```
import os
import pandas as pd
import pyarrow.parquet as pq
import sklearn
import matplotlib.pyplot as plt
import gc # для gc.collect()

cwd = r'C:\Projects\ODS_Avito_hack'
os.chdir(cwd)
os.getcwd()
```

```
'C:\\Projects\\ODS_Avito_hack'
```

```
#Source
train_data = cwd + r'\train.parquet'
test_data = cwd + r'\test.parquet.parquet'
categories_data = cwd + r'\categories.parquet.csv.parquet' # данные о
логических категориях
campaigns_data = cwd + r'\campaigns_meta.parquet.parquet' # данные о
рекламных кампаниях

# Загрузка данных с указанием batch_size
train_pq = pq.ParquetFile(train_data)
# обычный способ
train = pd.read_parquet(train_data)
test = pd.read_parquet(test_data)
categories = pd.read_parquet(categories_data)
campaigns = pd.read_parquet(campaigns_data)

# Бейзлайн от Авито - если пользователь уже видел рекламу и кликнул на
нее, то он снова сделает клик. Если же реклама была показана, но клик
не последовал, то и в следующий раз клик маловероятен.
user_ads_clicks = train.groupby(["user_id", "adv_campaign_id"],
as_index=False)["target"].max()
test = test.merge(user_ads_clicks, on=["user_id", "adv_campaign_id"],
how="left")
test["predict"] = test["target"].fillna(0.5)
test[["user_id", "adv_campaign_id",
"predict"]].to_csv("sample_submission.csv", index=False)

 #|-- platform_id: id платформы (Android, Ios и т.п.)
 #|-- user_id: id Пользователя
 #|-- adv_campaign_id: id рекламной компании
 #|-- target: клик / не клик
 #|-- banner_code: код баннера
 #|-- adv_creative_id: индификатор креатива
 #|-- event_date: date Дата показа рекламной кампании пользователю
 #|-- is_main: boolean True - показ рекламы был осуществлен с главной
страницы
```

```
print(train.info())
print('**********************************')
print(train.head())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 114741035 entries, 0 to 114741034
Data columns (total 8 columns):
 #   Column           Dtype 
---  ------           ----- 
 0   user_id          int64 
 1   adv_campaign_id  int64 
 2   platform_id      int64 
 3   adv_creative_id  int64 
 4   event_date       object
 5   banner_code      int64 
 6   is_main          bool  
 7   target           int32 
dtypes: bool(1), int32(1), int64(5), object(1)
memory usage: 5.7+ GB
None
**********************************
   user_id  adv_campaign_id  platform_id  adv_creative_id  event_date
\
0  2853707             3352            3             3075  2024-09-17

1  2537244             4029            2             3260  2024-09-17

2    63033             1578            3             1109  2024-09-17

3   164702             3434            1             1079  2024-09-17

4  2802905             2208            3             3576  2024-09-17


   banner_code  is_main  target
0            6     True       0
1            8     True       0
2            6     True       0
3            7     True       0
4            6     True       0

###### ДОРАБОТКИ, НОРМАЛИЗАЦИЯ ДАТАСЕТОВ ###########
train['event_date']=pd.to_datetime(train['event_date']) #
преобразовали в дату
train["target"] = train['target'].astype(bool)
train["platform_id"] = train['platform_id'].astype('int8')
train["adv_campaign_id"] = train['adv_campaign_id'].astype('int16')
train["adv_creative_id"] = train['adv_creative_id'].astype('int16')
train["banner_code"] = train['banner_code'].astype('int8')
train["user_id"] = train['user_id'].astype('int32')
```

```
print(train.info())
print('**********************************')
print(train.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 114741035 entries, 0 to 114741034
Data columns (total 8 columns):
 #   Column           Dtype
---  ------           -----
 0   user_id          int32
 1   adv_campaign_id  int16
 2   platform_id      int8
 3   adv_creative_id  int16
 4   event_date       datetime64[ns]
 5   banner_code      int8
 6   is_main          bool
 7   target           bool
dtypes: bool(2), datetime64[ns](1), int16(2), int32(1), int8(2)
memory usage: 2.1 GB
None
**********************************
   user_id  adv_campaign_id  platform_id  adv_creative_id
event_date  \
0  2853707             3352            3             3075 2024-09-17

1  2537244             4029            2             3260 2024-09-17

2    63033             1578            3             1109 2024-09-17

3   164702             3434            1             1079 2024-09-17

4  2802905             2208            3             3576 2024-09-17


   banner_code  is_main  target
0            6     True   False
1            8     True   False
2            6     True   False
3            7     True   False
4            6     True   False
```

```
gc.collect()
```

```
2248
```

```
print(train.describe())
```

```
----------------------------------------------------------------------
-----
MemoryError                               Traceback (most recent call
last)
```

```
Cell In[63], line 1
----> 1 print(train.describe())

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\
generic.py:10819, in NDFrame.describe(self, percentiles, include,
exclude)
  10577 @final
  10578 def describe(
  10579     self: NDFrameT,
   (...)
  10582     exclude=None,
  10583 ) -> NDFrameT:
  10584     """
  10585     Generate descriptive statistics.
  10586 
   (...)
  10817     max             NaN      3.0
  10818     """
> 10819     return describe_ndframe(
  10820         obj=self,
  10821         include=include,
  10822         exclude=exclude,
  10823         percentiles=percentiles,
  10824     )

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\methods\
describe.py:94, in describe_ndframe(obj, include, exclude,
percentiles)
     87 else:
     88     describer = DataFrameDescriber(
     89         obj=cast("DataFrame", obj),
     90         include=include,
     91         exclude=exclude,
     92     )
---> 94 result = describer.describe(percentiles=percentiles)
     95 return cast(NDFrameT, result)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\methods\
describe.py:162, in DataFrameDescriber.describe(self, percentiles)
    161 def describe(self, percentiles: Sequence[float] | np.ndarray)
-> DataFrame:
--> 162     data = self._select_data()
    164     ldesc: list[Series] = []
    165     for _, series in data.items():

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\methods\
describe.py:183, in DataFrameDescriber._select_data(self)
    180 if (self.include is None) and (self.exclude is None):
    181     # when some numerics are found, keep only numerics
    182     default_include: list[npt.DTypeLike] = [np.number,
"datetime"]
--> 183     data = self.obj.select_dtypes(include=default_include)
    184     if len(data.columns) == 0:
    185         data = self.obj

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\
frame.py:4708, in DataFrame.select_dtypes(self, include, exclude)
   4704             return False
   4706     return True
-> 4708 mgr = self._mgr._get_data_subset(predicate).copy(deep=None)
   4709 return type(self)(mgr).__finalize__(self)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\internals\
managers.py:664, in BaseBlockManager.copy(self, deep)
    661         res._blklocs = self._blklocs.copy()
    663 if deep:
--> 664     res._consolidate_inplace()
    665 return res

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\internals\
managers.py:1829, in BlockManager._consolidate_inplace(self)
   1823 def _consolidate_inplace(self) -> None:
   1824     # In general, _consolidate_inplace should only be called
via
   1825     #  DataFrame._consolidate_inplace, otherwise we will fail
to invalidate
   1826     #  the DataFrame's _item_cache. The exception is for
newly-created
   1827     #  BlockManager objects not yet attached to a DataFrame.
   1828     if not self.is_consolidated():
-> 1829         self.blocks = _consolidate(self.blocks)
   1830         self._is_consolidated = True
   1831         self._known_consolidated = True

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\internals\
managers.py:2272, in _consolidate(blocks)
   2270 new_blocks: list[Block] = []
   2271 for (_can_consolidate, dtype), group_blocks in grouper:
-> 2272     merged_blocks, _ = _merge_blocks(
   2273         list(group_blocks), dtype=dtype,
can_consolidate=_can_consolidate
   2274     )
   2275     new_blocks = extend_blocks(merged_blocks, new_blocks)
   2276 return tuple(new_blocks)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\internals\
managers.py:2304, in _merge_blocks(blocks, dtype, can_consolidate)
   2301     new_values = bvals2[0]._concat_same_type(bvals2, axis=0)
   2303 argsort = np.argsort(new_mgr_locs)
-> 2304 new_values = new_values[argsort]
```

```
   2305 new_mgr_locs = new_mgr_locs[argsort]
   2307 bp = BlockPlacement(new_mgr_locs)

MemoryError: Unable to allocate 3.42 GiB for an array with shape (4,
114741035) and data type int64
```

```python
# Проверка типов
print(train['user_id'].min())
print(train['user_id'].max())
# print(sorted(train['user_id'].unique()))
```

```
1
3263622
```

```
IOPub data rate exceeded.
The Jupyter server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--ServerApp.iopub_data_rate_limit`.

Current values:
ServerApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
ServerApp.rate_limit_window=3.0 (secs)
```

```python
# Итерация по батчам (ТЕСТ)
for batch in train.iter_batches(batch_size=10000):
    # Преобразование батча в pandas DataFrame
    df_batch = batch.to_pandas()
    # Обработка батча
    print(df_batch.describe())
```

```
----------------------------------------------------------------------
-----
AttributeError                           Traceback (most recent call
last)
Cell In[20], line 2
      1 # Итерация по батчам (тест)
----> 2 for batch in train.iter_batches(batch_size=10000):
      3     # Преобразование батча в pandas DataFrame
      4     df_batch = batch.to_pandas()
      5     # Обработка батча

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\core\
generic.py:5989, in NDFrame.__getattr__(self, name)
   5982 if (
   5983     name not in self._internal_names_set
   5984     and name not in self._metadata
   5985     and name not in self._accessors
   5986     and
self._info_axis._can_hold_identifiers_and_holds_name(name)
```

```
   5987 ):
   5988     return self[name]
-> 5989 return object.__getattribute__(self, name)

AttributeError: 'DataFrame' object has no attribute 'iter_batches'
```

```python
print(train.isnull().sum())
```

```
user_id            0
adv_campaign_id    0
platform_id        0
adv_creative_id    0
event_date         0
banner_code        0
is_main            0
target             0
dtype: int64
```

```python
print(train['target'].value_counts(normalize=True))
```

```
target
0    0.99463
1    0.00537
Name: proportion, dtype: float64
```

```python
#|-- microcat_id: id микрокатегории
#|-- level_id: id уровня в дереве микрокатегорий
#|-- parent_microcat_id: id родительской микрокатегории
#|-- logcat_id: id логической категории
#|-- vertical_id: id вертикали
#|-- category_id: id категории

print(categories.head())
print('**********************************')
print(categories.describe())
print('**********************************')
print(categories.isnull().sum())
```

```
   microcat_id  level_id  parent_microcat_id  logcat_id
vertical_id  \
0        33482       7.0             40172.0       54.0          3.0

1        27254       5.0             48637.0       55.0          5.0

2        37005       6.0             15332.0       54.0          3.0

3        31376       8.0             28137.0       58.0          8.0

4        20493       4.0             18343.0       24.0          8.0
```

```
   category_id
0          3.0
1          4.0
2          3.0
3          4.0
4          9.0
**********************************
        microcat_id      level_id  parent_microcat_id     logcat_id  \
count  25891.000000  25890.000000        25888.000000  25883.000000   
mean   25094.792438      6.021514        25007.398061     33.397867   
std    14417.501552      0.887269        14519.892783     18.233059   
min        3.000000      1.000000           12.000000      1.000000   
25%    12599.000000      5.000000        12272.000000     18.000000   
50%    25201.000000      6.000000        24291.000000     37.000000   
75%    37631.000000      7.000000        38448.000000     54.000000   
max    49951.000000      8.000000        49936.000000     66.000000   

        vertical_id   category_id  
count  25883.000000  25887.000000  
mean       5.096859      3.956503  
std        1.651994      1.402598  
min        1.000000      1.000000  
25%        4.000000      3.000000  
50%        5.000000      4.000000  
75%        5.000000      4.000000  
max        9.000000     12.000000  
**********************************
microcat_id           0
level_id              1
parent_microcat_id    3
logcat_id             8
vertical_id           8
category_id           4
dtype: int64

#|-- adv_campaign_id: id рекламной компании
#|-- start_date: date дата начала рекламной компании
#|-- end_date: date дата завершения рекламной компании
#|-- goal_cost: цена за клик на рекламу
#|-- goal_budget: общий бюджет рекламной компании
#|-- logcat_id: id логической категории товаров из рекламной кампании
#|-- location_ids: id локации, на которую рекламная компания
распространяется

print(campaigns.head())
print(campaigns.describe())
print(campaigns.isnull().sum())

   adv_campaign_id  start_date    end_date  goal_cost  goal_budget  \
0             2153  2024-09-21  2024-10-02   6.661659  9429.056096   
```

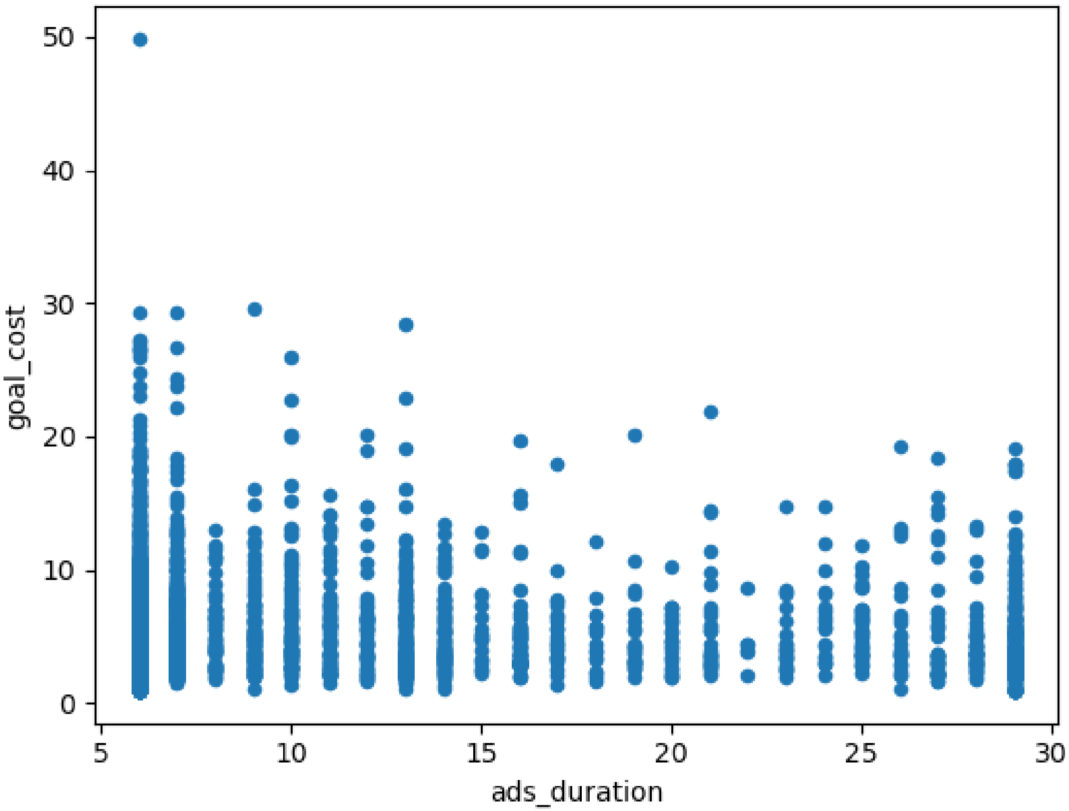
```
1             3103  2024-09-10  2024-09-16   2.853378  3844.482933
2             2816  2024-09-10  2024-09-17   3.058230  1455.156612
3             3603  2024-09-10  2024-09-16   4.395015  2592.232475
4             1328  2024-09-10  2024-09-16   3.891329  2836.139672

   location_id  logcat_id
0           70         59
1           30         40
2           56         65
3           30         50
4           30         51
       adv_campaign_id    goal_cost    goal_budget  location_id
logcat_id
count       4031.00000  4031.000000    4031.000000  4031.000000
4031.000000
mean        2099.56512     5.242840    7113.443134    39.466882
40.219052
std         1213.41340     3.547009   14102.599591    21.446496
18.289209
min            1.00000     0.950574       6.702396     1.000000
1.000000
25%         1049.50000     3.018347    1002.285864    27.000000
26.000000
50%         2103.00000     4.221662    3282.187078    46.000000
40.000000
75%         3152.50000     6.437592    7509.600093    55.500000
56.000000
max         4200.00000    49.866865  227679.963364    79.000000
66.000000
adv_campaign_id    0
start_date         0
end_date           0
goal_cost          0
goal_budget        0
location_id        0
logcat_id          0
dtype: int64

# Корреляционная матрица (Пирсон)
correlation_matrix = train.corr()
print(correlation_matrix)

                  user_id  adv_campaign_id  platform_id
adv_creative_id  \
user_id          1.000000         0.000245     0.001500        -
0.000136
adv_campaign_id  0.000245         1.000000    -0.005439
0.014144
platform_id      0.001500        -0.005439     1.000000
0.003247
```

```
adv_creative_id -0.000136         0.014144     0.003247
1.000000
event_date      -0.000360         0.021205    -0.009016        -
0.005474
banner_code     -0.000742         0.002935    -0.612264        -
0.007801
is_main          0.000354        -0.001313    -0.054624        -
0.006872
target           0.000007         0.000171    -0.009822        -
0.001688

                 event_date  banner_code   is_main    target
user_id           -0.000360    -0.000742  0.000354  0.000007
adv_campaign_id    0.021205     0.002935 -0.001313  0.000171
platform_id       -0.009016    -0.612264 -0.054624 -0.009822
adv_creative_id   -0.005474    -0.007801 -0.006872 -0.001688
event_date         1.000000    -0.000413 -0.003765  0.004787
banner_code       -0.000413     1.000000  0.681946 -0.006958
is_main           -0.003765     0.681946  1.000000 -0.025686
target             0.004787    -0.006958 -0.025686  1.000000

# JOIN 2x датасетов
merge_df_1 = pd.merge(train, campaigns, on='adv_campaign_id',
how='left')

#print(merge_df_1.info())
print('**********************************')
print(merge_df_1.head())

**********************************
   user_id  adv_campaign_id  platform_id  adv_creative_id
event_date  \
0  2853707             3352            3             3075 2024-09-17

1  2537244             4029            2             3260 2024-09-17

2    63033             1578            3             1109 2024-09-17

3   164702             3434            1             1079 2024-09-17

4  2802905             2208            3             3576 2024-09-17


   banner_code  is_main  target  start_date    end_date  goal_cost  \
0            6     True       0  2024-09-16  2024-09-25   5.131051
1            8     True       0  2024-09-16  2024-09-22   4.931622
2            6     True       0  2024-09-04  2024-09-18   3.711480
3            7     True       0  2024-09-17  2024-09-24   4.030369
4            6     True       0  2024-09-16  2024-09-22   4.931534

   goal_budget  location_id  logcat_id
```

```
0  2647.795831          9        56
1  6953.261023         46        65
2  7035.724050          1        65
3  5034.412852         47        65
4  7024.725026         46        65
```

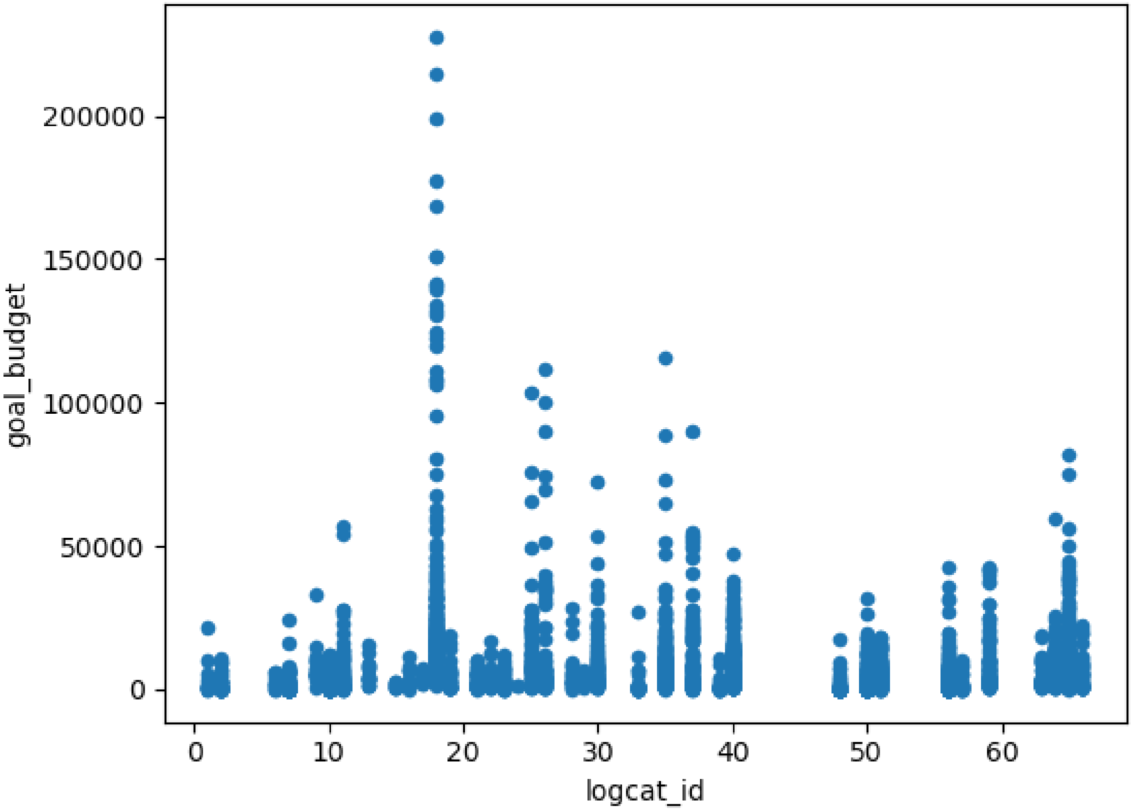
```
# проверка фич

# Смотрим campaigns
campaigns.start_date = pd.to_datetime(campaigns.start_date)
campaigns.end_date = pd.to_datetime(campaigns.end_date)
campaigns['ads_duration'] = (campaigns.end_date -
campaigns.start_date).dt.days
campaigns.plot.scatter(x='ads_duration', y='goal_cost')
```

```
<Axes: xlabel='ads_duration', ylabel='goal_cost'>
```



```
campaigns.plot.scatter(x='logcat_id', y='goal_budget')
```

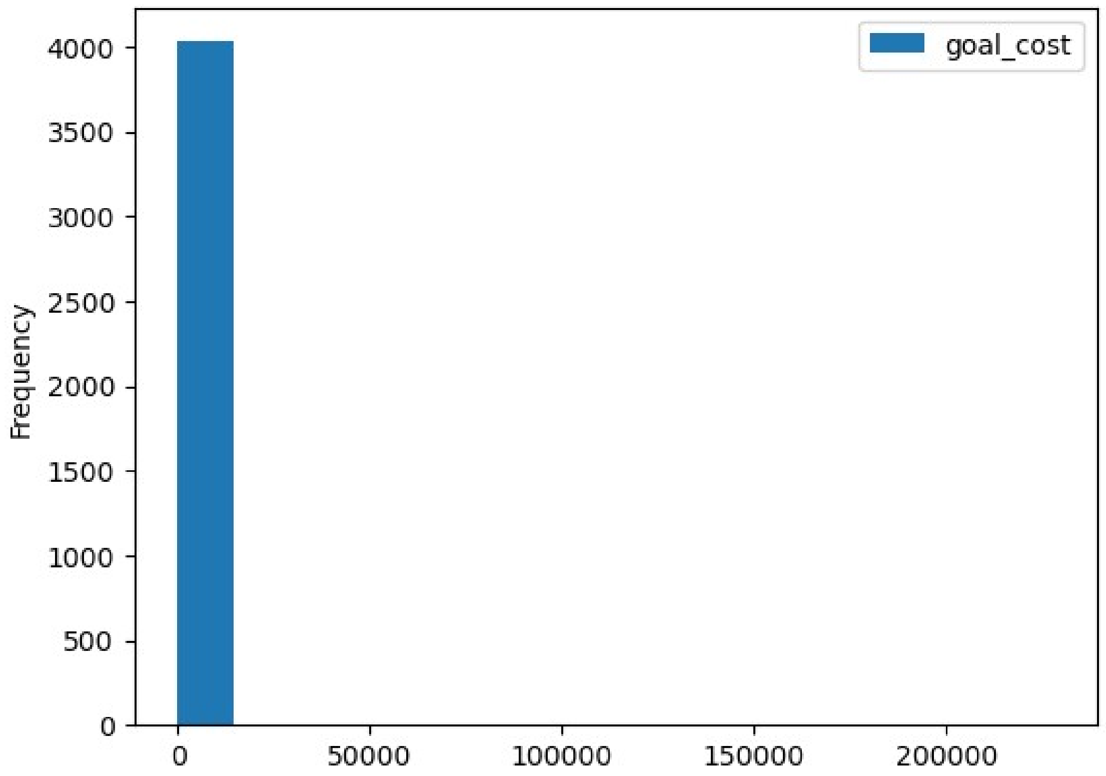
```
<Axes: xlabel='logcat_id', ylabel='goal_budget'>
```



Выводы: длительность рекламы и ее направление не критично влияет
на стоимость - есть просто дорогая и дешевая реклама

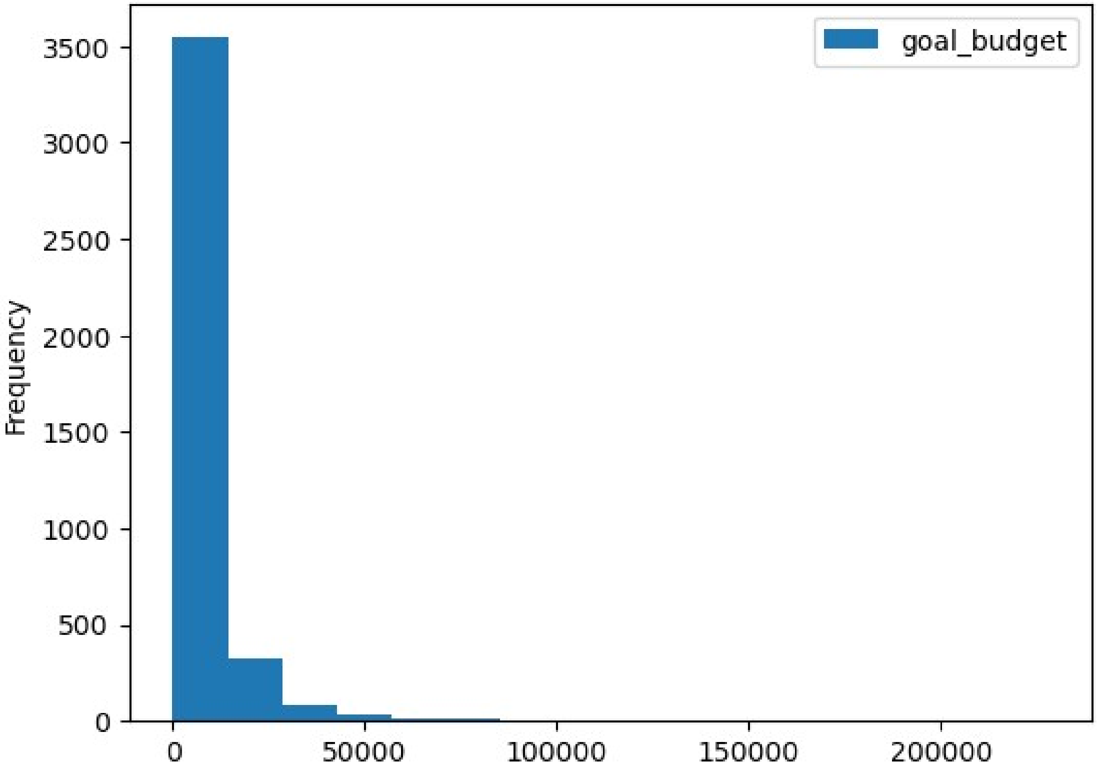
```
campaigns.plot.hist(column='goal_cost', bins=16)
```

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



```
campaigns.plot.hist(column='goal_budget', bins=16)
```

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



```
campaigns[campaigns['goal_budget'] >
100000].sort_values(by='goal_budget', ascending=False)
```

```
      adv_campaign_id  start_date    end_date  goal_cost
goal_budget  \
2238             2155  2024-09-17  2024-09-23  17.779562
227679.963364
3510              739  2024-09-18  2024-09-30  13.372574
214395.923782
3463              217  2024-08-20  2024-08-31  14.050671
198957.100367
2822             1535  2024-07-19  2024-07-26  14.826908
177596.710739
2821             2388  2024-07-22  2024-07-31  12.142784
168557.254495
2871             2456  2024-09-12  2024-09-20  13.029493
151036.590034
172              2656  2024-07-29  2024-08-05  13.373868
150649.439149
2347             1453  2024-08-08  2024-08-31   8.549973
141543.498925
2870              699  2024-09-12  2024-09-19  13.839542
140677.624582
3606             1674  2024-08-08  2024-08-31   8.307261
139413.638385
```

```
1964             1809  2024-08-08  2024-09-04   6.955160
133876.688625
737              4080  2024-08-08  2024-08-31   8.168774
131834.656011
2830             2160  2024-08-27  2024-09-03  12.816965
130381.090468
825              1395  2024-08-08  2024-09-02   6.898775
124257.868872
3114               74  2024-08-30  2024-09-28  11.028458
122677.087615
3844             1254  2024-08-27  2024-09-02  12.565424
120023.552766
3749              272  2024-09-17  2024-10-16  10.484921
115893.600276
1628             1194  2024-09-10  2024-09-30   3.342502
111807.927745
3840             3862  2024-07-22  2024-07-28  12.861852
110952.476721
3489             3873  2024-09-11  2024-09-17  13.467875
108175.844872
284              1693  2024-09-12  2024-09-22  13.044078
107535.875532
2255              435  2024-08-07  2024-08-13  12.089944
106419.784027
2360             3342  2024-09-03  2024-09-30  15.454881
103374.543089
1739             3874  2024-08-16  2024-08-30   3.775655
100310.363301

      location_id  logcat_id
2238           46         18
3510           30         18
3463           30         18
2822           30         18
2821           46         18
2871           30         18
172            46         18
2347           47         18
2870           46         18
3606           47         18
1964           47         18
737            47         18
2830           46         18
825            47         18
3114           46         18
3844           46         18
3749           46         35
1628           46         26
3840           46         18
```

```
3489           30         18
284            31         18
2255           46         18
2360           30         25
1739           46         26
```

```
campaigns['price_per_day'] = campaigns.goal_budget / 
campaigns.ads_duration
campaigns.corr()
```

```
                 adv_campaign_id  start_date  end_date  goal_cost  \
adv_campaign_id         1.000000   -0.006608 -0.007785   0.024338   
start_date             -0.006608    1.000000  0.863427   0.035714   
end_date               -0.007785    0.863427  1.000000   0.009941   
goal_cost               0.024338    0.035714  0.009941   1.000000   
goal_budget            -0.014800   -0.209563 -0.130941   0.347853   
location_id            -0.016127   -0.028201 -0.017183  -0.053019   
logcat_id              -0.000839   -0.014357 -0.034059  -0.247352   
price_per_day          -0.007470   -0.149300 -0.203493   0.371533   
ads_duration           -0.002262   -0.260024  0.262610  -0.049282   

                 goal_budget  location_id  logcat_id  price_per_day  \
adv_campaign_id    -0.014800    -0.016127  -0.000839      -0.007470   
start_date         -0.209563    -0.028201  -0.014357      -0.149300   
end_date           -0.130941    -0.017183  -0.034059      -0.203493   
goal_cost           0.347853    -0.053019  -0.247352       0.371533   
goal_budget         1.000000     0.007195  -0.120989       0.863990   
location_id         0.007195     1.000000  -0.018685      -0.006791   
logcat_id          -0.120989    -0.018685   1.000000      -0.111028   
price_per_day       0.863990    -0.006791  -0.111028       1.000000   
ads_duration        0.150198     0.021050  -0.037731      -0.103938   

                 ads_duration  
adv_campaign_id     -0.002262  
start_date          -0.260024  
end_date             0.262610  
goal_cost           -0.049282  
goal_budget          0.150198  
location_id          0.021050  
logcat_id           -0.037731  
price_per_day       -0.103938  
ads_duration         1.000000  
```

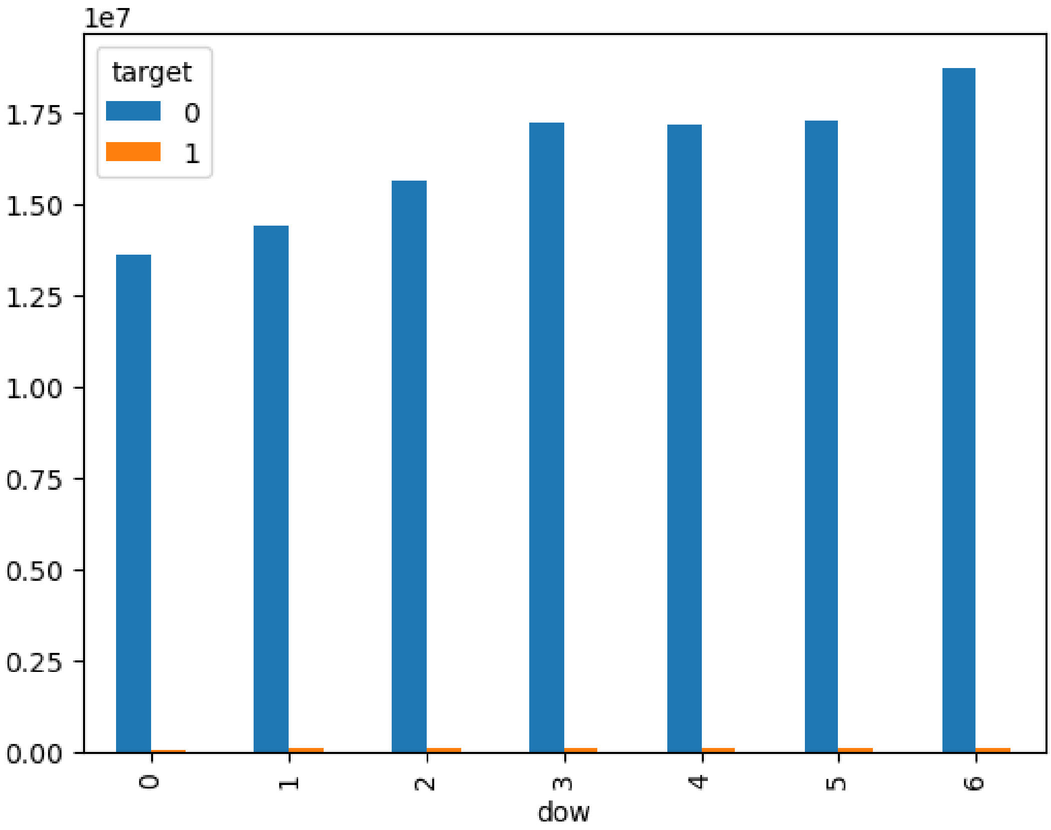
```
campaings_cat = campaigns.merge(categories, on='logcat_id')
print(campaings_cat.head())
```

```
   adv_campaign_id start_date   end_date  goal_cost  goal_budget  
location_id  \
0             2153 2024-09-21 2024-10-02   6.661659  9429.056096  
70
```

```
1             2153 2024-09-21 2024-10-02   6.661659  9429.056096
70
2             2153 2024-09-21 2024-10-02   6.661659  9429.056096
70
3             2153 2024-09-21 2024-10-02   6.661659  9429.056096
70
4             2153 2024-09-21 2024-10-02   6.661659  9429.056096
70

   logcat_id  price_per_day  ads_duration  microcat_id  level_id  \
0         59     857.186918            11        25237       4.0
1         59     857.186918            11        41723       4.0
2         59     857.186918            11        18532       4.0
3         59     857.186918            11         6006       3.0
4         59     857.186918            11        37286       4.0

   parent_microcat_id  vertical_id  category_id
0              6006.0          8.0          5.0
1              6006.0          8.0          5.0
2              6006.0          8.0          5.0
3             29785.0          8.0          5.0
4              6006.0          8.0          5.0
```

```
train['dow'] = train['event_date'].dt.weekday
count_data = train.groupby(['dow',
'target']).size().unstack(fill_value=0)
count_data.plot(kind='bar')
```

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



```
train[train['dow'] == 6]['target'].value_counts()
```

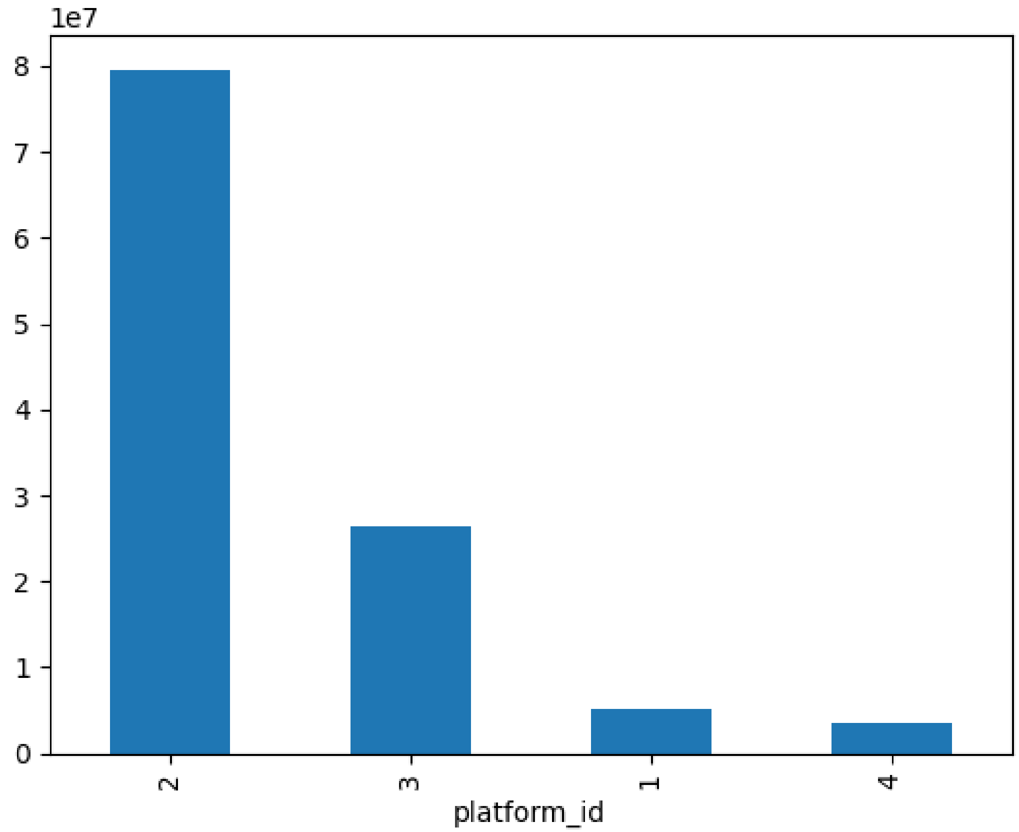
```
target
0    18733310
1      103418
Name: count, dtype: int64
```

```
train[train['dow'] == 0]['target'].value_counts()
```

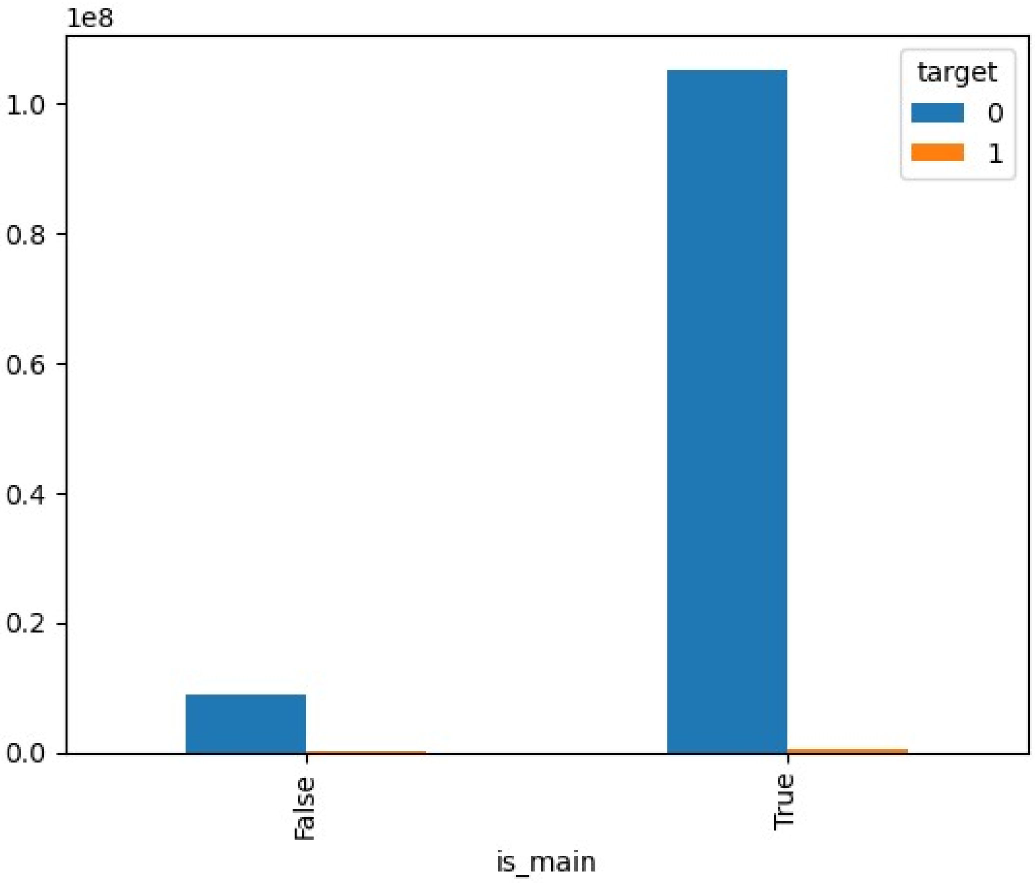
```
target
0    13609760
1       69815
Name: count, dtype: int64
```

```
train['platform_id'].value_counts().plot.bar()
```

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



```
count_data = train.groupby(['is_main',
'target']).size().unstack(fill_value=0).plot.bar()
```



```
[0 1]
```