In [55]:

```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
```

In [56]:

```
#Importing libraries and Datasets!
df1=pd.read_csv('AdvWorksCusts.csv')
df2=pd.read_csv('AW_BikeBuyer.csv')
# Reading test file to predict
test=pd.read_csv('AW_test.csv')
```

In [57]:

```
#Creating a master dataset from all 3 datasets!
df=pd.merge(df1, df2, on='CustomerID')
```

In [58]:

```
#Checking the dataset columns!
df.columns
```

Out[58]:

```
Index(['CustomerID', 'Title', 'FirstName', 'MiddleName', 'LastName',
'Suffix',
       'AddressLine1', 'AddressLine2', 'City', 'StateProvinceName',
       'CountryRegionName', 'PostalCode', 'PhoneNumber', 'BirthDate',
       'Education', 'Occupation', 'Gender', 'MaritalStatus', 'HomeOwne
rFlag',
       'NumberCarsOwned', 'NumberChildrenAtHome', 'TotalChildren',
       'YearlyIncome', 'BikeBuyer'],
      dtype='object')
```

In [59]:

```
#Checking the datatypes of the columns
df.dtypes
```

Out[59]:

```
CustomerID                int64
Title                    object
FirstName                object
MiddleName               object
LastName                 object
Suffix                   object
AddressLine1             object
AddressLine2             object
City                     object
StateProvinceName        object
CountryRegionName        object
PostalCode               object
PhoneNumber              object
BirthDate                object
Education                object
Occupation               object
Gender                   object
MaritalStatus            object
HomeOwnerFlag             int64
NumberCarsOwned           int64
NumberChildrenAtHome      int64
TotalChildren             int64
YearlyIncome              int64
BikeBuyer                 int64
dtype: object
```

In [60]:

```
#Lets see how our data looks like
df.head()
```

Out[60]:

	CustomerID	Title	FirstName	MiddleName	LastName	Suffix	AddressLine1	AddressLine2	
0	11000	NaN	Jon	V	Yang	NaN	3761 N. 14th St	NaN	F
1	11001	NaN	Eugene	L	Huang	NaN	2243 W St.	NaN	
2	11002	NaN	Ruben	NaN	Torres	NaN	5844 Linden Land	NaN	
3	11003	NaN	Christy	NaN	Zhu	NaN	1825 Village Pl.	NaN	
4	11004	NaN	Elizabeth	NaN	Johnson	NaN	7553 Harness Circle	NaN	

5 rows × 24 columns

Now time to select features to make this model. Let's study it one by one!

CustomerID : This is compulsory as all data will be indexed to this unique ID.

Title,**FirstName**,**MiddleName**,**LastName**,**Suffix**: All these doesn't determine any significant thing as AveMonthSpend and BikeBuyer doesn't depend on Name!

AddressLine1,**AddressLine2** : I'm avoiding these to prevent the overfitting the data as we already have 2-3 more location parameters!

City,**StateProvinceName**,**CountryRegionName**: All these will play major role as they gives us the location and economy varies economy to economy.

PostalCode: Leaving this behind since we already have lots of location variable.

PhoneNumber : Phone number and our target variable doesn't have any relation!

BirthDate : It is good idea to extract the Age of customer from this column.

Education ,**Occupation** : They will be key factor in predicitng the total capital someone have! So we must select them.

Gender, **MaritalStatus** : Both plays important role in determining the present condition of someone.It is not directly linked but still very important factor. Females are tends to buy less bikes. Same goes for the Married People if we compare them to the Unmarried Young adults.

NumberCarsOwned,**TotalChildren**,**NumberChildrenAtHome** : All are key financial factors and must be considered!

In [61]:

```
#Creating a new feature from Date of Birth column!!
df['Age'] = pd.DatetimeIndex(df['BirthDate']).year
df['Age']=2019-df.Age
test['Age'] = pd.DatetimeIndex(test['BirthDate']).year
test['Age']=2019-test.Age
```

EDUCATION REPRESENTS PLENTY OF THINGS

Education has always been the trump card for the middle-class in their aspirations to become wealthy. Every year we see thousands of students enter college or university for higher education in the hopes of getting high-paying jobs and becoming rich as soon as possible. It has always been believed that education leads to financial prosperity and success. Education is an investment in human capital and it is believed to boost efficiency and productivity.The more specialized you get, the higher your earnings become. The type of degree matters too. Empirically, a professional degree commands much greater incomes than normal graduate degrees. A four-year degree can be the key to financial success in the long run.

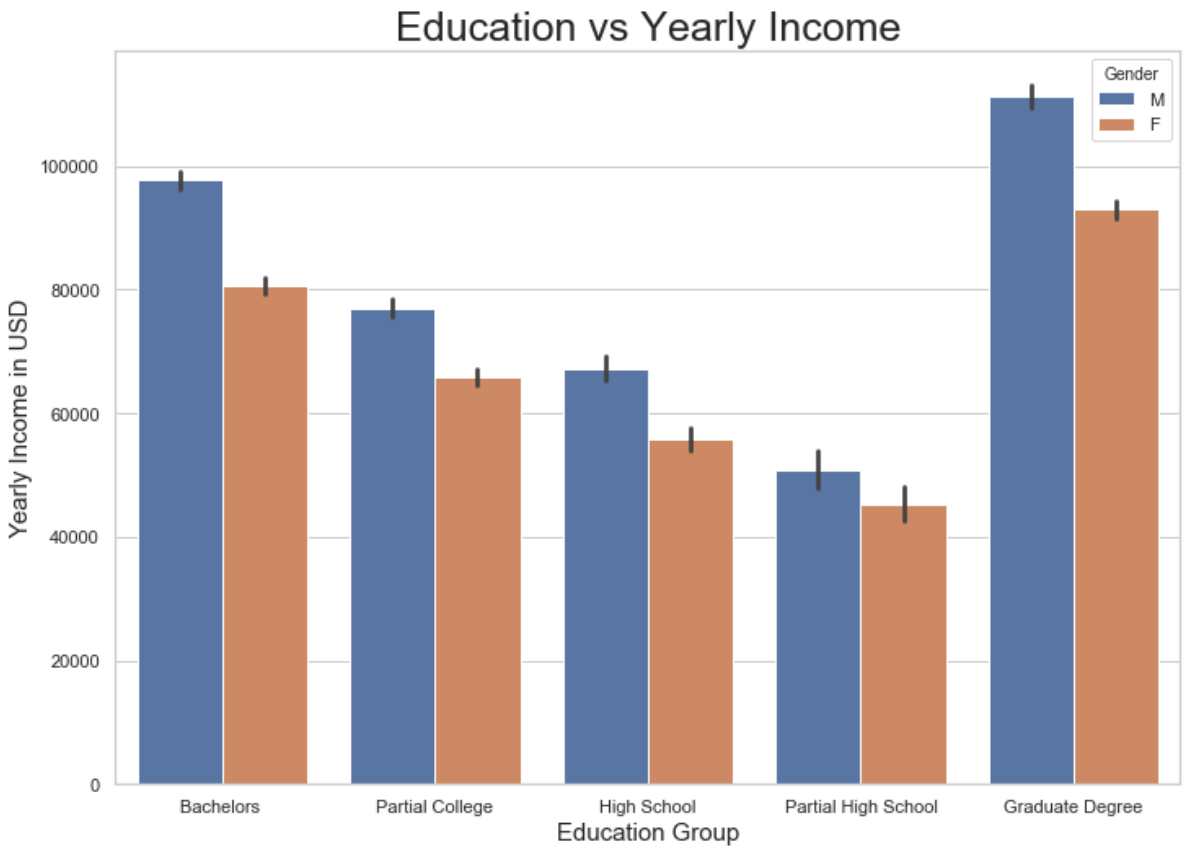
When we talk about the connection between education and wealth, the focus should be on income more than wealth.This plot is definitely showing us the high education gives us more wealth and ability to buy new things.

In [62]:

```
rcParams['figure.figsize'] = 11.7,8.27
sns.set(style="whitegrid")
ax = sns.barplot(x="Education", y="YearlyIncome",hue='Gender' ,data=df)
ax.set_title('Education vs Yearly Income',size=25)
ax.set_ylabel('Yearly Income in USD',size=15)
ax.set_xlabel('Education Group',size=15)
```

Out[62]:

```
Text(0.5, 0, 'Education Group')
```



BACHELORS BUYS THE MOST BIKES

Bachelors definitely have the most chances have the most chances of buying bikes. There can be many reasons behind this. One of the top reason can be ,he/she is in better position than any other group except a graduate. This is definitely following the our first plot motto,i.e, Higher Education can bring more wealth.This is pretty much clear from the bar chart mentioned below.

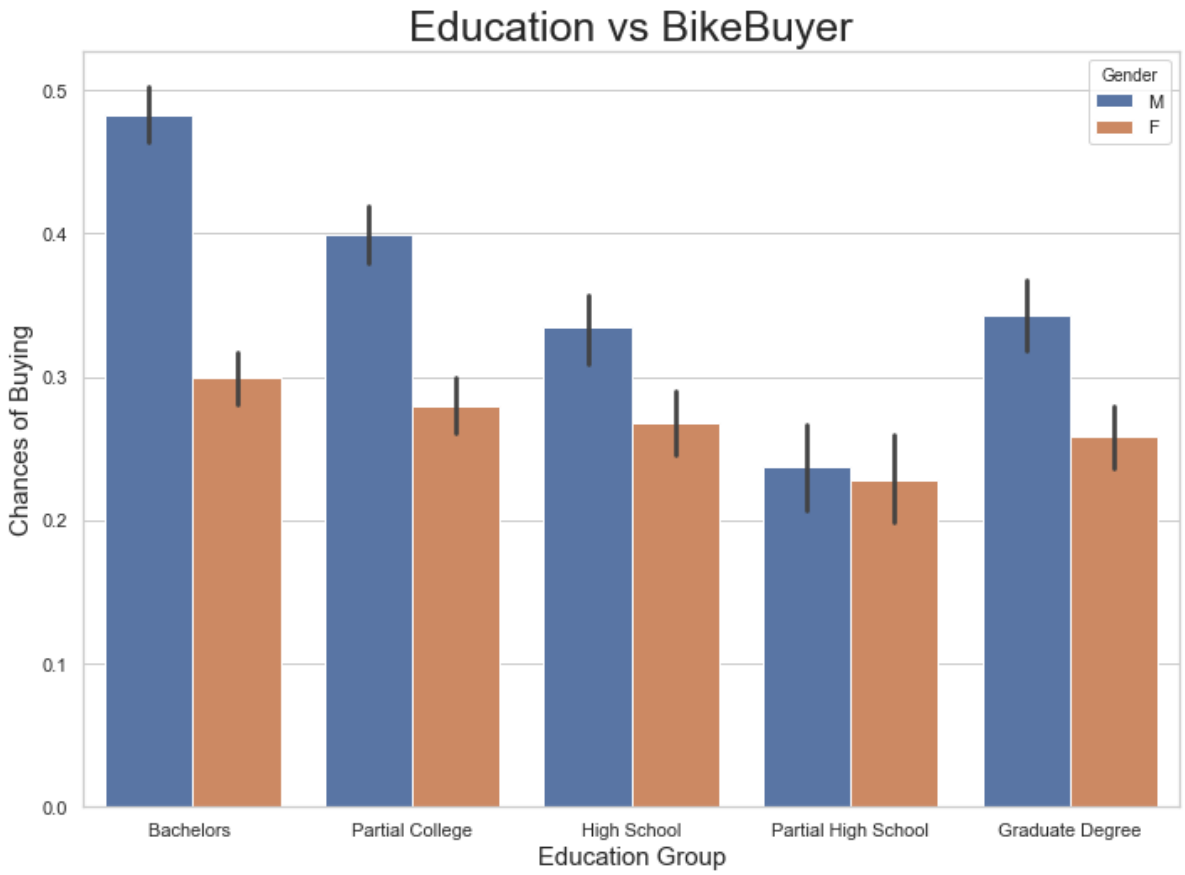
In [63]:

```
from matplotlib import rcParams

# figure size in inches
rcParams['figure.figsize'] = 11.7,8.27
sns.set(style="whitegrid")
ax = sns.barplot(x="Education", y="BikeBuyer",hue='Gender' ,data=df)
ax.set_title('Education vs BikeBuyer',size=25)
ax.set_ylabel('Chances of Buying',size=15)
ax.set_xlabel('Education Group',size=15)
```

Out[63]:

```
Text(0.5, 0, 'Education Group')
```

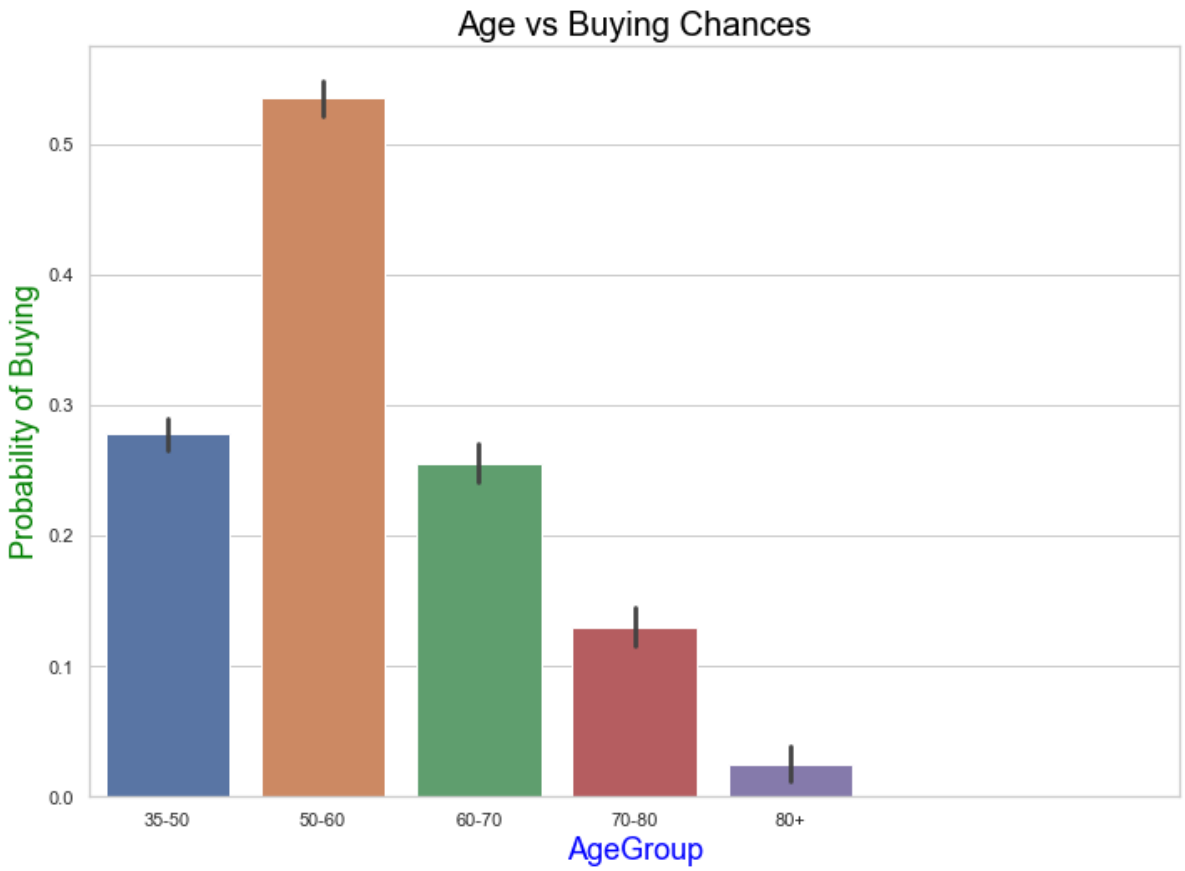


EXPLORING AGE TRENDS

In [64]:

```
bins = [ 35,50,60,70,80,90,100, np.inf]
labels = [ '35-50', '50-60', '60-70', '70-80', '80+', ' ', '  ']
df['AgeGroup'] = pd.cut(df["Age"], bins, labels = labels)
print(df[['AgeGroup','BikeBuyer']].groupby(['AgeGroup'],as_index=False).mean())
sns.barplot(x="AgeGroup",y="BikeBuyer",data=df)
plt.xlabel('AgeGroup',color='blue',size=18)
plt.ylabel('Probability of Buying',color='green',size=18)
plt.title('Age vs Buying Chances',color='Black',size=20)
plt.show()
```

```
  AgeGroup  BikeBuyer
0    35-50   0.277530
1    50-60   0.535330
2    60-70   0.255453
3    70-80   0.130090
4      80+   0.024641
5            0.000000
6            0.000000
```



WORKING PEOPLE ARE NOT MUCH INTO BUYING.

This might seems surprising but yes data reflects the reality. People in age group 35-50 tends to be the non-retired person. They are the backbone of any country senior level workforce. They are not buying the bikes in comparison to the 50-60 age group. Let's try to deduce the factors:

1.Working force may prefer cars over bikes.

2.People in Age Group 50-60 are more free in terms of financial stress. At this point of life, they are trying new and new things. So, they can be one of the safest bet to sell the bike.

3.People in Age Group 35-50 are bearning more stress financially and may be paying fees of their childrens' educations.Let's dig it bit deeper in the next plot.

In [65]:

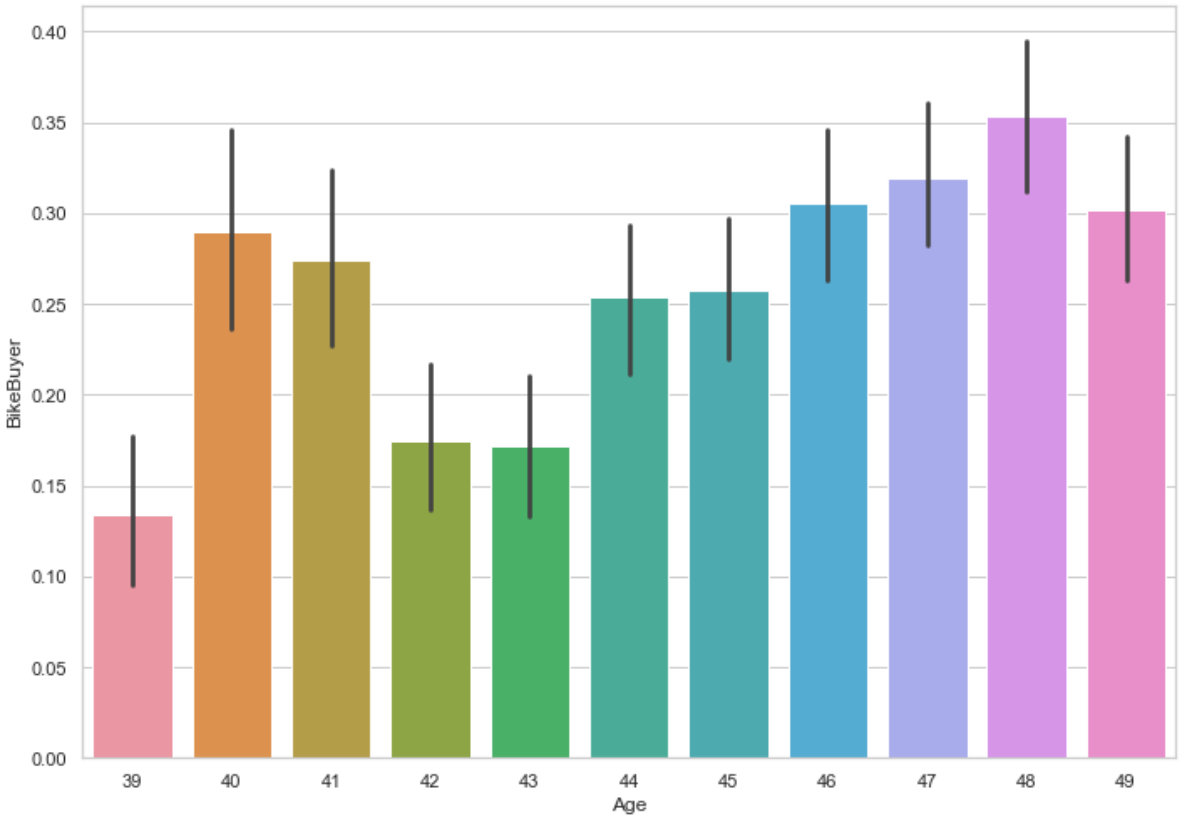
```
df_35_50=df[(df['Age']>35) & (df['Age']<50)]
```

In [66]:

```
sns.barplot(x="Age",y="BikeBuyer",data=df_35_50)
```

Out[66]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a232c3cf8>
```



It is clear from the above bar chart as the person grows older,there are more chances of settling down in terms of wealth.

GENDER PLAYS A HUGE ROLE

Generally men prefer to ride high speed vehicles while women don't, practically they are ok with normal speed vehicles without manual gear shift. It's basically about the choice.

Also that too is a old scenario even lot of women are seen driving geared bikes these days.

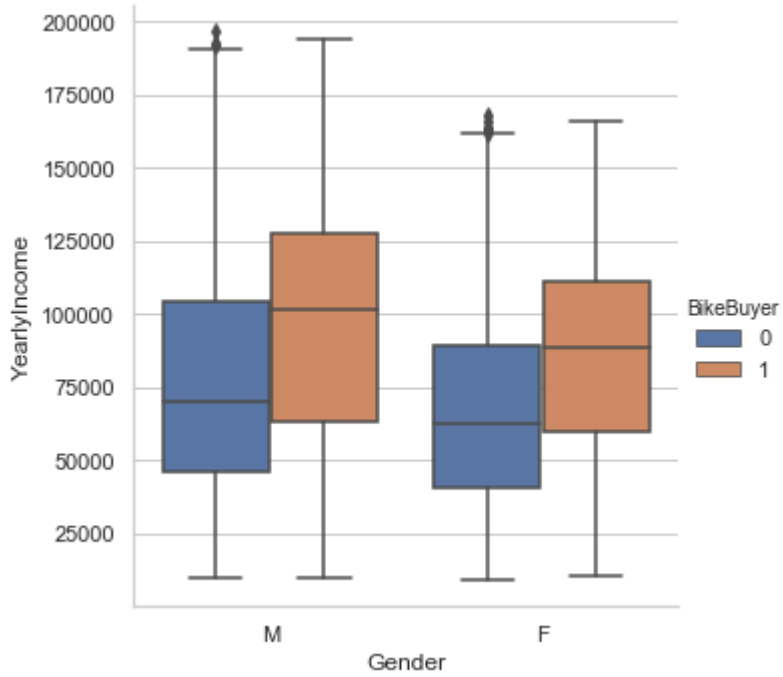
It is again true that automatic geared ones are also well designed to match the performance of manual geared ones.

In [67]:

```
sns.catplot(x="Gender", y="YearlyIncome", hue="BikeBuyer", kind="box", data=df)
```

Out[67]:

```
<seaborn.axisgrid.FacetGrid at 0x1a23302400>
```



LET'S STUDY NATURE OF COUNTRIES

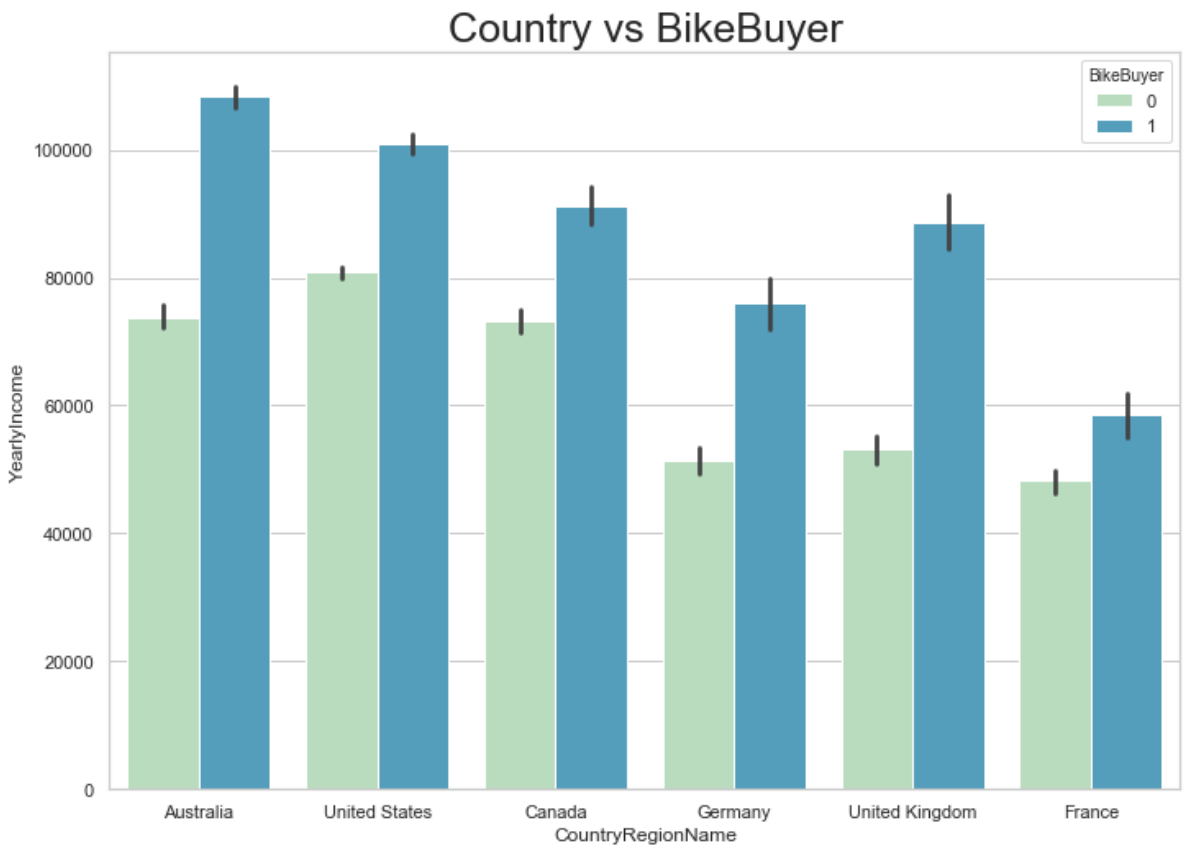
This reflects the average income countrywise.Australia and USA seems the most richest here. It is again following the same principle, healthy wealth gives the leverage to spend.

In [68]:

```
sns.set(style="whitegrid")
ax = sns.barplot(x="CountryRegionName", y="YearlyIncome",hue='BikeBuyer',data=df,pal
ax.set_title('Country vs BikeBuyer',size=25)
```

Out[68]:

```
Text(0.5, 1.0, 'Country vs BikeBuyer')
```



WHICH OCCUPATION PAYS YOU MOST?

A new study has found that the average manager is worth 1.75 employees.

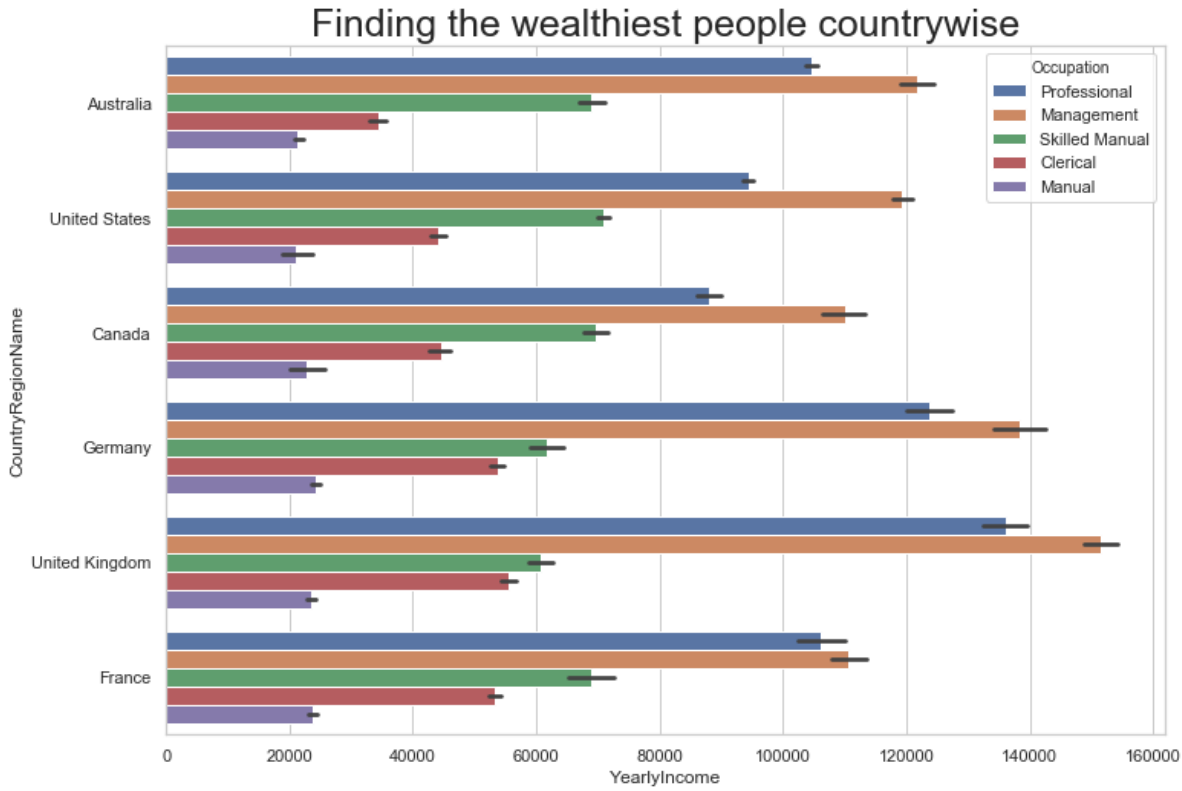
And if you ever complained about how much more they earn, the research suggests the average boss motivates and teaches employees skills which last.

In [69]:

```
sns.set(style="whitegrid")
ax = sns.barplot(x="YearlyIncome", y="CountryRegionName",hue='Occupation' ,data=df)
ax.set_title('Finding the wealthiest people countrywise',size=25)
```

Out[69]:

```
Text(0.5, 1.0, 'Finding the wealthiest people countrywise')
```



MANAGEMENTS GUYS ARE NOT INTO BUYING BIKES.

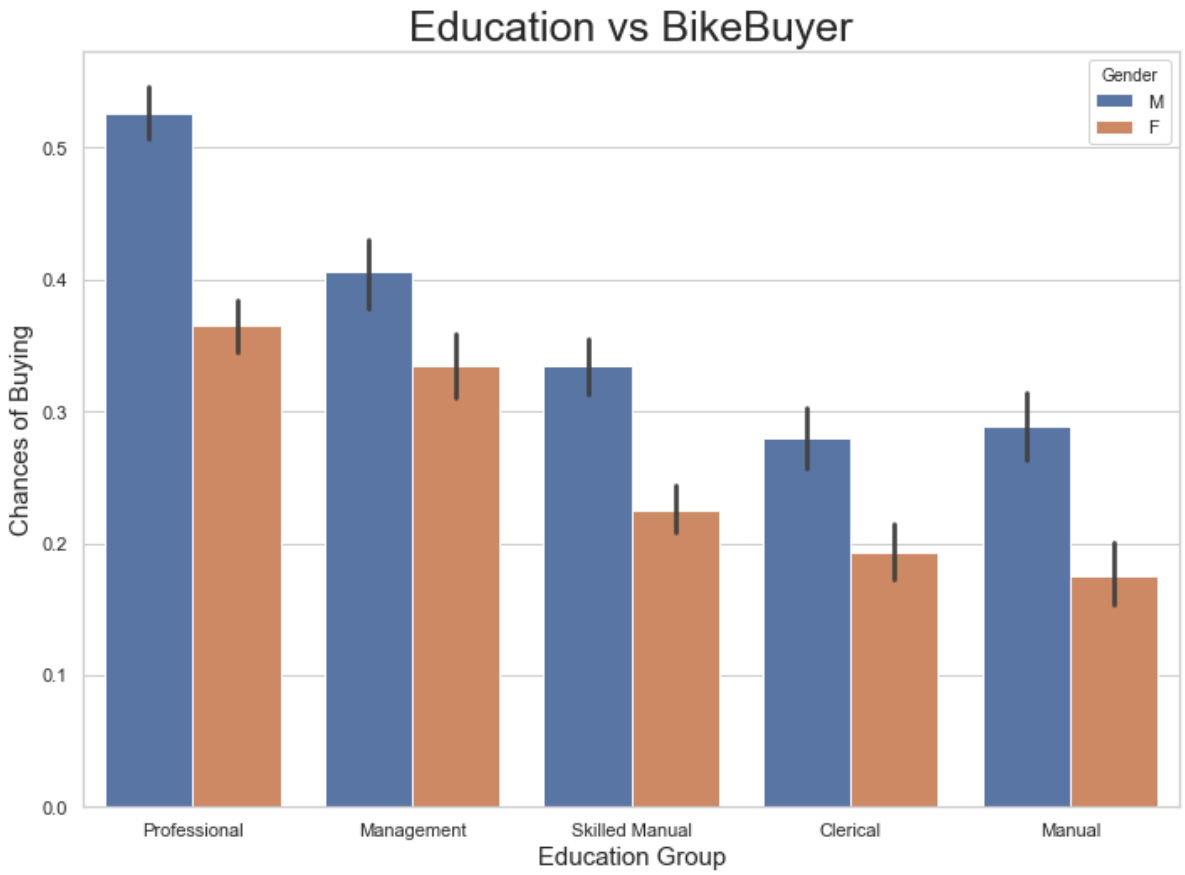
We normally imagine the managers into the sedans and data don't lie. Managers don't want to buy bikes in comparison to the professional people. Other categories may have been pretty weak in terms of finance, so even if they want to buy bikes, they are unable to do so.

In [70]:

```
ax = sns.barplot(x="Occupation", y="BikeBuyer",hue='Gender' ,data=df)
ax.set_title('Education vs BikeBuyer',size=25)
ax.set_ylabel('Chances of Buying',size=15)
ax.set_xlabel('Education Group',size=15)
```

Out[70]:

```
Text(0.5, 0, 'Education Group')
```



In [71]:

```
#I will be selecting important columns for our model!
cols=['CustomerID','City','StateProvinceName','CountryRegionName','NumberCarsOwned',
      'Education','Occupation','Gender','Age',
     'MaritalStatus','HomeOwnerFlag','NumberChildrenAtHome',
      'TotalChildren','YearlyIncome']
```

In [72]:

```
#Lets see how our test dataset looks like!
test.head()
```

Out[72]:

	CustomerID	Title	FirstName	MiddleName	LastName	Suffix	AddressLine1	AddressLine2	
0	18988	NaN	Courtney	A	Baker	NaN	8727 Buena Vista Ave.	NaN	
1	29135	NaN	Adam	C	Allen	NaN	3491 Cook Street	NaN	
2	12156	NaN	Bonnie	NaN	Raji	NaN	359 Pleasant Hill Rd	NaN	
3	13749	NaN	Julio	C	Alonso	NaN	8945 Euclid Ave.	NaN	E
4	27780	NaN	Christy	A	Andersen	NaN	42, boulevard Tremblay	NaN	[

5 rows × 24 columns

In [73]:

```
#Selecting the important features
test=test[cols]
df_d=df[cols]
```

In [74]:

```
#Checking whether our both the data looks similar or not!
if (len(df_d.columns)== len(test.columns)):
    print("Your data seems cool!! \nMOVE AHEAD")
```

```
Your data seems cool!!
MOVE AHEAD
```

In [75]:

```
train_objs_num = len(df)
```

In [76]:

```
#Merging both datasets to create dummy variables
dataset = pd.concat([df_d,test],axis=0)
dummies=pd.get_dummies(data=dataset,columns=['City', 'StateProvinceName','CountryRe
                                             ,'Gender','MaritalStatus'])
dataset=pd.concat([dataset,dummies],axis=1)
```

In [77]:

```
#Separating the datasset
train= dataset[:train_objs_num]
test = dataset[train_objs_num:]
```

In [78]:

```
#Dropping the columns for which we have already created dummmies!
train.drop(['City', 'StateProvinceName','CountryRegionName','Education','Occupation
        'MaritalStatus'],axis=1,inplace=True)
test.drop(['City', 'StateProvinceName','CountryRegionName','Education','Occupation',
        'MaritalStatus'],axis=1,inplace=True)
```

```
//anaconda3/lib/python3.7/site-packages/pandas/core/frame.py:3940: Set
tingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy (http://pandas.pyd
ata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
  errors=errors)
```

We will be using **RandomForestRegressor** to predict **AveMonthSpend** for out test set.

Now lets try to build a model for our target variable called **BikeBuyer**.

In [79]:

```
predictors=train.drop(['CustomerID'],axis=1)
target=df.BikeBuyer
x_train,x_cv,y_train,y_cv=train_test_split(predictors,target,test_size=0.35,random_s
```

In [80]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.neural_network import MLPClassifier
```

In [81]:

```
def scores1(i):
    abc = i()
    abc.fit(x_train, y_train)
    y_pred=abc.predict(x_cv)
    abc = round(accuracy_score(y_pred, y_cv) * 100, 2)
    k.append(abc)
#Checking the scores by using our function
algos=[DecisionTreeClassifier,RandomForestClassifier,MLPClassifier,KNeighborsClassi
k=[]
for i in algos:
    scores1(i)
```

```
//anaconda3/lib/python3.7/site-packages/sklearn/ensemble/forest.py:24
5: FutureWarning: The default value of n_estimators will change from 1
0 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
//anaconda3/lib/python3.7/site-packages/sklearn/neural_network/multila
yer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Maxim
um iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

In [82]:

```
dtc = DecisionTreeClassifier()
dtc.fit(x_train, y_train)
y_pred=dtc.predict(x_cv)
dtc_r = round(accuracy_score(y_pred, y_cv) * 100, 2)
print(dtc_r)
```

```
75.73
```

In [83]:

```
knn = KNeighborsClassifier()
knn.fit(x_train, y_train)
y_pred=knn.predict(x_cv)
knn_r = round(accuracy_score(y_pred, y_cv) * 100, 2)
print(knn_r)
```

```
65.29
```

In [84]:

```
rfc = RandomForestClassifier()
rfc.fit(x_train, y_train)
y_pred=rfc.predict(x_cv)
rfc_r = round(accuracy_score(y_pred, y_cv) * 100, 2)
print(rfc_r)
```

```
//anaconda3/lib/python3.7/site-packages/sklearn/ensemble/forest.py:24
5: FutureWarning: The default value of n_estimators will change from 1
0 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)

78.08
```

In [85]:

```
from sklearn.neural_network import MLPClassifier
nn = MLPClassifier()
nn.fit(x_train, y_train)
y_pred=nn.predict(x_cv)
nn = round(accuracy_score(y_pred, y_cv) * 100, 2)
print(nn)
```

```
48.41

//anaconda3/lib/python3.7/site-packages/sklearn/neural_network/multila
yer_perceptron.py:566: ConvergenceWarning: Stochastic Optimizer: Maxim
um iterations (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

In [86]:

```
models = pd.DataFrame({
    'Method': ['DecisionTreeClassifier',
               'RandomForestClassifier', 'MLPClassifier','KneighborsClassifier'],
    'Score': [k[0],k[1],k[2],k[3]]})
models.sort_values(by='Score', ascending=False)
```

Out[86]:

	Method	Score
1	RandomForestClassifier	77.83
2	MLPClassifier	77.30
0	DecisionTreeClassifier	74.88
3	KneighborsClassifier	65.29

We will be using **RandomForestClassifier** to predict the **BikeBuyer** feature for the test set!

In [87]:

```
#Time for predictition for our test datatset
```

In [88]:

```
#predicting  BikeBuyer
model_bike = RandomForestClassifier()
model_bike.fit(x_train, y_train)
y_pred = model_bike.predict(test.drop(test.columns[0],axis=1))
print(y_pred)
```

```
//anaconda3/lib/python3.7/site-packages/sklearn/ensemble/forest.py:24
5: FutureWarning: The default value of n_estimators will change from 1
0 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)

[0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 1 1 1 1 1 0 0 0
1 1
 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 1 0 1 0 0
0 0
 0 1 1 1 0 0 1 0 0 1 0 0 1 1 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 1 1 0 0 0 0
0 1
 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 1 1 0 1 1 0 1 1 0 0 0 0 1 0
0 0
 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1
0 0
 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 0 1 0
0 0
 0 1 0 0 0 1 1 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0
0 1
 0 0 0 1 0 1 1 0 1 1 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 1
 1 0 0 0 0 0 0 1 0 0 1 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1
0 1
 0 1 0 1 0 1 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 1 1 0 1 0 0 0 0
0 0
 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 0 0 1 0 0 1 0 0 1 1 0 0 0 0 0 1 1 1
0 1
 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 1
0 0
 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 1 0 0 0 0 0 0 0 1 0
1 1
 1 1 0 0 0 1 0 0 0 1 1 0 0 0 0 1 1 0 0]
```

In [89]:

```
#Adding these values into out orginal test set!
test_org=pd.read_csv('AW_test.csv')
```

In [90]:

```
#Assigning our predictions
test_org['BikeBuyer']=y_pred
```

In [91]:

```
#Saving our predictions in a csv file for further use!
test_org.to_csv('AW_test1.csv')
```

In [92]:

```
test_org.head()
```

Out[92]:

	CustomerID	Title	FirstName	MiddleName	LastName	Suffix	AddressLine1	AddressLine2	
0	18988	NaN	Courtney	A	Baker	NaN	8727 Buena Vista Ave.	NaN	
1	29135	NaN	Adam	C	Allen	NaN	3491 Cook Street	NaN	
2	12156	NaN	Bonnie	NaN	Raji	NaN	359 Pleasant Hill Rd	NaN	
3	13749	NaN	Julio	C	Alonso	NaN	8945 Euclid Ave.	NaN	E
4	27780	NaN	Christy	A	Andersen	NaN	42, boulevard Tremblay	NaN	[

5 rows × 24 columns

In [ ]:

```

```

In [ ]:

```

```