pre_dobilling

May 7, 2025

[7]:
```
import pandas as pd
import psycopg2
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import numpy as np
import seaborn as sns
import statsmodels.stats.proportion as proportion
from scipy.stats import ttest_ind,mannwhitneyu,shapiro,norm
from statsmodels.stats.weightstats import ztest
from tqdm import tqdm
import timeit
from scipy import stats
import math
from datetime import date, datetime, timedelta
import time
from sqlalchemy import create_engine, text
from sqlalchemy.orm import sessionmaker
import warnings
warnings.filterwarnings("ignore")
import clickhouse_connect



from credential import postgres_secret,clickhouse_dwh_secret

def get_engine(user):
    if user == postgres_secret['user']:
        db_name = postgres_secret['db_name']
        password = postgres_secret['password']
        host = postgres_secret['host']
        engine = create_engine(f'postgresql://{user}:{password}@{host}:6432/
 ↪{db_name}')
    elif user == clickhouse_dwh_secret['user']:
            db_name = clickhouse_dwh_secret['db_name']
            password = clickhouse_dwh_secret['password']
            host = clickhouse_dwh_secret['host']
```

```
            engine = create_engine(f'clickhouse://{user}:{password}@{host}:8123/
 ↪{db_name}')
    return engine

connection_clickhouse = clickhouse_connect.get_client(
    host = clickhouse_dwh_secret['host'],
    port= '8123',
    username = clickhouse_dwh_secret['user'],
    password = clickhouse_dwh_secret['password'],
    database='datamarts'
    )


def execute(SQL, user):
    start_time = time.time()  #
    engine = get_engine(user)
    Session = sessionmaker(bind=engine)  # sessions factory ()
    with Session() as session: # open session
        result = session.execute(text(SQL))
        df = pd.DataFrame(result.fetchall(), columns=result.keys())

    end_time = time.time()  #
    execution_time = round(end_time - start_time,4) #

    print(f"              : {execution_time}    ")
    print()
    return df
```

[4]:
```
query = f'''SELECT * FROM datamarts.marketing_dash
            WHERE created_at BETWEEN '2024-08-01' AND '2025-01-31'
            AND domain_locale='ru'
            AND platform IN ('cloudpayments','payture')
            '''

df = execute(query,user='kmekhtiev')
df['created_at'] = pd.to_datetime(df['created_at'])
df['subscription_date'] =  df['created_at'].dt.strftime("%Y-%m-%d")
df['subscription_date'] =  pd.to_datetime(df['subscription_date'])
df['first_prolong_date'] = pd.to_datetime(df['first_prolong_date'])
df['first_payment_date'] = df['first_prolong_date'].dt.strftime("%Y-%m-%d")
df['first_payment_date'] =  pd.to_datetime(df['first_payment_date'])
```

```
              : 6.6395
```

[5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 47990 entries, 0 to 47989
Data columns (total 36 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   user_id             47990 non-null  object
 1   visitor_id          47990 non-null  object
 2   profile_id          47990 non-null  object
 3   reg_date            47990 non-null  object
 4   created_at          47990 non-null  datetime64[ns]
 5   ends_at             47990 non-null  object
 6   renewal_off_date    47990 non-null  datetime64[ns]
 7   state               47990 non-null  object
 8   price_cents         47990 non-null  int64
 9   offer_duration      47990 non-null  object
 10  recurrent           47990 non-null  object
 11  bonus_title         47990 non-null  object
 12  promo               47990 non-null  object
 13  ub_type             47990 non-null  object
 14  money               47990 non-null  object
 15  bonus_activated_at  47990 non-null  object
 16  bonus_starts_at     47990 non-null  object
 17  bonus_ends_at       47990 non-null  object
 18  last_prolong_date   47990 non-null  datetime64[ns]
 19  first_prolong_date  47990 non-null  datetime64[ns]
 20  promo_type          47990 non-null  object
 21  reg_source          47990 non-null  object
 22  reg_campaign        47990 non-null  object
 23  reg_medium          47990 non-null  object
 24  payer               47990 non-null  int64
 25  len                 47990 non-null  int64
 26  device              47990 non-null  object
 27  deleted_at          47990 non-null  object
 28  platform            47990 non-null  object
 29  domain_locale       47990 non-null  object
 30  first_payment       47990 non-null  int64
 31  device_os           47990 non-null  object
 32  free_days           47990 non-null  int64
 33  price_currency      47990 non-null  object
 34  subscription_date   47990 non-null  datetime64[ns]
 35  first_payment_date  47990 non-null  datetime64[ns]
dtypes: datetime64[ns](6), int64(5), object(25)
memory usage: 13.2+ MB
```

1

1.1

[10]:
```
df['payment_type'] = df['first_payment_date'].apply(lambda x:0 if x==pd.
 ↪to_datetime('1970-01-01') else 1)
df['subscription_month'] = df['subscription_date'].dt.to_period('M')
df['subscription_month'] = df['subscription_month'].dt.to_timestamp()
```

[11]:
```
df_agg = df.groupby('subscription_month',as_index=False).agg({'profile_id':
 ↪'count','payment_type':'sum'})
df_agg.rename(columns={'profile_id':'trial_cnt','payment_type':
 ↪'paid_cnt'},inplace=True)
df_agg['cr_to_payment'] = df_agg['paid_cnt']/df_agg['trial_cnt']
```

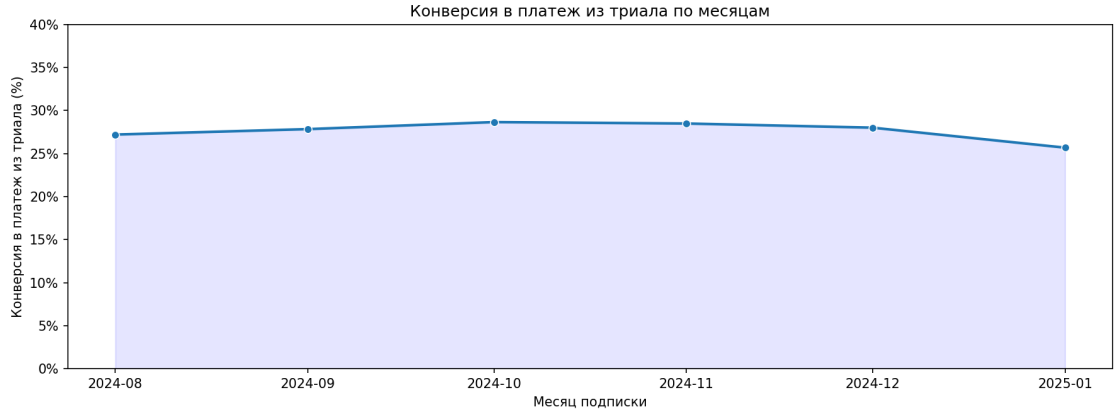
[12]:
```
df_agg
```

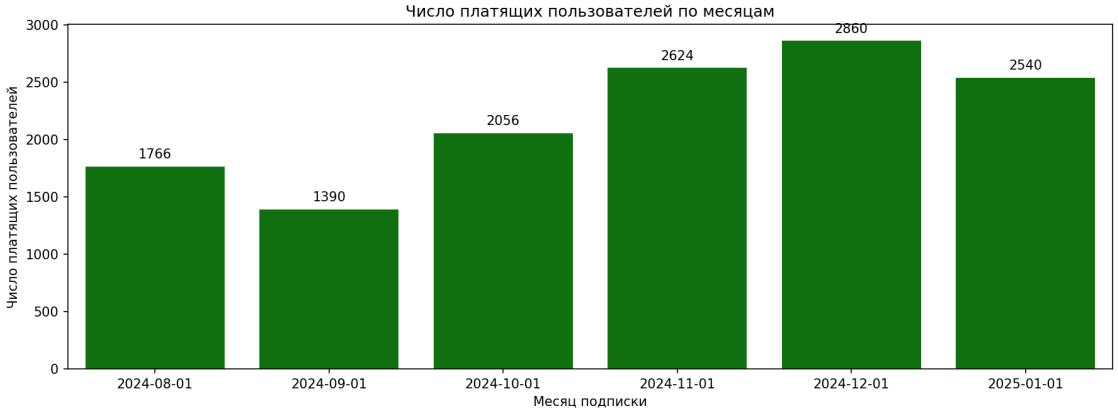
[12]:

	subscription_month	trial_cnt	paid_cnt	cr_to_payment
0	2024-08-01	6494	1766	0.271943
1	2024-09-01	4995	1390	0.278278
2	2024-10-01	7178	2056	0.286431
3	2024-11-01	9214	2624	0.284784
4	2024-12-01	10216	2860	0.279953
5	2025-01-01	9893	2540	0.256747

[13]:
```
import matplotlib.ticker as ticker

#      figure   axes (                )
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 9), dpi=150)

# ---          (cr_to_payment) ---
sns.lineplot(data=df_agg, x='subscription_month', y='cr_to_payment',␣
 ↪marker='o', palette='magma', linestyle='-', linewidth=2, ax=ax1)
ax1.fill_between(df_agg['subscription_month'], df_agg['cr_to_payment'],␣
 ↪color='blue', alpha=0.1)
ax1.yaxis.set_major_formatter(ticker.FuncFormatter(lambda y, _: f'{int(y *␣
 ↪100)}%'))
ax1.set_ylim(bottom=0, top=0.4)
ax1.set_xlabel("        ")
ax1.set_ylabel("                  (%)")
ax1.set_title("                       ")

# ---          (paid_cnt) ---
barplot = sns.barplot(data=df_agg, x='subscription_month', y='paid_cnt',␣
 ↪color='green',ax=ax2)
ax2.set_xlabel("        ")
ax2.set_ylabel("                ")
```

```
ax2.set_title("                          ")
ax2.ticklabel_format(style='plain', axis='y') #

for p in barplot.patches:
    barplot.annotate(format(p.get_height(), '.0f'),  #
                     (p.get_x() + p.get_width() / 2., p.get_height()),
                     ha = 'center', va = 'center',
                     xytext = (0, 9),   #
                     textcoords = 'offset points')

plt.tight_layout()
plt.show()
```





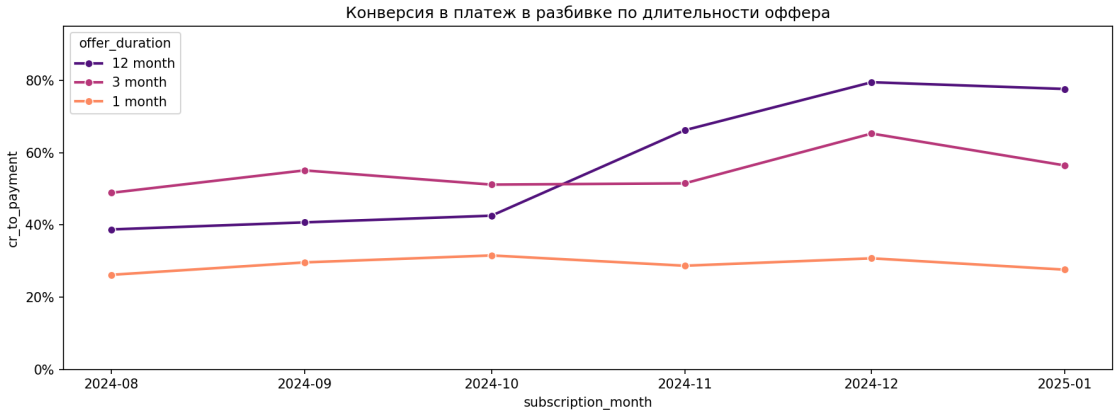
[14]:
```
print('           CR   6     :',round(df_agg['cr_to_payment'].median(),4))
print('                          6     :',int(df_agg['paid_cnt'].median()))
```

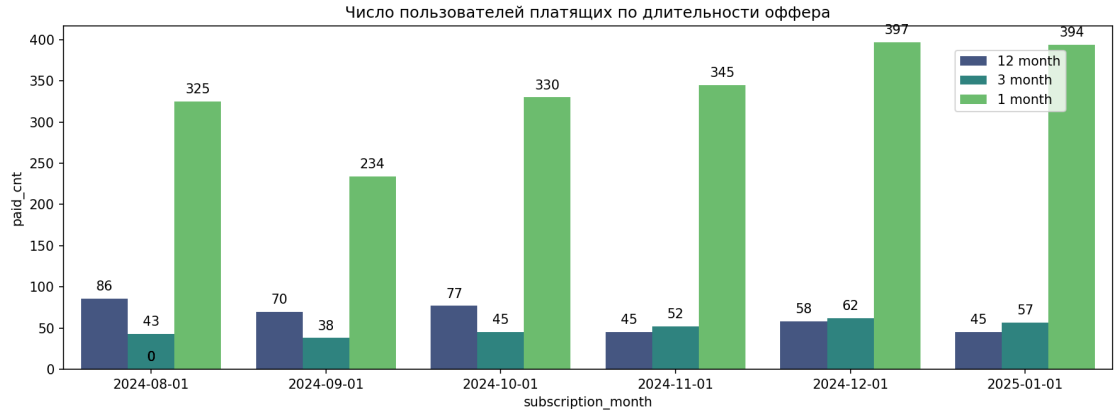
```
          CR   6     : 0.2791
                         6     : 2298
```

1.2 3 14

[21]:
```
df_organic = df[(df['offer_duration'].isin(['1 month','3 month','12 month'])) &␣
 ↪(df['free_days'].isin([3,14]))]

df_organic = df_organic[~((df_organic['offer_duration'] == '1 month') &␣
 ↪(df_organic['free_days'] == 3))]
df_organic = df_organic[~((df_organic['offer_duration'] == '3 month') &␣
 ↪(df_organic['free_days'] == 14))]
df_organic = df_organic[~((df_organic['offer_duration'] == '12 month') &␣
 ↪(df_organic['free_days'] == 14))]



df_organic_agg = df_organic.
 ↪groupby(['subscription_month','free_days','offer_duration'],as_index=False).
 ↪agg({'profile_id':'count','payment_type':'sum'})
df_organic_agg.rename(columns={'profile_id':'trial_cnt','payment_type':
 ↪'paid_cnt'},inplace=True)
df_organic_agg['cr_to_payment'] = df_organic_agg['paid_cnt']/
 ↪df_organic_agg['trial_cnt']
```

[23]:
```
#      figure   axes (                )
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 9), dpi=150)

sns.lineplot(data=df_organic_agg, x='subscription_month', y='cr_to_payment',␣
 ↪marker='o', palette='magma',linestyle='-', linewidth=2,␣
 ↪hue='offer_duration', ax=ax1)
ax1.yaxis.set_major_formatter(ticker.FuncFormatter(lambda y, _: f'{int(y *␣
 ↪100)}%'))
ax1.set_ylim(bottom=0,top=0.95)
ax1.set_title('                                  ')

barplot = sns.barplot(data=df_organic_agg, x='subscription_month',␣
 ↪y='paid_cnt', palette='viridis',hue='offer_duration', ax=ax2)
ax2.legend(loc=(0.85,0.75))
ax2.set_title('                                ')

for p in barplot.patches:
    barplot.annotate(format(p.get_height(), '.0f'),  #
                     (p.get_x() + p.get_width() / 2., p.get_height()),
                     ha = 'center', va = 'center',
                     xytext = (0, 9),   #
                     textcoords = 'offset points')


plt.tight_layout()
plt.show()
```





[25]:
```
print('           CR     14      6     :
 ↪',round(df_organic_agg[df_organic_agg['offer_duration']=='1␣
 ↪month']['cr_to_payment'].median(),4))
print('           CR     3     3       6     :
 ↪',round(df_organic_agg[df_organic_agg['offer_duration']=='3␣
 ↪month']['cr_to_payment'].median(),4))
print('           CR     3     12        6     :
 ↪',round(df_organic_agg[df_organic_agg['offer_duration']=='12␣
 ↪month']['cr_to_payment'].median(),4))
```

```
           CR     14      6     : 0.2916
           CR     3     3       6     : 0.5328
           CR     3     12        6     : 0.5436
```

[27]:
```
print('           CR     14      6     :
 ↪',int(df_organic_agg[df_organic_agg['offer_duration']=='1␣
 ↪month']['paid_cnt'].median()))
print('           CR     3         3       6     :
 ↪',int(df_organic_agg[df_organic_agg['offer_duration']=='3␣
 ↪month']['paid_cnt'].median()))
```

```
print('           CR     3          12        6     :
 ↪',int(df_organic_agg[df_organic_agg['offer_duration']=='12␣
 ↪month']['paid_cnt'].median()))
```

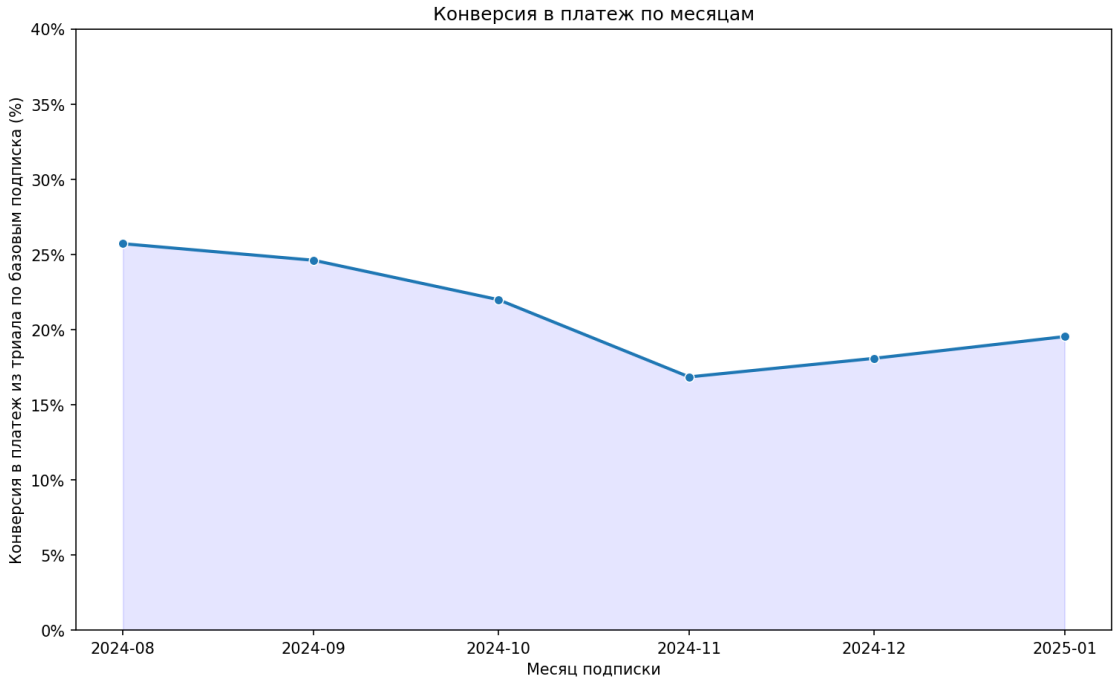
```
          CR     14       6     : 337
          CR     3          3        6     : 48
          CR     3          12         6     : 64
```

1.3

[30]:
```
df_organic_agg2 = df_organic_agg.
 ↪groupby('subscription_month',as_index=False)['paid_cnt'].sum()

df_merge = pd.
 ↪merge(df_agg,df_organic_agg2,on='subscription_month',how='left',suffixes=('_all','_organic'
df_merge['cnt_paid_organic_frac'] = df_merge['paid_cnt_organic']/
 ↪df_merge['paid_cnt_all']
```

[32]:
```
#      figure   axes (                )
fig, ax1 = plt.subplots(1, 1, figsize=(12, 7), dpi=150)

sns.lineplot(data=df_merge, x='subscription_month', y='cnt_paid_organic_frac',␣
 ↪marker='o', palette='magma',linestyle='-', linewidth=2, ax=ax1)
ax1.fill_between(df_merge['subscription_month'],␣
 ↪df_merge['cnt_paid_organic_frac'], color='blue', alpha=0.1)
ax1.yaxis.set_major_formatter(ticker.FuncFormatter(lambda y, _: f'{int(y *␣
 ↪100)}%'))
ax1.set_ylim(bottom=0, top=0.4)
ax1.set_xlabel("        ")
ax1.set_ylabel("                                (%)")
ax1.set_title("                  ")
```

[32]:
```
Text(0.5, 1.0, '                 ')
```



[34]:
```
print('                                    6     :
 ↪',round(df_merge['cnt_paid_organic_frac'].median(),4))
```

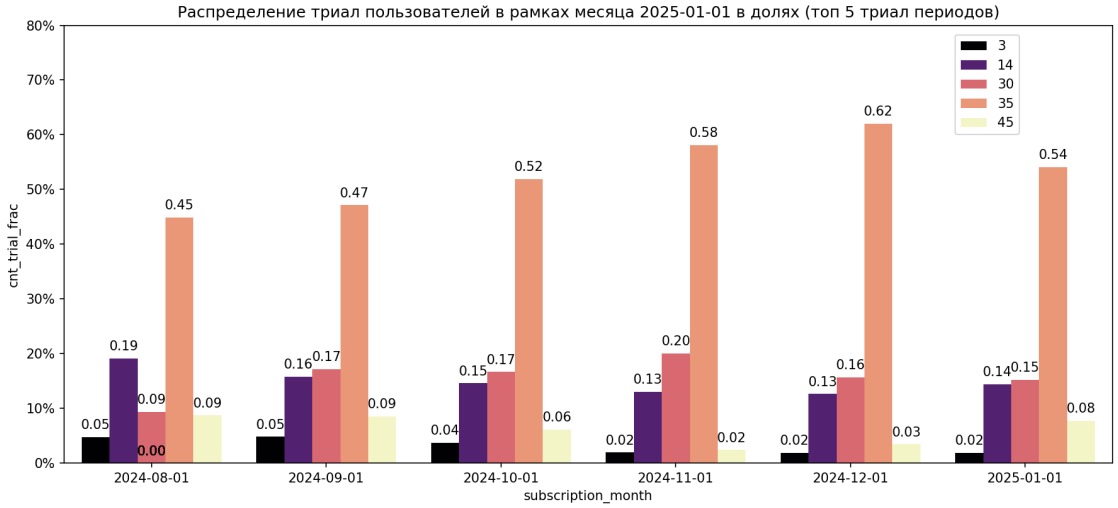
```
                                    6     : 0.2076
```

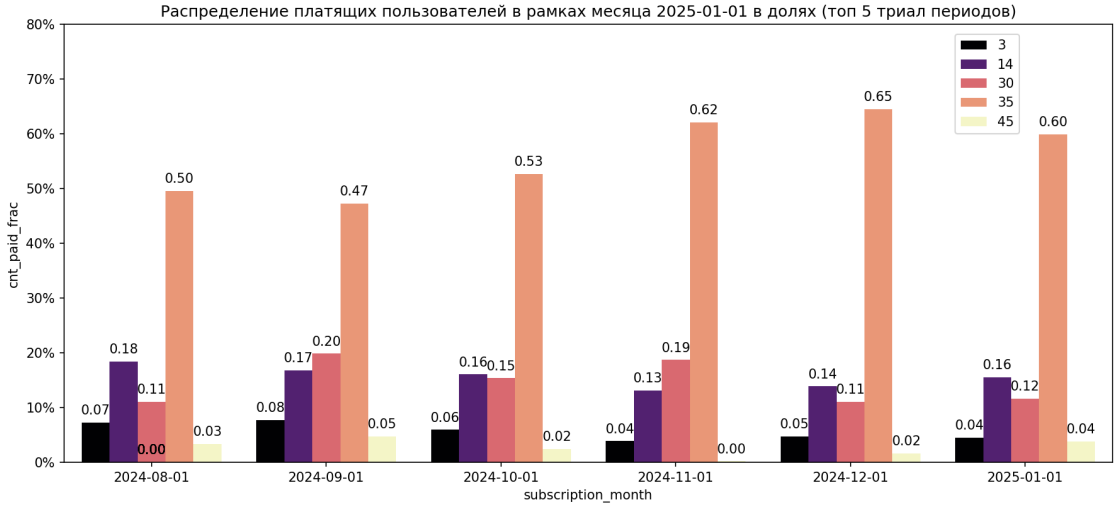
1.4

[37]:
```
df_check = df.groupby(['subscription_month','free_days']).agg({'profile_id':
 ↪'count','payment_type':'sum'}).reset_index()
df_check.rename(columns={'payment_type':'cnt_paid','profile_id':
 ↪'cnt_trial'},inplace=True)
df_check['cnt_paid_all_by_day'] = df_check.groupby('subscription_month').
 ↪transform('sum')['cnt_paid']
df_check['cnt_trial_all_by_day'] = df_check.groupby('subscription_month').
 ↪transform('sum')['cnt_trial']


df_check['cnt_paid_frac'] = df_check['cnt_paid']/df_check['cnt_paid_all_by_day']
df_check['cnt_trial_frac'] = df_check['cnt_trial']/
 ↪df_check['cnt_trial_all_by_day']
df_check = df_check[df_check['free_days'].isin([3,14,30,35,45])]
```

[39]:
```
#      figure   axes (                )
fig, (ax1,ax2) = plt.subplots(2, 1, figsize=(12, 11), dpi=150)
```

```
barplot = sns.barplot(data=df_check, x='subscription_month',␣
 ↪y='cnt_trial_frac', hue='free_days', palette='magma', ax=ax1)
ax1.yaxis.set_major_formatter(ticker.FuncFormatter(lambda y, _: f'{int(y *␣
 ↪100)}%'))
ax1.set_ylim(bottom=0,top=0.8)
ax1.set_title('                                2025-01-01      (   5         )')
ax1.legend(title='',loc=(0.85,0.75))

for p in barplot.patches:
    ax1.annotate(format(p.get_height(), '.2f'),  #       :      '.2f'
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha = 'center', va = 'center',
                xytext = (0, 9),
                textcoords = 'offset points')

barplot2 = sns.barplot(data=df_check, x='subscription_month',␣
 ↪y='cnt_paid_frac', hue='free_days', palette='magma', ax=ax2)
ax2.yaxis.set_major_formatter(ticker.FuncFormatter(lambda y, _: f'{int(y *␣
 ↪100)}%'))
ax2.set_ylim(bottom=0,top=0.8)
ax2.set_title('                                  2025-01-01      (   5         )')
ax2.legend(title='',loc=(0.85,0.75))

for p in barplot2.patches:
    ax2.annotate(format(p.get_height(), '.2f'),  #       :      '.2f'
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha = 'center', va = 'center',
                xytext = (0, 9),
                textcoords = 'offset points')

plt.tight_layout()
plt.show()
```





[41]:
```
#
df_check.groupby('free_days').median().reset_index()
```

[41]:

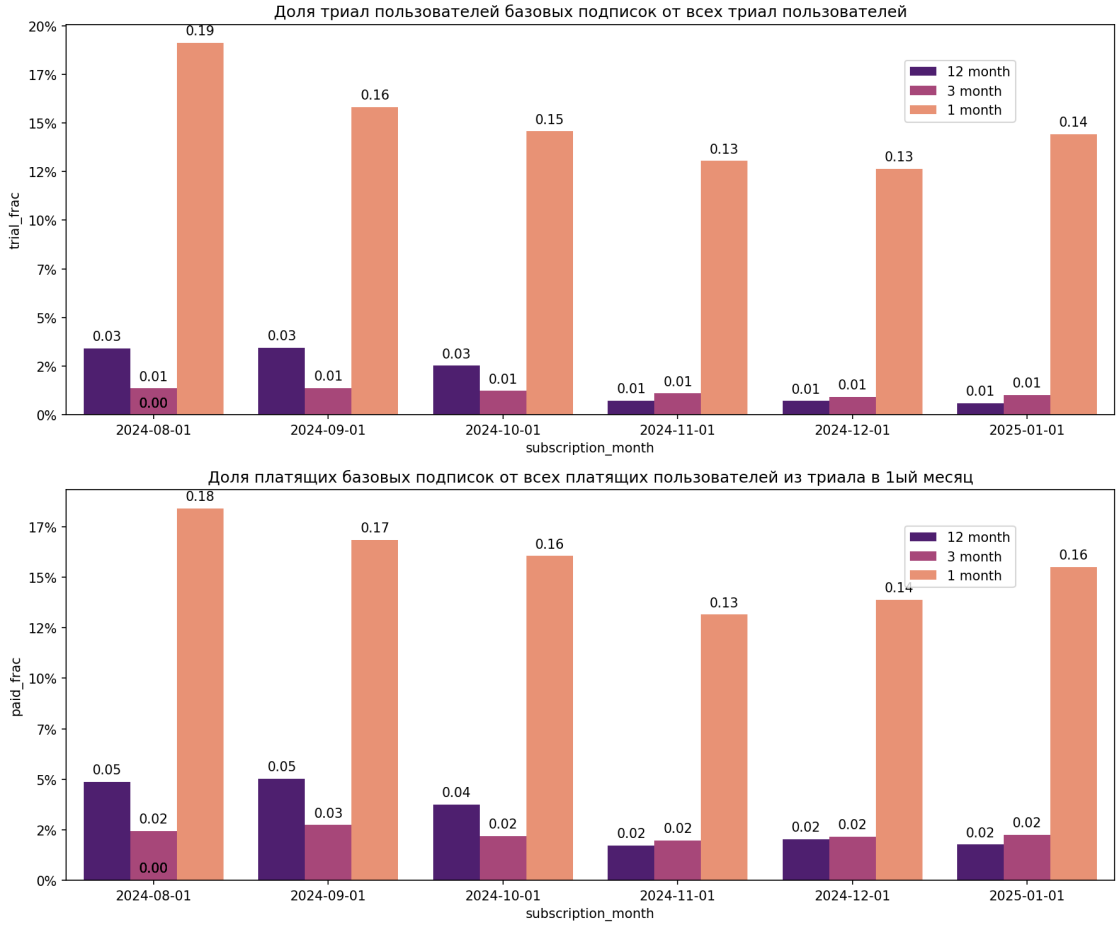
	free_days	subscription_month	cnt_trial	cnt_paid	cnt_paid_all_by_day	cnt_trial_all_by_day	cnt_paid_frac	cnt_trial_frac
0	3	2024-10-16 12:00:00	218.0	118.0	2298.0	8196.0	0.053271	0.028506
1	14	2024-10-16 12:00:00	1221.5	337.5	2298.0	8196.0	0.157812	0.145173
2	30	2024-10-16 12:00:00	1350.0	306.0	2298.0	8196.0	0.135116	0.161717
3	35	2024-10-16 12:00:00	4539.0	1302.5	2298.0	8196.0	0.562982	0.530189
4	45	2024-10-16 12:00:00	433.0	55.5	2298.0			

```
4                8196.0        0.029390         0.069414
```

[43]:
```
df_check_organic = df_organic.
 ↪groupby(['subscription_month','free_days','offer_duration']).
 ↪agg({'profile_id':'count','payment_type':'sum'}).reset_index()
df_check_organic.rename(columns={'payment_type':'cnt_paid','profile_id':
 ↪'cnt_trial'},inplace=True)

df_check_organic_frac = pd.merge(df_check_organic,df_check.
 ↪groupby('subscription_month')[['cnt_trial_all_by_day','cnt_paid_all_by_day']].
 ↪mean().reset_index(),on='subscription_month',how='left')

df_check_organic_frac['trial_frac'] = df_check_organic_frac['cnt_trial'] /␣
 ↪df_check_organic_frac['cnt_trial_all_by_day']
df_check_organic_frac['paid_frac'] = df_check_organic_frac['cnt_paid'] /␣
 ↪df_check_organic_frac['cnt_paid_all_by_day']
```

[45]:
```
#      figure   axes (                )
fig, (ax1,ax2) = plt.subplots(2, 1, figsize=(12, 10), dpi=150)

barplot = sns.barplot(data=df_check_organic_frac, x='subscription_month',␣
 ↪y='trial_frac', hue='offer_duration', palette='magma', ax=ax1)
ax1.yaxis.set_major_formatter(ticker.FuncFormatter(lambda y, _: f'{int(y *␣
 ↪100)}%'))
ax1.set_title('                                          ')
ax1.legend(title='',loc=(0.8,0.75))

for p in barplot.patches:
    ax1.annotate(format(p.get_height(), '.2f'),  #      :      '.2f'
                 (p.get_x() + p.get_width() / 2., p.get_height()),
                 ha = 'center', va = 'center',
                 xytext = (0, 9),
                 textcoords = 'offset points')

barplot2 = sns.barplot(data=df_check_organic_frac, x='subscription_month',␣
 ↪y='paid_frac', hue='offer_duration', palette='magma', ax=ax2)
ax2.yaxis.set_major_formatter(ticker.FuncFormatter(lambda y, _: f'{int(y *␣
 ↪100)}%'))
ax2.set_title('                                            1     ')
ax2.legend(title='',loc=(0.8,0.75))

for p in barplot2.patches:
    ax2.annotate(format(p.get_height(), '.2f'),  #      :      '.2f'
                 (p.get_x() + p.get_width() / 2., p.get_height()),
                 ha = 'center', va = 'center',
                 xytext = (0, 9),
```

```
                textcoords = 'offset points')

plt.tight_layout()
plt.show()
```



1.5 2-

[50]:
```
query = '''WITH payment AS (SELECT
                paid_date,
                subscription_id,
                user_id,
                platform,
                free_days,
                ROW_NUMBER() OVER (PARTITION BY user_id ORDER BY paid_date) AS␣
 ↪row_num
                FROM datamarts.finance f
                WHERE paid_date BETWEEN '2024-08-01' AND '2025-03-16'
                AND platform IN ('cloudpayments','payture')
```

```
                ),


other_info AS (SELECT * FROM datamarts.marketing_dash
                WHERE domain_locale='ru'
                AND first_prolong_date>='2024-07-01'
                AND platform IN ('cloudpayments','payture')
            )


SELECT t1.*,offer_duration,created_at::date AS subscription_date,␣
 ↪first_prolong_date
FROM payment AS t1
INNER JOIN other_info AS t2  ON t1.user_id=t2.user_id
WHERE row_num<=2 AND first_prolong_date<='2025-01-31'
'''

df_repayment = execute(query,user='kmekhtiev')

df_repayment['paid_date'] = pd.to_datetime(df_repayment['paid_date'])
df_repayment['first_prolong_date'] = pd.
 ↪to_datetime(df_repayment['first_prolong_date'])
```

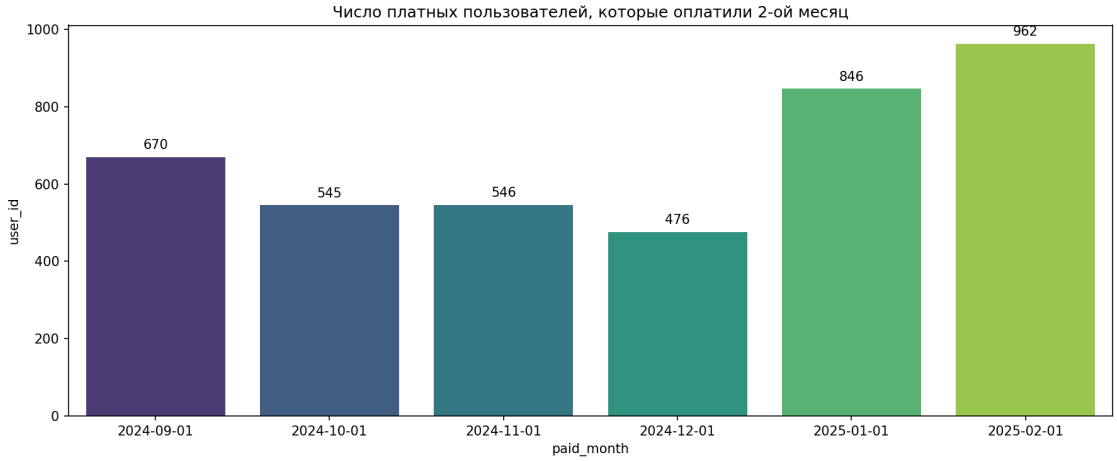
: 2.6822

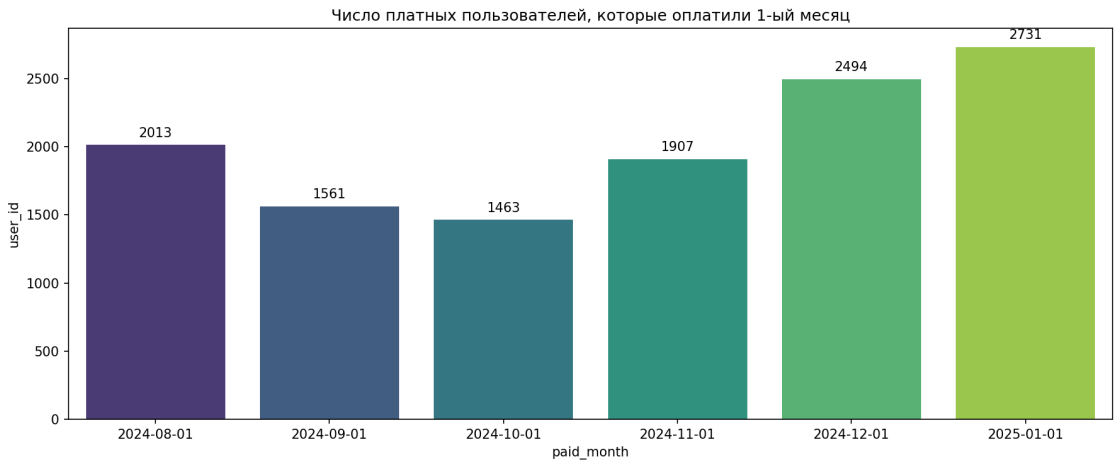
[52]:
```
df_repayment['paid_month'] = df_repayment['paid_date'].dt.to_period('M')
df_repayment['paid_month'] = df_repayment['paid_month'].dt.to_timestamp()
df_repayment_14_trial = df_repayment[(df_repayment['free_days']=='14') &␣
 ↪(df_repayment['offer_duration']=='1 month')]

df_repayment_second_payment = df_repayment[df_repayment['row_num']==2].
 ↪groupby('paid_month')['user_id'].nunique().reset_index()
df_repayment_first_payment = df_repayment[df_repayment['row_num']==1].
 ↪groupby('paid_month')['user_id'].nunique().reset_index()
```

1.6

[55]:
```
fig, (ax1,ax2) = plt.subplots(2, 1, figsize=(12, 10), dpi=150)

df_repayment_second_payment =␣
 ↪df_repayment_second_payment[(df_repayment_second_payment['paid_month']>='2024-09-01')␣
 ↪& (df_repayment_second_payment['paid_month']<'2025-03-01')]
barplot = sns.barplot(data=df_repayment_second_payment, x='paid_month',␣
 ↪y='user_id', palette='viridis', ax=ax1)
ax1.set_title('                ,           2-     ')
for p in barplot.patches:
```

```
    barplot.annotate(format(p.get_height(), '.0f'),  #
                     (p.get_x() + p.get_width() / 2., p.get_height()),
                     ha = 'center', va = 'center',
                     xytext = (0, 9),   #
                     textcoords = 'offset points')

df_repayment_first_payment =␣
 ↪df_repayment_first_payment[(df_repayment_first_payment['paid_month']<'2025-02-01')]
barplot2 = sns.barplot(data=df_repayment_first_payment, x='paid_month',␣
 ↪y='user_id', palette='viridis', ax=ax2)
ax2.set_title('               ,           1-     ')
for p in barplot2.patches:
    barplot2.annotate(format(p.get_height(), '.0f'),  #
                     (p.get_x() + p.get_width() / 2., p.get_height()),
                     ha = 'center', va = 'center',
                     xytext = (0, 9),   #
                     textcoords = 'offset points')


plt.tight_layout()
plt.show()
```





[57]:
```
print('            2-            6    :',\
      df_repayment[df_repayment['row_num']==2]['user_id'].nunique() / \
      df_repayment[df_repayment['row_num']==1]['user_id'].nunique())

print('                ,       2-            6    :',\
     int(df_repayment_second_payment['user_id'].median()))
```

```
          2-            6    : 0.3642863012572931
              ,       2-            6    : 608
```

1.7 14

[60]:
```
df_repayment_14_trial = df_repayment[(df_repayment['free_days']=='14') &␣
 ↪(df_repayment['offer_duration']=='1 month')]

df_repayment_second_payment_14_trial =␣
 ↪df_repayment_14_trial[df_repayment_14_trial['row_num']==2].
 ↪groupby('paid_month')['user_id'].nunique().reset_index()
```

```
df_repayment_first_payment_14_trial =␣
 ↪df_repayment_14_trial[df_repayment_14_trial['row_num']==1].
 ↪groupby('paid_month')['user_id'].nunique().reset_index()

fig, (ax1,ax2) = plt.subplots(2, 1, figsize=(12, 10), dpi=100)

df_repayment_second_payment_14_trial =␣
 ↪df_repayment_second_payment_14_trial[(df_repayment_second_payment_14_trial['paid_month']>='
 ↪& (df_repayment_second_payment_14_trial['paid_month']<'2025-03-01')]
barplot = sns.barplot(data=df_repayment_second_payment_14_trial,␣
 ↪x='paid_month', y='user_id', palette='viridis', ax=ax1)
ax1.set_title('                ,           2-     ')
for p in barplot.patches:
    barplot.annotate(format(p.get_height(), '.0f'),  #
                     (p.get_x() + p.get_width() / 2., p.get_height()),
                     ha = 'center', va = 'center',
                     xytext = (0, 9),   #
                     textcoords = 'offset points')

df_repayment_first_payment_14_trial =␣
 ↪df_repayment_first_payment_14_trial[(df_repayment_first_payment_14_trial['paid_month']<'202
barplot2 = sns.barplot(data=df_repayment_first_payment_14_trial,␣
 ↪x='paid_month', y='user_id', palette='viridis', ax=ax2)
ax2.set_title('                ,           1-     ')
for p in barplot2.patches:
    barplot2.annotate(format(p.get_height(), '.0f'),  #
                     (p.get_x() + p.get_width() / 2., p.get_height()),
                     ha = 'center', va = 'center',
                     xytext = (0, 9),   #
                     textcoords = 'offset points')


plt.tight_layout()
plt.show()
```

[62]:
```
print('            2-            6         14   :',\
      df_repayment_14_trial[df_repayment_14_trial['row_num']==2]['user_id'].
 ↪nunique() / \
      df_repayment_14_trial[df_repayment_14_trial['row_num']==1]['user_id'].
 ↪nunique())

print('                  2-            6         14   :',\
      int(df_repayment_second_payment_14_trial['user_id'].median()))
```

```
            2-            6         14   :
0.5726618705035971
                  2-            6         14   :
197
```

[64]:
```
print('               14   ,            2-          ,          2-     :',\
      df_repayment_second_payment_14_trial['user_id'].sum() /␣
 ↪df_repayment_second_payment['user_id'].sum())
```

```
               14   ,            2-          ,
```

2- : 0.29443757725587144

1.8 LTV

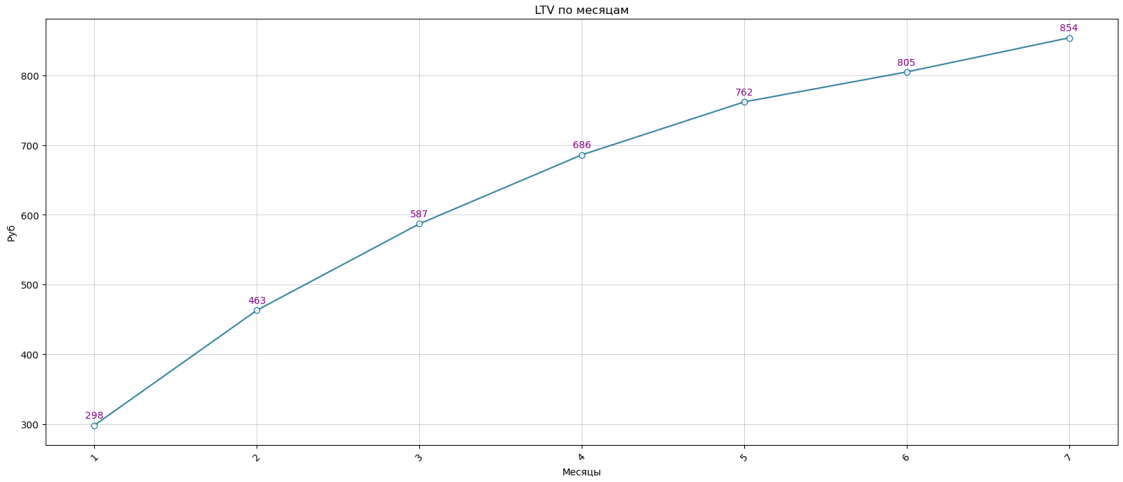
[67]:
```
list_date = pd.date_range("2024-08-01", "2025-02-01", freq='MS',normalize=True)
result = []
for date in list_date:
    date_str = date.strftime('%Y-%m-%d')
    query = f'''
    WITH payment AS (SELECT
                '{date_str}' AS trial_month,
                paid_date,
                subscription_id,
                user_id,
                platform,
                free_days,
                payment,
                ROW_NUMBER() OVER (PARTITION BY user_id ORDER BY paid_date) AS␣
 ↪row_num
                FROM datamarts.finance f
                WHERE platform IN ('cloudpayments','payture')
                AND paid_date BETWEEN '{date_str}' AND '2025-02-28'
                ),


    first_payment AS (SELECT * FROM datamarts.marketing_dash
                  WHERE domain_locale='ru'
                  AND DATE_TRUNC('month', first_prolong_date)='{date_str}'
                  AND platform IN ('cloudpayments','payture')
            )

    SELECT
    t1.*,
    offer_duration,
    created_at::date AS subscription_date,
    first_prolong_date
    FROM payment AS t1
    INNER JOIN first_payment AS t2  ON t1.user_id=t2.user_id

            '''
    df_temp = execute(query,user = 'kmekhtiev')
    result.append(df_temp)
    print(f"""    '{date_str}'      """)
    print()

df_ltv = pd.concat(result)
df_ltv['paid_date'] = df_ltv['paid_date'].astype('datetime64[ns]')
```

```
df_ltv['trial_month'] = df_ltv['trial_month'].astype('datetime64[ns]')
df_ltv['payment'] = df_ltv['payment'].astype('int32')
```

```
            : 1.3229

  '2024-08-01'

            : 1.2979

  '2024-09-01'

            : 1.3489

  '2024-10-01'

            : 1.3084

  '2024-11-01'

            : 1.3422

  '2024-12-01'

            : 1.301

  '2025-01-01'

            : 1.2459

  '2025-02-01'
```

[69]:
```
df_ltv['paid_month'] = df_ltv['paid_date'].dt.to_period('M')
df_ltv = df_ltv[(df_ltv['free_days']=='14') & (df_ltv['offer_duration']=='1␣
 ↪month')]
df_agg = df_ltv.groupby(['trial_month','paid_month']).agg({'payment':
 ↪'sum','user_id':'count'}).reset_index()

#                ,
df_agg['cumulative'] = df_agg.groupby('trial_month')['payment'].cumsum()
df_agg['uniq_user'] = df_agg.groupby('trial_month')['user_id'].transform('max')

df_agg['ltv'] = df_agg.cumulative/df_agg.uniq_user

#
df_agg['num_of_month'] = df_agg.groupby('trial_month')['paid_month'].
 ↪rank(method='first').astype('int')
```

```
df_agg = df_agg[df_agg['num_of_month']<10] #         6-


#      pivot
df_pivot = pd.pivot(data = df_agg,index = 'trial_month',columns =␣
 ↪'num_of_month',values = 'ltv')
df_pivot['uniq_user'] = df_agg.groupby('trial_month')['uniq_user'].max()


#      DataFrame                          LTV
weighted_ltv = df_agg.pivot(index='trial_month', columns='num_of_month',␣
 ↪values='ltv')
user_counts = df_agg.pivot(index='trial_month', columns='num_of_month',␣
 ↪values='uniq_user')


#                       LTV         'num_of_month'
weighted_avg_ltv = ((weighted_ltv * user_counts).sum() / user_counts.sum()).
 ↪round().astype('int')


weighted_avg_ltv = weighted_avg_ltv.reset_index()
weighted_avg_ltv = weighted_avg_ltv.rename(columns={0:'cumsum'})

#                   LTV
plt.figure(figsize=(20,8))
plt.plot(weighted_avg_ltv['num_of_month'], weighted_avg_ltv['cumsum'],␣
 ↪marker='o', linestyle='-', color='#005f80',markerfacecolor='white',alpha=0.8)

for i in range(len(weighted_avg_ltv['num_of_month'])):
    plt.text(weighted_avg_ltv['num_of_month'][i],␣
 ↪weighted_avg_ltv['cumsum'][i]+10,
             str(weighted_avg_ltv['cumsum'][i]),
             ha='center', fontsize=10,color='purple')
plt.xlabel('    ')
plt.ylabel('  ')
plt.title('LTV       ')
plt.grid(True,linewidth=0.4)
plt.xticks(rotation=45)
plt.show()
```



2

2.1

[72]:
```
query = '''WITH all_info AS (SELECT
        s.user_id AS user_id,
        s.created_at as subscription_created_at,
        s.id AS id,
        leadInFrame(id) OVER (PARTITION BY user_id ORDER BY invoice_created_at ␣
↪ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS next_id,
        i.state as invoice_state,
        leadInFrame(invoice_state) OVER (PARTITION BY user_id ORDER BY␣
↪invoice_created_at  ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED␣
↪FOLLOWING) AS next_invoice_state,
        i.id as invoice_id,
        i.created_at as invoice_created_at,
        leadInFrame(invoice_created_at) OVER (PARTITION BY user_id ORDER BY␣
↪invoice_created_at  ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED␣
↪FOLLOWING) AS next_invoice_created_at,
        next_invoice_created_at::date - invoice_created_at::date delta_date,
        i.price_cents AS price_cents,
        i.price_currency AS price_currency,
        i.refund_amount_cents AS refund_amount_cents,
        ROW_NUMBER() OVER (PARTITION BY user_id,id,id,user_type,invoice_state␣
↪ORDER BY invoice_created_at) AS rn_num,
        CASE WHEN price_cents<=100 AND invoice_state='success' THEN 'trial'
             WHEN price_cents<=100 AND invoice_state IN ('failure','initial')␣
↪THEN 'not_success_trial'
             ELSE 'subs'
             END user_type,
```

```
       leadInFrame(user_type) OVER (PARTITION BY user_id ORDER BY␣
 ↪invoice_created_at  ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED␣
 ↪FOLLOWING) AS next_user_type,
       t2.renewal_off_date AS renewal_off_date,
       CASE WHEN renewal_off_date!='1970-01-01' THEN 1 ELSE 0 END␣
 ↪unsubscribe_type,
       t3.free_days AS free_days,
       t3.offer_duration AS offer_duration,
       t3.promo_type,
       i.paid_at AS paid_at,
       --i.period_end,
       s.platform AS platform,
      -- i.payment_target,
       i.deleted_at,
       i.refunded_at
   FROM raw.viju__product_x__public__invoices i
   LEFT JOIN raw.viju__product_x__public__subscriptions s
       ON i.subscription_id = s.id
   LEFT JOIN (SELECT
               subscription_id,
               max(created_at) as renewal_off_date
               FROM raw.viju__product_x__public__subscription_cancel_reasons
               WHERE created_at >= '2022-03-01'
                --AND subscription_id='74bcff94-4ada-40ef-85de-25074d615d57'
               GROUP by 1
               ) AS t2 ON i.subscription_id=t2.subscription_id
   INNER JOIN datamarts.marketing_dash AS t3 ON s.user_id=t3.user_id
   WHERE s.platform != 'api'
   ORDER BY s.user_id, invoice_created_at
   )
--
SELECT * FROM all_info
WHERE invoice_created_at::date BETWEEN '2024-08-01' AND '2025-03-16'
--AND user_type!='not_success_trial'
AND platform!='payture'
'''

df_failure = execute(query,user='kmekhtiev')
```

```
            : 29.2811
```

[73]:
```
df_failure['invoice_created_at'] =  df_failure['invoice_created_at'].dt.
 ↪strftime("%Y-%m-%d")
df_failure['invoice_created_at'] =  pd.
 ↪to_datetime(df_failure['invoice_created_at'])
```

```
df_failure['invoice_created_month'] = df_failure['invoice_created_at'].dt.
 ↪to_period('M')
```

[74]:
```
df_failure[df_failure['user_type']=='trial'].
 ↪groupby(['invoice_created_month','invoice_state'],as_index=False)['user_id'].
 ↪nunique()
```

[74]:

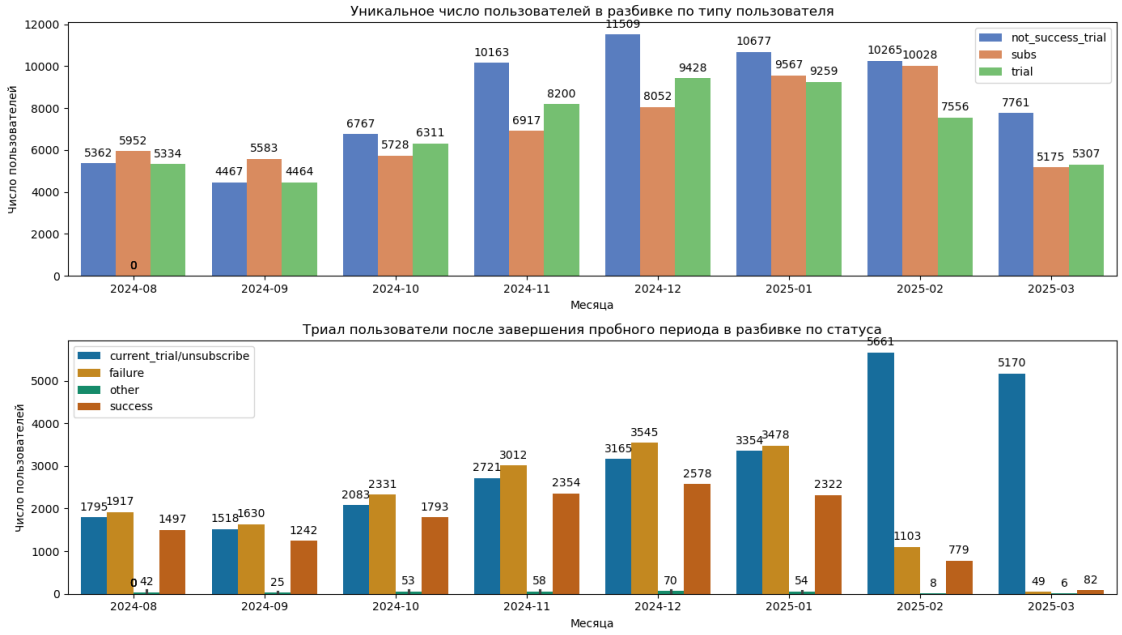
	invoice_created_month	invoice_state	user_id
0	2024-08	success	5334
1	2024-09	success	4464
2	2024-10	success	6311
3	2024-11	success	8200
4	2024-12	success	9428
5	2025-01	success	9259
6	2025-02	success	7556
7	2025-03	success	5307

[75]:
```
df_failure[df_failure['user_type']=='not_success_trial'].
 ↪groupby('price_cents')['user_id'].count()
```

[75]:
```
price_cents
0    68907
Name: user_id, dtype: int64
```

[76]:
```
fig, (ax1,ax2) = plt.subplots(2,1,figsize=(14,8),dpi=100)

t = df_failure.
 ↪groupby(['invoice_created_month','user_type'],as_index=False)['user_id'].
 ↪nunique()
barplot = sns.barplot(data =␣
 ↪t,x='invoice_created_month',y='user_id',hue='user_type',palette =␣
 ↪'muted',ax=ax1)
ax1.set_ylabel('           ')
ax1.set_xlabel('    ')
ax1.set_title('                                        ')
ax1.legend(title='')
for p in barplot.patches:
    barplot.annotate(format(p.get_height(), '.0f'),  #
                     (p.get_x() + p.get_width() / 2., p.get_height()),
                     ha = 'center', va = 'center',
                     xytext = (0, 9),   #
                     textcoords = 'offset points')


r = df_failure[df_failure['user_type']=='trial'].
 ↪groupby(['invoice_created_month','next_invoice_state'],as_index=False)['user_id'].
 ↪nunique()
```

```
r['next_invoice_state'] = r['next_invoice_state'].apply(lambda x:␣
 ↪'current_trial/unsubscribe' if x == ''  else
                                                          'other' if x in␣
 ↪['initial','refunded','processing']
                                                          else x)

barplot2 = sns.barplot(data =␣
 ↪r,x='invoice_created_month',y='user_id',hue='next_invoice_state',palette =␣
 ↪'colorblind',ax=ax2)
ax2.set_ylabel('           ')
ax2.set_xlabel('   ')
ax2.legend(title='')
ax2.set_title('                                                ')
for p in barplot2.patches:
    barplot2.annotate(format(p.get_height(), '.0f'),  #
                      (p.get_x() + p.get_width() / 2., p.get_height()),
                      ha = 'center', va = 'center',
                      xytext = (0, 9),   #
                      textcoords = 'offset points')

plt.tight_layout()
plt.show()
```



[92]:
```
r2 = r[r['invoice_created_month']<'2025-02-01'].
 ↪groupby('next_invoice_state')['user_id'].median().reset_index()
r2['user_id'] = r2['user_id'].astype(int)
```

```
#
r2
```

[92]:

	next_invoice_state	user_id
0	current_trial/unsubscribe	2402
1	failure	2671
2	other	35
3	success	2057

2.2 . !!

[95]:
```
fig, (ax1,ax2,ax3,ax4,ax5) = plt.subplots(5,1,figsize=(17,16),dpi=200)

df_failure_organic = df_failure[(df_failure['offer_duration'].isin(['1␣
 ↪month','3 month','12 month'])) & (df_failure['free_days'].isin([3,14]))]
df_failure_organic = df_failure_organic[~((df_failure_organic['offer_duration']␣
 ↪== '1 month') & (df_failure_organic['free_days'] == 3))]
df_failure_organic = df_failure_organic[~((df_failure_organic['offer_duration']␣
 ↪== '3 month') & (df_failure_organic['free_days'] == 14))]
df_failure_organic = df_failure_organic[~((df_failure_organic['offer_duration']␣
 ↪== '12 month') & (df_failure_organic['free_days'] == 14))]

t = df_failure_organic.
 ↪groupby(['invoice_created_month','user_type'],as_index=False)['user_id'].
 ↪nunique()
barplot = sns.barplot(data =␣
 ↪t,x='invoice_created_month',y='user_id',hue='user_type',palette =␣
 ↪'muted',ax=ax1)
ax1.set_ylabel('           ')
ax1.set_xlabel('    ')
ax1.set_title('                                                    ')
ax1.legend(title='')
for p in barplot.patches:
    barplot.annotate(format(p.get_height(), '.0f'),  #
                     (p.get_x() + p.get_width() / 2., p.get_height()),
                     ha = 'center', va = 'center',
                     xytext = (0, 9),   #
                     textcoords = 'offset points')


e = df_failure_organic[(df_failure_organic['user_type']=='trial') &␣
 ↪(df_failure_organic['invoice_state']=='success')].
 ↪groupby(['invoice_created_month','next_invoice_state'],as_index=False)['user_id'].
 ↪nunique()
e['next_invoice_state'] = e['next_invoice_state'].apply(lambda x:␣
 ↪'current_trial/unsubscribe' if x == ''  else
```

```
                                                                'other' if x in␣
 ↪['initial','refunded','processing']
                                                                else x)
barplot2 = sns.barplot(data =␣
 ↪e,x='invoice_created_month',y='user_id',hue='next_invoice_state',palette =␣
 ↪'colorblind',ax=ax2)
ax2.set_ylabel('           ')
ax2.set_xlabel('    ')
ax2.legend(title='',loc='upper left')
ax2.set_title('           c                                                ')
for p in barplot2.patches:
    barplot2.annotate(format(p.get_height(), '.0f'),  #
                      (p.get_x() + p.get_width() / 2., p.get_height()),
                      ha = 'center', va = 'center',
                      xytext = (0, 9),   #
                      textcoords = 'offset points')


e_14 = df_failure_organic[(df_failure_organic['user_type']=='trial') &␣
 ↪(df_failure_organic['invoice_state']=='success') &␣
 ↪(df_failure_organic['free_days']==14)].
 ↪groupby(['invoice_created_month','next_invoice_state'],as_index=False)['user_id'].
 ↪nunique()
e_14['next_invoice_state'] = e_14['next_invoice_state'].apply(lambda x:␣
 ↪'current_trial/unsubscribe' if x == ''  else
                                                                'other' if x in␣
 ↪['initial','refunded','processing']
                                                                else x)
barplot3 = sns.barplot(data =␣
 ↪e_14,x='invoice_created_month',y='user_id',hue='next_invoice_state',palette␣
 ↪= 'colorblind',ax=ax3)
ax3.set_ylabel('           ')
ax3.set_xlabel('    ')
ax3.legend(title='',loc='upper left')
ax3.set_title('                    14                              ')
for p in barplot3.patches:
    barplot3.annotate(format(p.get_height(), '.0f'),  #
                      (p.get_x() + p.get_width() / 2., p.get_height()),
                      ha = 'center', va = 'center',
                      xytext = (0, 9),   #
                      textcoords = 'offset points')
```

```
e_12 = df_failure_organic[(df_failure_organic['user_type']=='trial') &␣
 ↪(df_failure_organic['invoice_state']=='success') &␣
 ↪(df_failure_organic['free_days']==3) &␣
 ↪(df_failure_organic['offer_duration']=='12 month')].
 ↪groupby(['invoice_created_month','next_invoice_state'],as_index=False)['user_id'].
 ↪nunique()
e_12['next_invoice_state'] = e_12['next_invoice_state'].apply(lambda x:␣
 ↪'current_trial/unsubscribe' if x == ''  else
                                                        'other' if x in␣
 ↪['initial','refunded','processing']
                                                        else x)
barplot4 = sns.barplot(data =␣
 ↪e_12,x='invoice_created_month',y='user_id',hue='next_invoice_state',palette␣
 ↪= 'colorblind',ax=ax4)
ax4.set_ylabel('           ')
ax4.set_xlabel('    ')
ax4.legend(title='',loc='upper left')
ax4.set_title('                    3                        12              ␣
 ↪    ')
for p in barplot4.patches:
    barplot4.annotate(format(p.get_height(), '.0f'),  #
                      (p.get_x() + p.get_width() / 2., p.get_height()),
                      ha = 'center', va = 'center',
                      xytext = (0, 9),   #
                      textcoords = 'offset points')


e_3 = df_failure_organic[(df_failure_organic['user_type']=='trial') &␣
 ↪(df_failure_organic['invoice_state']=='success') &␣
 ↪(df_failure_organic['free_days']==3) &␣
 ↪(df_failure_organic['offer_duration']=='3 month')].
 ↪groupby(['invoice_created_month','next_invoice_state'],as_index=False)['user_id'].
 ↪nunique()
e_3['next_invoice_state'] = e_3['next_invoice_state'].apply(lambda x:␣
 ↪'current_trial/unsubscribe' if x == ''  else
                                                        'other' if x in␣
 ↪['initial','refunded','processing']
                                                        else x)
barplot5 = sns.barplot(data =␣
 ↪e_3,x='invoice_created_month',y='user_id',hue='next_invoice_state',palette =␣
 ↪'colorblind',ax=ax5)
ax5.set_ylabel('           ')
ax5.set_xlabel('    ')
ax5.legend(title='',loc='upper left')
ax5.set_title('                    3                        3              ␣
 ↪    ')
```

```
for p in barplot5.patches:
    barplot5.annotate(format(p.get_height(), '.0f'),  #
                      (p.get_x() + p.get_width() / 2., p.get_height()),
                      ha = 'center', va = 'center',
                      xytext = (0, 9),   #
                      textcoords = 'offset points')

plt.tight_layout()
plt.show()
```

[97]:
```
e_142 = e_14[e_14['invoice_created_month']<'2025-02-01'].
  ↪groupby('next_invoice_state')['user_id'].median().reset_index()
e_142['user_id'] = e_142['user_id'].astype(int)
```

```
#                              14
e_142
```

[97]:

	next_invoice_state	user_id
0	current_trial/unsubscribe	466
1	failure	417
2	other	7
3	success	330

[99]:
```
e_122 = e_12[e_12['invoice_created_month']<'2025-02-01'].
 ↪groupby('next_invoice_state')['user_id'].median().reset_index()
e_122['user_id'] = e_122['user_id'].astype(int)

#                              3         12
e_122
```

[99]:

	next_invoice_state	user_id
0	current_trial/unsubscribe	24
1	failure	30
2	other	2
3	success	63

[101]:
```
e_32 = e_3 [e_3 ['invoice_created_month']<'2025-02-01'].
 ↪groupby('next_invoice_state')['user_id'].median().reset_index()
e_32['user_id'] = e_3 ['user_id'].astype(int)

#                              3         12
e_32
```

[101]:

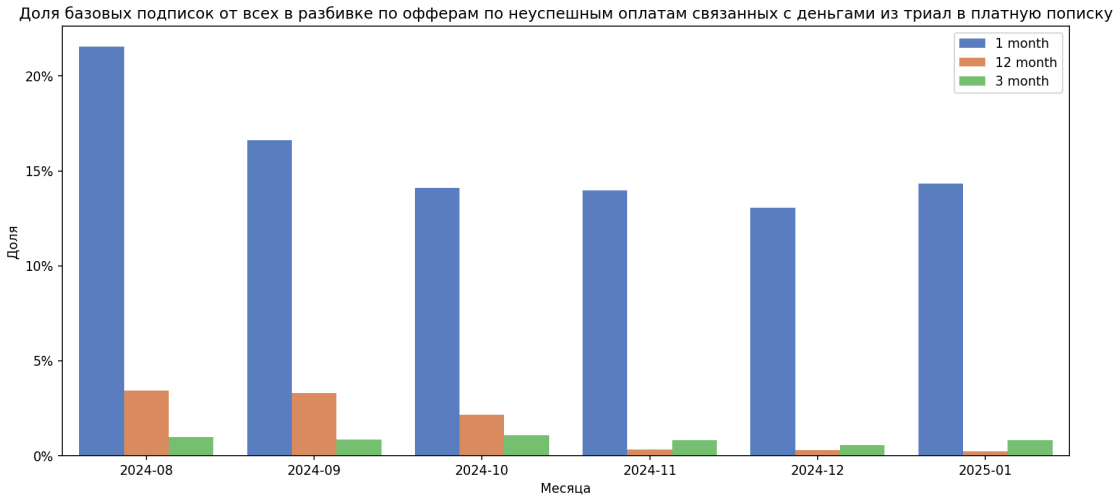
	next_invoice_state	user_id
0	current_trial/unsubscribe	22
1	failure	19
2	other	1
3	success	40

2.3

[104]:
```
df_failure_trial_success = df_failure[(df_failure['invoice_state']=='success')␣
 ↪& (df_failure['user_type']=='trial') &␣
 ↪(df_failure['next_invoice_state']=='failure') &␣
 ↪(df_failure['invoice_created_month']<'2025-02-01')]
df_failure_trial_success_agg = df_failure_trial_success.
 ↪groupby('invoice_created_month')['user_id'].nunique().reset_index()
```

```
df_failure_organic_trial =␣
 ↪df_failure_organic[(df_failure_organic['invoice_state']=='success') &␣
 ↪(df_failure_organic['user_type']=='trial') &␣
 ↪(df_failure_organic['next_invoice_state']=='failure') &␣
 ↪(df_failure_organic['invoice_created_month']<'2025-02-01')]
df_failure_organic_trial_agg = df_failure_organic_trial.
 ↪groupby(['invoice_created_month','offer_duration'])['user_id'].nunique().
 ↪reset_index()

merge_agg_trial = pd.
 ↪merge(df_failure_organic_trial_agg,df_failure_trial_success_agg,how='left',on='invoice_crea

merge_agg_trial['frac'] = merge_agg_trial['user_id_x'] /␣
 ↪merge_agg_trial['user_id_y']
```

[106]:
```
fig, (ax1) = plt.subplots(1,1,figsize=(14,6),dpi=150)

sns.barplot(data =␣
 ↪merge_agg_trial,x='invoice_created_month',y='frac',hue='offer_duration',palette␣
 ↪= 'muted',ax=ax1)
ax1.set_ylabel('  ')
ax1.set_xlabel('   ')
ax1.set_title('                                                              ␣
 ↪         ')
ax1.legend(title='')
ax1.yaxis.set_major_formatter(ticker.FuncFormatter(lambda y, _: f'{int(y *␣
 ↪100)}%'))
```



[108]:
```
merge_agg_trial.groupby('offer_duration')['frac'].median().reset_index()
```

[108]:

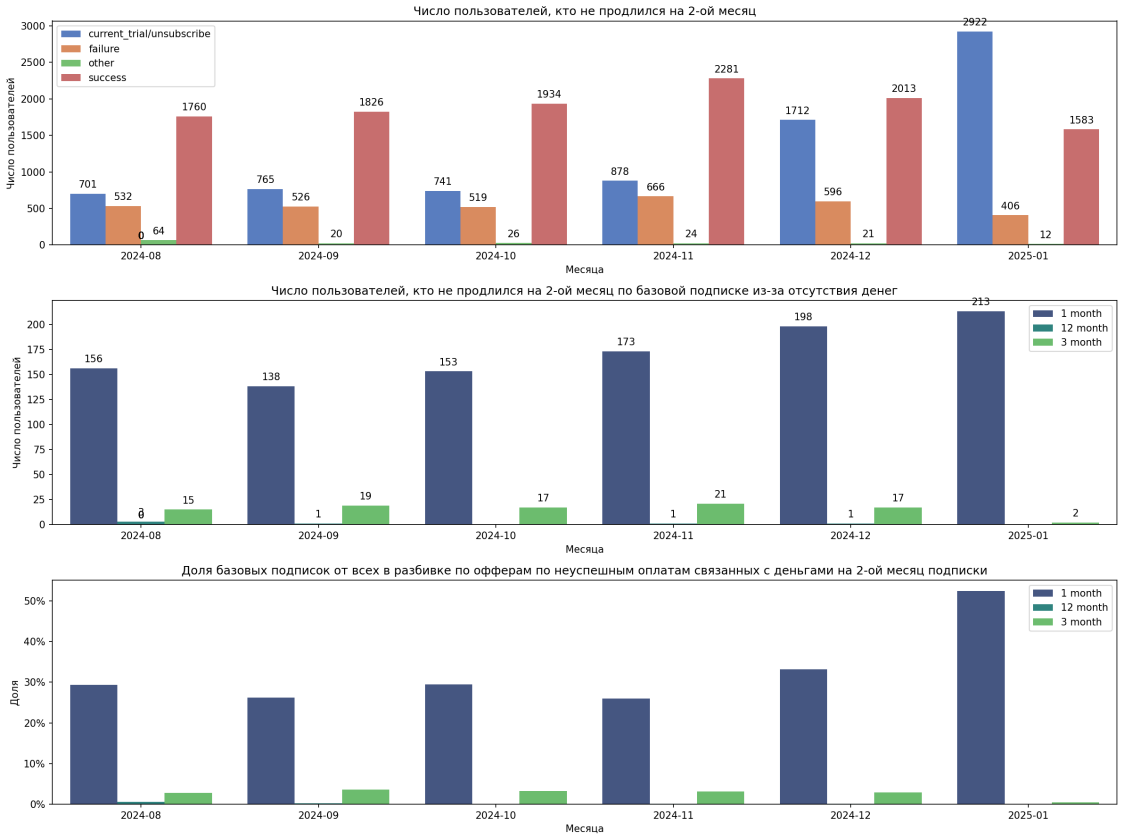
	offer_duration	frac
0	1 month	0.142163
1	12 month	0.012385
2	3 month	0.008464

2.4 1 2

[111]:
```
df_failure_all = df_failure[(df_failure['user_type']=='subs') &␣
 ↪(df_failure['invoice_state']=='success') &␣
 ↪(df_failure['invoice_created_month']<'2025-02-01')]
df_failure_all['next_invoice_state'] = df_failure_all['next_invoice_state'].
 ↪apply(lambda x: 'current_trial/unsubscribe' if x == ''  else
                                                        'other' if x in␣
 ↪['initial','refunded','processing']
                                                        else x)


df_failure_subs = df_failure[(df_failure['user_type']=='subs') &␣
 ↪(df_failure['invoice_state']=='success') &␣
 ↪(df_failure['next_invoice_state']=='failure') &␣
 ↪(df_failure['invoice_created_month']<'2025-02-01')]
df_failure_subs_agg = df_failure_subs.
 ↪groupby('invoice_created_month')['user_id'].nunique().reset_index()


df_failure_organic_subs =␣
 ↪df_failure_organic[(df_failure_organic['invoice_state']=='success') &␣
 ↪(df_failure_organic['user_type']=='subs') &␣
 ↪(df_failure_organic['next_invoice_state']=='failure') &␣
 ↪(df_failure_organic['invoice_created_month']<'2025-02-01')]
df_failure_organic_subs_agg = df_failure_organic_subs.
 ↪groupby(['invoice_created_month','offer_duration'])['user_id'].nunique().
 ↪reset_index()

merge_agg_subs = pd.
 ↪merge(df_failure_organic_subs_agg,df_failure_subs_agg,how='left',on='invoice_created_month'

merge_agg_subs['frac'] = merge_agg_subs['user_id_x'] /␣
 ↪merge_agg_subs['user_id_y']
```

[113]:
```
fig, (ax1,ax2,ax3) = plt.subplots(3,1,figsize=(16,12),dpi=150)

df_failure_all_agg = df_failure_all.
 ↪groupby(['invoice_created_month','next_invoice_state'])['user_id'].nunique().
 ↪reset_index()
```

```
barplot = sns.barplot(data =␣
 ↪df_failure_all_agg,x='invoice_created_month',y='user_id',hue='next_invoice_state',palette␣
 ↪= 'muted',ax=ax1)
ax1.set_ylabel('           ')
ax1.set_xlabel('    ')
ax1.set_title('           ,             2-     ')
ax1.legend(title='')
for p in barplot.patches:
    barplot.annotate(format(p.get_height(), '.0f'),  #
                     (p.get_x() + p.get_width() / 2., p.get_height()),
                     ha = 'center', va = 'center',
                     xytext = (0, 9),   #
                     textcoords = 'offset points')


barplot2=sns.barplot(data =␣
 ↪merge_agg_subs,x='invoice_created_month',y='user_id_x',hue='offer_duration',palette␣
 ↪= 'viridis',ax=ax2)
ax2.set_ylabel('           ')
ax2.set_xlabel('    ')
ax2.set_title('           ,             2-                  -           ')
ax2.legend(title='')

for p in barplot2.patches:
    barplot2.annotate(format(p.get_height(), '.0f'),  #
                     (p.get_x() + p.get_width() / 2., p.get_height()),
                     ha = 'center', va = 'center',
                     xytext = (0, 9),   #
                     textcoords = 'offset points')


sns.barplot(data =␣
 ↪merge_agg_subs,x='invoice_created_month',y='frac',hue='offer_duration',palette␣
 ↪= 'viridis',ax=ax3)
ax3.set_ylabel('   ')
ax3.set_xlabel('    ')
ax3.set_title('                                                    2- ␣
 ↪        ')
ax3.legend(title='')
ax3.yaxis.set_major_formatter(ticker.FuncFormatter(lambda y, _: f'{int(y *␣
 ↪100)}%'))

plt.tight_layout()
plt.show()
```



[115]:
```
merge_agg_subs2 =␣
 ↪merge_agg_subs[merge_agg_subs['invoice_created_month']<'2025-02-01'].
 ↪groupby('offer_duration')[['frac','user_id_x']].median().reset_index()
merge_agg_subs2['user_id_x'] = merge_agg_subs['user_id_x'].astype(int)

#                              ,                2-        -                ␣
 ↪
#                              ,                2-        -                    ␣
 ↪
merge_agg_subs2
```

[115]:

	offer_duration	frac	user_id_x
0	1 month	0.294015	156
1	12 month	0.001789	3
2	3 month	0.030028	15

3 android, cloudpayment ,
-

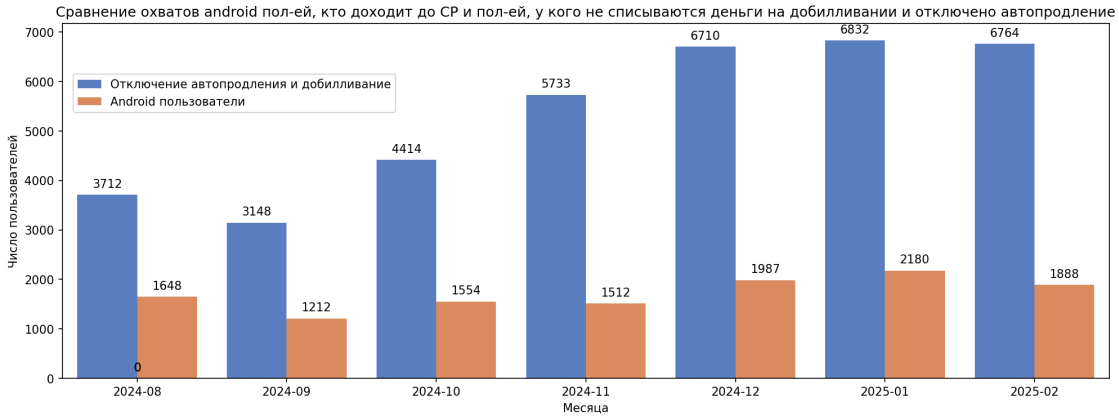
[119]:
```
query = '''SELECT utc_timestamp,date,visitor_id,user_id,event_name,client_type
            FROM datamarts.clean_event
            WHERE  date BETWEEN '2024-08-01' AND '2025-02-28'
            AND event_name='click_subscribe_offer_choose'
'''

df_android = execute(query,user='kmekhtiev')
df_android['date'] = pd.to_datetime(df_android['date'])
```

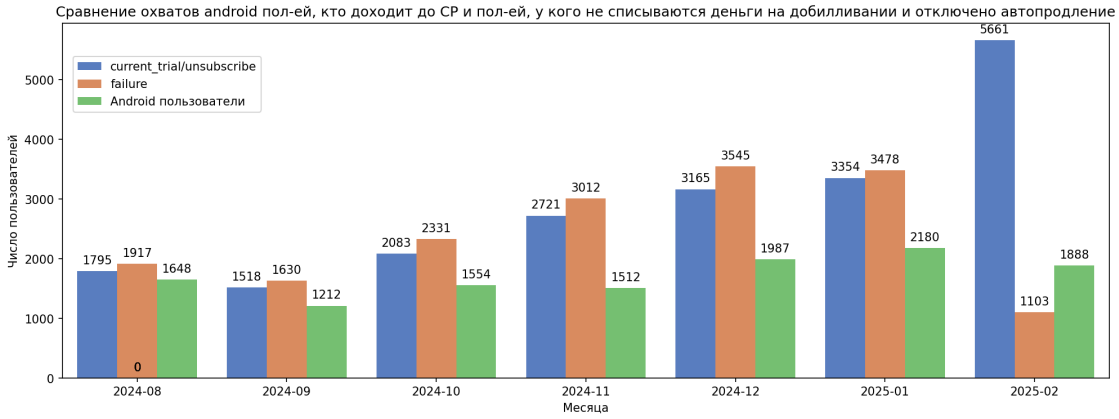
: 3.3917

[120]:
```
df_android['date_month'] = df_android['date'].dt.to_period('M')
df_android_agg = df_android[df_android['client_type']=='android'].
 ↪groupby(['client_type','date_month'])['user_id'].nunique().reset_index()
```

[121]:
```
fig, (ax1,ax2) = plt.subplots(2,1,figsize=(16,12),dpi=150)


df_invoice = r[(r['next_invoice_state'].isin(['current_trial/unsubscribe',␣
 ↪'failure'])) & (r['invoice_created_month'] <= '2025-02-01')]
df_invoice['source'] = '                        '  #           source
df_invoice = df_invoice.groupby(['invoice_created_month','source'])['user_id'].
 ↪sum().reset_index()

df_android_agg['source'] = 'Android        '  #           source

#          DataFrame
df_combined = pd.concat([
    df_invoice[['invoice_created_month', 'user_id', 'source']],
    df_android_agg[['date_month', 'user_id', 'source']].
 ↪rename(columns={'date_month': 'invoice_created_month'})
], ignore_index=True)

barplot = sns.barplot(data =␣
 ↪df_combined,x='invoice_created_month',y='user_id',hue='source',palette =␣
 ↪'muted',ax=ax1, ci=None)
ax1.set_ylabel('           ')
ax1.set_xlabel('    ')
ax1.set_title('           android   - ,          CP     - ,                    ␣
 ↪                      ')
ax1.legend(title='',loc=(0.01,0.75))
for p in barplot.patches:
    barplot.annotate(format(p.get_height(), '.0f'),  #
```

```
                        (p.get_x() + p.get_width() / 2., p.get_height()),
                        ha = 'center', va = 'center',
                        xytext = (0, 9),   #
                        textcoords = 'offset points')




df_invoice2 = r[(r['next_invoice_state'].isin(['current_trial/unsubscribe',␣
 ↪'failure'])) & (r['invoice_created_month'] <= '2025-02-01')]
df_invoice2['source'] = df_invoice2['next_invoice_state']  #            source

df_android_agg['source'] = 'Android        '  #            source

#          DataFrame
df_combined2 = pd.concat([
    df_invoice2[['invoice_created_month', 'user_id', 'source']],
    df_android_agg[['date_month', 'user_id', 'source']].
 ↪rename(columns={'date_month': 'invoice_created_month'})
], ignore_index=True)

barplot2 = sns.barplot(data =␣
 ↪df_combined2,x='invoice_created_month',y='user_id',hue='source',palette =␣
 ↪'muted',ax=ax2, ci=None)
ax2.set_ylabel('           ')
ax2.set_xlabel('    ')
ax2.set_title('           android   - ,           CP     - ,                    ␣
 ↪                       ')
ax2.legend(title='',loc=(0.01,0.75))
for p in barplot2.patches:
    barplot2.annotate(format(p.get_height(), '.0f'),  #
                        (p.get_x() + p.get_width() / 2., p.get_height()),
                        ha = 'center', va = 'center',
                        xytext = (0, 9),   #
                        textcoords = 'offset points')
```





[125]:
```
median = df_combined2[df_combined2['invoice_created_month']<'2025-02-01'].
  ↪groupby('source')['user_id'].median().reset_index()
median['user_id'] = median['user_id'].astype(int)

#                              14
median
```

[125]:

	source	user_id
0	Android	1601
1	current_trial/unsubscribe	2402
2	failure	2671

4 inititial

[982]:
```
query = '''WITH all_info AS (SELECT
        s.user_id AS user_id,
        t3.profile_id AS profile_id,
        t3.created_at AS subscription_created_at,
        i.created_at::date AS created_invoice,
        t3.reg_date AS reg_date,
```

```
       date_trunc('week',reg_date)::date AS reg_week,
       s.id AS id,
       t3.device AS device,
       t3.reg_source AS reg_source,
       t3.first_prolong_date AS first_prolong_date,
       leadInFrame(id) OVER (PARTITION BY user_id ORDER BY invoice_created_at ␣
↪ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) AS next_id,
       i.state as invoice_state,
       leadInFrame(invoice_state) OVER (PARTITION BY user_id ORDER BY␣
↪invoice_created_at  ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED␣
↪FOLLOWING) AS next_invoice_state,
       i.id as invoice_id,
       i.created_at as invoice_created_at,
       leadInFrame(invoice_created_at) OVER (PARTITION BY user_id ORDER BY␣
↪invoice_created_at  ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED␣
↪FOLLOWING) AS next_invoice_created_at,
       next_invoice_created_at::date - invoice_created_at::date delta_date,
       i.price_cents AS price_cents,
       i.price_currency AS price_currency,
       i.refund_amount_cents AS refund_amount_cents,
       ROW_NUMBER() OVER (PARTITION BY user_id,id,id,user_type,invoice_state␣
↪ORDER BY invoice_created_at) AS rn_num,
       CASE WHEN price_cents<=100 AND invoice_state='success' THEN 'trial'
            WHEN price_cents<=100 AND invoice_state IN ('failure','initial')␣
↪THEN 'not_success_trial'
            ELSE 'subs'
            END user_type,
       leadInFrame(user_type) OVER (PARTITION BY user_id ORDER BY␣
↪invoice_created_at  ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED␣
↪FOLLOWING) AS next_user_type,
       t2.renewal_off_date AS renewal_off_date,
       CASE WHEN renewal_off_date!='1970-01-01' THEN 1 ELSE 0 END␣
↪unsubscribe_type,
       t3.free_days AS free_days,
       t3.offer_duration AS offer_duration,
       t3.promo_type AS promo_type,
       i.paid_at AS paid_at,
       --i.period_end,
       s.platform AS platform,
      -- i.payment_target,
       i.deleted_at,
       i.refunded_at
  FROM raw.viju__product_x__public__invoices i
  LEFT JOIN raw.viju__product_x__public__subscriptions s
       ON i.subscription_id = s.id
  LEFT JOIN (SELECT
```

```
                subscription_id,
                max(created_at) as renewal_off_date
                FROM raw.viju__product_x__public__subscription_cancel_reasons
                WHERE created_at >= '2022-03-01'
                 --AND subscription_id='74bcff94-4ada-40ef-85de-25074d615d57'
                GROUP by 1
                ) AS t2 ON i.subscription_id=t2.subscription_id
    INNER JOIN datamarts.marketing_dash AS t3 ON s.user_id=t3.user_id
    WHERE s.platform != 'api'
    ORDER BY s.user_id, invoice_created_at
    )
--
SELECT * FROM all_info
WHERE invoice_created_at::date BETWEEN '2024-08-01' AND '2025-05-04'
AND reg_date>='2025-03-01'
--AND user_type!='not_success_trial'
AND platform=='cloudpayments'
'''

df_initial = execute(query,user='kmekhtiev')
df_initial['reg_date'] = pd.to_datetime(df_initial['reg_date'])
df_initial['reg_week'] = pd.to_datetime(df_initial['reg_week'])
df_initial['first_prolong_date'] = pd.
 ↪to_datetime(df_initial['first_prolong_date'])
df_initial['created_invoice'] = pd.to_datetime(df_initial['created_invoice'])
```

```
: 143.6255
```

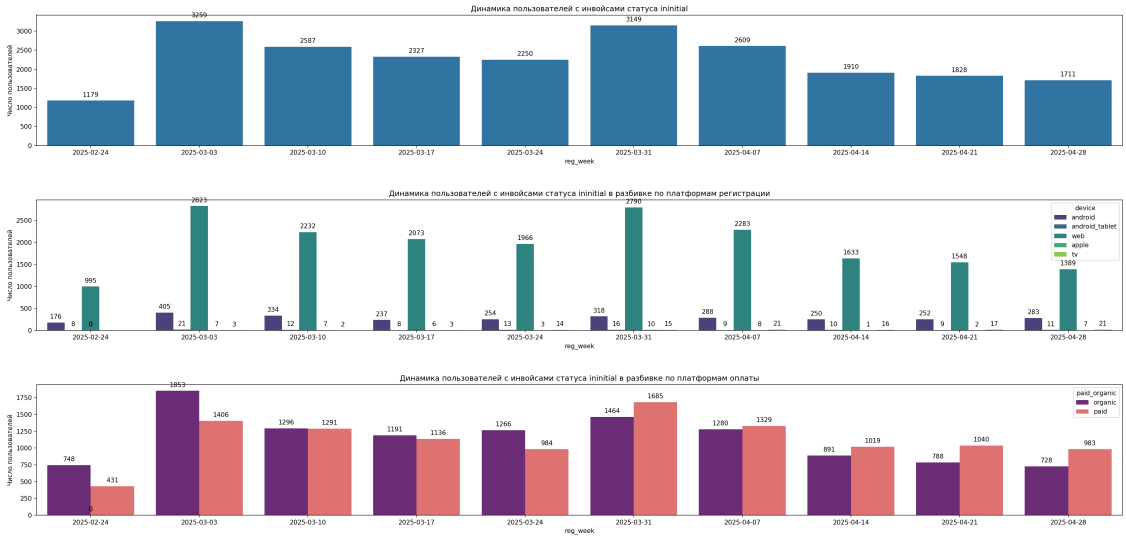
[984]:
```
df_initial_agg = df_initial.groupby('user_id')['invoice_state'].nunique().
 ↪reset_index()
only_one_state_users = df_initial_agg[df_initial_agg['invoice_state'] == 1]

initial_only_df = df_initial[df_initial['invoice_state'] == 'initial']
result = initial_only_df[initial_only_df['user_id'].
 ↪isin(only_one_state_users['user_id'])]

result = result[result['subscription_created_at'] == '1970-01-01']

#result['paid_organic'] = result.apply(lambda x:'organic' if x=='none' else␣
 ↪'paid',raw = True)
result['paid_organic'] = result['reg_source'].apply(lambda x: 'organic' if x ==␣
 ↪'none' else 'paid')
```

[986]:
```
result['reg_week'].unique()
```

[986]:
```
<DatetimeArray>
['2025-04-21 00:00:00', '2025-04-07 00:00:00', '2025-03-31 00:00:00',
 '2025-03-03 00:00:00', '2025-03-24 00:00:00', '2025-04-14 00:00:00',
 '2025-04-28 00:00:00', '2025-03-10 00:00:00', '2025-03-17 00:00:00',
 '2025-02-24 00:00:00']
Length: 10, dtype: datetime64[ns]
```

[988]:
```
fig,(ax1,ax2,ax3) = plt.subplots(3,1,figsize=(25,12),dpi=150)

result_agg = result.groupby('reg_week')['user_id'].nunique().reset_index()
barplot = sns.barplot(data = result_agg,x='reg_week',y='user_id', ci=None,␣
 ↪ax=ax1)
ax1.set_ylabel('           ')
ax1.set_title('                          ininitial')
for p in barplot.patches:
    barplot.annotate(format(p.get_height(), '.0f'),  #
                     (p.get_x() + p.get_width() / 2., p.get_height()),
                     ha = 'center', va = 'center',
                     xytext = (0, 9),   #
                     textcoords = 'offset points')



result_agg2 = result.groupby(['reg_week','device'])['user_id'].nunique().
 ↪reset_index()
barplot2 = sns.barplot(data =␣
 ↪result_agg2,x='reg_week',y='user_id',hue='device',palette='viridis',␣
 ↪ci=None, ax=ax2)
ax2.set_ylabel('           ')
ax2.set_title('                          ininitial                      ')
for p in barplot2.patches:
    barplot2.annotate(format(p.get_height(), '.0f'),  #
                     (p.get_x() + p.get_width() / 2., p.get_height()),
                     ha = 'center', va = 'center',
                     xytext = (0, 9),   #
                     textcoords = 'offset points')



result_agg3 = result.groupby(['reg_week','paid_organic'])['user_id'].nunique().
 ↪reset_index()

barplot3 = sns.barplot(data =␣
 ↪result_agg3,x='reg_week',y='user_id',hue='paid_organic',palette='magma',␣
 ↪ci=None, ax=ax3)
ax3.set_ylabel('           ')
ax3.set_title('                          ininitial                    ')
```

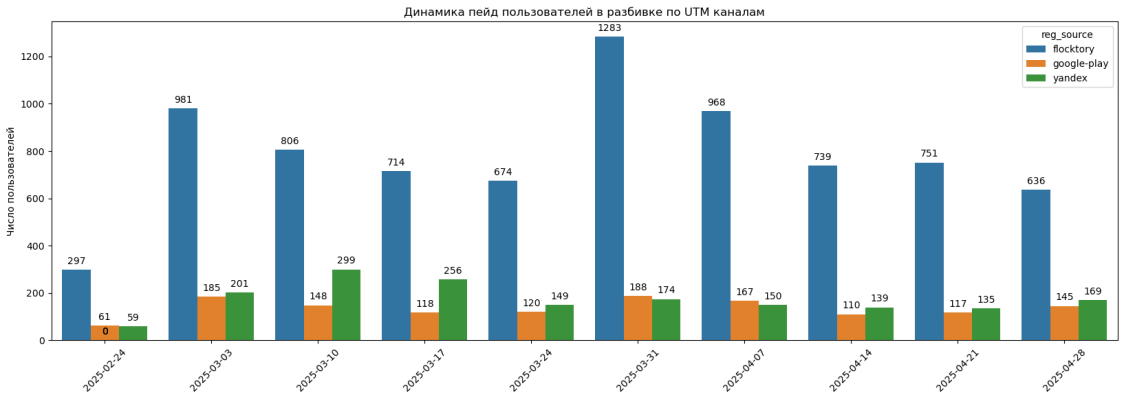
```
for p in barplot3.patches:
    barplot3.annotate(format(p.get_height(), '.0f'),  #
                      (p.get_x() + p.get_width() / 2., p.get_height()),
                      ha = 'center', va = 'center',
                      xytext = (0, 9),   #
                      textcoords = 'offset points')


plt.subplots_adjust(hspace=5)  # Increase this value for more space
plt.tight_layout()
plt.show()
```



4.1

[999]:
```
plt.figure(figsize=(20,6))

result_agg4 = result[result['paid_organic']=='paid'].
 ↪groupby(['reg_week','reg_source'])['user_id'].nunique().reset_index()
barplot = sns.barplot(data =␣
 ↪result_agg4[result_agg4['user_id']>30],x='reg_week',y='user_id',hue='reg_source',␣
 ↪ci=None)
plt.ylabel('           ')
plt.xlabel('')
plt.xticks(rotation=45)
plt.title('                          UTM     ')
for p in barplot.patches:
    barplot.annotate(format(p.get_height(), '.0f'),  #
                      (p.get_x() + p.get_width() / 2., p.get_height()),
                      ha = 'center', va = 'center',
```

```
                        xytext = (0, 9),    #
                        textcoords = 'offset points')

plt.show()
```



[1001]:
```
query = '''
SELECT
        date,
        user_id,
        client_type,
    device_type,
    app_version,
    os_version,
    browser_version,
    country,
    entry_point,
    entry_point || '_' || event_page as entry_point_event_page,
    user_type,
    event_page,
    offer_type,
    if(countIf (event_name = 'goto_offer_page')>0, 1, 0) as␣
 ↪step_1_goto_offer_page,
    if(countIf (event_name = 'click_subscribe_offer_choose')>0, 1, 0) as␣
 ↪step_2_click_subscribe_offer_choose,
    if(countIf (event_name = 'click_subscribe_pay')>0, 1, 0) as␣
 ↪step_3_click_subscribe_pay
FROM(
SELECT
    date_raw,
    user_type_raw,
    user_id,
    client_type,
```

```
  device_type,
  os_version,
  app_version,
  browser_version,
  country,
  entry_point_raw,
  offer_type_raw,
  event_name,
  utc_timestamp,
  row_number() over (partition by user_id order by utc_timestamp) as step_num,
  case
    when event_name = 'click_subscribe_offer_choose'
      and lagInFrame(event_name, 1) over (partition by user_id order by␣
↪utc_timestamp) == 'goto_offer_page'
    then lagInFrame(entry_point_raw, 1) over (partition by user_id order by␣
↪utc_timestamp)
    when event_name = 'click_subscribe_pay'
      and lagInFrame(event_name, 2) over (partition by user_id order by␣
↪utc_timestamp) == 'goto_offer_page'
      and lagInFrame(event_name, 1) over (partition by user_id order by␣
↪utc_timestamp) == 'click_subscribe_offer_choose'
    then lagInFrame(entry_point_raw, 2) over (partition by user_id order by␣
↪utc_timestamp)
    else entry_point_raw
  end as entry_point,
  case
    when event_name = 'click_subscribe_offer_choose'
      and lagInFrame(event_name, 1) over (partition by user_id order by␣
↪utc_timestamp) == 'goto_offer_page'
    then lagInFrame(date_raw, 1) over (partition by user_id order by␣
↪utc_timestamp)
    when event_name = 'click_subscribe_pay'
      and lagInFrame(event_name, 2) over (partition by user_id order by␣
↪utc_timestamp) == 'goto_offer_page'
      and lagInFrame(event_name, 1) over (partition by user_id order by␣
↪utc_timestamp) == 'click_subscribe_offer_choose'
    then lagInFrame(date_raw, 2) over (partition by user_id order by␣
↪utc_timestamp)
    else date_raw
  end as date,  --   :            ,     /    -     +1     .            ␣
↪
  case
    when event_name = 'click_subscribe_offer_choose'
      and lagInFrame(event_name, 1) over (partition by user_id order by␣
↪utc_timestamp) == 'goto_offer_page'
```

```
    then lagInFrame(user_type_raw, 1) over (partition by user_id order by␣
↪utc_timestamp)
    when event_name = 'click_subscribe_pay'
      and lagInFrame(event_name, 2) over (partition by user_id order by␣
↪utc_timestamp) == 'goto_offer_page'
      and lagInFrame(event_name, 1) over (partition by user_id order by␣
↪utc_timestamp) == 'click_subscribe_offer_choose'
    then lagInFrame(user_type_raw, 2) over (partition by user_id order by␣
↪utc_timestamp)
    else user_type_raw
  end as user_type,  --
  case
    when event_name = 'click_subscribe_offer_choose'
      and lagInFrame(event_name, 1) over (partition by user_id order by␣
↪utc_timestamp) == 'goto_offer_page'
    then lagInFrame(event_page_raw, 1) over (partition by user_id order by␣
↪utc_timestamp)
    when event_name = 'click_subscribe_pay'
      and lagInFrame(event_name, 2) over (partition by user_id order by␣
↪utc_timestamp) == 'goto_offer_page'
      and lagInFrame(event_name, 1) over (partition by user_id order by␣
↪utc_timestamp) == 'click_subscribe_offer_choose'
    then lagInFrame(event_page_raw, 2) over (partition by user_id order by␣
↪utc_timestamp)
    else event_page_raw
  end as event_page, --
  case
    when event_name = 'click_subscribe_offer_choose'
    and lagInFrame(event_name, 1) over (partition by user_id order by␣
↪utc_timestamp desc) == 'click_subscribe_pay'
    then lagInFrame(offer_type_raw, 1) over (partition by user_id order by␣
↪utc_timestamp desc)
    when event_name = 'goto_offer_page'
    and lagInFrame(event_name, 1) over (partition by user_id order by␣
↪utc_timestamp desc) == 'click_subscribe_offer_choose'
    and lagInFrame(event_name, 2) over (partition by user_id order by␣
↪utc_timestamp desc) == 'click_subscribe_pay'
    then lagInFrame(offer_type_raw, 2) over (partition by user_id order by␣
↪utc_timestamp desc)
    else offer_type_raw
  end as offer_type
  FROM (
    SELECT
      date as date_raw,
      event_name,
      user_id,
```

```
       client_type, -- android, ios, web_desktop, web_mobile
       utc_timestamp,
       first_prolong_date,
       state,
       reg_date,
       device_type,
       os_version,
       app_version,
       browser_version,
       created_at, --              /
       ends_at, --                                    -
       country,
       JSONExtractString(payload,'from') as payload_from,
       if (event_page='' or event_page is null, 'other', event_page) as␣
↪event_page_raw,
       case
         when event_name in ('click_subscribe_offer_choose',␣
↪'click_subscribe_pay')  then '' --      2   3
         when payload_from = 'action_button'
           and event_page in ('main', 'movie', 'series', 'tvchannel', 'tv',␣
↪'mychannel', 'collections','catalog', 'compilation') then '   '
         when payload_from in ('', 'banner') and event_page = 'main' then␣
↪'    ' --          -       banner
         when payload_from = 'watch_button' and event_page = 'main' then '    '␣
↪--
         when payload_from = 'watch_button' and event_page in ('item',␣
↪'tvchannel', 'mychannel', 'movie', 'series', 'collections') then '     '
         when payload_from = 'tbd_    _  _ _   ' then '        '
         --
         when payload_from = 'after_free_watch' and event_page in ('item',␣
↪'series') then '     after_free'
         when payload_from = 'action_button' and event_page = 'account' then␣
↪'    '
         when payload_from = 'free_tv' and event_page = 'tvchannel' then '   .
↪  pop-up'
         when payload_from = 'after_auth' and event_page = 'account' then␣
↪'after_auth'
         else '    '
       end as entry_point_raw, --       1
       case
            when profile_id is null then '       '
            --                ,
            when (first_prolong_date != '1970-01-01' and first_prolong_date␣
↪<= date)  then '      (  /  )'
         when (first_prolong_date = '1970-01-01' or first_prolong_date > date)␣
↪and (created_at != '1970-01-01 00:00:00' and created_at <= date)
```

```
                then '      (  /  )'
        when (first_prolong_date = '1970-01-01' or first_prolong_date > date)␣
↪and (created_at = '1970-01-01 00:00:00' or created_at > date)
                and (reg_date != '1970-01-01' and reg_date < date) then␣
↪'   '
        when (first_prolong_date = '1970-01-01' or first_prolong_date > date)␣
↪and (created_at = '1970-01-01 00:00:00' or created_at > date)
                and reg_date = date then '    (    )'
        when reg_date = '1970-01-01' and profile_id is not null then '    (no␣
↪data)' --
        else '    '
      end as user_type_raw, --       1
        case
             when event_name in ('goto_offer_page',␣
↪'click_subscribe_offer_choose')  then ''
                --                :
              when free_days = 14 and price_cents = 29900 and date <=␣
↪'2025-04-10'
                   or free_days = 14 and price_cents = 39900 and date >=␣
↪'2025-04-10'  then '1   '
                when free_days = 3 and price_cents = 49900 and date <=␣
↪'2025-04-10'
                        or free_days = 3 and price_cents = 64900 and date␣
↪>= '2025-04-10' then '3   '
                when free_days = 3 and price_cents = 119000 and date <=␣
↪'2025-04-10'
                        or free_days = 3 and price_cents = 155000 and date␣
↪>= '2025-04-10'  then '12   '
                when free_days not in (3, 14) and price_cents not in␣
↪(29900, 49900, 119000) and date <= '2025-04-10'
                        or free_days not in (3, 14) and price_cents not in␣
↪(39900, 64900, 155000) and date >= '2025-04-10'
                then '   :   + '
                when free_days not in (3, 14) then '   :   '
                when price_cents not in (29900, 49900, 119000) and date <=␣
↪'2025-04-10'
                        or price_cents not in (39900, 64900, 155000) and␣
↪date >= '2025-04-10'  then '   :   '
                else '    '
              end as offer_type_raw --       3
    FROM datamarts.clean_event AS ce
    WHERE event_name in ('goto_offer_page',  'click_subscribe_offer_choose',␣
↪'click_subscribe_pay','show_subscribe_offer_page','auto_subscribe_cloudpayments_displayed')
    and client_type in ('web_desktop', 'web_mobile')
    and date >= '2025-03-01'
  )
```

```
) where entry_point != '' AND user_id IS NOT NULL
GROUP BY
  date, user_id, client_type, device_type, os_version, app_version,␣
 ↪browser_version, country, entry_point, user_type, event_page, offer_type
ORDER BY step_1_goto_offer_page DESC
'''

df_entry_point = execute(query,user='kmekhtiev')
```

```
              : 28.918
```

[715]:
```
result[result['device']=='web']['user_id'].nunique()
```

[715]:
```
18535
```

[1003]:
```
merge = pd.
 ↪merge(result[result['device']=='web'],df_entry_point[['user_id','event_page','entry_point',
```

[1004]:
```
merge.groupby('entry_point')['user_id'].nunique()
```

[1004]:
```
entry_point
after_auth               3370
                            1
                        19339
                           59
                          241
     after_free             4
                          148
Name: user_id, dtype: int64
```

[1007]:
```
merge[merge['entry_point']=='    '].groupby('client_type')['user_id'].nunique()
```

[1007]:
```
client_type
web_desktop     3592
web_mobile     15870
Name: user_id, dtype: int64
```

[1009]:
```
merge[merge['entry_point']=='    '].
 ↪groupby('device_type',as_index=False)['user_id'].nunique().
 ↪sort_values(by='user_id', ascending = False).head(20)
```

[1009]:

	device_type	user_id
324	Web / Desktop APP / Unknown	8351
856	Web / SM-A515F / Unknown	123
87	Web / 23124RA7EO / Unknown	120
841	Web / SM-A325F / Unknown	117
731	Web / RMX3834 / Unknown	114

```
529           Web / M2101K6G / Unknown      109
858           Web / SM-A525F / Unknown      104
870           Web / SM-A556E / Unknown      104
536         Web / M2102J20SG / Unknown      101
533         Web / M2101K7BNY / Unknown       98
425     Web / Infinix X6833B / Unknown       97
83          Web / 23117RA68G / Unknown       96
776   Web / Redmi Note 8 Pro / Unknown       92
1072         Web / TECNO BG6 / Unknown       85
867           Web / SM-A546E / Unknown       84
825           Web / SM-A155F / Unknown       81
1017          Web / SM-S928B / Unknown       80
1122        Web / TECNO LH7n / Unknown       79
1049           Web / STK-LX1 / Unknown       79
60          Web / 23021RAA2Y / Unknown       79
```

[1011]:
```
merge[(merge['entry_point']=='    ') & (merge['client_type']=='web_desktop')].
↪groupby('device_type',as_index=False)['user_id'].nunique().
↪sort_values(by='user_id', ascending = False).head(20)
```

[1011]:
```
                        device_type  user_id
27       Web / Desktop APP / Unknown     3426
4         Web / 23073RPBFG / Unknown        8
3         Web / 23043RP34G / Unknown        7
0    Samsung / Desktop APP / Unknown        6
17          Web / AGS6-W09 / Unknown        5
102          Web / Unknown / Unknown        5
83           Web / SM-X205 / Unknown        5
68           Web / SM-P615 / Unknown        4
74           Web / SM-T585 / Unknown        4
13          Web / AGS2-L09 / Unknown        4
1          Web / 21051182G / Unknown        4
72           Web / SM-T505 / Unknown        4
30           Web / ELN-L09 / Unknown        3
14         Web / AGS3K-L09 / Unknown        3
26           Web / DBY-W09 / Unknown        2
53    Web / Lenovo TB128FU / Unknown        2
50   Web / Lenovo TB-X606X / Unknown        2
81           Web / SM-T975 / Unknown        2
86          Web / SM-X816B / Unknown        2
48   Web / Lenovo TB-X306X / Unknown        2
```

[1013]:
```
merge[(merge['entry_point']=='    ') & (merge['client_type']=='web_mobile')].
↪groupby('device_type',as_index=False)['user_id'].nunique().
↪sort_values(by='user_id', ascending = False).head(20)
```

[1013]:

	device_type	user_id
305	Web / Desktop APP / Unknown	4958
804	Web / SM-A515F / Unknown	123
82	Web / 23124RA7EO / Unknown	120
789	Web / SM-A325F / Unknown	117
680	Web / RMX3834 / Unknown	114
487	Web / M2101K6G / Unknown	109
818	Web / SM-A556E / Unknown	104
806	Web / SM-A525F / Unknown	104
494	Web / M2102J20SG / Unknown	101
491	Web / M2101K7BNY / Unknown	98
398	Web / Infinix X6833B / Unknown	97
78	Web / 23117RA68G / Unknown	96
724	Web / Redmi Note 8 Pro / Unknown	92
1000	Web / TECNO BG6 / Unknown	85
815	Web / SM-A546E / Unknown	84
773	Web / SM-A155F / Unknown	81
964	Web / SM-S928B / Unknown	80
57	Web / 23021RAA2Y / Unknown	79
982	Web / STK-LX1 / Unknown	79
1050	Web / TECNO LH7n / Unknown	79

[1015]:
```
merge[(merge['entry_point']=='    ') & (merge['client_type']=='web_mobile')].
↪groupby('os_version',as_index=False)['user_id'].nunique().
↪sort_values(by='user_id', ascending = False).head(20)
```

[1015]:

	os_version	user_id
11	Android v.14.0.0	3691
9	Android v.13.0.0	2692
7	Android v.12.0.0	1363
112	iPhone iOS v.18.32	1040
1	Android v.	928
5	Android v.11.0.0	921
13	Android v.15.0.0	917
3	Android v.10.0.0	900
111	iPhone iOS v.18.31	786
107	iPhone iOS v.18.11	224
23	Android v.9.0.0	222
113	iPhone iOS v.18.4	186
100	iPhone iOS v.17.61	176
114	iPhone iOS v.18.41	155
0	v.	155
79	iPhone iOS v.16.71	109
98	iPhone iOS v.17.51	98
10	Android v.14	97
109	iPhone iOS v.18.21	90
110	iPhone iOS v.18.3	88

[898]:
```
df_initial_success = df_initial[df_initial['invoice_state']=='success']
df_initial_success['paid_organic'] = df_initial_success['reg_source'].
 ↪apply(lambda x: 'organic' if x == 'none' else 'paid')

merge = pd.
 ↪merge(df_initial_success[df_initial_success['device']=='web'],df_entry_point[['user_id','ev


merge.groupby('entry_point')['user_id'].nunique()
```

[898]:
```
entry_point
after_auth                3063
   .  pop-up               4
                          15097
                             19
                           200
     after_free            88
                            220
Name: user_id, dtype: int64
```

4.2

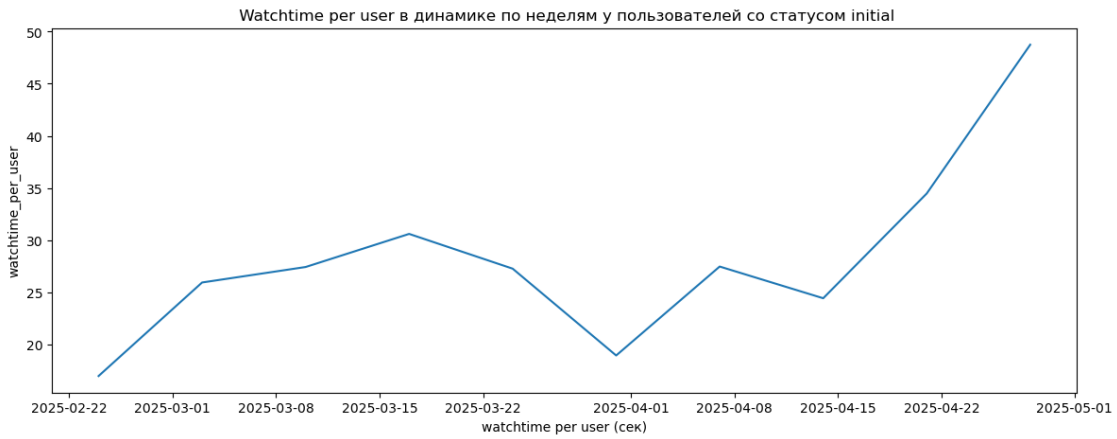
[1047]:
```
query = f''' SELECT date,
             date_trunc('week',date) AS date_week,
             profile_id,
             item_type,
             sum(watchtime) AS watchtime
             FROM datamarts.watchtime_by_day
             WHERE date>='2025-03-01'
             GROUP BY 1,2,3,4
             '''

df_watchtime = execute(query,user='kmekhtiev')
df_watchtime['date'] = pd.to_datetime(df_watchtime['date'])
df_watchtime['date_week'] = pd.to_datetime(df_watchtime['date_week'])
```

```
             : 14.6794
```

[1032]:
```
result['created_invoice_max'] = result.groupby(['user_id'])['created_invoice'].
 ↪transform('max')

result_initial = result.
 ↪groupby(['profile_id','reg_date','created_invoice_max'])['user_id'].
 ↪nunique().reset_index()
```

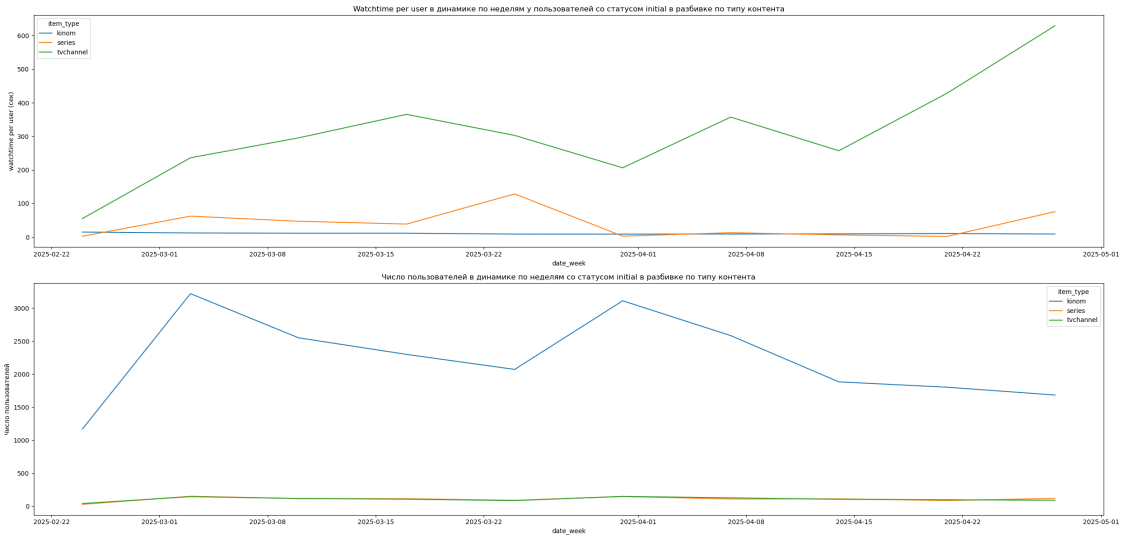
```
merge = pd.
 ↪merge(df_watchtime[['date','date_week','profile_id','watchtime','item_type']],result_initia
 ↪= ['profile_id','date'], right_on = ['profile_id','reg_date'], how='inner')

merge_agg = merge.groupby('date_week').agg({'profile_id':'nunique','watchtime':
 ↪'sum'}).reset_index()
merge_agg['watchtime_per_user'] = merge_agg['watchtime'] /␣
 ↪merge_agg['profile_id']


plt.figure(figsize=(14,5))

sns.lineplot(merge_agg,x='date_week',y='watchtime_per_user')
plt.xlabel('watchtime per user (  )')
plt.title (' Watchtime per user                                    initial')
```

[1032]:
```
Text(0.5, 1.0, ' Watchtime per user
     initial')
```



[1034]:
```
merge_agg = merge.groupby(['date_week','item_type']).agg({'profile_id':
 ↪'nunique','watchtime':'sum'}).reset_index()
merge_agg['watchtime_per_user'] = merge_agg['watchtime'] /␣
 ↪merge_agg['profile_id']

merge_agg = merge_agg[merge_agg['item_type'].
 ↪isin(['kinom','tvchannel','series'])]

fig,(ax1,ax2) = plt.subplots(2,1,figsize=(25,12))

lineplot = sns.
 ↪lineplot(merge_agg,x='date_week',y='watchtime_per_user',hue='item_type',ax=ax1)
```

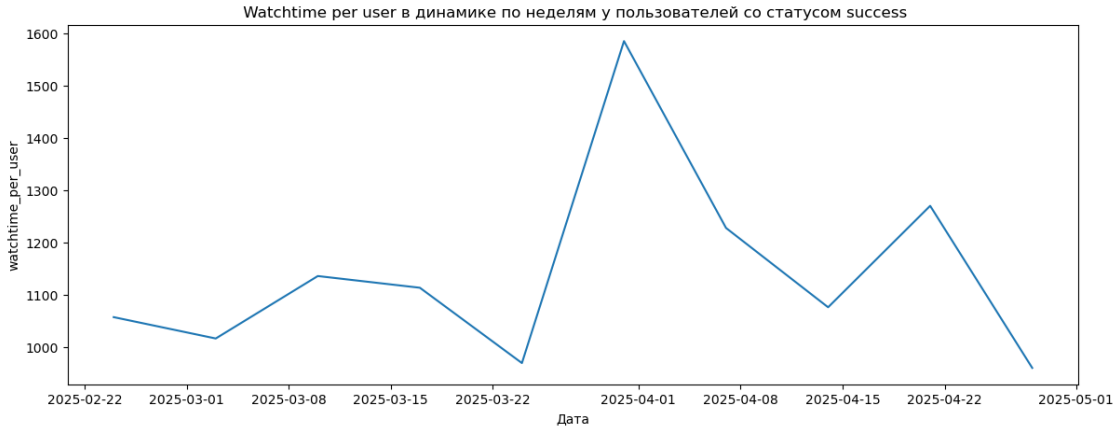
```
ax1.set_ylabel('watchtime per user (  )')
ax1.set_title ('Watchtime per user                                    initial
  ↪          ')


lineplot = sns.
  ↪lineplot(merge_agg,x='date_week',y='profile_id',hue='item_type',ax=ax2)
ax2.set_ylabel('           ')
ax2.set_title ('                                initial                 ')


plt.subplots_adjust(hspace=5)  # Increase this value for more space
plt.tight_layout()
plt.show()
```



[1036]:
```
df_initial_success['created_invoice_max'] = df_initial_success.
  ↪groupby(['profile_id'])['created_invoice'].transform('max')

df_initial_success_agg = df_initial_success.
  ↪groupby(['profile_id','reg_date','created_invoice_max'])['user_id'].
  ↪nunique().reset_index()



df_initial_success_agg[df_initial_success_agg['reg_date'] !=
  ↪df_initial_success_agg['created_invoice_max']]['profile_id'].nunique()
```

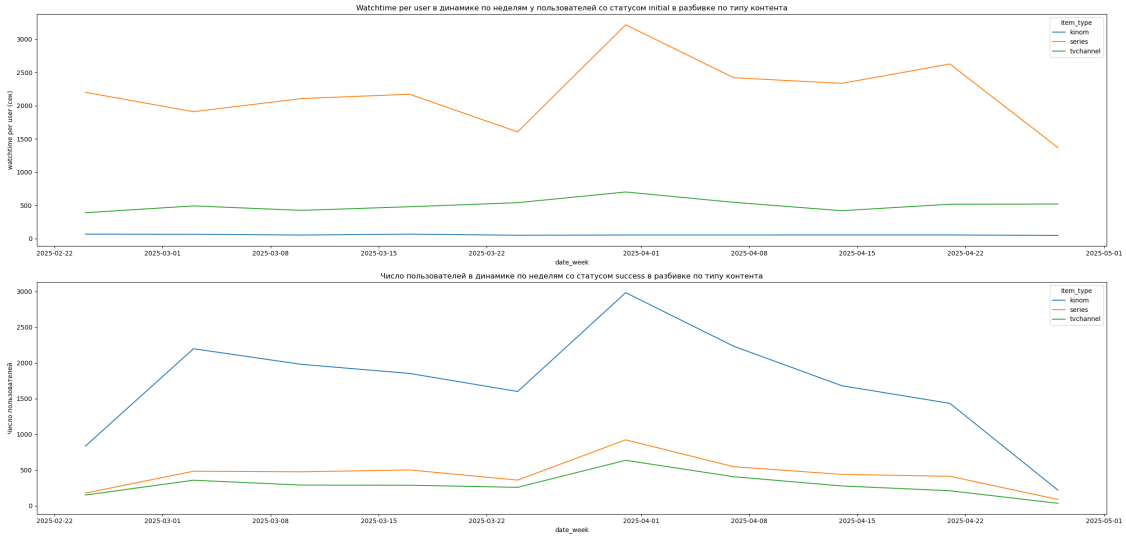
```
merge = pd.
 ↪merge(df_watchtime[['date','date_week','profile_id','watchtime','item_type']],df_initial_su
 ↪= ['profile_id','date'], right_on = ['profile_id','reg_date'], how='inner')

merge_agg = merge.groupby('date_week').agg({'profile_id':'nunique','watchtime':
 ↪'sum'}).reset_index()
merge_agg['watchtime_per_user'] = merge_agg['watchtime'] /␣
 ↪merge_agg['profile_id']


plt.figure(figsize=(14,5))

sns.lineplot(merge_agg,x='date_week',y='watchtime_per_user')
plt.xlabel('  ')
plt.title (' Watchtime per user                                    success')
```

[1036]:
```
Text(0.5, 1.0, ' Watchtime per user
     success')
```



[1044]:
```
merge_agg = merge.groupby(['date_week','item_type']).agg({'profile_id':
 ↪'nunique','watchtime':'sum'}).reset_index()
merge_agg['watchtime_per_user'] = merge_agg['watchtime'] /␣
 ↪merge_agg['profile_id']


merge_agg = merge_agg[merge_agg['item_type'].
 ↪isin(['kinom','tvchannel','series'])]

fig,(ax1,ax2) = plt.subplots(2,1,figsize=(25,12))
```

```
lineplot = sns.
 ↪lineplot(merge_agg,x='date_week',y='watchtime_per_user',hue='item_type',ax=ax1)
ax1.set_ylabel('watchtime per user (  )')
ax1.set_title ('Watchtime per user                                    initial
 ↪          ')


lineplot = sns.
 ↪lineplot(merge_agg,x='date_week',y='profile_id',hue='item_type',ax=ax2)
ax2.set_ylabel('           ')
ax2.set_title ('                                  success                  ')


plt.subplots_adjust(hspace=5)  # Increase this value for more space
plt.tight_layout()
plt.show()
```



[ ]: