In [1]:

```
import numpy as np # linear algebra
import matplotlib as mlp
import matplotlib.pyplot as plt
import statistics
import seaborn as sns
import sys
import types
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your
 credentials.
# You might want to remove those credentials before you share your notebook.
client_3c45f252d9e64c45adc8afbaa5384d66 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='lbRkHZDcGyGWE5kxPmMoq1BQgcBOh0TEjOZOk9AP8b9J',
    ibm_auth_endpoint="https://iam.ng.bluemix.net/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3-api.us-geo.objectstorage.service.networklayer.com')

body = client_3c45f252d9e64c45adc8afbaa5384d66.get_object(Bucket='capstone-donotdelete-
pr-mg3sh8gg3rlh1f',Key='BlackFriday.csv')['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

data = pd.read_csv(body)
data.head()
```

Out[1]:

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

In [2]:

```
# checking median of data (another technique of check the central tendency of the data)
# We take median and mean of numeric variables
# we take Mode of Binary, Catagorical(Nominal,Ordinal) attributes

print("mean:Purchases ", statistics.mean(data['Purchase']))
print("median:Purchases ", statistics.median(data['Purchase']))
print("max: Purchases ", max(data['Purchase']))
print("min: Purchases ", min(data['Purchase']))
print('Mode:Age ',statistics.mode(data['Age']))   #people of age 26-35 buy the most
print('Mode:gender',statistics.mode(data['Gender'])) #male buy more products than femal
e
print('Mode:Occupation',statistics.mode(data['Occupation'])) #Oppuation having id 4 buy
 the most product (They may be doctors ,enginners bussnissMan etc)
print('Mode:Martial Status',statistics.mode(data['Marital_Status'])) # married people b
uy more than un married peoples
```

```
mean:Purchases  9333.859852635065
median:Purchases  8062
max: Purchases  23961
min: Purchases  185
Mode:Age  26-35
Mode:gender M
Mode:Occupation 4
Mode:Martial Status 0
```

In [3]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 537577 entries, 0 to 537576
Data columns (total 12 columns):
User_ID                     537577 non-null int64
Product_ID                  537577 non-null object
Gender                      537577 non-null object
Age                         537577 non-null object
Occupation                  537577 non-null int64
City_Category               537577 non-null object
Stay_In_Current_City_Years  537577 non-null object
Marital_Status              537577 non-null int64
Product_Category_1          537577 non-null int64
Product_Category_2          370591 non-null float64
Product_Category_3          164278 non-null float64
Purchase                    537577 non-null int64
dtypes: float64(2), int64(5), object(5)
memory usage: 49.2+ MB
```

In [4]:

```
## checking which columns have null values.
data.isna().any()
```

Out[4]:

```
User_ID                     False
Product_ID                  False
Gender                      False
Age                         False
Occupation                  False
City_Category               False
Stay_In_Current_City_Years  False
Marital_Status              False
Product_Category_1          False
Product_Category_2           True
Product_Category_3           True
Purchase                    False
dtype: bool
```

In [5]:

```
## assigning value zero for the NaN cases
data.fillna(value=0,inplace=True)
```

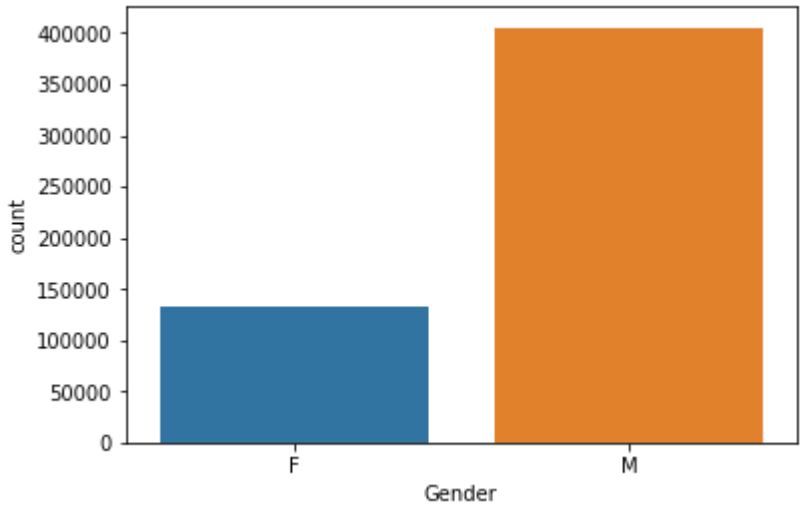
In [10]:

```
sns.countplot(data['Gender'])
```

```
/usr/local/src/conda3_runtime/home/envs/DSX-Python35-Spark/lib/python3.5/s
ite-packages/seaborn/categorical.py:1460: FutureWarning: remove_na is depr
ecated and is a private function. Do not use.
  stat_data = remove_na(group_data)
```

Out[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6ef81c4978>
```



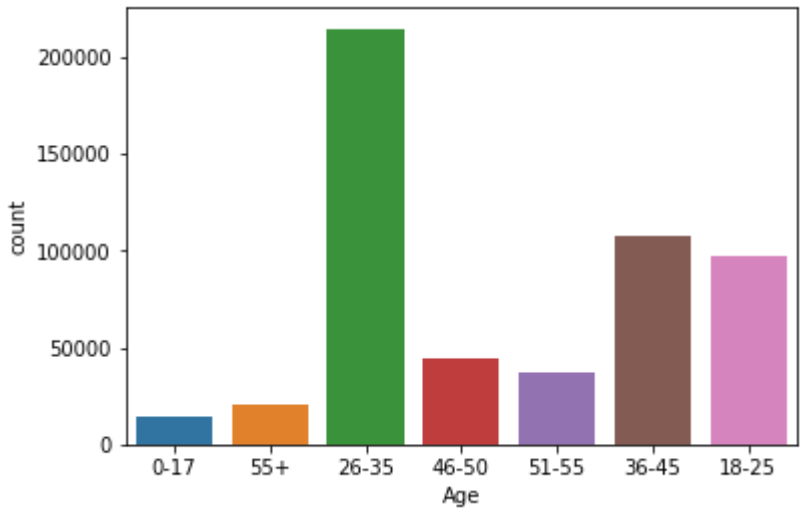
In [11]:

```
sns.countplot(data['Age'])
```

```
/usr/local/src/conda3_runtime/home/envs/DSX-Python35-Spark/lib/python3.5/s
ite-packages/seaborn/categorical.py:1460: FutureWarning: remove_na is depr
ecated and is a private function. Do not use.
  stat_data = remove_na(group_data)
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6ef8143b38>
```



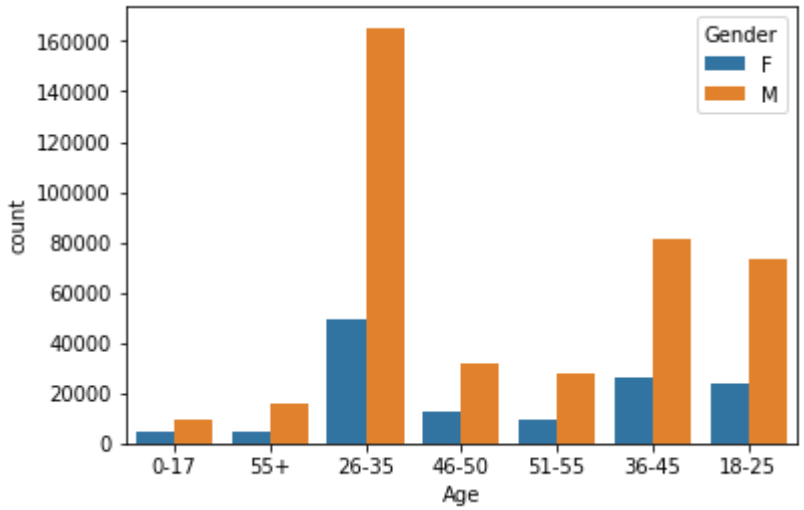
In [12]:

```
sns.countplot(data['Age'],hue=data['Gender'])
```

```
/usr/local/src/conda3_runtime/home/envs/DSX-Python35-Spark/lib/python3.5/s
ite-packages/seaborn/categorical.py:1508: FutureWarning: remove_na is depr
ecated and is a private function. Do not use.
  stat_data = remove_na(group_data[hue_mask])
```

Out[12]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6ef8151898>
```



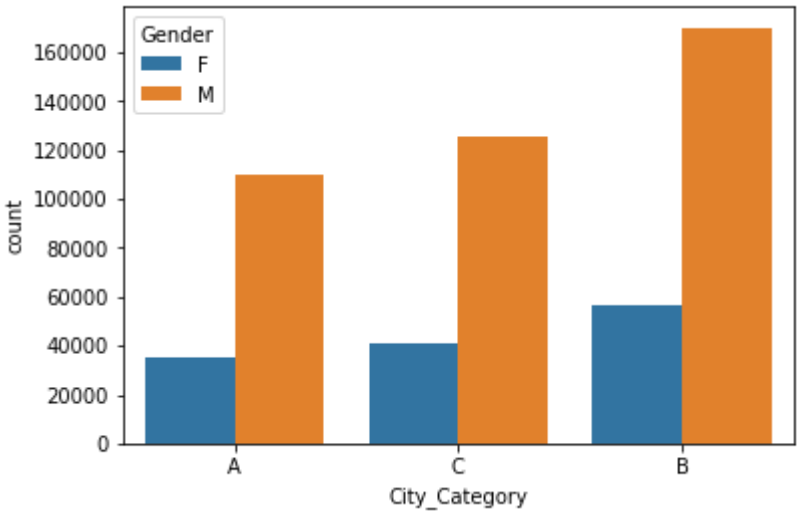
In [13]:

```
sns.countplot(data['City_Category'],hue=data['Gender'])
```

```
/usr/local/src/conda3_runtime/home/envs/DSX-Python35-Spark/lib/python3.5/s
ite-packages/seaborn/categorical.py:1508: FutureWarning: remove_na is depr
ecated and is a private function. Do not use.
  stat_data = remove_na(group_data[hue_mask])
```

Out[13]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6ef80c5b00>
```



In [8]:

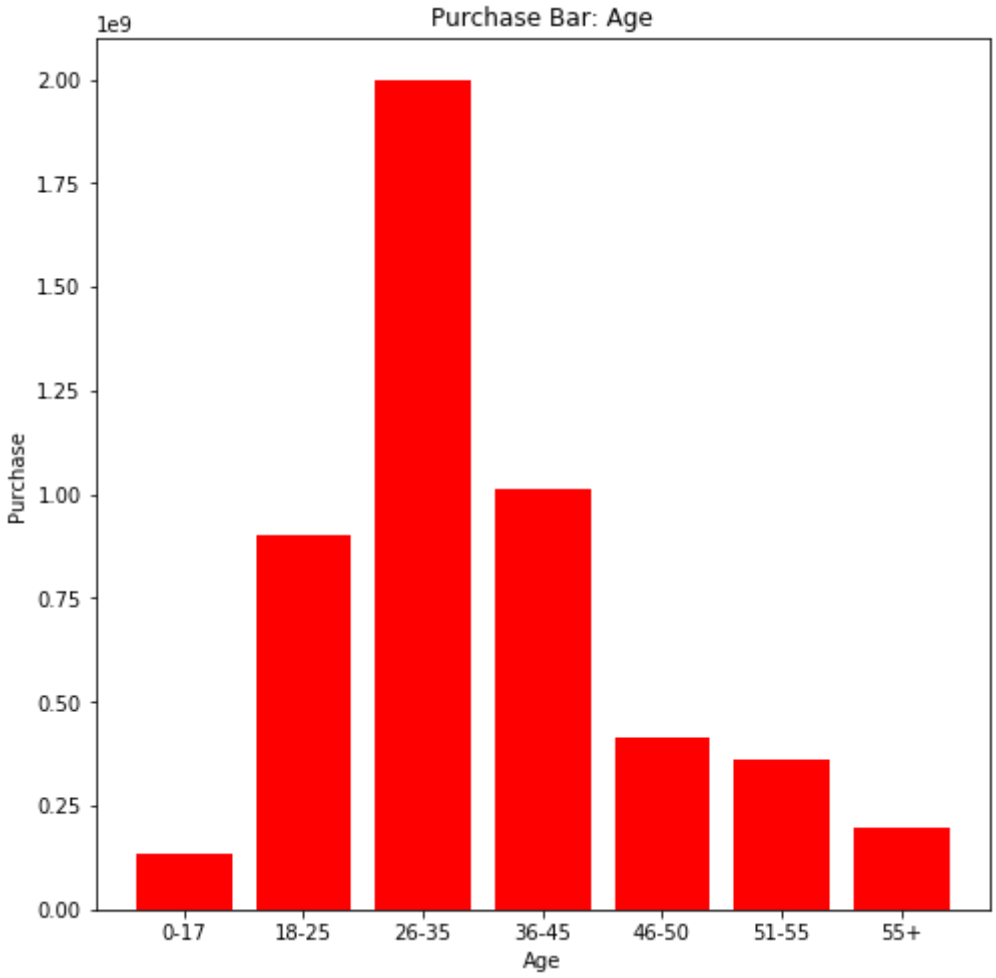
```
# use dictionary because age features is string
ageDict = {}
for purch, userAge in zip( data['Purchase'], data['Age']):
    if ageDict.get(userAge, -1) != -1:
        ageDict[userAge] +=  purch
    else:
        ageDict[userAge] = purch

# total purchase by age but no sort
print( ageDict)
print('---'*50)
# we sorted dictionary for more beatiful graph
from collections import OrderedDict
# key -> x[0]:sort dict keys, x[1]:sort dict values
age_sorted = OrderedDict( sorted( ageDict.items(), key=lambda x: x[0]))
print( age_sorted)
#print('---'*50)
#age_sorted1 = OrderedDict( sorted( ageDict.items(), key=lambda x: x[1]))
#print( age_sorted1)
```

```
{'51-55': 361908356, '46-50': 413418223, '55+': 197614842, '26-35': 199974
9106, '36-45': 1010649565, '18-25': 901669280, '0-17': 132659006}
------------------------------------------------------------------------
------------------------------------------------------------------------
--
OrderedDict([('0-17', 132659006), ('18-25', 901669280), ('26-35', 19997491
06), ('36-45', 1010649565), ('46-50', 413418223), ('51-55', 361908356),
('55+', 197614842)])
```

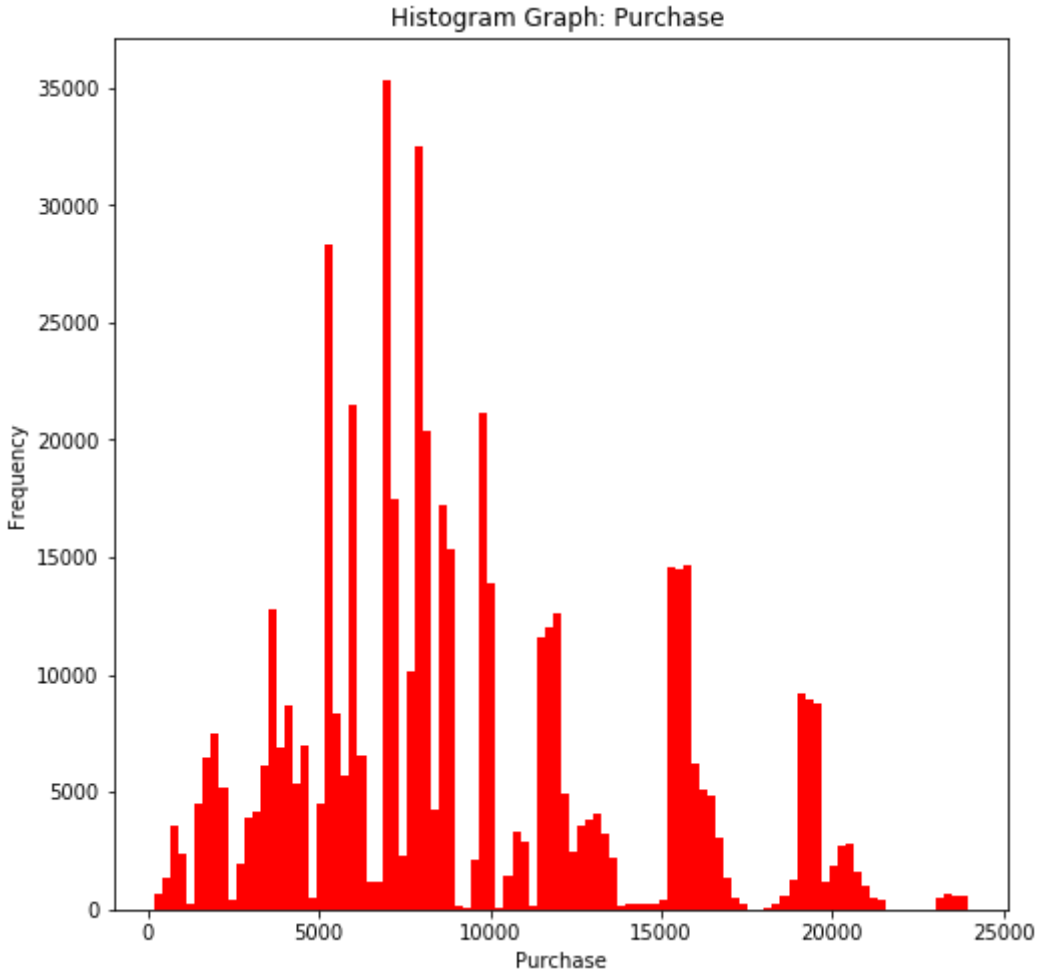
In [16]:

```
# draw figure
fig = plt.figure(figsize=(8,8))
# add graph
ax = fig.add_subplot(111)
# draw graph
ax.bar( age_sorted.keys(), age_sorted.values(), color='r')
plt.title('Purchase Bar: Age')
plt.xlabel('Age')
plt.ylabel('Purchase')
plt.show()
```



In [6]:

```
# plot Purchase
# kind = kine, color = color, bins = number of bar in figure, figsize = figure size
data.Purchase.plot(kind = 'hist', color = "r", bins = 100, figsize=(8, 8))
plt.title('Histogram Graph: Purchase ')
plt.xlabel('Purchase')
plt.show()
```



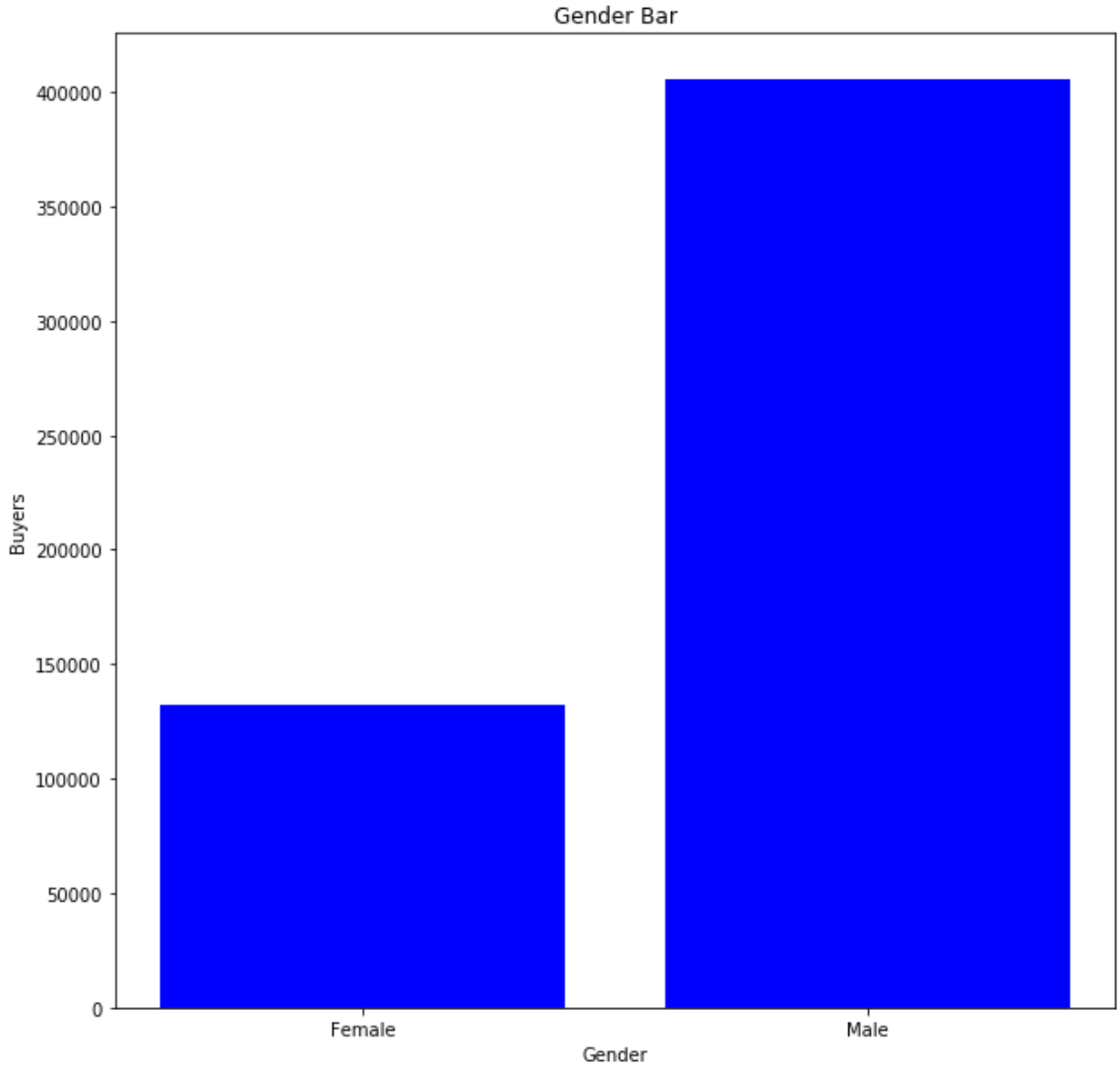
In [17]:

```
# get male and female user
male_user = data[ data[ 'Gender'] == 'M'].count()[0]
female_user = data[ data[ 'Gender'] == 'F'].count()[0]
print('Male user: {0}'.format(male_user))
print('Female user: {0}'.format(female_user))
```

```
Male user: 405380
Female user: 132197
```

In [19]:

```
# draw figure
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(111)
ax.bar( ['Male', 'Female'], [male_user, female_user], color='b')
plt.title('Gender Bar')
plt.xlabel('Gender')
plt.ylabel('Buyers')
plt.show()
```



In [20]:

```
# total purchase by gender
male_purch = data[ data['Gender']=='M']['Purchase'].sum()
female_purch = data[ data['Gender']=='F']['Purchase'].sum()
print('Male user: {0}'.format(male_purch))
print('Female user: {0}'.format(female_purch))
```

```
Male user: 3853044357
Female user: 1164624021
```

In [22]:

```
# draw figure
fig = plt.figure(figsize=(15,10))

# graph total purchase by gender
ax1 = fig.add_subplot(121)
ax1.bar( ['Male', 'Female'], [male_purch, female_purch], color='b')
plt.title('Total Purchase by Gender')
plt.ylabel('Purchase')

# graph total purchase by gender
ax2 = fig.add_subplot(122)
ax2.bar( ['Male', 'Female'], [male_purch/male_user, female_purch/female_user], color=
'r')
plt.title('Average Purchase by Gender')
plt.ylabel('Purchase')

plt.show()
```