In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```
df=pd.read_csv("C:/Users/USER/Desktop/Mith/Train_data/Insurance_Claim_Info_data.c
df.head()
```

Out[2]:

	Claim Number	City Code	City	Enterprise Type	Claim Type	Claim Site	Product Insured
0	DQW1NZO0PL	NSK	Nashik	Public Limited Company	Property Loss	In Transit	Inventory - Raw Material
1	JS5GAPRN5B	BOM	Mumbai	One Person Company	Property Loss	In Transit	Cameras and other Misc. Security Equipment
2	ZTSVAQSEAQ	LKO	Lucknow	Public Limited Company	Property Loss	In Transit	Fixtures
3	EW7NWHI7LI	DEL	Delhi	Sole Proprietorship	Property Loss	In Transit	Pumps and Motors
4	UJOFDC41EL	DEL	Delhi	One Person Company	Property Loss	In Transit	Misc. Engineering Tools

In [3]:
```
df1=pd.read_csv("C:/Users/USER/Desktop/Mith/Train_data/Insurance_Date_data.csv")
df1.head()
```

Out[3]:

	Claim Number	Incident Date	Date Received
0	BCHKRDM32K	21-10-2007	31-10-2007
1	B3GPD5IZQW	26-05-2006	14-06-2006
2	EB757CV6XW	18-01-2004	10-02-2004
3	SP0Z0Q95OV	28-04-2004	06-05-2004
4	VKZUK7J3KK	04-11-2007	14-11-2007

In [4]:
```
df2=pd.read_csv("C:/Users/USER/Desktop/Mith/Train_data/Insurance_Result_data.csv"
df2.head()
```

Out[4]:

	Claim Number	Claim Amount	Close Amount	Disposition
0	Y5VA9KOE89	100.00	0.00	Deny
1	P51DOJLR8W	199.99	0.00	Deny
2	OUUZFP7EFL	410.00	59.85	Settle
3	CGP3L1CCP2	240.00	0.00	Deny
4	JDFLPD7J9Z	11.36	11.36	Approve in Full

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
Claim Number       0
City Code          0
City               0
Enterprise Type    0
Claim Type         0
Claim Site         0
Product Insured    0
dtype: int64
```

In [6]:
```
df.shape
```

Out[6]:
```
(34110, 7)
```

In [7]:
```
df1.isnull().sum()
```

Out[7]:
```
Claim Number     0
Incident Date    0
Date Received    0
dtype: int64
```

In [8]:
```
df1.shape
```

Out[8]:
```
(34110, 3)
```

In [9]:
```
df2.isnull().sum()
```

Out[9]:
```
Claim Number    0
Claim Amount    0
Close Amount    0
Disposition     0
dtype: int64
```

In [10]:
```
df2.shape
```

Out[10]:
```
(34110, 4)
```

In [11]:
```
X= np.concatenate((df,df1,df2),axis=1) #concatinating all the three datasets toge
```

In [12]:
```
x=pd.DataFrame(X)
x
```

Out[12]:

	0	1	2	3	4	5	6	
0	DQW1NZO0PL	NSK	Nashik	Public Limited Company	Property Loss	In Transit	Inventory - Raw Material	BCHKRDM
1	JS5GAPRN5B	BOM	Mumbai	One Person Company	Property Loss	In Transit	Cameras and other Misc. Security Equipment	B3GPD5IZ
2	ZTSVAQSEAQ	LKO	Lucknow	Public Limited Company	Property Loss	In Transit	Fixtures	EB757CV6
3	EW7NWHI7LI	DEL	Delhi	Sole Proprietorship	Property Loss	In Transit	Pumps and Motors	SP0Z0Q95
4	UJOFDC41EL	DEL	Delhi	One Person Company	Property Loss	In Transit	Misc. Engineering Tools	VKZUK7J3
...	...	...	...	...	...	...	...	
34105	5CFGWQ6IR5	AGR	Agra	Public Limited Company	Property Loss	In Transit	Misc. Engineering Tools	AF9GJPNK
34106	QQ6EAWA4Q5	LKO	Lucknow	Partnership Firm	Property Damage	In Transit	Misc. Electronic Items	IB6C791V
34107	X1J58PT1J5	HYD	Hyderabad	One Person Company	Property Damage	In Transit	Misc. Electronic Items	PGEDMDDH
34108	AGOXXE8KII	MAA	Chennai	Sole Proprietorship	Property Damage	Warehouse	Pumps and Motors	MNSM4JNT
34109	WT5RH23GPC	LKO	Lucknow	Private Ltd. MSME - Medium	Property Loss	In Transit	Inventory - Raw Material	5YKXQ6YT

34110 rows × 14 columns

In [13]:
```
x.columns=['Claim Number','City Code','City','Enterprise Type','Claim Type','Clai
           'claim Number','Incident Date','Date Received',
           'Claim number','Claim Amount','Close Amount','Disposition'] #giving co
```

In [15]:
```
x.head(5)
```

Out[15]:

	Claim Number	City Code	City	Enterprise Type	Claim Type	Claim Site	Product Insured	claim Number	Incide Da
0	DQW1NZO0PL	NSK	Nashik	Public Limited Company	Property Loss	In Transit	Inventory - Raw Material	BCHKRDM32K	21-1 20
1	JS5GAPRN5B	BOM	Mumbai	One Person Company	Property Loss	In Transit	Cameras and other Misc. Security Equipment	B3GPD5IZQW	26-(20
2	ZTSVAQSEAQ	LKO	Lucknow	Public Limited Company	Property Loss	In Transit	Fixtures	EB757CV6XW	18-(20
3	EW7NWHI7LI	DEL	Delhi	Sole Proprietorship	Property Loss	In Transit	Pumps and Motors	SP0Z0Q95OV	28-(20
4	UJOFDC41EL	DEL	Delhi	One Person Company	Property Loss	In Transit	Misc. Engineering Tools	VKZUK7J3KK	04- 20

In [16]:
```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34110 entries, 0 to 34109
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Claim Number     34110 non-null  object
 1   City Code        34110 non-null  object
 2   City             34110 non-null  object
 3   Enterprise Type  34110 non-null  object
 4   Claim Type       34110 non-null  object
 5   Claim Site       34110 non-null  object
 6   Product Insured  34110 non-null  object
 7   claim Number     34110 non-null  object
 8   Incident Date    34110 non-null  object
 9   Date Received    34110 non-null  object
 10  Claim number     34110 non-null  object
 11  Claim Amount     34110 non-null  object
 12  Close Amount     34110 non-null  object
 13  Disposition      34110 non-null  object
dtypes: object(14)
memory usage: 3.6+ MB
```

In [17]:
```
x["Date Received"] = pd.to_datetime(x["Date Received"])
x["Incident Date"] = pd.to_datetime(x["Incident Date"])
```

...

In [18]:
```
x["Date Received_year"] = x["Date Received"].dt.year
x["Date Received_month"] = x["Date Received"].dt.month
x["Date Received_day"] = x["Date Received"].dt.day
```

In [19]:
```
x["Incident Date_year"] = x["Incident Date"].dt.year
x["Incident Date_month"] = x["Incident Date"].dt.month
x["Incident Date_day"] = x["Incident Date"].dt.day
```

In [20]:
```
x.columns
```

Out[20]:
```
Index(['Claim Number', 'City Code', 'City', 'Enterprise Type', 'Claim Type',
       'Claim Site', 'Product Insured', 'claim Number', 'Incident Date',
       'Date Received', 'Claim number', 'Claim Amount', 'Close Amount',
       'Disposition', 'Date Received_year', 'Date Received_month',
       'Date Received_day', 'Incident Date_year', 'Incident Date_month',
       'Incident Date_day'],
      dtype='object')
```

In [21]:
```
x['Reporting Delay'] = (x['Date Received'] - x['Incident Date']) / np.timedelta64
```

In [22]:
```
x['Reporting Delay']
```

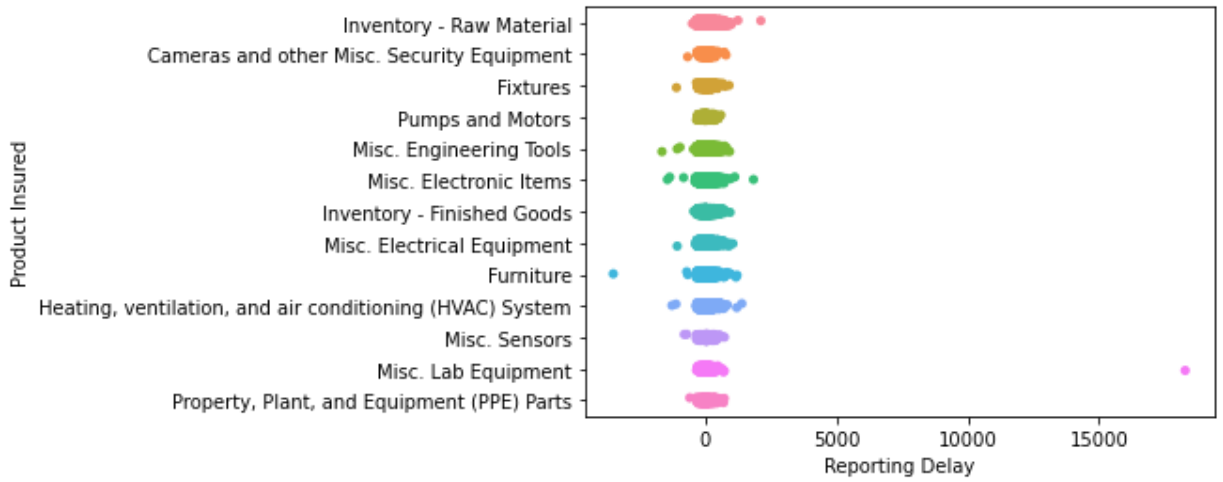
Out[22]:
```
0         10.0
1         19.0
2        258.0
3         38.0
4        217.0
         ...
34105    221.0
34106     62.0
34107    123.0
34108    230.0
34109    -72.0
Name: Reporting Delay, Length: 34110, dtype: float64
```

In [23]:
```
import klib #missing values plot using Klib
klib.missingval_plot(x)
```

```
No missing values found in the dataset.
```

In [24]:
```
sns.stripplot(x='Reporting Delay', y='Product Insured', data=x)
```

Out[24]: <AxesSubplot:xlabel='Reporting Delay', ylabel='Product Insured'>

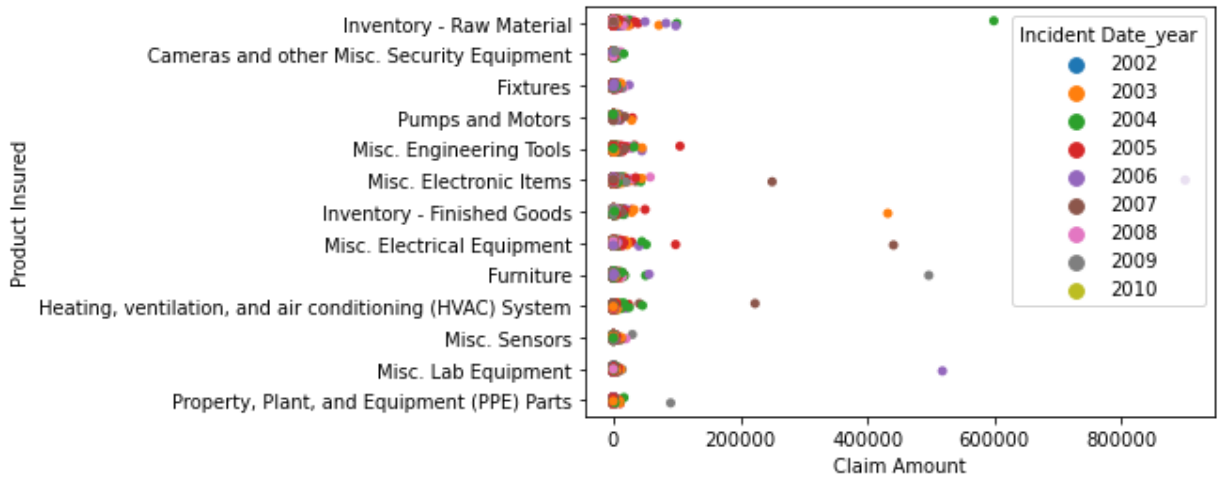


In [25]:
```
sns.stripplot(x='Claim Type', y='Claim Amount',hue= "Incident Date_year", data=x)
```

Out[25]: <AxesSubplot:xlabel='Claim Type', ylabel='Claim Amount'>

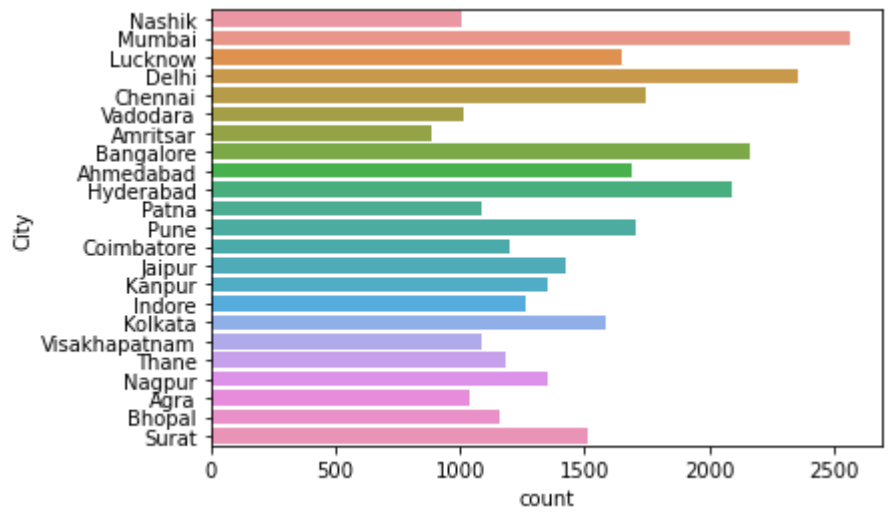
In [26]:
```
sns.stripplot(x='Claim Amount', y='Product Insured', hue= "Incident Date_year",da
```

Out[26]: <AxesSubplot:xlabel='Claim Amount', ylabel='Product Insured'>



In [27]:
```
sns.countplot(data=x, y="City")  #4. Which City has reported the highest number c
```

Out[27]: <AxesSubplot:xlabel='count', ylabel='City'>



In [28]:
```
sns.countplot(x='Claim Type',data=x)  #5. Which Type of Claims have the highest r
```

Out[28]:
```
<AxesSubplot:xlabel='Claim Type', ylabel='count'>
```

In [29]:
```
x.columns
```

Out[29]:
```
Index(['Claim Number', 'City Code', 'City', 'Enterprise Type', 'Claim Type',
       'Claim Site', 'Product Insured', 'claim Number', 'Incident Date',
       'Date Received', 'Claim number', 'Claim Amount', 'Close Amount',
       'Disposition', 'Date Received_year', 'Date Received_month',
       'Date Received_day', 'Incident Date_year', 'Incident Date_month',
       'Incident Date_day', 'Reporting Delay'],
      dtype='object')
```

In [30]:
```
x.drop(["Claim Number","claim Number","Claim number"
        ,"City","Enterprise Type","Product Insured",
       "City Code","Incident Date","Date Received_year","Date Received_month",
        "Date Received_day","Incident Date_year","Incident Date_month",
        "Incident Date_day","Date Received"],axis=1,inplace=True) #dropping the u
```

In [31]:
```
x.columns
```

Out[31]:
```
Index(['Claim Type', 'Claim Site', 'Claim Amount', 'Close Amount',
       'Disposition', 'Reporting Delay'],
      dtype='object')
```

In [33]:
```
x.drop(["Reporting Delay"],axis=1,inplace=True)
```

In [34]:
```
x["Claim Amount"]=x["Claim Amount"].astype('int')
x["Close Amount"]=x["Close Amount"].astype('int') #converting claim amount and cl
```

In [35]:
```
cat_at=x.select_dtypes(include=['object','category','character','string']).columr
x[cat_at]=x[cat_at].astype('category')
cat_at
```

Out[35]:
```
Index(['Claim Type', 'Claim Site', 'Disposition'], dtype='object')
```

In [36]:
```
x.dtypes
```

Out[36]:
```
Claim Type      category
Claim Site      category
Claim Amount       int32
Close Amount       int32
Disposition     category
dtype: object
```

In [37]:
```
X=x.drop('Disposition',axis=1)
Y=x['Disposition']            #splitting the dataset into x and y
```

In [38]:
```
from sklearn.model_selection import train_test_split      #
x_train, x_test, y_train, y_test=train_test_split(X, Y, test_size=0.3, random_sta
```

In [39]:
```
print(pd.value_counts(y_train)/y_train.count() * 100)

print(pd.value_counts(y_test) /y_test.count() * 100)
```

```
Deny               52.364200
Approve in Full    26.272145
Settle             21.363655
Name: Disposition, dtype: float64
Deny               52.232972
Approve in Full    27.157236
Settle             20.609792
Name: Disposition, dtype: float64
```

In [40]:
```
print("train size X : ",x_train.shape)
print("train size y : ",y_train.shape)
print("test size X : ",x_test.shape)
print("test size y : ",y_test.shape)
```

```
train size X :  (23877, 4)
train size y :  (23877,)
test size X :  (10233, 4)
test size y :  (10233,)
```

In [41]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
```

In [42]:
```
num_at=x.select_dtypes(include=['int','float','bool']).columns
num_at
```

Out[42]:
```
Index(['Claim Amount', 'Close Amount'], dtype='object')
```

In [43]:
```
x_train[num_at]=scaler.fit_transform(x_train[num_at])
x_test[num_at]=scaler.transform(x_test[num_at])
```

In [46]:
```
from sklearn.preprocessing import OneHotEncoder
ohc = OneHotEncoder()
```

In [47]:
```
cat=cat_at.drop(['Disposition'])
cat
```

Out[47]:
```
Index(['Claim Type', 'Claim Site'], dtype='object')
```

In [48]:
```
x_train_ohc = ohc.fit_transform(x_train[cat]).toarray()
x_test_ohc = ohc.transform(x_test[cat]).toarray()
```

In [49]:
```
x_test_ohc
```

Out[49]:
```
array([[1., 0., 1., 0., 0.],
       [0., 1., 1., 0., 0.],
       [0., 1., 1., 0., 0.],
       ...,
       [1., 0., 0., 0., 1.],
       [0., 1., 1., 0., 0.],
       [1., 0., 1., 0., 0.]])
```

In [50]:
```
x_train_cn = np.concatenate((x_train[num_at],x_train_ohc),axis=1)
x_test_cn = np.concatenate((x_test[num_at],x_test_ohc),axis=1)
```

In [51]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [52]:
```
y_train_le = le.fit_transform(y_train)
y_test_le = le.transform(y_test)
```

In [53]:
```
y_train_le
```

Out[53]:
```
array([1, 1, 1, ..., 1, 2, 2])
```

In [ ]:
```
```

Baseline model building - Decision Tree

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion='gini',max_depth=3)
model_dt = dt.fit(x_train_cn,y_train_le)
```

In [ ]:
```
y_pred_train_dt = model_dt.predict(x_train_cn)
y_pred_test_dt = model_dt.predict(x_test_cn)
```

In [ ]:
```
ev_metrics(y_train_le,y_pred_train_dt)
ev_metrics(y_test_le,y_pred_test_dt)
```

In [ ]:
```
cm = confusion_matrix(y_test_le,y_pred_test_dt)
cm
```