```
import pandas as pd
store=pd.read_csv(r'/content/train.csv')
store
```

	User_ID	Product_ID	Gender	Age	Occupation
0	1000001	P00069042	F	0-17	10
1	1000001	P00248942	F	0-17	10
2	1000001	P00087842	F	0-17	10
3	1000001	P00085442	F	0-17	10
4	1000002	P00285442	M	55+	16
...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13
550064	1006035	P00375436	F	26-35	1
550065	1006036	P00375436	F	26-35	15
550066	1006038	P00375436	F	55+	1
550067	1006039	P00371644	F	46-50	0

550068 rows × 12 columns

```
pd.__version__
```

```
'2.2.2'
```

```
len(store)
```

```
550068
```

```
store.shape
```

```
(550068, 12)
```

```
store.columns
```

Gemini

b vis3=sns.lmplot(data=store,x='Age',y=

Okay, let's break this down:

This line of code is creating a scatter plot with a fitted regression line using the `seaborn` library.

Here's a step-by-step explanation:

1. **`vis3 = ...`**: This part assigns the resulting plot to a variable named `vis3`, allowing you to further customize or display it later.
2. **`sns.lmplot(...)`**: This calls the `lmplot` function from the `seaborn` library, which is designed for creating these types of plots.
3. **`data=store`**: This specifies that the data for the plot should come from your Pandas DataFrame called `store`.
4. **`x='Age', y='Occupation'`**: This determines which columns from your DataFrame will be used for the x and y axes of the scatter plot. In this case, 'Age' values will be plotted on the horizontal axis, and 'Occupation' values on the vertical axis.
5. **`fit_reg=True`**: This argument is crucial for including the regression line. By setting it to `True`, `seaborn` will calculate and draw a line of best fit through your data points. This line helps visualize the general trend or relationship between 'Age' and 'Occupation'.

To see the plot, run the code.

Rate this answer

```
Index(['User_ID', 'Product_ID', 'Gender', 'Age',
'Occupation', 'City_Category',
       'Stay_In_Current_City_Years',
'Marital_Status', 'Product_Category_1',
       'Product_Category_2', 'Product_Category_3',
'Purchase'],
      dtype='object')
```

```
type(store)
```

```
pandas.core.frame.DataFrame
def __init__(data=None, index: Axes | None=None,
columns: Axes | None=None, dtype: Dtype |
None=None, copy: bool | None=None) -> None
```

```
/usr/local/lib/python3.11/dist-packages/pandas/co
Two-dimensional, size-mutable, potentially hetero

Data structure also contains labeled axes (rows a
Arithmetic operations align on both row and colum
```

```
store
```

	User_ID	Product_ID	Gender	Age	Occupation
0	1000001	P00069042	F	0-17	10
1	1000001	P00248942	F	0-17	10
2	1000001	P00087842	F	0-17	10
3	1000001	P00085442	F	0-17	10
4	1000002	P00285442	M	55+	16
...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13
550064	1006035	P00375436	F	26-35	1
550065	1006036	P00375436	F	26-35	15
550066	1006038	P00375436	F	55+	1
550067	1006039	P00371644	F	46-50	0

550068 rows × 12 columns

```
len(store.columns)
```

b vis3=sns.lmplot(data=store,x='Age',y=
what is the error

Okay, let's figure out the error.

The error you're encountering with the code `vis3=sns.lmplot(data=store,x='` is likely because the **'Age'** column in your DataFrame might not be numerical. The `lmplot` function in `seaborn` primarily works with numerical data for regression analysis.

Here's a breakdown of the error and the solution:

Error:

- `lmplot` expects numerical data for 'x' and 'y' to perform regression and plot the line of best fit. If 'Age' contains non-numeric values (e.g., strings, categories), the function will raise an error.

Solution:

1. **Check the data type of 'Age':**

```
print(store['Age'].dtype)
```

Use code with caution

- If it's not a numerical type (like `int64` or `float64`), you'll need to convert it.

2. **Convert 'Age' to numerical if needed:**
    - If 'Age' has categories represented as strings (e.g., '0-17', '18-25', etc.), you can use methods like `pd.factorize()` or `pd.Categorical()` to convert them to numerical representations. Here's an example using `pd.factorize()`:

```
12
```

```
store.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 12 columns):
 #   Column                      Non-Null Count   Dty
---  ------                      --------------   ---
 0   User_ID                     550068 non-null  int
 1   Product_ID                  550068 non-null  obj
 2   Gender                      550068 non-null  obj
 3   Age                         550068 non-null  obj
 4   Occupation                  550068 non-null  int
 5   City_Category               550068 non-null  obj
 6   Stay_In_Current_City_Years  550068 non-null  obj
 7   Marital_Status              550068 non-null  int
 8   Product_Category_1          550068 non-null  int
 9   Product_Category_2          376430 non-null  flc
 10  Product_Category_3          166821 non-null  flc
 11  Purchase                    550068 non-null  int
dtypes: float64(2), int64(5), object(5)
memory usage: 50.4+ MB
```

```
store.head()
```

	User_ID	Product_ID	Gender	Age	Occupation	City
0	1000001	P00069042	F	0-17	10	
1	1000001	P00248942	F	0-17	10	
2	1000001	P00087842	F	0-17	10	
3	1000001	P00085442	F	0-17	10	
4	1000002	P00285442	M	55+	16	

```
store.tail()
```

```
l.factorize(store['Age'])[0] # (
=store, x='Age_Encoded', y='Oc
```

Use code with caution

Explanation:

- We first encoded the 'Age' column into numerical values using `pd.factorize()`. This creates a new column (e.g., 'Age_Encoded') with numerical representations of the age categories.
- Then, in the `lmplot` function, we used this new 'Age_Encoded' column for the x-axis, ensuring the regression analysis is performed on numerical data.

By converting 'Age' to a numerical format, you should be able to resolve the error and generate the scatter plot with the regression line successfully.

Rate this answer

	User_ID	Product_ID	Gender	Age	Occupation
550063	1006033	P00372445	M	51-55	13
550064	1006035	P00375436	F	26-35	1
550065	1006036	P00375436	F	26-35	15
550066	1006038	P00375436	F	55+	1
550067	1006039	P00371644	F	46-50	0

```
store.tail(2)
```

	User_ID	Product_ID	Gender	Age	Occupation
550066	1006038	P00375436	F	55+	1
550067	1006039	P00371644	F	46-50	0

```
store.head(3)
```

	User_ID	Product_ID	Gender	Age	Occupation	City
0	1000001	P00069042	F	0-17	10	
1	1000001	P00248942	F	0-17	10	
2	1000001	P00087842	F	0-17	10	

```
store.head(7)
```

	User_ID	Product_ID	Gender	Age	Occupation	City
0	1000001	P00069042	F	0-17	10	
1	1000001	P00248942	F	0-17	10	
2	1000001	P00087842	F	0-17	10	
3	1000001	P00085442	F	0-17	10	
4	1000002	P00285442	M	55+	16	
5	1000003	P00193542	M	26-35	15	
6	1000004	P00184942	M	46-50	7	

```
store[::-1]
```

	User_ID	Product_ID	Gender	Age	Occupation
550067	1006039	P00371644	F	46-50	0
550066	1006038	P00375436	F	55+	1
550065	1006036	P00375436	F	26-35	15
550064	1006035	P00375436	F	26-35	1
550063	1006033	P00372445	M	51-55	13
...	...	...	...	...	...
4	1000002	P00285442	M	55+	16
3	1000001	P00085442	F	0-17	10
2	1000001	P00087842	F	0-17	10
1	1000001	P00248942	F	0-17	10
0	1000001	P00069042	F	0-17	10

550068 rows × 12 columns

```
store[:5]
```

	User_ID	Product_ID	Gender	Age	Occupation	City
0	1000001	P00069042	F	0-17	10	
1	1000001	P00248942	F	0-17	10	
2	1000001	P00087842	F	0-17	10	
3	1000001	P00085442	F	0-17	10	
4	1000002	P00285442	M	55+	16	

```
store[3:]
```

	User_ID	Product_ID	Gender	Age	Occupation
3	1000001	P00085442	F	0-17	10
4	1000002	P00285442	M	55+	16
5	1000003	P00193542	M	26-35	15
6	1000004	P00184942	M	46-50	7
7	1000004	P00346142	M	46-50	7
...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13
550064	1006035	P00375436	F	26-35	1
550065	1006036	P00375436	F	26-35	15
550066	1006038	P00375436	F	55+	1
550067	1006039	P00371644	F	46-50	0

550065 rows × 12 columns

```
store[0:200:10]
```

	User_ID	Product_ID	Gender	Age	Occupation	Ci
0	1000001	P00069042	F	0-17	10	
10	1000005	P00251242	M	26-35	20	
20	1000008	P00220442	M	26-35	12	
30	1000010	P00118742	F	36-45	1	
40	1000010	P00087242	F	36-45	1	
50	1000012	P00304242	M	26-35	12	
60	1000015	P00333042	M	26-35	7	
70	1000018	P00366542	F	18-25	3	
80	1000018	P0094142	F	18-25	3	
90	1000019	P00249642	M	0-17	10	
100	1000022	P00195942	M	18-25	15	
110	1000022	P00280542	M	18-25	15	
120	1000023	P00032042	M	36-45	0	
130	1000026	P00043242	M	26-35	7	
140	1000026	P00101342	M	26-35	7	
150	1000028	P00178942	F	26-35	1	
160	1000031	P00322042	M	55+	7	
170	1000033	P00351842	M	46-50	3	
180	1000034	P00247042	F	18-25	0	
190	1000035	P00285442	M	46-50	1	

```
store.describe()
```

	User_ID	Occupation	Marital_Status	P
count	5.500680e+05	550068.000000	550068.000000	
mean	1.003029e+06	8.076707	0.409653	
std	1.727592e+03	6.522660	0.491770	
min	1.000001e+06	0.000000	0.000000	
25%	1.001516e+06	2.000000	0.000000	
50%	1.003077e+06	7.000000	0.000000	
75%	1.004478e+06	14.000000	1.000000	
max	1.006040e+06	20.000000	1.000000	

```
store.describe().transpose()
```

	count	mean	st
User_ID	550068.0	1.003029e+06	1727.59158
Occupation	550068.0	8.076707e+00	6.52266
Marital_Status	550068.0	4.096530e-01	0.49177
Product_Category_1	550068.0	5.404270e+00	3.93621
Product_Category_2	376430.0	9.842329e+00	5.08659
Product_Category_3	166821.0	1.266824e+01	4.12533
Purchase	550068.0	9.263969e+03	5023.06539

```
store.head()
```

	User_ID	Product_ID	Gender	Age	Occupation	City
0	1000001	P00069042	F	0-17	10	
1	1000001	P00248942	F	0-17	10	
2	1000001	P00087842	F	0-17	10	
3	1000001	P00085442	F	0-17	10	
4	1000002	P00285442	M	55+	16	

```
len(store.columns)
```

```
12
```

```
#store.columns=['a','b','c','d','e','f','g','h','i','j','
```

```
store.head()
```

	User_ID	Product_ID	Gender	Age	Occupation	City
0	1000001	P00069042	F	0-17	10	
1	1000001	P00248942	F	0-17	10	
2	1000001	P00087842	F	0-17	10	
3	1000001	P00085442	F	0-17	10	
4	1000002	P00285442	M	55+	16	

```
store.columns
```

```
Index(['User_ID', 'Product_ID', 'Gender', 'Age',
'Occupation', 'City_Category',
       'Stay_In_Current_City_Years',
'Marital_Status', 'Product_Category_1',
       'Product_Category_2', 'Product_Category_3',
'Purchase'],
      dtype='object')
```

```
store.head()
```

	User_ID	Product_ID	Gender	Age	Occupation	City
0	1000001	P00069042	F	0-17	10	
1	1000001	P00248942	F	0-17	10	
2	1000001	P00087842	F	0-17	10	
3	1000001	P00085442	F	0-17	10	
4	1000002	P00285442	M	55+	16	

```
store.columns
```

```
Index(['User_ID', 'Product_ID', 'Gender', 'Age',
'Occupation', 'City_Category',
       'Stay_In_Current_City_Years',
'Marital_Status', 'Product_Category_1',
       'Product_Category_2', 'Product_Category_3',
'Purchase'],
      dtype='object')
```

```
store[:]
```

	User_ID	Product_ID	Gender	Age	Occupation
0	1000001	P00069042	F	0-17	10
1	1000001	P00248942	F	0-17	10
2	1000001	P00087842	F	0-17	10
3	1000001	P00085442	F	0-17	10
4	1000002	P00285442	M	55+	16
...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13
550064	1006035	P00375436	F	26-35	1
550065	1006036	P00375436	F	26-35	15
550066	1006038	P00375436	F	55+	1
550067	1006039	P00371644	F	46-50	0

550068 rows × 12 columns

```
store.isnull().sum
```

```
pandas.core.frame.DataFrame.sum
def sum(axis: Axis | None=0, skipna: bool=True,
numeric_only: bool=False, min_count: int=0,
**kwargs)

/usr/local/lib/python3.11/dist-packages/pandas/co
Return the sum of the values over the requested a

This is equivalent to the method ``numpy.sum``.
```

```
store.isnull()
```

	User_ID	Product_ID	Gender	Age	Occupatio
0	False	False	False	False	Fals
1	False	False	False	False	Fals
2	False	False	False	False	Fals
3	False	False	False	False	Fals
4	False	False	False	False	Fals
...	...	...	...	...	.
550063	False	False	False	False	Fals
550064	False	False	False	False	Fals
550065	False	False	False	False	Fals
550066	False	False	False	False	Fals
550067	False	False	False	False	Fals

550068 rows × 12 columns

```
store.isna()
```

	User_ID	Product_ID	Gender	Age	Occupatio
0	False	False	False	False	Fals
1	False	False	False	False	Fals
2	False	False	False	False	Fals
3	False	False	False	False	Fals
4	False	False	False	False	Fals
...	...	...	...	...	.
550063	False	False	False	False	Fals
550064	False	False	False	False	Fals
550065	False	False	False	False	Fals
550066	False	False	False	False	Fals
550067	False	False	False	False	Fals

550068 rows × 12 columns

```
store.isnull().sum()
```

```
                                    0
User_ID                             0
Product_ID                          0
Gender                              0
Age                                 0
Occupation                          0
City_Category                       0
Stay_In_Current_City_Years          0
Marital_Status                      0
Product_Category_1                  0
Product_Category_2             173638
Product_Category_3             383247
Purchase                            0

dtype: int64
```

```
store.dtypes
```

```
                                    0
User_ID                         int64
Product_ID                     object
Gender                         object
Age                            object
Occupation                      int64
City_Category                  object
Stay_In_Current_City_Years     object
Marital_Status                  int64
Product_Category_1              int64
Product_Category_2            float64
Product_Category_3            float64
Purchase                        int64

dtype: object
```

```
store.columns
```

```
Index(['User_ID', 'Product_ID', 'Gender', 'Age',
'Occupation', 'City_Category',
       'Stay_In_Current_City_Years',
'Marital_Status', 'Product_Category_1',
       'Product_Category_2', 'Product_Category_3',
'Purchase'],
      dtype='object')
```

```
store.head()
```

	User_ID	Product_ID	Gender	Age	Occupation	City
0	1000001	P00069042	F	0-17	10	
1	1000001	P00248942	F	0-17	10	
2	1000001	P00087842	F	0-17	10	
3	1000001	P00085442	F	0-17	10	
4	1000002	P00285442	M	55+	16	

```
store
```

	User_ID	Product_ID	Gender	Age	Occupation
0	1000001	P00069042	F	0-17	10
1	1000001	P00248942	F	0-17	10
2	1000001	P00087842	F	0-17	10
3	1000001	P00085442	F	0-17	10
4	1000002	P00285442	M	55+	16
...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13
550064	1006035	P00375436	F	26-35	1
550065	1006036	P00375436	F	26-35	15
550066	1006038	P00375436	F	55+	1
550067	1006039	P00371644	F	46-50	0

550068 rows × 12 columns

```
store.head()
```

	User_ID	Product_ID	Gender	Age	Occupation	City
0	1000001	P00069042	F	0-17	10	
1	1000001	P00248942	F	0-17	10	
2	1000001	P00087842	F	0-17	10	
3	1000001	P00085442	F	0-17	10	
4	1000002	P00285442	M	55+	16	

```
store
```

	User_ID	Product_ID	Gender	Age	Occupation
0	1000001	P00069042	F	0-17	10
1	1000001	P00248942	F	0-17	10
2	1000001	P00087842	F	0-17	10
3	1000001	P00085442	F	0-17	10
4	1000002	P00285442	M	55+	16
...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13
550064	1006035	P00375436	F	26-35	1
550065	1006036	P00375436	F	26-35	15
550066	1006038	P00375436	F	55+	1
550067	1006039	P00371644	F	46-50	0

550068 rows × 12 columns

```
store_categoical=store[['Age','Occupation','User_ID','Pro
store_categoical.head()
```

	Age	Occupation	User_ID	Product_ID
0	0-17	10	1000001	P00069042
1	0-17	10	1000001	P00248942
2	0-17	10	1000001	P00087842
3	0-17	10	1000001	P00085442
4	55+	16	1000002	P00285442

```
store.describe()
```

	User_ID	Occupation	Marital_Status	P
count	5.500680e+05	550068.000000	550068.000000	
mean	1.003029e+06	8.076707	0.409653	
std	1.727592e+03	6.522660	0.491770	
min	1.000001e+06	0.000000	0.000000	
25%	1.001516e+06	2.000000	0.000000	
50%	1.003077e+06	7.000000	0.000000	
75%	1.004478e+06	14.000000	1.000000	
max	1.006040e+06	20.000000	1.000000	

```
store.num=store[['User_ID','Occupation']]
store.num.head()
```

	User_ID	Occupation
0	1000001	10
1	1000001	10
2	1000001	10
3	1000001	10
4	1000002	16

```
store.num.describe()
```

	User_ID	Occupation
count	5.500680e+05	550068.000000
mean	1.003029e+06	8.076707
std	1.727592e+03	6.522660
min	1.000001e+06	0.000000
25%	1.001516e+06	2.000000
50%	1.003077e+06	7.000000
75%	1.004478e+06	14.000000
max	1.006040e+06	20.000000

```
store
```

	User_ID	Product_ID	Gender	Age	Occupation
0	1000001	P00069042	F	0-17	10
1	1000001	P00248942	F	0-17	10
2	1000001	P00087842	F	0-17	10
3	1000001	P00085442	F	0-17	10
4	1000002	P00285442	M	55+	16
...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13
550064	1006035	P00375436	F	26-35	1
550065	1006036	P00375436	F	26-35	15
550066	1006038	P00375436	F	55+	1
550067	1006039	P00371644	F	46-50	0

550068 rows × 12 columns

```
store.Gender.value_counts()
```

```
          count
Gender
M        414259
F        135809

dtype: int64
```

```
store.Age.value_counts()
```

```
        count
Age
26-35  219587
36-45  110013
18-25   99660
46-50   45701
51-55   38501
55+     21504
0-17    15102

dtype: int64
```

```
store.Occupation<10
```

```
           Occupation
0               False
1               False
2               False
3               False
4               False
...               ...
550063          False
550064           True
550065          False
550066           True
550067           True
550068 rows × 1 columns

dtype: bool
```

```
males = store[store['Gender'] == 'M']
males
```

	User_ID	Product_ID	Gender	Age	Occupation
4	1000002	P00285442	M	55+	16
5	1000003	P00193542	M	26-35	15
6	1000004	P00184942	M	46-50	7
7	1000004	P00346142	M	46-50	7
8	1000004	P0097242	M	46-50	7
...	...	...	...	...	...
550057	1006023	P00370853	M	26-35	0
550058	1006024	P00372445	M	26-35	12
550060	1006026	P00371644	M	36-45	6
550062	1006032	P00372445	M	46-50	7
550063	1006033	P00372445	M	51-55	13

414259 rows × 12 columns

```
females=store[store['Gender'] == 'F']
females
```

	User_ID	Product_ID	Gender	Age	Occupation
0	1000001	P00069042	F	0-17	10
1	1000001	P00248942	F	0-17	10
2	1000001	P00087842	F	0-17	10
3	1000001	P00085442	F	0-17	10
14	1000006	P00231342	F	51-55	9
...	...	...	...	...	...
550061	1006029	P00372445	F	26-35	1
550064	1006035	P00375436	F	26-35	1
550065	1006036	P00375436	F	26-35	15
550066	1006038	P00375436	F	55+	1
550067	1006039	P00371644	F	46-50	0

135809 rows × 12 columns

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
plt.rcParams['figure.figsize']=10,5
import warnings
warnings.filterwarnings('ignore')
```

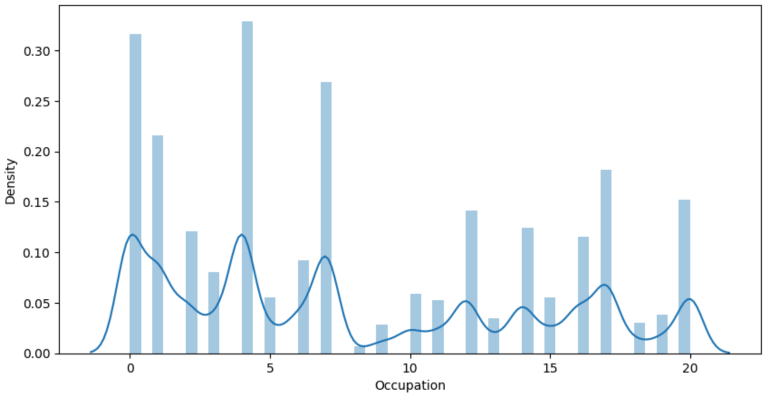
```
store.head()
```

	User_ID	Product_ID	Gender	Age	Occupation	City
0	1000001	P00069042	F	0-17	10	
1	1000001	P00248942	F	0-17	10	
2	1000001	P00087842	F	0-17	10	
3	1000001	P00085442	F	0-17	10	
4	1000002	P00285442	M	55+	16	

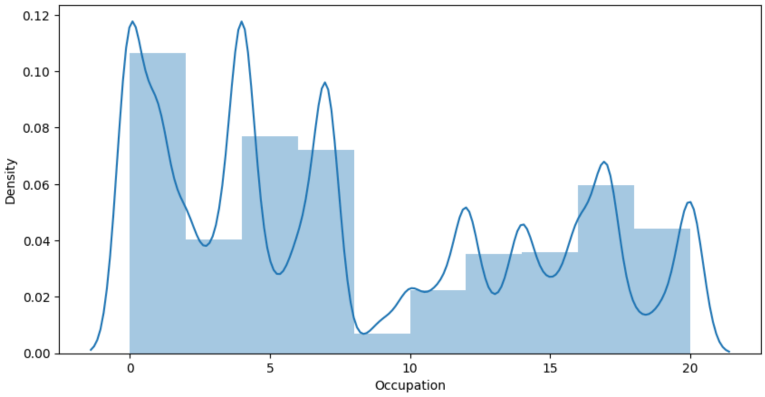
```
store['Age']
```

```
          Age
     0   0-17
     1   0-17
     2   0-17
     3   0-17
     4    55+
   ...    ...
550063  51-55
550064  26-35
550065  26-35
550066    55+
550067  46-50

550068 rows × 1 columns

dtype: object
```

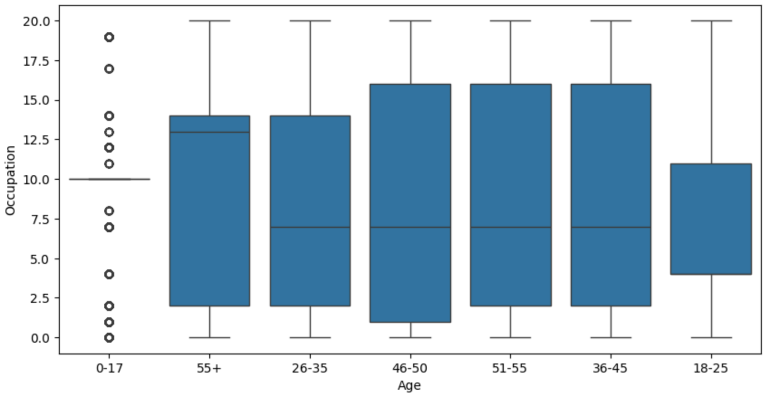
```
store['Occupation']
vis1=sns.distplot(store['Occupation'])
```



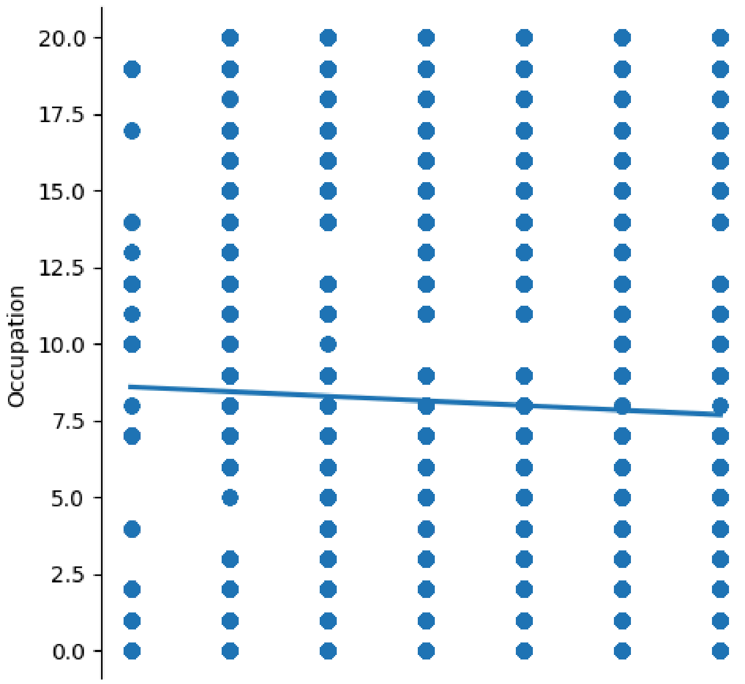
```
vis1=sns.distplot(store['Occupation'],bins=10)
```



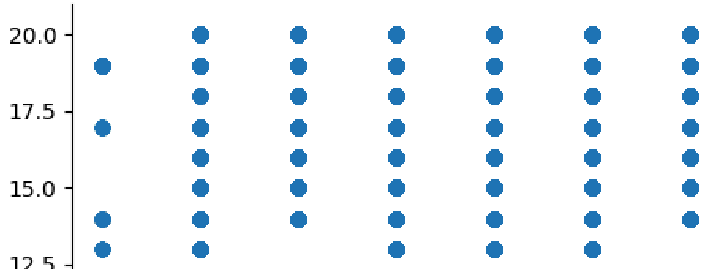
```
vis2=sns.boxplot(data=store,x='Age',y='Occupation')
```



```
store['Age_Encoded'] = pd.factorize(store['Age'])[0]
vis3 = sns.lmplot(data=store, x='Age_Encoded', y='Occupat
```



```
store['Age_Encoded'] = pd.factorize(store['Age'])[0]
vis4 = sns.lmplot(data=store, x='Age_Encoded', y='Occupat
```



Enter a prompt here

0 / 2000

Gemini can make mistakes so double-check responses and use code with caution. Learn more