EDA - Análisis exploratorio.

Genera data_df.csv para que lo levante ML.

In [76]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [77]:
```
# Lectura del dataset generado en el proceso de ETL.
# El dataset en Python lo voy a manejar en un dataframe de Pandas.
df = pd.read_csv('C:\\Trabajo\\Procesos\\TFI\\dataset.csv', sep=';', low_memory=False, encoding
```

In [78]:
```
# Filas y columnas del dataframe.
df.shape
```

Out[78]: (469370, 24)

In [79]:
```
# Muestro 5 primeros registros del dataframe para realizar un análisis superficial del mismo.
pd.options.display.max_columns=None
df.head(5)
```

Out[79]:

	pol	endoso	item	tipend	idaseg	ase_antig_an	ase_cp	ase_prof	ase_codnac	ase_nac	idprod
0	200000	0	1	EMISION	4508	6	1640	NaN	200.0	ARGENTINA	1521
1	200001	0	1	EMISION	5350	6	1678	NaN	205.0	COLOMBIA	5001
2	200001	1	1	ENDOSO DATOS GENERALES	5350	6	1678	NaN	205.0	COLOMBIA	5001
3	200001	2	1	ENDOSO CAMBIO COBERTURA	5350	6	1678	NaN	205.0	COLOMBIA	5001
4	200004	0	1	EMISION	7618	3	1406	NaN	205.0	COLOMBIA	1527

In [80]:
```
# Muestro diseño del dataframe, nombre de columnas, cuenta de NA y tipo de dato.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 469370 entries, 0 to 469369
Data columns (total 24 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   pol           469370 non-null  int64
 1   endoso        469370 non-null  int64
 2   item          469370 non-null  int64
 3   tipend        469370 non-null  object
 4   idaseg        469370 non-null  int64
 5   ase_antig_an  469370 non-null  int64
 6   ase_cp        469370 non-null  int64
```

```
 7   ase_prof      0 non-null       float64
 8   ase_codnac    351480 non-null  float64
 9   ase_nac       351480 non-null  object
 10  idprod        469370 non-null  int64
 11  pro_antig_an  469370 non-null  int64
 12  tipveh        469370 non-null  int64
 13  marca         469370 non-null  int64
 14  modelo        469370 non-null  int64
 15  anio          469370 non-null  int64
 16  uso           469370 non-null  int64
 17  uso_desc      469370 non-null  object
 18  cober         469370 non-null  object
 19  codcober      469370 non-null  int64
 20  cober_desc    469370 non-null  object
 21  cob_fecuma    451613 non-null  object
 22  cob_ef        207224 non-null  object
 23  fraude        469370 non-null  int64
dtypes: float64(2), int64(15), object(7)
memory usage: 85.9+ MB
```

In [81]:

```
# Descripción del dataset de entrada.

# Column        Description
# ------        -----------
# pol           Número de póliza.
# endoso        Número de endoso a póliza.
# item          Identificador del Vehículo dentro de la Póliza.
# tipend        Tipo de Endoso.
# idaseg        ID del Asegurado.
# ase_antig_an  Antiguedad del Asegurado. Viene medida en años del proceso de ETL.
# ase_cp        CP del domicilio del Asegurado
# ase_prof      Profesión del Asegurado
# ase_codnac    Código de Pais de la Nacionalidad del Asegurado
# ase_nac       Nacionalidad del Asegurado
# idprod        ID del Productor
# pro_antig_an  Antiguedad del Productor. Viene medida en años del proceso de ETL.
# tipveh        Tipo de Vehículo
# marca         Marca del Vehículo
# modelo        Modelo del Vehículo
# anio          Año de Fabricación del Vehículo
# uso           Uso del Vehículo
# uso_desc      Descripción
# cober         Cobertura del Seguro
# codcober      Código de Cobertura del Seguro
# cober_desc    Descripción de la Cobertura
# cob_fecuma    Fecha de la última Cobranza
# cob_ef        Tiene Cobranza en Efectivo (S/N)
# fraude        Variable Objetivo. La póliza tuvo siniestros considerados como fraude por Expe
```

In [82]:

```
# Otra forma de ver las columnas del dataframe.
df.columns
```

Out[82]:

```
Index(['pol', 'endoso', 'item', 'tipend', 'idaseg', 'ase_antig_an', 'ase_cp',
       'ase_prof', 'ase_codnac', 'ase_nac', 'idprod', 'pro_antig_an', 'tipveh',
       'marca', 'modelo', 'anio', 'uso', 'uso_desc', 'cober', 'codcober',
       'cober_desc', 'cob_fecuma', 'cob_ef', 'fraude'],
      dtype='object')
```

In [83]:

```
# Chequeo de valores faltantes en el dataframe.
df.isnull()
```

Out[83]:

	pol	endoso	item	tipend	idaseg	ase_antig_an	ase_cp	ase_prof	ase_codnac	ase_nac	idprod	pro_
0	False	False	False	False	False	False	False	True	False	False	False	
1	False	False	False	False	False	False	False	True	False	False	False	
2	False	False	False	False	False	False	False	True	False	False	False	

	pol	endoso	item	tipend	idaseg	ase_antig_an	ase_cp	ase_prof	ase_codnac	ase_nac	idprod	pro_
3	False	False	False	False	False	False	False	True	False	False	False	
4	False	False	False	False	False	False	False	True	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	...	
469365	False	False	False	False	False	False	False	True	True	True	False	
469366	False	False	False	False	False	False	False	True	True	True	False	
469367	False	False	False	False	False	False	False	True	False	False	False	
469368	False	False	False	False	False	False	False	True	True	True	False	
469369	False	False	False	False	False	False	False	True	False	False	False	

469370 rows × 24 columns

In [84]:
```
# Chequeo de valores faltantes en el dataframe.
df.isnull().sum() # True == 1 y False == 0
```

Out[84]:
```
pol                 0
endoso              0
item                0
tipend              0
idaseg              0
ase_antig_an        0
ase_cp              0
ase_prof       469370
ase_codnac     117890
ase_nac        117890
idprod              0
pro_antig_an        0
tipveh              0
marca               0
modelo              0
anio                0
uso                 0
uso_desc            0
cober               0
codcober            0
cober_desc          0
cob_fecuma      17757
cob_ef         262146
fraude              0
dtype: int64
```

In [85]:
```
# 'ase_prof'   tiene NA en todos los registros, no la tendremos en cuenta para el proceso, así
# 'ase_codnac' idem 'ase_nac'
# 'ase_nac'    tiene NA en un 25% de los registro, consultada la parte comercial operativa
#              de la Aseguradora, nos indican que cuando no tienen la nacionalidad se asume 'Ar
# 'cob_fecuma' tiene valores NA y es razonable que así sea, son operaciones que aún no tuvieror
# 'cob_ef'     es una variable categórica ('S'/'N'), por lo analizado de sus valores, cuando es
#              tiene ese valor, pero cuando es 'N' tiene NA. Deberemos arreglar el contenido de
```

In [86]:
```
# formas de resolver los registros con variables en nulo.
# para una variable continua podría encontrar el valor modal y usarlo para rellenar.
# xvar_mode = df.xvar.mode()[0]
# df.xvar.fillna(xvar_mode, inplace = True)
# df.xvar.isnull().sum()
```

In [87]:
```
# para el caso de las columnas 'cob_ef' y 'ase_nac'/'ase_codnac'
df.cob_ef.fillna('N', inplace=True)
```

```
df.ase_nac.fillna('ARGENTINA', inplace=True)
df.ase_codnac.fillna(200, inplace=True)
```

In [88]:

```
# Chequeo de valores faltantes en el dataframe.
df.isnull().sum() # True == 1 y False == 0
```

Out[88]:

```
pol                 0
endoso              0
item                0
tipend              0
idaseg              0
ase_antig_an        0
ase_cp              0
ase_prof       469370
ase_codnac          0
ase_nac             0
idprod              0
pro_antig_an        0
tipveh              0
marca               0
modelo              0
anio                0
uso                 0
uso_desc            0
cober               0
codcober            0
cober_desc          0
cob_fecuma      17757
cob_ef              0
fraude              0
dtype: int64
```

In [89]:

```
# para la columna 'ase_prof', la decisión es eliminarla del dataframe.
if 'ase_prof' in df.columns:
    del df['ase_prof']
```

In [90]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 469370 entries, 0 to 469369
Data columns (total 23 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   pol           469370 non-null  int64
 1   endoso        469370 non-null  int64
 2   item          469370 non-null  int64
 3   tipend        469370 non-null  object
 4   idaseg        469370 non-null  int64
 5   ase_antig_an  469370 non-null  int64
 6   ase_cp        469370 non-null  int64
 7   ase_codnac    469370 non-null  float64
 8   ase_nac       469370 non-null  object
 9   idprod        469370 non-null  int64
 10  pro_antig_an  469370 non-null  int64
 11  tipveh        469370 non-null  int64
 12  marca         469370 non-null  int64
 13  modelo        469370 non-null  int64
 14  anio          469370 non-null  int64
 15  uso           469370 non-null  int64
 16  uso_desc      469370 non-null  object
 17  cober         469370 non-null  object
 18  codcober      469370 non-null  int64
 19  cober_desc    469370 non-null  object
 20  cob_fecuma    451613 non-null  object
 21  cob_ef        469370 non-null  object
 22  fraude        469370 non-null  int64
dtypes: float64(1), int64(15), object(7)
memory usage: 82.4+ MB
```

Análisis exploratorio univariable.

In [91]:
```
# Utilizo primero el método describe del df entero para mostrar las estadísticas descriptivas i
# media, meadiana, máx, mín, std, etc. El método describe solo regresa los valores de estas est
# para las columnas numéricas (Ssalvo que se instruya al método con el argumento "include='all
print(df.describe())
```

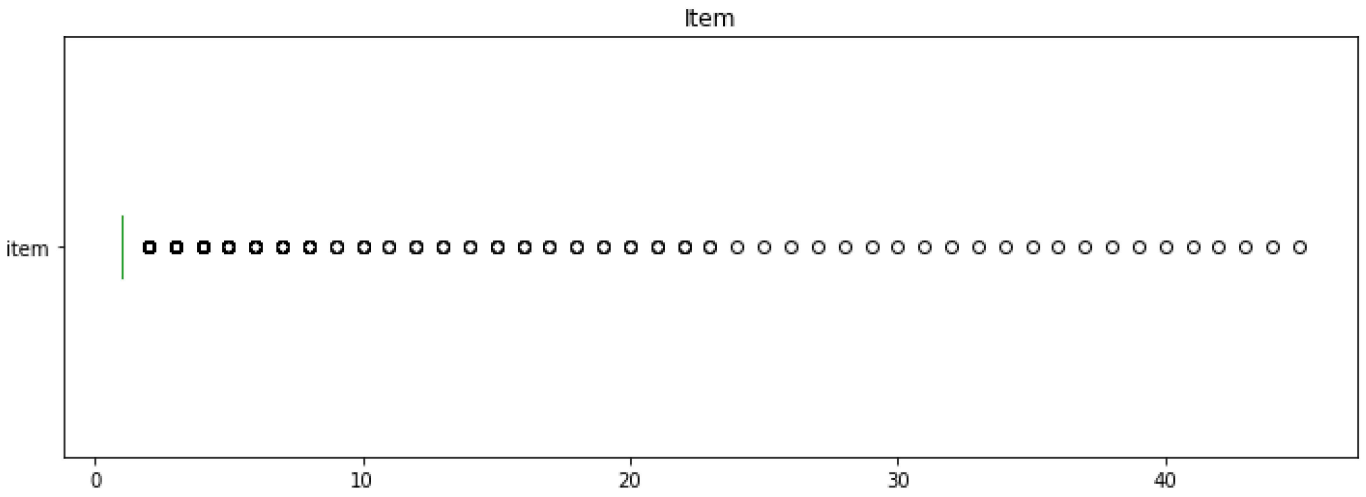
```
                pol          endoso            item        idaseg  \
count  469370.000000  469370.000000  469370.000000  4.693700e+05
mean   434654.229388       0.001300       1.012576  1.155120e+07
std    135460.995433       0.075156       0.309774  1.046871e+07
min    200000.000000       0.000000       1.000000  2.000000e+00
25%    317344.250000       0.000000       1.000000  4.772250e+05
50%    434686.500000       0.000000       1.000000  1.515232e+07
75%    551941.750000       0.000000       1.000000  1.536369e+07
max    669277.000000       8.000000      45.000000  3.597326e+07

         ase_antig_an         ase_cp     ase_codnac         idprod  \
count  469370.000000  469370.000000  469370.000000  469370.000000
mean        3.022564    1523.887046     200.028257    2233.585542
std         1.736485    1095.782110       0.569121    1389.886033
min         0.000000    1001.000000     200.000000       1.000000
25%         2.000000    1001.000000     200.000000    1362.000000
50%         2.000000    1001.000000     200.000000    1735.000000
75%         4.000000    1708.000000     200.000000    2107.000000
max        32.000000    9420.000000     310.000000    5005.000000

        pro_antig_an         tipveh          marca         modelo  \
count  469370.000000  469370.000000  469370.000000  469370.000000
mean        9.708938       1.907193      37.424358     434.674807
std         2.256888       1.178078      20.413349     261.009605
min         4.000000       1.000000       2.000000       7.000000
25%        10.000000       1.000000      25.000000     268.000000
50%        11.000000       1.000000      26.000000     304.000000
75%        11.000000       3.000000      57.000000     658.000000
max        12.000000       9.000000      78.000000    1000.000000

               anio            uso       codcober         fraude
count  469370.000000  469370.000000  469370.000000  469370.000000
mean     2000.566114       1.211358      18.021073       0.036722
std         7.087862       0.766356      15.482007       0.188078
min      1990.000000       1.000000       1.000000       0.000000
25%      1995.000000       1.000000       2.000000       0.000000
50%      2000.000000       1.000000      30.000000       0.000000
75%      2004.000000       1.000000      32.000000       0.000000
max      2018.000000       7.000000      43.000000       1.000000
```

Algunas variables de interés para analizar.

Variable 'item'

In [92]:
```
df['item'].plot(title='Item', kind='box', vert=False, figsize=[12, 4])
```

Out[92]:
```
<AxesSubplot:title={'center':'Item'}>
```



In [93]:
```
# Busco el valor Modal para entender los outliers.
df['item'].mode()[0]
```

Out[93]: 1

In [94]:
```
# Se detectan valores atípicos (outliers) superiores.
# Filtrar para verlos y determinar que son, si son válidos y como tratarlos.
filtro=df['item']>1
df[filtro].sort_values(by='item', ascending=False)
```

Out[94]:

	pol	endoso	item	tipend	idaseg	ase_antig_an	ase_cp	ase_codnac	ase_nac	idprod	pro_
323607	523522	0	45	EMISION	15131205	4	1646	200.0	ARGENTINA	1875	
323606	523522	0	44	EMISION	15131205	4	1646	200.0	ARGENTINA	1875	
323605	523522	0	43	EMISION	15131205	4	1646	200.0	ARGENTINA	1875	
323604	523522	0	42	EMISION	15131205	4	1646	200.0	ARGENTINA	1875	
323603	523522	0	41	EMISION	15131205	4	1646	200.0	ARGENTINA	1875	
...	...	...	...	...	...	...	...	...	...	...	...
179828	379830	0	2	EMISION	15168524	2	1001	200.0	ARGENTINA	1762	
180025	380027	0	2	EMISION	15330890	2	1001	200.0	ARGENTINA	1104	
180435	380437	0	2	EMISION	73848	2	1001	200.0	ARGENTINA	1255	
180577	380579	0	2	EMISION	235795	6	1646	200.0	ARGENTINA	5005	

	pol	endoso	item	tipend	idaseg	ase_antig_an	ase_cp	ase_codnac	ase_nac	idprod	pro_
469008	668916	0	2	EMISION	15345687	2	1001	200.0	ARGENTINA	1246	

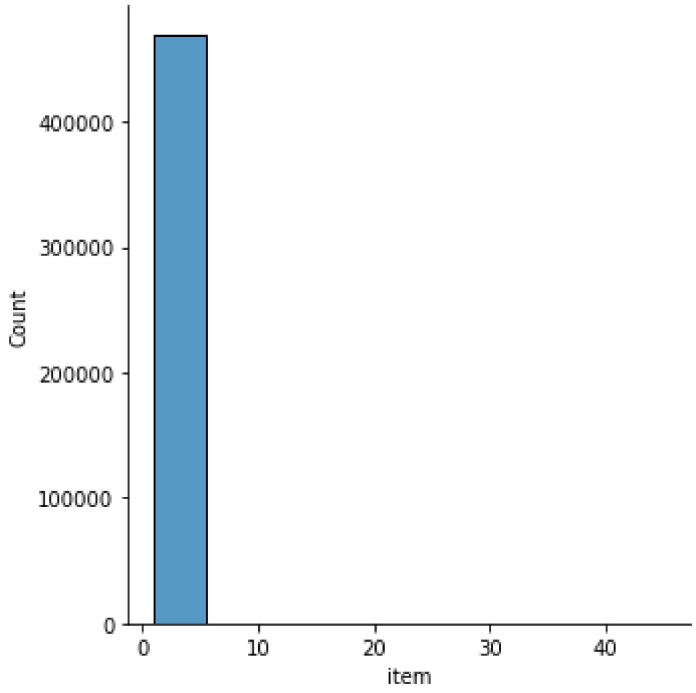
3376 rows × 23 columns

In [95]:
```
# Reviso la frecuencia de cada valor.
df.groupby('item')['item'].count()
```

Out[95]:
```
item
1     465994
2       2580
3        494
4        136
5         44
6         21
7         18
8         10
9          4
10         5
11         3
12         4
13         4
14         3
15         4
16         4
17         3
18         3
19         3
20         3
21         3
22         3
23         2
24         1
25         1
26         1
27         1
28         1
29         1
30         1
31         1
32         1
33         1
34         1
35         1
36         1
37         1
38         1
39         1
40         1
41         1
42         1
43         1
44         1
45         1
Name: item, dtype: int64
```

In [96]:
```
# Plot de Distribución.
sns.displot(df['item'], bins=10)
```

Out[96]:
```
<seaborn.axisgrid.FacetGrid at 0x27db52deee0>
```



In [97]:
```
# Se comprueba que la mayoría de las operaciones son individuales (de un solo vehículo).
# Se verifica también que la operación que tiene 45 items es una póliza flota. Lo mismo las dem
# más de un item. Se dejan los registros intactos.
```

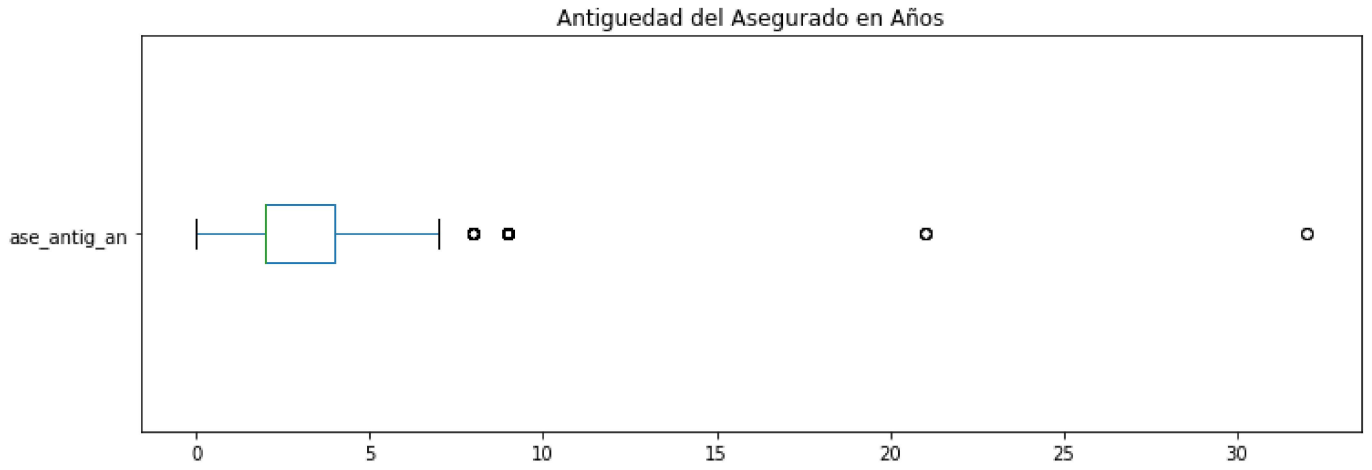
Variable 'ase_antig_an'

In [98]:
```
df['ase_antig_an'].describe()
```

Out[98]:
```
count    469370.000000
mean          3.022564
std           1.736485
min           0.000000
25%           2.000000
50%           2.000000
75%           4.000000
max          32.000000
Name: ase_antig_an, dtype: float64
```

In [99]:
```
df['ase_antig_an'].plot(title='Antiguedad del Asegurado en Años', kind='box', vert=False, figsi
```

Out[99]:
```
<AxesSubplot:title={'center':'Antiguedad del Asegurado en Años'}>
```



In [100…
```
# Se detectan valores atípicos (outliers) superiores.
# Filtrar para verlos y determinar que son, si son válidos o si es necesario eliminar las filas
filtro=df['ase_antig_an']>=15
df[filtro]
```

Out[100…

	pol	endoso	item	tipend	idaseg	ase_antig_an	ase_cp	ase_codnac	ase_nac	idprod	pro_
44010	244012	0	1	EMISION	35957109	21	1646	200.0	ARGENTINA	1428	
59155	259157	0	1	EMISION	35880056	32	1619	200.0	ARGENTINA	2001	
60350	260352	0	1	EMISION	35880056	32	1619	200.0	ARGENTINA	1103	
172804	372806	0	1	EMISION	35957109	21	1646	200.0	ARGENTINA	1476	
228095	428097	0	1	EMISION	15357066	21	1605	200.0	ARGENTINA	1361	
380678	580593	0	1	EMISION	15357066	21	1605	200.0	ARGENTINA	2013	

In [101…

```
# Se comprueba que corresponden a 3 clientes.
# Códigos 35957109, 35880056 y 15357066. Se verifica con el área Comercial de la Aseguradora y
# los Asegurados es correcta. Se dejan los registros intactos.

# Otra forma de encontrar los outliers superiores:
# df['ase_antig_an'].nlargest(10)
# df.nlargest(10, 'ase_antig_an') # 10 registros con Asegurados de mayor antiguedad.
# df['ase_antig_an'].nsmallest(5)
```

In [102…

```
# Otra forma, ordenar datos para ver valores mas altos (o mas bajos).
df.sort_values(by='ase_antig_an', ascending=False) # con inplace=True grabo el df en ese orden.
```

Out[102…

	pol	endoso	item	tipend	idaseg	ase_antig_an	ase_cp	ase_codnac	ase_nac	idprod	pro_
59155	259157	0	1	EMISION	35880056	32	1619	200.0	ARGENTINA	2001	
60350	260352	0	1	EMISION	35880056	32	1619	200.0	ARGENTINA	1103	
380678	580593	0	1	EMISION	15357066	21	1605	200.0	ARGENTINA	2013	
228095	428097	0	1	EMISION	15357066	21	1605	200.0	ARGENTINA	1361	
44010	244012	0	1	EMISION	35957109	21	1646	200.0	ARGENTINA	1428	
…	…	…	…	…	…	…	…	…	…	…	…

	pol	endoso	item	tipend	idaseg	ase_antig_an	ase_cp	ase_codnac	ase_nac	idprod	pro_
427703	627618	0	1	EMISION	15421626	0	8332	200.0	ARGENTINA	1522	
348201	548116	0	1	EMISION	15409249	0	1714	200.0	ARGENTINA	1565	
270076	470057	0	1	EMISION	15426771	0	1651	200.0	ARGENTINA	1024	
348205	548120	0	1	EMISION	15410986	0	1888	200.0	ARGENTINA	1817	
286182	486141	0	1	EMISION	15394299	0	8000	200.0	ARGENTINA	1787	

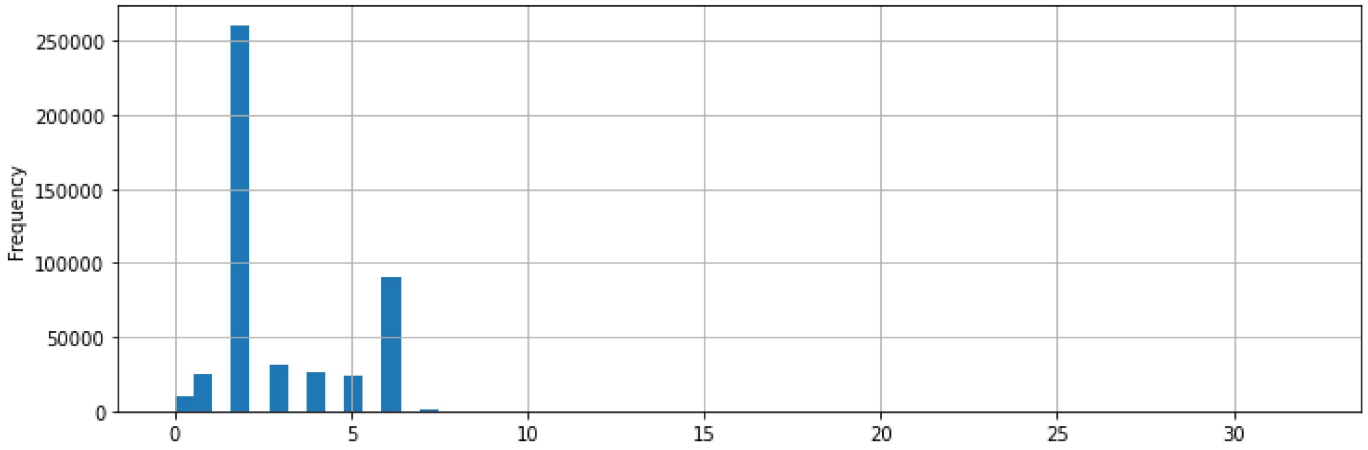
469370 rows × 23 columns

In [103…

```
# Histograma para ver la frecuencia de cada valor.
df['ase_antig_an'].plot(kind='hist', bins=60, grid=True, figsize=[12,4])
```

Out[103…

```
<AxesSubplot:ylabel='Frequency'>
```



In [104…

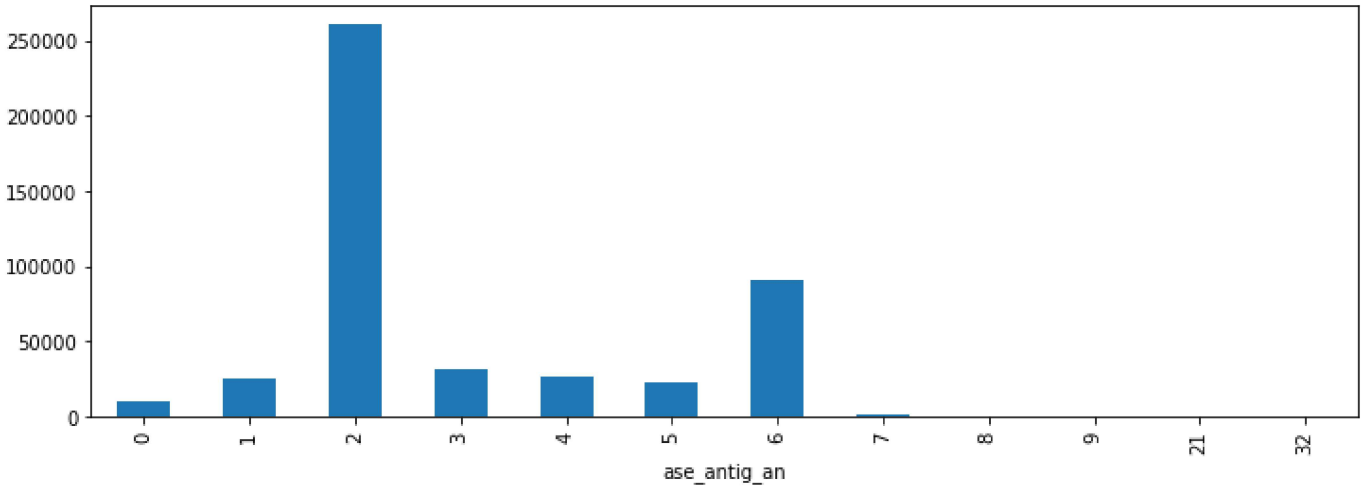
```
# Otra forma de ver la frecuencia de cada valor.
df.groupby('ase_antig_an')['ase_antig_an'].count()
```

Out[104…

```
ase_antig_an
0       9634
1      25447
2     260999
3      31726
4      26022
5      23089
6      90970
7       1345
8         26
9        106
21         4
32         2
Name: ase_antig_an, dtype: int64
```

In [105…

```
# Otra forma de ver la frecuencia de cada valor.
df.groupby('ase_antig_an')['ase_antig_an'].count().plot.bar(figsize=[12,4])
plt.show()
```



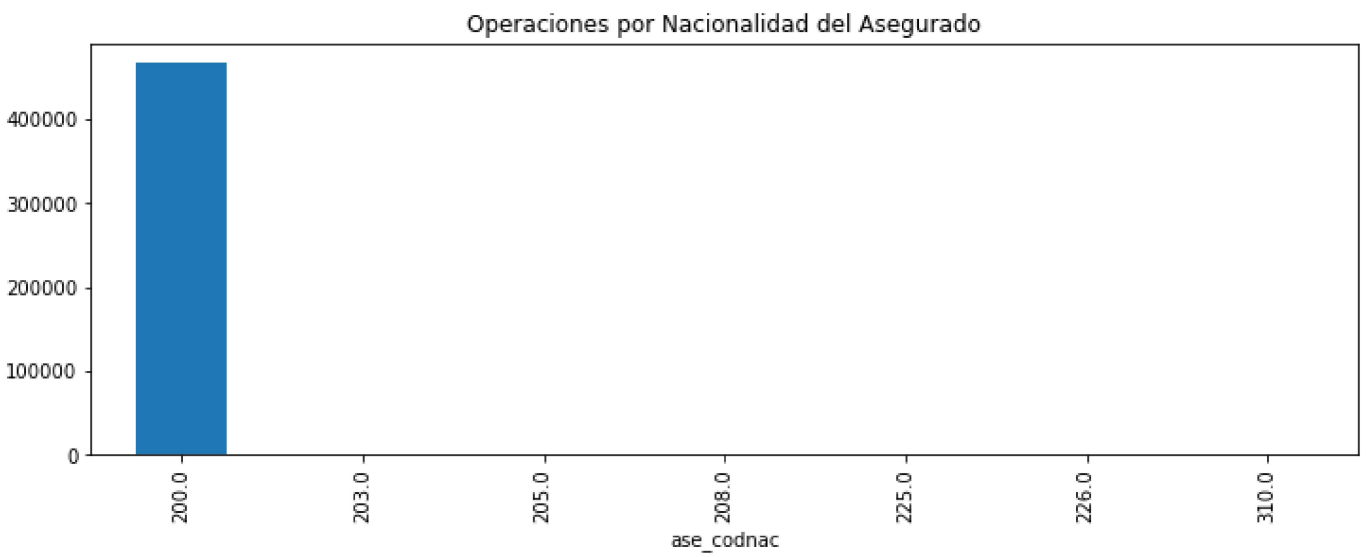
Variable 'ase_nac' y 'ase_codnac'

In [106…
```
df['ase_nac'].describe()
```

Out[106…
```
count         469370
unique             7
top        ARGENTINA
freq          467255
Name: ase_nac, dtype: object
```

In [107…
```
# Veo la frecuencia de cada valor.
df.groupby('ase_nac')['ase_nac'].count().plot.bar(figsize=[12,4])
plt.title('Operaciones por Nacionalidad del Asegurado')
plt.show()
```

In [108…
```
# Veo la frecuencia de cada valor.
df.groupby('ase_codnac')['ase_codnac'].count().plot.bar(figsize=[12,4])
plt.title('Operaciones por Nacionalidad del Asegurado')
plt.show()
```



In [109…
```
# Reviso la frecuencia de cada valor.
df.groupby('ase_nac')['ase_nac'].count()
```

Out[109…
```
ase_nac
ARGENTINA    467255
BRASIL          327
CHILE           263
CHINA             2
COLOMBIA       1406
URUGUAY         114
VENEZUELA         3
Name: ase_nac, dtype: int64
```

In [110…
```
# Reviso la frecuencia de cada valor.
df.groupby('ase_codnac')['ase_codnac'].count()
```

Out[110…
```
ase_codnac
200.0    467255
203.0       327
205.0      1406
208.0       263
225.0       114
226.0         3
310.0         2
Name: ase_codnac, dtype: int64
```

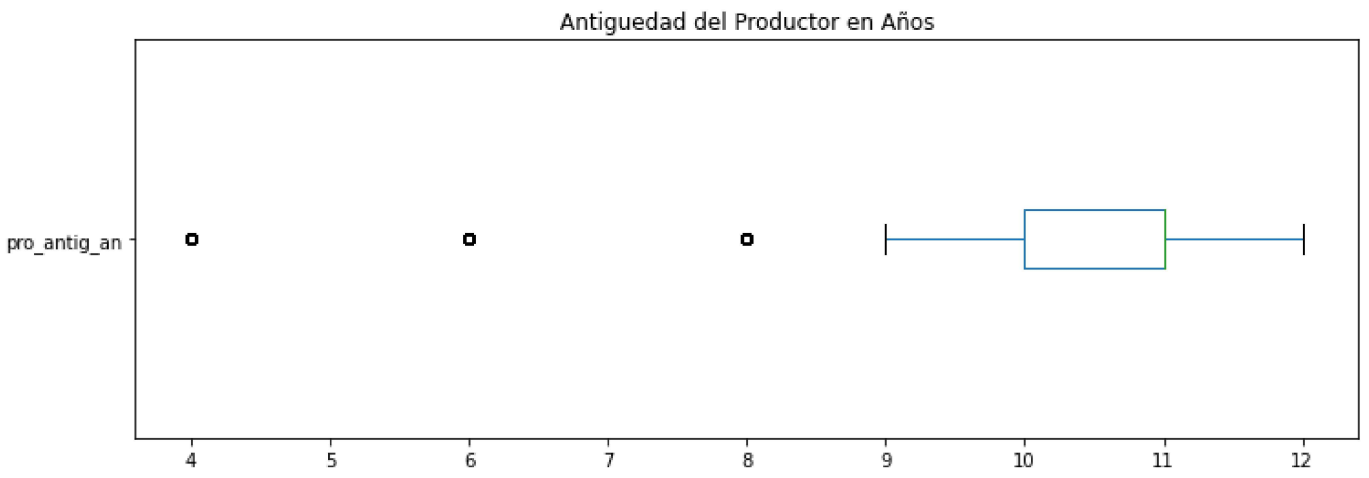
Variable 'pro_antig_an'

In [111…
```
df['pro_antig_an'].describe()
```

Out[111…
```
count    469370.000000
mean          9.708938
std           2.256888
min           4.000000
25%          10.000000
50%          11.000000
75%          11.000000
max          12.000000
Name: pro_antig_an, dtype: float64
```

In [112…
```
df['pro_antig_an'].plot(title='Antiguedad del Productor en Años', kind='box', vert=False, figsi
```

Out[112…
```
<AxesSubplot:title={'center':'Antiguedad del Productor en Años'}>
```



In [113…
```
df.groupby('pro_antig_an')['pro_antig_an'].count()
```

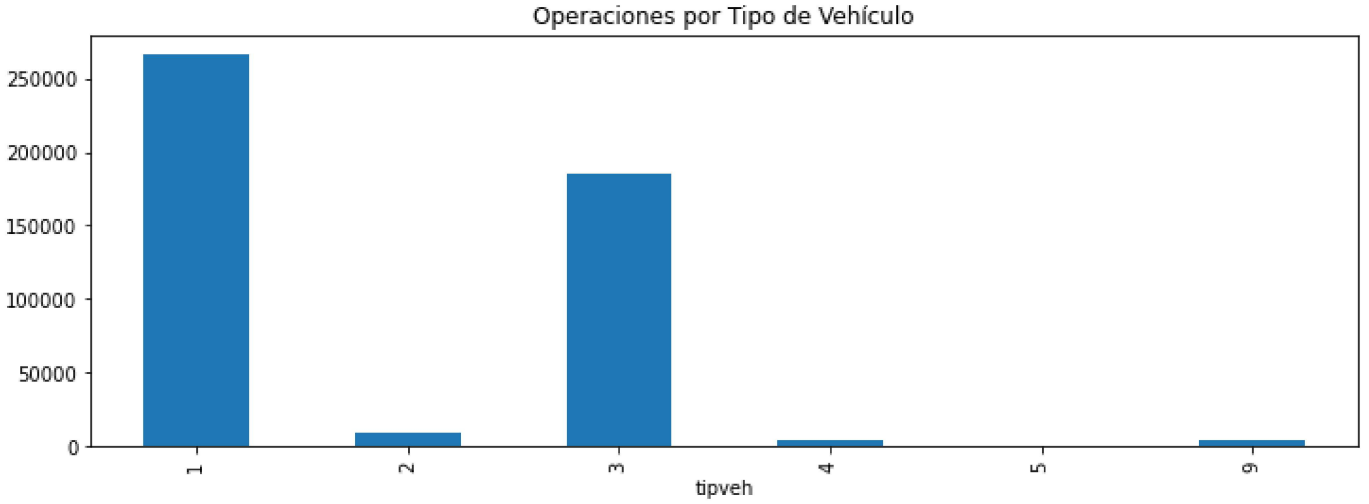
Out[113…
```
pro_antig_an
4      12936
6     100050
8       1531
9       1916
10     11660
11    336376
12      4901
Name: pro_antig_an, dtype: int64
```

Variable 'tipveh'

In [114…
```
df['tipveh'].describe()
```

Out[114…
```
count    469370.000000
mean          1.907193
std           1.178078
min           1.000000
25%           1.000000
50%           1.000000
75%           3.000000
max           9.000000
Name: tipveh, dtype: float64
```

In [115…
```
# Veo la frecuencia de cada valor.
df.groupby('tipveh')['tipveh'].count().plot.bar(figsize=[12,4])
plt.title('Operaciones por Tipo de Vehículo')
plt.show()
```



In [116…
```
df.groupby('tipveh')['tipveh'].count()
```

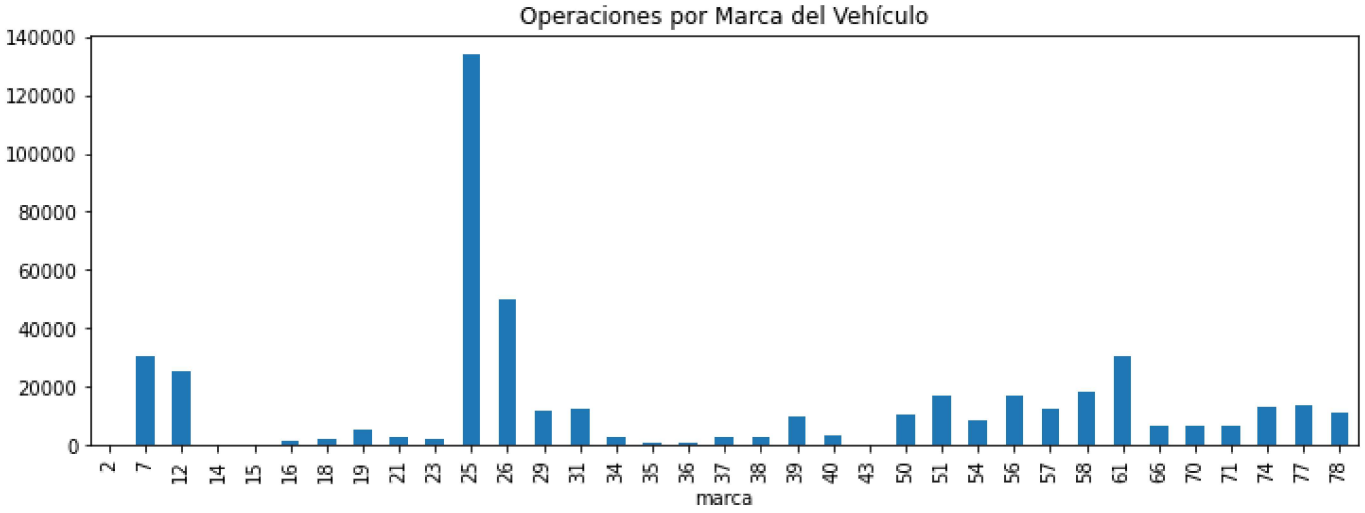
Out[116…
```
tipveh
1    265652
2      9151
3    185840
4      4946
5        27
9      3754
Name: tipveh, dtype: int64
```

Variable 'marca'

In [117…
```
df['marca'].describe()
```

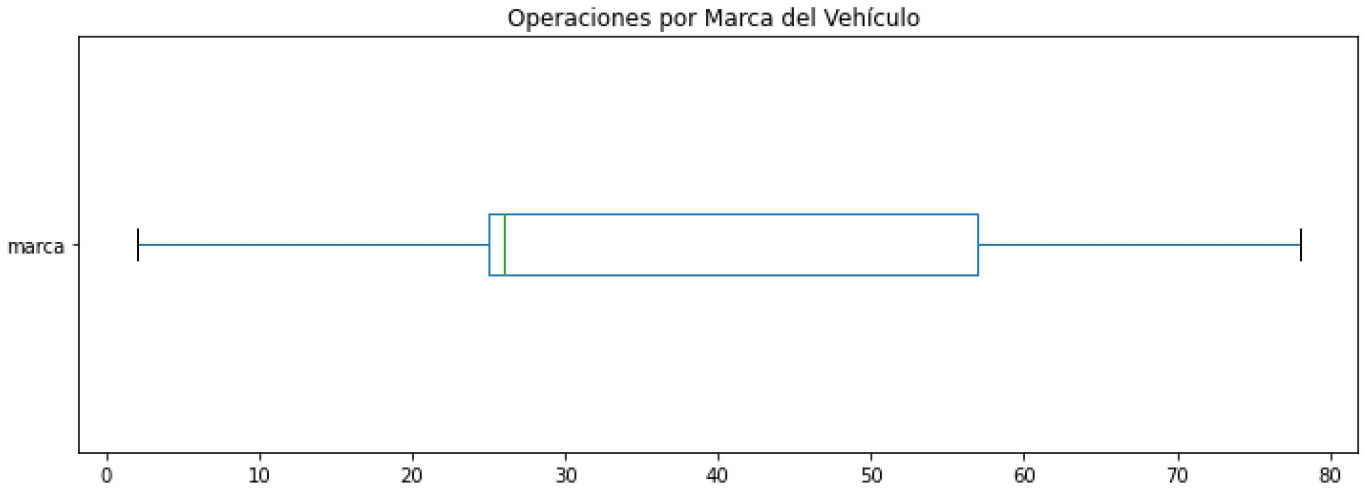
Out[117…
```
count    469370.000000
mean         37.424358
std          20.413349
min           2.000000
25%          25.000000
50%          26.000000
75%          57.000000
max          78.000000
Name: marca, dtype: float64
```

In [118…
```
# Veo la frecuencia de cada valor.
df.groupby('marca')['marca'].count().plot.bar(figsize=[12,4])
plt.title('Operaciones por Marca del Vehículo')
plt.show()
```



In [119…
```
df['marca'].plot(title='Operaciones por Marca del Vehículo ', kind='box', vert=False, figsize=[
```

Out[119…
```
<AxesSubplot:title={'center':'Operaciones por Marca del Vehículo '}>
```



In [120…
```
df.groupby('marca')['marca'].count()
```

Out[120…
```
marca
2        166
7      30312
12     25597
14        28
15        46
16      1110
18      2113
19      4961
21      2525
23      2036
25    134085
26     49958
29     11466
31     12055
34      2447
35       851
36       889
37      2450
38      2913
39      9938
40      3466
43         2
50     10081
51     16747
54      8289
56     16932
57     12614
58     18065
61     30701
66      6600
70      6193
71      6216
74     12899
77     13416
78     11203
Name: marca, dtype: int64
```

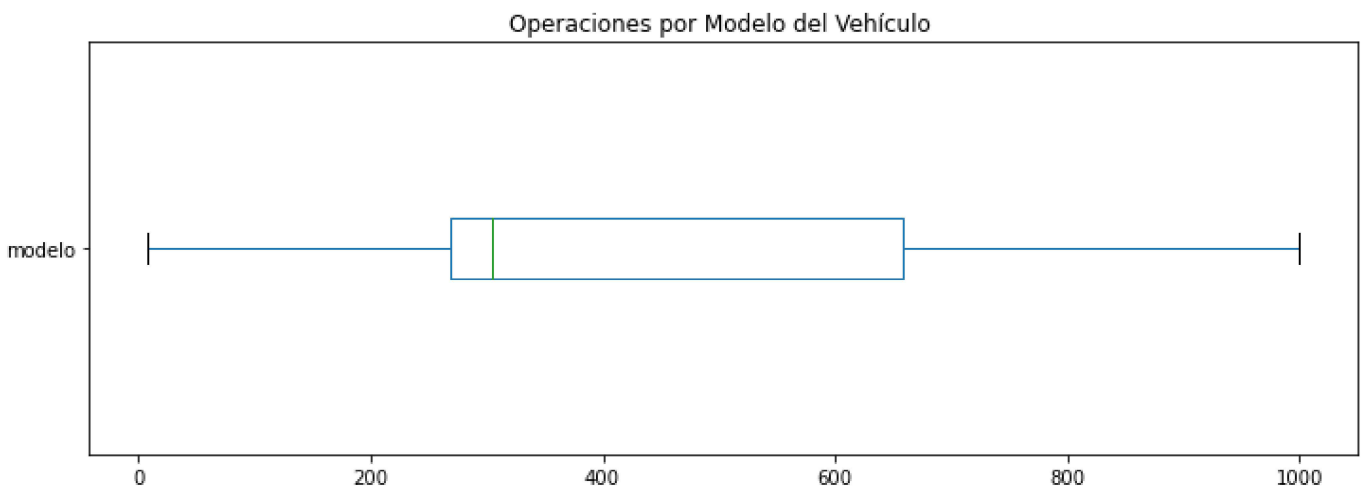
Variable 'modelo'

In [121…
```
df['modelo'].describe()
```

Out[121…
```
count    469370.000000
mean        434.674807
std         261.009605
min           7.000000
25%         268.000000
50%         304.000000
75%         658.000000
max        1000.000000
Name: modelo, dtype: float64
```

In [122…
```
df['modelo'].plot(title='Operaciones por Modelo del Vehículo ', kind='box', vert=False, figsize
```

Out[122…
```
<AxesSubplot:title={'center':'Operaciones por Modelo del Vehículo '}>
```



In [123…
```
df.groupby('modelo')['modelo'].count()
```

Out[123…
```
modelo
7         166
74      30312
102      1659
103      1696
104      1781
        ...
996       415
997       452
998       413
999       420
1000      435
Name: modelo, Length: 587, dtype: int64
```

Variable 'anio'

In [124…
```
df['anio'].describe()
```

Out[124…
```
count    469370.000000
mean       2000.566114
std           7.087862
min        1990.000000
25%        1995.000000
50%        2000.000000
75%        2004.000000
max        2018.000000
Name: anio, dtype: float64
```

In [125…
```
# Veo la frecuencia de cada valor.
df.groupby('anio')['anio'].count().plot.bar(figsize=[12,4])
plt.title('Operaciones por Año de Fabricación del Vehículo')
plt.show()
```

In [126…
```
df.groupby('anio')['anio'].count()
```

Out[126…
```
anio
1990    21787
1991    21799
1992    21876
1993    21663
1994    21963
1995    21595
1996    21548
1997    21712
1998    21740
1999    21727
2000    27311
2001    27581
2002    27148
2003    27261
2004    27682
2005    27515
2006     6567
2007     6423
2008     6856
2009     6763
2010     6766
2011     6827
2012     6712
2013     6903
2014     6888
2015     6977
2016     6468
2017     6557
2018     6755
Name: anio, dtype: int64
```

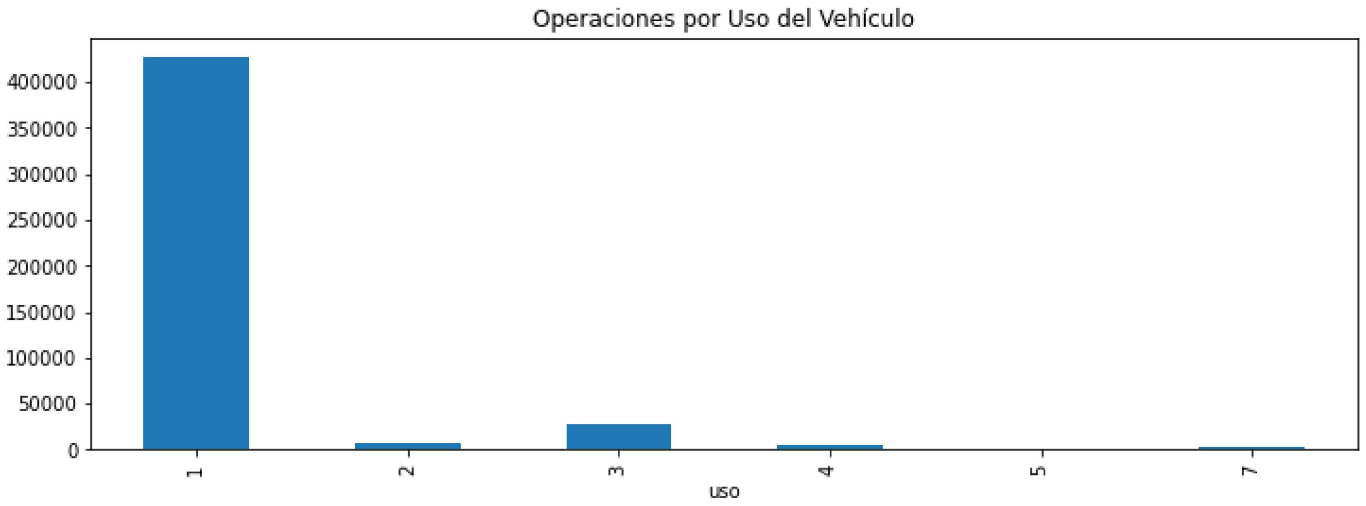
In [127…
```
### Variable 'uso'
```

In [128…
```
df['uso'].describe()
```

Out[128…
```
count    469370.000000
mean          1.211358
std           0.766356
min           1.000000
25%           1.000000
50%           1.000000
75%           1.000000
max           7.000000
Name: uso, dtype: float64
```

In [129…
```
# Veo la frecuencia de cada valor.
```

```
df.groupby('uso')['uso'].count().plot.bar(figsize=[12,4])
plt.title('Operaciones por Uso del Vehículo')
plt.show()
```



In [130…
```
df.groupby('uso')['uso'].count()
```

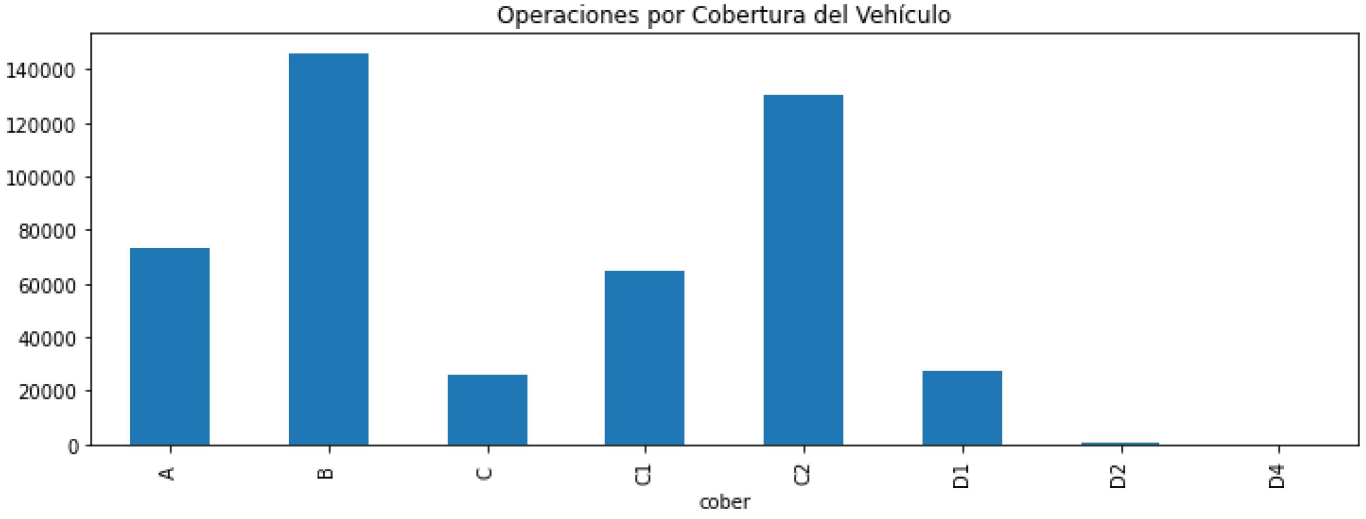
Out[130…
```
uso
1    426447
2      6509
3     27846
4      4776
5        38
7      3754
Name: uso, dtype: int64
```

Variable 'cober'

In [131…
```
df['cober'].describe()
```

Out[131…
```
count     469370
unique         8
top            B
freq      146285
Name: cober, dtype: object
```

In [132…
```
# Veo la frecuencia de cada valor.
df.groupby('cober')['cober'].count().plot.bar(figsize=[12,4])
plt.title('Operaciones por Cobertura del Vehículo')
plt.show()
```



In [133…
```
df.groupby('cober')['cober'].count()
```

Out[133…
```
cober
A      73483
B     146285
C      26017
C1     65076
C2    130747
D1     27594
D2       128
D4        40
Name: cober, dtype: int64
```

Variable 'cob_fecuma'

In [134…
```
df['cob_fecuma'].describe()
```

Out[134…
```
count       451613
unique         615
top       1/5/2018
freq          1623
Name: cob_fecuma, dtype: object
```

In [135…
```
df.groupby('cob_fecuma')['cob_fecuma'].count()
```

Out[135…
```
cob_fecuma
1/1/2018      291
1/1/2019      938
1/10/2018    1405
1/10/2019       1
1/11/2018    1184
             ...
9/5/2019       34
9/6/2018     1295
9/7/2018     1177
9/8/2018     1245
9/9/2018     1230
Name: cob_fecuma, Length: 615, dtype: int64
```

In [136…
```
df['cob_fecuma'].str.slice(6, 10).value_counts(dropna=False) # Registros por Año.
```

Out[136…
```
018     241947
18       86056
2018     77910
019      29169
NaN      17757
19       16411
2019       120
Name: cob_fecuma, dtype: int64
```

In [137…
```
df.dtypes
```

Out[137…
```
pol              int64
endoso           int64
item             int64
tipend          object
idaseg           int64
ase_antig_an     int64
ase_cp           int64
ase_codnac     float64
ase_nac         object
idprod           int64
pro_antig_an     int64
tipveh           int64
marca            int64
modelo           int64
anio             int64
uso              int64
uso_desc        object
```

```
cober            object
codcober          int64
cober_desc       object
cob_fecuma       object
cob_ef           object
fraude            int64
dtype: object
```

In [138…
```
# Ordena la variable como 'texto', mezcla el 2018 con el 2019. Para facilitar el operar de ser
# la convierto a tipo 'date'.
# df['cob_fecuma'] = df['cob_fecuma'].astype('datetime64[ns]')
df['cob_fecuma']=pd.to_datetime(df.cob_fecuma)
```

In [139…
```
df['cob_fecuma'].dtypes
```

Out[139… `dtype('<M8[ns]')`

In [140…
```
df.cob_fecuma.dt.year.value_counts(dropna=False) # Registros por Año.
```

Out[140…
```
2018.0    405913
2019.0     45700
NaN        17757
Name: cob_fecuma, dtype: int64
```

In [141…
```
df['cob_fecuma'].describe(datetime_is_numeric=True)
```

Out[141…
```
count                           451613
mean     2018-08-06 07:43:46.082951680
min                2018-01-01 00:00:00
25%                2018-04-25 00:00:00
50%                2018-07-30 00:00:00
75%                2018-11-04 00:00:00
max                2019-12-21 00:00:00
Name: cob_fecuma, dtype: object
```

In [142…
```
df.groupby('cob_fecuma')['cob_fecuma'].count()
```

Out[142…
```
cob_fecuma
2018-01-01     291
2018-01-02     366
2018-01-03    1365
2018-01-04    1246
2018-01-05    1623
              ...
2019-12-15       1
2019-12-16       2
2019-12-17       2
2019-12-18       3
2019-12-21       2
Name: cob_fecuma, Length: 615, dtype: int64
```

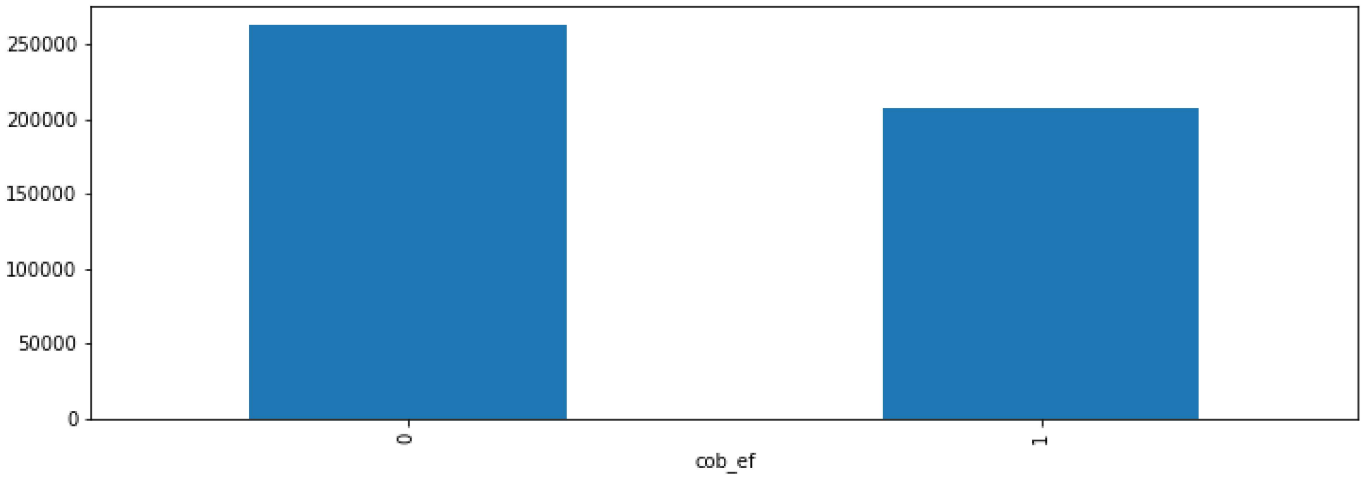
Variable 'cob_ef'

In [143…
```
# Encoding variable 'cob_ef'
df['cob_ef'] = df['cob_ef'].map( {'N':0,'S':1} )
```

In [144…
```
df.groupby('cob_ef')['cob_ef'].count()
```

Out[144…
```
cob_ef
0    262146
1    207224
Name: cob_ef, dtype: int64
```

In [145…
```
# Veo la frecuencia de cada valor.
df.groupby('cob_ef')['cob_ef'].count().plot.bar(figsize=[12,4])
plt.show()
```

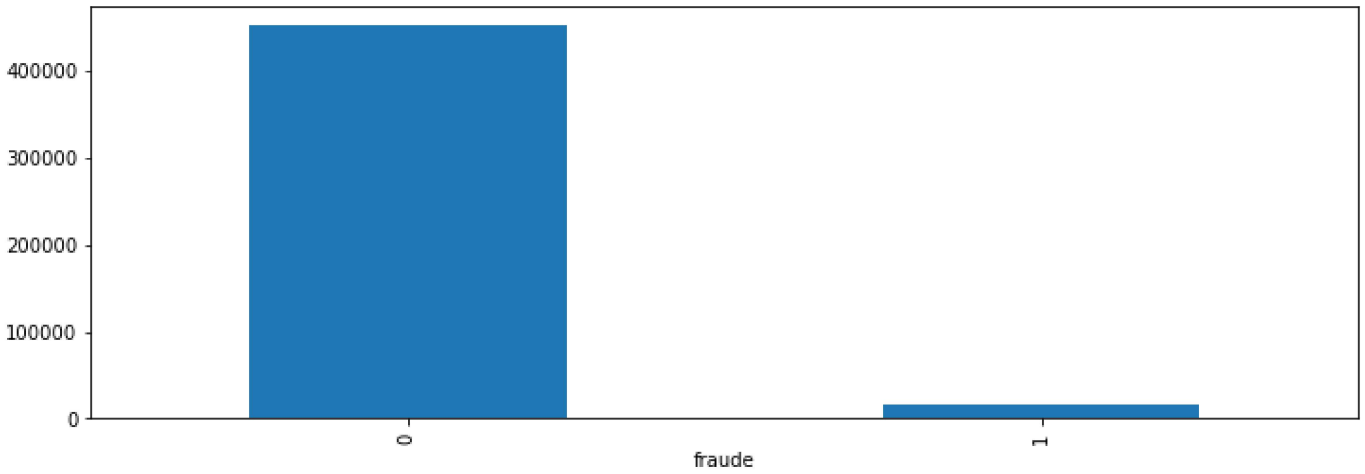


Variable 'fraude'

In [146…
```
df.groupby('fraude')['fraude'].count()
```

Out[146…
```
fraude
0    452134
1     17236
Name: fraude, dtype: int64
```

In [147…
```
# Veo la frecuencia de cada valor.
df.groupby('fraude')['fraude'].count().plot.bar(figsize=[12,4])
plt.show()
```



Análisis exploratorio multivariable.

In [148…
```
# Me permitirá ver que variables están más relacionadas con 'fraude'.
# Pandas df.corr() is used to find the pairwise correlation of all columns in the dataframe. Ar
# excluded. For any non-numeric data type columns in the dataframe it is ignored.
matriz_corr=df.corr(method='pearson').round(2)
# .style.background_gradient()
matriz_corr
```

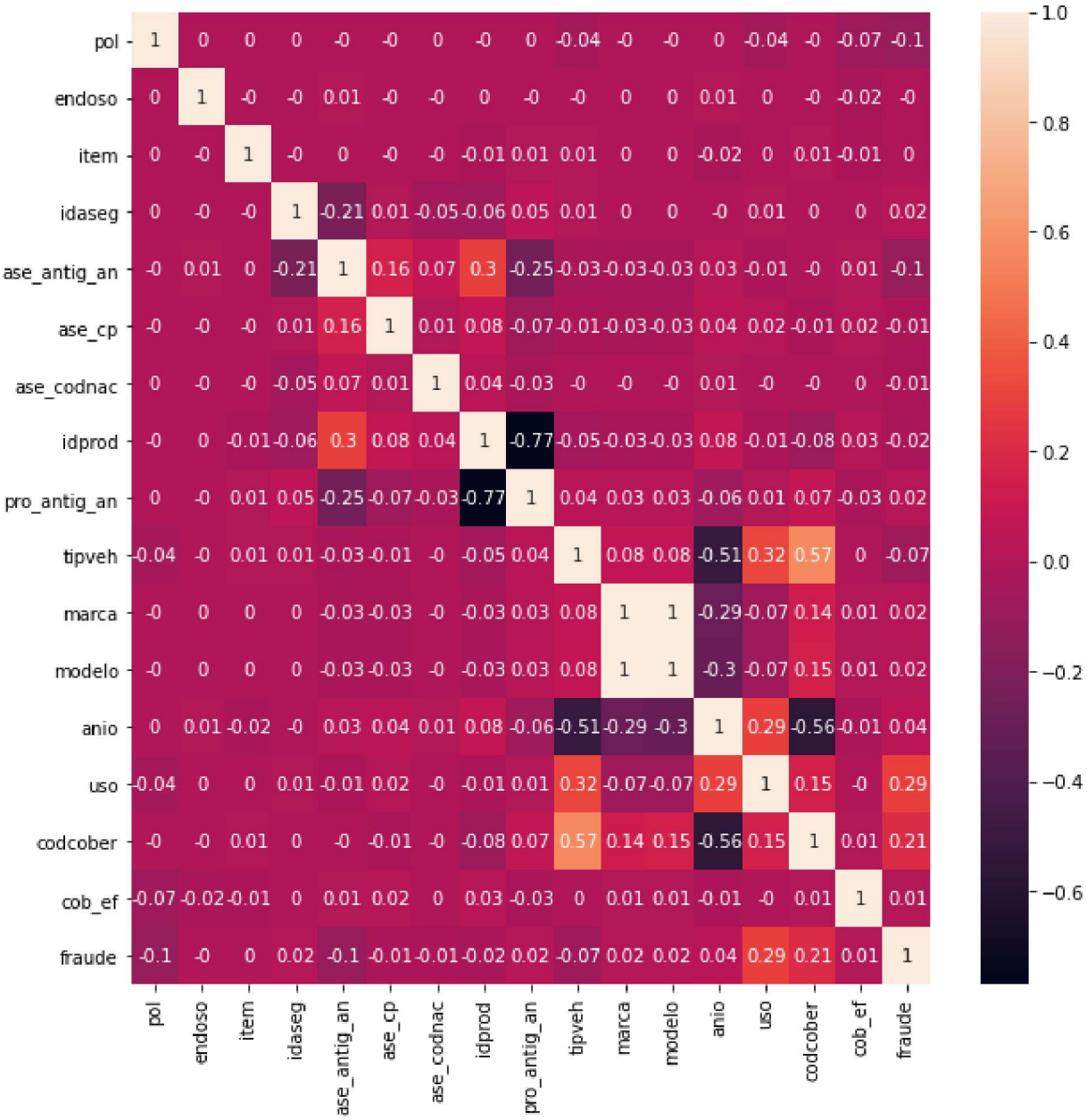
Out[148…

	pol	endoso	item	idaseg	ase_antig_an	ase_cp	ase_codnac	idprod	pro_antig_an	tipveh	mar
pol	1.00	0.00	0.00	0.00	-0.00	-0.00	0.00	-0.00	0.00	-0.04	-0.
endoso	0.00	1.00	-0.00	-0.00	0.01	-0.00	-0.00	0.00	-0.00	-0.00	0.

	pol	endoso	item	idaseg	ase_antig_an	ase_cp	ase_codnac	idprod	pro_antig_an	tipveh	mar
item	0.00	-0.00	1.00	-0.00	0.00	-0.00	-0.00	-0.01	0.01	0.01	0.
idaseg	0.00	-0.00	-0.00	1.00	-0.21	0.01	-0.05	-0.06	0.05	0.01	0.
ase_antig_an	-0.00	0.01	0.00	-0.21	1.00	0.16	0.07	0.30	-0.25	-0.03	-0.
ase_cp	-0.00	-0.00	-0.00	0.01	0.16	1.00	0.01	0.08	-0.07	-0.01	-0.
ase_codnac	0.00	-0.00	-0.00	-0.05	0.07	0.01	1.00	0.04	-0.03	-0.00	-0.
idprod	-0.00	0.00	-0.01	-0.06	0.30	0.08	0.04	1.00	-0.77	-0.05	-0.
pro_antig_an	0.00	-0.00	0.01	0.05	-0.25	-0.07	-0.03	-0.77	1.00	0.04	0.
tipveh	-0.04	-0.00	0.01	0.01	-0.03	-0.01	-0.00	-0.05	0.04	1.00	0.
marca	-0.00	0.00	0.00	0.00	-0.03	-0.03	-0.00	-0.03	0.03	0.08	1.
modelo	-0.00	0.00	0.00	0.00	-0.03	-0.03	-0.00	-0.03	0.03	0.08	1.
anio	0.00	0.01	-0.02	-0.00	0.03	0.04	0.01	0.08	-0.06	-0.51	-0.
uso	-0.04	0.00	0.00	0.01	-0.01	0.02	-0.00	-0.01	0.01	0.32	-0.
codcober	-0.00	-0.00	0.01	0.00	-0.00	-0.01	-0.00	-0.08	0.07	0.57	0.
cob_ef	-0.07	-0.02	-0.01	0.00	0.01	0.02	0.00	0.03	-0.03	0.00	0.
fraude	-0.10	-0.00	0.00	0.02	-0.10	-0.01	-0.01	-0.02	0.02	-0.07	0.

In [149…

```
# fig, ax = plt.subplots(figsize=[10, 10])
sns.heatmap(data=matriz_corr, annot=True)
plt.rcParams['figure.figsize'] = [10,10]
plt.show()
```



In [150…

```
df.to_csv('data_df.csv', header=True, index=False)
```

In [ ]: