Homework 2 - IEEE Fraud Detection

For all parts below, answer all parts as shown in the Google document for Homework 2. Be sure to include both code that justifies your answer as well as text to answer the questions. We also ask that code be commented to make it easier to follow.

Part 1 - Fraudulent vs Non-Fraudulent Transaction

In [1271]:
```
# TODO: code and runtime results
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import datetime
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.datasets import make_classification
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
import sklearn.metrics as metrics
from sklearn.utils import resample
import sys

class color:
   PURPLE = '\033[95m'
   CYAN = '\033[96m'
   DARKCYAN = '\033[36m'
   BLUE = '\033[94m'
   GREEN = '\033[92m'
   YELLOW = '\033[93m'
   RED = '\033[91m'
   BOLD = '\033[1m'
   UNDERLINE = '\033[4m'
   END = '\033[0m'
```

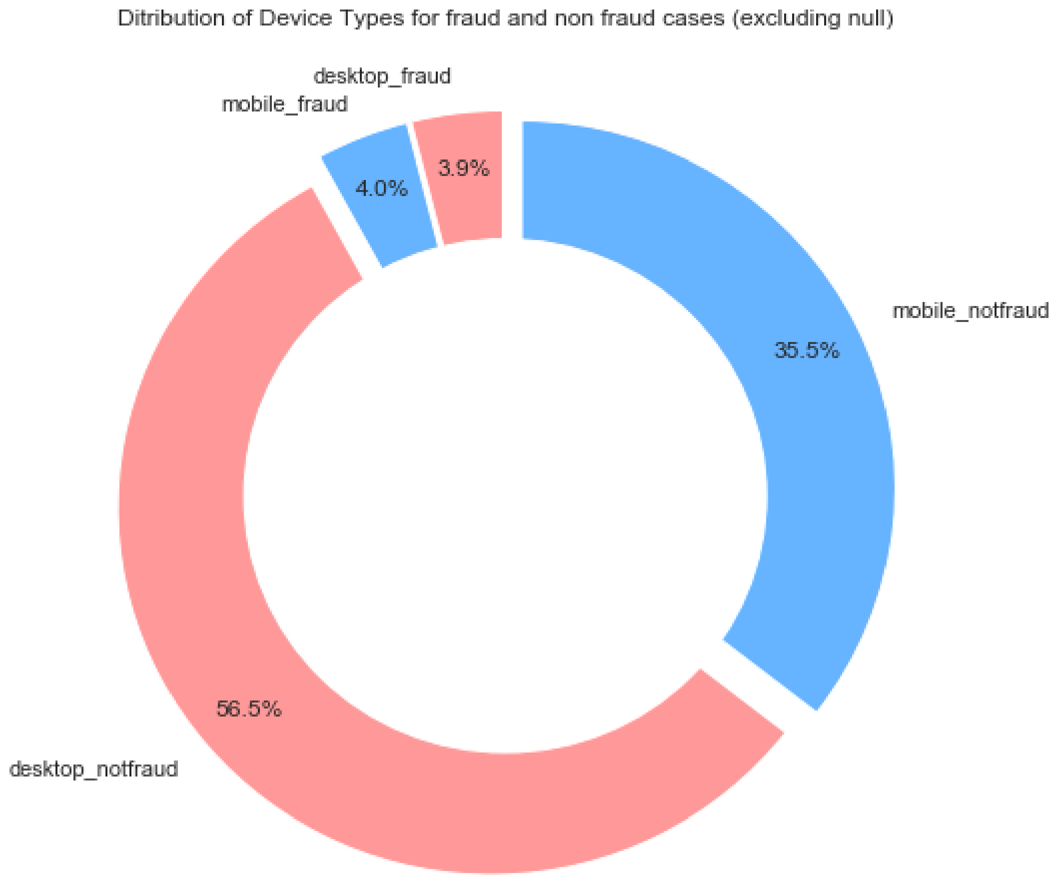
In [1156]:
```
transactions = pd.read_csv("train_transaction.csv")
transactions_test = pd.read_csv("test_transaction.csv")
identity = pd.read_csv("train_identity.csv")
identity_test = pd.read_csv("test_identity.csv")
merged_tables = pd.merge(left=transactions, how='left', right=identity,
left_on='TransactionID', right_on='TransactionID')
merged_tables_test = pd.merge(left=transactions_test, how='left', right=
identity_test, left_on='TransactionID', right_on='TransactionID')
```

In [1272]:
```
dataset = pd.DataFrame(merged_tables, columns=['TransactionID', 'isFraud', 'DeviceType', 'DeviceInfo', 'TransactionDT', 'TransactionAmt', 'ProductCD', 'card4', 'card6', 'P_emaildomain', 'R_emaildomain', 'addr1', 'addr2', 'dist1', 'dist2'])
# Removing NaN and null data
dataset['addr1'].fillna(dataset['addr1'].mean(), inplace=True)
dataset['dist1'].fillna(-9999, inplace=True)
dataset['dist2'].fillna(-9999, inplace=True)
dataset['addr2'].fillna(-9999, inplace=True)
dataset['DeviceType'].fillna('Blank', inplace=True)
dataset['DeviceInfo'].fillna('Blank', inplace=True)
dataset['TransactionAmt'].fillna(dataset['TransactionAmt'].mean(), inplace=True)
dataset['P_emaildomain'].fillna('Blank', inplace=True)
dataset['R_emaildomain'].fillna('Blank', inplace=True)

# Time reference selected as 1st January 2019
reference = datetime.datetime.strptime('2019-01-01', '%Y-%m-%d')
dataset['TransactionDT'] = dataset['TransactionDT'].apply(lambda x: (reference + datetime.timedelta(seconds = x)))
dataset['hours'] = dataset.TransactionDT.dt.hour

# Finding the number of rows
size_dataset = dataset.shape[0]
size_trans_fraud = trans_fraud.shape[0]
size_trans_not_fraud = trans_not_fraud.shape[0]

# Separating data to 2 Dataframes
trans_fraud = dataset.loc[dataset['isFraud'] == 1]
trans_not_fraud =  dataset.loc[dataset['isFraud'] != 1]



# Preprocessing test data
dataset_test = pd.DataFrame(merged_tables_test, columns=['TransactionID', 'isFraud', 'DeviceType', 'DeviceInfo', 'TransactionDT', 'TransactionAmt', 'ProductCD', 'card4', 'card6', 'P_emaildomain', 'R_emaildomain', 'addr1', 'addr2', 'dist1', 'dist2'])
dataset_test['addr1'].fillna('Blank', inplace=True)
dataset_test['dist1'].fillna(-9999, inplace=True)
dataset_test['dist2'].fillna(-9999, inplace=True)
dataset_test['addr2'].fillna('Blank', inplace=True)
dataset_test['DeviceType'].fillna('Blank', inplace=True)
dataset_test['DeviceInfo'].fillna('Blank', inplace=True)
dataset_test['TransactionAmt'].fillna(-9999, inplace=True)
dataset_test['P_emaildomain'].fillna('Blank', inplace=True)
dataset_test['R_emaildomain'].fillna('Blank', inplace=True)

# Time reference selected as 1st January 2019
reference = datetime.datetime.strptime('2019-01-01', '%Y-%m-%d')
dataset_test['TransactionDT'] = dataset_test['TransactionDT'].apply(lambda x: (reference + datetime.timedelta(seconds = x)))
dataset_test['hours'] = dataset_test.TransactionDT.dt.hour
```

In [1273]:
```
# Evaluating addr2
dataset_copy = dataset.copy()
frequency_country_codes = dataset_copy.groupby("addr2").size().rename_ax
is("addr2").reset_index(name="count")
frequency_country_codes = frequency_country_codes.sort_values(by='count'
, ascending=False).head()

percent_top = (frequency_country_codes.max()[1] * 100) / size_dataset
print "The country code has the maximum frequency ",percent_top,"% for c
ountry code ", list(frequency_country_codes['addr2'])[0]
```

```
The country code has the maximum frequency  88.13645138347952 % for cou
ntry code  87.0
```

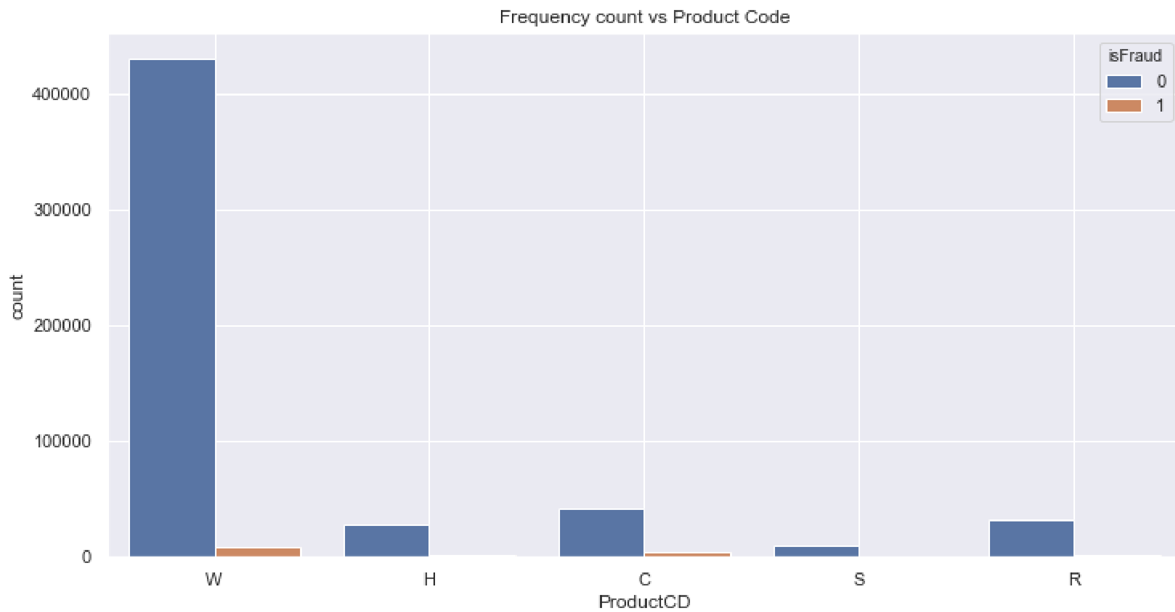
In [1274]:
```
# Distribution of DeviceType
plt.figure(figsize= (7,7))
sns.set(style="darkgrid")

trans_fraud_copy = trans_fraud.copy()
trans_not_fraud_copy = trans_not_fraud.copy()

# Pie chart
unique_devices_count_fraud = trans_fraud_copy.groupby("DeviceType").size
().rename_axis("DeviceType").reset_index(name="count")
unique_devices_count_notfraud = trans_not_fraud_copy.groupby("DeviceTyp
e").size().rename_axis("DeviceType").reset_index(name="count")

# Add percent
unique_devices_count_fraud['percent'] = (unique_devices_count_fraud['cou
nt'] * 100) / size_dataset
unique_devices_count_notfraud['percent'] = (unique_devices_count_notfrau
d['count'] * 100) / size_dataset

# Remove 'Blank' value
unique_devices_count_fraud = unique_devices_count_fraud[unique_devices_c
ount_fraud['DeviceType'] != 'Blank']
unique_devices_count_notfraud = unique_devices_count_notfraud[unique_dev
ices_count_notfraud['DeviceType'] != 'Blank']

unique_devices_count_fraud.DeviceType = unique_devices_count_fraud.Devic
eType + "_fraud"
unique_devices_count_notfraud.DeviceType = unique_devices_count_notfraud
.DeviceType + "_notfraud"

x_values = list(unique_devices_count_fraud.DeviceType) + list(unique_dev
ices_count_notfraud.DeviceType)
y_values = list(unique_devices_count_fraud.percent ) + list(unique_devic
es_count_notfraud.percent)

colors = ['#ff9999','#66b3ff']
explode = (0.05,0.05,0.05,0.05)

plt.pie(y_values, colors = colors, labels=x_values, autopct='%1.1f%%', s
tartangle=90, pctdistance=0.85, explode = explode)
#draw circle
centre_circle = plt.Circle((0,0),0.70,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)
# Equal aspect ratio ensures that pie is drawn as a circle
ax1.axis('equal')
plt.title("Ditribution of Device Types for fraud and non fraud cases (ex
cluding null)")
plt.tight_layout()
plt.show()
```



The plot shows that it is almost likely that the fraud transaction can happen on mobile with similar probability as on desktop. But the chance of fraud happening is quite less than the chance of fraud not happening.

In [1292]:
```
# Distribution of DeviceInfo
plt.figure(figsize= (12,6))
dataset_copy = dataset.copy()
pick=dataset_copy["DeviceInfo"].value_counts()[:5].index
top_device_info=dataset_copy.loc[dataset["DeviceInfo"].isin(pick)]
sns.countplot(x="ProductCD", hue="isFraud", data=top_device_info)
plt.grid(True)
plt.title("Frequency count vs Product Code")
```

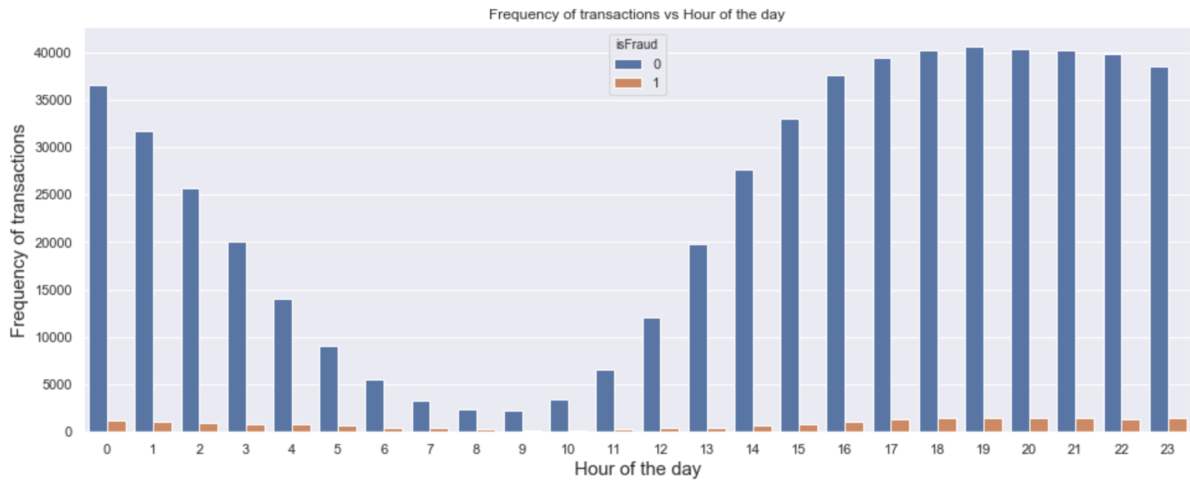
Out[1292]: Text(0.5,1,'Frequency count vs Product Code')



The plot shows that Windows devices are more prone to fraudulent transactions than any other device type.

In [1293]:
```
# Distribution of TransactionDT
plt.figure(figsize= (16,6))
sns.countplot(x=dataset.hours, hue='isFraud', data=dataset)
plt.xlabel('Hour of the day', size= 15)
plt.ylabel('Frequency of transactions', size= 15)
plt.title("Frequency of transactions vs Hour of the day")
```

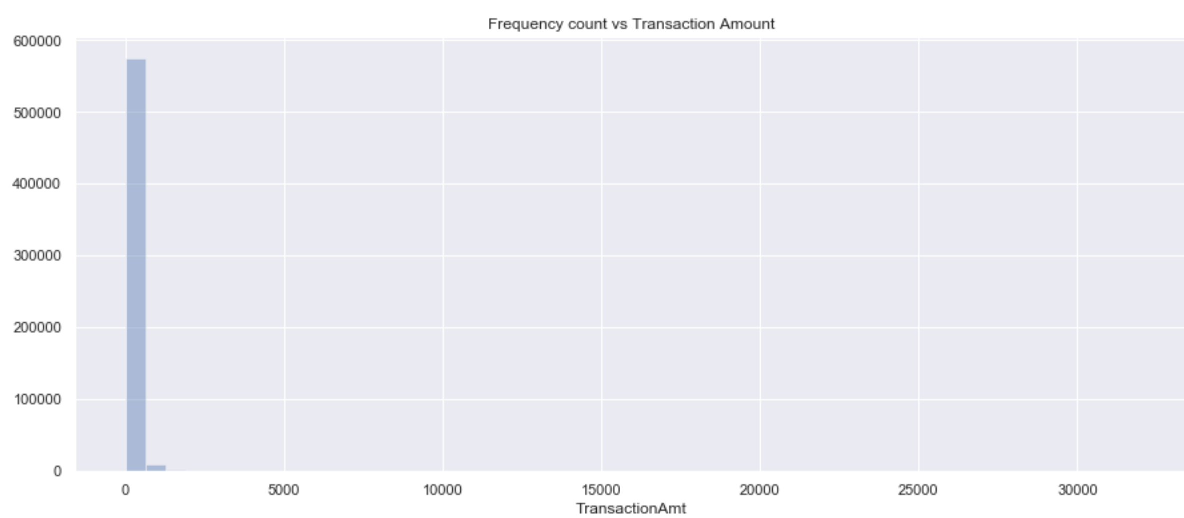
Out[1293]: Text(0.5,1,'Frequency of transactions vs Hour of the day')



Plotting the distribution of hours of the day with the frequency of transactions, we see that the number of fraud and non-fraud transactions follow the same pattern in which the transactions are maximum in the night and minimum between 7th to 10th hour of the day.

In [1294]:
```
# Distribution of TransactionAmt
plt.figure(figsize= (15,6))
sns.distplot(dataset.TransactionAmt, kde = False, rug = False)
plt.title("Frequency count vs Transaction Amount")
```

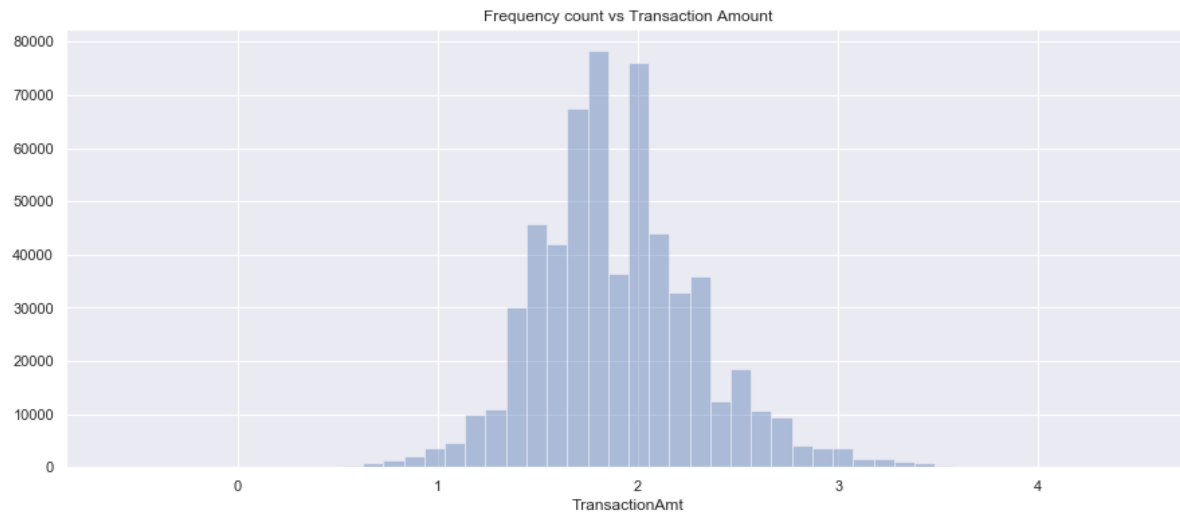
Out[1294]: Text(0.5,1,'Frequency count vs Transaction Amount')



This plot does not show proper distribution of transaction amount with the frequency of transactions.

In [1295]:
```
# Distribution of TransactionAmt
plt.figure(figsize= (15,6))
sns.distplot(np.log10(dataset.TransactionAmt), kde = False, rug = False)
plt.title("Frequency count vs Transaction Amount")
```

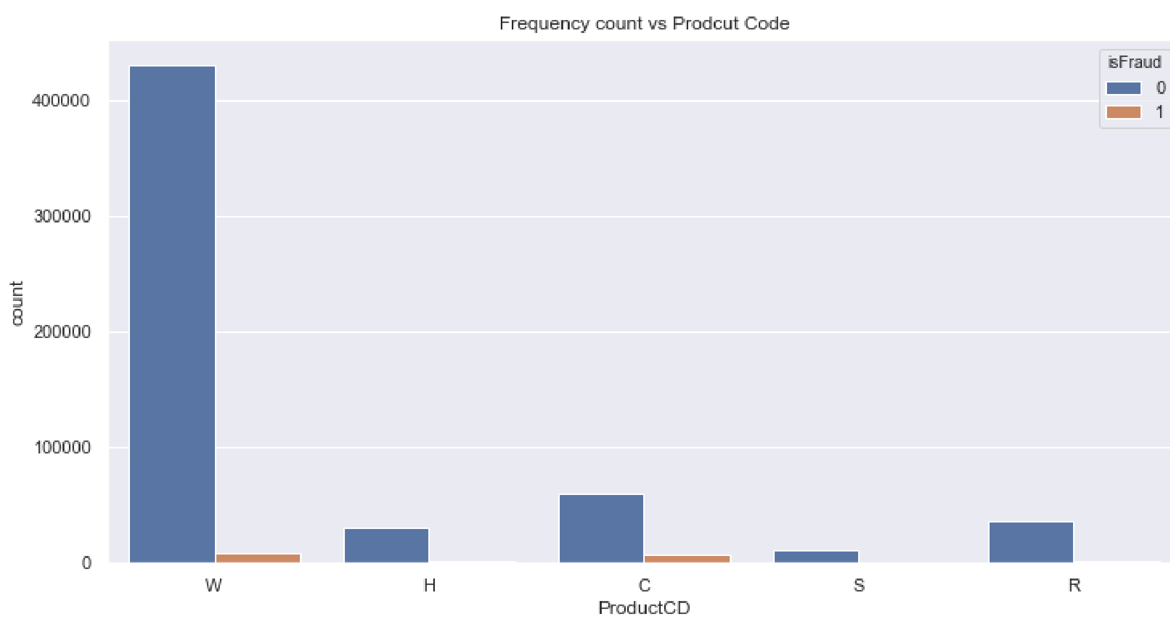
Out[1295]: `Text(0.5,1,'Frequency count vs Transaction Amount')`



Taking the log of the transaction amounts shows a proper distribution of transaction amounts with the frequency of transactions. We can see that a majority of transaction amounts are around 100 amount and outliers can be seen lesser than 10 and greater than 1000.

In [1296]:
```
# Distribution of ProdcutCD
plt.figure(figsize= (12,6))
sns.countplot(x="ProductCD", hue="isFraud", data=dataset)
plt.title("Frequency count vs Prodcut Code")
```

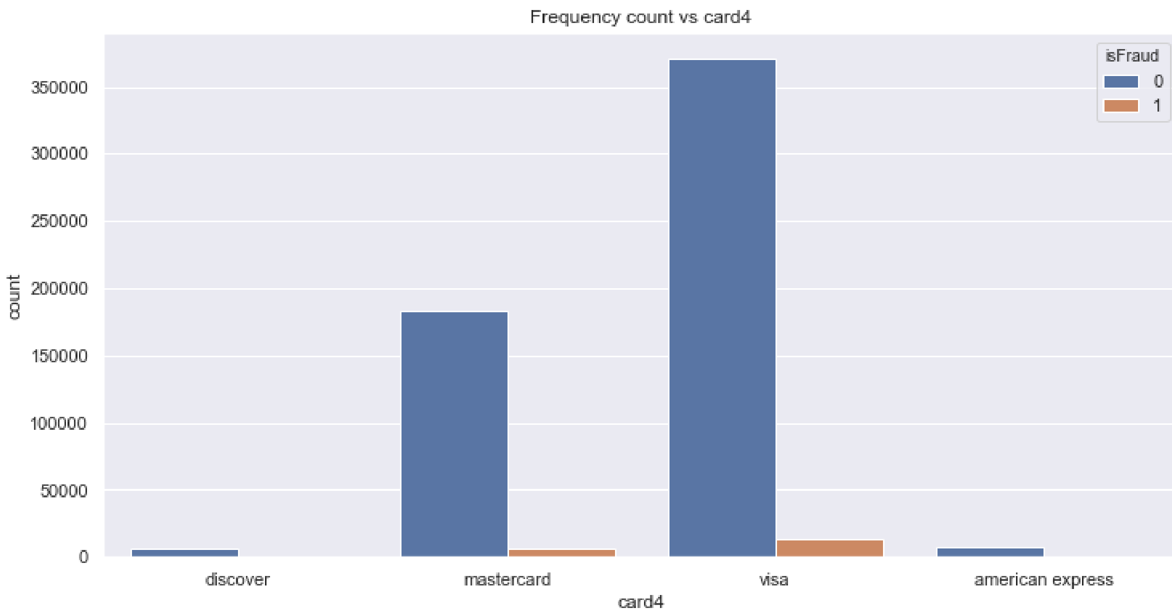
Out[1296]: `Text(0.5,1,'Frequency count vs Prodcut Code')`



The plot shows that products with code C are the most vulnerable products for fraudulent transactions.

In [1297]:
```
# Distribution of card4
plt.figure(figsize= (12,6))
sns.countplot(x="card4", hue="isFraud", data=dataset)
plt.title("Frequency count vs card4")
```

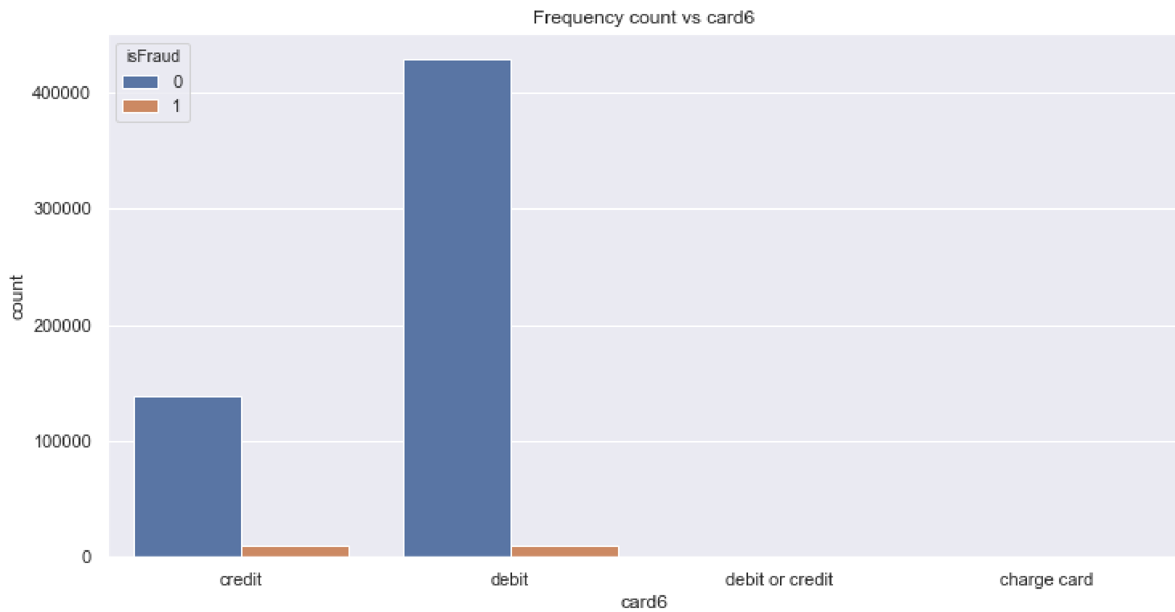
Out[1297]:
```
Text(0.5,1,'Frequency count vs card4')
```



The plot shows that 'Visa' cards are the most vulnerable cards when it comes to fraudulent transactions.

In [1298]:
```
# Distribution of card6
plt.figure(figsize= (12,6))
sns.countplot(x="card6", hue="isFraud", data=dataset)
plt.title("Frequency count vs card6")
```

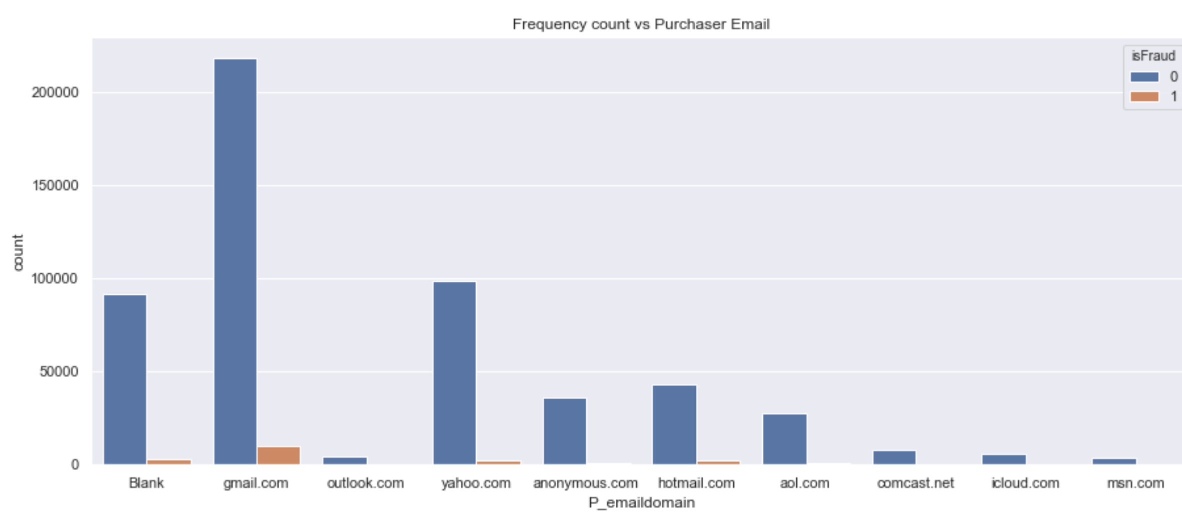
Out[1298]: `Text(0.5,1,'Frequency count vs card6')`



The plot shows that credit cards have a slightly more chance to encounter a fraudulent transaction than the debit card.

In [1299]:
```
# Distribution of P_emaildomain
plt.figure(figsize= (15,6))
dataset_copy = dataset.copy()
pick=dataset_copy["P_emaildomain"].value_counts()[:10].index
temp=dataset_copy.loc[dataset["P_emaildomain"].isin(pick)]
sns.countplot(x="P_emaildomain",hue="isFraud",data=temp)
plt.title("Frequency count vs Purchaser Email")
```

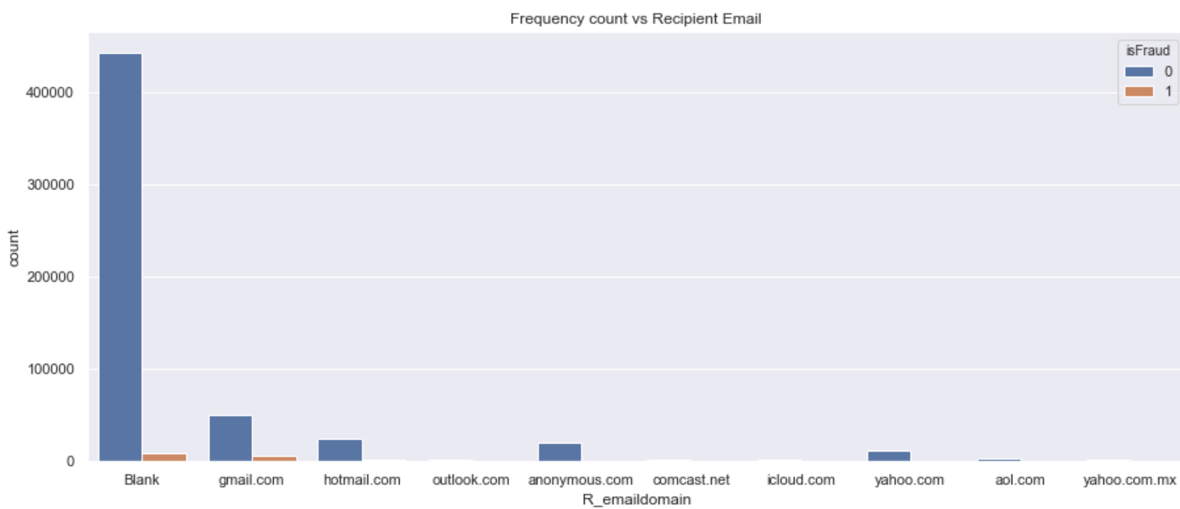
Out[1299]: `Text(0.5,1,'Frequency count vs Purchaser Email')`



The plot shows that Gmail accounts for purchasers are the most susceptible to fraudulent transactions than any other domain.

In [1300]:
```
# Distribution of R_emaildomain
plt.figure(figsize= (15,6))
dataset_copy = dataset.copy()
pick=dataset_copy["R_emaildomain"].value_counts()[:10].index
temp=dataset_copy.loc[dataset["R_emaildomain"].isin(pick)]
sns.countplot(x="R_emaildomain",hue="isFraud",data=temp)
plt.title("Frequency count vs Recipient Email")
```

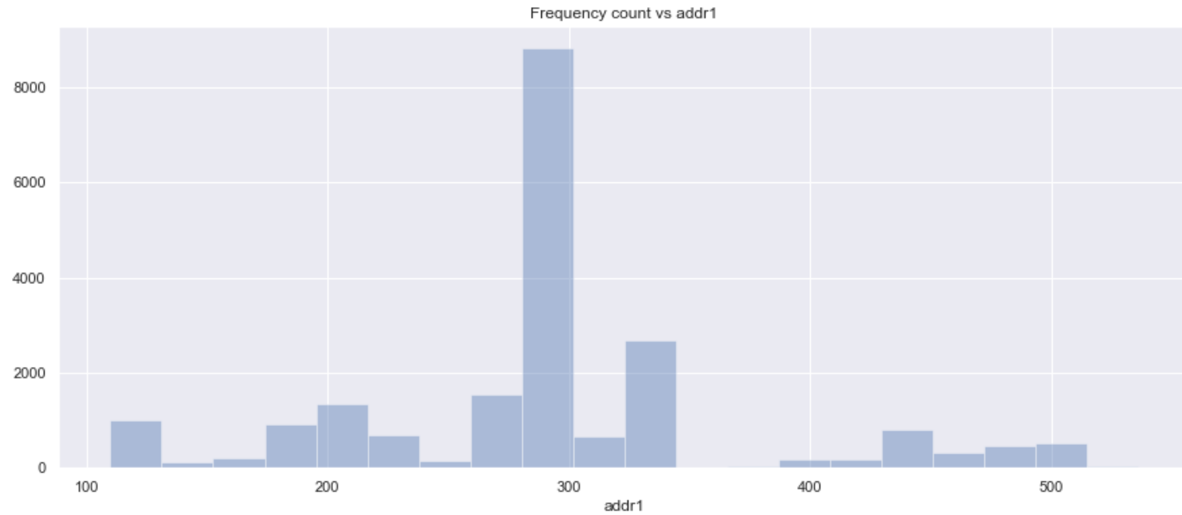
Out[1300]:
```
Text(0.5,1,'Frequency count vs Recipient Email')
```



The plot shows that Gmail accounts for recipients are the most susceptible to fraudulent transactions than any other domain.

In [1301]:
```
# Distribution of addr1
# Fraud transactions
plt.figure(figsize= (15,6))
sns.distplot(trans_fraud.addr1, kde = False, rug = False, bins = 20)
plt.title("Frequency count vs addr1")
```

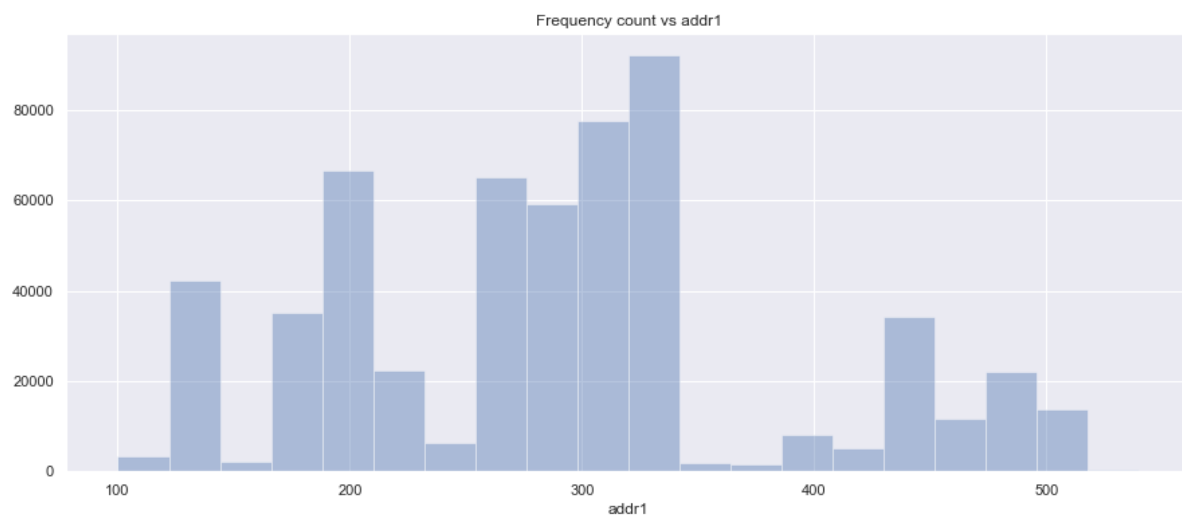
Out[1301]: Text(0.5,1,'Frequency count vs addr1')



Address1 does not show ample correlation with the frequency of fraudulent transactions.

In [1302]:
```
# Distribution of addr1
# Not fraud transactions
plt.figure(figsize= (15,6))
sns.distplot(trans_not_fraud.addr1, kde = False, rug = False, bins = 20)
plt.title("Frequency count vs addr1")
```

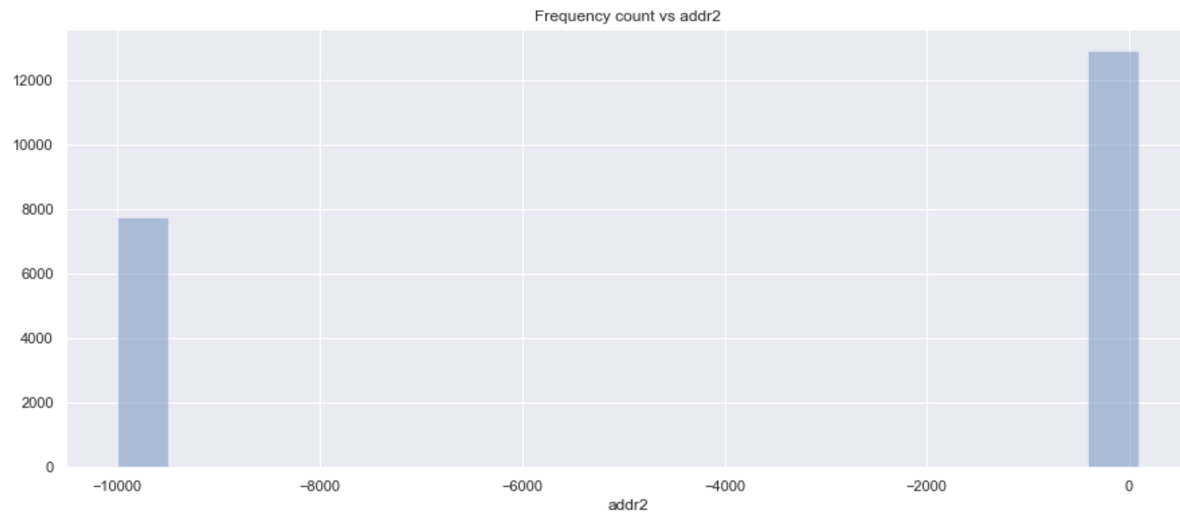
Out[1302]: Text(0.5,1,'Frequency count vs addr1')



Address1 does not show ample correlation with the frequency of non fraudulent transactions.

In [1303]:
```
# Distribution of addr2
# Fraud transactions
plt.figure(figsize= (15,6))
sns.distplot(trans_fraud.addr2, kde = False, rug = False, bins = 20)
plt.title("Frequency count vs addr2")
```

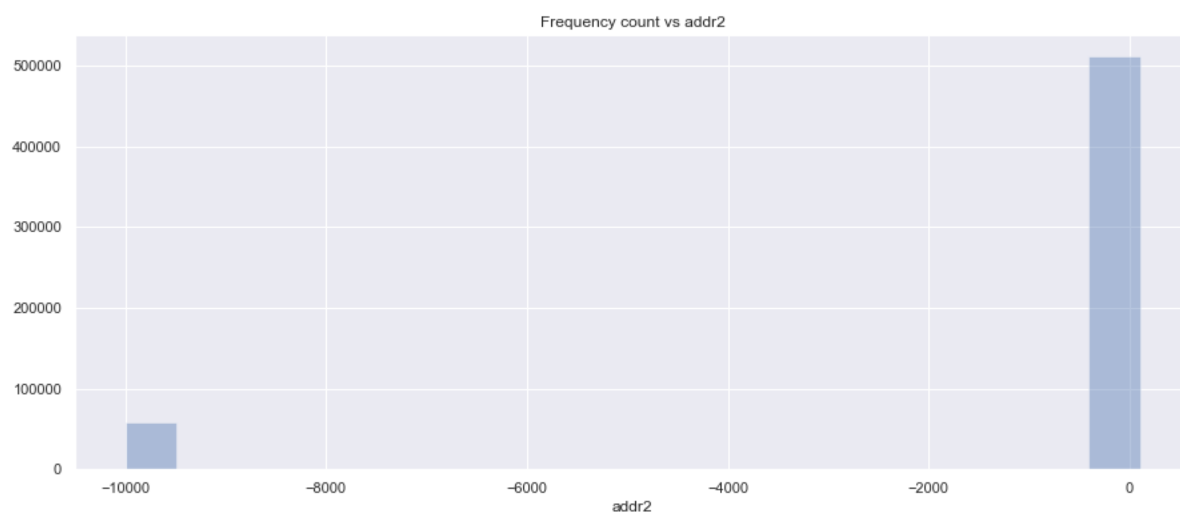
Out[1303]: Text(0.5,1,'Frequency count vs addr2')



Address2 does not show enough correlation with the frequency of fraudulent transactions.

In [1304]:
```
# Distribution of addr2
# Non Fraud transactions
plt.figure(figsize= (15,6))
sns.distplot(trans_not_fraud.addr2, kde = False, rug = False, bins = 20)
plt.title("Frequency count vs addr2")
```

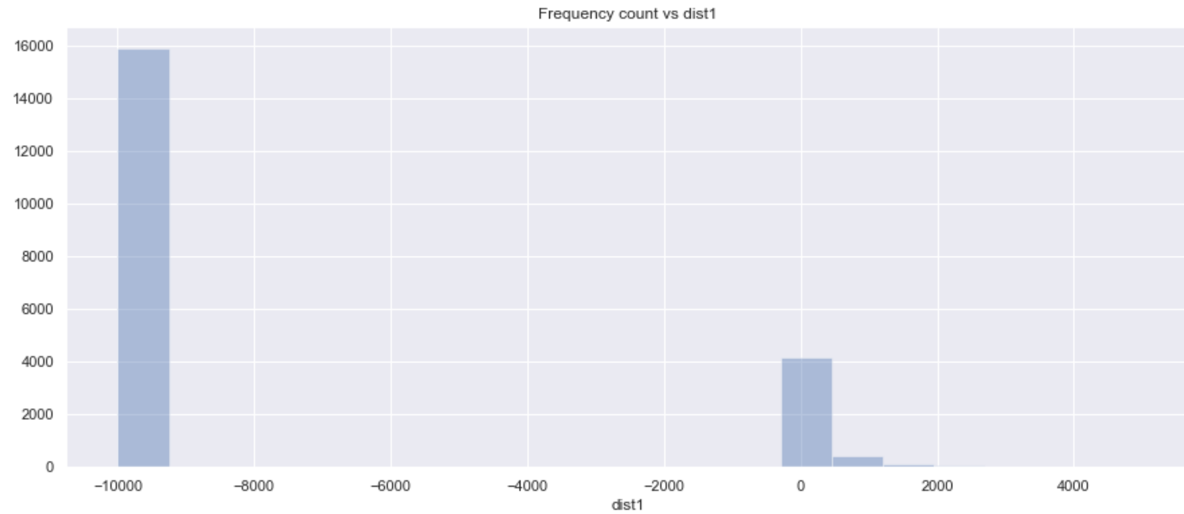
Out[1304]: Text(0.5,1,'Frequency count vs addr2')



Address2 does not show enough correlation with the frequency of non fraudulent transactions.

In [1305]:
```
# Distribution of dist1
# Fraudulent transactions
plt.figure(figsize= (15,6))
sns.distplot(trans_fraud.dist1, kde = False, rug = False, bins = 20)
plt.title("Frequency count vs dist1")
```

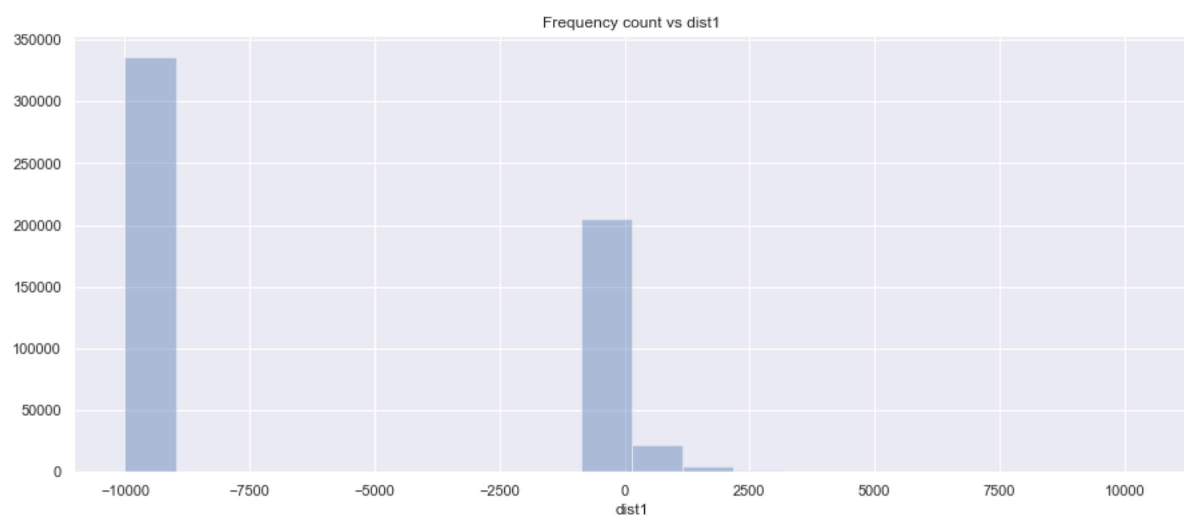
Out[1305]: Text(0.5,1,'Frequency count vs dist1')



Dist1 does not show enough correlation with the frequency of fraudulent transactions.

In [1306]:
```
# Distribution of dist1
# Non Fraudulent transactions
plt.figure(figsize= (15,6))
sns.distplot(trans_not_fraud.dist1, kde = False, rug = False, bins = 20)
plt.title("Frequency count vs dist1")
```

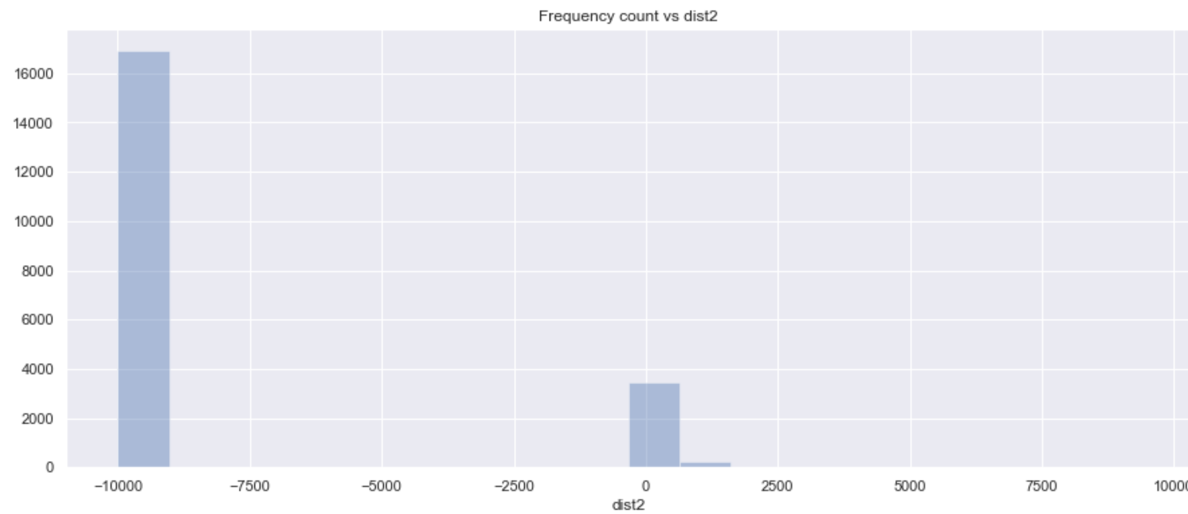
Out[1306]: Text(0.5,1,'Frequency count vs dist1')



Dist1 does not show enough correlation with the frequency of non fraudulent transactions.

In [1307]:
```
# Distribution of dist2
# Fradulent transactions
plt.figure(figsize= (15,6))
sns.distplot(trans_fraud.dist2, kde = False, rug = False, bins = 20)
plt.title("Frequency count vs dist2")
```

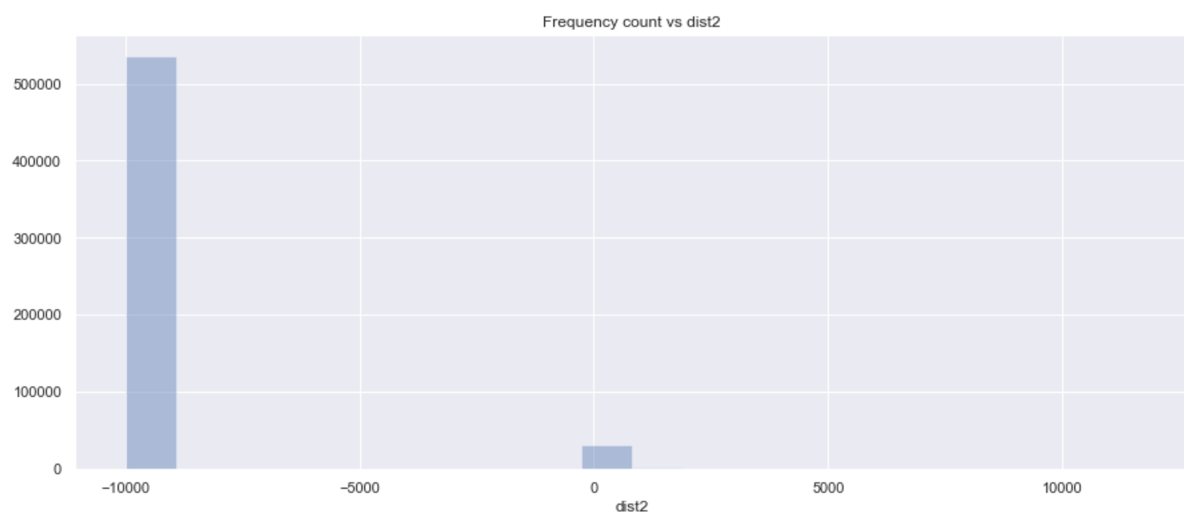
Out[1307]: Text(0.5,1,'Frequency count vs dist2')



Dist2 does not show enough correlation with the frequency of fraudulent transactions.

In [1308]:
```
# Distribution of dist2
# Non fraudulent transactions
plt.figure(figsize= (15,6))
sns.distplot(trans_not_fraud.dist2, kde = False, rug = False, bins = 20)
plt.title("Frequency count vs dist2")
```

Out[1308]: Text(0.5,1,'Frequency count vs dist2')

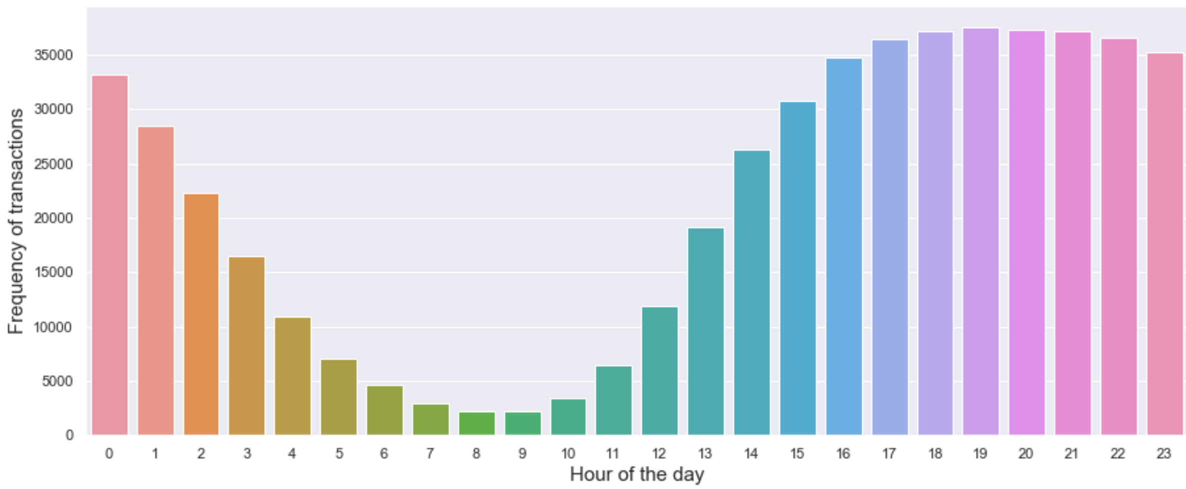


Dist2 does not show enough correlation with the frequency of non fraudulent transactions.

Part 2 - Transaction Frequency

In [1309]:
```
# Modifying Transaction Date
plt.figure(figsize= (15,6))
dataset_copy = dataset.copy()
pick=dataset_copy["addr2"].value_counts()[:10].index
most_freq_country_code = pick[0]

trans_most_freq_country = dataset_copy[dataset_copy["addr2"] == most_freq_country_code]
sns.countplot(x=trans_most_freq_country.hours, data=trans_most_freq_country)
plt.xlabel('Hour of the day', size= 15)
plt.ylabel('Frequency of transactions', size= 15)
```

Out[1309]:
```
Text(0,0.5,'Frequency of transactions')
```

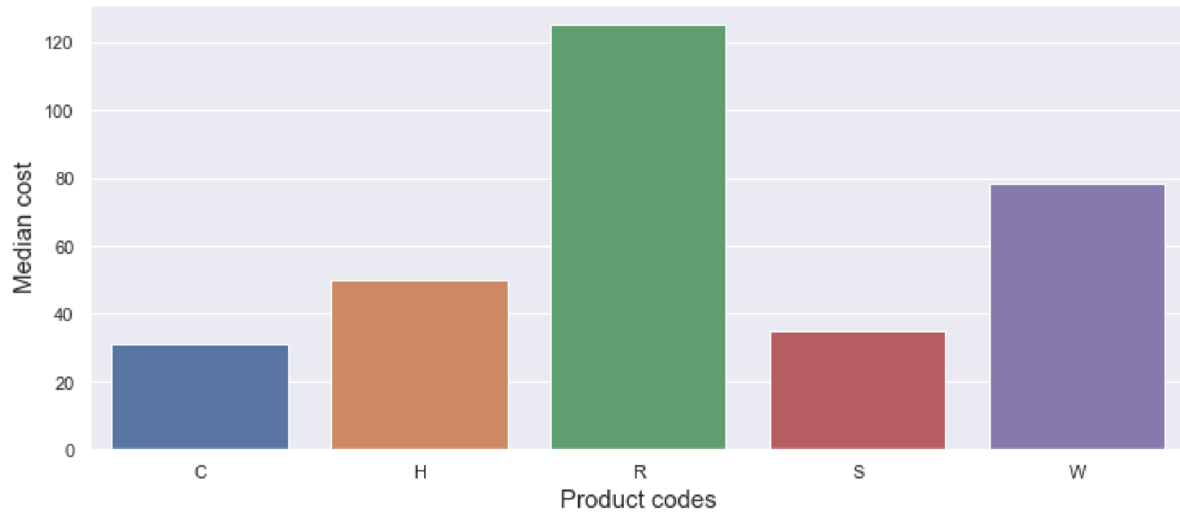


The plot shows that for the country with code 87.0 has maximum transactions during late night after 12 am which get considerably reduced as and when the morning approaches around 9 am. Then the frequency of transactions increase from 11 am and at a greater rate till midnight.

Part 3 - Product Code

In [1310]:
```
# TODO: code to analyze prices for different product codes
plt.figure(figsize= (12,5))
dataset_copy = dataset.copy()
median_cost = dataset_copy.groupby("ProductCD")["TransactionAmt"].median
().rename_axis("ProductCD").reset_index(name="Median")

# Barplot between ProductCD and Median
sns.barplot(x="ProductCD",y="Median",data=median_cost)
plt.xlabel('Product codes', size= 15)
plt.ylabel('Median cost', size= 15)
```

Out[1310]:
```
Text(0,0.5,'Median cost')
```

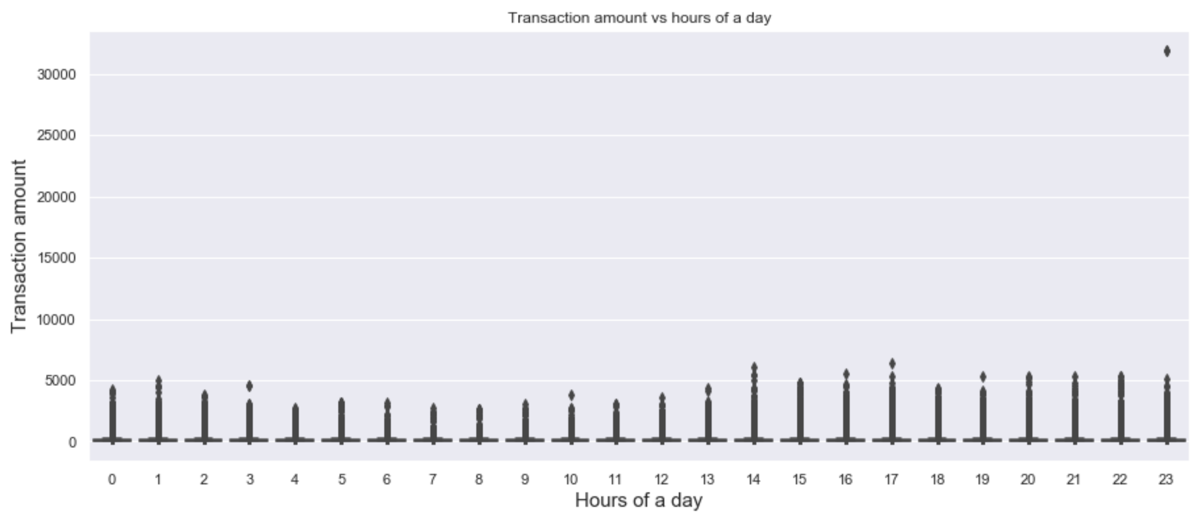


According to the plot, the most expensive product is type R and the cheapest product is type C. This is because the median of transaction amount is greatest for type R and least for type C. This means that at least 50% of product R are more costly than any other product type making it the most expensive and C as the least expensive.

Part 4 - Correlation Coefficient

In [1311]:
```
plt.figure(figsize= (15,6))
sns.boxplot(x="hours",y="TransactionAmt",data=dataset)
plt.xlabel('Hours of a day', size= 15)
plt.ylabel('Transaction amount', size= 15)

plt.title('Transaction amount vs hours of a day')
print  color.BOLD + "\n\n\t\t\tPlotting hours of the day with the transa
ction amount for the entire dataset" + color.END
```

Plotting hours of the day with the transaction amount for the entire dataset

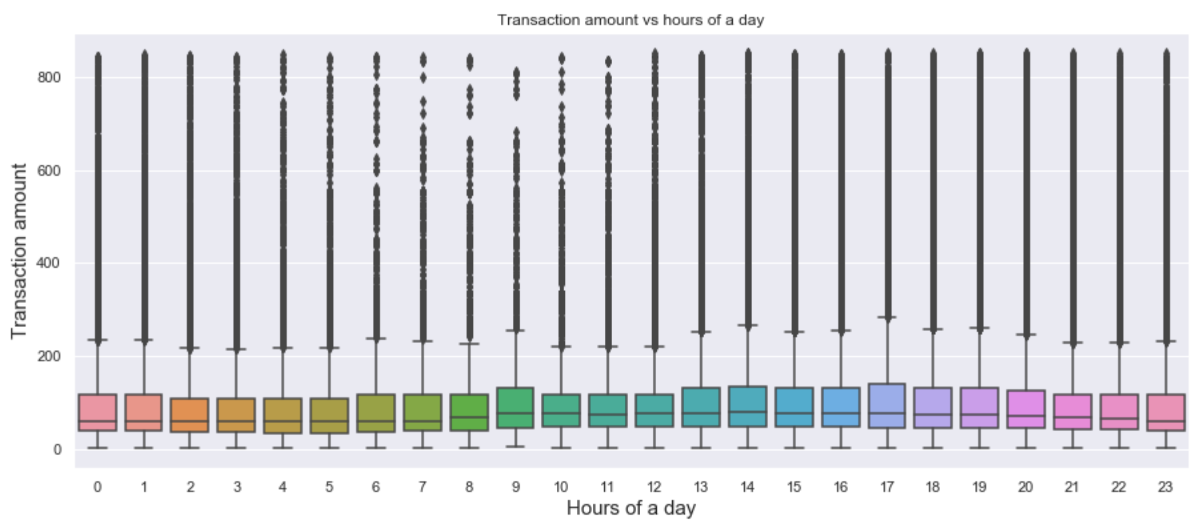


Issue -- Since transaction amounts are not pruned, the boxed values for a given hours are too small to be viewed and understood properly. Hence, the transaction amount column must be pruned.

In [1313]:
```
### Removing outliers from transaction amounts (>200 are removed)
plt.figure(figsize= (15,6))
dataset_copy = dataset.copy()
# Calculating the Z-score and removing outliers based on the selected th
reshold
col_zscore =  'trans_amt_zscore'
dataset_copy[col_zscore] = (dataset_copy['TransactionAmt'] - dataset_cop
y['TransactionAmt'].mean())/dataset_copy['TransactionAmt'].std(ddof=0)

# Update the values of TransactionAmt to max if it lies outside 3 sigma
zscore_dataset = dataset_copy.copy()
dataset_greater_3sigma = zscore_dataset[zscore_dataset['trans_amt_zscor
e'] > 3]
col_index = dataset_greater_3sigma.index
zscore_dataset.loc[col_index]['TransactionAmt'] = -9999

size_dataset_zscore = zscore_dataset[zscore_dataset['trans_amt_zscore']
< 3].shape[0]
percent_needed_trans_amt = (size_dataset_zscore * 100)/size_dataset
print "We chose the threshold for Z-score as 3 since",color.BOLD, percen
t_needed_trans_amt,  color.END, "% data for transaction amount can be fo
und within this threshold."

sns.boxplot(x="hours",y="TransactionAmt",data=zscore_dataset[zscore_data
set['trans_amt_zscore'] < 3])
plt.xlabel('Hours of a day', size= 15)
plt.ylabel('Transaction amount', size= 15)
plt.title('Transaction amount vs hours of a day')
```

We chose the threshold for Z-score as 3 since **98** % data for transaction amount can be found within this threshold.

Out[1313]: Text(0.5,1,'Transaction amount vs hours of a day')



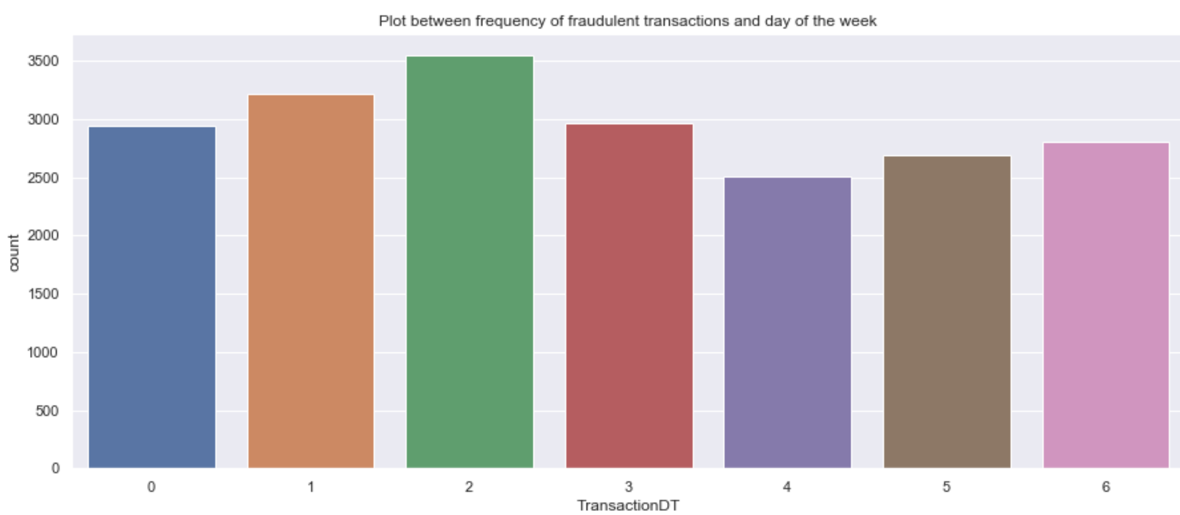
In [1314]:
```
# Calculating median of transaction amounts and finding the correlation
 with the hours of a day
medians = dataset_greater_3sigma.groupby("hours", as_index=False)["Trans
actionAmt"].median()
print "Pearson correlation coefficient : ", medians.corr(method='pearso
n')["TransactionAmt"][0]
print "Spearman correlation coefficient : ", medians.corr(method='spearm
an')["TransactionAmt"][0]
```

```
Pearson correlation coefficient :  0.5203293139242384
Spearman correlation coefficient :  0.49826265223266003
```

Part 5 - Interesting Plot

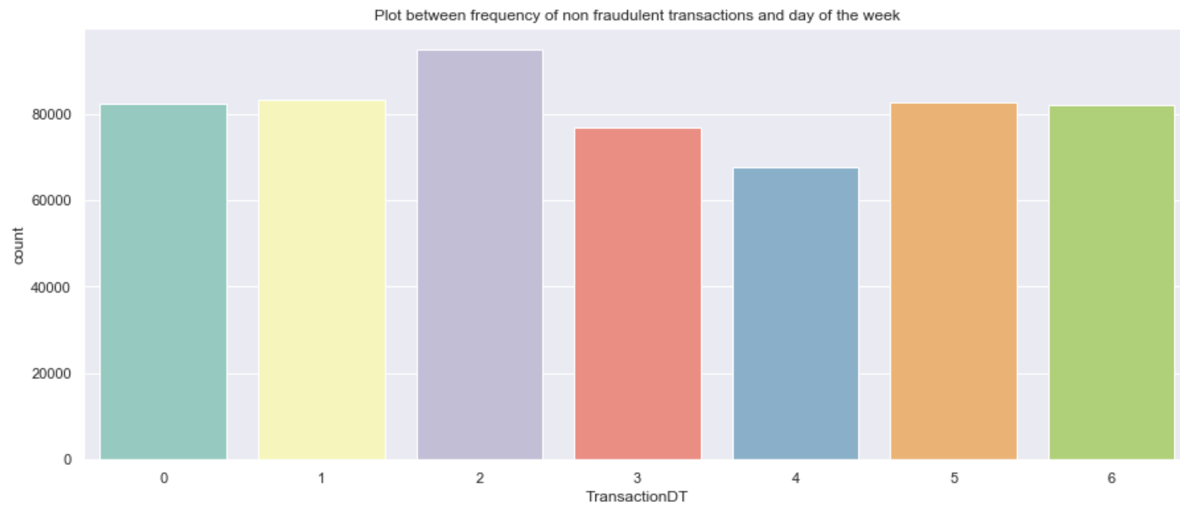
In [1315]:
```
plt.figure(figsize= (15,6))
dataset_copy = dataset.copy()
sns.countplot(x=trans_fraud.TransactionDT.dt.dayofweek, data=trans_fraud
)
plt.title("Plot between frequency of fraudulent transactions and day of
 the week")
```

Out[1315]:
```
Text(0.5,1,'Plot between frequency of fraudulent transactions and day o
f the week')
```



In [1316]:
```
plt.figure(figsize= (15,6))
sns.countplot(x=trans_not_fraud.TransactionDT.dt.dayofweek, data=trans_n
ot_fraud, palette="Set3")
plt.title("Plot between frequency of non fraudulent transactions and day
of the week")
```

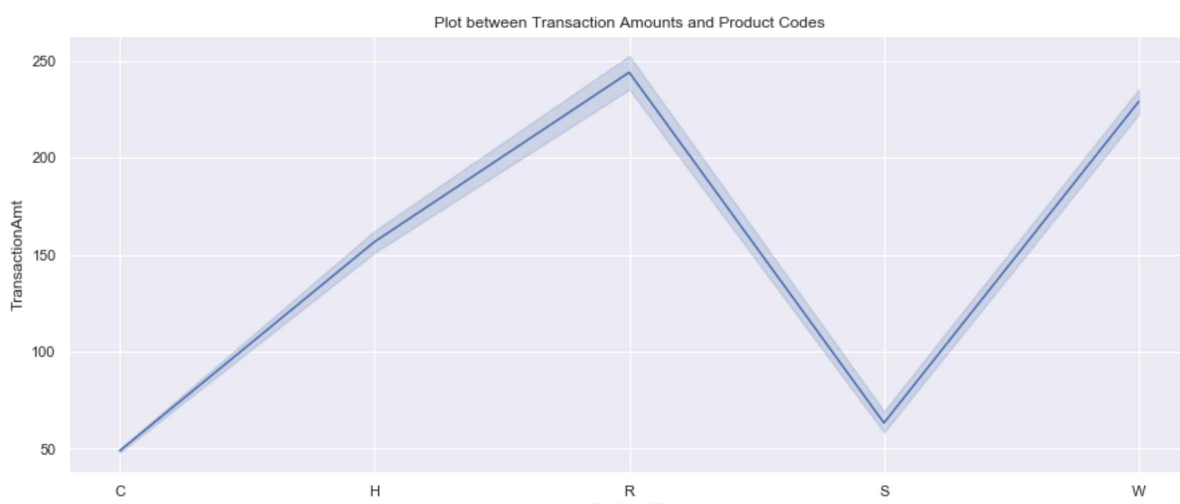
Out[1316]:
```
Text(0.5,1,'Plot between frequency of non fraudulent transactions and d
ay of the week')
```



We can see that on day 2 of the week, the number of transactions are all time high. It is during this day that the frequency of fraudulent transactions are at peak. The frequency of fraudulent transactions increases from day 0 to day 2 and then increases from day 4 to day 6.

In [1317]:
```
plt.figure(figsize= (15,6))
sns.lineplot(x=trans_fraud.ProductCD, y=trans_fraud.TransactionAmt, data
=trans_fraud)
plt.title("Plot between Transaction Amounts and Product Codes")
print "The plot shows that Product R was the most fraud prone product wi
th max transaction cost of around 250 "
```

```
The plot shows that Product R was the most fraud prone product with max
transaction cost of around 250
```



In [1319]:

```
plt.figure(figsize= (15,6))
sns.lineplot(x=trans_not_fraud.ProductCD, y=trans_not_fraud.TransactionA
mt, data=trans_not_fraud)
plt.title("Plot between Transaction Amounts and Product Codes")
print "The plot shows that Product R was the lest fraud prone product wi
th max transaction cost of around 160 "
```

```
The plot shows that Product R was the lest fraud prone product with max
transaction cost of around 160
```

Part 6 - Prediction Model

In [1320]:

```
# By deduction done at the top
filtered_dataset = zscore_dataset.copy()
filtered_dataset['addr2'] = np.where(filtered_dataset['addr2']!=87.0, 'B
lank', filtered_dataset['addr2'])
```

In [1321]:
```
# Separate majority and minority classes
filtered_dataset_majority = filtered_dataset[filtered_dataset.isFraud==0
]
filtered_dataset_minority = filtered_dataset[filtered_dataset.isFraud==1
]

# Upsample minority class
minority_upsampled = resample(filtered_dataset_minority,
                                 replace=True,     # sample with replace
ment
                                 n_samples=len(filtered_dataset_majority
),    # to match majority class
                                 random_state=123) # reproducible result
s

# Combine majority class with upsampled minority class
filtered_dataset_upsampled = pd.concat([filtered_dataset_majority, minor
ity_upsampled])

# Display new class counts
filtered_dataset_upsampled.isFraud.value_counts()
print "Minority data set upsampled to 569877 rows equalling that of the
 majority dataset."
```

```
Minority data set upsampled to 569877 rows equalling that of the majori
ty dataset.
```

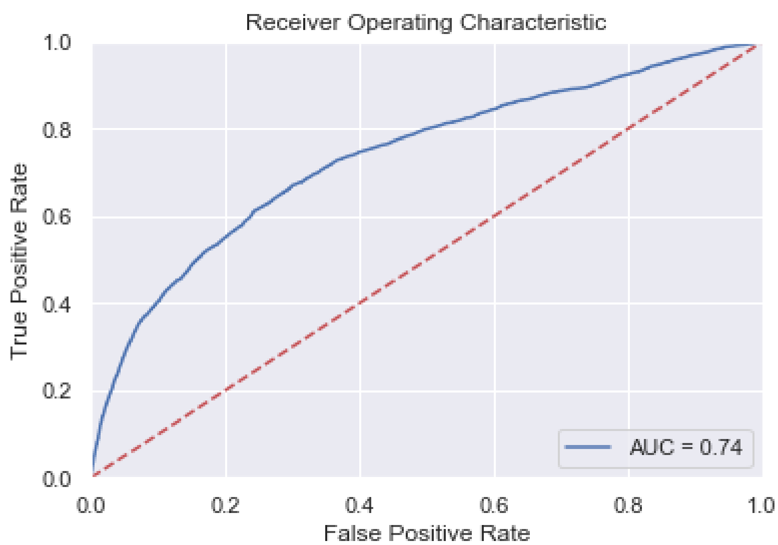
In [1322]:
```
y = filtered_dataset['isFraud']

# Splitting the data into 70% training and 30% test data
test = filtered_dataset.copy()
test = test.drop(['TransactionID','dist1','dist2','isFraud','addr1','Tra
nsactionDT','trans_amt_zscore','DeviceInfo','R_emaildomain'], axis=1)
```

In [1323]:
```
one_hot_vectors = pd.get_dummies(test)
X = one_hot_vectors

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

In [1324]:
```
# TODO: code for your final model
clf = LogisticRegression(solver='lbfgs')
clf.fit(X_train, y_train)
preds = clf.predict(X_test)

print("Predicting if the talk is related to Technology using a Logistic
 Regression classifier:")
print("\nThe classifier's accuracy is... *drum roll*\n\n%s percent!\n" %
round(100*accuracy_score(y_test, preds), 2))
```

```
Predicting if the talk is related to Technology using a Logistic Regres
sion classifier:

The classifier's accuracy is... *drum roll*

96.52 percent!
```

In [1325]:
```
# Plotting the ROC curve
probs = clf.predict_proba(X_test)
preds = probs[:,1]
fpr, tpr, thresholds = metrics.roc_curve(y_test, preds)
roc_auc = metrics.auc(fpr,tpr)

plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



In [1326]:
```
dataset_test_copy = dataset_test.copy()
dataset_test_copy['addr2'] = np.where(dataset_test_copy['addr2']!=87.0,
'Blank', dataset_test_copy['addr2'])
trans_id = dataset_test_copy.TransactionID
test2 = dataset_test_copy.drop(['TransactionID','dist1','dist2','isFrau
d','addr1','TransactionDT','DeviceInfo','R_emaildomain'], axis=1)

X_test2 = pd.get_dummies(test2)
```

In [1327]:
```
probs_test = clf.predict_proba(X_test2)
preds = probs_test[:,1]
```

In [1328]:
```
data = {'TransactionID':trans_id, 'isFraud':preds}
result_df = pd.DataFrame(data)

result_df['isFraud'] =result_df['isFraud'].apply(lambda x: round(x, 3))
result_df.to_csv('Data/ieee-fraud-detection/result2.csv', index=False)
```

Part 7 - Final Result

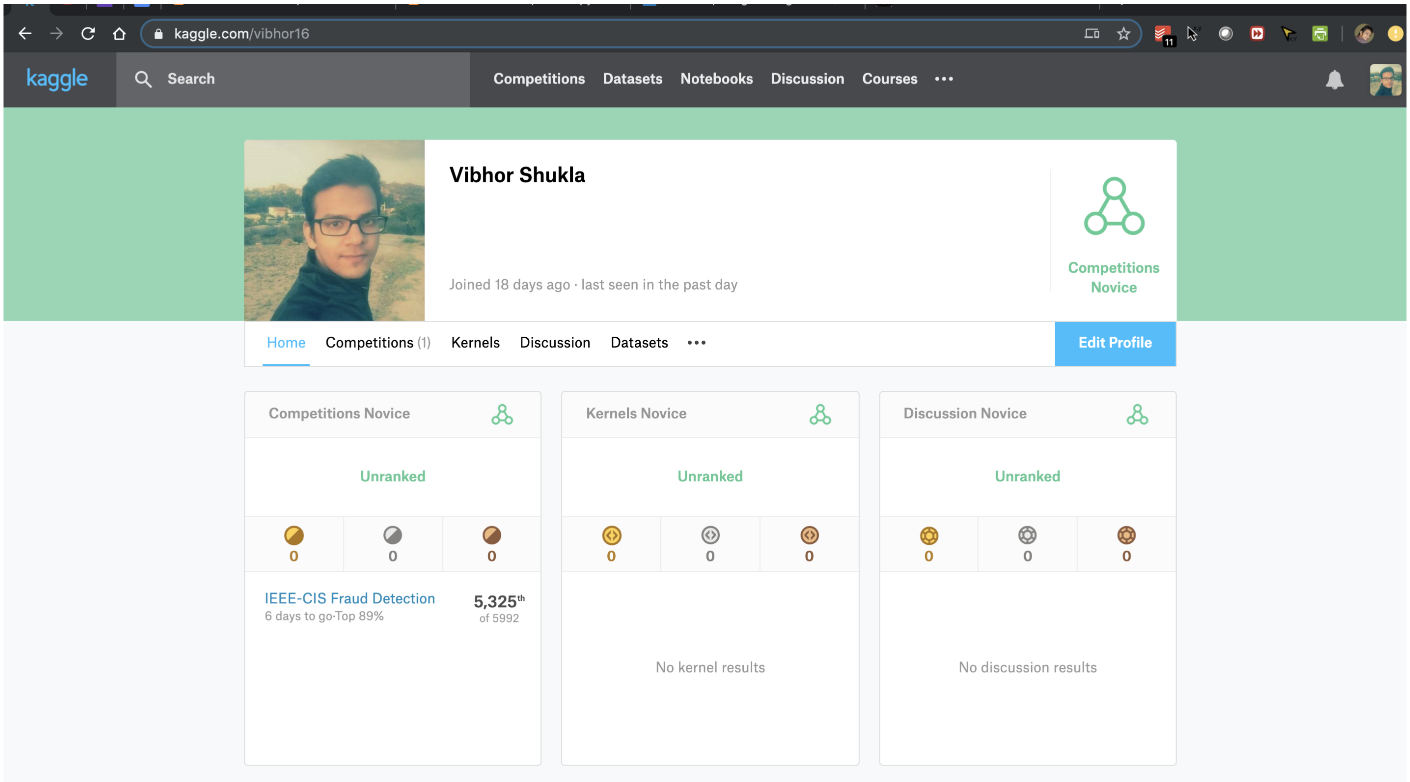
Report the rank, score, number of entries, for your highest rank. Include a snapshot of your best score on the leaderboard as confirmation. Be sure to provide a link to your Kaggle profile. Make sure to include a screenshot of your ranking. Make sure your profile includes your face and affiliation with SBU.

Kaggle Link: [https://www.kaggle.com/vibhor16 (https://www.kaggle.com/vibhor16)](https://www.kaggle.com/vibhor16)

Highest Rank: 5325

Score: 0.8017

Number of entries: 2


kaggle.com/vibhor16
kaggle
Search
Competitions
Datasets
Notebooks
Discussion
Courses
Vibhor Shukla
Joined 18 days ago · last seen in the past day
Competitions Novice
Home
Competitions (1)
Kernels
Discussion
Datasets
Edit Profile
Competitions Novice
Unranked
0
0
0
IEEE-CIS Fraud Detection
6 days to go·Top 89%
5,325th
of 5992
Kernels Novice
Unranked
0
0
0
No kernel results
Discussion Novice
Unranked
0
0
0
No discussion results