IMPORTING LIBRARIES

In [1]:
```
import numpy as np
import pandas as pd
from dateutil import parser
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
import time
%matplotlib inline
```

LOADING DATASET

In [2]:
```
data=pd.read_csv('FineTech_appData.csv')
df=pd.DataFrame(data)
```

In [3]:
```
df.head()
```

Out[3]:

	user	first_open	dayofweek	hour	age	screen_list
0	235136	2012-12-27 02:14:51.273	3	02:00:00	23	idscreen,joinscreen,Cycle,product_review,ScanP...
1	333588	2012-12-02 01:16:00.905	6	01:00:00	24	joinscreen,product_review,product_review2,Scan...
2	254414	2013-03-19 19:19:09.157	1	19:00:00	23	Splash,Cycle,Loan
3	234192	2013-07-05 16:08:46.354	4	16:00:00	28	product_review,Home,product_review,Loan3,Finan...
4	51549	2013-02-26 18:50:48.661	1	18:00:00	31	idscreen,joinscreen,Cycle,Credit3Container,Sca...

In [4]:
```
df.tail()
```

Out[4]:

	user	first_open	dayofweek	hour	age	screen
49995	222774	2013-05-09 13:46:17.871	3	13:00:00	32	Splash,Home,ScanPreview,VerifyPhone,VerifyS
49996	169179	2013-04-09 00:05:17.823	1	00:00:00	35	Cycle,Splash,Home,RewardsConta
49997	302367	2013-02-20 22:41:51.165	2	22:00:00	39	joinscreen,product_review,product_review2,Sc
49998	324905	2013-04-28 12:33:04.288	6	12:00:00	27	Cycle,Home,product_review,product_review,pro
49999	27047	2012-12-14 01:22:44.638	4	01:00:00	25	product_review,ScanPreview,VerifyDateOfBirt

In [5]:
```
len(df)
```

Out[5]:
```
50000
```

In [6]:
```
df.shape
```

Out[6]:
```
(50000, 12)
```

In [7]:
```
df.describe()
```

Out[7]:

	user	dayofweek	age	numscreens	minigame	used_premium_featu
count	50000.000000	50000.000000	50000.00000	50000.000000	50000.000000	50000.0000
mean	186889.729900	3.029860	31.72436	21.095900	0.107820	0.1720
std	107768.520361	2.031997	10.80331	15.728812	0.310156	0.3774
min	13.000000	0.000000	16.00000	1.000000	0.000000	0.0000
25%	93526.750000	1.000000	24.00000	10.000000	0.000000	0.0000
50%	187193.500000	3.000000	29.00000	18.000000	0.000000	0.0000
75%	279984.250000	5.000000	37.00000	28.000000	0.000000	0.0000
max	373662.000000	6.000000	101.00000	325.000000	1.000000	1.0000

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype 
---  ------                --------------  ----- 
 0   user                  50000 non-null  int64 
 1   first_open            50000 non-null  object
 2   dayofweek             50000 non-null  int64 
 3   hour                  50000 non-null  object
 4   age                   50000 non-null  int64 
 5   screen_list           50000 non-null  object
 6   numscreens            50000 non-null  int64 
 7   minigame              50000 non-null  int64 
 8   used_premium_feature  50000 non-null  int64 
 9   enrolled              50000 non-null  int64 
 10  enrolled_date         31074 non-null  object
 11  liked                 50000 non-null  int64 
dtypes: int64(8), object(4)
memory usage: 4.6+ MB
```

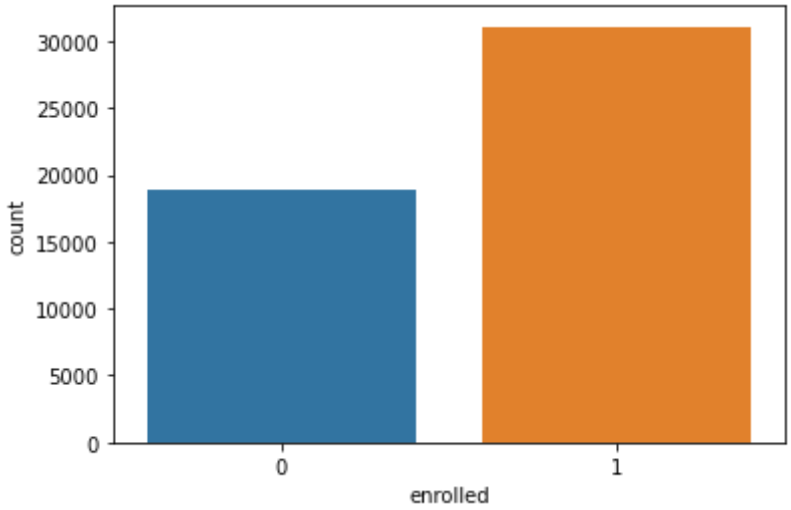
In [9]:
```
df.isnull().sum()
```

Out[9]:
```
user                      0
first_open                0
dayofweek                 0
hour                      0
age                       0
screen_list               0
numscreens                0
minigame                  0
used_premium_feature      0
enrolled                  0
enrolled_date         18926
liked                     0
dtype: int64
```

In [10]:
```
sns.countplot(df['enrolled'],data=df)
```

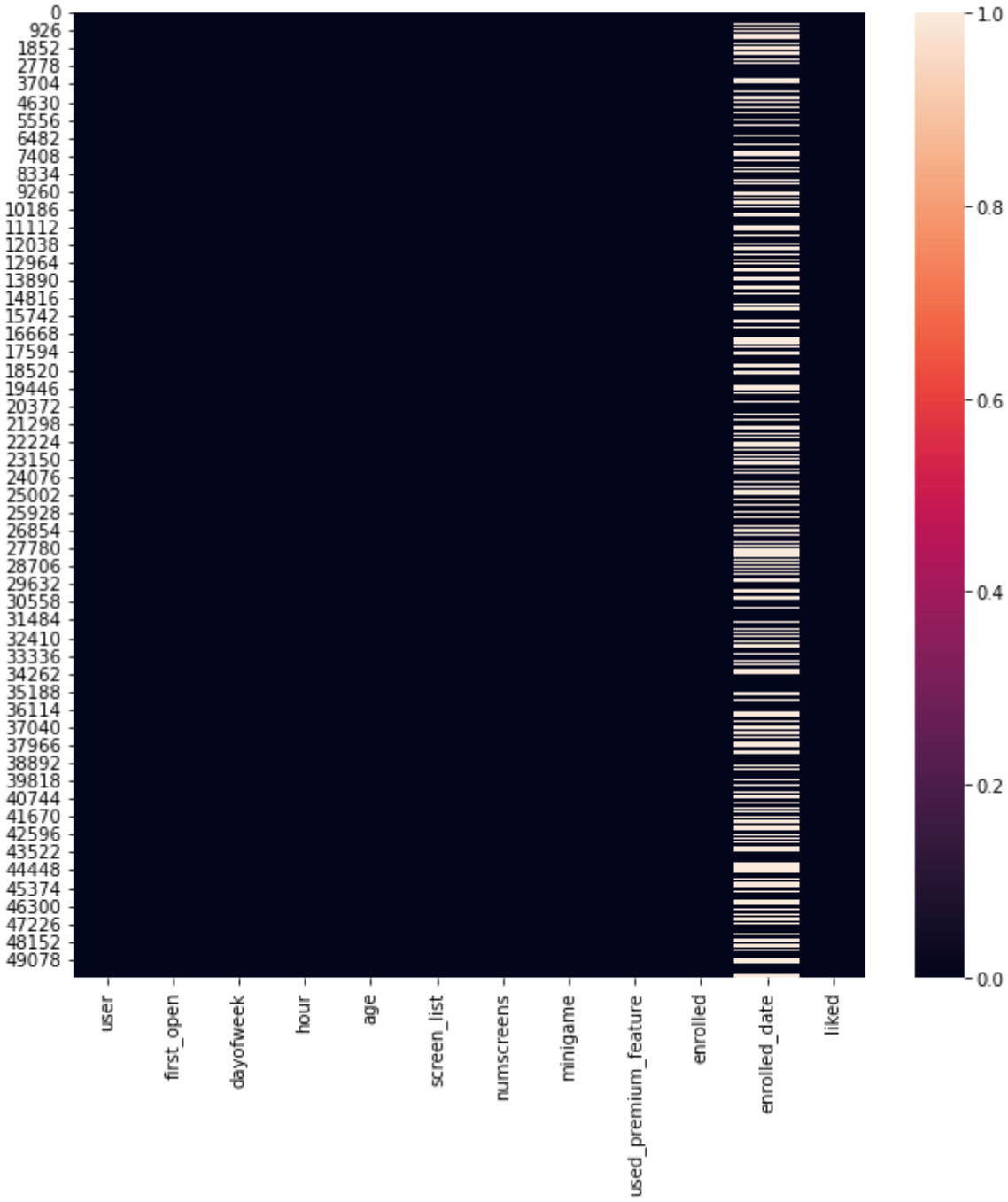
```
C:\Users\kgyan\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only va
lid positional argument will be `data`, and passing other arguments without an exp
licit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[10]: <AxesSubplot:xlabel='enrolled', ylabel='count'>



In [11]:
```
f = plt.figure()
f.set_figwidth(10)
f.set_figheight(10)
sns.heatmap(df.isnull())
```

Out[11]: <AxesSubplot:>



Converting 'hour' column into integer

In [12]:
```
df['hour'].str.slice(1,3).astype(int)
```

Out[12]:
```
0         2
1         1
2        19
3        16
4        18
         ..
49995    13
49996     0
49997    22
49998    12
49999     1
Name: hour, Length: 50000, dtype: int32
```

In [13]:
```
df['hour']=df['hour'].str.slice(1,3).astype(int)
```

In [14]:
```
df.head()
```

Out[14]:

	user	first_open	dayofweek	hour	age	screen_list	nu
0	235136	2012-12-27 02:14:51.273	3	2	23	idscreen,joinscreen,Cycle,product_review,ScanP...	
1	333588	2012-12-02 01:16:00.905	6	1	24	joinscreen,product_review,product_review2,Scan...	
2	254414	2013-03-19 19:19:09.157	1	19	23	Splash,Cycle,Loan	
3	234192	2013-07-05 16:08:46.354	4	16	28	product_review,Home,product_review,Loan3,Finan...	
4	51549	2013-02-26 18:50:48.661	1	18	31	idscreen,joinscreen,Cycle,Credit3Container,Sca...	

CREATING A COPY OF THE DATASET AND REMOVING THE COLUNMS THAT ARE NOT NEEDED

In [15]:
```
new_df=df.copy().drop(columns=['user','first_open','screen_list','enrolled','enroll
```

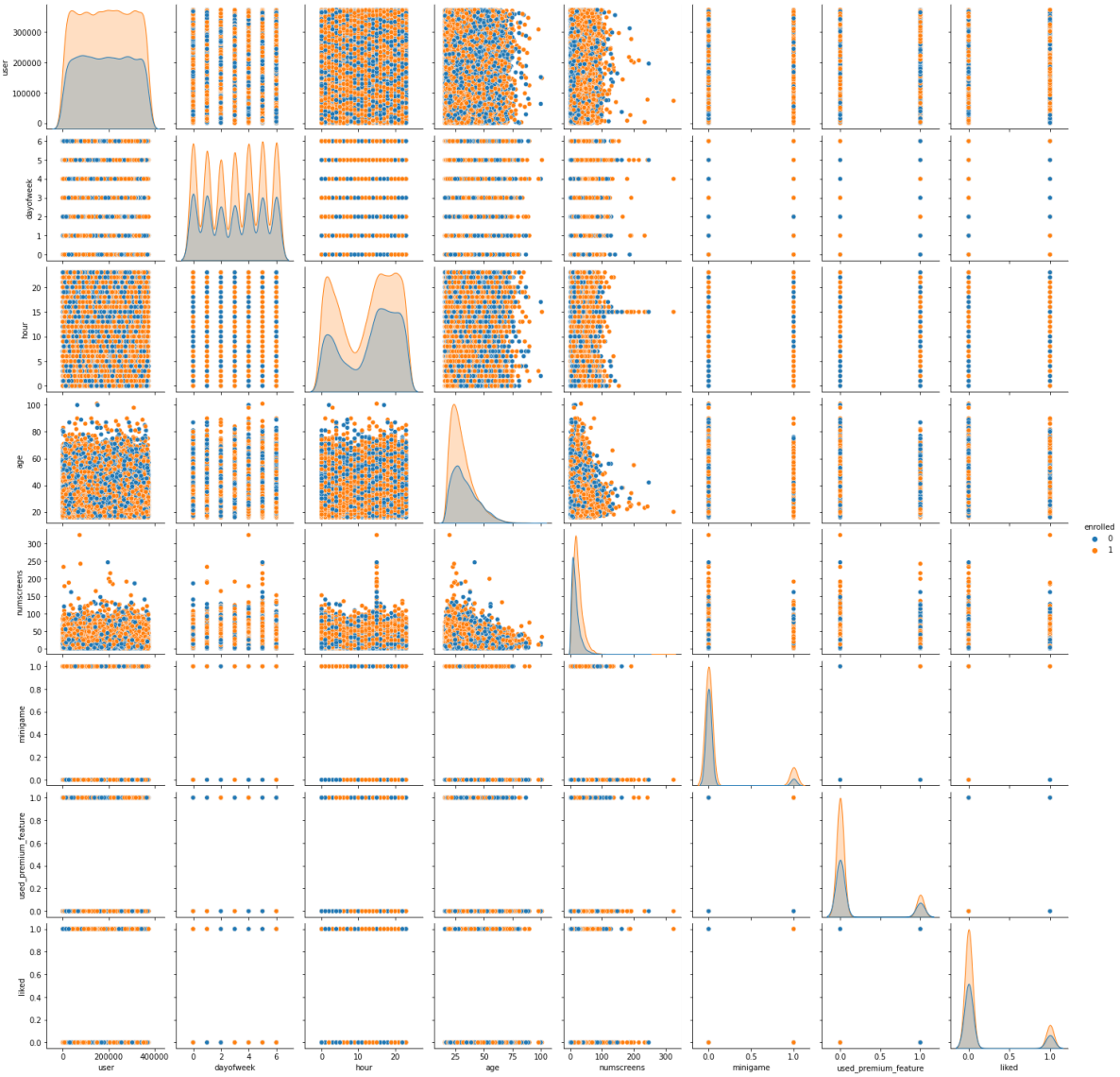
In [16]:
```
new_df.head()
```

Out[16]:

	dayofweek	hour	age	numscreens	minigame	used_premium_feature	liked
0	3	2	23	15	0	0	0
1	6	1	24	13	0	0	0
2	1	19	23	3	0	1	1
3	4	16	28	40	0	0	0
4	1	18	31	32	0	0	1

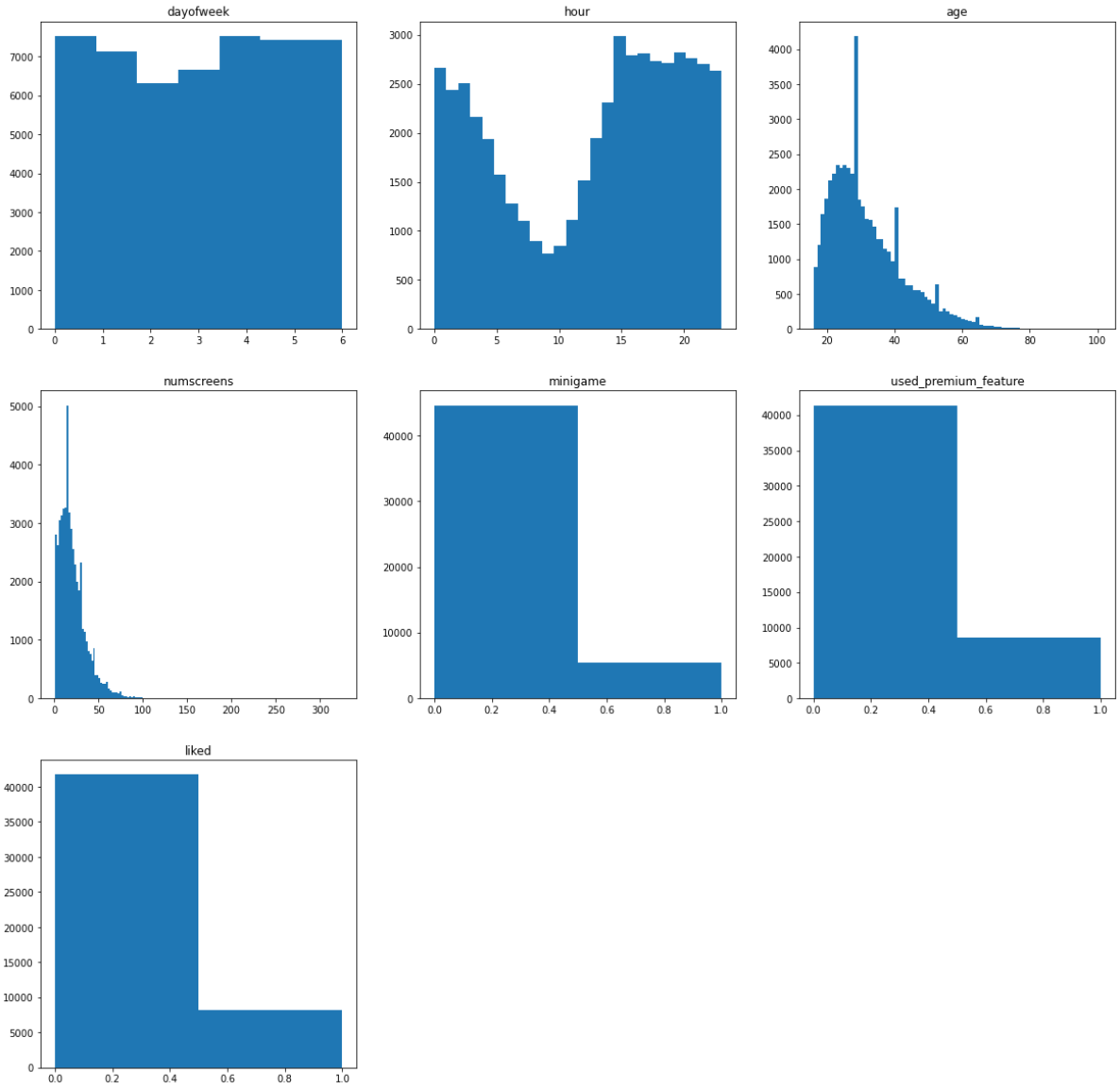
In [17]:
```
sns.pairplot(df,hue='enrolled')
```

Out[17]:
```
<seaborn.axisgrid.PairGrid at 0x20d2efa4250>
```



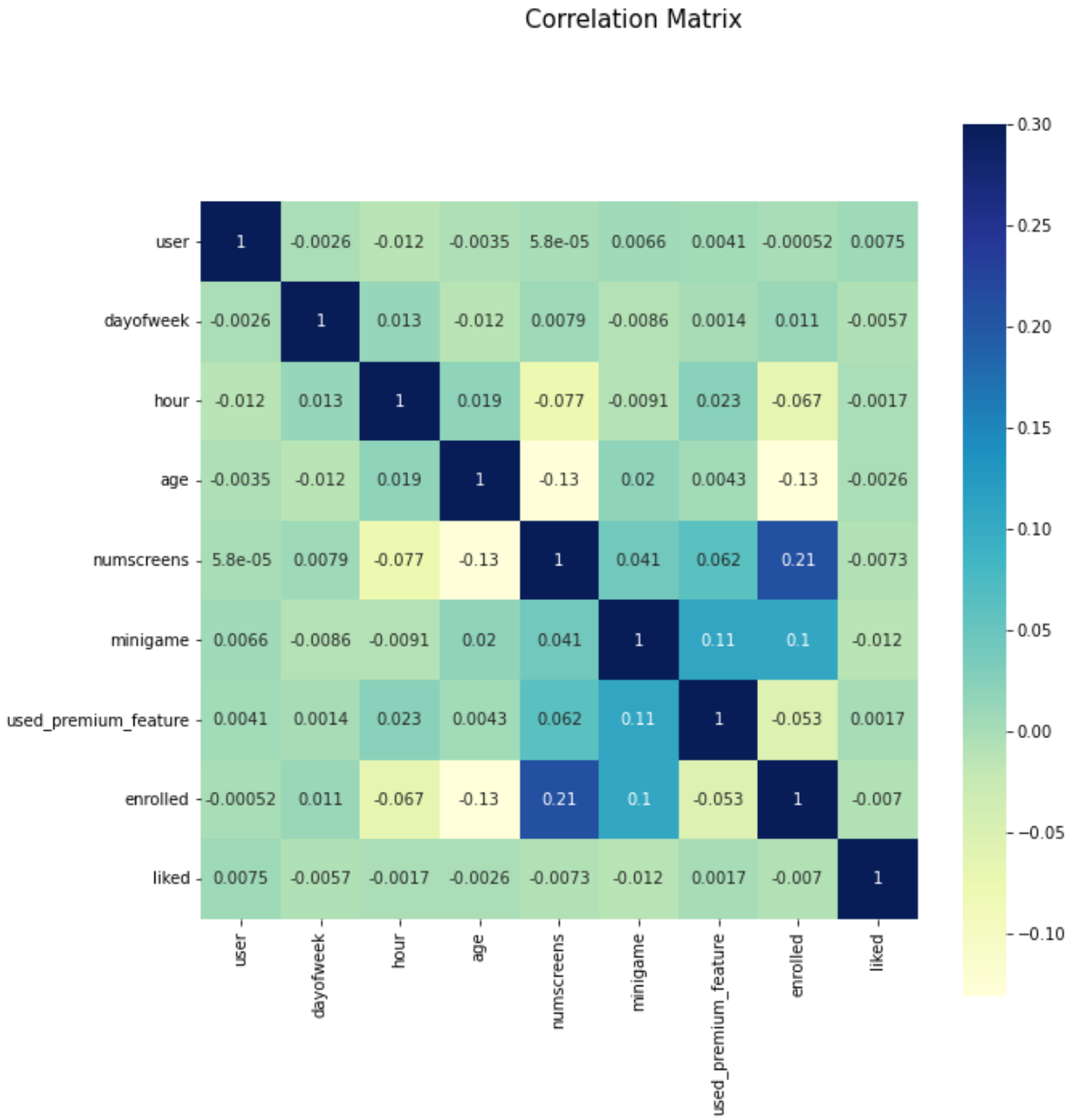
In [18]:
```
plt.figure(figsize=(20,20))
plt.suptitle('Histograms',fontsize=20)
for i in range(1,new_df.shape[1]+1):
    plt.subplot(3,3,i)
    f=plt.gca()
    f.set_title(new_df.columns.values[i-1])
    #setting bins for histogram
    vals=np.size(new_df.iloc[:,i-1].unique())

    plt.hist(new_df.iloc[:,i-1],bins=vals)
```

Histograms



In [19]:
```
#original dataset
plt.figure(figsize=(10,10))
plt.suptitle('Correlation Matrix',fontsize=15)
sns.heatmap(df.corr(),annot=True,square=True,vmax=.3,cmap="YlGnBu")
```

Out[19]:
```
<AxesSubplot:>
```



In [20]:
```
new_df.corr()
```

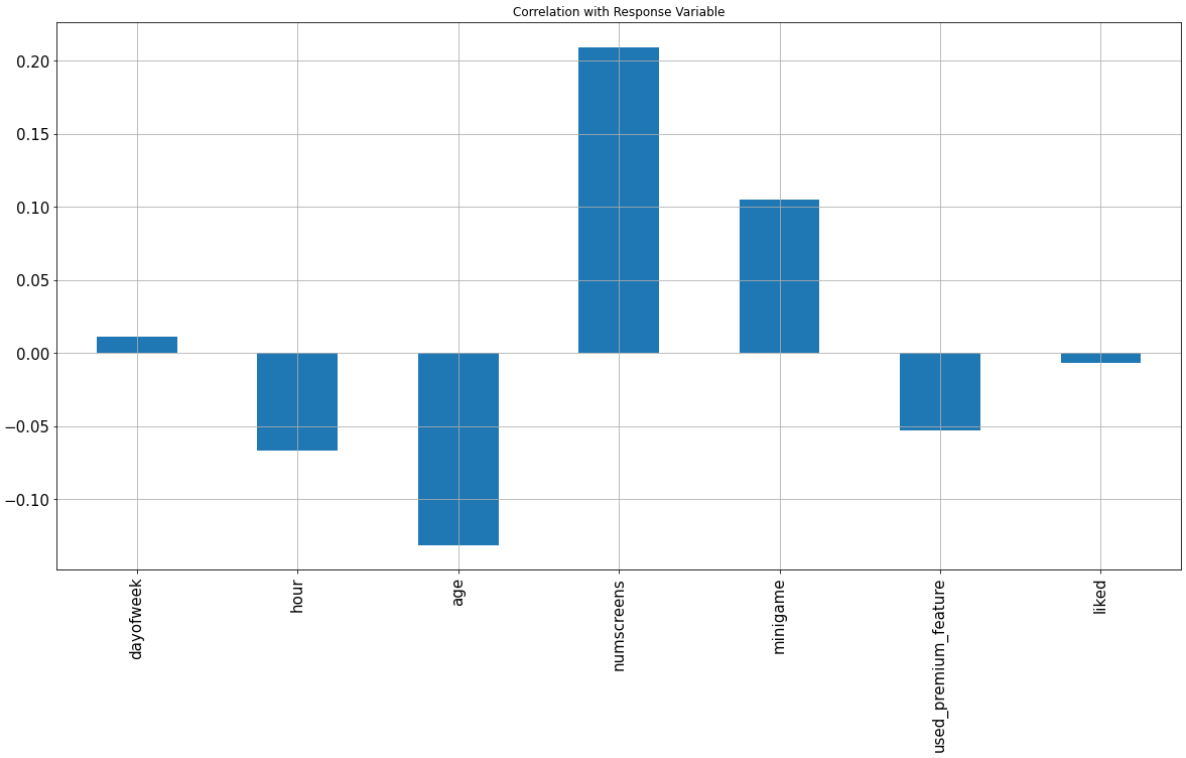
Out[20]:

	dayofweek	hour	age	numscreens	minigame	used_premium_fe
dayofweek	1.000000	0.013249	-0.012326	0.007925	-0.008631	0.0
hour	0.013249	1.000000	0.018859	-0.076756	-0.009120	0.0
age	-0.012326	0.018859	1.000000	-0.128739	0.019745	0.0
numscreens	0.007925	-0.076756	-0.128739	1.000000	0.041154	0.0
minigame	-0.008631	-0.009120	0.019745	0.041154	1.000000	0.1
used_premium_feature	0.001439	0.022553	0.004301	0.061972	0.108780	1.0
liked	-0.005737	-0.001725	-0.002593	-0.007349	-0.012250	0.0

In [21]:
```
new_df.corrwith(df['enrolled'])
```

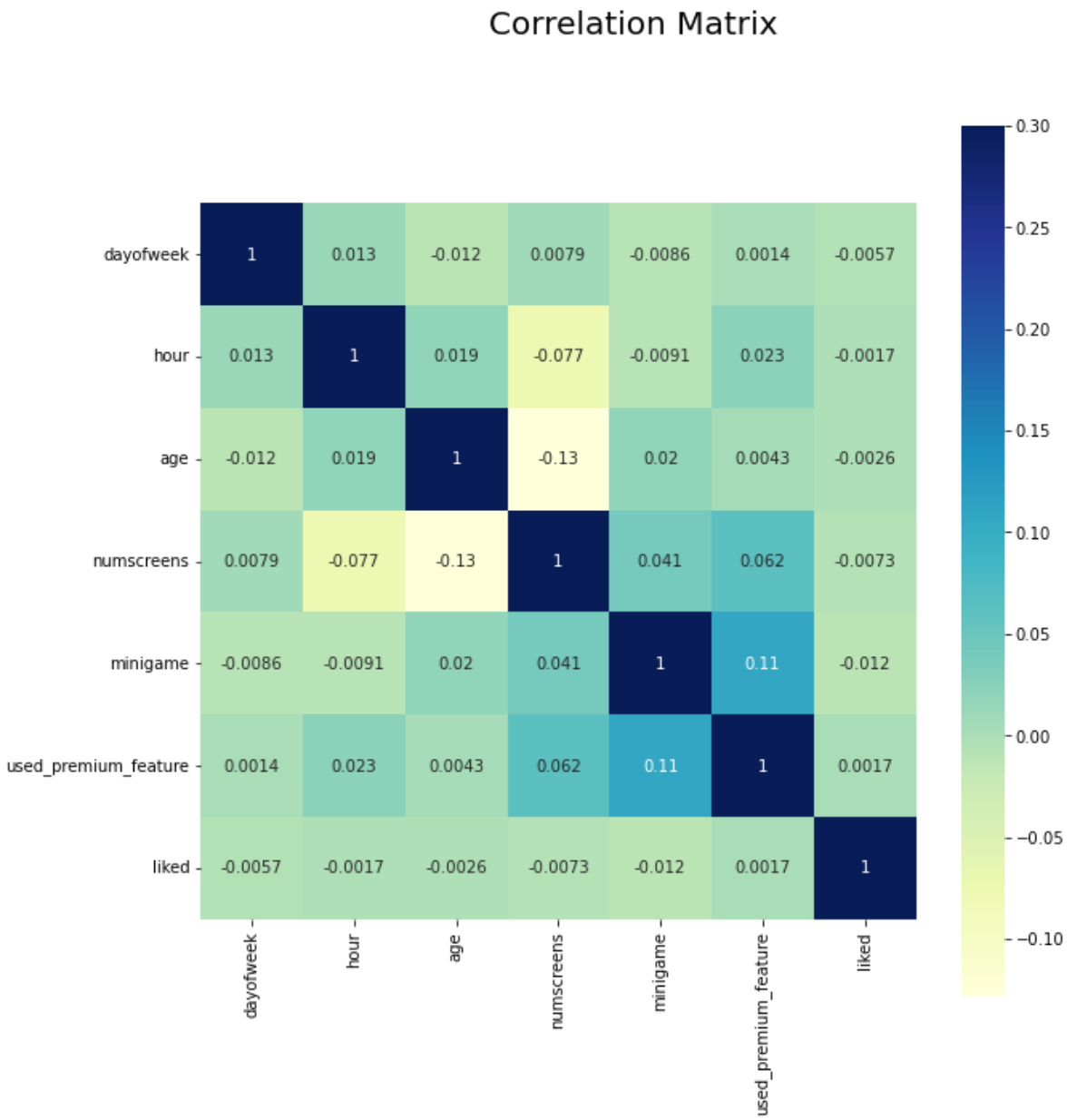
Out[21]:
```
dayofweek              0.011326
hour                  -0.066694
age                   -0.131303
numscreens             0.209457
minigame               0.104979
used_premium_feature  -0.052703
liked                 -0.007022
dtype: float64
```

In [22]:
```
a=new_df.corrwith(df['enrolled']).plot.bar(figsize=(20,10),title='Correlation with
```



In [23]:
```
#copy dataset
plt.figure(figsize=(10,10))
plt.suptitle('Correlation Matrix',fontsize=20)
sns.heatmap(new_df.corr(),annot=True,square=True,vmax=.3,cmap="YlGnBu")
```

Out[23]:
```
<AxesSubplot:>
```



FEATURE ENGINEERING

In [24]:
```
#converting the object datatype in datatime format
df['first_open']=[parser.parse(row_data) for row_data in df['first_open']]
df['enrolled_date']=[parser.parse(row_data) if isinstance(row_data,str) else row_da
```

In [25]:
```
df.info()
```

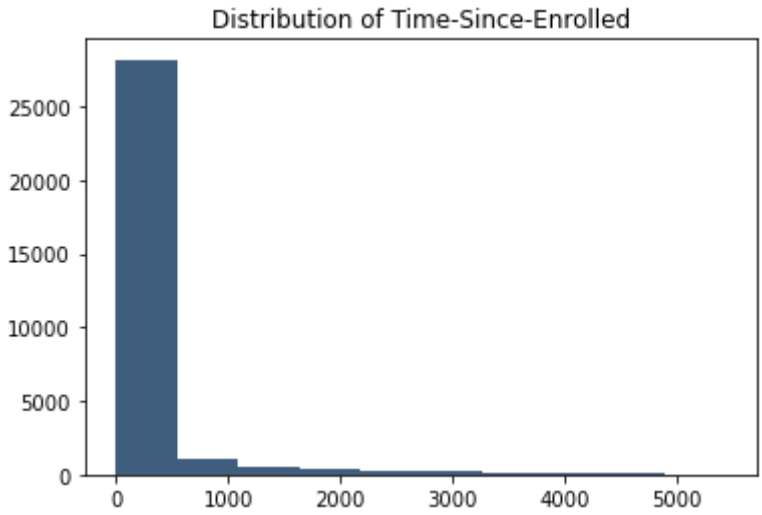
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   user                  50000 non-null  int64
 1   first_open            50000 non-null  datetime64[ns]
 2   dayofweek             50000 non-null  int64
 3   hour                  50000 non-null  int32
 4   age                   50000 non-null  int64
 5   screen_list           50000 non-null  object
 6   numscreens            50000 non-null  int64
 7   minigame              50000 non-null  int64
 8   used_premium_feature  50000 non-null  int64
 9   enrolled              50000 non-null  int64
 10  enrolled_date         31074 non-null  datetime64[ns]
 11  liked                 50000 non-null  int64
dtypes: datetime64[ns](2), int32(1), int64(8), object(1)
memory usage: 4.4+ MB
```

In [26]:
```
df['difference']=(df['enrolled_date']-df['first_open']).astype('timedelta64[h]')
df['difference']
```

Out[26]:
```
0        NaN
1        NaN
2        NaN
3        0.0
4        0.0
        ...
49995    0.0
49996    NaN
49997    NaN
49998    0.0
49999    NaN
Name: difference, Length: 50000, dtype: float64
```

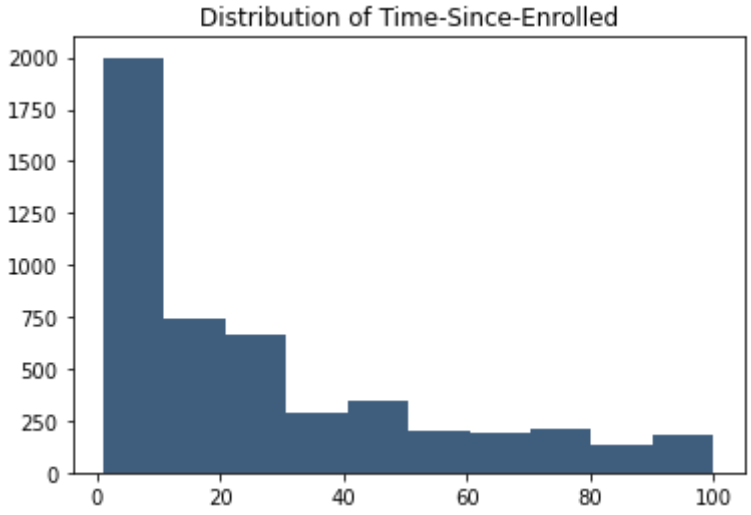
In [27]:
```
plt.hist(df['difference'].dropna(),color='#3F5D7D')
plt.title('Distribution of Time-Since-Enrolled')
```

Out[27]:
```
Text(0.5, 1.0, 'Distribution of Time-Since-Enrolled')
```



In [28]:
```
plt.hist(df['difference'].dropna(),color='#3F5D7D',range=[1,100])
plt.title('Distribution of Time-Since-Enrolled')
```

Out[28]:
```
Text(0.5, 1.0, 'Distribution of Time-Since-Enrolled')
```



In [29]:
```
df.loc[df['difference']>48,'enrolled']=0
```

In [30]:
```
df.tail()
```

Out[30]:

	user	first_open	dayofweek	hour	age	screen_lis
49995	222774	2013-05-09 13:46:17.871	3	13	32	Splash,Home,ScanPreview,VerifyPhone,VerifySSN,.
49996	169179	2013-04-09 00:05:17.823	1	0	35	Cycle,Splash,Home,RewardsContaine
49997	302367	2013-02-20 22:41:51.165	2	22	39	joinscreen,product_review,product_review2,Scan.
49998	324905	2013-04-28 12:33:04.288	6	12	27	Cycle,Home,product_review,product_review,produ.
49999	27047	2012-12-14 01:22:44.638	4	1	25	product_review,ScanPreview,VerifyDateOfBirth,V.

In [31]:
```
df=df.drop(columns=['first_open','enrolled_date'])
```

In [32]:
```
df.head()
```

Out[32]:

	user	dayofweek	hour	age	screen_list	numscreens	m
0	235136	3	2	23	idscreen,joinscreen,Cycle,product_review,ScanP...	15	
1	333588	6	1	24	joinscreen,product_review,product_review2,Scan...	13	
2	254414	1	19	23	Splash,Cycle,Loan	3	
3	234192	4	16	28	product_review,Home,product_review,Loan3,Finan...	40	
4	51549	1	18	31	idscreen,joinscreen,Cycle,Credit3Container,Sca...	32	

In [33]:
```
top_screens=pd.read_csv('top_screens.csv')
df_s=pd.DataFrame(top_screens)
a=df_s['top_screens'].values
a
```

Out[33]:
```
array(['Loan2', 'location', 'Institutions', 'Credit3Container',
       'VerifyPhone', 'BankVerification', 'VerifyDateOfBirth',
       'ProfilePage', 'VerifyCountry', 'Cycle', 'idscreen',
       'Credit3Dashboard', 'Loan3', 'CC1Category', 'Splash', 'Loan',
       'CC1', 'RewardsContainer', 'Credit3', 'Credit1', 'EditProfile',
       'Credit2', 'Finances', 'CC3', 'Saving9', 'Saving1', 'Alerts',
       'Saving8', 'Saving10', 'Leaderboard', 'Saving4', 'VerifyMobile',
       'VerifyHousing', 'RewardDetail', 'VerifyHousingAmount',
       'ProfileMaritalStatus', 'ProfileChildren ', 'ProfileEducation',
       'Saving7', 'ProfileEducationMajor', 'Rewards', 'AccountView',
       'VerifyAnnualIncome', 'VerifyIncomeType', 'Saving2', 'Saving6',
       'Saving2Amount', 'Saving5', 'ProfileJobTitle', 'Login',
       'ProfileEmploymentLength', 'WebView', 'SecurityModal', 'Loan4',
       'ResendToken', 'TransactionList', 'NetworkFailure', 'ListPicker'],
      dtype=object)
```

In [34]:
```
df['screen_list']=df['screen_list'].astype(str)+','
for sc in a:
    df[sc]=df['screen_list'].str.contains(sc).astype(int)
    df['screen_list']=df['screen_list'].str.replace(sc+',','')

df['Others']=df['screen_list'].str.count(",")
```

In [35]:
```
df=df.drop(columns=['screen_list','difference'])
```

In [36]:
```
df
```

Out[36]:

	user	dayofweek	hour	age	numscreens	minigame	used_premium_feature	enrolled
0	235136	3	2	23	15	0	0	0
1	333588	6	1	24	13	0	0	0
2	254414	1	19	23	3	0	1	0
3	234192	4	16	28	40	0	0	1
4	51549	1	18	31	32	0	0	1
...	...	...	...	...	...	...	...	...
49995	222774	3	13	32	13	0	0	1
49996	169179	1	0	35	4	0	1	0
49997	302367	2	22	39	25	0	0	0
49998	324905	6	12	27	26	0	0	1
49999	27047	4	1	25	26	0	0	0

50000 rows × 68 columns

In [37]:
```
df.columns
```

Out[37]:
```
Index(['user', 'dayofweek', 'hour', 'age', 'numscreens', 'minigame',
       'used_premium_feature', 'enrolled', 'liked', 'Loan2', 'location',
       'Institutions', 'Credit3Container', 'VerifyPhone', 'BankVerification',
       'VerifyDateOfBirth', 'ProfilePage', 'VerifyCountry', 'Cycle',
       'idscreen', 'Credit3Dashboard', 'Loan3', 'CC1Category', 'Splash',
       'Loan', 'CC1', 'RewardsContainer', 'Credit3', 'Credit1', 'EditProfile',
       'Credit2', 'Finances', 'CC3', 'Saving9', 'Saving1', 'Alerts', 'Saving8',
       'Saving10', 'Leaderboard', 'Saving4', 'VerifyMobile', 'VerifyHousing',
       'RewardDetail', 'VerifyHousingAmount', 'ProfileMaritalStatus',
       'ProfileChildren ', 'ProfileEducation', 'Saving7',
       'ProfileEducationMajor', 'Rewards', 'AccountView', 'VerifyAnnualIncome',
       'VerifyIncomeType', 'Saving2', 'Saving6', 'Saving2Amount', 'Saving5',
       'ProfileJobTitle', 'Login', 'ProfileEmploymentLength', 'WebView',
       'SecurityModal', 'Loan4', 'ResendToken', 'TransactionList',
       'NetworkFailure', 'ListPicker', 'Others'],
      dtype='object')
```

In [38]:
```
saving_screens=['Saving9', 'Saving1','Saving8', 'Saving10', 'Saving4','Saving2', 'S
df['SavingCount']=df[saving_screens].sum(axis=1)
df=df.drop(columns=saving_screens)
cm=['Credit3', 'Credit1', 'Credit2','Credit3Dashboard','Credit3Container']
df['CMCount']=df[cm].sum(axis=1)
df=df.drop(columns=cm)
cc=['CC1','CC3','CC1Category']
df['CCCount']=df[cc].sum(axis=1)
df=df.drop(columns=cc)
loan=['Loan','Loan2','Loan3','Loan4']
df['LoanCount']=df[loan].sum(axis=1)
df=df.drop(columns=loan)
```

In [39]:
```
df.head()
```

Out[39]:

	user	dayofweek	hour	age	numscreens	minigame	used_premium_feature	enrolled	liked
0	235136	3	2	23	15	0	0	0	0
1	333588	6	1	24	13	0	0	0	0
2	254414	1	19	23	3	0	1	0	1
3	234192	4	16	28	40	0	0	1	0
4	51549	1	18	31	32	0	0	1	1

5 rows × 50 columns

In [40]:
```
df.columns
```

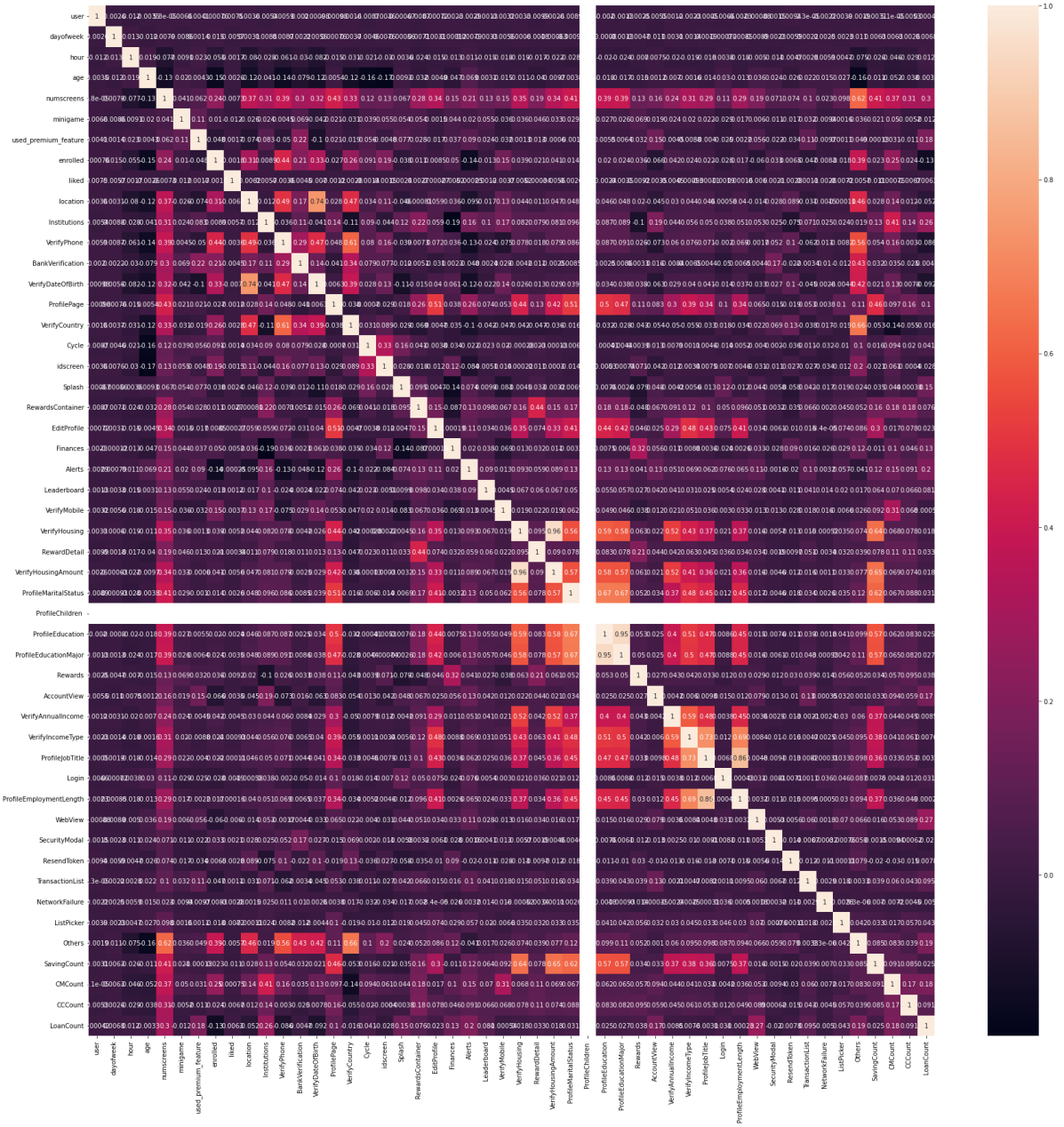
Out[40]:
```
Index(['user', 'dayofweek', 'hour', 'age', 'numscreens', 'minigame',
       'used_premium_feature', 'enrolled', 'liked', 'location', 'Institutions',
       'VerifyPhone', 'BankVerification', 'VerifyDateOfBirth', 'ProfilePage',
       'VerifyCountry', 'Cycle', 'idscreen', 'Splash', 'RewardsContainer',
       'EditProfile', 'Finances', 'Alerts', 'Leaderboard', 'VerifyMobile',
       'VerifyHousing', 'RewardDetail', 'VerifyHousingAmount',
       'ProfileMaritalStatus', 'ProfileChildren ', 'ProfileEducation',
       'ProfileEducationMajor', 'Rewards', 'AccountView', 'VerifyAnnualIncome',
       'VerifyIncomeType', 'ProfileJobTitle', 'Login',
       'ProfileEmploymentLength', 'WebView', 'SecurityModal', 'ResendToken',
       'TransactionList', 'NetworkFailure', 'ListPicker', 'Others',
       'SavingCount', 'CMCount', 'CCCount', 'LoanCount'],
      dtype='object')
```

In [41]:
```
df.to_csv('RevisedAppData.csv')
```

In [42]:
```
new1=df.copy()
```

In [43]:
```
plt.figure(figsize=(30,30))
sns.heatmap(new1.corr(),annot=True)
```

Out[43]:
```
<AxesSubplot:>
```



In [44]:
```
new1.head()
```

Out[44]:

	user	dayofweek	hour	age	numscreens	minigame	used_premium_feature	enrolled	liked
0	235136	3	2	23	15	0	0	0	0
1	333588	6	1	24	13	0	0	0	0
2	254414	1	19	23	3	0	1	0	1
3	234192	4	16	28	40	0	0	1	0
4	51549	1	18	31	32	0	0	1	1

5 rows × 50 columns

In [45]:
```
x=df.drop(columns='enrolled')
y=df['enrolled']
```

In [46]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

In [47]:
```
x_train.isnull().sum()
```

Out[47]:
```
user                       0
dayofweek                  0
hour                       0
age                        0
numscreens                 0
minigame                   0
used_premium_feature       0
liked                      0
location                   0
Institutions               0
VerifyPhone                0
BankVerification           0
VerifyDateOfBirth          0
ProfilePage                0
VerifyCountry              0
Cycle                      0
idscreen                   0
Splash                     0
RewardsContainer           0
EditProfile                0
Finances                   0
Alerts                     0
Leaderboard                0
VerifyMobile               0
VerifyHousing              0
RewardDetail               0
VerifyHousingAmount        0
ProfileMaritalStatus       0
ProfileChildren            0
ProfileEducation           0
ProfileEducationMajor      0
Rewards                    0
AccountView                0
VerifyAnnualIncome         0
VerifyIncomeType           0
ProfileJobTitle            0
Login                      0
ProfileEmploymentLength    0
WebView                    0
SecurityModal              0
ResendToken                0
TransactionList            0
NetworkFailure             0
ListPicker                 0
Others                     0
SavingCount                0
CMCount                    0
CCCount                    0
LoanCount                  0
dtype: int64
```

In [48]:
```
y_train.isnull().sum()
```

Out[48]:
```
0
```

In [49]:
```
print('Shape of x_train = ', x_train.shape)
print('Shape of x_test = ', x_test.shape)
print('Shape of y_train = ', y_train.shape)
print('Shape of y_test = ', y_test.shape)
```

```
Shape of x_train =  (40000, 49)
Shape of x_test =  (10000, 49)
Shape of y_train =  (40000,)
Shape of y_test =  (10000,)
```

In [50]:
```
train_identifier=x_train['user']
x_train=x_train.drop(columns='user')
test_identifier=x_test['user']
x_test=x_test.drop(columns='user')
```

Feature Scaling

The multiple features in the different units so for the best accuracy need to convert all features in a single unit.

In [51]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train_sc = pd.DataFrame(sc.fit_transform(x_train))
X_test_sc = pd.DataFrame(sc.fit_transform(x_test))
```

In [52]:
```
X_train_sc
```

Out[52]:

	0	1	2	3	4	5	6	7	
0	-0.504961	1.000837	0.025525	-1.026726	-0.346830	2.186018	2.246319	-1.039218	-0.6
1	-0.997389	1.135280	-0.898034	1.328829	2.883254	-0.457453	-0.445173	0.962262	1.5
2	-1.489818	-1.150250	-0.528611	4.066366	2.883254	-0.457453	-0.445173	-1.039218	1.5
3	0.479896	0.059736	-0.620967	0.182883	2.883254	-0.457453	-0.445173	0.962262	-0.6
4	-0.012532	0.463065	1.687932	-0.644744	-0.346830	-0.457453	-0.445173	0.962262	-0.6
...	...	...	...	...	...	...	...	...	
39995	0.972325	1.404166	-1.175102	-0.963062	-0.346830	-0.457453	-0.445173	-1.039218	-0.6
39996	-1.489818	0.328622	-0.898034	-1.090390	2.883254	2.186018	-0.445173	-1.039218	-0.6
39997	-0.012532	-0.881364	-0.620967	1.392493	-0.346830	-0.457453	-0.445173	-1.039218	1.5
39998	0.479896	0.059736	-0.436255	-1.090390	2.883254	-0.457453	2.246319	-1.039218	-0.6
39999	-0.997389	0.731951	-1.082746	-0.517417	-0.346830	-0.457453	2.246319	0.962262	-0.6

40000 rows × 48 columns

In [53]:
```
X_test_sc
```

Out[53]:

	0	1	2	3	4	5	6	7	
0	-1.496239	-1.554724	0.866223	0.824043	-0.350843	-0.449198	2.262731	0.976477	-0.64
1	0.958624	-0.882463	-0.535024	0.571073	-0.350843	-0.449198	-0.441944	0.976477	-0.64
2	-1.496239	0.327606	0.492557	-0.630533	-0.350843	2.226191	-0.441944	-1.024090	-0.64
3	0.958624	0.865414	-1.002106	-0.504049	-0.350843	-0.449198	2.262731	-1.024090	-0.64
4	0.958624	-1.554724	-0.161358	0.697558	-0.350843	-0.449198	-0.441944	-1.024090	-0.64
...	...	...	...	...	...	...	...	...	
9995	-1.005267	1.268771	0.679390	0.634315	-0.350843	-0.449198	-0.441944	0.976477	-0.64
9996	-1.496239	-0.075750	0.212308	-0.377564	-0.350843	-0.449198	-0.441944	-1.024090	-0.64
9997	0.958624	0.596510	-0.441608	2.974286	-0.350843	-0.449198	2.262731	-1.024090	1.55
9998	-1.005267	-1.285819	-1.002106	0.254861	-0.350843	-0.449198	-0.441944	0.976477	-0.64
9999	0.467652	-1.554724	5.443629	3.416983	2.850279	-0.449198	-0.441944	-1.024090	1.55

10000 rows × 48 columns

In [54]:
```
#givig back the columns name to each dataset
X_train_sc.columns=x_train.columns.values
X_test_sc.columns=x_test.columns.values
X_train_sc.index=x_train.index.values
X_test_sc.index=x_test.index.values
```

In [55]:
```
X_train_sc.head()
```

Out[55]:

	dayofweek	hour	age	numscreens	minigame	used_premium_feature	liked
20330	-0.504961	1.000837	0.025525	-1.026726	-0.346830	2.186018	2.24631
17532	-0.997389	1.135280	-0.898034	1.328829	2.883254	-0.457453	-0.44517
45819	-1.489818	-1.150250	-0.528611	4.066366	2.883254	-0.457453	-0.44517
34807	0.479896	0.059736	-0.620967	0.182883	2.883254	-0.457453	-0.44517
31888	-0.012532	0.463065	1.687932	-0.644744	-0.346830	-0.457453	-0.44517

5 rows × 48 columns

In [56]:
```
X_test_sc.head()
```

Out[56]:

	dayofweek	hour	age	numscreens	minigame	used_premium_feature	liked
11841	-1.496239	-1.554724	0.866223	0.824043	-0.350843	-0.449198	2.26273
19602	0.958624	-0.882463	-0.535024	0.571073	-0.350843	-0.449198	-0.44194
45519	-1.496239	0.327606	0.492557	-0.630533	-0.350843	2.226191	-0.44194
25747	0.958624	0.865414	-1.002106	-0.504049	-0.350843	-0.449198	2.26273
42642	0.958624	-1.554724	-0.161358	0.697558	-0.350843	-0.449198	-0.44194

5 rows × 48 columns

In [57]:
```
#comparing each dataset
X_train_sc=x_train
X_test_sc=x_test
```

MODEL BUILDING

In [58]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report,confusion_matrix
classifier_model=LogisticRegression(random_state=0,penalty='l1', solver='liblinear
```

In [59]:
```
classifier_model
```

Out[59]:
```
LogisticRegression(penalty='l1', random_state=0, solver='liblinear')
```

In [60]:
```
classifier_model.fit(x_train,y_train)
```

Out[60]:
```
LogisticRegression(penalty='l1', random_state=0, solver='liblinear')
```

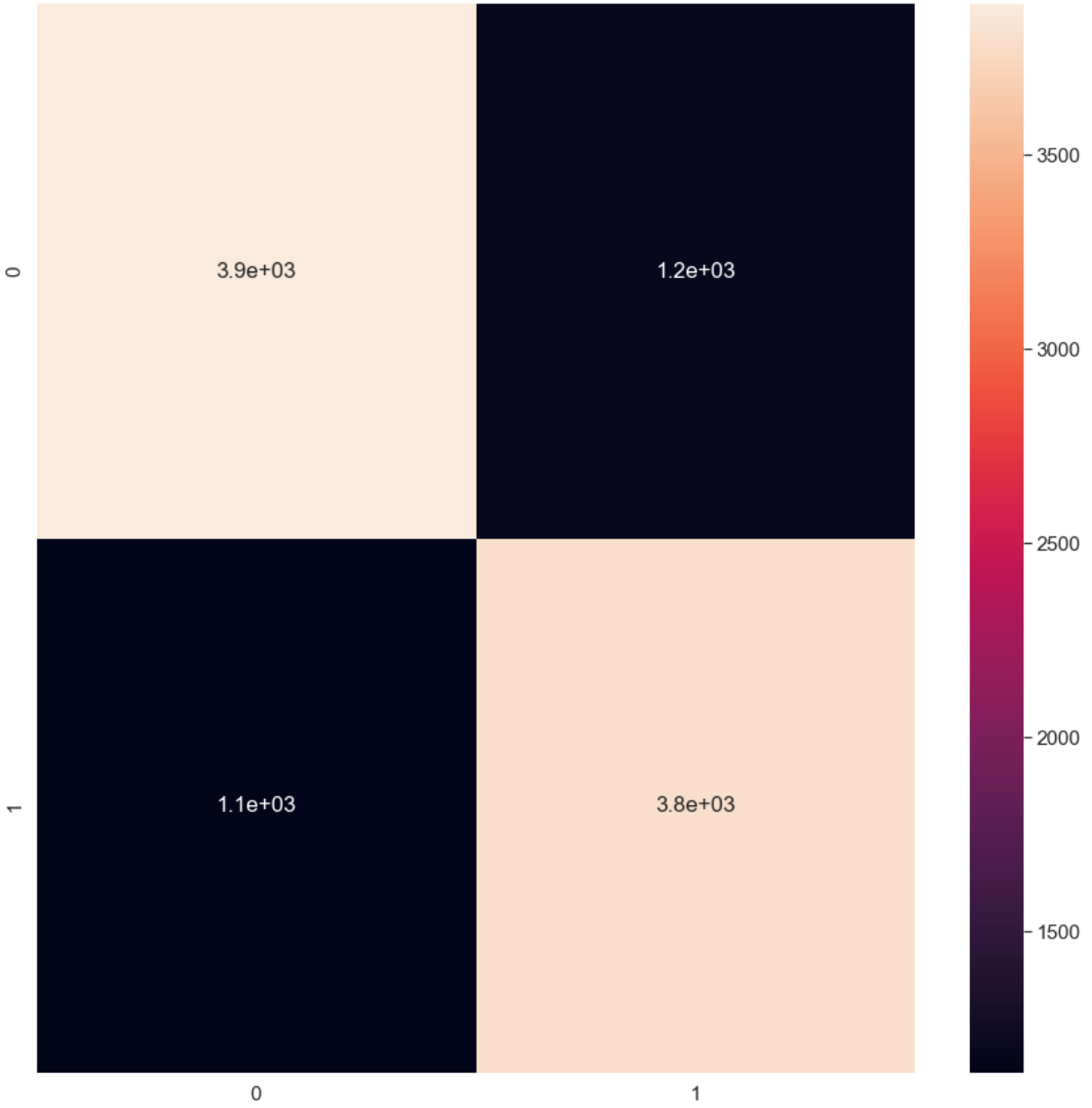
In [61]:
```
y_predict=classifier_model.predict(x_test)
```

In [62]:
```
y_predict
```

Out[62]:
```
array([1, 1, 0, ..., 0, 1, 1], dtype=int64)
```

In [63]:
```
plt.figure(figsize=(15,15))
cmm=confusion_matrix(y_test,y_predict)
sns.set(font_scale=1.4)
sns.heatmap(cmm,annot=True)
```

Out[63]:
```
<AxesSubplot:>
```



In [64]:
```
print(classification_report(y_test,y_predict))
```

```
              precision    recall  f1-score   support

           0       0.77      0.77      0.77      5072
           1       0.76      0.77      0.77      4928

    accuracy                           0.77     10000
   macro avg       0.77      0.77      0.77     10000
weighted avg       0.77      0.77      0.77     10000
```

In [ ]:
```
```

Mapping predicted output to the target

In the below output, you can find the predicted output by model and actual target output.

In [65]:
```
final_result = pd.concat([test_identifier, y_test], axis = 1)
final_result['predicted result'] = y_predict

print(final_result)
```

	user	enrolled	predicted result
11841	239786	1	1
19602	279644	1	1
45519	98290	0	0
25747	170150	1	1
42642	237568	1	1
...	...	...	...
25091	143036	1	0
27853	91158	1	1
47278	248318	0	0
37020	142418	1	1
2217	279355	1	1

[10000 rows x 3 columns]

In [ ]: