In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv(r"C:\Users\611ev\OneDrive\Desktop\evainternship\task 03\bank\bank.csv")
df.head()
```

Out[2]:

	age;"job";"marital";"education";"default";"balance";"housing";"loan";"contact";"day";"month";"duration";"campaign";"pdays";"previous";"pou
0	30;"unemployed";"married";"primary"
1	33;"services";"married";"secondary"
2	35;"management";"single";"tertiary"
3	30;"management";"married";"tertiary'
4	59;"blue-collar";"married";"second

In [3]:
```
df.tail()
```

Out[3]:

	age;"job";"marital";"education";"default";"balance";"housing";"loan";"contact";"day";"month";"duration";"campaign";"pdays";"previous";"
4516	33;"services";"married";"second
4517	57;"self-employed";"married";
4518	57;"technician";"married";"secon
4519	28;"blue-collar";"married";"seco
4520	44;"entrepreneur";"single";"ter

In [4]:
```
df.shape
```

Out[4]: (4521, 1)

In [5]:
```
df.columns
```

Out[5]:
```
Index(['age;"job";"marital";"education";"default";"balance";"housing";"loan";"contact";"day";"month";"duration";"campaign";"p
days";"previous";"poutcome";"y"'], dtype='object')
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4521 entries, 0 to 4520
Data columns (total 1 columns):
 #   Column
Non-Null Count  Dtype
---  ------
--------------  -----
 0   age;"job";"marital";"education";"default";"balance";"housing";"loan";"contact";"day";"month";"duration";"campaign";"pday
s";"previous";"poutcome";"y"  4521 non-null   object
dtypes: object(1)
memory usage: 35.4+ KB
```

In [7]:
```
df.describe()
```

Out[7]:

	age;"job";"marital";"education";"default";"balance";"housing";"loan";"contact";"day";"month";"duration";"campaign";"pdays";"previous'
count	
unique	
top	30;"unemployed";"married";"pri
freq	

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
age;"job";"marital";"education";"default";"balance";"housing";"loan";"contact";"day";"month";"duration";"campaign";"pdays";"p
revious";"poutcome";"y"    0
dtype: int64
```

In [9]:
```
plt.figure(figsize = (16,9))
sns.countplot(x = "job",data = df)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[9], line 2
      1 plt.figure(figsize = (16,9))
----> 2 sns.countplot(x = "job",data = df)

File ~\AppData\Roaming\Python\Python312\site-packages\seaborn\categorical.py:2631, in countplot(data, x, y, hue, order, hue_ord
er, orient, color, palette, saturation, fill, hue_norm, stat, width, dodge, gap, log_scale, native_scale, formatter, legend, a
x, **kwargs)
   2628 elif x is not None and y is not None:
   2629     raise TypeError("Cannot pass values for both `x` and `y`.")
-> 2631 p = _CategoricalAggPlotter(
   2632     data=data,
   2633     variables=dict(x=x, y=y, hue=hue),
   2634     order=order,
   2635     orient=orient,
   2636     color=color,
   2637     legend=legend,
   2638 )
   2640 if ax is None:
   2641     ax = plt.gca()

File ~\AppData\Roaming\Python\Python312\site-packages\seaborn\categorical.py:67, in _CategoricalPlotter.__init__(self, data, va
riables, order, orient, require_numeric, color, legend)
     56 def __init__(
     57     self,
     58     data=None,
   (...)
     64     legend="auto",
     65 ):
---> 67     super().__init__(data=data, variables=variables)
     69     # This method takes care of some bookkeeping that is necessary because the
     70     # original categorical plots (prior to the 2021 refactor) had some rules that
     71     # don't fit exactly into VectorPlotter logic. It may be wise to have a second
   (...)
     76     # default VectorPlotter rules. If we do decide to make orient part of the
     77     # _base variable assignment, we'll want to figure out how to express that.
     78     if self.input_format == "wide" and orient in ["h", "y"]:

File ~\AppData\Roaming\Python\Python312\site-packages\seaborn\_base.py:634, in VectorPlotter.__init__(self, data, variables)
```

```
    629 # var_ordered is relevant only for categorical axis variables, and may
    630 # be better handled by an internal axis information object that tracks
    631 # such information and is set up by the scale_* methods. The analogous
    632 # information for numeric axes would be information about log scales.
    633 self._var_ordered = {"x": False, "y": False}  # alt., used DefaultDict
--> 634 self.assign_variables(data, variables)
    636 # TODO Lots of tests assume that these are called to initialize the
    637 # mappings to default values on class initialization. I'd prefer to
    638 # move away from that and only have a mapping when explicitly called.
    639 for var in ["hue", "size", "style"]:

File ~\AppData\Roaming\Python\Python312\site-packages\seaborn\_base.py:679, in VectorPlotter.assign_variables(self, data, variables)
    674 else:
    675     # When dealing with long-form input, use the newer PlotData
    676     # object (internal but introduced for the objects interface)
    677     # to centralize / standardize data consumption logic.
    678     self.input_format = "long"
--> 679     plot_data = PlotData(data, variables)
    680     frame = plot_data.frame
    681     names = plot_data.names

File ~\AppData\Roaming\Python\Python312\site-packages\seaborn\_core\data.py:58, in PlotData.__init__(self, data, variables)
     51 def __init__(
     52     self,
     53     data: DataSource,
     54     variables: dict[str, VariableSpec],
     55 ):
     57     data = handle_data_source(data)
---> 58     frame, names, ids = self._assign_variables(data, variables)
     60     self.frame = frame
     61     self.names = names

File ~\AppData\Roaming\Python\Python312\site-packages\seaborn\_core\data.py:232, in PlotData._assign_variables(self, data, variables)
    230     else:
    231         err += "An entry with this name does not appear in `data`."
--> 232     raise ValueError(err)
    234 else:
    235
    236     # Otherwise, assume the value somehow represents data
```

```
    237
    238     # Ignore empty data structures
    239     if isinstance(val, Sized) and len(val) == 0:

ValueError: Could not interpret value `job` for `x`. An entry with this name does not appear in `data`.
<Figure size 1600x900 with 0 Axes>
```

In [10]:
```
sns.countplot(x = "job",data = df)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[10], line 1
----> 1 sns.countplot(x = "job",data = df)

File ~\AppData\Roaming\Python\Python312\site-packages\seaborn\categorical.py:2631, in countplot(data, x, y, hue, order, hue_ord
er, orient, color, palette, saturation, fill, hue_norm, stat, width, dodge, gap, log_scale, native_scale, formatter, legend, a
x, **kwargs)
   2628 elif x is not None and y is not None:
   2629     raise TypeError("Cannot pass values for both `x` and `y`.")
-> 2631 p = _CategoricalAggPlotter(
   2632     data=data,
   2633     variables=dict(x=x, y=y, hue=hue),
   2634     order=order,
   2635     orient=orient,
   2636     color=color,
   2637     legend=legend,
   2638 )
   2640 if ax is None:
   2641     ax = plt.gca()

File ~\AppData\Roaming\Python\Python312\site-packages\seaborn\categorical.py:67, in _CategoricalPlotter.__init__(self, data, va
riables, order, orient, require_numeric, color, legend)
     56 def __init__(
     57     self,
     58     data=None,
   (...)
     64     legend="auto",
     65 ):
---> 67     super().__init__(data=data, variables=variables)
     69     # This method takes care of some bookkeeping that is necessary because the
     70     # original categorical plots (prior to the 2021 refactor) had some rules that
     71     # don't fit exactly into VectorPlotter logic. It may be wise to have a second
   (...)
     76     # default VectorPlotter rules. If we do decide to make orient part of the
     77     # _base variable assignment, we'll want to figure out how to express that.
     78     if self.input_format == "wide" and orient in ["h", "y"]:

File ~\AppData\Roaming\Python\Python312\site-packages\seaborn\_base.py:634, in VectorPlotter.__init__(self, data, variables)
    629 # var_ordered is relevant only for categorical axis variables, and may
```

```
    630 # be better handled by an internal axis information object that tracks
    631 # such information and is set up by the scale_* methods. The analogous
    632 # information for numeric axes would be information about log scales.
    633 self._var_ordered = {"x": False, "y": False}  # alt., used DefaultDict
--> 634 self.assign_variables(data, variables)
    636 # TODO Lots of tests assume that these are called to initialize the
    637 # mappings to default values on class initialization. I'd prefer to
    638 # move away from that and only have a mapping when explicitly called.
    639 for var in ["hue", "size", "style"]:

File ~\AppData\Roaming\Python\Python312\site-packages\seaborn\_base.py:679, in VectorPlotter.assign_variables(self, data, varia
bles)
    674 else:
    675     # When dealing with long-form input, use the newer PlotData
    676     # object (internal but introduced for the objects interface)
    677     # to centralize / standardize data consumption logic.
    678     self.input_format = "long"
--> 679     plot_data = PlotData(data, variables)
    680     frame = plot_data.frame
    681     names = plot_data.names

File ~\AppData\Roaming\Python\Python312\site-packages\seaborn\_core\data.py:58, in PlotData.__init__(self, data, variables)
     51 def __init__(
     52     self,
     53     data: DataSource,
     54     variables: dict[str, VariableSpec],
     55 ):
     57     data = handle_data_source(data)
---> 58     frame, names, ids = self._assign_variables(data, variables)
     60     self.frame = frame
     61     self.names = names

File ~\AppData\Roaming\Python\Python312\site-packages\seaborn\_core\data.py:232, in PlotData._assign_variables(self, data, vari
ables)
    230     else:
    231         err += "An entry with this name does not appear in `data`."
--> 232     raise ValueError(err)
    234 else:
    235
    236     # Otherwise, assume the value somehow represents data
    237
```

```
    238     # Ignore empty data structures
    239     if isinstance(val, Sized) and len(val) == 0:

ValueError: Could not interpret value `job` for `x`. An entry with this name does not appear in `data`.
```

In [ ]: