ИМПОРТЫ И ЗАГРУЗКА ДАННЫХ

In [534…

```
import pandas as pd
import numpy as np
from scipy.stats import norm

import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objs as go
from plotly.subplots import make_subplots

from statsmodels.tsa.statespace.sarimax import SARIMAX

from datetime import datetime
%matplotlib inline
plt.rcParams["figure.figsize"] = (12,8)

import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

from sklearn.preprocessing import OneHotEncoder
```

In [535…

```
# Загрузить данные из Google Sheets в Pandas DataFrame
url = 'https://docs.google.com/spreadsheets/d/12o1iofQx6V-UhInjUjLjpKxx3Z
df = pd.read_csv(url)
```

In [536…

```
df.head()
```

Out[536]:

	MONTH	SERVICE_CATEGORY	CLAIM_SPECIALTY	PAYER	PAID_AMOUNT
0	201801	AncillaryFFS	NaN	Payer F	4281
1	201801	AncillaryFFS	NaN	Payer H	2221
2	201801	AncillaryFFS	NaN	Payer O	3937
3	201801	AncillaryFFS	NaN	Payer W	268
4	201801	AncillaryFFS	ACH	Payer W	151

In [537…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52152 entries, 0 to 52151
Data columns (total 5 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   MONTH             52152 non-null  int64 
 1   SERVICE_CATEGORY  52152 non-null  object
 2   CLAIM_SPECIALTY   51901 non-null  object
 3   PAYER             52152 non-null  object
 4   PAID_AMOUNT       52152 non-null  int64 
dtypes: int64(2), object(3)
memory usage: 2.0+ MB
```

In [538…

```
df.dtypes
```

Out[538]:

```
MONTH                int64
SERVICE_CATEGORY    object
CLAIM_SPECIALTY     object
PAYER               object
PAID_AMOUNT          int64
dtype: object
```

In [539…

```
# Проверка на наличие пропущенных значений
df.isnull().sum()
```

Out[539]:

```
MONTH                 0
SERVICE_CATEGORY      0
CLAIM_SPECIALTY     251
PAYER                 0
PAID_AMOUNT           0
dtype: int64
```

In [540…

```
# Проверка на наличие дубликатов
df.duplicated().sum()
```

Out[540]:

```
0
```

EDA

MONTH

In [541…

```
df['MONTH'].value_counts().sort_index()
```

```
Out[541]: 201801    1712
          201802    1664
          201803    1723
          201804    1714
          201805    1759
          201806    1745
          201807    1706
          201808    1775
          201809    1767
          201810    1748
          201811    1747
          201812    1718
          201900      11
          201901    1790
          201902    1751
          201903    1769
          201904    1767
          201905    1823
          201906    1746
          201907    1877
          201908    1828
          201909    1774
          201910    1863
          201911    1723
          201912    1772
          202001    1782
          202002    1711
          202003    1769
          202004    1539
          202005    1668
          202006    1344
          202007      67
          Name: MONTH, dtype: int64
```

удалим июль 2020 года (мало данных) и несуществующую дату 201900

```
In [542…  df = df[df['MONTH'].isin([201900, 202007]) == False]
```

```
In [543…  df.shape
```

```
Out[543]: (52074, 5)
```

преобразуем в дату

```
In [544…  df['MONTH'] = df['MONTH'].astype(str)
          df['MONTH'] = pd.to_datetime(df['MONTH'], format='%Y%m')
```

```
In [546…  df['MONTH'].value_counts().sort_index()
```

Out[546]:

```
2018-01-01    1712
2018-02-01    1664
2018-03-01    1723
2018-04-01    1714
2018-05-01    1759
2018-06-01    1745
2018-07-01    1706
2018-08-01    1775
2018-09-01    1767
2018-10-01    1748
2018-11-01    1747
2018-12-01    1718
2019-01-01    1790
2019-02-01    1751
2019-03-01    1769
2019-04-01    1767
2019-05-01    1823
2019-06-01    1746
2019-07-01    1877
2019-08-01    1828
2019-09-01    1774
2019-10-01    1863
2019-11-01    1723
2019-12-01    1772
2020-01-01    1782
2020-02-01    1711
2020-03-01    1769
2020-04-01    1539
2020-05-01    1668
2020-06-01    1344
Name: MONTH, dtype: int64
```

сделаем фичу по полугодию для дальнейшей группировки

In [547…

```
df['HALF_YEAR'] = df['MONTH'].apply(lambda x: datetime(x.year, 6,1) if x.
```

In [548…

```
df['HALF_YEAR'].value_counts().sort_index()
```

Out[548]:

```
2018-06-01    10317
2018-12-01    10461
2019-06-01    10646
2019-12-01    10837
2020-06-01     9813
Name: HALF_YEAR, dtype: int64
```

SERVICE_CATEGORY

In [549…

```
df['SERVICE_CATEGORY'].value_counts().sort_index()
```

Out[549]:

```
ASCServices            2639
AncillaryFFS           9682
ERServices             3690
InpatientServices      9413
OutpatientServices     7572
PCPEncounter           1221
PCPFFS                 1401
SNFServices            2497
SpecialistFFS          1684
SpecialistsFFS        12275
Name: SERVICE_CATEGORY, dtype: int64
```

SpecialistsFFS приравняем с SpecialistFFS, т.к. по сути - одно и то же

In [550…

```
df['SERVICE_CATEGORY'] = df['SERVICE_CATEGORY'].apply(lambda x: 'Speciali
```

In [551…

```
df['SERVICE_CATEGORY'].value_counts().sort_index()
```

Out[551]:

```
ASCServices            2639
AncillaryFFS           9682
ERServices             3690
InpatientServices      9413
OutpatientServices     7572
PCPEncounter           1221
PCPFFS                 1401
SNFServices            2497
SpecialistFFS         13959
Name: SERVICE_CATEGORY, dtype: int64
```

CLAIM_SPECIALTY

In [552…

```
df['CLAIM_SPECIALTY'].isnull().sum()
```

Out[552]:

```
251
```

In [553…

```
df.head(5)
```

Out[553]:

	MONTH	SERVICE_CATEGORY	CLAIM_SPECIALTY	PAYER	PAID_AMOUNT	HALF_YEA
0	2018-01-01	AncillaryFFS	NaN	Payer F	4281	2018-06-0
1	2018-01-01	AncillaryFFS	NaN	Payer H	2221	2018-06-0
2	2018-01-01	AncillaryFFS	NaN	Payer O	3937	2018-06-0
3	2018-01-01	AncillaryFFS	NaN	Payer W	268	2018-06-0
4	2018-01-01	AncillaryFFS	ACH	Payer W	151	2018-06-0

In [554…

```
df.CLAIM_SPECIALTY.unique().shape
```

Out[554]:

```
(906,)
```

предобработаем специальности для более точной аналитики (приведем в нижний регистр, сгруппируем похожие, обьединим малочисленные специальности в other) Возможно маппинг будет не до конца некорректен, т.к. необходимо небольшое погружение в предметную область + необходимо взаимодействие с заказчиком для уточенния правильного категорирования специализаций

In [555…

```
df['CLAIM_SPECIALTY'] = df['CLAIM_SPECIALTY'].apply(lambda x: str(x).lowe
```

In [556…

```
def speciality(x):
    if 'assistant' in x or 'nursing' in x or 'nurs' in x:
        return 'assistant_nurse'
    elif 'family practice' in x:
        return 'family practice'
    elif 'radiology' in x or 'nuclear' in x or 'x-ray' in x:
        return 'radiology'
    elif 'emergency' in x or 'ambulance' in x or 'critical care' in x or
        return 'emergency'
    elif 'general practice' in x:
        return 'general practice'
    elif 'pathology' in x:
        return 'pathology'
    elif 'gastroenterology' in x:
        return 'gastroenterology'
    elif 'surg' in x and 'plastic' not in x:
        return 'surgery'
    elif 'surgery' in x and 'plastic' in x:
        return 'plastic surgery'
    elif 'cardiology' in x or 'cardiac' in x:
        return 'cardiology'
    elif 'anesth' in x and 'nurs' not in x:
        return 'anesthesiology'
    elif 'neurology' in x:
        return 'neurology'
    elif 'urology' in x:
        return 'urology'
    elif 'nephrology' in x:
        return 'nephrology'
    elif 'podiatry' in x:
        return 'podiatry'
    elif 'pain management' in x:
        return 'pain management'
    elif 'hospital' in x:
        return 'hospital'
    elif 'hematology' in x:
        return 'hematology'
    elif 'oncology' in x:
        return 'oncology'
    elif 'endocrinology' in x:
        return 'endocrinology'
    elif 'pulmonary' in x:
        return 'pulmonary'
    elif 'lab' in x:
        return 'laboratory'
    elif 'derma' in x:
        return 'dermatology'
    elif 'infecti' in x:
        return 'infectious'
    elif 'rehab' in x:
        return 'rehab'
    elif 'gynecology' in x:
        return 'gynecology'
    elif 'psychiatry' in x:
        return 'psychiatry'
    elif 'geriatrics' in x:
        return 'geriatrics'
    elif 'osteo' in x:
        return 'osteopath'
    elif 'opt' in x or 'ophth' in x or 'visio' in x:
        return 'opthalmology'
    elif 'physical therapy' in x:
        return 'physical therapy'
    elif 'rheumatology' in x:
        return 'rheumatology'
    elif 'pediatrics' in x:
        return 'pediatrics'
    elif 'pediatrics' in x:
        return 'pediatrics'
    elif 'internal medicine' in x:
        return 'internal medicine'
    elif 'podiatrist' in x or 'orthoped' in x:
        return 'podiatrist'
    elif 'psychology' in x:
        return 'psychology'
    else:
        return 'other'
```

In [557…

```
df['CLAIM_SPECIALTY_GRP'] = df['CLAIM_SPECIALTY'].apply(speciality)
```

In [558…

```
df.CLAIM_SPECIALTY.unique().shape
```

Out[558]:

```
(795,)
```

In [559…

```
df.groupby(['CLAIM_SPECIALTY_GRP', 'CLAIM_SPECIALTY'])['PAYER'].count().t
```

PAYER

In [560…

```
df['PAYER'].value_counts().sort_index()
```

Out[560]:

```
Payer B      3631
Payer CA     8681
Payer CO     3677
Payer F      9012
Payer H      2291
Payer O      6277
Payer S      4189
Payer UL      685
Payer UN     6526
Payer W      7105
Name: PAYER, dtype: int64
```

In [561…

```
df.head()
```

Out[561]:

	MONTH	SERVICE_CATEGORY	CLAIM_SPECIALTY	PAYER	PAID_AMOUNT	HALF_YEA
0	2018-01-01	AncillaryFFS	nan	Payer F	4281	2018-06-0
1	2018-01-01	AncillaryFFS	nan	Payer H	2221	2018-06-0
2	2018-01-01	AncillaryFFS	nan	Payer O	3937	2018-06-0
3	2018-01-01	AncillaryFFS	nan	Payer W	268	2018-06-0
4	2018-01-01	AncillaryFFS	ach	Payer W	151	2018-06-0

PAID_AMOUNT

In [562…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 52074 entries, 0 to 52084
Data columns (total 7 columns):
 #   Column                Non-Null Count  Dtype         
---  ------                --------------  -----         
 0   MONTH                 52074 non-null  datetime64[ns]
 1   SERVICE_CATEGORY      52074 non-null  object        
 2   CLAIM_SPECIALTY       52074 non-null  object        
 3   PAYER                 52074 non-null  object        
 4   PAID_AMOUNT           52074 non-null  int64         
 5   HALF_YEAR             52074 non-null  datetime64[ns]
 6   CLAIM_SPECIALTY_GRP   52074 non-null  object        
dtypes: datetime64[ns](2), int64(1), object(4)
memory usage: 3.2+ MB
```

In [563…

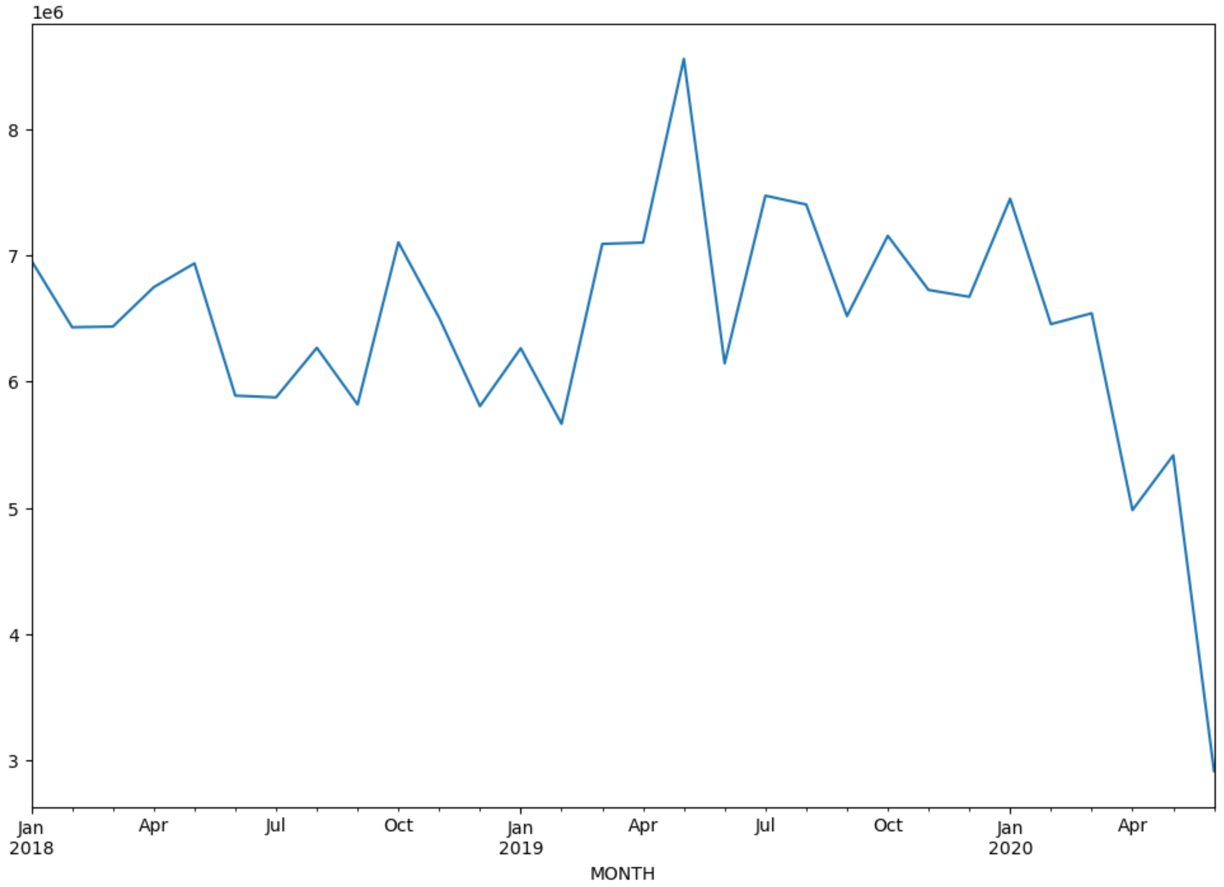
```
# удалим их, чтобы не искажали
```

In [564…

```
df.groupby('MONTH')['PAID_AMOUNT'].sum().plot()
```

Out[564]:

```
<Axes: xlabel='MONTH'>
```



Вывод: За всю историю наблюдений по выплатам от страховых компаний видим пик выплат в июне 2019 года, и сильный спад по выплатам начиная с апреля 2020.

Аналитика по выплатам по компаниям + нулевых выплат и отрицательных (возвраты)

In [565...

```
# создадим сводный датасет в котором сделаем аналитику по выпдатам
total_paid = df[df['PAID_AMOUNT'] > 0 ].groupby('HALF_YEAR')['PAID_AMOUNT
total_paid.columns = ['HALF_YEAR','TOTAL_PAID']
total_null_paid = df[df['PAID_AMOUNT'] == 0 ].groupby('HALF_YEAR')['PAID_
total_null_paid.columns = ['HALF_YEAR','TOTAL_NULL_COUNT']
total_refund_paid = df[df['PAID_AMOUNT'] < 0 ].groupby('HALF_YEAR')['PAID
total_refund_paid.columns = ['HALF_YEAR','TOTAL_REFUND_PAID']
```

In [566...

```
df_grouped = df[df['PAID_AMOUNT'] > 0 ].groupby(['PAYER', 'HALF_YEAR'])['
```

In [567...

```
df_grouped_null  = df[df['PAID_AMOUNT']  == 0 ].groupby(['PAYER', 'HALF_Y
df_grouped_null.columns = ['PAYER','HALF_YEAR','COUNT_NULL']
```

In [568...

```
df_grouped_refund  = df[df['PAID_AMOUNT']  < 0 ].groupby(['PAYER', 'HALF_
df_grouped_refund.columns = ['PAYER','HALF_YEAR','PAID_REFUND']
```

In [569…

```
df_paid = df_grouped.merge(df_grouped_null, on= ['PAYER', 'HALF_YEAR'],ho
df_paid = df_paid.merge(df_grouped_refund, on= ['PAYER', 'HALF_YEAR'],how

df_paid = df_paid.merge(total_paid, on = 'HALF_YEAR', how='outer')
df_paid = df_paid.merge(total_null_paid, on= 'HALF_YEAR', how='outer')
df_paid = df_paid.merge(total_refund_paid, on= 'HALF_YEAR', how='outer')
```

In [570…

```
df_paid['percent_amount'] = (df_paid['sum'] / df_paid['TOTAL_PAID'] ) * 1
df_paid['percent_null'] = (df_paid['COUNT_NULL'] / df_paid['TOTAL_NULL_CO
df_paid['percent_refund'] = (df_paid['PAID_REFUND'] / df_paid['TOTAL_REFU
df_paid['relat_refund_total'] = (df_paid['PAID_REFUND'] / df_paid['sum']
```

In [571…

```
df_paid = df_paid.fillna(0)
```

In [572…

```
df_paid.head(1)
```

Out[572]:

	PAYER	HALF_YEAR	sum	mean	COUNT_NULL	PAID_REFUND	TOTAL_PAID
0	Payer B	2018-06-01	366501	660.362162	93.0	-1562.0	39446322

In [573…

```
fig = px.line(df_paid, x='HALF_YEAR', y='sum', color='PAYER',
                 width=1000, height=600)
fig.show()
```

In [574…

```
fig = px.bar(df_paid, x='HALF_YEAR', y='sum', color='PAYER', barmode='rel
fig.update_layout(title='Выплаты по полугодиям')
fig.update_layout(height=800, width=1000)
fig.show()
```

In [575…

```
fig = px.bar(df_paid, x='HALF_YEAR', y='relat_refund_total', color='PAYER
fig.update_layout(title='Доля возвратов к общей сумме выплат')
fig.update_layout(height=800, width=1000)
fig.show()
```

Выводы: во втором полугодии 2018 и в первом 2019ого доля отношение возвратов к выплатам у компании UN составило 12 и 9 % соооответственно, данная компания существенно выделяется по этому показателю от остальных страховщиков.

In [576…

```
fig = px.bar(df_paid, x='HALF_YEAR', y='percent_amount', color='PAYER', b
fig.update_layout(title='Процентное соотношение выплат по полугодиям')
fig.update_layout(height=800, width=1000)
fig.show()
```

Выводы: Крупнейшим страховщиком является Payer B, но с течением времени доля среди остальных страховщиков упала с 46 до 42%. Стоит отметить Payer H, который нарастил долю с 9 % до 14% за весь период наблюдений. Так же за последние пол года наблюдений Payer UN вырос с 3 до 5,5%

In [577…

```
fig = px.bar(df_paid, x='HALF_YEAR', y='percent_null', color='PAYER', bar
fig.update_layout(title='Процентное соотношение "нулевых" выплат по полуг
fig.update_layout(height=800, width=1000)
fig.show()
```

Выводы: В среднем 33% занимает Payer UN по нулевым выплатам, это возможно качество данных, на 2ом месте - payer CA. Но ниже приведен график по отрицательным выплатам (предположительно возвраты), в них так же лидирует Payer UN, где возвраты составляют от 63 до 88 % от всех возвратов по полугодию). И на 2ом месте по возвратам, как и по нулевым выплатам - Payer CA.

In [578…

```
fig = px.bar(df_paid, x='HALF_YEAR', y='percent_refund', color='PAYER', b
fig.update_layout(title='Процентное соотношение отрицательных выплат по п
fig.update_layout(height=800, width=1000)
fig.show()
```

In [579…

```
df.head(1)
```

Out[579]:

	MONTH	SERVICE_CATEGORY	CLAIM_SPECIALTY	PAYER	PAID_AMOUNT	HALF_YEAI
0	2018-01-01	AncillaryFFS	nan	Payer F	4281	2018-06-0

Аналитика по возвратам в разрезе SERVICE_CATEGORY и CLAIM_SPECIALTY_GRP для ТОП-2 страховщиков по отрицательным выплатам

In [580…

```
df[(df['PAID_AMOUNT'] < 0)&(df['PAYER'].isin(['Payer CA', 'Payer UN']))].
```

Out[580]:

	PAYER	SERVICE_CATEGORY	PAID_AMOUNT
0	Payer CA	AncillaryFFS	2
1	Payer CA	PCPEncounter	81
2	Payer UN	AncillaryFFS	223

In [581…

```
df[(df['PAID_AMOUNT'] < 0)&(df['PAYER'].isin(['Payer CA', 'Payer UN']))].
```

Out[581]:

	PAYER	CLAIM_SPECIALTY_GRP	PAID_AMOUNT
19	Payer UN	opthalmology	33
3	Payer CA	internal medicine	29
1	Payer CA	general practice	28
17	Payer UN	internal medicine	21
0	Payer CA	family practice	19
15	Payer UN	hematology	19
28	Payer UN	surgery	18
20	Payer UN	other	18
26	Payer UN	radiology	16
29	Payer UN	urology	11
7	Payer UN	cardiology	10
8	Payer UN	dermatology	10
21	Payer UN	pain management	9
11	Payer UN	family practice	8
23	Payer UN	physical therapy	8
24	Payer UN	podiatry	6
27	Payer UN	rheumatology	5
6	Payer UN	assistant_nurse	5
5	Payer UN	anesthesiology	5
4	Payer CA	other	4
14	Payer UN	gynecology	4
2	Payer CA	hospital	3
16	Payer UN	infectious	3
22	Payer UN	pediatrics	3
18	Payer UN	neurology	3
13	Payer UN	general practice	2
12	Payer UN	gastroenterology	2
10	Payer UN	endocrinology	2
25	Payer UN	psychology	1
9	Payer UN	emergency	1

In [613…

```
df.head(1)
```

Out[613]:

	MONTH	SERVICE_CATEGORY	CLAIM_SPECIALTY	PAYER	PAID_AMOUNT	HALF_YEA
0	2018-01-01	AncillaryFFS	nan	Payer F	4281	2018-06-0

Anomaly detection

Далее для поиска выбросов и аномалий исключим нулевые выплаты и отрицательные, т.к. по ним анализ закончен.

Т.к. выбросы зависят таких факторов, как вид врачебной деятельности, тип услуги, то поиск аномалий предлагаю производить в зависимости них. Т.к. например выброс в терапии, может быть нормой в хирургии.

Выбросы будем производит по методу 3-х сигм, но т.к. у нас распределение не нормальное, пролагорифмируем его для корректного нахожденеия выбросов

In [582…

```
df = df[df['PAID_AMOUNT'] > 0].reset_index(drop=True)
```

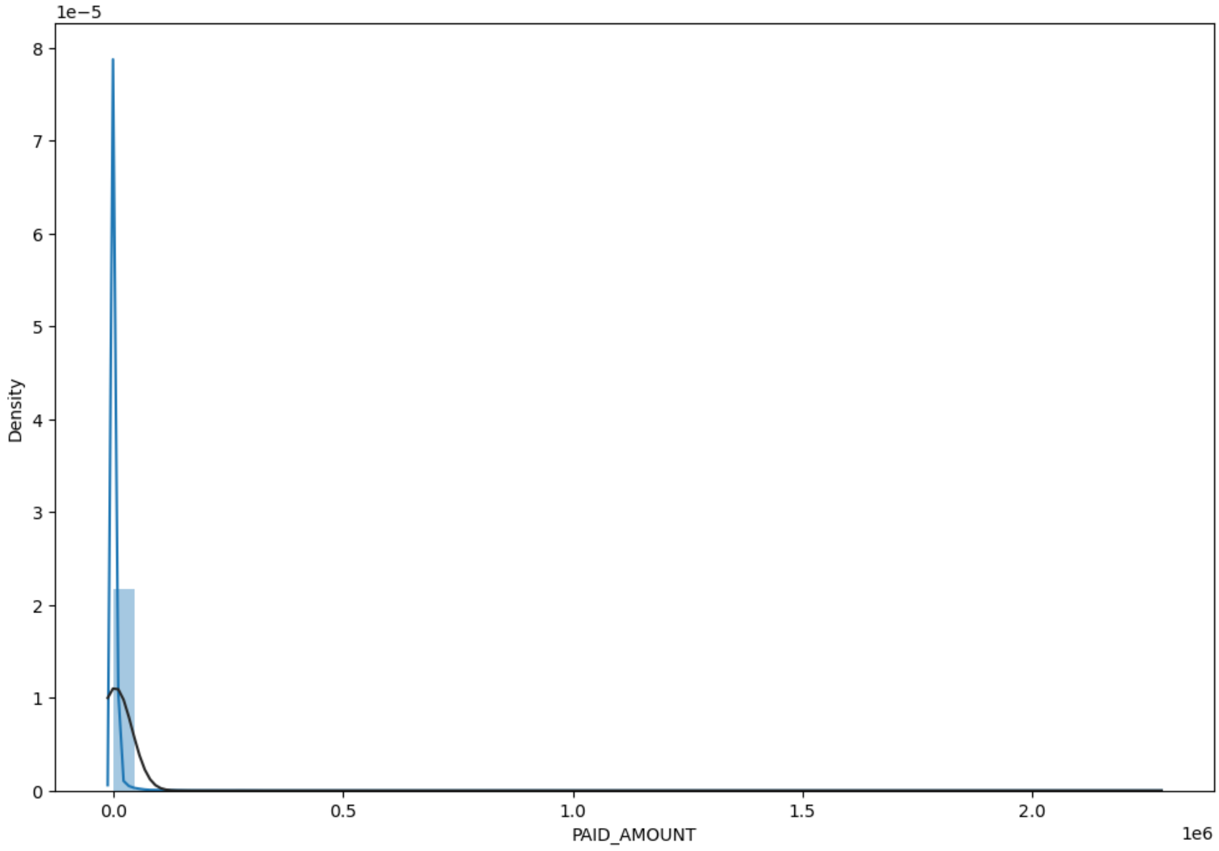
In [583…

```
sns.distplot(df['PAID_AMOUNT'], fit=norm)
```

```
/var/folders/36/tcpthx013zjb5h_v448f323r0000gn/T/ipykernel_15266/20138777
81.py:1: UserWarning:



`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.

Please adapt your code to use either `displot` (a figure-level function w
ith
similar flexibility) or `histplot` (an axes-level function for histograms
).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

Out[583]:

```
<Axes: xlabel='PAID_AMOUNT', ylabel='Density'>
```



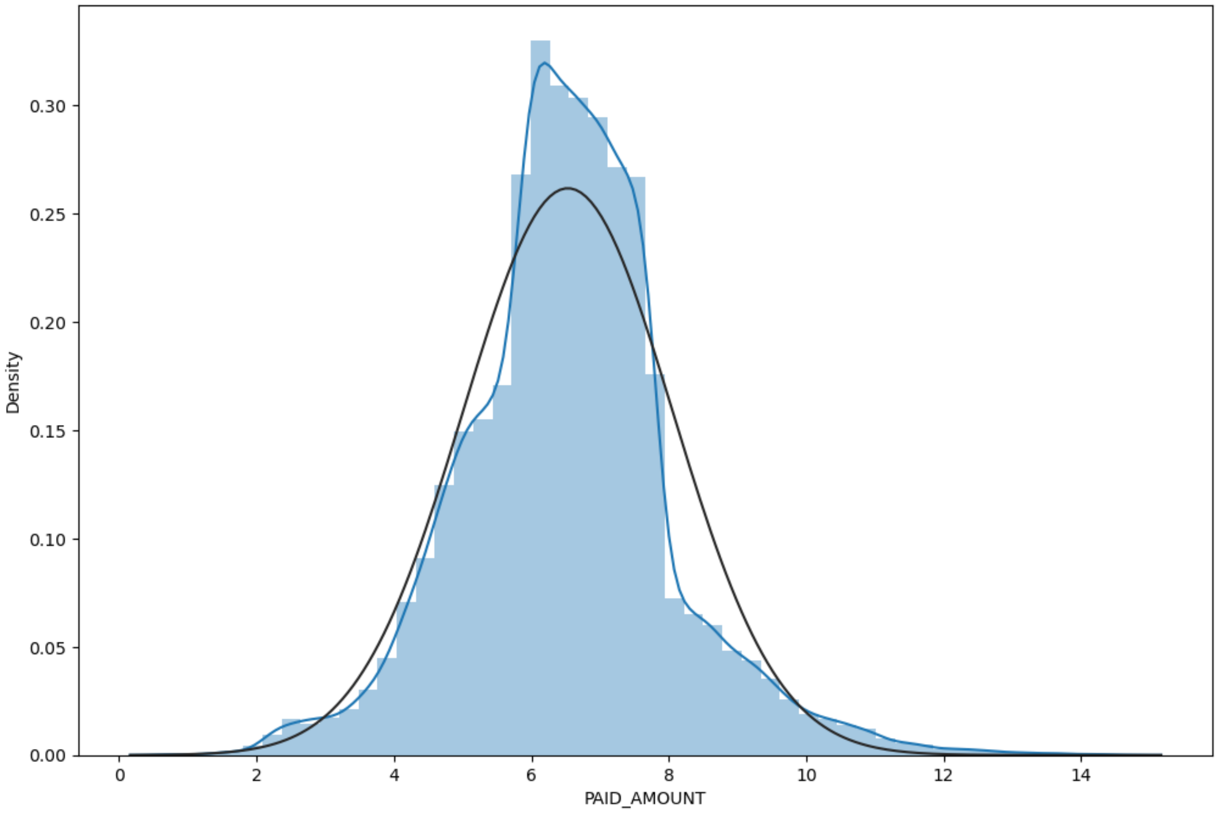
In [584…

```
sns.distplot(np.log(df['PAID_AMOUNT']+1), fit=norm)
```

```
/var/folders/36/tcpthx013zjb5h_v448f323r0000gn/T/ipykernel_15266/19526025
32.py:1: UserWarning:



`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.

Please adapt your code to use either `displot` (a figure-level function w
ith
similar flexibility) or `histplot` (an axes-level function for histograms
).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

Out[584]:

```
<Axes: xlabel='PAID_AMOUNT', ylabel='Density'>
```

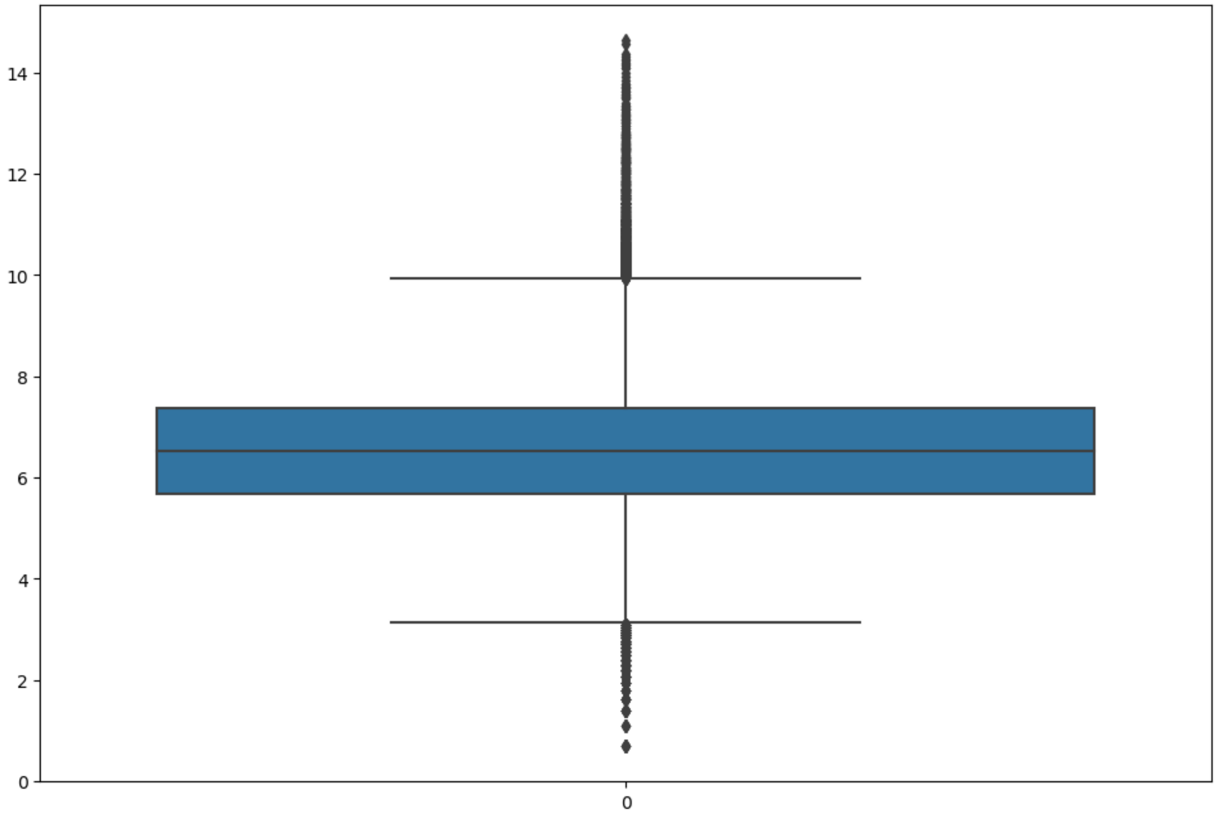


In [585…

```
sns.boxplot(np.log(df['PAID_AMOUNT']+1))
```

Out[585]:

```
<Axes: >
```



In [586…

```
df.head(1)
```

Out[586]:

	MONTH	SERVICE_CATEGORY	CLAIM_SPECIALTY	PAYER	PAID_AMOUNT	HALF_YEAI
0	2018-01-01	AncillaryFFS	nan	Payer F	4281	2018-06-0

In [587…

```
df['PAID_AMOUNT_LOG'] = np.log(df['PAID_AMOUNT']+1)
```

In [588…

```
df.head(1)
```

Out[588]:

	MONTH	SERVICE_CATEGORY	CLAIM_SPECIALTY	PAYER	PAID_AMOUNT	HALF_YEAI
0	2018-01-01	AncillaryFFS	nan	Payer F	4281	2018-06-0

In [601…

```
df_grp = df.groupby(['SERVICE_CATEGORY', 'CLAIM_SPECIALTY_GRP'])['PAID_AM
df_grp['up_bound'] = df_grp['mean'] + 3 * df_grp['std']
df_grp['low_bound'] = df_grp['mean'] - 3 * df_grp['std']
```

In [602…

```
df_new = df.merge(df_grp, on = ['SERVICE_CATEGORY', 'CLAIM_SPECIALTY_GRP'
```

In [614…

```
df_new.head(1)
```

Out[614]:

	MONTH	SERVICE_CATEGORY	CLAIM_SPECIALTY	PAYER	PAID_AMOUNT	HALF_YEAI
0	2018-01-01	AncillaryFFS	nan	Payer F	4281	2018-06-0

In [725…

```
df_new[['MONTH', 'SERVICE_CATEGORY', 'CLAIM_SPECIALTY', 'PAYER', 'PAID_AM
```

In [604…

```
df_new['is_lower_outlier'] = df_new.apply(lambda x: 1 if x['PAID_AMOUNT_L
df_new['is_upper_outlier'] = df_new.apply(lambda x: 1 if x['PAID_AMOUNT_L
```

Вывод: выбросы по нижней границе содержат 154 строк, что незначительно от общего числа выплат ()

In [637…

```
df1 = df_new.groupby(['HALF_YEAR', 'SERVICE_CATEGORY', 'CLAIM_SPECIALTY_G
df_out = df_new[df_new['is_upper_outlier'] == 1].groupby(['PAYER','HALF_Y
```

In [638…

```
df_out.columns = ['PAYER','HALF_YEAR', 'SERVICE_CATEGORY','CLAIM_SPECIALT
```

In [639…

```
df_out = df_out.merge(df1, on = ['HALF_YEAR', 'SERVICE_CATEGORY','CLAIM_S
```

In [640…

```
df_out['percent_outliers'] = df_out['UPPER_OUTLIERS']/df_out['PAID_AMOUNT
```

In [642…

```
df_out[df_out['percent_outliers'] > 0].sort_values(by = 'percent_outliers
```

Out[642]:

	PAYER	HALF_YEAR	SERVICE_CATEGORY	CLAIM_SPECIALTY_GRP	UPPER_OUTLIERS
8	Payer F	2019-06-01	InpatientServices	other	7757673.0
11	Payer F	2019-12-01	InpatientServices	other	6795904.0
14	Payer F	2020-06-01	InpatientServices	other	5029680.0
1	Payer F	2018-06-01	InpatientServices	other	8153070.0
4	Payer F	2018-12-01	InpatientServices	other	5975374.0
22	Payer S	2018-06-01	InpatientServices	emergency	234789.0
23	Payer S	2018-12-01	InpatientServices	emergency	238095.0
10	Payer F	2019-06-01	OutpatientServices	other	1599293.0
19	Payer H	2019-12-01	SNFServices	hospital	16807.0
13	Payer F	2019-12-01	OutpatientServices	other	931708.0
3	Payer F	2018-06-01	OutpatientServices	other	995860.0
21	Payer H	2020-06-01	SpecialistFFS	other	432457.0
20	Payer H	2019-12-01	SpecialistFFS	other	406283.0
15	Payer F	2020-06-01	OutpatientServices	other	535230.0
16	Payer H	2018-06-01	SpecialistFFS	other	272388.0
7	Payer F	2018-12-01	OutpatientServices	other	611795.0
18	Payer H	2019-06-01	SpecialistFFS	other	247003.0
5	Payer O	2018-12-01	InpatientServices	other	1571705.0
17	Payer H	2018-12-01	SpecialistFFS	other	126208.0
2	Payer O	2018-06-01	InpatientServices	other	1607934.0
0	Payer CA	2018-12-01	ASCServices	surgery	48486.0

6	Payer F	2018-12-01	InpatientServices	surgery	40136.
12	Payer O	2019-12-01	InpatientServices	other	325711.
9	Payer O	2019-06-01	InpatientServices	other	296576.

Выводы по аномалиям. Из вышеуказанной таблицы можно сделать выводы, что в основном высокие затраты на выплты от страховых компаний приходятся на вид медицинских услуг - прочее (ранее категорировалось).

Так, например за 1 квартал 2019 года компания F в категории InpatientServices платы по прочим врачебным специализациям составили 80% от аналогичного периода по всем остальным компаниям в этой же категории, аналогичная ситуация во 2-ом квартале 2019 и 1ом 2020 года.

Попытка прогноза на пол-года. Попробуем спрогнозировать цену на 2 квартал 2020 года. Применяем SARIMAX

In [690…

```
def plot_data(df):
    df.plot(figsize=(16,4))
    plt.xticks(rotation=45)
    plt.show()
```

In [708…

```
url = 'https://docs.google.com/spreadsheets/d/12o1iofQx6V-UhInjUjLjpKxx3Z
data_series = pd.read_csv(url)
```

In [709…

```
data_series = data_series[data_series['MONTH'].isin([201900, 202007]) ==
data_series['MONTH'] = data_series['MONTH'].astype(str)
data_series['MONTH'] = pd.to_datetime(data_series['MONTH'], format='%Y%m'
```

In [710…

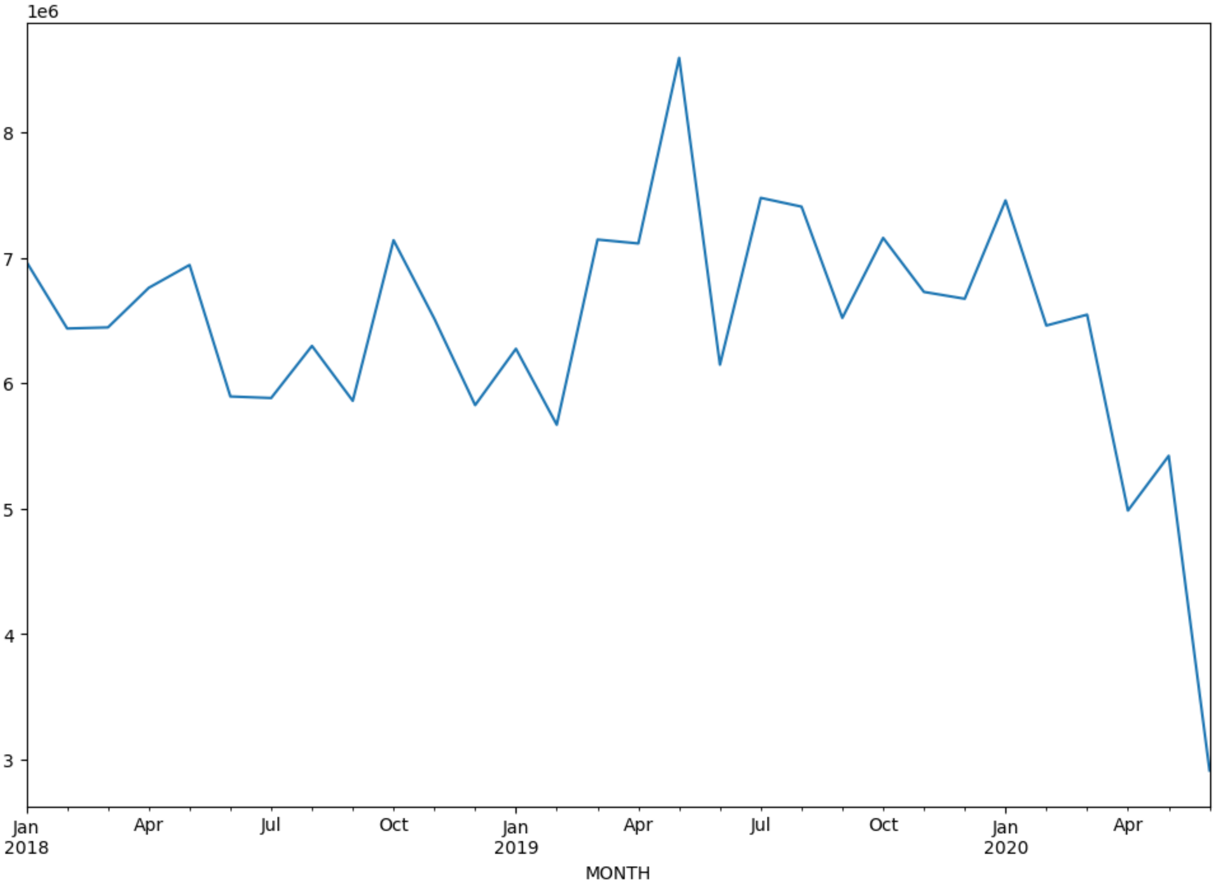
```
data_series.head(1)
```

Out[710]:

	MONTH	SERVICE_CATEGORY	CLAIM_SPECIALTY	PAYER	PAID_AMOUNT
0	2018-01-01	AncillaryFFS	NaN	Payer F	4281

In [711...
```
df.groupby('MONTH')['PAID_AMOUNT'].sum().plot()
```

Out[711]:
```
<Axes: xlabel='MONTH'>
```



In [712...
```
data_series = data_series.groupby('MONTH')['PAID_AMOUNT'].sum().reset_ind
```

In [713...
```
data_series = data_series.set_index('MONTH')
```

In [762...
```
data_series
```

Out[762]:

	PAID_AMOUNT
MONTH	
2018-01-01	6959445
2018-02-01	6430877
2018-03-01	6436167
2018-04-01	6748037
2018-05-01	6937332
2018-06-01	5888847
2018-07-01	5874723
2018-08-01	6268040
2018-09-01	5818710
2018-10-01	7103820
2018-11-01	6505783
2018-12-01	5805446
2019-01-01	6264273
2019-02-01	5665218
2019-03-01	7091354
2019-04-01	7101915
2019-05-01	8558681
2019-06-01	6144328
2019-07-01	7473785
2019-08-01	7403710
2019-09-01	6519659
2019-10-01	7156007
2019-11-01	6726444
2019-12-01	6672237
2020-01-01	7449768
2020-02-01	6455708
2020-03-01	6541616
2020-04-01	4981590
2020-05-01	5415924
2020-06-01	2913437

In [767…

```
# Обучение модели
model = SARIMAX(data_series, order=(6,1,1), seasonal_order=(1,0,0,12), fr
results = model.fit()

# Прогнозирование на 2ое полугодие 2020
forecast = results.get_forecast(steps=6)
forecast_ci = forecast.conf_int()

# Вывод результатов
print(forecast.predicted_mean)
```

```
/Users/viktoriakalasnikova/anaconda3/lib/python3.10/site-packages/statsmo
dels/tsa/base/tsa_model.py:471: ValueWarning:

No frequency information was provided, so inferred frequency MS will be u
sed.

 This problem is unconstrained.
```

```
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            9     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  1.54806D+01    |proj g|=  3.53557D-01

At iterate    5    f=  1.46145D+01    |proj g|=  1.91871D-01

At iterate   10    f=  1.45942D+01    |proj g|=  6.67546D-03

At iterate   15    f=  1.45910D+01    |proj g|=  8.21265D-02

At iterate   20    f=  1.45842D+01    |proj g|=  4.14871D-03

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    9     23     28      1     0     0   2.852D-05   1.458D+01
  F =   14.584236990351368

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
2020-07-01    3.580341e+06
2020-08-01    3.541477e+06
2020-09-01    3.492311e+06
2020-10-01    3.772339e+06
2020-11-01    3.821688e+06
2020-12-01    3.930559e+06
Freq: MS, Name: predicted_mean, dtype: float64
```

Выводы: Ожидаемая прибыль на 2-ое полугодие 2020 - от 3,5 до 4 млн в месяц, если параметрах модели задать период 6 месяцев - для прогнозирование следующего значения

```
In [ ]:
```