In [1]:
```python
## Importing basic libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

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

In [2]:
```python
# Laoding the dataset

df = pd.read_csv('https://raw.githubusercontent.com/nanthasnk/Black-Friday-Sales-Prediction/master/Data/BlackFridaySales.csv')
```

In [3]:
```python
# top 5 rows of the df

df.head()
```

Out[3]:

	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
0	1000001	P00069042	F	0-17	10	A	2	0	3	NaN	NaN	8370
1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	14.0	15200
2	1000001	P00087842	F	0-17	10	A	2	0	12	NaN	NaN	1422
3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	NaN	1057
4	1000002	P00285442	M	55+	16	C	4+	0	8	NaN	NaN	7969

In [4]:
```python
# Getting shape of the dataset

df.shape
```

Out[4]:
```
(550068, 12)
```

There are 550068 rows and 12 columns present in the dataset.

In [5]:
```python
# Getting names of the columns

df.columns
```

Out[5]:
```
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')
```

In [6]:
```python
# Count of types of data type
df.dtypes.value_counts()
```

Out[6]:
```
int64      5
object     5
float64    2
dtype: int64
```

In [7]:
```python
# Basic information about dataset

df.info()
```

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

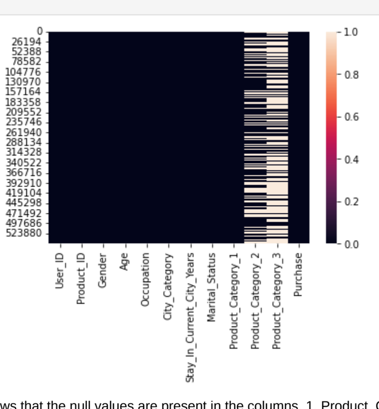
Age should be treated as a numerical column City_Category we can convert this to a numerical column and should look at the frequency of each city category. Gender has two values and should be converted to binary values Product_Category_2 and Product_Category_3 have null values

In [8]:
```python
# Cheking null values

df.isnull().sum()
```

Out[8]:
```
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
```

Product Category 2 and Product Category 3 has nan values.

In [9]:
```python
sns.heatmap(df.isnull())
plt.show()
```



Heatmap shows that the null values are present in the columns. 1. Product_Category_2 2. Product_Category_3

In [10]:
```python
# Total null values in dataset

df.isnull().sum().sum()
```

Out[10]:
```
556885
```

In [11]:
```python
# Null values in percentage

df.isnull().sum()/df.shape[0]*100
```

Out[11]:
```
User_ID                        0.000000
Product_ID                     0.000000
Gender                         0.000000
Age                            0.000000
Occupation                     0.000000
City_Category                  0.000000
Stay_In_Current_City_Years     0.000000
Marital_Status                 0.000000
Product_Category_1             0.000000
Product_Category_2            31.566643
Product_Category_3            69.672659
Purchase                       0.000000
dtype: float64
```

There are 31% of the null values in the Product_Category_2 and 69% null values in the Product_Category_3

In [12]:
```python
# Unique elements in each attributes

df.nunique()
```

Out[12]:
```
User_ID                       5891
Product_ID                    3631
Gender                           2
Age                              7
Occupation                      21
City_Category                    3
Stay_In_Current_City_Years       5
Marital_Status                   2
Product_Category_1              20
Product_Category_2              17
Product_Category_3              15
Purchase                     18105
dtype: int64
```

We can drop User_ID and Product_ID for model prediction as it has more unique values.

In [13]:
```python
# Checking for duplicates

df.duplicated().sum()
```

Out[13]:
```
0
```

In [14]:
```python
# Dropping out unnecessary columns

df.drop(['Product_ID'],inplace=True,axis=1)
```

In [15]:
```python
df.head()
```

Out[15]:

	User_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
0	1000001	F	0-17	10	A	2	0	3	NaN	NaN	8370
1	1000001	F	0-17	10	A	2	0	1	6.0	14.0	15200
2	1000001	F	0-17	10	A	2	0	12	NaN	NaN	1422
3	1000001	F	0-17	10	A	2	0	12	14.0	NaN	1057
4	1000002	M	55+	16	C	4+	0	8	NaN	NaN	7969

In [16]:
```python
# Data Cleaning

df['Stay_In_Current_City_Years'].unique()
```

Out[16]:
```
array(['2', '4+', '3', '1', '0'], dtype=object)
```

In [17]:
```python
def cities(value):
    if '+' in value:
        value=value.replace('+','')
        return int(value)
    else:
        return int(value)
df['Stay_In_Current_City_Years']=df['Stay_In_Current_City_Years'].apply(cities)
```

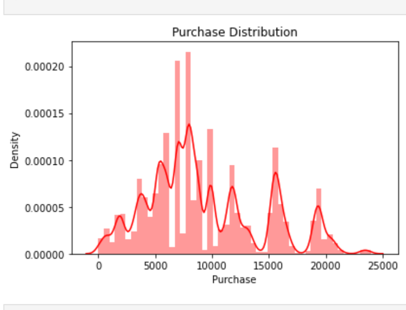
In [18]:
```python
df['Stay_In_Current_City_Years'].unique()
```

Out[18]:
```
array([2, 4, 3, 1, 0], dtype=int64)
```

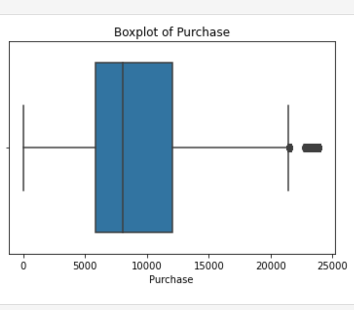
Univariate Analysis

Target Variable Purchase

In [19]:
```python
sns.distplot(df['Purchase'],color='r')
plt.title('Purchase Distribution')
plt.show()
```



In [20]:
```python
sns.boxplot(df['Purchase'])
plt.title('Boxplot of Purchase')
plt.show()
```



In [21]:
```python
df['Purchase'].skew()
```

Out[21]:
```
0.6001400037087128
```

In [22]:
```python
df['Purchase'].kurtosis()
```

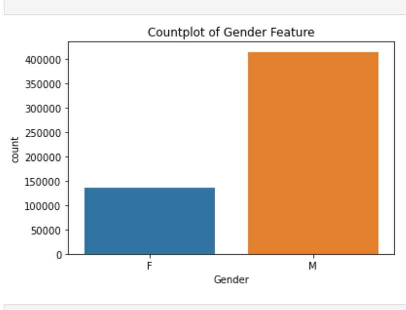
Out[22]:
```
-0.3383775655851702
```

In [23]:
```python
df['Purchase'].describe()
```

Out[23]:
```
count    550068.000000
mean       9263.968713
std        5023.065394
min          12.000000
25%        5823.000000
50%        8047.000000
75%       12054.000000
max       23961.000000
Name: Purchase, dtype: float64
```

Gender

In [24]:
```python
sns.countplot(df['Gender'])
plt.title('Countplot of Gender Feature')
plt.show()
```



In [25]:
```python
df['Gender'].value_counts()
```

Out[25]:
```
M    414259
F    135809
Name: Gender, dtype: int64
```

In [26]:
```python
# getting percentage of the male and female
df['Gender'].value_counts(normalize=True)*100
```

Out[26]:
```
M    75.310507
F    24.689493
Name: Gender, dtype: float64
```

There are more male than females

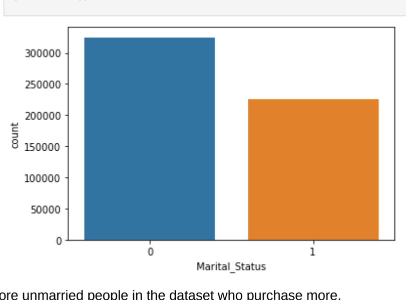
In [27]:
```python
df.groupby('Gender')['Purchase'].mean()
```

Out[27]:
```
Gender
F    8734.565765
M    9437.526040
Name: Purchase, dtype: float64
```

On Average the male gender spends more money on purchase.

Marital Status

In [28]:
```python
sns.countplot(df['Marital_Status'])
plt.show()
```



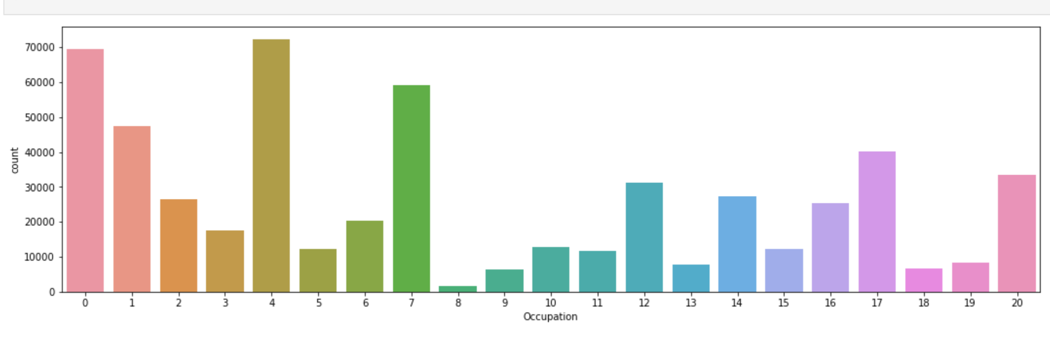
There are more unmarried people in the dataset who purchase more.

In [29]:
```python
df.groupby('Marital_Status')['Purchase'].mean()
```

Out[29]:
```
Marital_Status
0    9265.907619
1    9261.174574
Name: Purchase, dtype: float64
```

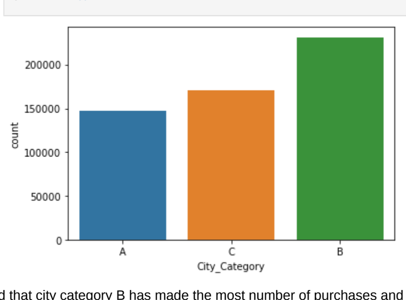
Occupation

In [30]:
```python
plt.figure(figsize=(18,5))
sns.countplot(df['Occupation'])
plt.show()
```



City Category

In [31]:
```python
sns.countplot(df['City_Category'])
plt.show()
```



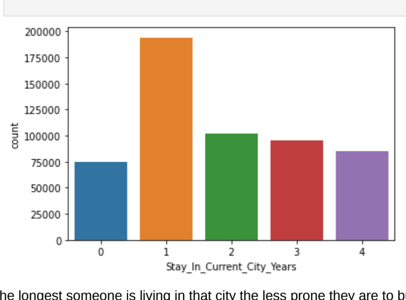
It is observed that city category B has made the most number of purchases and least is of categroy A.

In [32]:
```python
df.groupby('City_Category')['Purchase'].mean()
```

Out[32]:
```
City_Category
A    8911.939216
B    9151.300563
C    9719.920993
Name: Purchase, dtype: float64
```

Stay In Current City Years

In [33]:
```python
sns.countplot(df['Stay_In_Current_City_Years'])
plt.show()
```



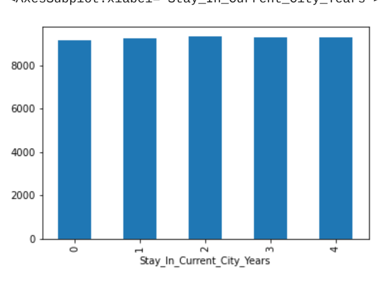
It looks like the longest someone is living in that city the less prone they are to buy new things.

In [34]:
```python
df.groupby('Stay_In_Current_City_Years')['Purchase'].mean()
```

Out[34]:
```
Stay_In_Current_City_Years
0    9180.075123
1    9250.145923
2    9320.429810
3    9286.904119
4    9275.598872
Name: Purchase, dtype: float64
```

In [35]:
```python
df.groupby('Stay_In_Current_City_Years')['Purchase'].mean().plot(kind='bar')
```

Out[35]:
```
<AxesSubplot:xlabel='Stay_In_Current_City_Years'>
```



Age

In [36]:
```python
df['Age'].unique()
```

Out[36]:
```
array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
      dtype=object)
```

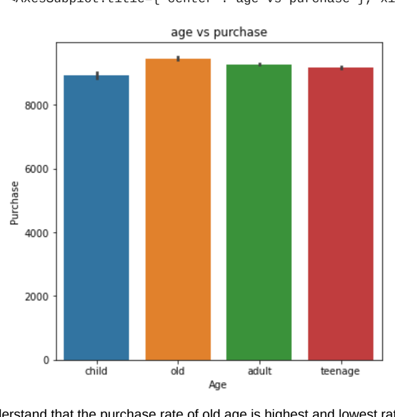
In [37]:
```python
def ages(value):
    if '0-17' in value:
        value=value.replace('0-17','child')
        return str(value)
    elif '26-35' in value:
        value=value.replace('26-35','adult')
        return str(value)
    elif '36-45' in value:
        value=value.replace('36-45','adult')
        return str(value)
    elif '18-25' in value:
        value=value.replace('18-25','teenage')
        return str(value)
    elif '46-50' in value:
        value=value.replace('46-50','adult')
        return str(value)
    elif '51-55' in value:
        value=value.replace('51-55','old')
        return str(value)
    else:
        value=value.replace('55+','old')
        return str(value)
df['Age']=df['Age'].apply(ages)
```

In [44]:
```python
df['Age'].unique()
```

Out[44]:
```
array(['child', 'old', 'adult', 'teenage'], dtype=object)
```

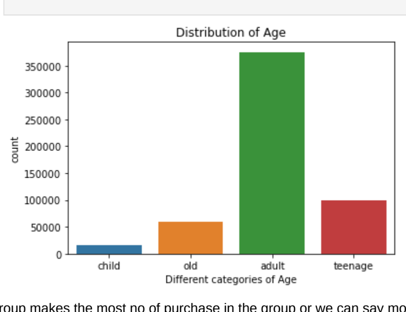
In [46]:
```python
plt.figure(figsize=(6,6))
plt.title('age vs purchase')
sns.barplot(x='Age', y='Purchase',data=df)
```

Out[46]:
```
<AxesSubplot:title={'center':'age vs purchase'}, xlabel='Age', ylabel='Purchase'>
```



Here we understand that the purchase rate of old age is highest and lowest rate is of child.

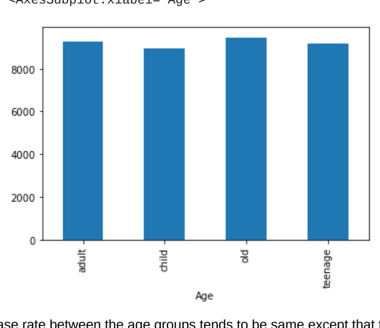
In [38]:
```python
sns.countplot(df['Age'])
plt.title('Distribution of Age')
plt.xlabel('Different categories of Age')
plt.show()
```



Age 26-35 group makes the most no of purchase in the group or we can say mostly adults visit the store.

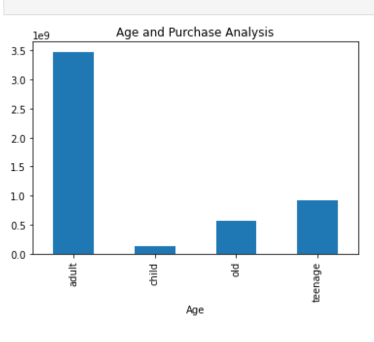
In [39]:
```python
df.groupby('Age')['Purchase'].mean().plot(kind='bar')
```

Out[39]:
```
<AxesSubplot:xlabel='Age'>
```



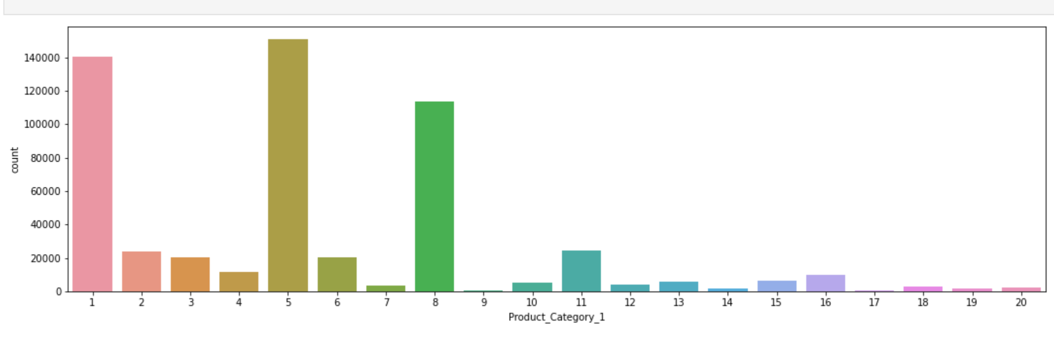
Mean purchase rate between the age groups tends to be same except that the 51-55 age group has a little higher average purchase amount.

In [40]:
```python
df.groupby('Age')['Purchase'].sum().plot(kind='bar')
plt.title('Age and Purchase Analysis')
plt.show()
```



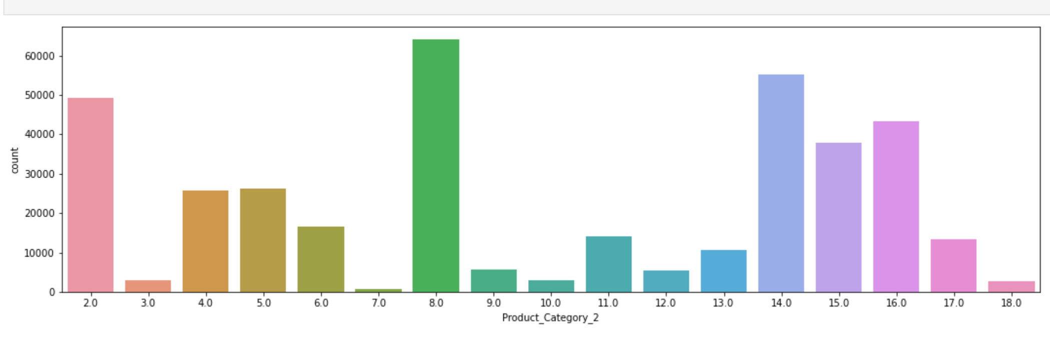
Product_Category_1

In [41]:
```python
plt.figure(figsize=(18,5))
sns.countplot(df['Product_Category_1'])
plt.show()
```



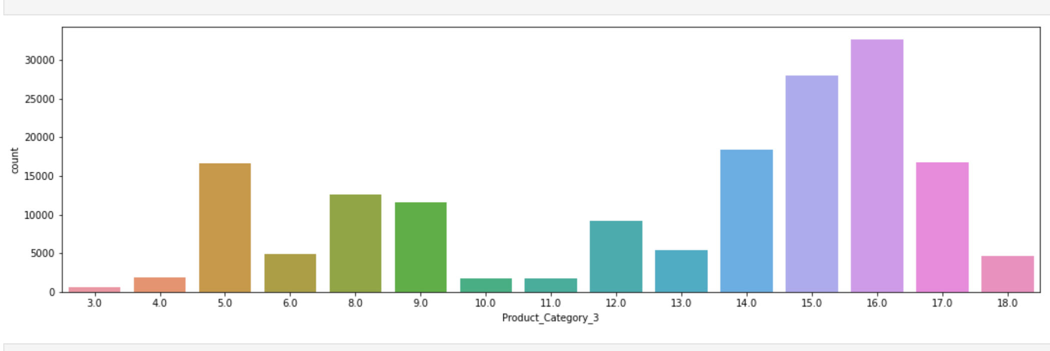
Product Category 2

In [42]:
```python
plt.figure(figsize=(18,5))
sns.countplot(df['Product_Category_2'])
plt.show()
```



Product Category 3

In [43]:
```python
plt.figure(figsize=(18,5))
sns.countplot(df['Product_Category_3'])
plt.show()
```

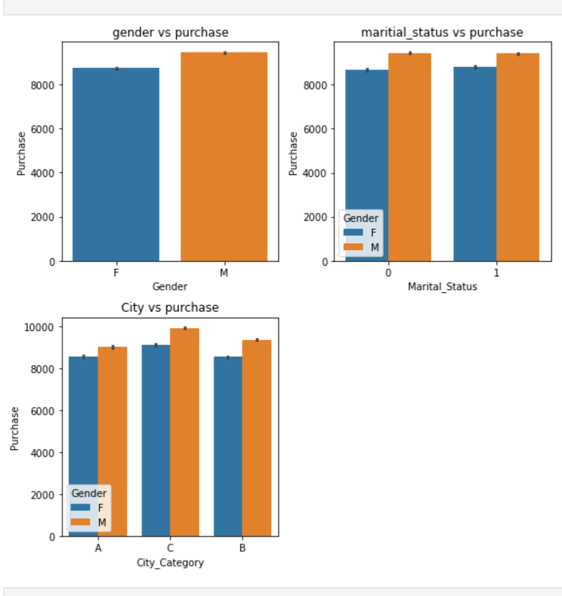


In [50]:
```python
df.columns
```

Out[50]:
```
Index(['User_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')
```

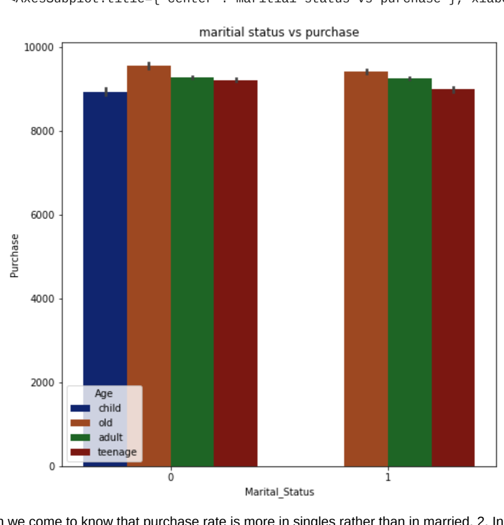
Multivariate Analysis

In [51]:
```python
plt.figure(figsize=(8,8))
plt.subplot(2,2,1)
plt.title('gender vs purchase')
sns.barplot(x='Gender',y='Purchase',data=df)

plt.subplot(2,2,2)
plt.title('maritial_status vs purchase')
sns.barplot(x='Marital_Status',y='Purchase',hue='Gender',data=df)

plt.subplot(2,2,3)
plt.title('City vs purchase')
sns.barplot(x='City_Category',y='Purchase',hue='Gender',data=df)

plt.tight_layout()
plt.show()
```



In [54]:
```python
plt.figure(figsize=(8,8))
plt.title('maritial status vs purchase')
sns.barplot(x='Marital_Status',y='Purchase',data=df,palette='dark',hue='Age')
```

Out[54]:
```
<AxesSubplot:title={'center':'maritial status vs purchase'}, xlabel='Marital_Status', ylabel='Purchase'>
```



1. This graph we come to know that purchase rate is more in singles rather than in married. 2. In singles purchase rate of old age is higher where as in married purchase rate of old is higher.