# LogisticRegression

April 16, 2019

Gahan Saraiya (18MCEC10)

**AIM: Implementation of Logistic Regression**

Logistic Regression

In [30]:
```
import pandas as pd
import numpy as np
from sklearn import preprocessing
import matplotlib.pyplot as plt
plt.rc("font", size=14)
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import seaborn as sns
sns.set(style="white")
sns.set(style="whitegrid", color_codes=True)
```

In [38]:
```
data = pd.read_csv("banking.csv", low_memory=False)
data = data.dropna()
data.head(5)
```

Out[38]:
```
   age          job  marital          education  default housing loan  \
0   44  blue-collar  married           basic.4y  unknown     yes   no
1   53   technician  married            unknown       no      no   no
2   28   management   single  university.degree       no     yes   no
3   39     services  married        high.school       no      no   no
4   55      retired  married           basic.4y       no     yes   no

    contact month day_of_week  ...  campaign  pdays  previous     poutcome  \
0  cellular   aug         thu  ...         1    999         0  nonexistent
1  cellular   nov         fri  ...         1    999         0  nonexistent
2  cellular   jun         thu  ...         3      6         2      success
3  cellular   apr         fri  ...         2    999         0  nonexistent
4  cellular   aug         fri  ...         1      3         1      success

  emp_var_rate  cons_price_idx  cons_conf_idx  euribor3m  nr_employed  y
0          1.4          93.444          -36.1      4.963       5228.1  0
1         -0.1          93.200          -42.0      4.021       5195.8  0
2         -1.7          94.055          -39.8      0.729       4991.6  1
3         -1.8          93.075          -47.1      1.405       5099.1  0
```

```
4          -2.9          92.201          -31.4       0.869        5076.2  1

[5 rows x 21 columns]
```

In [40]:
```
data.education.unique()
```

Out[40]:
```
array(['basic.4y', 'unknown', 'university.degree', 'high.school',
       'basic.9y', 'professional.course', 'basic.6y', 'illiterate'],
      dtype=object)
```

group “basic.4y”, “basic.9y” and “basic.6y” together and call them “basic”.

In [41]:
```
data['education']=np.where(data['education'] =='basic.9y', 'Basic', data['education']
data['education']=np.where(data['education'] =='basic.6y', 'Basic', data['education']
data['education']=np.where(data['education'] =='basic.4y', 'Basic', data['education']
```

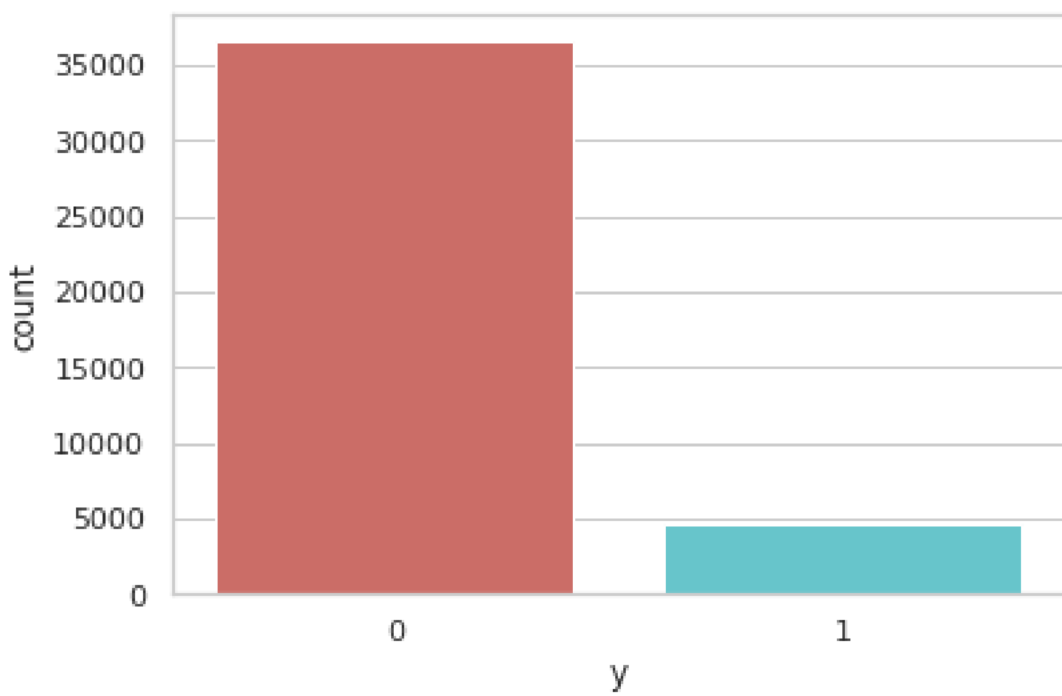
In [42]:
```
data.education.unique()
```

Out[42]:
```
array(['Basic', 'unknown', 'university.degree', 'high.school',
       'professional.course', 'illiterate'], dtype=object)
```

In [43]:
```
data.y.value_counts()
```

Out[43]:
```
0    36548
1     4640
Name: y, dtype: int64
```

In [44]:
```
sns.countplot(x='y', data=data, palette='hls')
plt.show()
```



```
In [45]: count_no_sub = len(data[data['y']==0])
         count_sub = len(data[data['y']==1])
         pct_of_no_sub = count_no_sub/(count_no_sub+count_sub)
         print("percentage of no subscription is", pct_of_no_sub*100)
         pct_of_sub = count_sub/(count_no_sub+count_sub)
         print("percentage of subscription", pct_of_sub*100)
```

```
percentage of no subscription is 88.73458288821988
percentage of subscription 11.265417111780131
```

```
In [46]: data.groupby('y').mean()
```

```
Out[46]:          age    duration  campaign       pdays  previous  emp_var_rate  \
         y
         0  39.911185  220.844807  2.633085  984.113878  0.132374      0.248875
         1  40.913147  553.191164  2.051724  792.035560  0.492672     -1.233448

            cons_price_idx  cons_conf_idx  euribor3m  nr_employed
         y
         0       93.603757     -40.593097   3.811491  5176.166600
         1       93.354386     -39.789784   2.123135  5095.115991
```

## 0.1 Intution:

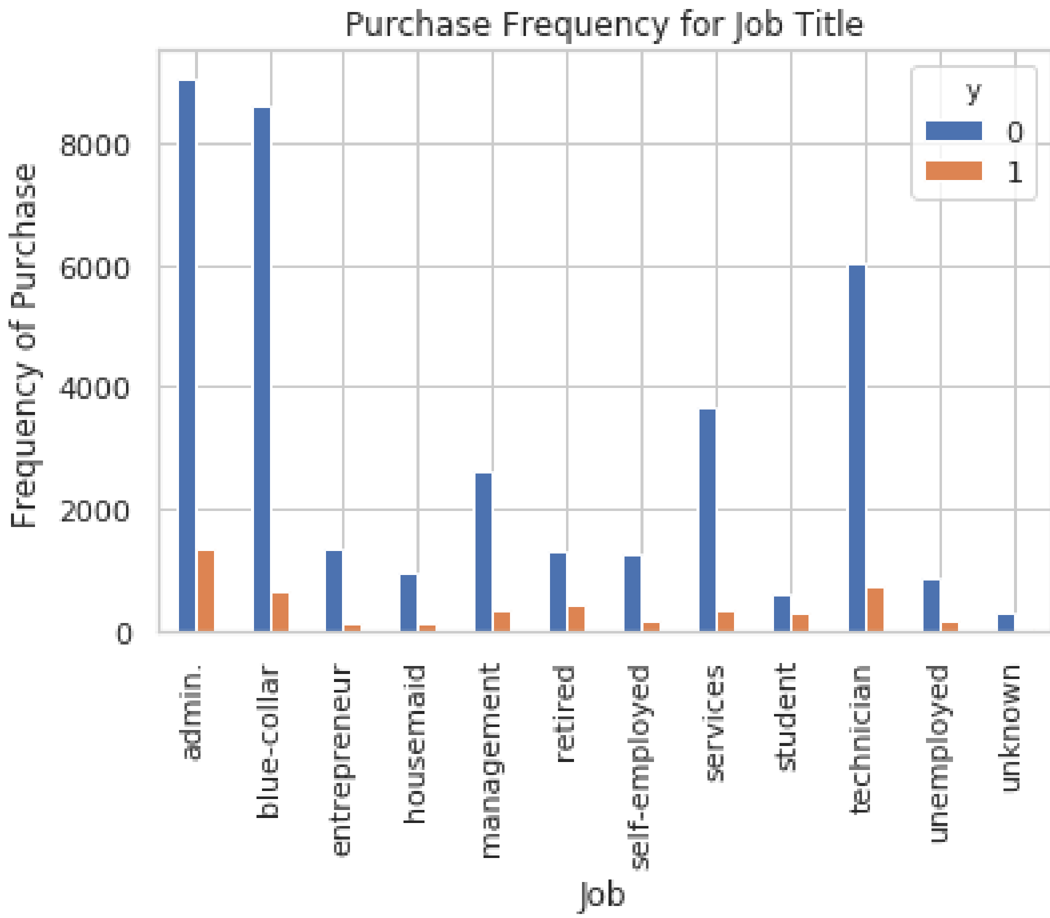
- The average age of customers who bought the term deposit is higher than that of the customers who didn't.
- The pdays (days since the customer was last contacted) is understandably lower for the customers who bought it. The lower the pdays, the better the memory of the last call and hence the better chances of a sale.
- Surprisingly, campaigns (number of contacts or calls made during the current campaign) are lower for customers who bought the term deposit.

```
In [47]: data.groupby('job').mean()
```

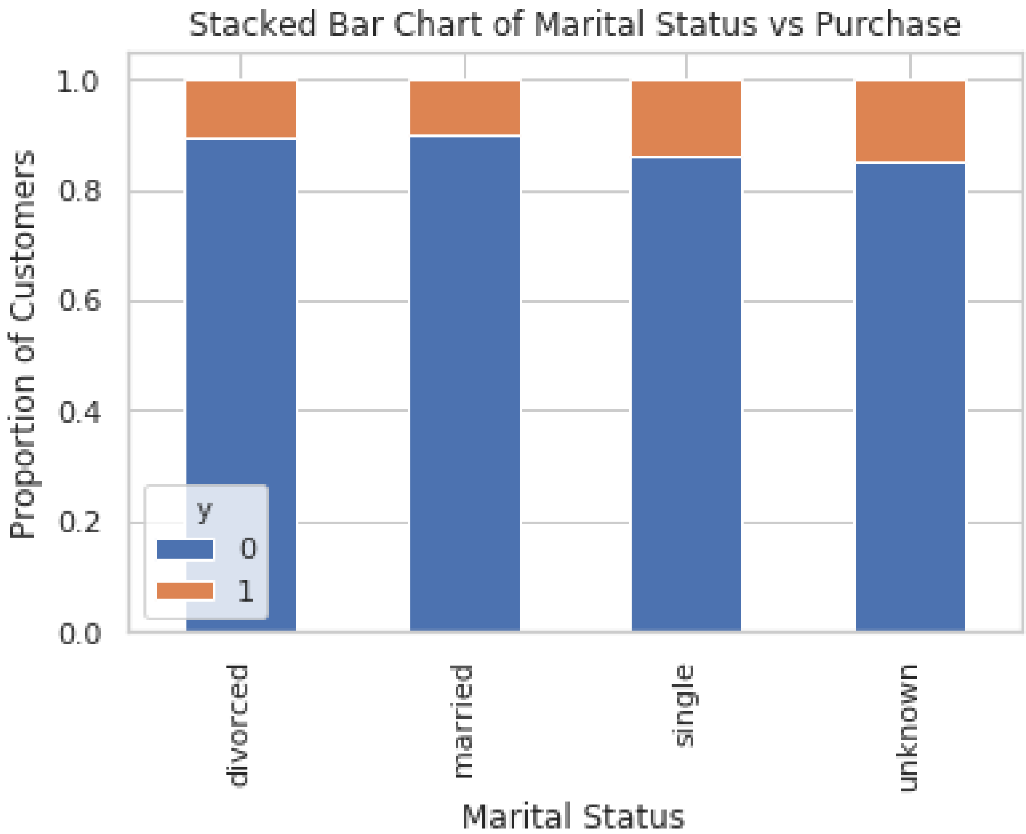
```
Out[47]:                      age    duration  campaign       pdays  previous  \
         job
         admin.         38.187296  254.312128  2.623489  954.319229  0.189023
         blue-collar    39.555760  264.542360  2.558461  985.160363  0.122542
         entrepreneur   41.723214  263.267857  2.535714  981.267170  0.138736
         housemaid      45.500000  250.454717  2.639623  960.579245  0.137736
         management     42.362859  257.058140  2.476060  962.647059  0.185021
         retired        62.027326  273.712209  2.476744  897.936047  0.327326
         self-employed  39.949331  264.142153  2.660802  976.621393  0.143561
         services       37.926430  258.398085  2.587805  979.974049  0.154951
         student        25.894857  283.683429  2.104000  840.217143  0.524571
```

```
technician     38.507638  250.232241  2.577339  964.408127  0.153789
unemployed     39.733728  249.451677  2.564103  935.316568  0.199211
unknown        45.563636  239.675758  2.648485  938.727273  0.154545

               emp_var_rate  cons_price_idx  cons_conf_idx  euribor3m  \
job
admin.             0.015563       93.534054     -40.245433   3.550274
blue-collar        0.248995       93.656656     -41.375816   3.771996
entrepreneur       0.158723       93.605372     -41.283654   3.791120
housemaid          0.433396       93.676576     -39.495283   4.009645
management        -0.012688       93.522755     -40.489466   3.611316
retired           -0.698314       93.430786     -38.573081   2.770066
self-employed      0.094159       93.559982     -40.488107   3.689376
services           0.175359       93.634659     -41.290048   3.699187
student           -1.408000       93.331613     -40.187543   1.884224
technician         0.274566       93.561471     -39.927569   3.820401
unemployed        -0.111736       93.563781     -40.007594   3.466583
unknown            0.357879       93.718942     -38.797879   3.949033

               nr_employed         y
job
admin.         5164.125350  0.129726
blue-collar    5175.615150  0.068943
entrepreneur   5176.313530  0.085165
housemaid      5179.529623  0.100000
management     5166.650513  0.112175
retired        5122.262151  0.252326
self-employed  5170.674384  0.104856
services       5171.600126  0.081381
student        5085.939086  0.314286
technician     5175.648391  0.108260
unemployed     5157.156509  0.142012
unknown        5172.931818  0.112121
```

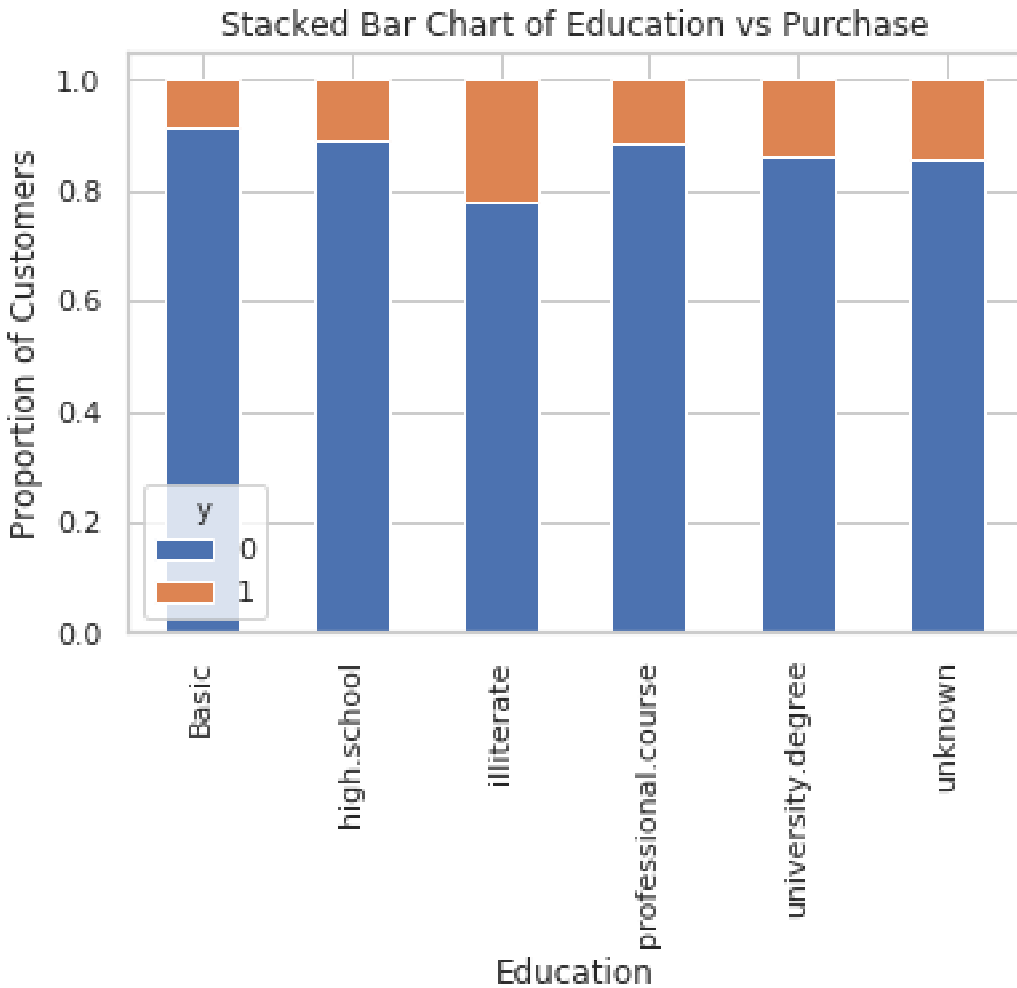
In [48]:
```
%matplotlib inline
pd.crosstab(data.job,data.y).plot(kind='bar')
plt.title('Purchase Frequency for Job Title')
plt.xlabel('Job')
plt.ylabel('Frequency of Purchase')
plt.savefig('purchase_fre_job')
```



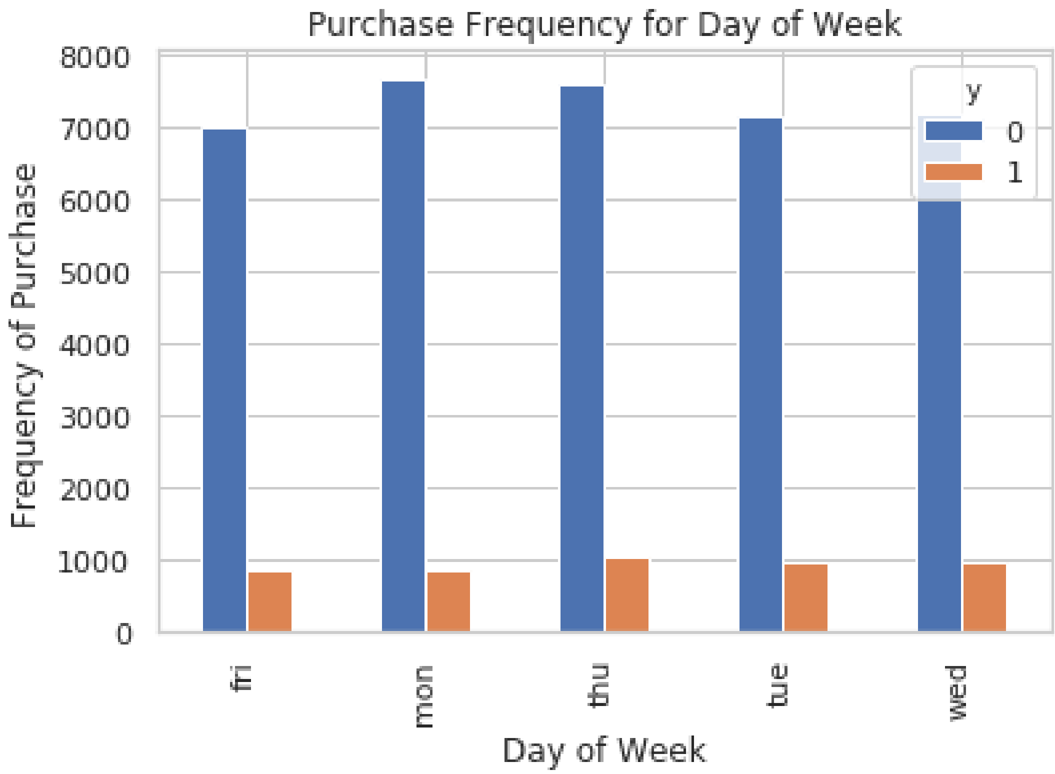

In [49]:
```
table=pd.crosstab(data.marital,data.y)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True)
plt.title('Stacked Bar Chart of Marital Status vs Purchase')
plt.xlabel('Marital Status')
plt.ylabel('Proportion of Customers')
plt.savefig('mariral_vs_pur_stack')
```



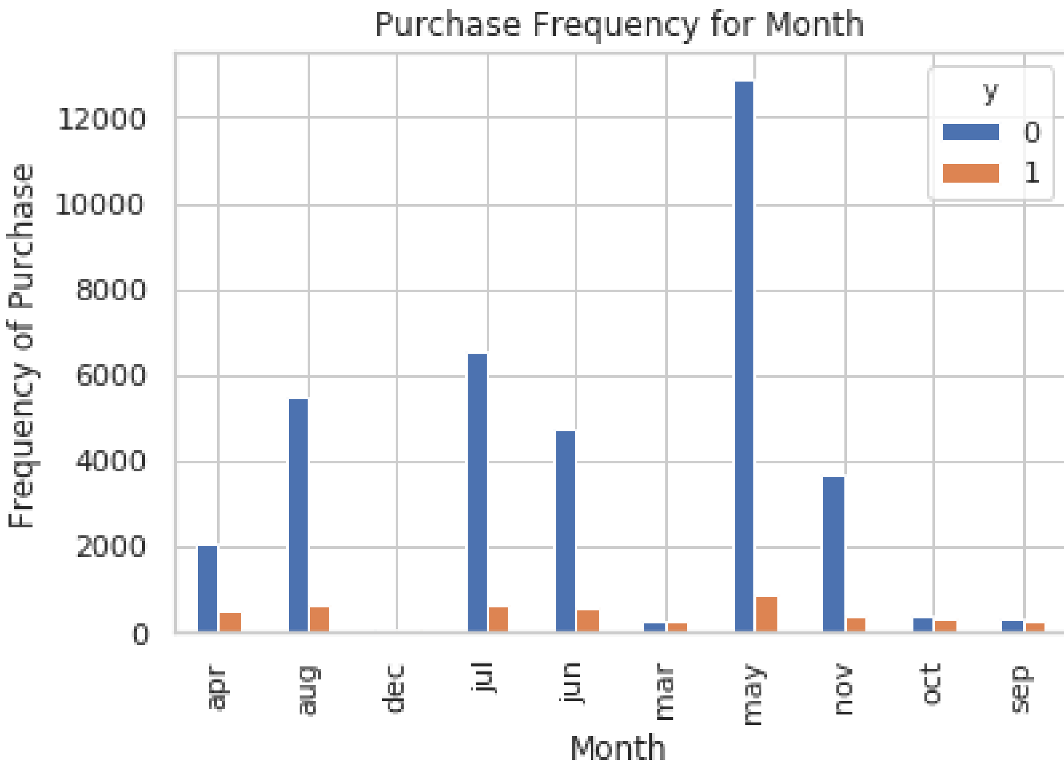

In [50]:
```
table=pd.crosstab(data.education,data.y)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True)
plt.title('Stacked Bar Chart of Education vs Purchase')
plt.xlabel('Education')
plt.ylabel('Proportion of Customers')
plt.savefig('edu_vs_pur_stack')
```



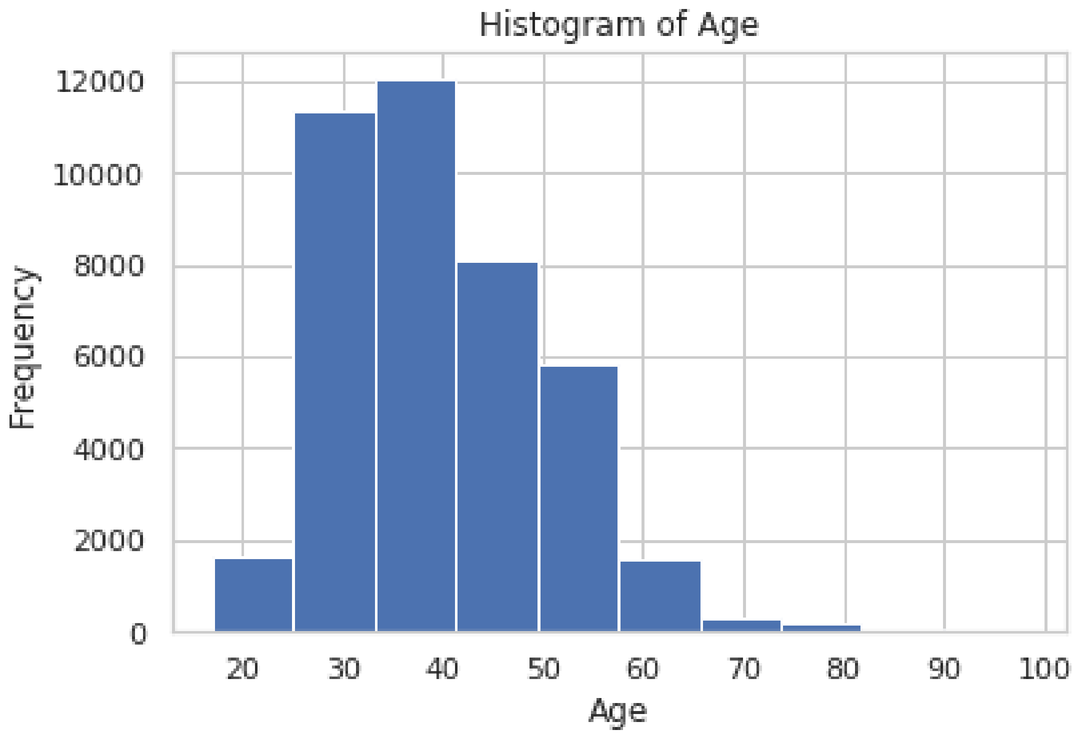

```
In [51]: pd.crosstab(data.day_of_week,data.y).plot(kind='bar')
         plt.title('Purchase Frequency for Day of Week')
         plt.xlabel('Day of Week')
         plt.ylabel('Frequency of Purchase')
         plt.savefig('pur_dayofweek_bar')
```




```
In [52]: pd.crosstab(data.month,data.y).plot(kind='bar')
         plt.title('Purchase Frequency for Month')
         plt.xlabel('Month')
         plt.ylabel('Frequency of Purchase')
         plt.savefig('pur_fre_month_bar')
```




```
In [53]: data.age.hist()
         plt.title('Histogram of Age')
         plt.xlabel('Age')
         plt.ylabel('Frequency')
         plt.savefig('hist_age')
```




```
In [56]: from sklearn.linear_model import LogisticRegression
         from sklearn import metrics
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=
         logreg = LogisticRegression()
         logreg.fit(X_train, y_train)
```

```
/usr/local/lib/python3.5/dist-packages/sklearn/linear_model/logistic.py:433: FutureWarning: De
  FutureWarning)
```

```
Out[56]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, max_iter=100, multi_class='warn',
                    n_jobs=None, penalty='l2', random_state=None, solver='warn',
                    tol=0.0001, verbose=0, warm_start=False)
```

```
In [58]: y_pred = logreg.predict(X_test)
         print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.s
```

```
Accuracy of logistic regression classifier on test set: 1.00
```

# 1 Confusion Matrix

```
In [59]: from sklearn.metrics import confusion_matrix
         confusion_matrix = confusion_matrix(y_test, y_pred)
```

```
        print(confusion_matrix)
```

```
[[16  0]
 [ 0 29]]
```

```
In [60]: from sklearn.metrics import classification_report
         print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

       False       1.00      1.00      1.00        16
        True       1.00      1.00      1.00        29

   micro avg       1.00      1.00      1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

```
In [ ]:
```