In [1]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version us
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

In [2]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from IPython.display import Image,display
%matplotlib inline
import missingno
```

In [3]:
```
train=pd.read_csv('data.csv')
test=pd.read_csv('test_data.csv')
```

In [4]:
```
train.head()
```

Out[4]:

	Unnamed: 0	CustomerID	Genre	Age	Annual Income (k$)	Class
0	0	10	Female	30	19	3
1	1	20	Female	35	23	3
2	2	200	Male	30	137	3
3	3	153	Female	44	78	1
4	4	4	Female	23	16	3

DROPPING UNNAMED

In [5]:
```
train.drop('Unnamed: 0',1,inplace=True)
```

In [6]:
```
train.head()
```

Out[6]:

	CustomerID	Genre	Age	Annual Income (k$)	Class
0	10	Female	30	19	3
1	20	Female	35	23	3
2	200	Male	30	137	3
3	153	Female	44	78	1
4	4	Female	23	16	3

In [7]:
```
test.head()
```

Out[7]:

	Unnamed: 0	CustomerID	Genre	Age	Annual Income (k$)
0	0	1	Male	19	15
1	1	147	Male	48	77
2	2	159	Male	34	78
3	3	177	Male	58	88
4	4	198	Male	32	126

In [8]:
```
test.drop('Unnamed: 0',1,inplace=True)
```

In [9]:
```
test.head()
```

Out[9]:

	CustomerID	Genre	Age	Annual Income (k$)
0	1	Male	19	15

	CustomerID	Genre	Age	Annual Income (k$)
1	147	Male	48	77
2	159	Male	34	78
3	177	Male	58	88
4	198	Male	32	126

In [10]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 160 entries, 0 to 159
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   CustomerID          160 non-null    int64 
 1   Genre               160 non-null    object
 2   Age                 160 non-null    int64 
 3   Annual Income (k$)  160 non-null    int64 
 4   Class               160 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 6.4+ KB
```

In [11]:
```
train.describe()
```

Out[11]:

	CustomerID	Age	Annual Income (k$)	Class
count	160.000000	160.000000	160.000000	160.000000
mean	98.675000	39.112500	59.962500	2.031250
std	59.264735	14.094911	27.006612	0.747506
min	2.000000	18.000000	15.000000	1.000000
25%	45.750000	29.000000	39.000000	1.000000
50%	98.500000	36.000000	60.500000	2.000000
75%	150.250000	49.000000	78.000000	3.000000
max	200.000000	70.000000	137.000000	3.000000

In [12]:
```
#dataframe describing our data
train_data_dict=pd.DataFrame(train.dtypes,columns=['dtype'])
train_data_dict['Missing_val']=train.isnull().sum()
train_data_dict['Unique_val']=train.nunique()
train_data_dict['Count']=train.count()
train_data_dict
```

Out[12]:

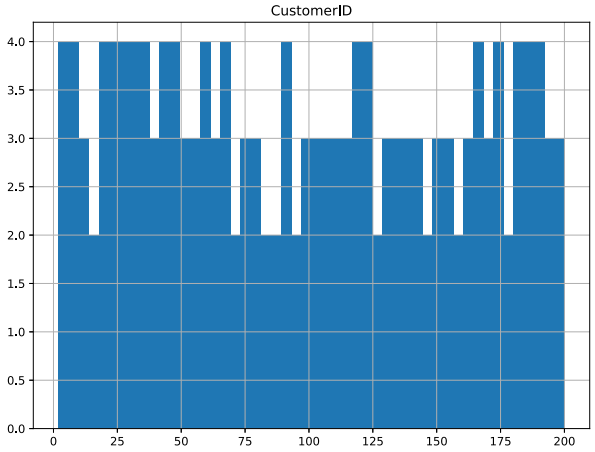
	dtype	Missing_val	Unique_val	Count
CustomerID	int64	0	160	160
Genre	object	0	2	160
Age	int64	0	48	160
Annual Income (k$)	int64	0	64	160
Class	int64	0	3	160

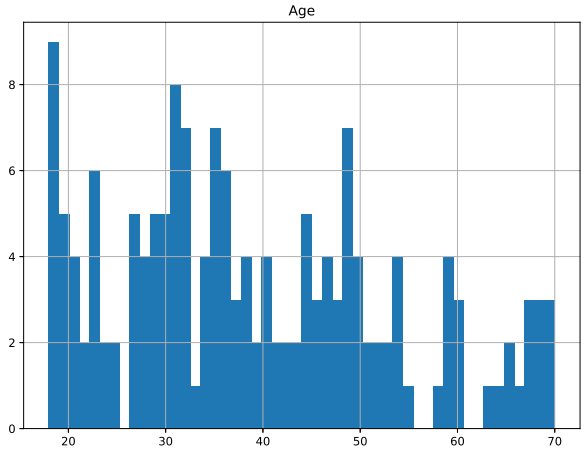
In [13]:
```
x=train.Genre.value_counts(normalize=True)
x*100
```

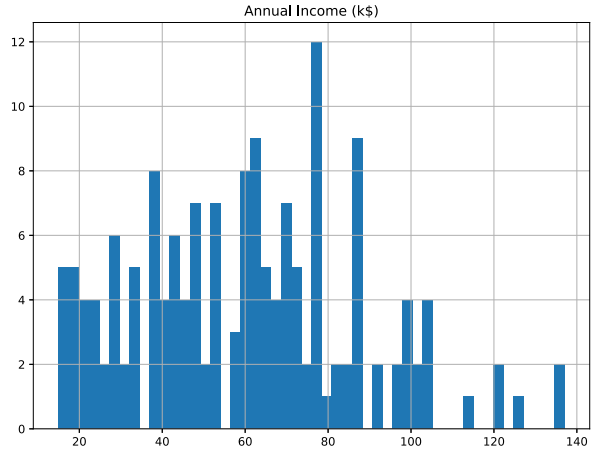
```
Female    60.0
Male      40.0
Name: Genre, dtype: float64
```

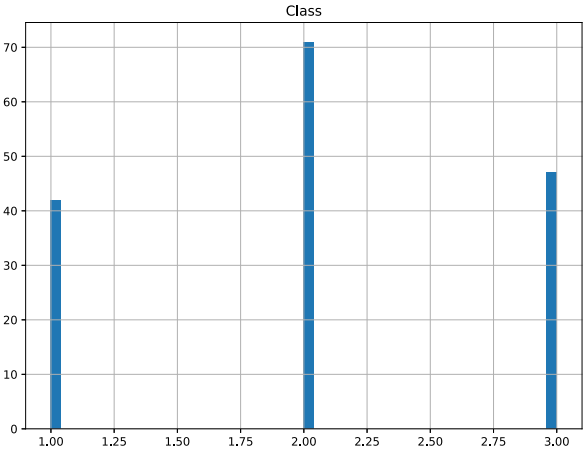
In [14]:
```
train.hist(bins=50,figsize=(20,15))
```

```
array([[<AxesSubplot:title={'center':'CustomerID'}>,
        <AxesSubplot:title={'center':'Age'}>],
       [<AxesSubplot:title={'center':'Annual Income (k$)'}>,
        <AxesSubplot:title={'center':'Class'}>]], dtype=object)
```





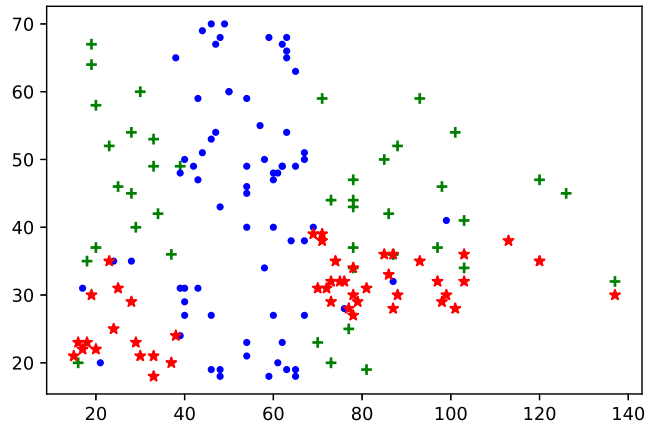




In [15]:
```
df1=train[train.Class==1]
df2=train[train.Class==2]
df3=train[train.Class==3]
```

In [16]:
```
plt.scatter(df1['Annual Income (k$)'], df1['Age'],color="green",marker='+')
plt.scatter(df2['Annual Income (k$)'], df2['Age'],color="blue",marker='.')
plt.scatter(df3['Annual Income (k$)'], df3['Age'],color="red",marker='*')
```

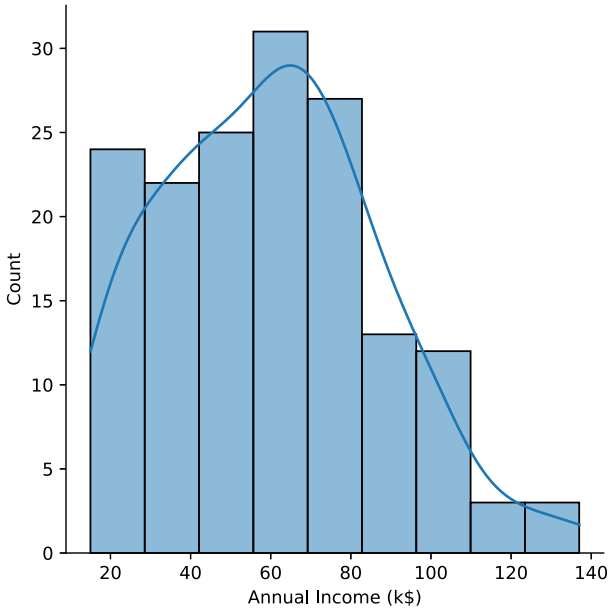
Out[16]:
```
<matplotlib.collections.PathCollection at 0x7f8e1ae0e450>
```

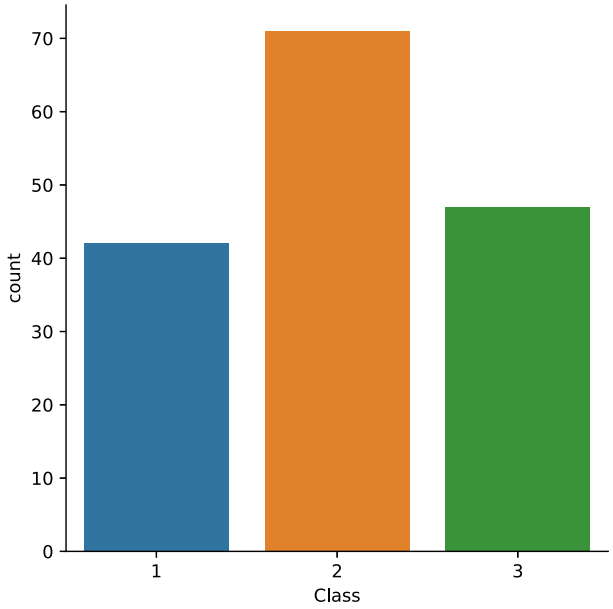


In [17]:
```
sns.displot(train['Annual Income (k$)'].dropna(),kde=True)
sns.displot(train['Age'].dropna(),kde=True)
sns.countplot('Class',data=train)
```

```
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass t
he following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing o
ther arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[17]:
```
<AxesSubplot:xlabel='Class', ylabel='count'>
```





Checking if Age and Annual Income have zero values

In [18]:
```
print((train['Annual Income (k$)']==0).sum())
```

0

In [19]:
```
print((train['Age']==0).sum())
```

0

Dealing with train categorical data

In [20]:
```
categorical_df=train.Genre
categorical=pd.get_dummies(categorical_df,drop_first=True)
categorical.head()
```

Out[20]:

	Male
0	0
1	0
2	1
3	0
4	0

In [21]:
```
train.drop(['CustomerID','Genre'],1,inplace=True)
```

```
train.head()
```

Out[21]:

	Age	Annual Income (k$)	Class
0	30	19	3
1	35	23	3
2	30	137	3
3	44	78	1
4	23	16	3

In [22]:

```
train_final=pd.concat([train,categorical],1)
train_final.head()
```

Out[22]:

	Age	Annual Income (k$)	Class	Male
0	30	19	3	0
1	35	23	3	0
2	30	137	3	1
3	44	78	1	0
4	23	16	3	0

Dealing with test categorical data

In [23]:

```
categorical_test=test.Genre
cate=pd.get_dummies(categorical_test,drop_first=True)
cate.head()
```

Out[23]:

	Male
0	1
1	1
2	1
3	1
4	1

In [24]:

```
test.drop(['CustomerID','Genre'],1,inplace=True)
test.head()
```

Out[24]:

	Age	Annual Income (k$)
0	19	15
1	48	77
2	34	78
3	58	88
4	32	126

In [25]:

```
test_final=pd.concat([test,cate],1)
test_final.head()
```

Out[25]:

	Age	Annual Income (k$)	Male
0	19	15	1
1	48	77	1
2	34	78	1
3	58	88	1
4	32	126	1

In [26]:

```
X_train=train_final.drop('Class',1)
y_train=train_final.Class
```

Importing random forest classifier which doesnt need feature scalling

In [27]:

```
from sklearn.ensemble import RandomForestClassifier
model=RandomForestClassifier(n_estimators=50)
model.fit(X_train,y_train)
```

Out[27]:

```
RandomForestClassifier(n_estimators=50)
```

In [28]:
```
Y_pred_test = model.predict(test_final)
Y_pred_test
```

Out[28]:
```
array([2, 1, 3, 1, 1, 2, 2, 2, 2, 3, 2, 2, 1, 2, 3, 3, 1, 3, 2, 2, 3, 3,
       1, 2, 2, 1, 1, 3, 2, 2, 2, 1, 1, 2, 3, 3, 2, 1, 2, 1])
```

Competition Accuracy Score

In [29]:
```
from sklearn.metrics import accuracy_score

test_class = pd.read_csv('test_class.csv')

print(accuracy_score(Y_pred_test, test_class.iloc[:, 1]))
```

```
0.725
```