FORMEL Skin BI Roles - Case Study

These questions are meant as a guide towards working with the dataset. They are not an exhaustive list of what can be discussed, nor is it entirely required to answer/present each of these points. If you have a well reasoned explanation for why other areas were/are a better investment of business time, that is ok.

- Present the change and total user counts over time
- Present the acquisition, churn and activity of users over time, with reference to %, deltas, and total counts.
- Present the recurring revenue as it changes over time, with respect to cumulative, and absolute and relative changes over time.
- Considering monthly marketing costs outline and a fixed consultation cost of 20 EUR represent profit, in a similar manner to above.
- Consideration of churn.

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
from datetime import date, timedelta
import matplotlib.cbook as cboo
import plotly.express as px
import plotly.graph_objects as go

spendings_df = pd.read_csv('marketing_spend.csv')
subs_df = pd.read_csv('subscriptions.csv',
                      dtype={"user_id": 'string', "subscription_id": "string"})
```
```
C:\Users\Oliver\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3071: DtypeWarning: Colu
mns (11) have mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

Viewing the data

As a first step we are viewing the columns for their importance, their data type and for missing values.

In [2]:
```
spendings_df
```
Out[2]:

	marketing_channel	monthly_total_spend
0	Direct	4759
1	Influencer	7477
2	Social Ads	4586
3	Social Media Native	9912
4	Partnerships	5695
5	Customer Referral Program	5416
6	Paid Search - Brand	7182
7	Paid Search - Non Brand	3775
8	Display	4857
9	CRM	4017
10	PR	8468
11	YouTube	4183
12	Internal	6828

In [3]:
```
subs_df.head()
```
Out[3]:

	user_id	subscription_id	first_voucher_type	purchase_date	subscription_start_date	subscription_end_dat
0	5edea4e73c4c4b5720177b82	617fdd813c945f0026801c6f	NaN	2021-11-01	2021-11-01	2021-12-0
1	5edea4e73c4c4b5720177b82	61a76a8ab1f9130024ada53d	NaN	2021-12-01	2021-12-01	2022-01-0
2	5edea4e73c4c4b5720177b82	61d0490b26020f0025f1954e	NaN	2022-01-01	2022-01-01	2022-02-2
3	5edea4e73c4c4b5720177b82	6219b500669deb0025a8c8aa	NaN	2022-02-26	2022-02-26	2022-03-2
4	5eec529809774a00238deaf4	6154ba281ea1650025c18a57	marketing 2x20	2021-09-29	2021-09-29	2021-10-2

In [4]:
```
subs_df.isnull().sum()
```
Out[4]:
```
user_id                        0
subscription_id                0
first_voucher_type         15100
purchase_date                  0
subscription_start_date        0
subscription_end_date          0
subscription_interval          0
cancellation_date          31141
lead_time_in_hours          2054
gross_price                    0
user_created_date              0
newsletter_subscription        4
gender                         0
age                            0
diagnosis_condition         3148
diagnosis_severity         11159
ask_your_doctor_count      32790
checkin_date               18894
checkin_score              37587
marketing_channel              8
dtype: int64
```

In [5]:
```
subs_df.dtypes
```
Out[5]:
```
user_id                     string
subscription_id             string
first_voucher_type          object
purchase_date               object
subscription_start_date     object
subscription_end_date       object
subscription_interval        int64
cancellation_date           object
lead_time_in_hours         float64
gross_price                float64
user_created_date           object
newsletter_subscription     object
gender                      object
age                          int64
diagnosis_condition         object
diagnosis_severity          object
ask_your_doctor_count      float64
checkin_date                object
checkin_score              float64
marketing_channel           object
dtype: object
```

Cleaning / Rearranging the subs_df

For further use we are encoding the ids, drop unnecessary columns and rearrange the subs_df into a user dataset und a subscription dataset.

In [6]:
```
#Encoding the input-list
def id_mapper(_list_):
    '''
    INPUT:
    _list_ - (list) a list

    OUTPUT:
    id_encoded - (list) an encoded version of the _list_
    '''
    coded_dict = dict()
    cter = 1
    id_encoded = []

    for val in _list_:
        if val not in coded_dict:    #to ensure each ID will only be encoded once
            coded_dict[val] = cter
            cter+=1

        id_encoded.append(coded_dict[val])
    return id_encoded
```

In [7]:
```
#Encoding the User_ID and the Subscription_ID
user_id_list = subs_df['user_id'].tolist()
sub_id_list = subs_df['subscription_id'].tolist()

id_encoded = id_mapper(user_id_list)
id_encoded_2 = id_mapper(sub_id_list)

id_list = id_encoded
sub_list = id_encoded_2

del subs_df['user_id']
subs_df['User_ID'] = id_list #Substituting the User ID

del subs_df['subscription_id']
subs_df['Subscription_ID'] = sub_list #Substituting the Sub ID

cols = subs_df.columns.tolist()
cols = cols[-1:] + cols[:-1] #Rearranging the Columns
cols = cols[-1:] + cols[:-1]
subs_df = subs_df[cols]

user_df = subs_df[['User_ID', 'gender', 'age', 'diagnosis_condition', 'diagnosis_severity']]
subs_df = subs_df.drop(['gender', 'age', 'diagnosis_condition', 'diagnosis_severity'], axis =1)

del subs_df['checkin_score'] #since there are no values inside that column, we will drop it
```

In [8]:
```
#Structure of the new User dataset
user_df.head()
```
Out[8]:

	User_ID	gender	age	diagnosis_condition	diagnosis_severity
0	1	female	34	Acne - Comedonica (L70.0)	NaN
1	1	female	34	Acne - Conglobata (L70.1)	NaN
2	1	female	34	Acne - Conglobata (L70.1)	NaN
3	1	female	35	Melasma (L81.1)	NaN
4	2	female	26	Slow-Aging	NaN

In [9]:
```
#turn dates into datetime-types
subs_df['purchase_date'] = subs_df['purchase_date'].astype('datetime64')
subs_df['subscription_start_date'] = subs_df['subscription_start_date'].astype('datetime64')
subs_df['subscription_end_date'] = subs_df['subscription_end_date'].astype('datetime64')
subs_df['cancellation_date'] = subs_df['cancellation_date'].astype('datetime64')
subs_df['user_created_date'] = subs_df['user_created_date'].astype('datetime64')
subs_df['checkin_date'] = subs_df['checkin_date'].astype('datetime64')
```

In [10]:
```
subs_df.head()
```
Out[10]:

	User_ID	Subscription_ID	first_voucher_type	purchase_date	subscription_start_date	subscription_end_date	subscription_interval	canc
0	1	1	NaN	2021-11-01	2021-11-01	2021-12-01	30	
1	1	2	NaN	2021-12-01	2021-12-01	2022-01-01	31	
2	1	3	NaN	2022-01-01	2022-01-01	2022-02-26	56	
3	1	4	NaN	2022-02-26	2022-02-26	2022-03-26	28	
4	2	5	marketing 2x20	2021-09-29	2021-09-29	2021-10-29	30	

Creating a date list for our later graphics

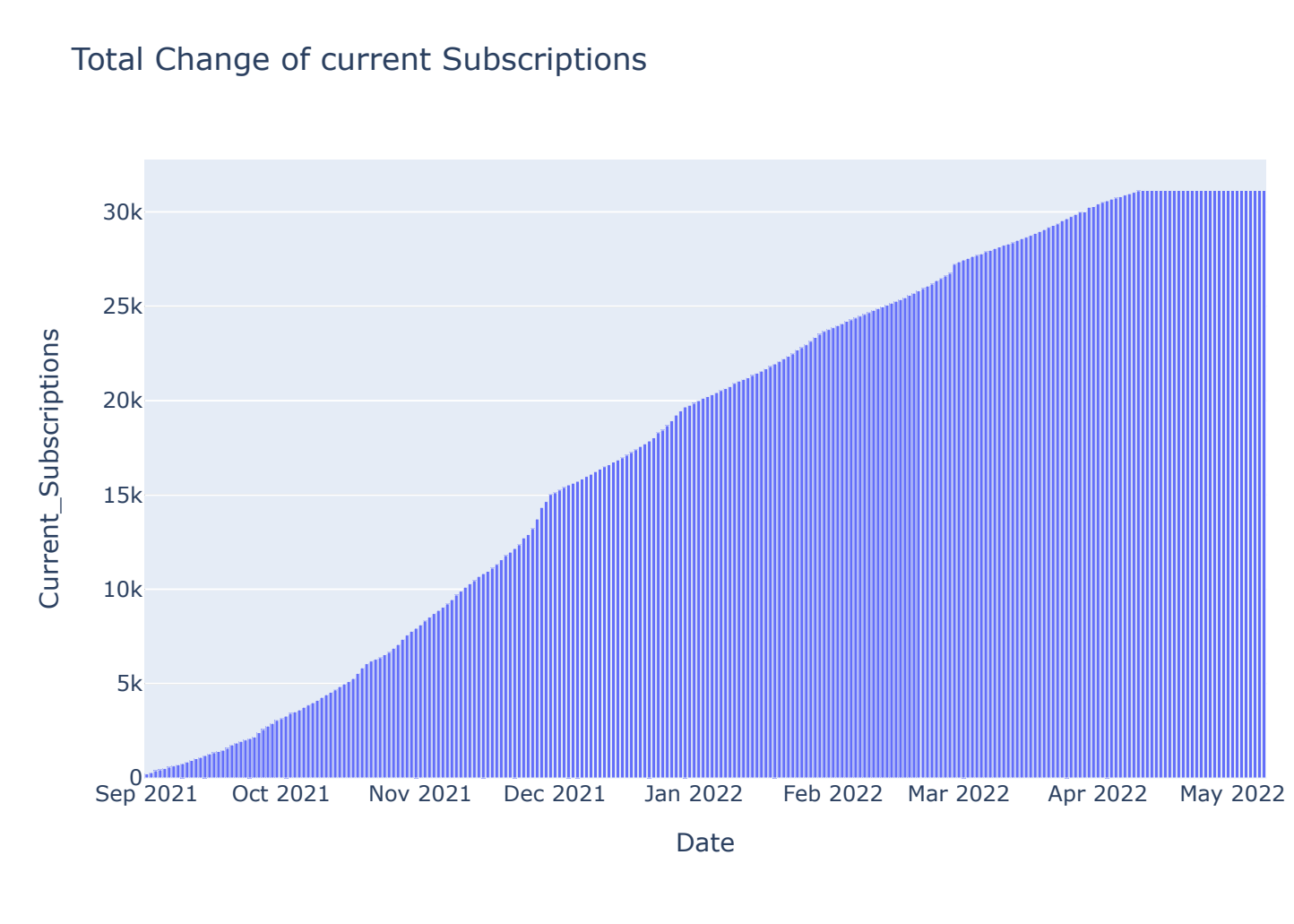
In [11]:
```
date_list = []
i  = 0 #i is the amount of days after the initial date
while i < 250:
    date_list.append(pd.Timestamp(2021, 9, 1) + timedelta(days=i))
    i = i + 1
```

Getting the Change of Total User Count

To get the daily change of the total user count we have to look at the columns "subscription_start_date" to see when a new user has been added and we have to look at the "cancellation_date" to see when a user left.

In [12]:
```
current_sub_list = []
for days in date_list:
    dummy_df = subs_df.loc[((subs_df['subscription_start_date'] <= days) & (subs_df['cancellation_date'
] > days)) | ((subs_df['subscription_start_date'] <= days) & (subs_df['cancellation_date'].isnull()))]
    current_sub_list.append(len(dummy_df))
```

In [13]:
```
#Turning the lists into df
total_df = pd.DataFrame({'Date' : date_list,
                         'Current_Subscriptions' : current_sub_list})

fig = px.bar(total_df, x='Date', y='Current_Subscriptions', title='Total Change of current Subscription
s')
print(fig.show())
```




```
None
```

Present the acquisition, churn and activity

Present the acquisition, churn and activity of users over time, with reference to %, deltas, and total counts.

In [25]:
```
#declaring lists
total_churn_list = []           #List for the churn with total numbers
percent_churn_list = []         #List for the churn in percent
delta_churn_list = []           #List for the churn with the delta
total_acqu_list = []            #List for the acquisition with total numbers
percent_acqu_list = []          #List for the acquisition in percent
delta_acqu_list = []            #List for the acquisition with the delta
percent_acti_list = []          #List for the activity in percent
delta_acti_list = []            #List for the activity with the delta
```

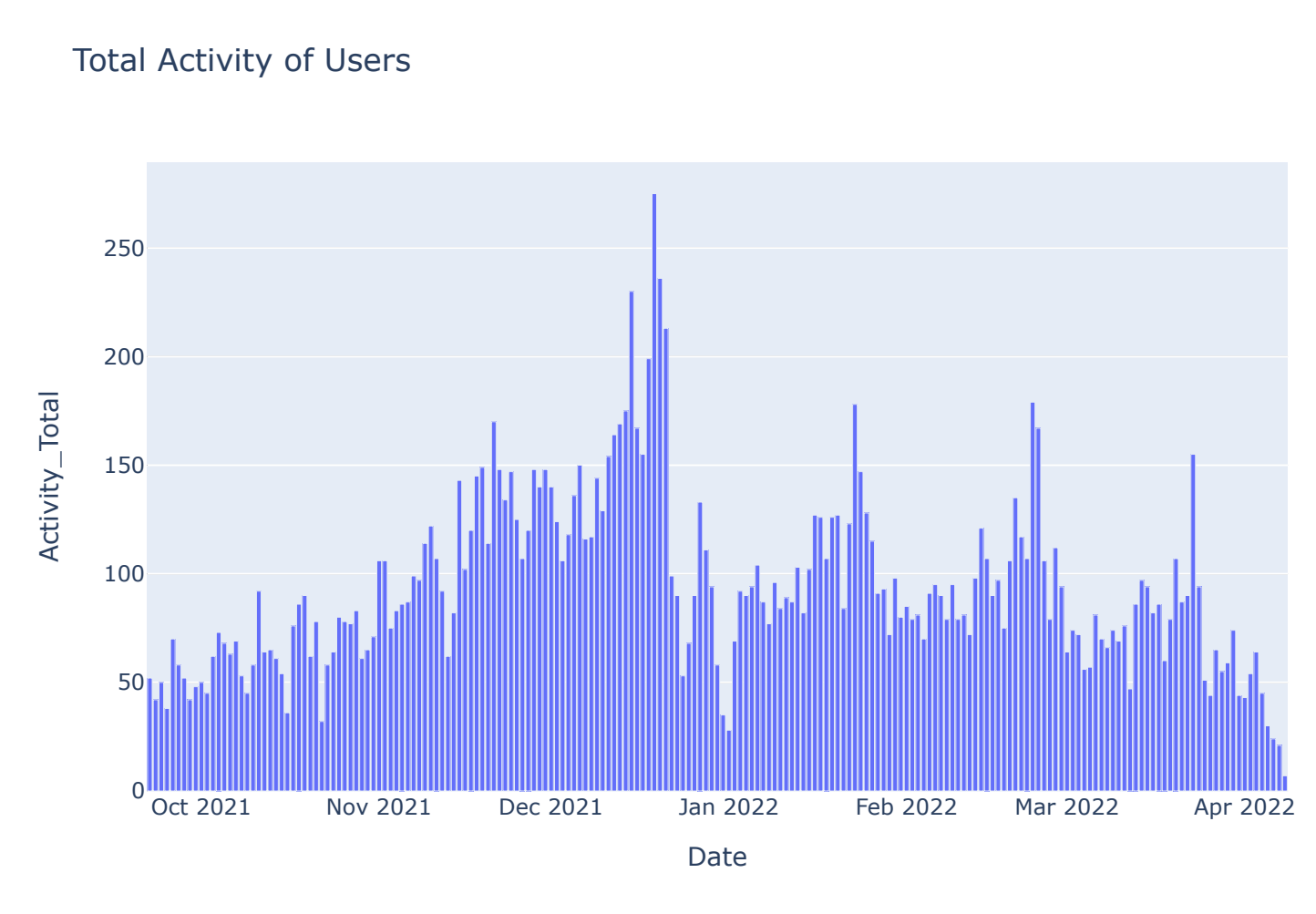
For the activity of the users we will look at the frequency of the check-ins on a daily basis. The delta and % orientate on the previous value.

In [26]:
```
activity_df = subs_df['checkin_date'].value_counts().sort_index().to_frame()
activity_date = activity_df.index.tolist()
activity_count = activity_df['checkin_date'].tolist()

count = 0
while count < (len(activity_count)-1):
    perc = (1 - (activity_count[count+1]/activity_count[count])) * (-1)
    percent_acti_list.append(perc)

    delta = activity_count[count+1] - activity_count[count]
    delta_acti_list.append(delta)

    count = count + 1

#Removing the last date since can't calculate the % and delta for the last value
perc_and_delta_date = activity_date[:-1]
```

In [27]:
```
#Turning the lists into df
acti_total_df = pd.DataFrame({'Date' : activity_date,
                              'Activity_Total' : activity_count})

acti_perdel_df = pd.DataFrame({'Date' : perc_and_delta_date,
                               'Activity_in_Percent' : percent_acti_list,
                               'Activity_in_Delta' : delta_acti_list })
```

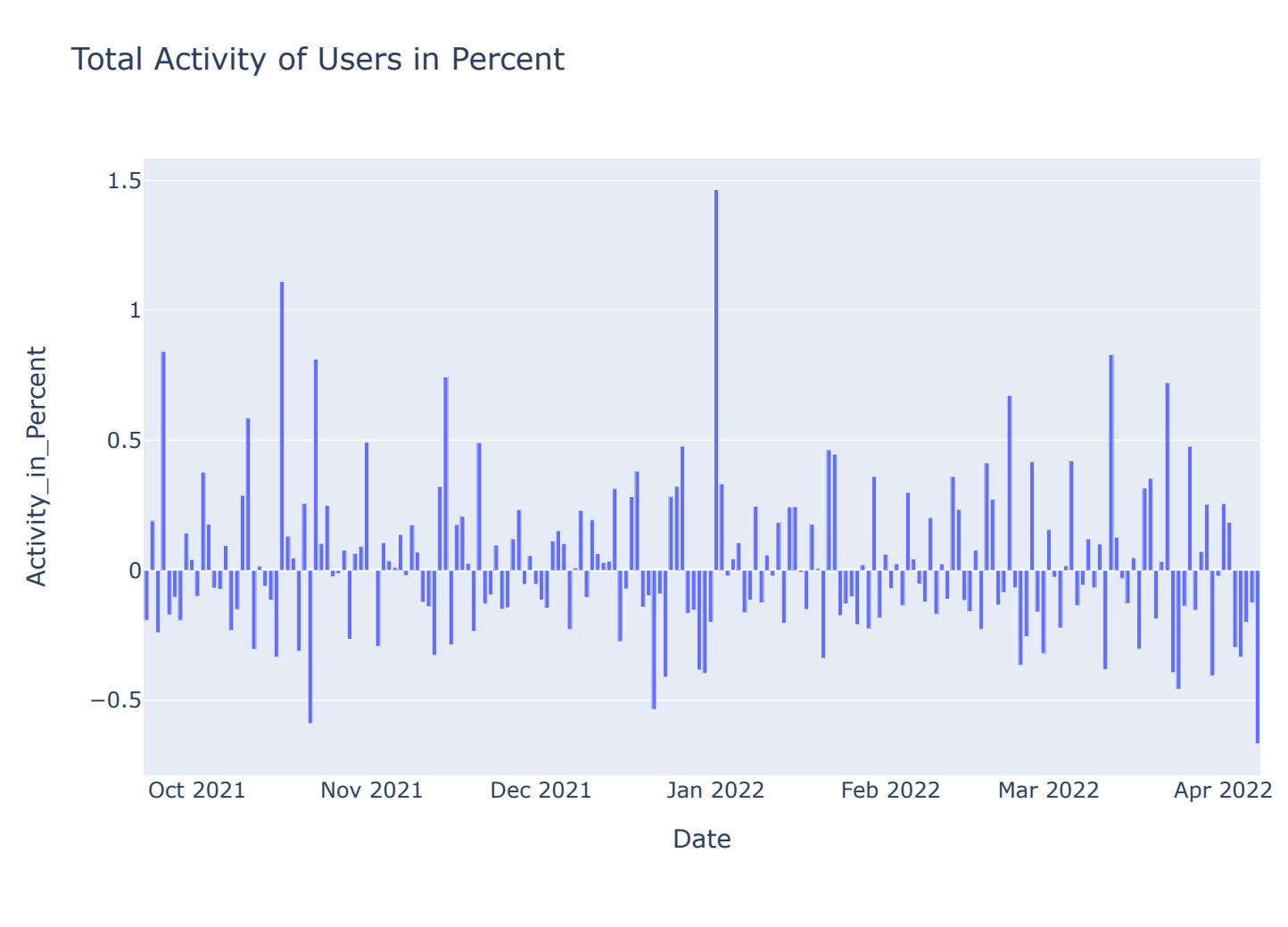
Total Activity

In [29]:
```
fig = px.bar(acti_total_df, x='Date', y='Activity_Total', title='Total Activity of Users')
fig.show()
```



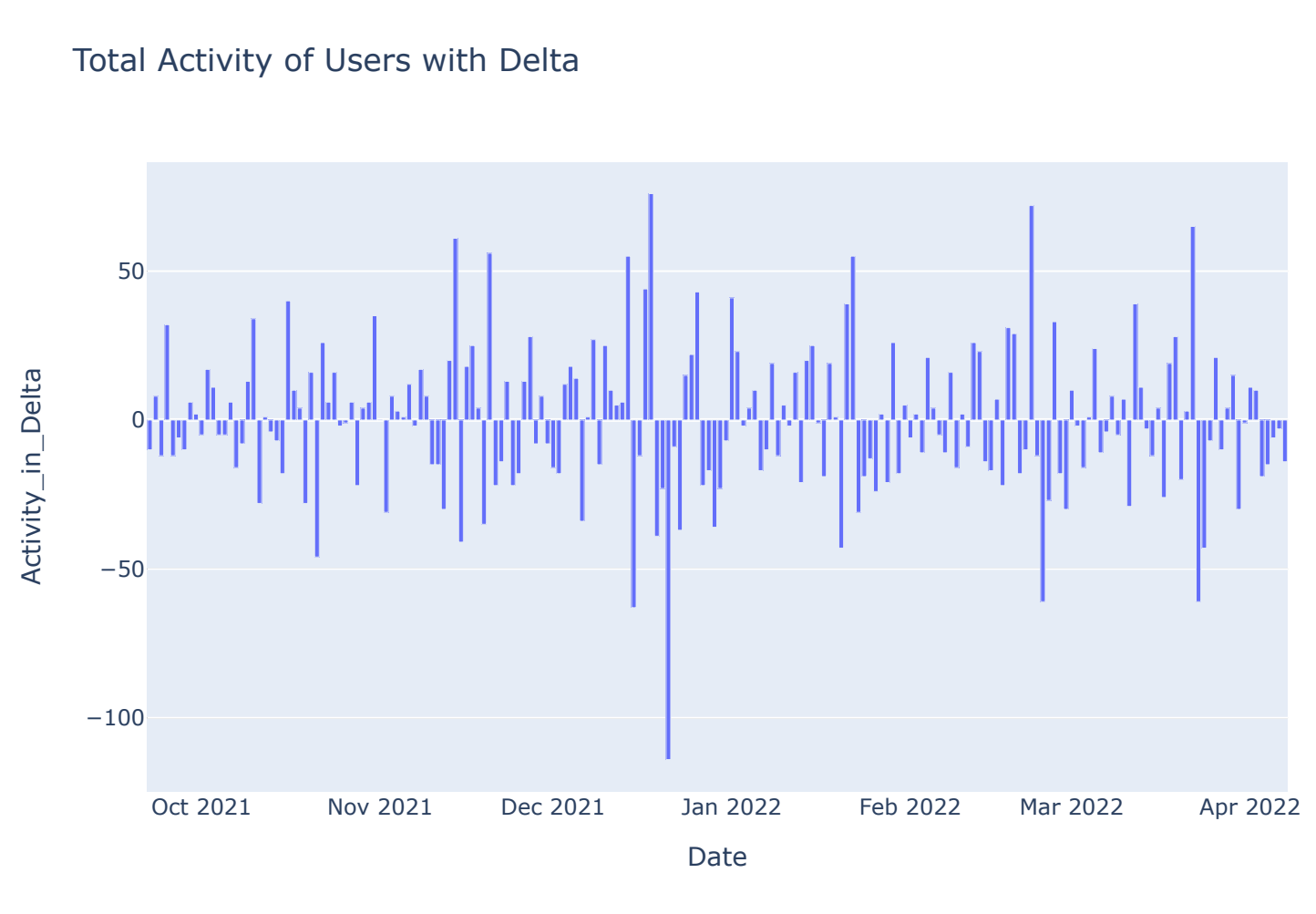
Activity in Percent

In [30]:
```
fig = px.bar(acti_perdel_df, x='Date', y='Activity_in_Percent',title='Total Activity of Users in Percen
t')
fig.show()
```



Activity with the Delta

In [32]:
```
fig = px.bar(acti_perdel_df, x='Date', y='Activity_in_Delta', title='Total Activity of Users with Delt
a')
fig.show()
```

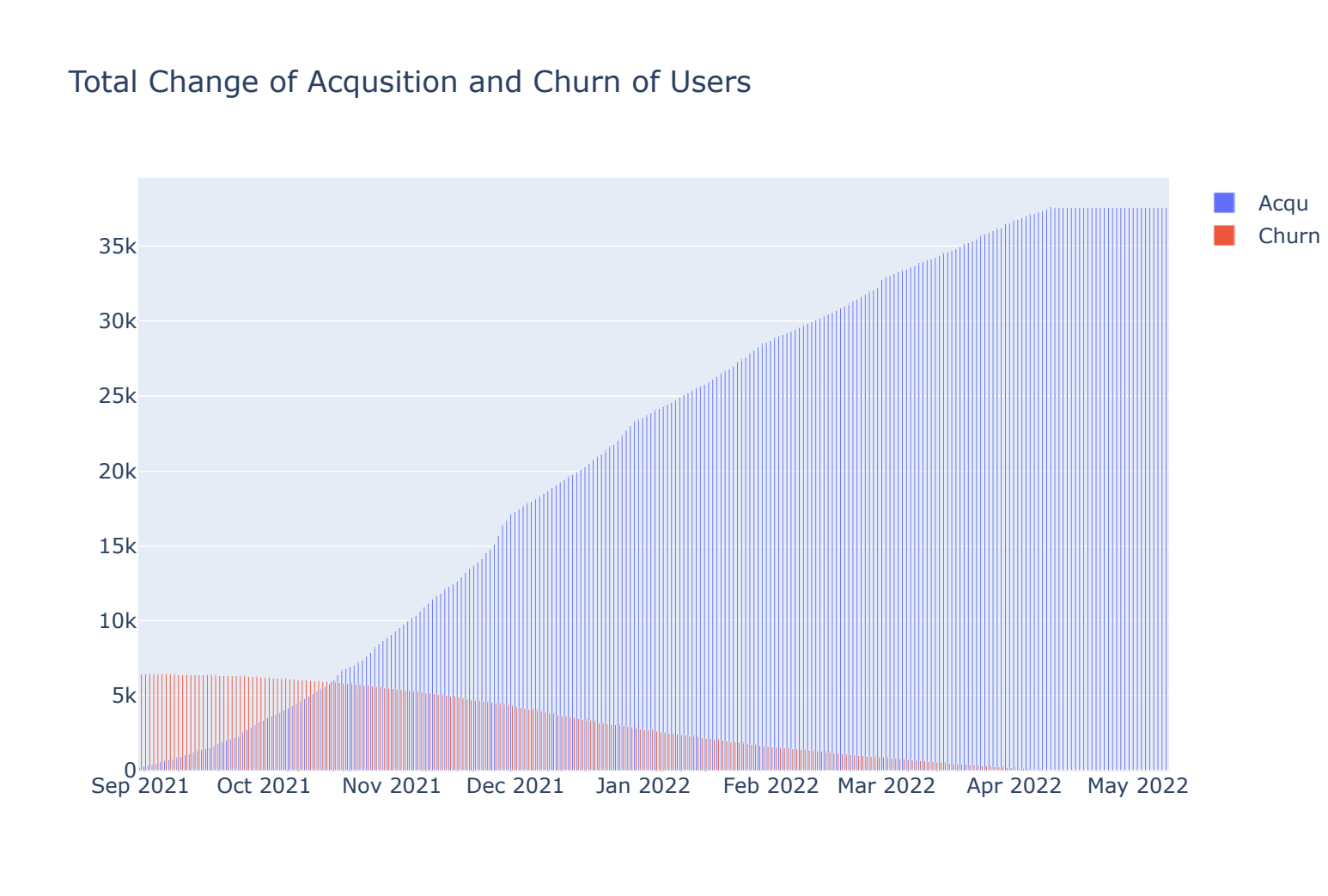


In [33]:
```
#Creating a list with successive days and getting the total user numbers
for days in date_list:
    churn_df = subs_df[subs_df['cancellation_date'] >= days]
    acquisition_df = subs_df[subs_df['subscription_start_date'] <= days]
    total_churn_list.append(churn_df['cancellation_date'].count())
    total_acqu_list.append(acquisition_df['subscription_start_date'].count())
```

In [34]:
```
#calculating the deltas and the change in percent
count = 0
while count < (len(total_churn_list)-1):
    perc = (1 - (total_churn_list[count+1]/total_churn_list[count])) * (-1)
    percent_churn_list.append(perc)

    delta = total_churn_list[count+1] - total_churn_list[count]
    delta_churn_list.append(delta)

    aperc = (1 - (total_acqu_list[count+1]/total_acqu_list[count])) * (-1)
    percent_acqu_list.append(aperc)

    adelta = total_acqu_list[count+1] - total_acqu_list[count]
    delta_acqu_list.append(adelta)

    count = count + 1

#Removing the last date since can't calculate the % and delta for the last value
perc_and_delta_date = date_list[:-1]
```
```
<ipython-input-34-add67c9ad8fe>:4: RuntimeWarning:

invalid value encountered in long_scalars
```

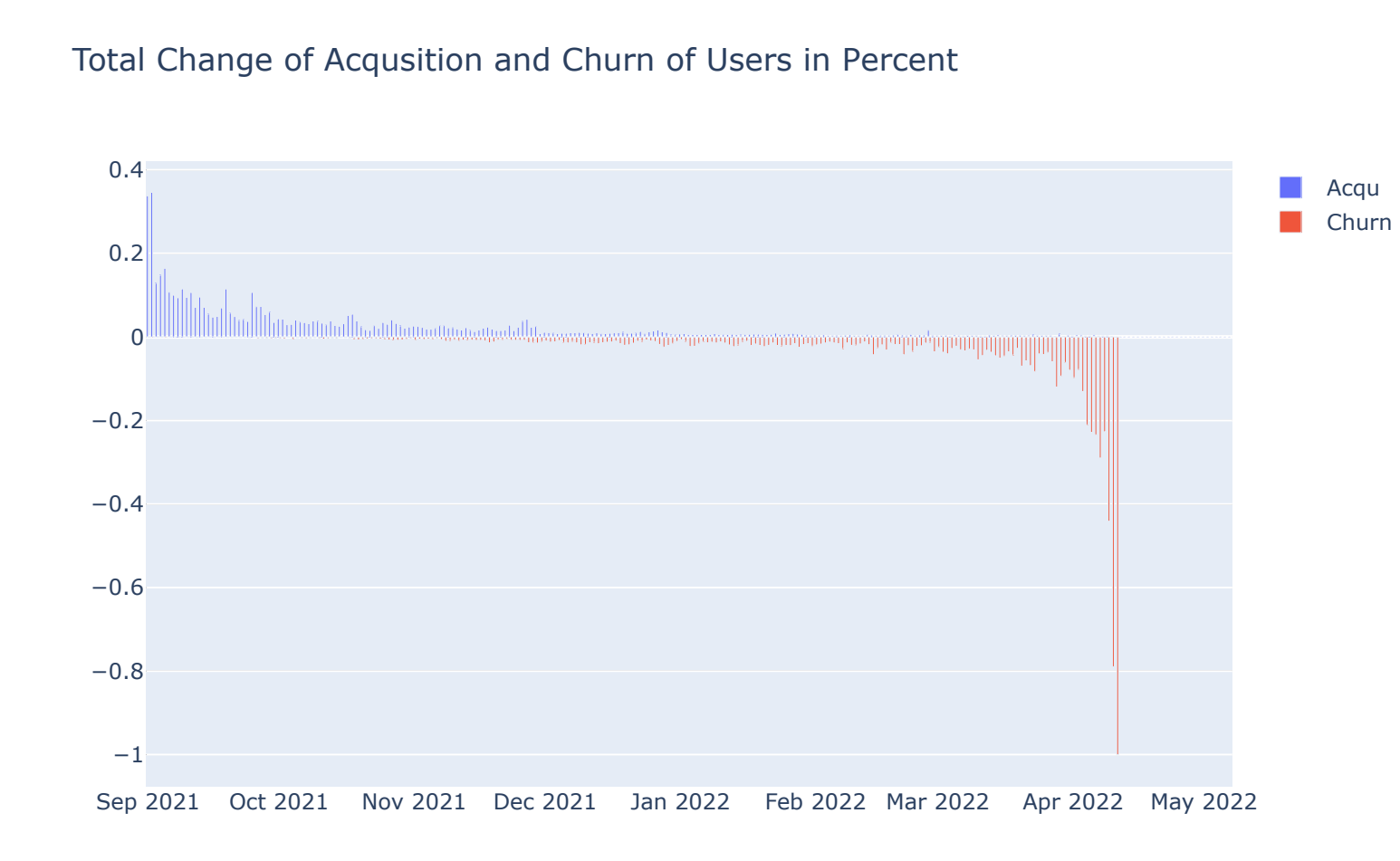
Total Change of Acqusition and Churn

In [35]:
```
fig = go.Figure(data=[
    go.Bar(name='Acqu', x=date_list, y= total_acqu_list),
    go.Bar(name='Churn', x=date_list, y= total_churn_list)
])
# Change the bar mode
fig.update_layout(barmode='group', title='Total Change of Acqusition and Churn of Users')
fig.show()
```



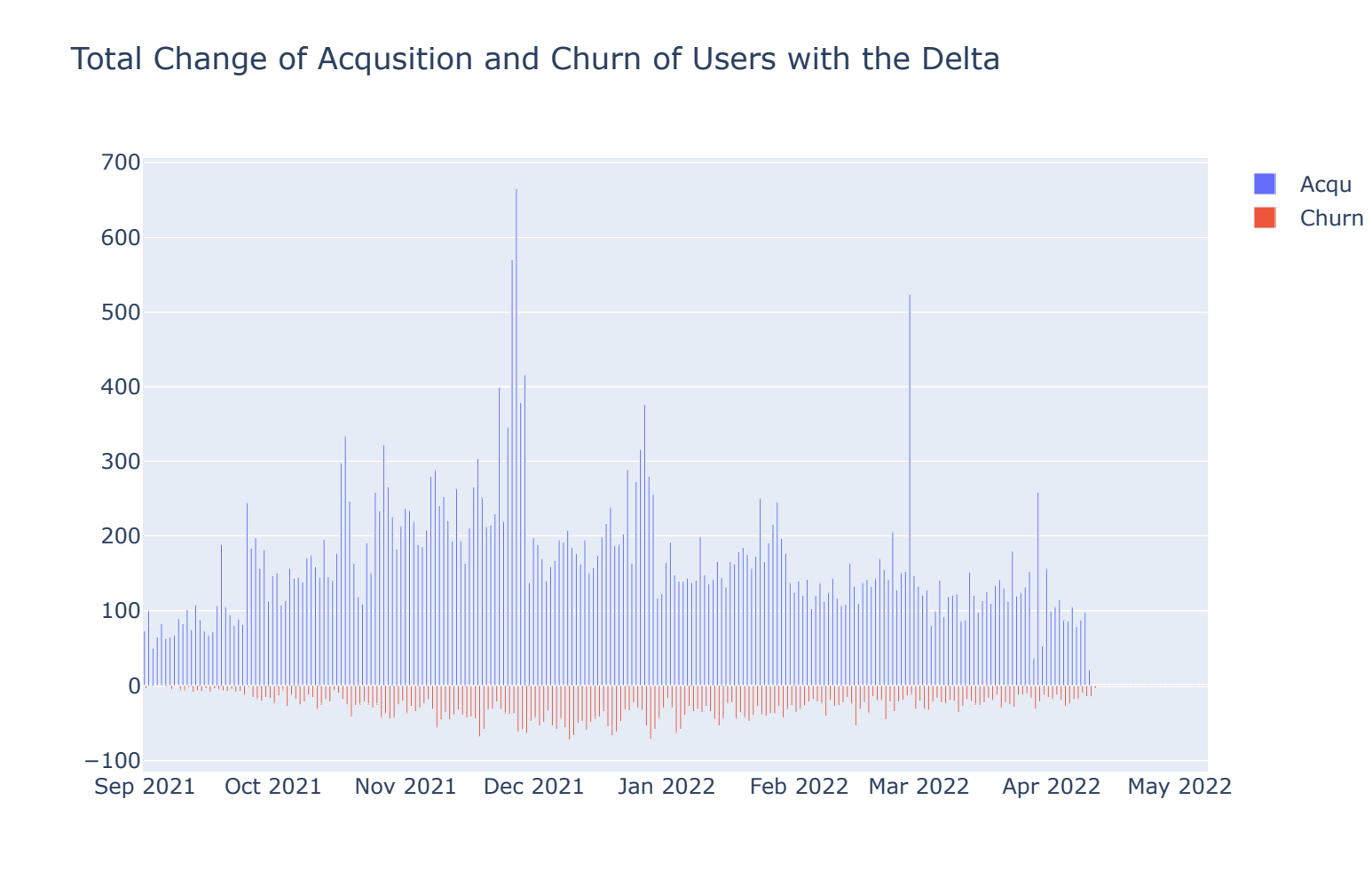
Change of Acqusition and Churn in Percent

In [36]:
```
fig = go.Figure(data=[
    go.Bar(name='Acqu', x=perc_and_delta_date, y= percent_acqu_list),
    go.Bar(name='Churn', x=perc_and_delta_date, y= percent_churn_list)
])
# Change the bar mode
fig.update_layout(barmode='group', title='Total Change of Acqusition and Churn of Users in Percent')
fig.show()
```



Change of Acqusition and Churn with the Delta

In [37]:
```
fig = go.Figure(data=[
    go.Bar(name='Acqu', x=perc_and_delta_date, y= delta_acqu_list),
    go.Bar(name='Churn', x=perc_and_delta_date, y= delta_churn_list)
])
# Change the bar mode
fig.update_layout(barmode='group', title='Total Change of Acqusition and Churn of Users with the Delta'
)
fig.show()
```



Recurring Revenue

Present the recurring revenue as it changes over time, with respect to cumulative, and absolute and relative changes over time

In [38]:
```
subs_df[['User_ID','Subscription_ID','purchase_date','subscription_interval', 'cancellation_date','gros
s_price']].head()
```
Out[38]:

	User_ID	Subscription_ID	purchase_date	subscription_interval	cancellation_date	gross_price
0	1	1	2021-11-01	30	NaT	39.0
1	1	2	2021-12-01	31	NaT	49.0
2	1	3	2022-01-01	56	NaT	49.0
3	1	4	2022-02-26	28	2022-03-14	49.0
4	2	5	2021-09-29	30	NaT	39.0

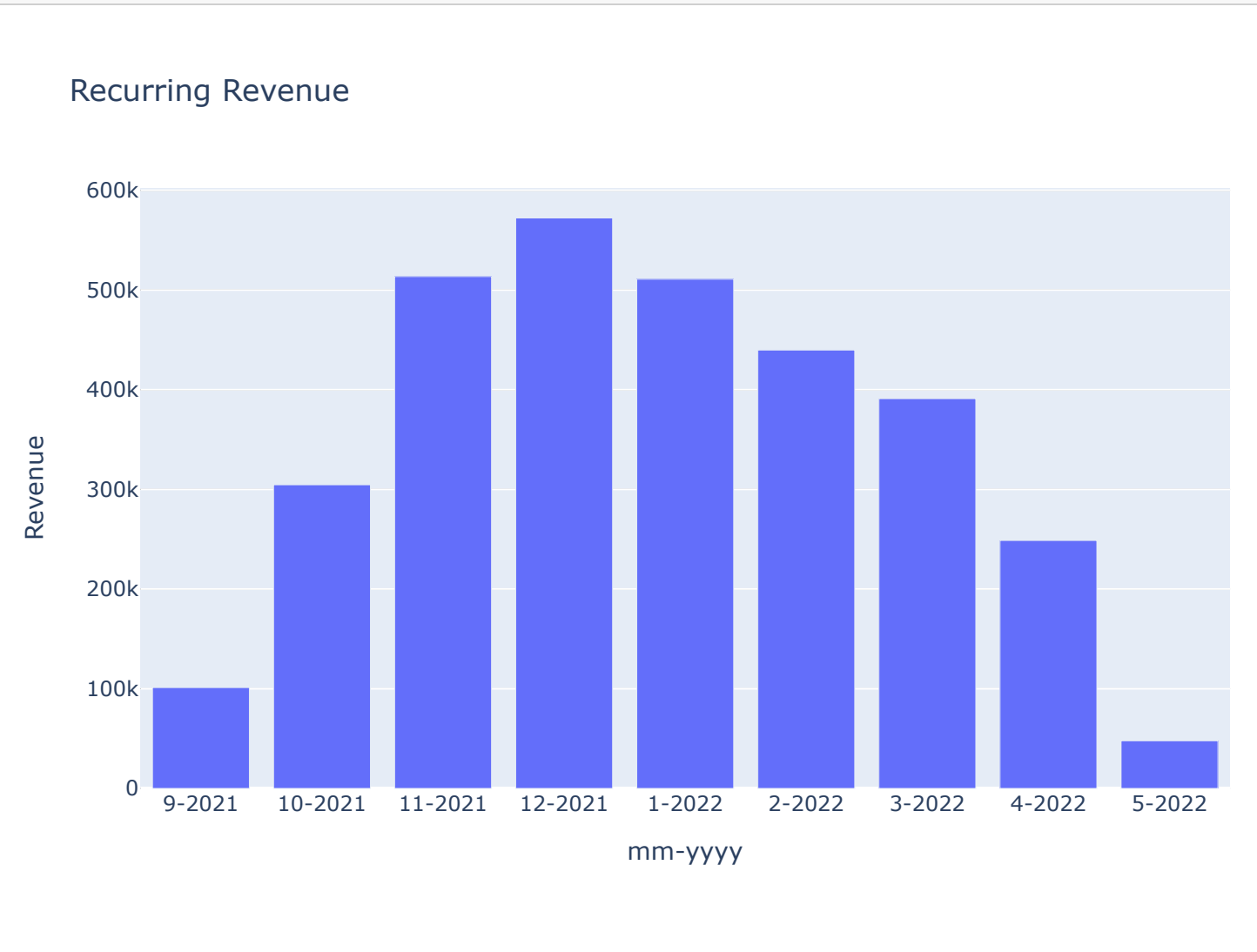
In [39]:
```
interval_list = subs_df['subscription_interval'].value_counts().index.tolist()
```

In [40]:
```
daily_revenue_list = []
for days in date_list:
    #getting the active users
    dummy_df = subs_df.loc[((subs_df['subscription_start_date'] <= days) & (subs_df['cancellation_date'
] >= days)) | ((subs_df['subscription_start_date'] <= days) & (subs_df['cancellation_date'].isnull()))]

    #getting enrollment fee
    enrollment_df = dummy_df.loc[(dummy_df['subscription_start_date'] == days)]
    daily_income = enrollment_df['gross_price'].sum()

    for interval in interval_list:
        sub_fee_df = subs_df.loc[(subs_df['subscription_interval'] == interval)]
        sub_fee_df = sub_fee_df.loc[(sub_fee_df['subscription_start_date'] + timedelta(days = interval
))== days]
        daily_income = daily_income + sub_fee_df['gross_price'].sum()

    daily_revenue_list.append(daily_income)
```

In [41]:
```
daily_revenue_df = pd.DataFrame(
    {'Dates': date_list,
     'Revenue': daily_revenue_list
    })

daily_revenue_df['mm'] = pd.to_datetime(daily_revenue_df['Dates']).dt.month
daily_revenue_df['yyyy'] = pd.to_datetime(daily_revenue_df['Dates']).dt.year
```

In [42]:
```
revenue_df = daily_revenue_df.groupby(["mm",'yyyy']).Revenue.sum().reset_index()
revenue_df = revenue_df.sort_values(['yyyy', 'mm'], ascending=[True, True])
revenue_df['mm'] = revenue_df['mm'].astype('str')
revenue_df['yyyy'] = revenue_df['yyyy'].astype('str')
revenue_df['mm-yyyy'] = revenue_df[['mm', 'yyyy']].agg('-'.join, axis=1)
```

In [47]:
```
fig = px.bar(revenue_df, x='mm-yyyy', y='Revenue', title='Recurring Revenue')
fig.show()
```



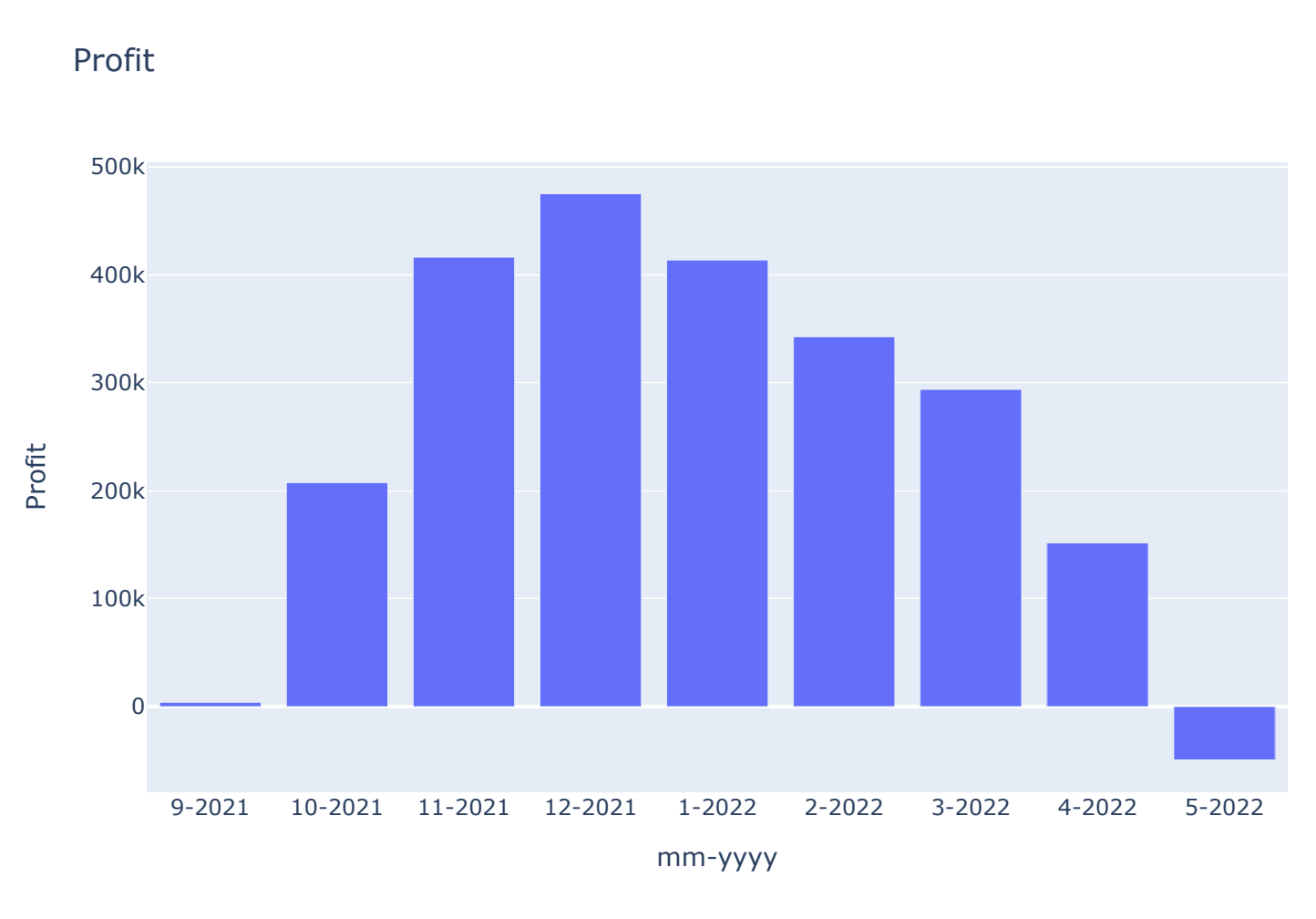
Monthly Marketing Costs

The monthly marketing costs are listed for each marketing channel in the dataset 'spendings_df'. Also we are supposed to add a monthly consultation fee of 20k per month.

In [48]:
```
#Sum up the costs of each marketing channel.
spendings = spendings_df['monthly_total_spend'].sum()
```

In [49]:
```
#Adding the costs to the previous dataset
revenue_df['monthly_total_spend'] = spendings + 20000
revenue_df['Profit'] = revenue_df['Revenue'] - revenue_df['monthly_total_spend'] #Substract marketing c
osts from revenue
```

In [51]:
```
fig = px.bar(revenue_df, x='mm-yyyy', y='Profit', title='Profit')
fig.show()
```



In [ ]: