x='month',hue='month',data=data)
plt.title("Distribution of the Months")
plt.xlabel('Month')
plt.ylabel('count')
plt.show()
```



Week 11 Assignment:

EDA presentation

In this assignment, we are plotting the relation between the dependent feature 'y' and independent features such as job, marital, education, default, housing, loan, contact, month, day_of_week, duration, campaign, and poutcome. And then, encoding a few features. Here, our data is imbalanced. Our problem is coming under an imbalanced binary classification.

jupyter Data Glacier project Last Checkpoint: 05/09/2023 (autosaved)

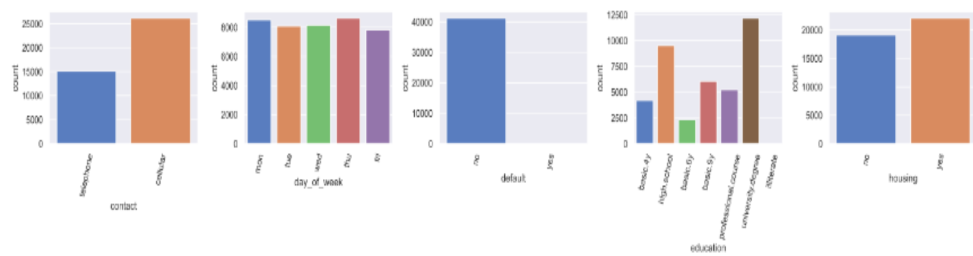
File Edit View Insert Cell Kernel Help Trusted Python 3 (ipykernel)

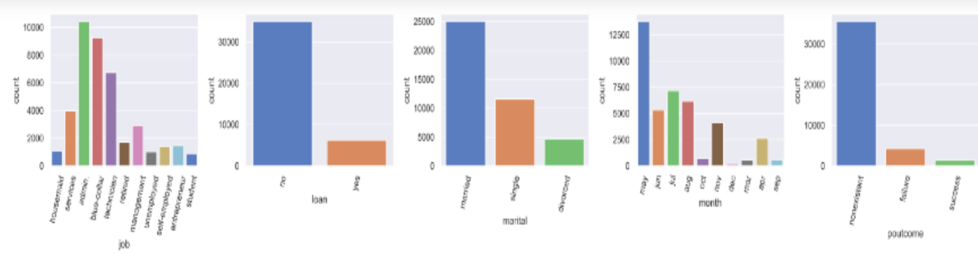
In [50]:
```
# Pick out the categorical, nominal, ordinal and numercial columns
cat_cols = np.array(['job', 'marital', 'education', 'default', 'housing', 'loan',
                     'contact', 'month', 'day_of_week', 'poutcome', 'y'])
ord_cols = np.array(['default', 'housing', 'loan', 'contact', 'poutcome', 'y'])
nom_cols = np.setdiff1d(cat_cols, ord_cols)
num_cols = np.setdiff1d(data.columns, cat_cols)
cat_cols = np.setdiff1d(cat_cols, ['y'])
ord_cols = np.setdiff1d(ord_cols, ['y'])

print("Categorical columns:", cat_cols)
print("Nominal columns:", ord_cols)
print("Ordinal columns:", nom_cols)
print("Numercial columns:", num_cols)

# data.info()
```

```
Categorical columns: ['contact' 'day_of_week' 'default' 'education' 'housing' 'job' 'loan'
 'marital' 'month' 'poutcome']
Nominal columns: ['contact' 'default' 'housing' 'loan' 'poutcome']
Ordinal columns: ['day_of_week' 'education' 'job' 'marital' 'month']
Numercial columns: ['age' 'campaign' 'cons.conf.idx' 'cons.price.idx' 'duration'
 'emp.var.rate' 'euribor3m' 'nr.employed' 'pdays' 'previous']
```

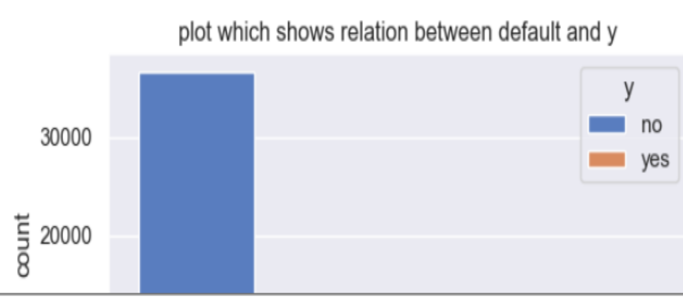
In [51]:
```
# Plot the distribbution of categorical attributes
def chunks(l, n):
    return [l[i:i + n] for i in range(0, len(l), n)]
```

```python
def histplot(df, cols, ncols):

    for lst in chunks(cols, ncols):
#         sns.set_style("white")  # Set the style to white
        sns.set(font_scale = 1)
        fig, axes = plt.subplots(nrows=1, ncols=ncols, figsize=(20, 4), dpi=600)
        for idx in range(0, len(lst)):
            attr = lst[idx]
            data = df[attr]
            sns.set_palette('muted', n_colors=30)
            g = sns.countplot(x=attr, data=df, ax=axes[idx])
            for item in g.get_xticklabels():
                item.set_rotation(75)
        plt.tight_layout()

#  sns.set_palette('Paired',30)
histplot(data[cat_cols], cat_cols, 5)
```



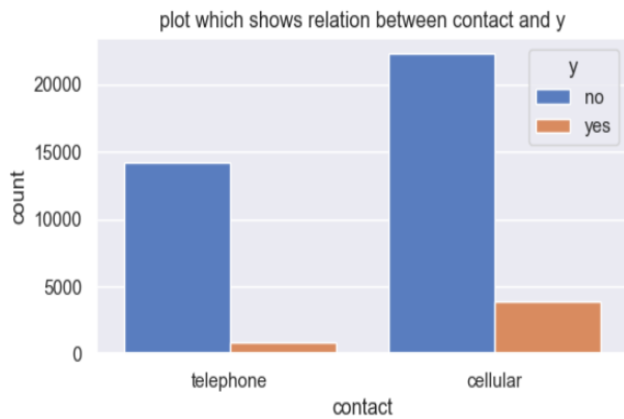


In [52]:
```python
plt.figure(figsize=(6, 3))
sns.countplot(x ='default', hue = "y", data = data)
plt.title('plot which shows relation between default and y')
# Show the plot
plt.show()
```



File Edit View Insert Cell Kernel Help Trusted Python 3 (ipykernel)

In [53]:
```
plt.figure(figsize=(6, 3))
sns.countplot(x ='contact', hue = "y", data = data)
plt.title('plot which shows relation between contact and y')
plt.show()
```

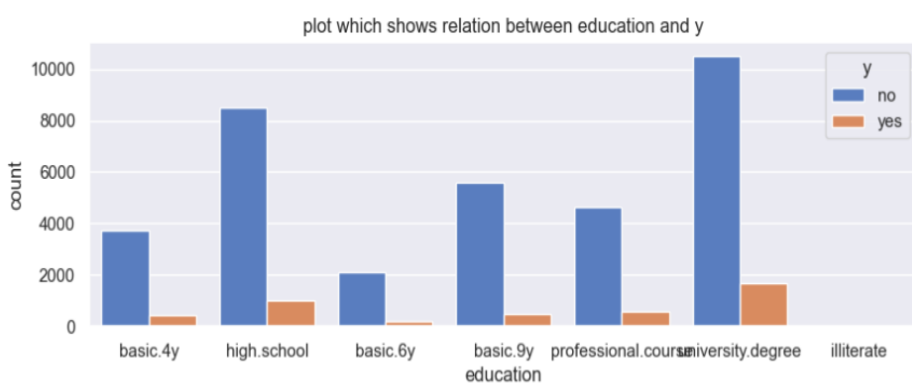


In [54]:
```
plt.figure(figsize=(10, 3))
sns.countplot(x ='education', hue = "y", data = data)
```

jupyter Data Glacier project Last Checkpoint: 05/09/2023 (autosaved) Logout

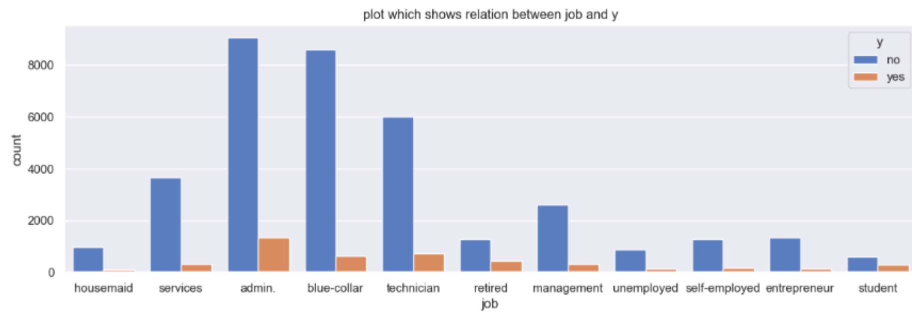
File Edit View Insert Cell Kernel Help Trusted Python 3 (ipykernel)

```
plt.title('plot which shows relation between education and y')
plt.show()
```



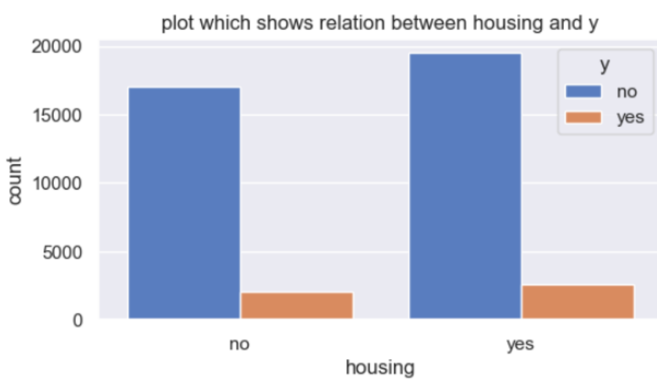
In [55]:
```
plt.figure(figsize=(14, 4))
sns.countplot(x ='job', hue = "y", data = data)
plt.title('plot which shows relation between job and y')
plt.show()
```

plot which shows relation between job and y

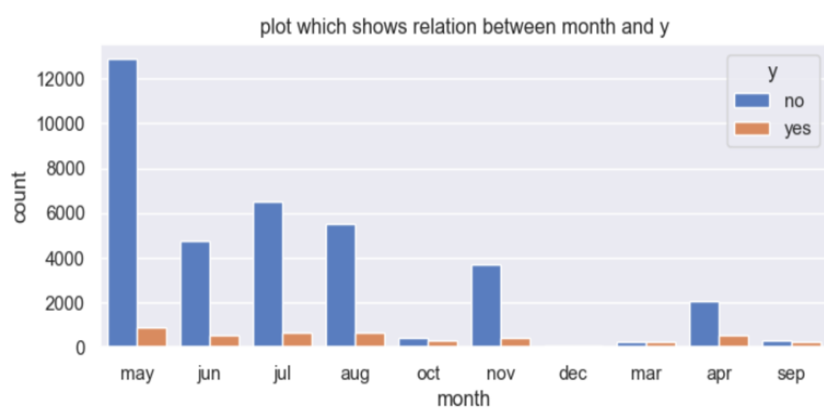



```
In [56]: plt.figure(figsize=(6, 3))
sns.countplot(x ='marital', hue = "y", data = data)
plt.title('plot which shows relation between marital and y')
plt.show()
```


```
In [57]: plt.figure(figsize=(6, 3))
sns.countplot(x ='housing', hue = "y", data = data)
plt.title('plot which shows relation between housing and y')
plt.show()
```

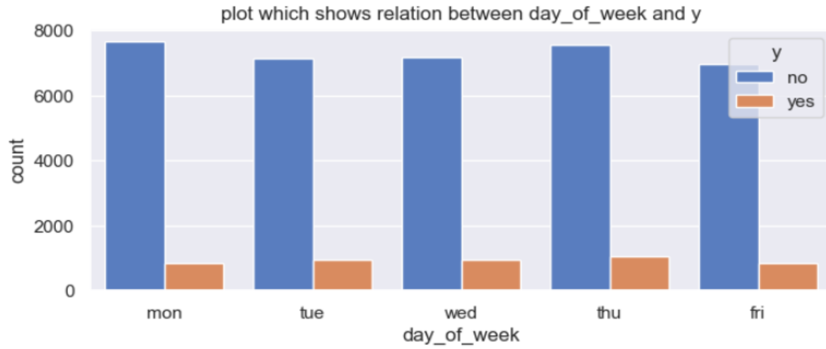



```
In [59]: plt.figure(figsize=(8, 3))
sns.countplot(x ='month', hue = "y", data = data)
plt.title('plot which shows relation between month and y')
plt.show()
```



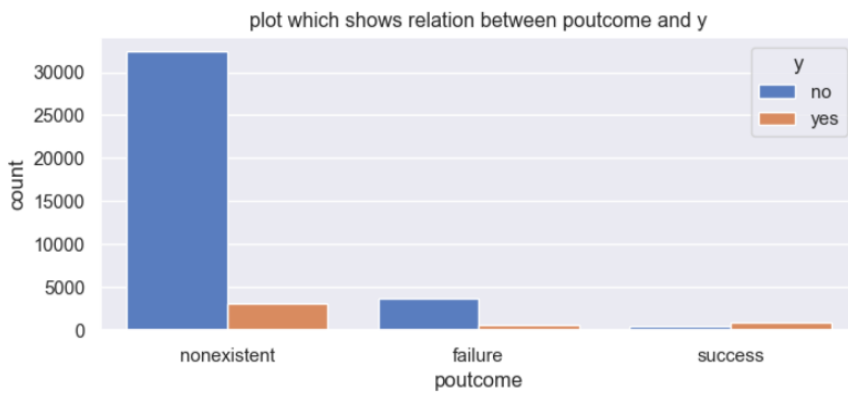
month

In [60]:
```
plt.figure(figsize=(8, 3))
sns.countplot(x ='day_of_week', hue = "y", data = data)
plt.title('plot which shows relation between day_of_week and y')
plt.show()
```

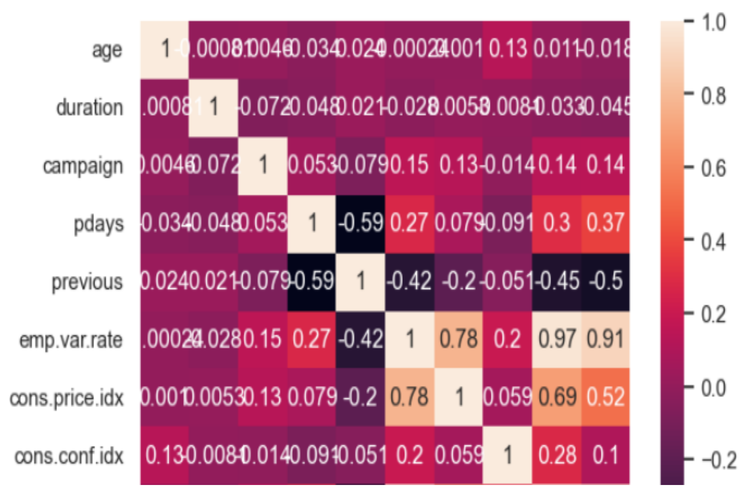


day_of_week

In [61]:
```
plt.figure(figsize=(8, 3))
sns.countplot(x ='poutcome', hue = "y", data = data)
plt.title('plot which shows relation between poutcome and y')
plt.show()
```




```
only valid columns or specify the value of numeric_only to silence this warning.
  matrix = data.corr()
```



In [66]:
```
data['contact'].replace({'telephone':0, 'cellular':1}, inplace=True)
data['default'].replace({'no':-1, 'unknown':0, 'yes':1}, inplace=True)
data['housing'].replace({'no':-1, 'unknown':0, 'yes':1}, inplace=True)
data['loan'].replace({'no':-1, 'unknown':0, 'yes':1}, inplace=True)
data['poutcome'].replace({'failure':-1, 'nonexistent':0, 'success':1}, inplace=True)
```

In [67]:
```
data.head(5)
```

Out[67]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous
0	56	housemaid	married	basic.4y	-1	-1	-1	0	may	mon	...	1	999	0
1	57	services	married	high.school	-1	-1	-1	0	may	mon	...	1	999	0
2	37	services	married	high.school	-1	1	-1	0	may	mon	...	1	999	0
3	40	admin.	married	basic.6y	-1	-1	-1	0	may	mon	...	1	999	0
4	56	services	married	high.school	-1	-1	1	0	may	mon	...	1	999	0

5 rows × 21 columns

In [ ]: