In [1]:
```
1 import pandas as pd
```

Ans1) Q1. Import the dataset and find the basic descriptive statistics by using many functions and develop insights (5)

In [2]:
```
1 df=pd.read_excel("E:\TRISEMESTER 4\\RPFM final.xlsx")
```

In [3]:
```
1 df
```

Out[3]:

	make	segment	model	fuel_type	...	is_brake_assist	is_power_door_locks	is_central_locking	is_power_steering	is_driver_seat_
)	1	A	M1	CNG	...	No	No	No	Yes	
3	1	A	M1	CNG	...	No	No	No	Yes	
5	1	A	M1	CNG	...	No	No	No	Yes	
2	1	C1	M2	Petrol	...	Yes	Yes	Yes	Yes	
3	2	A	M3	Petrol	...	No	Yes	Yes	Yes	
.	...	...	...	...	...	...	...	...	...	
4	2	A	M3	Petrol	...	No	Yes	Yes	Yes	
3	1	A	M1	CNG	...	No	No	No	Yes	
3	1	A	M1	CNG	...	No	No	No	Yes	
4	1	B2	M6	Petrol	...	Yes	Yes	Yes	Yes	
4	3	C2	M4	Diesel	...	Yes	Yes	Yes	Yes	

In [4]:

```
print(df)
```

```
      policy_id  policy_tenure  age_of_car  age_of_policyholder area_cluster  \
0       ID00001       0.515874        0.05             0.644231           C1
1       ID00002       0.672619        0.02             0.375000           C2
2       ID00003       0.841110        0.02             0.384615           C3
3       ID00004       0.900277        0.11             0.432692           C4
4       ID00005       0.596403        0.11             0.634615           C5
...         ...            ...         ...                  ...          ...
58587   ID58588       0.355089        0.13             0.644231           C8
58588   ID58589       1.199642        0.02             0.519231          C14
58589   ID58590       1.162273        0.05             0.451923           C5
58590   ID58591       1.236307        0.14             0.557692           C8
58591   ID58592       0.124429        0.02             0.442308           C8

       population_density  make segment model fuel_type  ... is_brake_assist  \
0                    4990     1       A    M1       CNG  ...              No
1                   27003     1       A    M1       CNG  ...              No
2                    4076     1       A    M1       CNG  ...              No
3                   21622     1      C1    M2    Petrol  ...             Yes
4                   34738     2       A    M3    Petrol  ...              No
...                   ...   ...     ...   ...       ...  ...             ...
58587                8794     2       A    M3    Petrol  ...              No
58588                7788     1       A    M1       CNG  ...              No
58589               34738     1       A    M1       CNG  ...              No
58590                8794     1      B2    M6    Petrol  ...             Yes
58591                8794     3      C2    M4    Diesel  ...             Yes

      is_power_door_locks is_central_locking  is_power_steering  \
0                      No                 No                Yes
1                      No                 No                Yes
2                      No                 No                Yes
3                     Yes                Yes                Yes
4                     Yes                Yes                Yes
...                   ...                ...                ...
58587                 Yes                Yes                Yes
58588                  No                 No                Yes
58589                  No                 No                Yes
58590                 Yes                Yes                Yes
58591                 Yes                Yes                Yes

      is_driver_seat_height_adjustable is_day_night_rear_view_mirror is_ecw  \
0                                   No                            No     No
1                                   No                            No     No
2                                   No                            No     No
3                                  Yes                           Yes    Yes
4                                   No                           Yes    Yes
...                                ...                           ...    ...
58587                               No                           Yes    Yes
58588                               No                            No     No
58589                               No                            No     No
58590                              Yes                           Yes    Yes
58591                              Yes                            No    Yes

      is_speed_alert ncap_rating is_claim
0                Yes           0        0
1                Yes           0        0
2                Yes           0        0
3                Yes           2        0
4                Yes           2        0
...              ...         ...      ...
58587            Yes           2        0
58588            Yes           0        0
58589            Yes           0        0
58590            Yes           2        0
58591            Yes           3        0

[58592 rows x 44 columns]
```

In [5]:
```
1 df.head()  # These are the first 5 rows of the data.
```

Out[5]:

nure	age_of_car	age_of_policyholder	area_cluster	population_density	make	segment	model	fuel_type	...	is_brake_assist	is_p
5874	0.05	0.644231	C1	4990	1	A	M1	CNG	...	No	
2619	0.02	0.375000	C2	27003	1	A	M1	CNG	...	No	
1110	0.02	0.384615	C3	4076	1	A	M1	CNG	...	No	
0277	0.11	0.432692	C4	21622	1	C1	M2	Petrol	...	Yes	
3403	0.11	0.634615	C5	34738	2	A	M3	Petrol	...	No	

In [6]:
```
1 df.tail() # These are the last 5 rows of the data.
```

Out[6]:

ıre	age_of_car	age_of_policyholder	area_cluster	population_density	make	segment	model	fuel_type	...	is_brake_assist	is_pow
89	0.13	0.644231	C8	8794	2	A	M3	Petrol	...	No	
42	0.02	0.519231	C14	7788	1	A	M1	CNG	...	No	
73	0.05	0.451923	C5	34738	1	A	M1	CNG	...	No	
07	0.14	0.557692	C8	8794	1	B2	M6	Petrol	...	Yes	
29	0.02	0.442308	C8	8794	3	C2	M4	Diesel	...	Yes	

In [8]:
```
1 df.shape   # In the data we have total 58592 number of observations in rows and 44 columns.
```

Out[8]: (58592, 44)

In [9]:
```
1 df.columns
```

Out[9]:
```
Index(['policy_id', 'policy_tenure', 'age_of_car', 'age_of_policyholder',
       'area_cluster', 'population_density', 'make', 'segment', 'model',
       'fuel_type', 'max_torque', 'max_power', 'engine_type', 'airbags',
       'is_esc', 'is_adjustable_steering', 'is_tpms', 'is_parking_sensors',
       'is_parking_camera', 'rear_brakes_type', 'displacement', 'cylinder',
       'transmission_type', 'gear_box', 'steering_type', 'turning_radius',
       'length', 'width', 'height', 'gross_weight', 'is_front_fog_lights',
       'is_rear_window_wiper', 'is_rear_window_washer',
       'is_rear_window_defogger', 'is_brake_assist', 'is_power_door_locks',
       'is_central_locking', 'is_power_steering',
       'is_driver_seat_height_adjustable', 'is_day_night_rear_view_mirror',
       'is_ecw', 'is_speed_alert', 'ncap_rating', 'is_claim'],
      dtype='object')
```

These are the names of all colunms which we have in our dataset.

In [10]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 58592 entries, 0 to 58591
Data columns (total 44 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   policy_id                         58592 non-null  object 
 1   policy_tenure                     58592 non-null  float64
 2   age_of_car                        58592 non-null  float64
 3   age_of_policyholder               58592 non-null  float64
 4   area_cluster                      58592 non-null  object 
 5   population_density                58592 non-null  int64  
 6   make                              58592 non-null  int64  
 7   segment                           58592 non-null  object 
 8   model                             58592 non-null  object 
 9   fuel_type                         58592 non-null  object 
 10  max_torque                        58592 non-null  object 
 11  max_power                         58592 non-null  object 
 12  engine_type                       58592 non-null  object 
 13  airbags                           58592 non-null  int64  
 14  is_esc                            58592 non-null  object 
 15  is_adjustable_steering            58592 non-null  object 
 16  is_tpms                           58592 non-null  object 
 17  is_parking_sensors                58592 non-null  object 
 18  is_parking_camera                 58592 non-null  object 
 19  rear_brakes_type                  58592 non-null  object 
 20  displacement                      58592 non-null  int64  
 21  cylinder                          58592 non-null  int64  
 22  transmission_type                 58592 non-null  object 
 23  gear_box                          58592 non-null  int64  
 24  steering_type                     58592 non-null  object 
 25  turning_radius                    58592 non-null  float64
 26  length                            58592 non-null  int64  
 27  width                             58592 non-null  int64  
 28  height                            58592 non-null  int64  
 29  gross_weight                      58592 non-null  int64  
 30  is_front_fog_lights               58592 non-null  object 
 31  is_rear_window_wiper              58592 non-null  object 
 32  is_rear_window_washer             58592 non-null  object 
 33  is_rear_window_defogger           58592 non-null  object 
 34  is_brake_assist                   58592 non-null  object 
 35  is_power_door_locks               58592 non-null  object 
 36  is_central_locking                58592 non-null  object 
 37  is_power_steering                 58592 non-null  object 
 38  is_driver_seat_height_adjustable  58592 non-null  object 
 39  is_day_night_rear_view_mirror     58592 non-null  object 
 40  is_ecw                            58592 non-null  object 
 41  is_speed_alert                    58592 non-null  object 
 42  ncap_rating                       58592 non-null  int64  
 43  is_claim                          58592 non-null  int64  
dtypes: float64(4), int64(12), object(28)
memory usage: 19.7+ MB
```

This is the basic info of the data which is showing which column is having which kind of data.

Int64 = complete number. Example= 4,5 float64 = Number in points which is not complete. For example= 4.5, 5.2 Object = which is text+number

In [11]:
```
# For Checking missing value
print(df.isnull().sum())
```

```
policy_id                           0
policy_tenure                       0
age_of_car                          0
age_of_policyholder                 0
area_cluster                        0
population_density                  0
make                                0
segment                             0
model                               0
fuel_type                           0
max_torque                          0
max_power                           0
engine_type                         0
airbags                             0
is_esc                              0
is_adjustable_steering              0
is_tpms                             0
is_parking_sensors                  0
is_parking_camera                   0
rear_brakes_type                    0
displacement                        0
cylinder                            0
transmission_type                   0
gear_box                            0
steering_type                       0
turning_radius                      0
length                              0
width                               0
height                              0
gross_weight                        0
is_front_fog_lights                 0
is_rear_window_wiper                0
is_rear_window_washer               0
is_rear_window_defogger             0
is_brake_assist                     0
is_power_door_locks                 0
is_central_locking                  0
is_power_steering                   0
is_driver_seat_height_adjustable    0
is_day_night_rear_view_mirror       0
is_ecw                              0
is_speed_alert                      0
ncap_rating                         0
is_claim                            0
dtype: int64
```

This is showing that we don't have any null value in our data set.

In [12]:
```
df.describe()
```

Out[12]:

n_density	make	airbags	displacement	cylinder	gear_box	turning_radius	length	width	
92.000000	58592.000000	58592.000000	58592.000000	58592.000000	58592.000000	58592.000000	58592.000000	58592.000000	5859
26.858667	1.763722	3.137066	1162.355851	3.626963	5.245443	4.852893	3850.476891	1672.233667	155
60.174792	1.136988	1.832641	266.304786	0.483616	0.430353	0.228061	311.457119	112.089135	7
90.000000	1.000000	1.000000	796.000000	3.000000	5.000000	4.500000	3445.000000	1475.000000	147
12.000000	1.000000	2.000000	796.000000	3.000000	5.000000	4.600000	3445.000000	1515.000000	147
94.000000	1.000000	2.000000	1197.000000	4.000000	5.000000	4.800000	3845.000000	1735.000000	153
03.000000	3.000000	6.000000	1493.000000	4.000000	5.000000	5.000000	3995.000000	1755.000000	163
30.000000	5.000000	6.000000	1498.000000	4.000000	6.000000	5.200000	4300.000000	1811.000000	182

In [13]:

```
print(df.describe())
```

```
       policy_tenure    age_of_car  age_of_policyholder  population_density  \
count   58592.000000  58592.000000         58592.000000        58592.000000
mean        0.611246      0.069424             0.469420        18826.858667
std         0.414156      0.056721             0.122886        17660.174792
min         0.002735      0.000000             0.288462          290.000000
25%         0.210250      0.020000             0.365385         6112.000000
50%         0.573792      0.060000             0.451923         8794.000000
75%         1.039104      0.110000             0.548077        27003.000000
max         1.396641      1.000000             1.000000        73430.000000

                make       airbags  displacement      cylinder      gear_box  \
count  58592.000000  58592.000000  58592.000000  58592.000000  58592.000000
mean       1.763722      3.137066   1162.355851      3.626963      5.245443
std        1.136988      1.832641    266.304786      0.483616      0.430353
min        1.000000      1.000000    796.000000      3.000000      5.000000
25%        1.000000      2.000000    796.000000      3.000000      5.000000
50%        1.000000      2.000000   1197.000000      4.000000      5.000000
75%        3.000000      6.000000   1493.000000      4.000000      5.000000
max        5.000000      6.000000   1498.000000      4.000000      6.000000

       turning_radius        length         width        height  gross_weight  \
count    58592.000000  58592.000000  58592.000000  58592.00000  58592.000000
mean         4.852893   3850.476891   1672.233667   1553.33537   1385.276813
std          0.228061    311.457119    112.089135     79.62227    212.423085
min          4.500000   3445.000000   1475.000000   1475.00000   1051.000000
25%          4.600000   3445.000000   1515.000000   1475.00000   1185.000000
50%          4.800000   3845.000000   1735.000000   1530.00000   1335.000000
75%          5.000000   3995.000000   1755.000000   1635.00000   1510.000000
max          5.200000   4300.000000   1811.000000   1825.00000   1720.000000

        ncap_rating      is_claim
count  58592.000000  58592.000000
mean       1.759950      0.063968
std        1.389576      0.244698
min        0.000000      0.000000
25%        0.000000      0.000000
50%        2.000000      0.000000
75%        3.000000      0.000000
max        5.000000      1.000000
```

```
This is showing the basic descriptive statistics of the data.
count is for total number of observation.
mean is for mean(average) of data.
std is for standard devation which is showing that how much variance in the data from the mean value.
min is for smallest value.
max is for the largest value.
25% is 1 quartile.
50% is half of the data.
75% is 1/3 of data.

```

Ans3. Develop predictive model using any classificication machine learning technique then validate the model by splitting the data into training and testing (15)

In [26]:

```
#Libraray- Lable encoder
from sklearn.preprocessing import LabelEncoder
le_is_front_fog_lights=LabelEncoder()
le_is_central_locking=LabelEncoder()

input=df.drop([ 'is_claim'],axis='columns')

```

In [27]:
```
1 input.columns
```

Out[27]:
```
Index(['policy_id', 'policy_tenure', 'age_of_car', 'age_of_policyholder',
       'area_cluster', 'population_density', 'make', 'segment', 'model',
       'fuel_type', 'max_torque', 'max_power', 'engine_type', 'airbags',
       'is_esc', 'is_adjustable_steering', 'is_tpms', 'is_parking_sensors',
       'is_parking_camera', 'rear_brakes_type', 'displacement', 'cylinder',
       'transmission_type', 'gear_box', 'steering_type', 'turning_radius',
       'length', 'width', 'height', 'gross_weight', 'is_front_fog_lights',
       'is_rear_window_wiper', 'is_rear_window_washer',
       'is_rear_window_defogger', 'is_brake_assist', 'is_power_door_locks',
       'is_central_locking', 'is_power_steering',
       'is_driver_seat_height_adjustable', 'is_day_night_rear_view_mirror',
       'is_ecw', 'is_speed_alert', 'ncap_rating'],
      dtype='object')
```

In [28]:
```
1 output=df.is_claim
```

In [51]:
```
1 input['is_front_fog_lights_n'] = le_Department.fit_transform(input['is_front_fog_lights'])
2 input['is_central_locking_n'] = le_salary.fit_transform(input['is_central_locking'])
```

In [67]:
```
1 input
```

Out[67]:

s	is_central_locking	is_power_steering	is_driver_seat_height_adjustable	is_day_night_rear_view_mirror	is_ecw	is_speed_alert	nca
o	No	Yes	No	No	No	Yes	
o	No	Yes	No	No	No	Yes	
o	No	Yes	No	No	No	Yes	
s	Yes	Yes	Yes	Yes	Yes	Yes	
s	Yes	Yes	No	Yes	Yes	Yes	
..	...	...	...	...	...	...	
s	Yes	Yes	No	Yes	Yes	Yes	
o	No	Yes	No	No	No	Yes	
o	No	Yes	No	No	No	Yes	
s	Yes	Yes	Yes	Yes	Yes	Yes	
s	Yes	Yes	Yes	No	Yes	Yes	

In [68]:
```
1 input_n =input.drop([ 'is_front_fog_lights', 'is_central_locking'],axis='columns')
2 input_n.columns
3
```

Out[68]:
```
Index(['policy_id', 'policy_tenure', 'age_of_car', 'age_of_policyholder',
       'area_cluster', 'population_density', 'make', 'segment', 'model',
       'fuel_type', 'max_torque', 'max_power', 'engine_type', 'airbags',
       'is_esc', 'is_adjustable_steering', 'is_tpms', 'is_parking_sensors',
       'is_parking_camera', 'rear_brakes_type', 'displacement', 'cylinder',
       'transmission_type', 'gear_box', 'steering_type', 'turning_radius',
       'length', 'width', 'height', 'gross_weight', 'is_rear_window_wiper',
       'is_rear_window_washer', 'is_rear_window_defogger', 'is_brake_assist',
       'is_power_door_locks', 'is_power_steering',
       'is_driver_seat_height_adjustable', 'is_day_night_rear_view_mirror',
       'is_ecw', 'is_speed_alert', 'ncap_rating', 'is_front_fog_lights_n',
       'is_central_locking_n'],
      dtype='object')
```

In [69]:
```
input_n_ =input.drop([ 'policy_id', 'policy_tenure', 'age_of_car', 'age_of_policyholder',
       'area_cluster', 'population_density', 'make', 'segment', 'model',
       'fuel_type', 'max_torque', 'max_power', 'engine_type',
       'is_esc', 'is_adjustable_steering', 'is_tpms', 'is_parking_sensors',
       'is_parking_camera', 'rear_brakes_type', 'displacement',
       'transmission_type', 'gear_box', 'steering_type', 'turning_radius',
       'length', 'width', 'height', 'gross_weight', 'is_rear_window_wiper',
       'is_rear_window_washer', 'is_rear_window_defogger', 'is_brake_assist',
       'is_power_door_locks', 'is_power_steering',
       'is_driver_seat_height_adjustable', 'is_day_night_rear_view_mirror',
       'is_ecw', 'is_speed_alert'],axis='columns')
input_n_.columns

```

Out[69]:
```
Index(['airbags', 'cylinder', 'is_front_fog_lights', 'is_central_locking',
       'ncap_rating', 'is_front_fog_lights_n', 'is_central_locking_n'],
      dtype='object')
```

In [70]:
```
Input =input_n_.drop(['is_front_fog_lights','is_central_locking' ],axis='columns')
```

In [72]:
```
Input
```

Out[72]:

	airbags	cylinder	ncap_rating	is_front_fog_lights_n	is_central_locking_n
0	2	3	0	0	0
1	2	3	0	0	0
2	2	3	0	0	0
3	2	4	2	1	1
4	2	3	2	0	1
...	...	...	...	...	...
58587	2	3	2	0	1
58588	2	3	0	0	0
58589	2	3	0	0	0
58590	2	4	2	1	1
58591	6	4	3	1	1

58592 rows × 5 columns

In [58]:
```
X=Input
y=output
```

In [59]:
```
#Train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.7)
```

Here we have take the input range which is X have variable under it ('airbags', 'cylinder', 'ncap_rating', 'is_front_fog_lights_n', 'is_central_locking_n')

For output range we have taken (is_claim)

After this i have divie the model in 70% and 30%. In this i will give 70% of the data to the model and model will predict rest of 30% data. We will train th model by 70% and take the result of rest 30% of the data. " #70% use to build the model and rest 30% is to test/ validate the modelX_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.7) #70% use to build the model and rest 30% is to test/ validate the model"

In [60]:
```
from sklearn.linear_model import LogisticRegression
model= LogisticRegression()
```

In [61]:
```
model.fit(X_train,y_train)
```

Out[61]:
```
LogisticRegression
LogisticRegression()
```

In [78]:

```
model.predict(X_test)
```

Out[78]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [84]:

```
model.score(X_test,y_test)
```

Out[84]:

```
0.9376493343952668
```

Ans4) Find the accuracy rate of the model (5)

In [85]:

```
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix,roc_curve,auc
```

In [86]:

```
#Evaluate the model
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:{:.2f}%".format(accuracy * 100))
```

```
Accuracy:93.76%
```

Here the accuracy score of my model is 93.76% which mean tha our model is able to predict the data with very high accuracy,the chances of wrong prediction is less because the accuracy is very high.

Ans5) Explain the confusion matrix the model

In [89]:

```
# evaluate the model
print("confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nclassification Report: \n", classification_report(y_test, y_pred))
```

```
confusion Matrix:
 [[16482     0]
 [ 1096     0]]

classification Report: 
               precision    recall  f1-score   support

           0       0.94      1.00      0.97     16482
           1       0.00      0.00      0.00      1096

    accuracy                           0.94     17578
   macro avg       0.47      0.50      0.48     17578
weighted avg       0.88      0.94      0.91     17578


C:\Users\Gaurav Singh Rawat\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Gaurav Singh Rawat\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Gaurav Singh Rawat\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Here we have the confusion metrics which showing that model is able to predict: 16482 as true negative which mean that 16482 are correct predict by model which doesn't raise claim. 1096 as false negative wich mean that 1096 are incorreclty prect by model which has raise the claim but modek predict that they didn't raise the claim. 0 as false positive which mean model have predict 0 as those who have not raise the claim and correctly predict as per the data. 0 as true positive which mean model have predict 0 as those who raise the claim and correctly predict as per the data.

Precision= It is showing the positively predict data with total number of positive data. Recall= It show that positively predict data with total number of correctly data whether it is true negative or true positive.

Ans 2) Import both the libraries matplotlib and seaborn to plot the cross tabulations and heatmap of the categorical and ration data and mention your insights

In [90]:
```
1 import seaborn as sns
```

In [91]:
```
1 import matplotlib as mtlyb
```

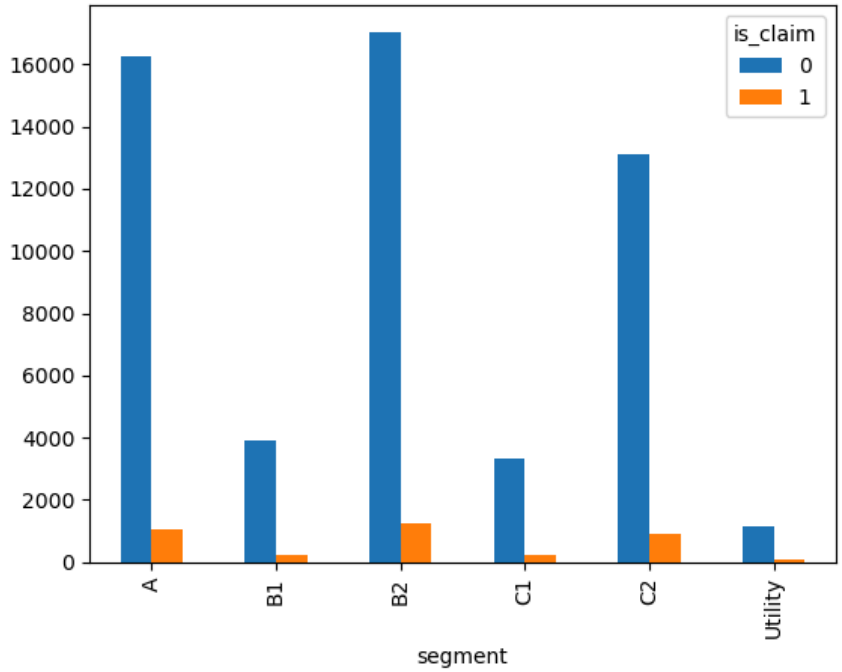
In [92]:
```
1 pd.crosstab(df.segment,df.is_claim)
```

Out[92]:

is_claim	0	1
segment		
A	16275	1046
B1	3929	244
B2	17058	1256
C1	3329	228
C2	13117	901
Utility	1136	73

In [93]:
```
1 pd.crosstab(df.segment,df.is_claim).plot(kind='bar')
```

Out[93]: <Axes: xlabel='segment'>



It is showing the cross tabulation of different segment with response to in which segment how many have raise the claim 1 is for claim 0 is for Not raise the claim In segment Utility minimum of claim has been raised.Maximum claim has been raised in B2.

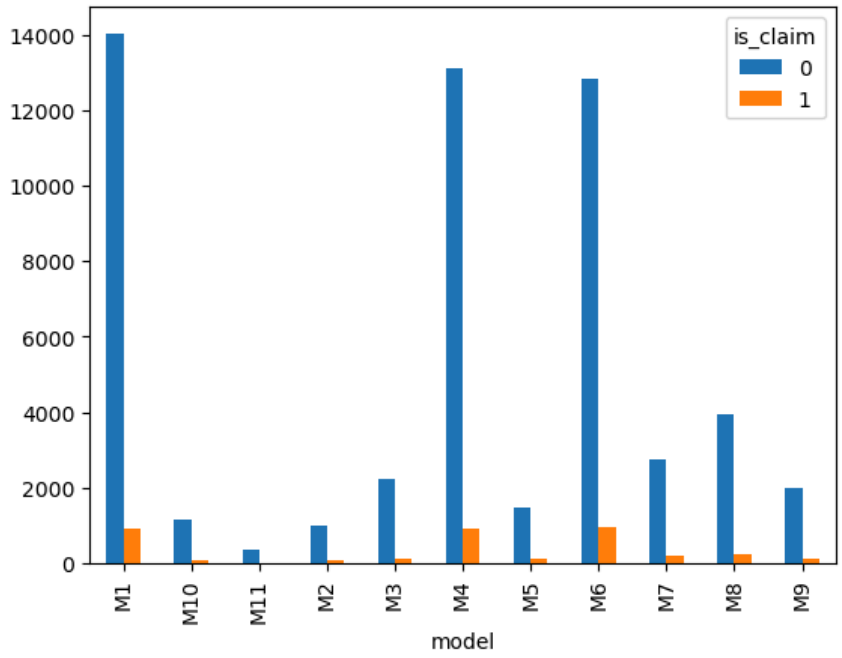
In [97]:
```
1 pd.crosstab(df.model,df.is_claim)
```

Out[97]:

is_claim	0	1
model		
M1	14030	918
M10	1136	73
M11	348	15
M2	1000	80
M3	2245	128
M4	13117	901
M5	1482	116
M6	12837	939
M7	2739	201
M8	3929	244
M9	1981	133

In [98]:
```
1 pd.crosstab(df.model,df.is_claim).plot(kind='bar')
```

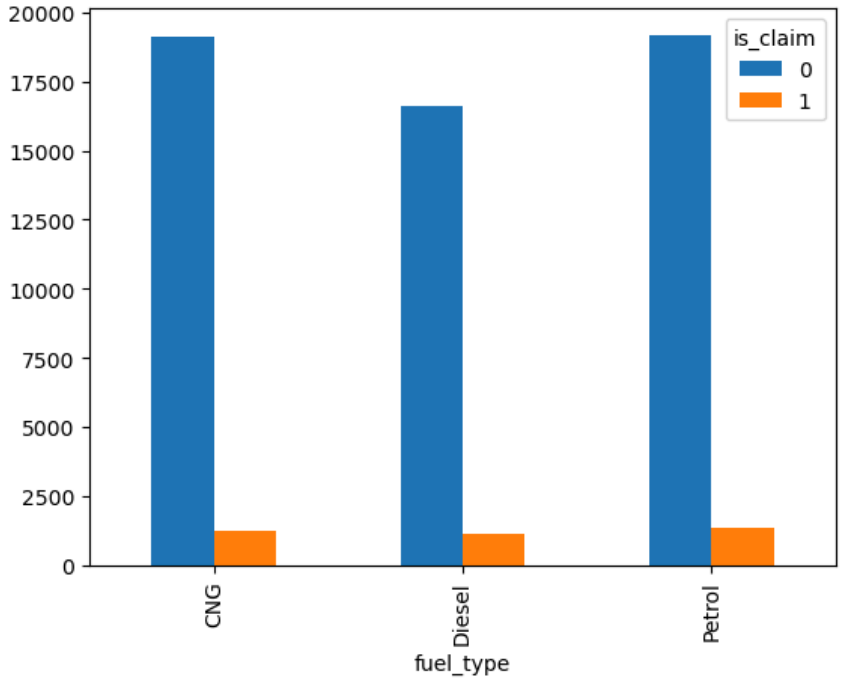
Out[98]: <Axes: xlabel='model'>




```
1 Here it is showimg the claim response on the basis of model in this max claim is by M1 and min is in
  M11.
```

In [100]:
```
pd.crosstab(df.fuel_type,df.is_claim)
pd.crosstab(df.fuel_type,df.is_claim).plot(kind='bar')
```

Out[100]: <Axes: xlabel='fuel_type'>




```
Here it is showimg the claim response on the basis of Fuel type in this max claim is of petrol and min is in Diesel.
```

In [107]:
```
# Compute correlation matrix
corr = df[['age_of_car', 'age_of_policyholder', 'population_density','is_claim']].corr()
```

In [108]:
```
# Heatmap of the correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix')
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[108], line 2
      1 # Heatmap of the correlation matrix
----> 2 plt.figure(figsize=(10, 8))
      3 sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f')
      4 plt.title('Correlation Matrix')

NameError: name 'plt' is not defined
```

In [ ]:
```
```