Drive Connect

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Libraries Load

```
pip install shap
```

```
Collecting shap
  Downloading https://files.pythonhosted.org/packages/b9/f4/c5b95cddae15be80f8e58b25edc
     |████████████████████████████████| 358kB 13.2MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from sh
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from sh
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from s
Requirement already satisfied: tqdm>4.25.0 in /usr/local/lib/python3.7/dist-packages (f
Collecting slicer==0.0.7
  Downloading https://files.pythonhosted.org/packages/78/c2/b3f55dfdb8af9812fdb9baf70ca
Requirement already satisfied: numba in /usr/local/lib/python3.7/dist-packages (from sh
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: llvmlite<0.35,>=0.34.0.dev0 in /usr/local/lib/python3.7/
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from
Building wheels for collected packages: shap
  Building wheel for shap (setup.py) ... done
  Created wheel for shap: filename=shap-0.39.0-cp37-cp37m-linux_x86_64.whl size=491627
  Stored in directory: /root/.cache/pip/wheels/15/27/f5/a8ab9da52fd159aae6477b5ede6eaae
Successfully built shap
Installing collected packages: slicer, shap
Successfully installed shap-0.39.0 slicer-0.0.7
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set(color_codes=True)
from itertools import combinations
from sklearn.preprocessing import KBinsDiscretizer
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier
from sklearn.model_selection import StratifiedKFold
```

```
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import roc_auc_score,auc,roc_curve,accuracy_score
import shap
```

Data Load

```
iginal=pd.read_csv(filepath_or_buffer='/content/drive/MyDrive/Colab Notebooks/PayPal/travel i
_original.copy()
```

```
df.shape
```

```
(63326, 11)
```

Data Analysis

```
df.head()
```

	Agency	Agency Type	Distribution Channel	Product Name	Claim	Duration	Destination	Net Sales	Co
0	CBH	Travel Agency	Offline	Comprehensive Plan	No	186	MALAYSIA	-29.0	
1	CBH	Travel Agency	Offline	Comprehensive Plan	No	186	MALAYSIA	-29.0	
2	CWT	Travel Agency	Online	Rental Vehicle Excess Insurance	No	65	AUSTRALIA	-49.5	
3	CWT	Travel Agency	Online	Rental Vehicle Excess Insurance	No	60	AUSTRALIA	-39.6	
		Travel		Rental Vehicle					

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 63326 entries, 0 to 63325
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Agency                63326 non-null  object
 1   Agency Type           63326 non-null  object
 2   Distribution Channel  63326 non-null  object
 3   Product Name          63326 non-null  object
 4   Claim                 63326 non-null  object
 5   Duration              63326 non-null  int64
 6   Destination           63326 non-null  object
```

```
 7   Net Sales             63326 non-null  float64
 8   Commision (in value)  63326 non-null  float64
 9   Gender                18219 non-null  object
 10  Age                   63326 non-null  int64
dtypes: float64(2), int64(2), object(7)
memory usage: 5.3+ MB
```

Gender feature is having high volume of nulls

```
df.columns=df.columns.map(lambda x:'_'.join(x.lower().replace('(','').replace(')','').split('
```

```
df.describe(include='all').T
```

	count	unique	top	freq	mean	std	min	25%	50
agency	63326	16	EPX	35119	NaN	NaN	NaN	NaN	Nal
agency_type	63326	2	Travel Agency	45869	NaN	NaN	NaN	NaN	Nal
distribution_channel	63326	2	Online	62219	NaN	NaN	NaN	NaN	Nal
product_name	63326	26	Cancellation Plan	18630	NaN	NaN	NaN	NaN	Nal
claim	63326	2	No	62399	NaN	NaN	NaN	NaN	Nal
duration	63326	NaN	NaN	NaN	49.3171	101.792	-2	9	2
destination	63326	149	SINGAPORE	13255	NaN	NaN	NaN	NaN	Nal
net_sales	63326	NaN	NaN	NaN	40.702	48.8456	-389	18	26.5
commision_in_value	63326	NaN	NaN	NaN	9.80999	19.8044	0	0	
gender	18219	2	M	9347	NaN	NaN	NaN	NaN	Nal

Observations:

agency: >50% of data is of 'EPX' Agency

agency_type: Majority of data is of Travel Agency

distribution_channel: Data is highly skewed. Almost all of the data is of Online.

product_name: Around 25% of data is Cancellation Product

claim: Highly skewed data. Imbalanced Dataset

duration: Ideally there shouldnt be -ve values in this column. Probably its a data issue. Also data is right skewed.

destination: No observations yet. Need to analyse further.

net_sales: -ve values present and data seems to be right skewed.

commision_in_value: >50% of data having 0 commission

gender: Too many nulls

age: Seems there is a data issue. age with value 0 is not making sense. And data seems to be bit right skewed. But should be fine for now.

```
# Get skewness of features
df.skew()
```

```
duration              23.179617
net_sales              3.272373
commision_in_value     4.032269
age                    2.987710
dtype: float64
```

```
numerical_feature_list=list(df.select_dtypes(exclude='object').columns)
categorical_feature_list=list(df.select_dtypes(include='object').columns)

print("Numerical Features: ",numerical_feature_list)
print("Categorical Features: ",categorical_feature_list)
```

```
Numerical Features:  ['duration', 'net_sales', 'commision_in_value', 'age']
Categorical Features:  ['agency', 'agency_type', 'distribution_channel', 'product_name'
```

Univariate Analysis

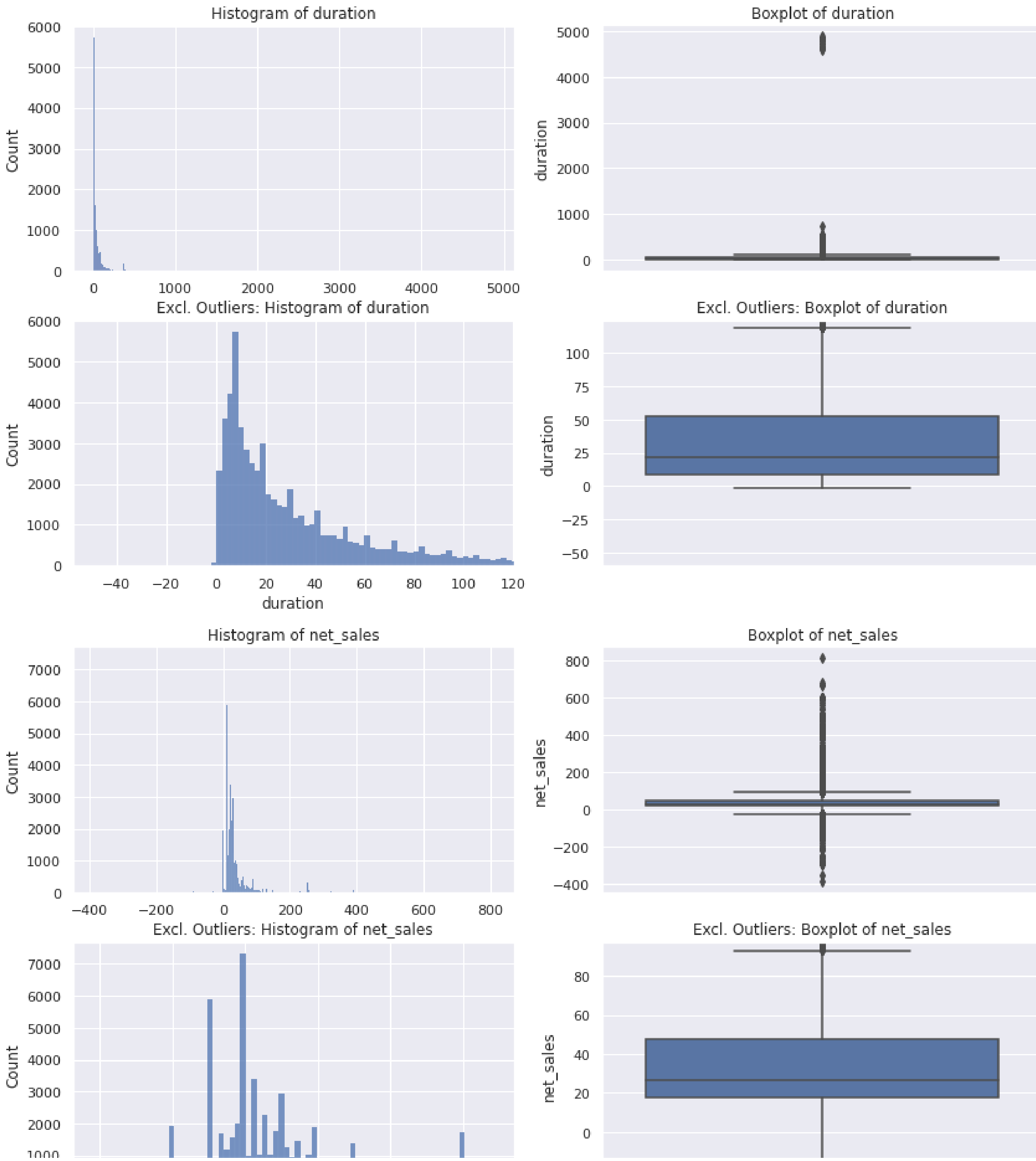
```
# Function for creating histogram and box-plot side by side for continuous variables
# Histogram plot is to help in understanding the distribution of the feature
# Box-plot is to see the data spread across quantiles and also to identify outliers if any
def numerical_feature_plot(feature,lower_whisker,upper_whisker):
    fig,ax=plt.subplots(nrows=2,ncols=2,figsize=(14,8))
    sns.histplot(df[feature],ax=ax[0][0])
    ax[0][0].set_title("Histogram of "+feature)
    ax[0][0].set_xlabel('')
    sns.boxplot(y=df[feature],ax=ax[0][1])
    ax[0][1].set_title("Boxplot of "+feature)
    sns.histplot(df[feature],ax=ax[1][0])
    ax[1][0].set_title("Excl. Outliers: Histogram of "+feature)
    ax[1][0].set_xlim([lower_whisker+(lower_whisker*0.01),upper_whisker+(upper_whisker*0.01)]
    sns.boxplot(y=df[feature],ax=ax[1][1])
    ax[1][1].set_title("Excl. Outliers: Boxplot of "+feature)
    ax[1][1].set_ylim([lower_whisker+(lower_whisker*0.05),upper_whisker+(upper_whisker*0.05)]
```

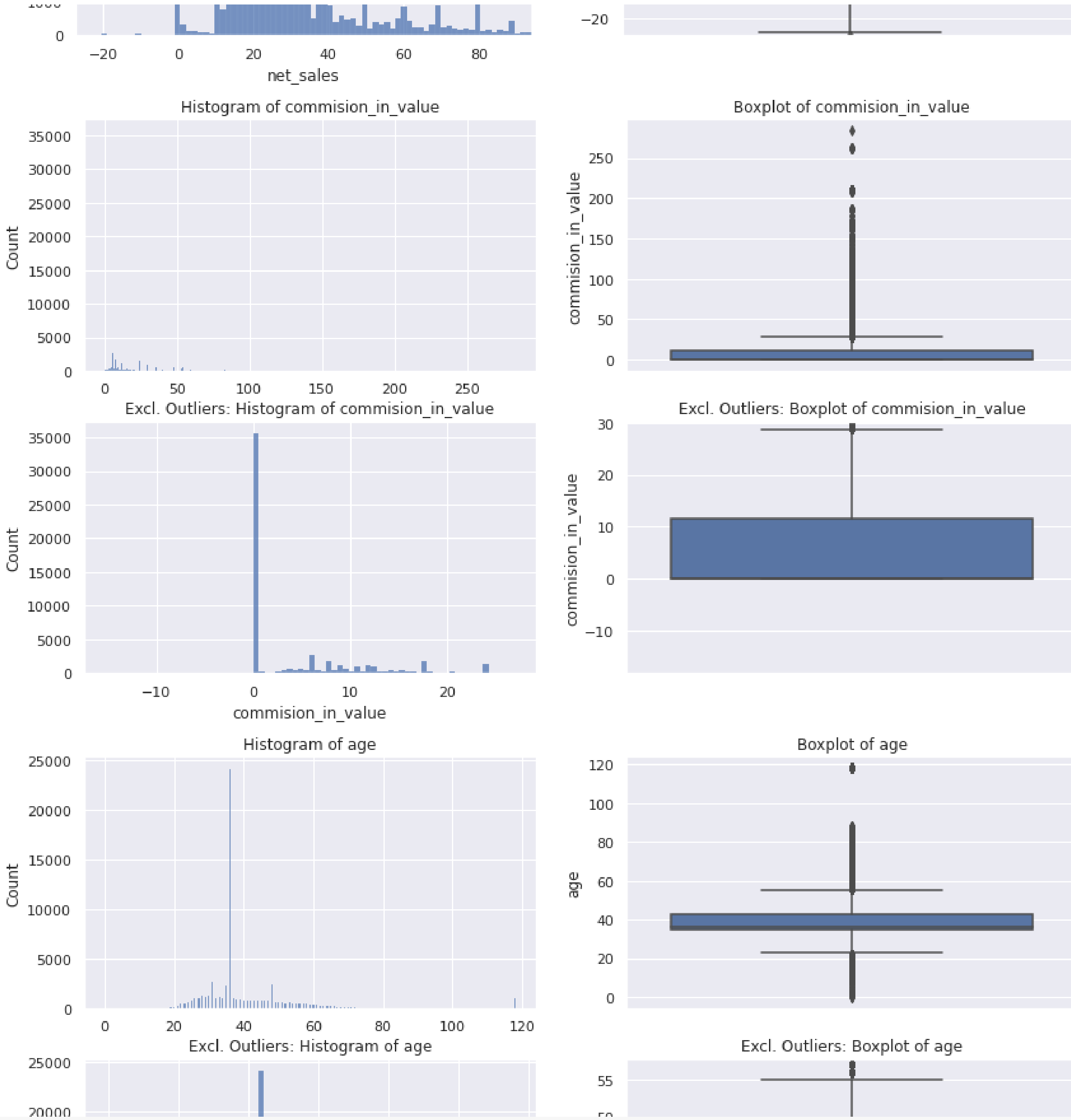
```
def numerical_feature_analysis(feature):
    feature_min=df[feature].min()
```

```
    feature_max=df[feature].max()
    feature_q1=np.quantile(df[feature],0.25)
    feature_q3=np.quantile(df[feature],0.75)
    feature_IQR=(feature_q3-feature_q1)
    feature_lower_whisker=feature_q1-(feature_IQR*1.5)
    feature_upper_whisker=feature_q3+(feature_IQR*1.5)
    numerical_feature_plot(feature,feature_lower_whisker,feature_upper_whisker)
    print("Feature: ",feature)
    if (feature_lower_whisker<feature_min) and (feature_upper_whisker>feature_max):
        print("No outliers.")
    else:
        if (feature_upper_whisker<feature_max):
            print("Outliers are present in maximum. ",feature," above ",round(feature_upper_w
        if (feature_lower_whisker>feature_min):
            print("Outliers are present in minimum. ",feature," below ",round(feature_lower_w

    print("---------------------------------------------------------------------------------
```

```
for i in numerical_feature_list:
    numerical_feature_analysis(i)
```

```
Feature:  duration
Outliers are present in maximum.  duration  above  119.0  are outliers and total are  5
-------------------------------------------------------------------------------------
Feature:  net_sales
Outliers are present in maximum.  net_sales  above  93.0  are outliers and total are  5
Outliers are present in minimum.  net_sales  below  -27.0  are outliers and total are
-------------------------------------------------------------------------------------
Feature:  commision_in_value
Outliers are present in maximum.  commision_in_value  above  28.88  are outliers and to
-------------------------------------------------------------------------------------
Feature:  age
Outliers are present in maximum.  age  above  55.0  are outliers and total are  6123
Outliers are present in minimum.  age  below  23.0  are outliers and total are  1299
-------------------------------------------------------------------------------------
```





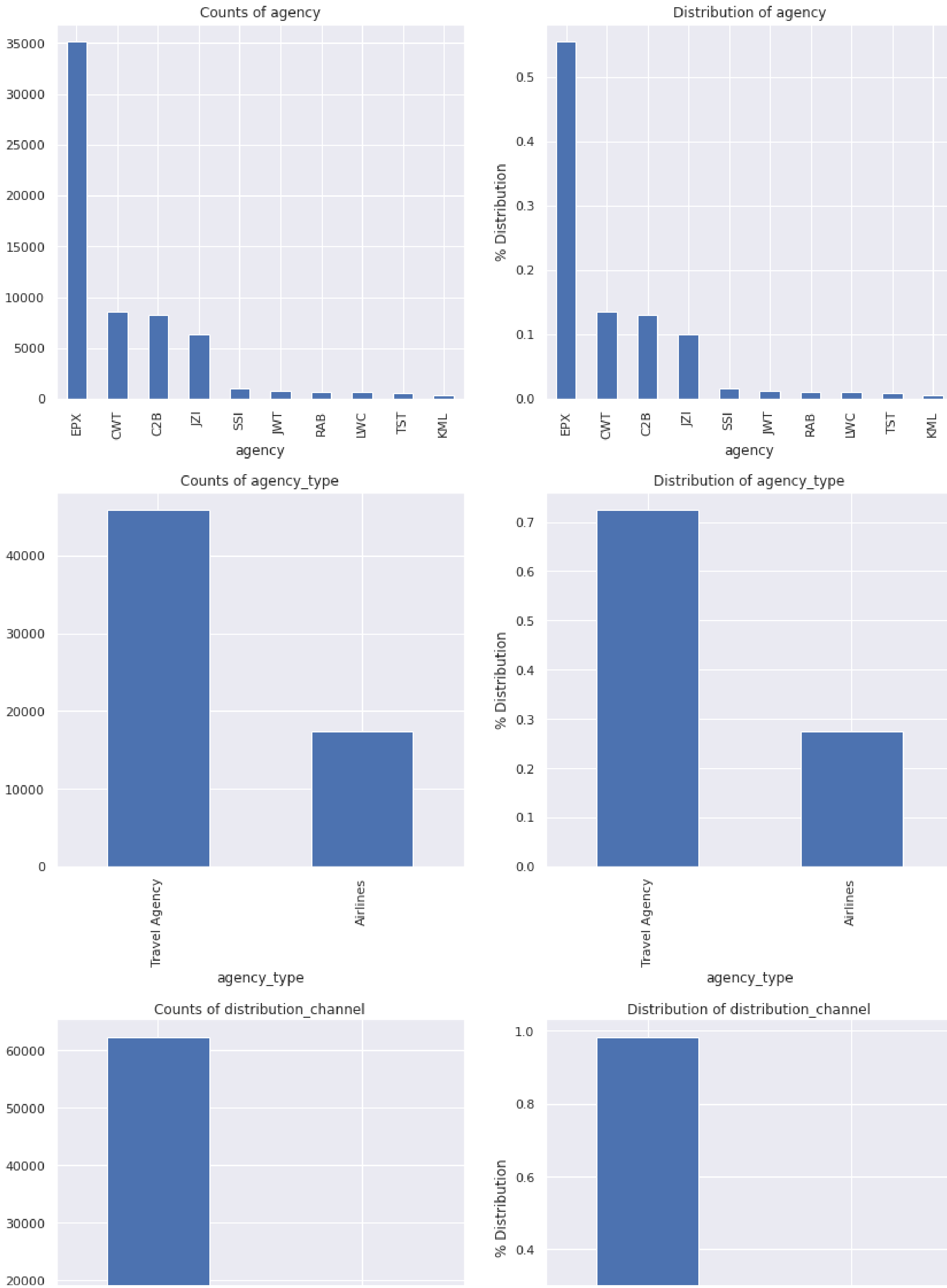

```
# Function for creating count plot and the corresponding percentage distribution plots for ca
# Count plot is to help in understanding the distribution of the feature
# % distribution plot helps us to see the % distribution of different categories for a featur
def categorical_feature_plot(feature):
    fig,ax=plt.subplots(nrows=1,ncols=2,figsize=(14,6))
    df[feature].value_counts()[:10].plot(kind='bar',ax=ax[0],rot=90,title="Counts of "+featur
    ax[0].set_xlabel(feature)
    #sns.countplot(df[feature],ax=ax[0],order=df[feature].unique())
    #ax[0].set_title("Counts of "+feature)
    #ax[0].set_xticklabels(df[feature].unique(), rotation=90)
    df[feature].value_counts(normalize=True)[:10].plot(kind='bar',ax=ax[1])
    ax[1].set_title("Distribution of "+feature)
    ax[1].set_xlabel(feature)
    ax[1].set_ylabel("% Distribution")
    ax[1].set_xticklabels(df[feature].value_counts(normalize=True).index[:10], rotation=90)
```

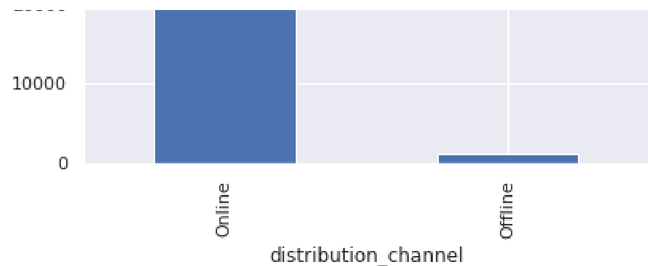
```
# Plotting for all categorical features
for i in categorical_feature_list:
    categorical_feature_plot(i)
    print("Feature:"+i+'\n')
    print("No. of unique values:"+ str(df[i].nunique())+"\n")
    print("Distribution: \n")
    print(df[i].value_counts(normalize=True).head(20))
    print("-------------------------------------------")
```

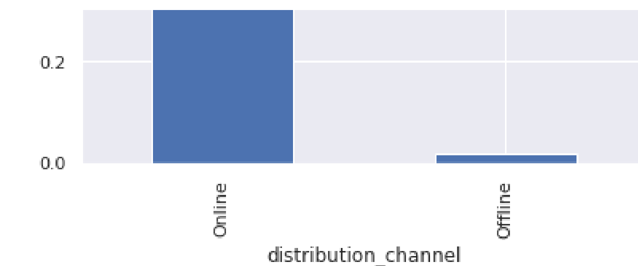
```
Feature:agency

No. of unique values:16

Distribution:

EPX    0.554575
CWT    0.135489
C2B    0.130547
JZI    0.099943
SSI    0.016676
JWT    0.011828
RAB    0.011449
LWC    0.010880
TST    0.008338
KML    0.006190
ART    0.005227
CCR    0.003064
CBH    0.001595
TTW    0.001548
CSR    0.001358
ADM    0.001295
Name: agency, dtype: float64
------------------------------------------
Feature:agency_type

No. of unique values:2

Distribution:

Travel Agency    0.724331
Airlines         0.275669
Name: agency_type, dtype: float64
------------------------------------------
Feature:distribution_channel

No. of unique values:2

Distribution:

Online     0.982519
Offline    0.017481
Name: distribution_channel, dtype: float64
------------------------------------------
Feature:product_name

No. of unique values:26

Distribution:

Cancellation Plan                      0.294192
2 way Comprehensive Plan               0.207782
Rental Vehicle Excess Insurance        0.135489
Basic Plan                             0.086363
Bronze Plan                            0.063939
1 way Comprehensive Plan               0.052601
Value Plan                             0.042873
```

```
Silver Plan                          0.035515
Annual Silver Plan                   0.022471
Ticket Protector                     0.016676
Travel Cruise Protect                0.008322
Comprehensive Plan                   0.005748
Gold Plan                            0.005559
24 Protect                           0.003900
Single Trip Travel Protect Gold      0.003221
Premier Plan                         0.003064
Annual Gold Plan                     0.003064
Single Trip Travel Protect Silver    0.002732
Annual Travel Protect Gold           0.001579
Annual Travel Protect Silver         0.001358
Name: product_name, dtype: float64
------------------------------------------
Feature:claim

No. of unique values:2

Distribution:

0    0.985361
1    0.014639
Name: claim, dtype: float64
------------------------------------------
Feature:destination

No. of unique values:149

Distribution:

SINGAPORE                    0.209314
MALAYSIA                     0.093642
THAILAND                     0.093074
CHINA                        0.075735
AUSTRALIA                    0.058333
INDONESIA                    0.054512
UNITED STATES                0.039952
PHILIPPINES                  0.039320
HONG KONG                    0.038073
INDIA                        0.035546
JAPAN                        0.032546
VIET NAM                     0.026356
KOREA, REPUBLIC OF           0.023355
UNITED KINGDOM               0.020671
TAIWAN, PROVINCE OF CHINA    0.017213
MYANMAR                      0.012728
BRUNEI DARUSSALAM            0.012317
NEW ZEALAND                  0.008480
CANADA                       0.008338
CAMBODIA                     0.007785
Name: destination, dtype: float64
------------------------------------------
Feature:gender

No. of unique values:2

Distribution:
```

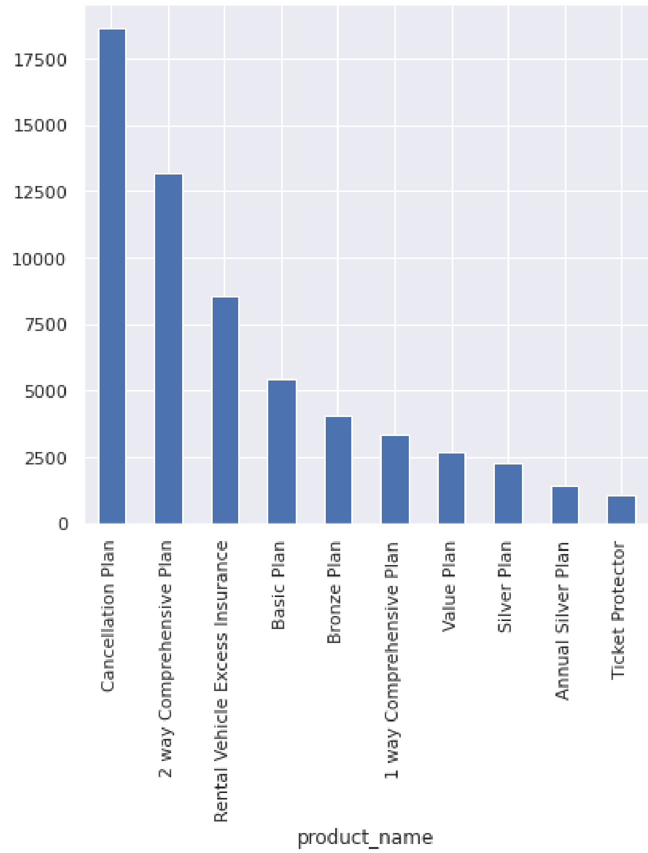
```
M    0.513036
F    0.486964
Name: gender, dtype: float64
----------------------------------------
```



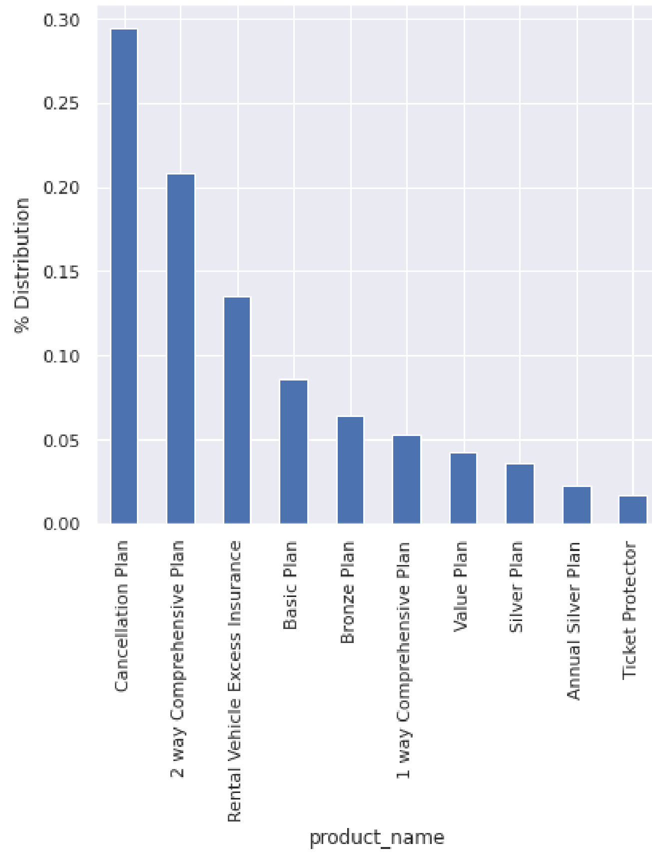




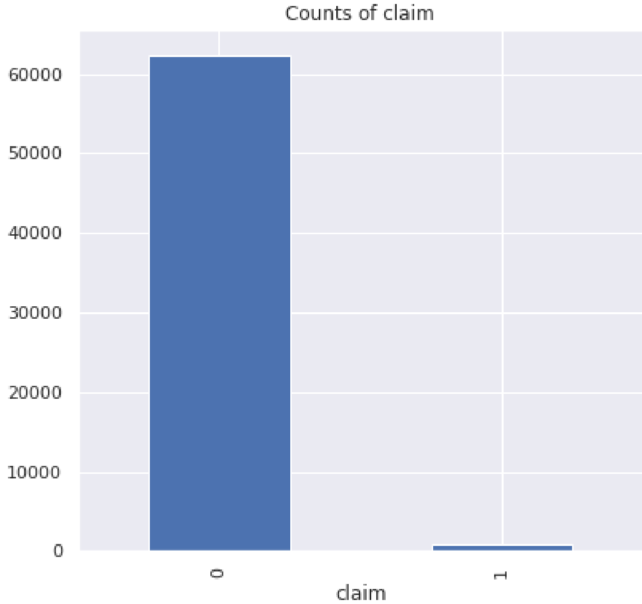
Counts of product_name



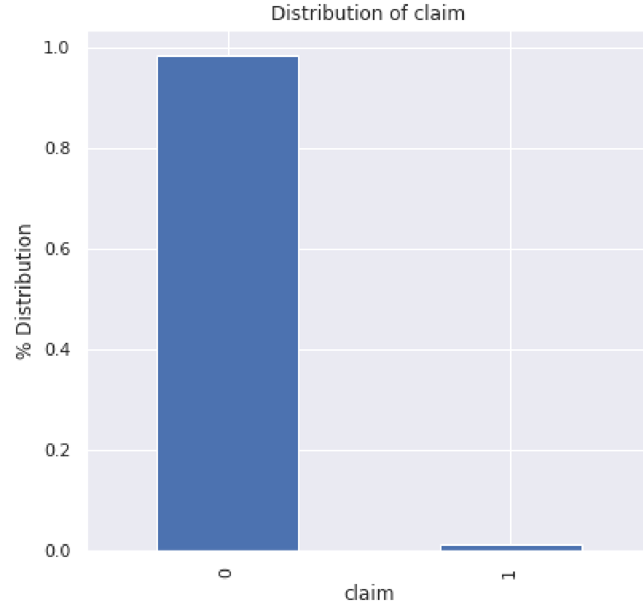
Distribution of product_name



Counts of claim

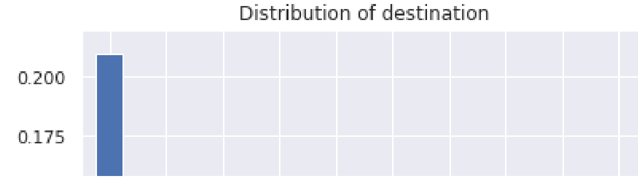


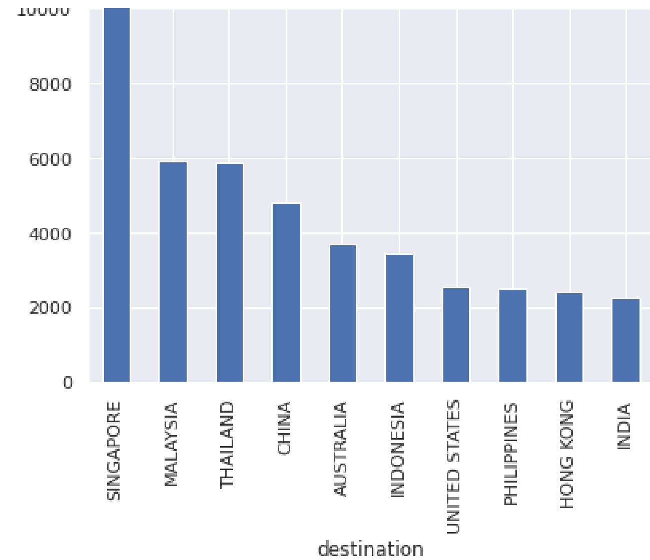
Distribution of claim

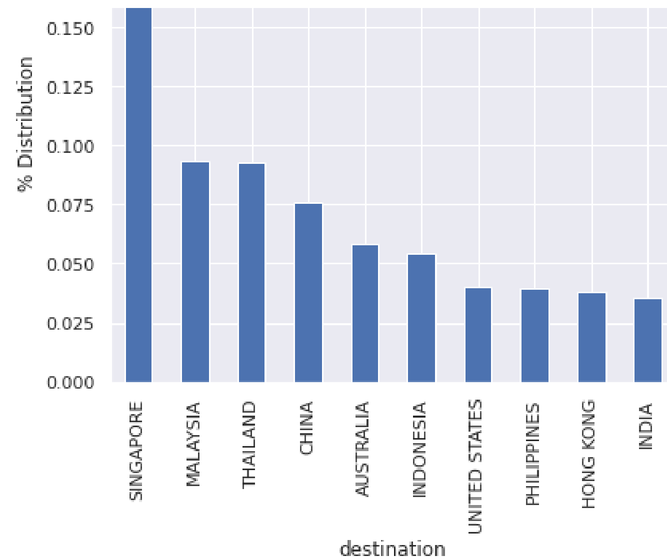


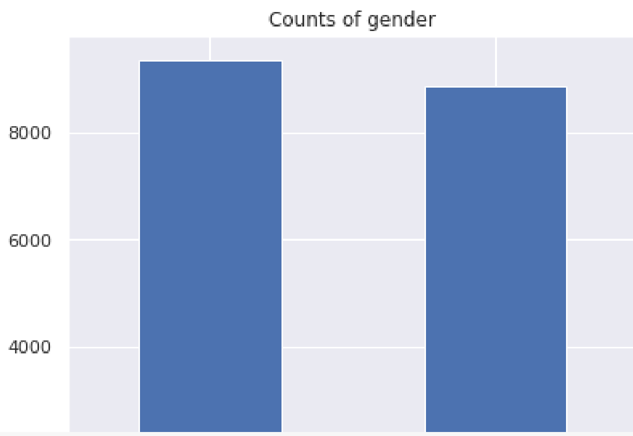
Counts of destination

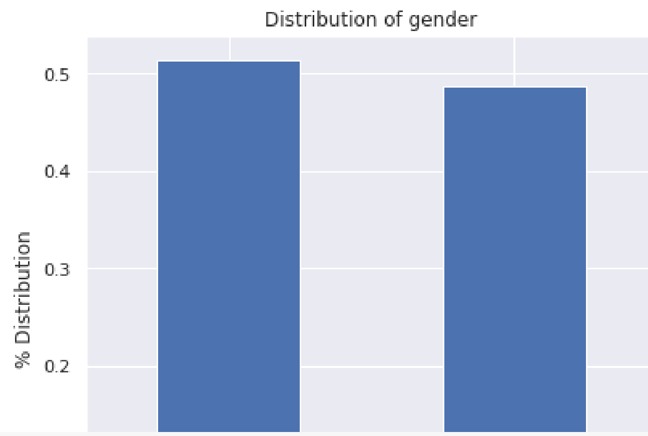
Distribution of destination












```
df[df['duration']==0].describe(include='all').T
```

	count	unique	top	freq	mean	std	min	25%	50%	75%
agency	61	8	JWT	20	NaN	NaN	NaN	NaN	NaN	NaN
agency_type	61	2	Airlines	56	NaN	NaN	NaN	NaN	NaN	NaN
distribution_channel	61	2	Online	60	NaN	NaN	NaN	NaN	NaN	NaN
product_name	61	8	Value Plan	25	NaN	NaN	NaN	NaN	NaN	NaN
claim	61	1	No	61	NaN	NaN	NaN	NaN	NaN	NaN
duration	61	NaN	NaN	NaN	0	0	0	0	0	0
destination	61	10	INDIA	22	NaN	NaN	NaN	NaN	NaN	NaN
net_sales	61	NaN	NaN	NaN	23.7448	23.7012	-14.4	9.77	18	31
commision_in_value	61	NaN	NaN	NaN	9.53082	9.63885	0	3.6	6.3	12.4
gender	38	2	M	26	NaN	NaN	NaN	NaN	NaN	NaN
age	61	NaN	NaN	NaN	71.6721	38.5831	23	48	49	118

Not able to deduce further

```
df[df['duration']<0].describe(include='all').T
```

	count	unique	top	freq	mean	std	min	25%	50%
agency	5	1	JZI	5	NaN	NaN	NaN	NaN	NaN
agency_type	5	1	Airlines	5	NaN	NaN	NaN	NaN	NaN
distribution_channel	5	1	Online	5	NaN	NaN	NaN	NaN	NaN
product_name	5	1	Basic Plan	5	NaN	NaN	NaN	NaN	NaN
claim	5	1	No	5	NaN	NaN	NaN	NaN	NaN
duration	5	NaN	NaN	NaN	-1.2	0.447214	-2	-1	-1
destination	5	5	BANGLADESH	1	NaN	NaN	NaN	NaN	NaN
net_sales	5	NaN	NaN	NaN	19.6	2.19089	18	18	18
commision_in_value	5	NaN	NaN	NaN	6.86	0.766812	6.3	6.3	6.3
gender	1	1	M	1	NaN	NaN	NaN	NaN	NaN
age	5	NaN	NaN	NaN	118	0	118	118	118

As per above data, below are some observations:

- All 5 negative values are coming from single agency JZI (Online).
- All have taken same product.
- All 5 customers have same age-118.

So considering these, it feels OK to turn them to +ve values. Lets change it for now. But this need to be discussed with business whether there can be -ve values or not.

```
df[df['net_sales']<=0]
```

	agency	agency_type	distribution_channel	product_name	claim	duration	desti
0	CBH	Travel Agency	Offline	Comprehensive Plan	No	186	MA
1	CBH	Travel Agency	Offline	Comprehensive Plan	No	186	MA
2	CWT	Travel Agency	Online	Rental Vehicle Excess Insurance	No	65	AUS
3	CWT	Travel Agency	Online	Rental Vehicle Excess Insurance	No	60	AUS
4	CWT	Travel Agency	Online	Rental Vehicle Excess Insurance	No	79	
...	...	...	...	...	...	...	
62806	C2B	Airlines	Online	Bronze Plan	No	12	SING
62833	CWT	Travel Agency	Online	Rental Vehicle Excess	No	165	AUS

By observing the data, it seems all these -ve values should be turned into positives. But its better not to touch them as we dont know the business context and the volume is also significant. Lets leave it as it is for now and proceed further.

Rental Vehicle

```
df[df['duration'].between(120,400)]
```

	agency	agency_type	distribution_channel	product_name	claim	duration	desti
0	CBH	Travel Agency	Offline	Comprehensive Plan	No	186	MA
1	CBH	Travel Agency	Offline	Comprehensive Plan	No	186	MA
9	CWT	Travel Agency	Online	Rental Vehicle Excess Insurance	No	186	AUS
20	CWT	Travel Agency	Online	Rental Vehicle Excess Insurance	No	136	N(

```
df[df['duration']>300]['product_name'].value_counts()
```

```
Cancellation Plan                       18630
2 way Comprehensive Plan                13158
Rental Vehicle Excess Insurance          8580
Basic Plan                               5469
Bronze Plan                              4049
1 way Comprehensive Plan                 3331
Value Plan                               2715
Silver Plan                              2249
Annual Silver Plan                       1423
Ticket Protector                         1056
Travel Cruise Protect                     527
Comprehensive Plan                        364
Gold Plan                                 352
24 Protect                                247
Single Trip Travel Protect Gold           204
Premier Plan                              194
Annual Gold Plan                          194
Single Trip Travel Protect Silver         173
Annual Travel Protect Gold                100
Annual Travel Protect Silver               86
Individual Comprehensive Plan              74
Single Trip Travel Protect Platinum        73
Annual Travel Protect Platinum             53
Spouse or Parents Comprehensive Plan       15
Child Comprehensive Plan                    9
Travel Cruise Protect Family                1
Name: product_name, dtype: int64
```

By looking at the product_name, Annual Gold Plan seems to contain all the features of Annual Silver Plan and some additional benefits. But by looking at the purchase counts, its clear that customers are not attracted to those additional benefits and hence they are buying only Annual Silver Plan. Probably business can do further research on it and can either decrease the Annual Gold Plan Price or increase/change the benefits.

```
df[df['duration']>750]['product_name'].value_counts()
```

```
df[df['duration']>730]['product_name'].value_counts()
```

```
Ticket Protector    14
Name: product_name, dtype: int64
```

Customers who are taking insurance for more than 10 years are opting for Ticket Protector Product.

```
np.sort(df['destination'].unique())
```

```
array(['ALBANIA', 'ANGOLA', 'ARGENTINA', 'ARMENIA', 'AUSTRALIA',
       'AUSTRIA', 'AZERBAIJAN', 'BAHRAIN', 'BANGLADESH', 'BARBADOS',
       'BELARUS', 'BELGIUM', 'BENIN', 'BERMUDA', 'BHUTAN', 'BOLIVIA',
       'BOSNIA AND HERZEGOVINA', 'BOTSWANA', 'BRAZIL',
       'BRUNEI DARUSSALAM', 'BULGARIA', 'CAMBODIA', 'CAMEROON', 'CANADA',
       'CAYMAN ISLANDS', 'CHILE', 'CHINA', 'COLOMBIA', 'COSTA RICA',
       'CROATIA', 'CYPRUS', 'CZECH REPUBLIC', 'DENMARK',
       'DOMINICAN REPUBLIC', 'ECUADOR', 'EGYPT', 'ESTONIA', 'ETHIOPIA',
       'FAROE ISLANDS', 'FIJI', 'FINLAND', 'FRANCE', 'FRENCH POLYNESIA',
       'GEORGIA', 'GERMANY', 'GHANA', 'GREECE', 'GUADELOUPE', 'GUAM',
       'GUATEMALA', 'GUINEA', 'GUINEA-BISSAU', 'GUYANA', 'HONG KONG',
       'HUNGARY', 'ICELAND', 'INDIA', 'INDONESIA',
       'IRAN, ISLAMIC REPUBLIC OF', 'IRELAND', 'ISRAEL', 'ITALY',
       'JAMAICA', 'JAPAN', 'JORDAN', 'KAZAKHSTAN', 'KENYA',
       "KOREA, DEMOCRATIC PEOPLE'S REPUBLIC OF", 'KOREA, REPUBLIC OF',
       'KUWAIT', 'KYRGYZSTAN', "LAO PEOPLE'S DEMOCRATIC REPUBLIC",
       'LATVIA', 'LEBANON', 'LIBYAN ARAB JAMAHIRIYA', 'LITHUANIA',
       'LUXEMBOURG', 'MACAO',
       'MACEDONIA, THE FORMER YUGOSLAV REPUBLIC OF', 'MALAYSIA',
       'MALDIVES', 'MALI', 'MALTA', 'MAURITIUS', 'MEXICO',
       'MOLDOVA, REPUBLIC OF', 'MONGOLIA', 'MOROCCO', 'MYANMAR',
       'NAMIBIA', 'NEPAL', 'NETHERLANDS', 'NEW CALEDONIA', 'NEW ZEALAND',
       'NIGERIA', 'NORTHERN MARIANA ISLANDS', 'NORWAY', 'OMAN',
       'PAKISTAN', 'PANAMA', 'PAPUA NEW GUINEA', 'PERU', 'PHILIPPINES',
       'POLAND', 'PORTUGAL', 'PUERTO RICO', 'QATAR',
       'REPUBLIC OF MONTENEGRO', 'REUNION', 'ROMANIA',
       'RUSSIAN FEDERATION', 'RWANDA', 'SAMOA', 'SAUDI ARABIA', 'SENEGAL',
       'SERBIA', 'SEYCHELLES', 'SIERRA LEONE', 'SINGAPORE', 'SLOVENIA',
       'SOLOMON ISLANDS', 'SOUTH AFRICA', 'SPAIN', 'SRI LANKA', 'SWEDEN',
       'SWITZERLAND', 'TAIWAN, PROVINCE OF CHINA', 'TAJIKISTAN',
       'TANZANIA, UNITED REPUBLIC OF', 'THAILAND', 'TIBET',
       'TRINIDAD AND TOBAGO', 'TUNISIA', 'TURKEY', 'TURKMENISTAN',
       'TURKS AND CAICOS ISLANDS', 'UGANDA', 'UKRAINE',
       'UNITED ARAB EMIRATES', 'UNITED KINGDOM', 'UNITED STATES',
       'URUGUAY', 'UZBEKISTAN', 'VANUATU', 'VENEZUELA', 'VIET NAM',
       'VIRGIN ISLANDS, U.S.', 'ZAMBIA', 'ZIMBABWE'], dtype=object)
```

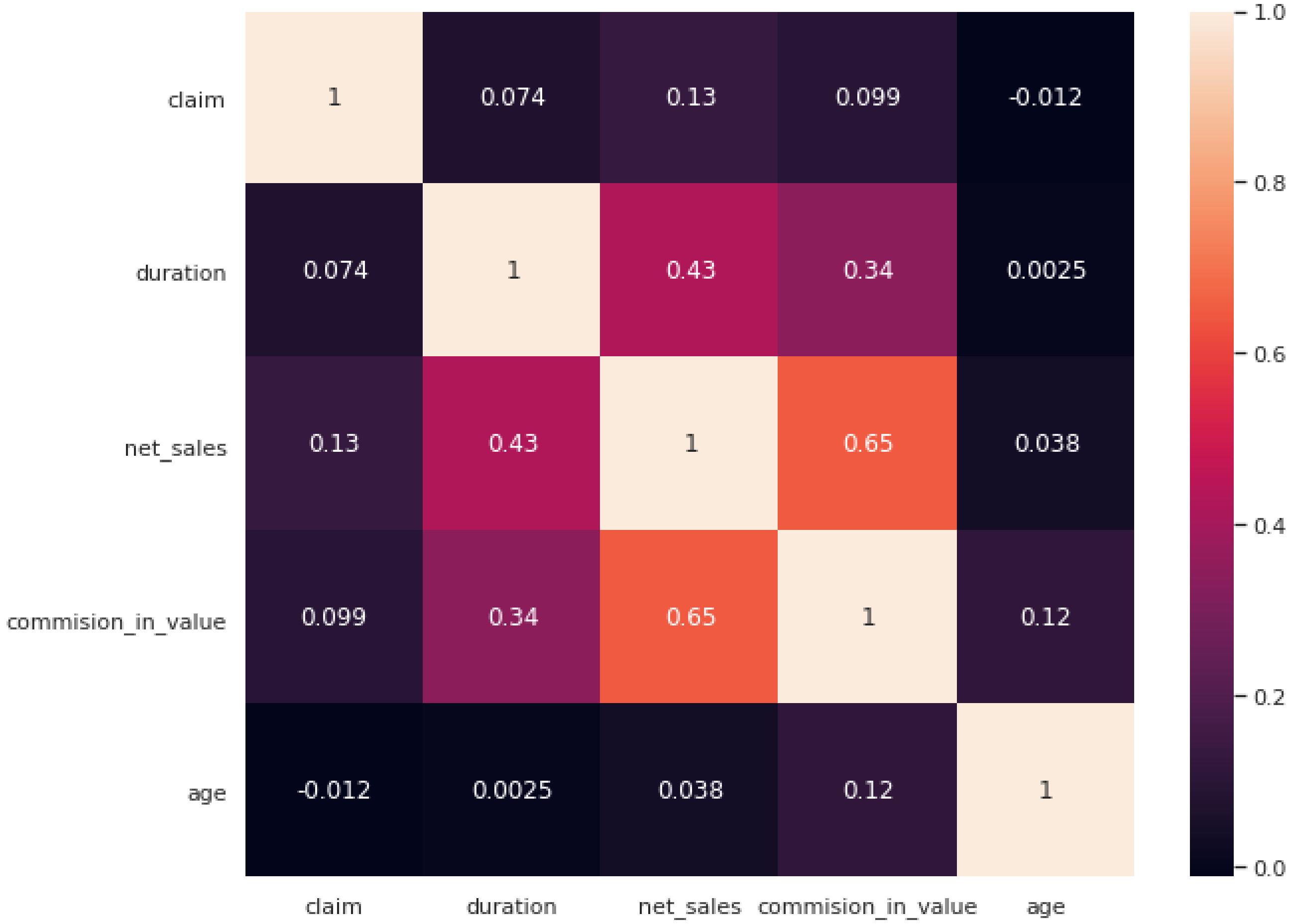
Multivariate Analysis

```
# Before starting analyis, its better to encode Target feature from YES/NO to 1/0
# This would help in doing the analysis better.
```

```
df['claim']=df['claim'].map({'Yes':1,'No':0})
```

```
# For continuous variables, Correlation with Heatmap serves as starting point
plt.figure(figsize=(10,8))
sns.heatmap(df.corr(),annot=True)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f1ab3ceaa10>

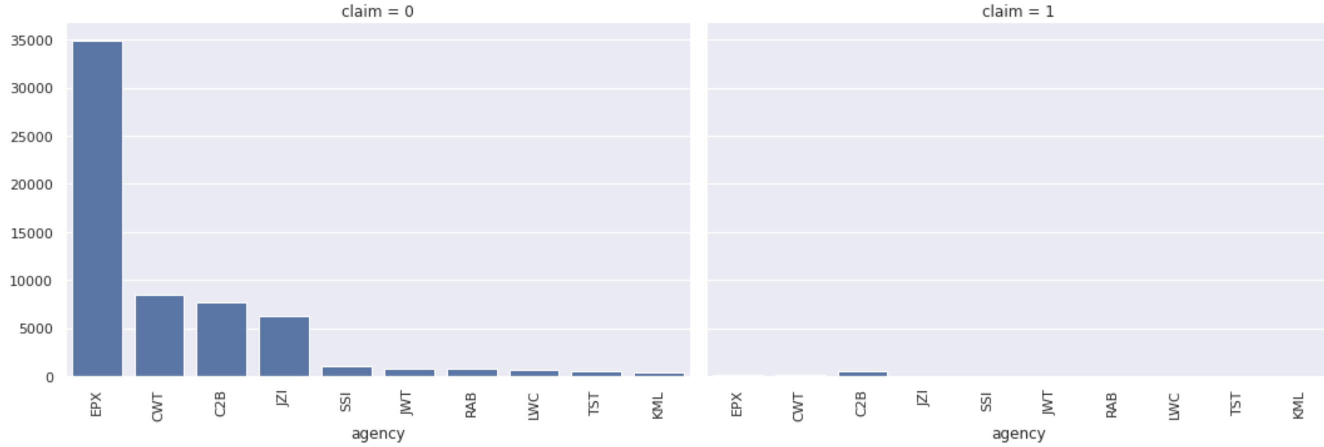


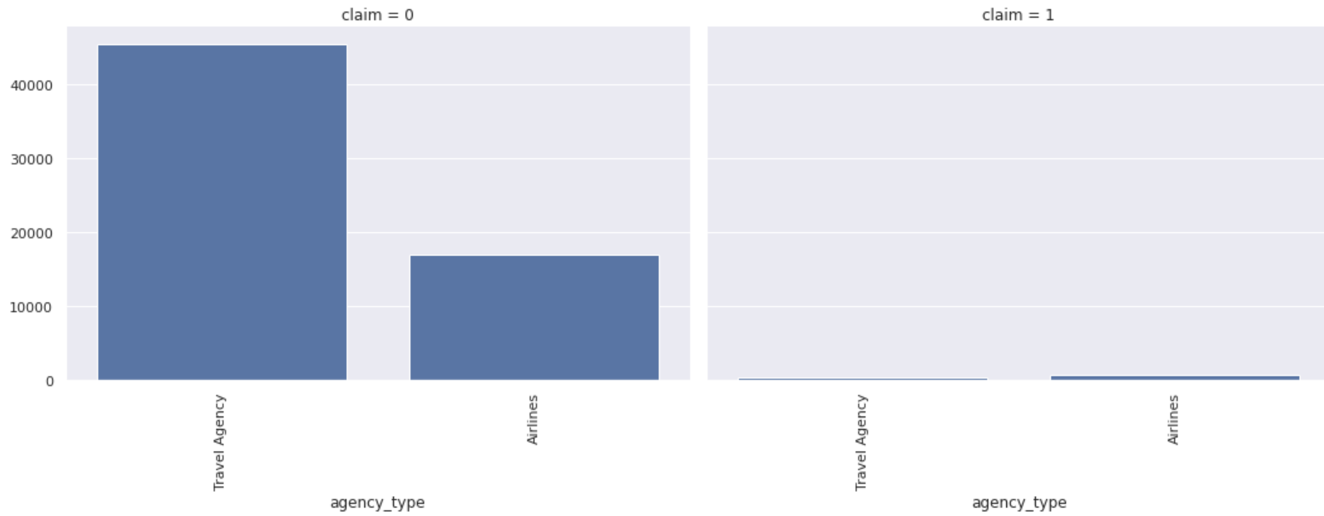

```
target='claim'
```

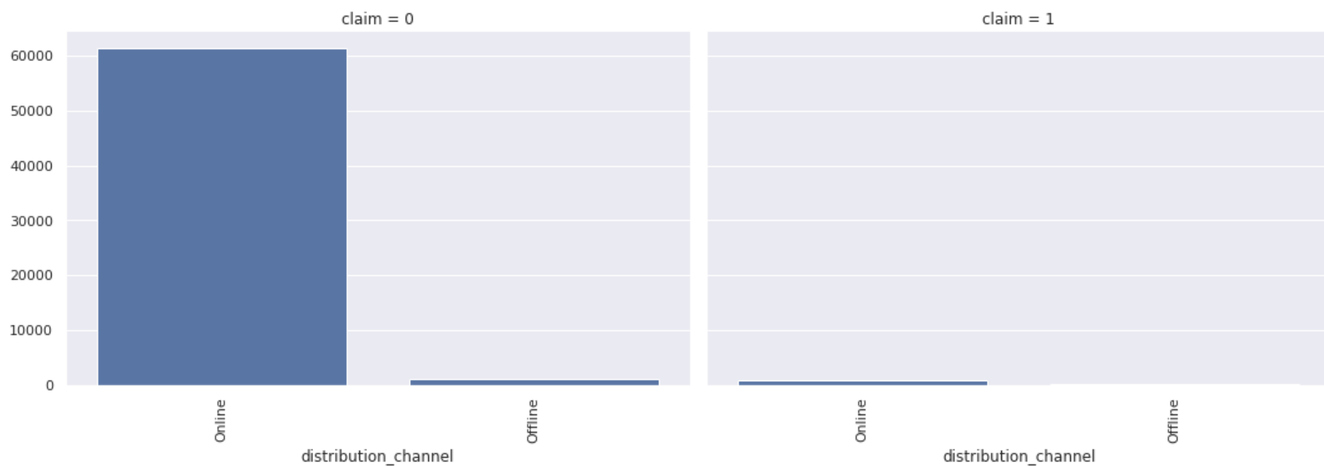
```
def feature_target_plot(feature):
    #plt.figure(figsize=(12,12))
    nbins=df[feature]
    if feature in numerical_feature_list:
        g=sns.FacetGrid(data=df,col=target,height=5, aspect =1.5)
        g.map(plt.hist,feature)
        g.set_xticklabels(rotation=90)
    else:
        g=sns.FacetGrid(data=df,col=target,height=5, aspect =1.5)
        g.map(sns.countplot,feature,order=df[feature].value_counts().index[:10])
        g.set_xticklabels(rotation=90)
```

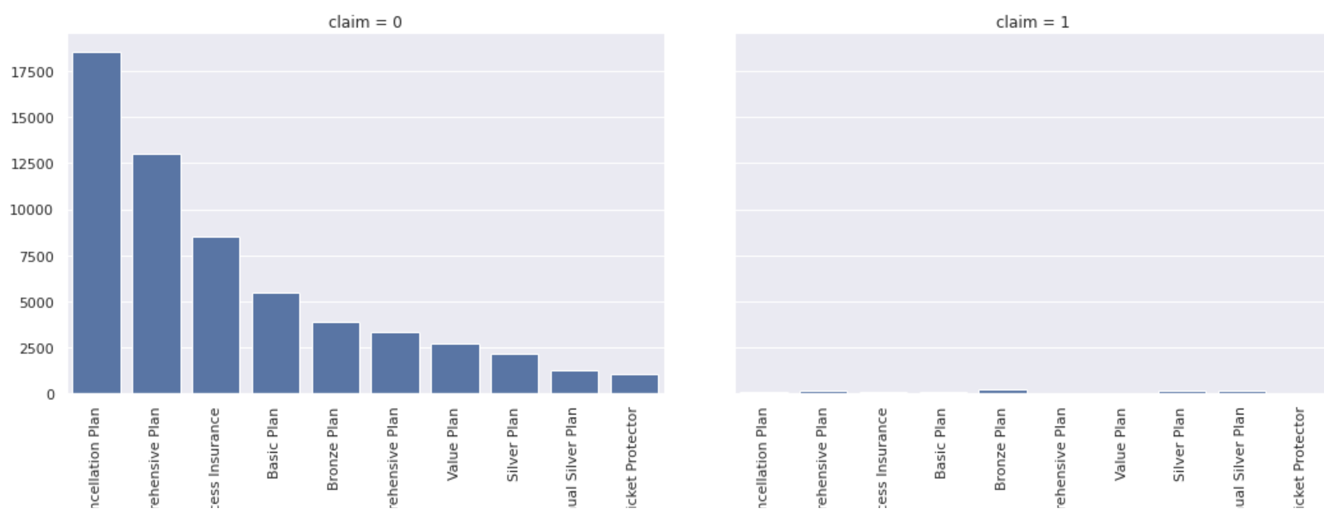
```
for i in df.columns:
```

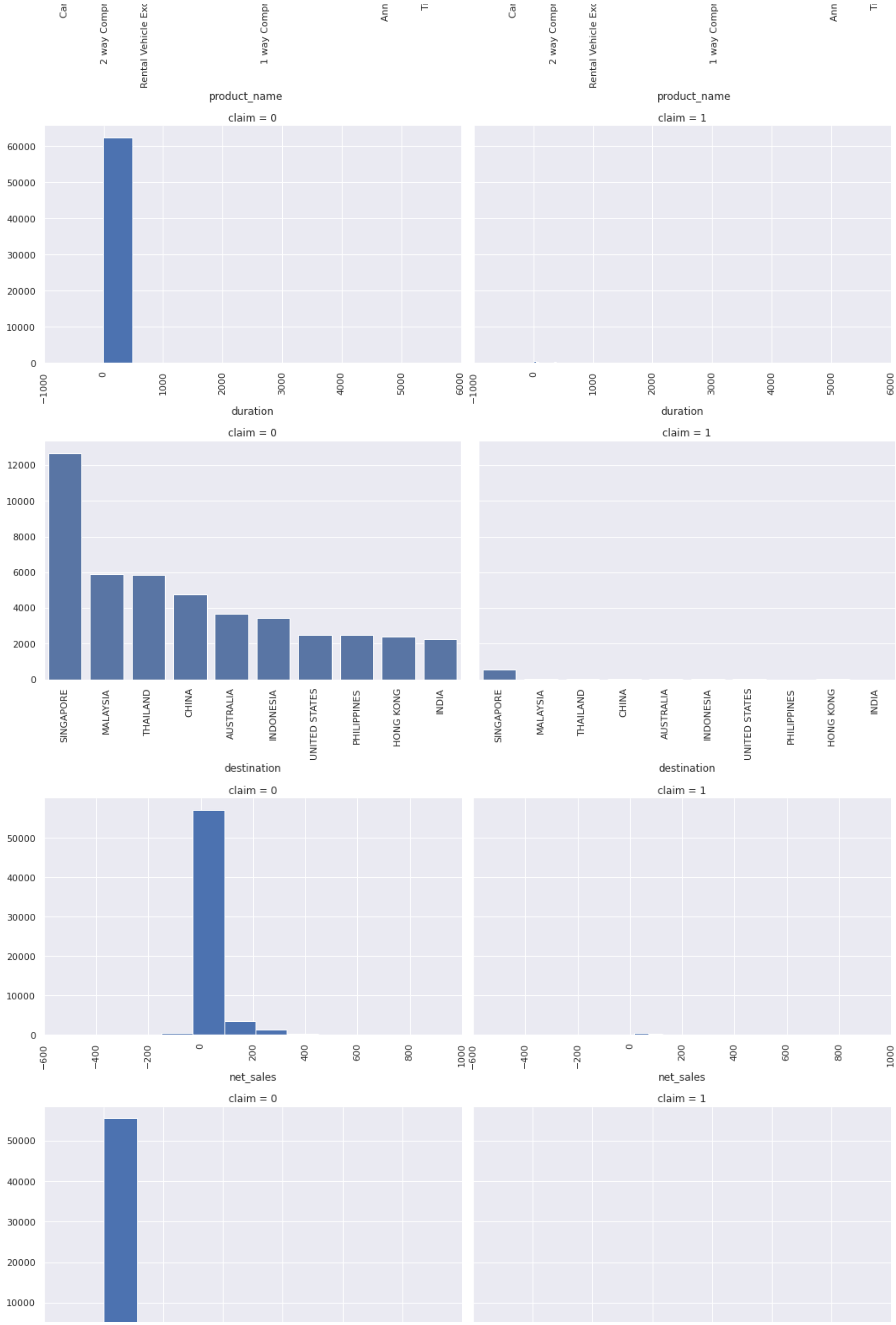
```
    if i!=target:
        feature_target_plot(i)
```


claim = 0
claim = 1
35000
30000
25000
20000
15000
10000
5000
0
EPX
CWT
C2B
JZI
SSI
JWT
RAB
LWC
TST
KML
agency


claim = 0
claim = 1
40000
30000
20000
10000
0
Travel Agency
Airlines
agency_type


claim = 0
claim = 1
60000
50000
40000
30000
20000
10000
0
Online
Offline
distribution_channel


claim = 0
claim = 1
17500
15000
12500
10000
7500
5000
2500
0
Basic Plan
Bronze Plan
Value Plan
Silver Plan



product_name
claim = 0
claim = 1
duration
destination
SINGAPORE
MALAYSIA
THAILAND
CHINA
AUSTRALIA
INDONESIA
UNITED STATES
PHILIPPINES
HONG KONG
INDIA
net_sales

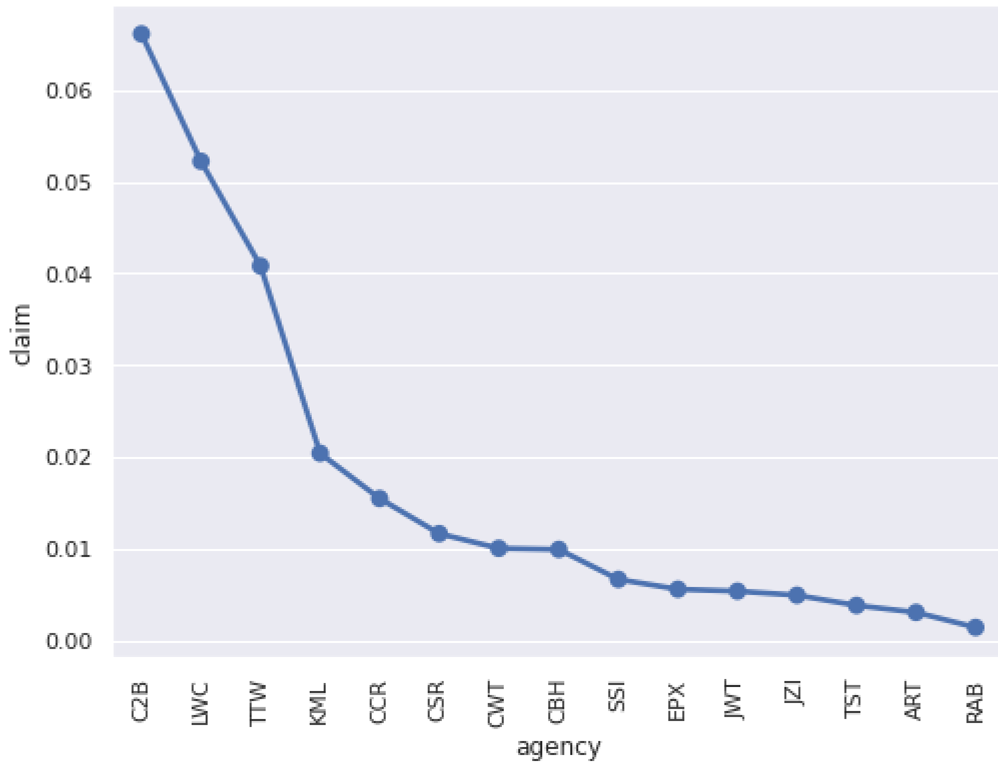

```
def feature_target_analysis(feature,count):
    if feature!=target:
        plt.figure(count,figsize=(8,6))
        if i in numerical_feature_list:
            nbins=(df[feature].max()-abs(df[feature].min()))/(np.std(df[feature]))
            new_feature=feature+'_Band'
            df[new_feature]=pd.qcut(x=df[feature],q=4,duplicates='drop')
            feature=new_feature
            feature_band_list.append(new_feature)
            sns.pointplot(x=feature,y=target,data=df[[target,feature]].groupby(by=feature,as_
            plt.xticks(rotation=90)
        else:
            sns.pointplot(x=feature,y=target,data=df[[target,feature]].groupby(by=feature,as_
            plt.xticks(rotation=90)
```

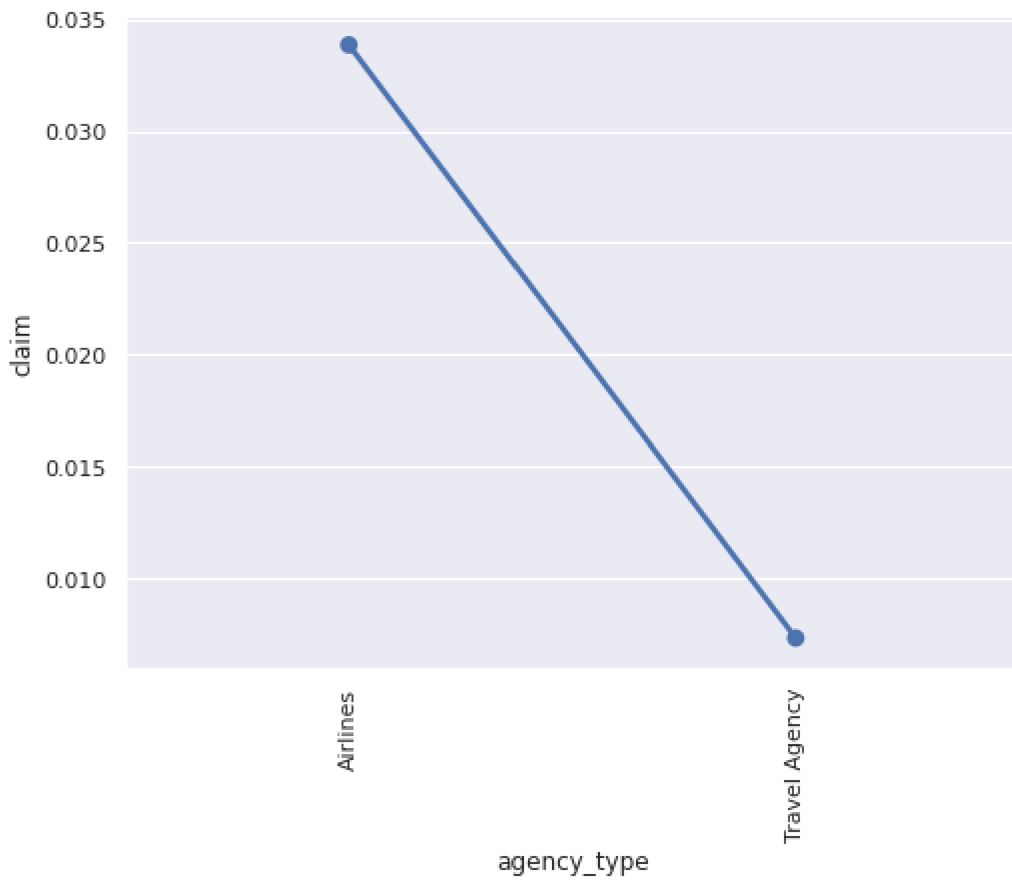
```
df[[target,'agency']].groupby(by='agency',as_index=False).mean().sort_values(by='claim',ascen
```

	agency	claim
2	C2B	0.066167
11	LWC	0.052250
15	TTW	0.040816
10	KML	0.020408
4	CCR	0.015464
5	CSR	0.011628
6	CWT	0.010023
3	CBH	0.009901
13	SSI	0.006629
7	EPX	0.005553

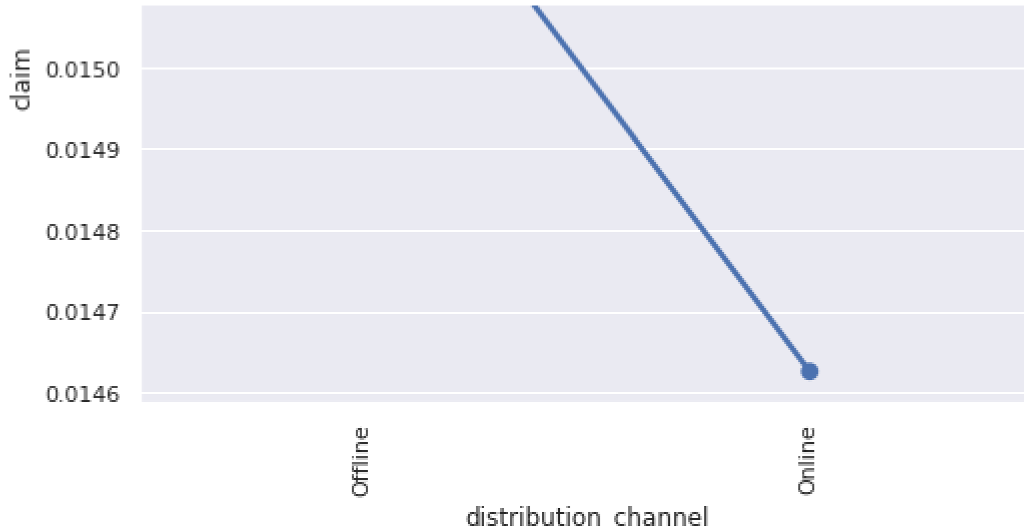
```
feature_band_list=[]
count=0
for i in categorical_feature_list+numerical_feature_list:
```

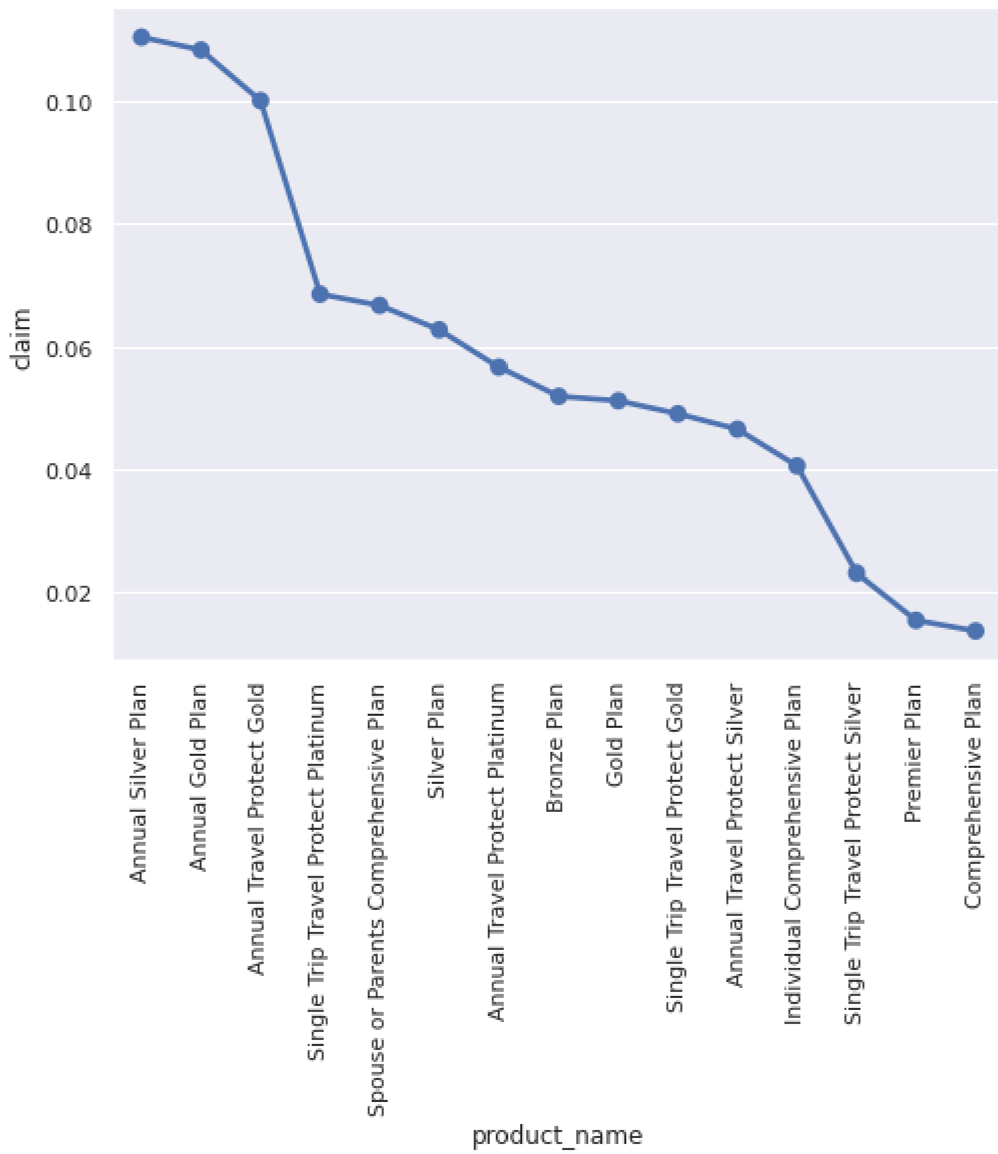
```
    count=count+1
    feature_target_analysis(i,count)
```


0.06
0.05
0.04
0.03
0.02
0.01
0.00
claim
C2B
LWC
TTW
KML
CCR
CSR
CWT
CBH
SSI
EPX
JWT
JZI
TST
ART
RAB
agency

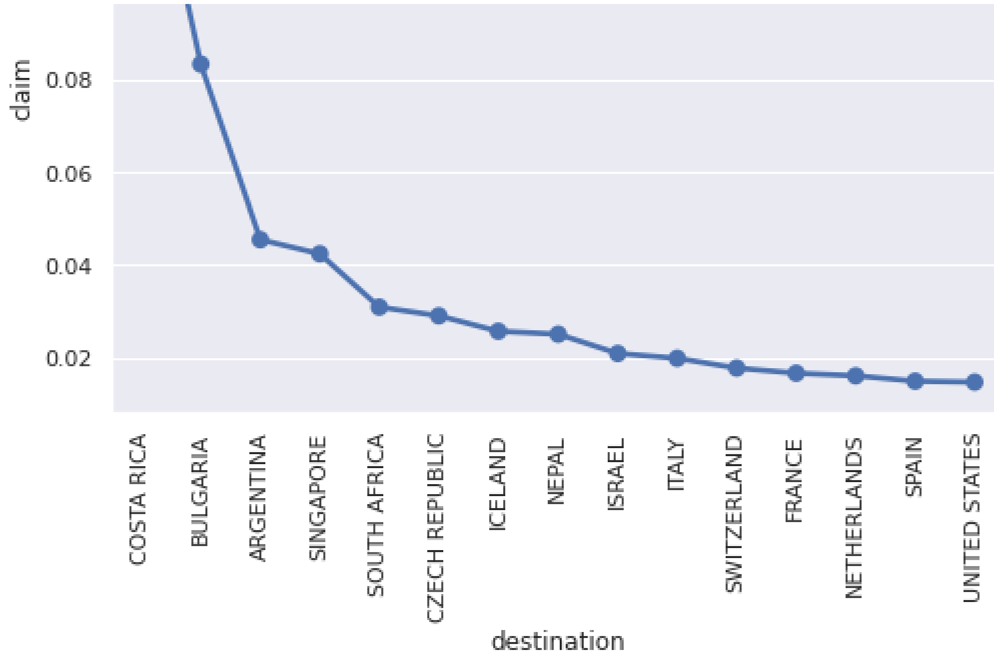

0.035
0.030
0.025
0.020
0.015
0.010
claim
Airlines
Travel Agency
agency_type

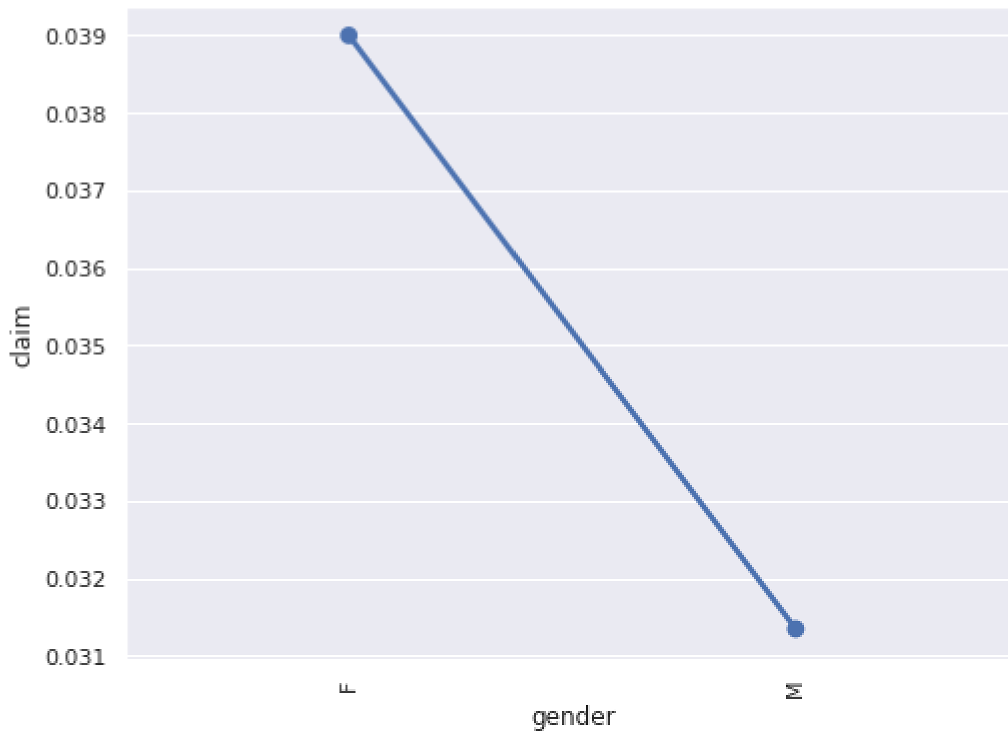

0.0153
0.0152
0.0151


claim
0.0150
0.0149
0.0148
0.0147
0.0146
Offline
Online
distribution_channel

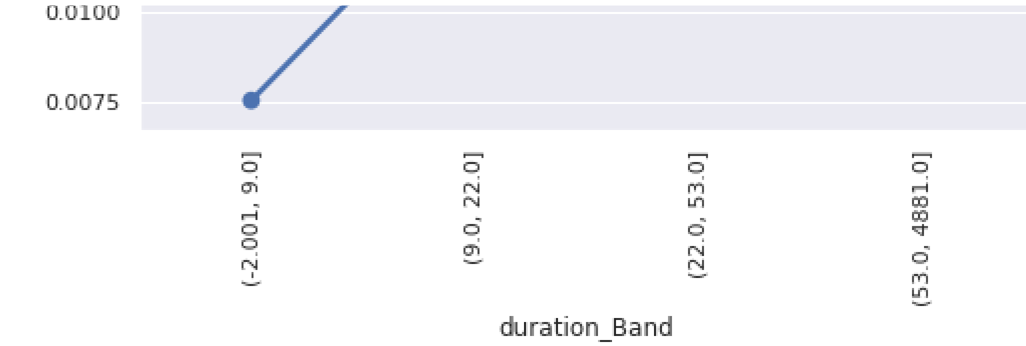

claim
0.10
0.08
0.06
0.04
0.02
Annual Silver Plan
Annual Gold Plan
Annual Travel Protect Gold
Single Trip Travel Protect Platinum
Spouse or Parents Comprehensive Plan
Silver Plan
Annual Travel Protect Platinum
Bronze Plan
Gold Plan
Single Trip Travel Protect Gold
Annual Travel Protect Silver
Individual Comprehensive Plan
Single Trip Travel Protect Silver
Premier Plan
Comprehensive Plan
product_name

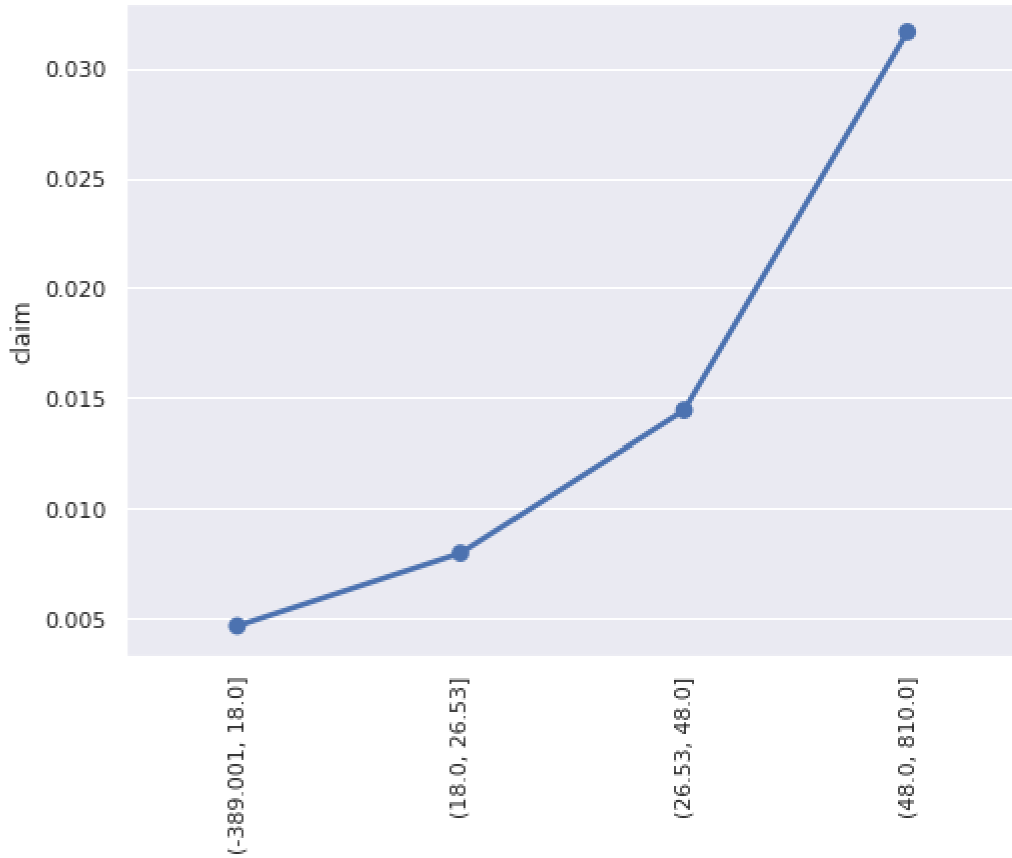

0.14
0.12
0.10


claim
0.08
0.06
0.04
0.02
COSTA RICA
BULGARIA
ARGENTINA
SINGAPORE
SOUTH AFRICA
CZECH REPUBLIC
ICELAND
NEPAL
ISRAEL
ITALY
SWITZERLAND
FRANCE
NETHERLANDS
SPAIN
UNITED STATES
destination


claim
0.039
0.038
0.037
0.036
0.035
0.034
0.033
0.032
0.031
F
M
gender


claim
0.0225
0.0200
0.0175
0.0150
0.0125






```
feature_band_list
```

    ['duration_Band', 'net_sales_Band', 'commision_in_value_Band', 'age_Band']

```
# Removing the Band features created for analysis and understanding of data
df.drop(feature_band_list,inplace=True,axis=1)
```

```
df[df['claim']==1].describe(include='all').T
```

	count	unique	top	freq	mean	std	min	25%
agency	927	15	C2B	547	NaN	NaN	NaN	NaN
agency_type	927	2	Airlines	591	NaN	NaN	NaN	NaN
distribution_channel	927	2	Online	910	NaN	NaN	NaN	NaN
product_name	927	23	Bronze Plan	210	NaN	NaN	NaN	NaN
claim	927	NaN	NaN	NaN	1	0	1	1
duration	927	NaN	NaN	NaN	110.789	141.375	1	15
destination	927	41	SINGAPORE	562	NaN	NaN	NaN	NaN
net_sales	927	NaN	NaN	NaN	94.3744	93.1749	-37	29
commision_in_value	927	NaN	NaN	NaN	25.8464	32.4823	0	4.25
gender	639	2	F	346	NaN	NaN	NaN	NaN
duration_Band	927	4	(53.0,	375	NaN	NaN	NaN	NaN

Observations:

agency:

```
df['age_bin'].value_counts()
```

```
2    32618
3    13363
0    11772
1     5573
Name: age_bin, dtype: int64
```

```
df[df['claim']=='Yes']['agency'].value_counts()
```

```
C2B    547
EPX    195
CWT     86
LWC     36
JZI     31
KML      8
SSI      7
JWT      4
TTW      4
CCR      3
TST      2
ART      1
RAB      1
CBH      1
CSR      1
Name: agency, dtype: int64
```

```
df[df['agency']=='C2B'][['agency','product name','claim']].groupby(by=['claim','product name'
```

		agency
claim	product_name	
No	Annual Gold Plan	173
	Annual Silver Plan	1266
	Bronze Plan	3839
	Gold Plan	334
	Silver Plan	2108
Yes	Annual Gold Plan	21
	Annual Silver Plan	157
	Bronze Plan	210
	Gold Plan	18
	Silver Plan	141

```
df[df['product_name']=='Annual Gold Plan']['agency'].unique()
```

```
array(['C2B'], dtype=object)
```

```
df[(df['agency']=='EPX') & (df['claim']=='Yes')]['product_name'].value_counts()
```

```
2 way Comprehensive Plan    142
Cancellation Plan            44
1 way Comprehensive Plan      9
Name: product_name, dtype: int64
```

```
numerical_feature_list
```

```
['duration', 'net_sales', 'commision_in_value', 'age']
```

```
df
```

in	gender_commision_in_value_std	gender_age_mean	gender_age_median	gender_age_max	ge
.0	24.874254	42.940825	38	118	
.0	24.874254	42.940825	38	118	
.0	16.009914	38.024985	36	118	
.0	16.009914	38.024985	36	118	
.0	16.009914	38.024985	36	118	
...	...	...	...	...	
.0	25.045656	46.536322	42	118	
.0	25.045656	46.536322	42	118	

agency vs agency_type:

```
df['agency_type'].nunique()
```

```
2
```

```
df['agency_agency_type_nunique'].value_counts()
```

```
1    63326
Name: agency_agency_type_nunique, dtype: int64
```

Agency Type for each Agency is either Travel Agency or Airline but not both.

```
df[['agency_type','claim']].groupby(by='agency_type').agg({'claim':['count','sum','mean']})
```

	claim		
	count	sum	mean
agency_type			
Airlines	17457	591	0.033855
Travel Agency	45869	336	0.007325

```
df[['agency','agency_type','claim',]].groupby(by=['agency','agency_type']).agg({'claim':['cou
```

		claim		
		count	sum	mean
agency	agency_type			
ADM	Travel Agency	82	0	0.000000
ART	Airlines	331	1	0.003021
C2B	Airlines	8267	547	0.066167
CBH	Travel Agency	101	1	0.009901
CCR	Travel Agency	194	3	0.015464
CSR	Travel Agency	86	1	0.011628
CWT	Travel Agency	8580	86	0.010023
EPX	Travel Agency	35119	195	0.005553
JWT	Airlines	749	4	0.005340
JZI	Airlines	6329	31	0.004898
KML	Travel Agency	392	8	0.020408
LWC	Travel Agency	689	36	0.052250
RAB	Airlines	725	1	0.001379
SSI	Airlines	1056	7	0.006629
TST	Travel Agency	528	2	0.003788
TTW	Travel Agency	98	4	0.040816

- 6.6% of insurances are claimed when taken through C2B(Airline)
- Rest other airlines are having lower claim ratio.
- Claims through LWC and TTM are around 5% and 4% respectively. Though the volumes are less right now, this need to be monitored.

agency vs distribution_channel:

```
df['distribution_channel'].nunique()
```

```
2
```

```
df['agency_distribution_channel_nunique'].value_counts()
```

```
2    52532
1    10794
Name: agency_distribution_channel_nunique, dtype: int64
```

```
df[df['agency_distribution_channel_nunique']==1]['agency'].nunique()
```

```
9
```

```
df[df['agency_distribution_channel_nunique']==1]['distribution_channel'].unique()
```

```
array(['Offline', 'Online'], dtype=object)
```

Out of 26 agencies, only 9 agencies are having only either Online or Offline servies. Rest other 17 agencies have both servies.

```
df[df['claim']==1]['distribution_channel'].agg()
```

```
Online     910
Offline     17
Name: distribution_channel, dtype: int64
```

agency vs product_name:

```
df['product_name'].nunique()
```

```
26
```

```
df['agency_product_name_nunique'].value_counts()
```

```
3    41546
1    11110
5     8267
2     1714
6      689
Name: agency_product_name_nunique, dtype: int64
```

```
def is_agency_cross_product_sell():
    cross_products=dict()
    for i in df['product_name'].unique():
       if df[df['product_name']==i]['agency'].nunique()>1:
         cross_products[i]= list(df[df['product_name']==i]['agency'].unique())
    return cross_products
```

```
is_agency_cross_product_sell()
```

```
{'Comprehensive Plan': ['CBH', 'CSR', 'CCR'],
 'Premier Plan': ['KML', 'CCR', 'ADM', 'CSR', 'JZI', 'CBH'],
 'Value Plan': ['JZI', 'JWT', 'RAB', 'KML', 'ART', 'ADM']}
```

Observations:

- Each Agency sells upto a max of 6 products out of total 26 products. Majority agencies sell only 3 products.
- Out of 26 products, only 3 products are sold across by multiple agencies.

agency vs duration:

```
df['duration_bins'].nunique()
```

```
2
```

```
','agency_fe','agency_duration_mean','agency_duration_median','agency_duration_min','agency_c
```

agency	agency_fe	agency_duration_mean	agency_duration_median	agency_duration_min
ADM	0.001295	54.914634	41.5	3
ART	0.005227	30.359517	14.0	1
C2B	0.130547	95.219910	21.0	0
CBH	0.001595	91.950495	64.0	5
CCR	0.003064	62.809278	45.0	2
CSR	0.001358	79.895349	58.0	5
CWT	0.135489	41.450350	24.0	0
EPX	0.554575	39.158860	22.0	0
JWT	0.011828	29.371162	16.0	0
JZI	0.099943	34.156423	20.0	-2
KML	0.006190	40.426020	27.0	2
LWC	0.010880	150.252540	29.0	0
RAB	0.011449	23.870345	11.0	0
SSI	0.016676	122.757576	42.0	0
TST	0.008338	40.594697	33.0	3
TTW	0.001548	369.316327	365.0	364

- All Products sold by TTW Agency are taken for around a year.

- Products sold by SSI are having higher variation in duration. Thats probably because of Ticket Protector Product taken by some customers for more than 10 years.
- Products sold by C2B are also having higher variation and its because of Annual Silver Plan

agency vs destination:

```
df['destination'].nunique()
```

```
149
```

```
df[['agency','agency_destination','agency_destination_nunique','agency_destination_count']].g
```

agency	agency_destination	agency_destination_nunique	agency_destination_count
ADM	0.000821	13	82
ART	0.001627	23	331
C2B	0.130547	1	8267
CBH	0.000805	3	101
CCR	0.001737	3	194
CSR	0.000790	4	86
CWT	0.039605	74	8580
EPX	0.059486	138	35119
JWT	0.011828	1	749
JZI	0.018160	68	6329
KML	0.003727	21	392
LWC	0.003142	45	689
RAB	0.011212	7	725
SSI	0.016676	1	1056
TST	0.004406	2	528
TTW	0.001153	6	98

- C2B Agency booked insurance for customers going to singapore only.

agency vs net_sales:

```
[i for i in df.columns if 'agency_net_sales' in i]
```

```
[i for i in df.columns if 'agency_net_sales' in i]
```

```
['agency_net_sales_mean',
 'agency_net_sales_median',
 'agency_net_sales_max',
 'agency_net_sales_min',
 'agency_net_sales_std']
```

```
[i for i in df.columns if 'claim' in i]
```

```
:y_fe','agency_claim_mean']+[i for i in df.columns if 'agency_net_sales' in i]].groupby(by='a
```

agency	agency_fe	agency_claim_mean	agency_net_sales_mean	agency_net_sales_median
ADM	0.001295	0.000000	53.256098	75.00
ART	0.005227	0.003021	28.691601	23.00
C2B	0.130547	0.066167	78.865811	35.50
CBH	0.001595	0.009901	27.168317	29.00
CCR	0.003064	0.015464	30.654639	29.00
CSR	0.001358	0.011628	32.802326	29.00
CWT	0.135489	0.010023	43.040769	39.60
EPX	0.554575	0.005553	32.564993	22.00
JWT	0.011828	0.005340	53.012016	39.00
JZI	0.099943	0.004898	32.338442	26.00
KML	0.006190	0.020408	55.448980	38.00
LWC	0.010880	0.052250	111.510813	47.00
RAB	0.011449	0.001379	17.817931	15.00
SSI	0.016676	0.006629	6.365208	4.23
TST	0.008338	0.003788	29.318182	30.00
TTW	0.001548	0.040816	93.204082	97.00

- Claims seem to be higher for agencies which have Average net sales mean
- net_sales in TTW agency is high and consistent with mean and median very close.
- But these stats will significantly vary if -ve values are turned out to be +ves.

agency vs commision_in_value:

```
df[['agency','agency_fe']+[i for i in df.columns if 'agency_commision_in_value' in i]].groupb
```

agency	agency_fe	agency_commision_in_value_mean	agency_commision_in_value_median	a
ADM	0.001295	38.254878	34.390	
ART	0.005227	10.553927	8.490	
C2B	0.130547	20.533124	9.750	
CBH	0.001595	10.030693	9.570	
CCR	0.003064	10.313351	9.570	
CSR	0.001358	10.824767	9.570	
CWT	0.135489	32.380615	23.760	
EPX	0.554575	0.000000	0.000	
JWT	0.011828	21.640053	15.600	
JZI	0.099943	11.766669	9.100	
KML	0.006190	21.812194	14.440	
LWC	0.010880	74.265791	31.530	
RAB	0.011449	7.226483	6.000	
SSI	0.016676	1.784886	1.185	
TST	0.008338	10.523201	10.500	
TTW	0.001548	0.000000	0.000	

- EPX agency sells more than 50% of total insurances. The probable reason being the commission value is Zero indicating that EPX sells insurance without taking any commission. May be thats the reason, insurance company has given authorization to sell 2 most sold products to EPX
- Even TTW is providing insurance with zero comission.

agency vs gender:

```
df['agency_gender_nunique'].value_counts()
```

```
1    36175
3    17518
2     9633
Name: agency_gender_nunique, dtype: int64
```

```
df['gender'].unique()
```

```
array(['F', 'X', 'M'], dtype=object)
```

```
df[['agency','gender','product_name']].groupby(by=['agency','gender']).count()
```

		product_name
agency	**gender**	
ADM	**F**	16
	M	17
	X	49
ART	**F**	96
	M	230
	X	5
C2B	**F**	4594
	M	3673
CBH	**F**	71
	M	30
CCR	**F**	143
	M	50

```
df.groupby(by=['agency','claim'])['product_name'].count()
```

```
agency  claim
ADM     0           82
ART     0          330
        1            1
C2B     0         7720
        1          547
CBH     0          100
        1            1
CCR     0          191
        1            3
CSR     0           85
        1            1
CWT     0         8494
        1           86
EPX     0        34924
        1          195
JWT     0          745
        1            4
JZI     0         6298
        1           31
KML     0          384
        1            8
LWC     0          653
        1           36
RAB     0          724
        1            1
SSI     0         1049
        1            7
TST     0          526
        1            2
```

```
TTW     0           94
        1            4
Name: product_name, dtype: int64
```

agency vs age:

```
df[['agency','agency_fe']+[i for i in df.columns if 'agency_age_' in i]].groupby(by='agency')
```

agency	agency_fe	agency_age_mean	agency_age_median	agency_age_max	agency_age_min
ADM	0.001295	23.804878	22.0	118	21
ART	0.005227	60.314199	48.0	118	22
C2B	0.130547	37.749244	34.0	88	1
CBH	0.001595	57.128713	65.0	87	8
CCR	0.003064	67.572165	67.0	118	5
CSR	0.001358	57.511628	63.5	84	8
CWT	0.135489	40.291841	38.0	118	21
EPX	0.554575	36.933740	36.0	118	7
JWT	0.011828	118.000000	118.0	118	118
JZI	0.099943	44.603571	44.0	118	1
KML	0.006190	47.918367	48.0	61	19
LWC	0.010880	39.992743	37.0	84	12
RAB	0.011449	42.689655	40.0	84	15
SSI	0.016676	49.193182	48.0	118	48
TST	0.008338	50.467803	54.0	88	0
TTW	0.001548	40.642857	40.5	65	3

Random

```
d=df.groupby(by=['agency','product_name']).agg({'agency_type':'count','claim':['sum','mean']}
d.columns = ['d' + '_'.join(c).strip('_') for c in d.columns]
d
```

agency	product_name			
ADM	**Premier Plan**	46	0	0.000000
	Value Plan	36	0	0.000000
ART	**24 Protect**	247	0	0.000000
	Value Plan	84	1	0.011905
C2B	**Annual Gold Plan**	194	21	0.108247
	Annual Silver Plan	1423	157	0.110330
	Bronze Plan	4049	210	0.051865
	Gold Plan	352	18	0.051136
	Silver Plan	2249	141	0.062695
CBH	**Comprehensive Plan**	99	1	0.010101
	Premier Plan	2	0	0.000000
CCR	**Comprehensive Plan**	181	3	0.016575
	Premier Plan	13	0	0.000000
CSR	**Comprehensive Plan**	84	1	0.011905
	Premier Plan	2	0	0.000000
CWT	**Rental Vehicle Excess Insurance**	8580	86	0.010023
EPX	**1 way Comprehensive Plan**	3331	9	0.002702
	2 way Comprehensive Plan	13158	142	0.010792
	Cancellation Plan	18630	44	0.002362
JWT	**Value Plan**	749	4	0.005340
JZI	**Basic Plan**	5469	23	0.004206
	Premier Plan	6	0	0.000000
	Value Plan	854	8	0.009368
KML	**Premier Plan**	125	3	0.024000
	Value Plan	267	5	0.018727
LWC	**Annual Travel Protect Gold**	100	10	0.100000
	Annual Travel Protect Platinum	53	3	0.056604
	Annual Travel Protect Silver	86	4	0.046512
	Single Trip Travel Protect Gold	204	10	0.049020
	Single Trip Travel Protect Platinum	73	5	0.068493