In [1]:
```python
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```python
import pandas as pd
import plotly.express as px
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.graph_objects as go
```

In [4]:
```python
df = pd.read_excel("D:\BPC\Python\Credit card project\default_of_credit_card_clients_0.xlsx")
df = pd.DataFrame(df)
```

EDA

In [8]:
```python
df.shape
```

Out[8]:
```
(30000, 25)
```

In [9]:
```python
df.info() #All columns are in integer type and non-null. So, the data is clear.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 25 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   ID                          30000 non-null  int64
 1   LIMIT_BAL                   30000 non-null  int64
 2   SEX                         30000 non-null  int64
 3   EDUCATION                   30000 non-null  int64
 4   MARRIAGE                    30000 non-null  int64
 5   AGE                         30000 non-null  int64
 6   PAY_0                       30000 non-null  int64
 7   PAY_2                       30000 non-null  int64
 8   PAY_3                       30000 non-null  int64
 9   PAY_4                       30000 non-null  int64
 10  PAY_5                       30000 non-null  int64
 11  PAY_6                       30000 non-null  int64
 12  BILL_AMT1                   30000 non-null  int64
 13  BILL_AMT2                   30000 non-null  int64
 14  BILL_AMT3                   30000 non-null  int64
 15  BILL_AMT4                   30000 non-null  int64
 16  BILL_AMT5                   30000 non-null  int64
 17  BILL_AMT6                   30000 non-null  int64
 18  PAY_AMT1                    30000 non-null  int64
 19  PAY_AMT2                    30000 non-null  int64
 20  PAY_AMT3                    30000 non-null  int64
 21  PAY_AMT4                    30000 non-null  int64
 22  PAY_AMT5                    30000 non-null  int64
 23  PAY_AMT6                    30000 non-null  int64
 24  default payment next month  30000 non-null  int64
dtypes: int64(25)
memory usage: 5.7 MB
```

In [10]:
```python
for col in df.columns:
    print(col)
    print(sorted(df[col].unique())) #unique values of specific columns sorted in ascending order
    print('-'*100)
```

```
308, 3309, 3310, 3311, 3312, 3313, 3314, 3315, 3316, 3317, 3318, 3319, 3320, 3321, 3322, 3323, 3324, 3325, 3326, 3327, 3328, 3329, 3330,
3331, 3332, 3333, 3334, 3335, 3336, 3337, 3338, 3339, 3340, 3341, 3342, 3343, 3344, 3345, 3346, 3347, 3348, 3349, 3350, 3351, 3352, 335
3, 3354, 3355, 3356, 3357, 3358, 3359, 3360, 3361, 3362, 3363, 3364, 3365, 3366, 3367, 3368, 3369, 3370, 3371, 3372, 3373, 3374, 3375, 3
376, 3377, 3378, 3379, 3380, 3381, 3382, 3383, 3384, 3385, 3386, 3387, 3388, 3389, 3390, 3391, 3392, 3393, 3394, 3395, 3396, 3397, 3398,
3399, 3400, 3401, 3402, 3403, 3404, 3405, 3406, 3407, 3408, 3409, 3410, 3411, 3412, 3413, 3414, 3415, 3416, 3417, 3418, 3419, 3420, 342
1, 3422, 3423, 3424, 3425, 3426, 3427, 3428, 3429, 3430, 3431, 3432, 3433, 3434, 3435, 3436, 3437, 3438, 3439, 3440, 3441, 3442, 3443, 3
444, 3445, 3446, 3447, 3448, 3449, 3450, 3451, 3452, 3453, 3454, 3455, 3456, 3457, 3458, 3459, 3460, 3461, 3462, 3463, 3464, 3465, 3466,
3467, 3468, 3469, 3470, 3471, 3472, 3473, 3474, 3475, 3476, 3477, 3478, 3479, 3480, 3481, 3482, 3483, 3484, 3485, 3486, 3487, 3488, 348
9, 3490, 3491, 3492, 3493, 3494, 3495, 3496, 3497, 3498, 3499, 3500, 3501, 3502, 3503, 3504, 3505, 3506, 3507, 3508, 3509, 3510, 3511, 3
512, 3513, 3514, 3515, 3516, 3517, 3518, 3519, 3520, 3521, 3522, 3523, 3524, 3525, 3526, 3527, 3528, 3529, 3530, 3531, 3532, 3533, 3534,
3535, 3536, 3537, 3538, 3539, 3540, 3541, 3542, 3543, 3544, 3545, 3546, 3547, 3548, 3549, 3550, 3551, 3552, 3553, 3554, 3555, 3556, 355
7, 3558, 3559, 3560, 3561, 3562, 3563, 3564, 3565, 3566, 3567, 3568, 3569, 3570, 3571, 3572, 3573, 3574, 3575, 3576, 3577, 3578, 3579, 3
580, 3581, 3582, 3583, 3584, 3585, 3586, 3587, 3588, 3589, 3590, 3591, 3592, 3593, 3594, 3595, 3596, 3597, 3598, 3599, 3600, 3601, 3602,
3603, 3604, 3605, 3606, 3607, 3608, 3609, 3610, 3611, 3612, 3613, 3614, 3615, 3616, 3617, 3618, 3619, 3620, 3621, 3622, 3623, 3624, 362
5, 3626, 3627, 3628, 3629, 3630, 3631, 3632, 3633, 3634, 3635, 3636, 3637, 3638, 3639, 3640, 3641, 3642, 3643, 3644, 3645, 3646, 3647, 3
648, 3649, 3650, 3651, 3652, 3653, 3654, 3655, 3656, 3657, 3658, 3659, 3660, 3661, 3662, 3663, 3664, 3665, 3666, 3667, 3668, 3669, 3670,
3671, 3672, 3673, 3674, 3675, 3676, 3677, 3678, 3679, 3680, 3681, 3682, 3683, 3684, 3685, 3686, 3687, 3688, 3689, 3690, 3691, 3692, 369
3, 3694, 3695, 3696, 3697, 3698, 3699, 3700, 3701, 3702, 3703, 3704, 3705, 3706, 3707, 3708, 3709, 3710, 3711, 3712, 3713, 3714, 3715, 3
716, 3717, 3718, 3719, 3720, 3721, 3722, 3723, 3724, 3725, 3726, 3727, 3728, 3729, 3730, 3731, 3732, 3733, 3734, 3735, 3736, 3737, 3738,
3739, 3740, 3741, 3742, 3743, 3744, 3745, 3746, 3747, 3748, 3749, 3750, 3751, 3752, 3753, 3754, 3755, 3756, 3757, 3758, 3759, 3760, 376
```

Above Line: 1 marker is missing under PAY_5 and PAY_6. It is assumed that no payment was delayed for one month i.e. payment was either duly paid or was delayed for more than one month for the columns concerned.

EDUCATION column has three invalid values: 0, 5, 6

MARRIAGE column has one invalid value: 0

In [11]:

```
#PAY_0 column renamed as PAY_1
df1 = df.rename(columns = {'PAY_0' : 'PAY_1'})
df1

#Dropping the rows with 0, 5, 6 values in EDUCATION and 0 in MARRIAGE columns
df1 = df1[df1['MARRIAGE'] != 0]
df1 = df1.query('EDUCATION != 0 and EDUCATION != 5 and EDUCATION != 6')

# Print modified DataFrame
print("\nModified DataFrame:")
df1
```

X	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	default payment next month
2	2	1	24	2	2	0	0	...	0	0	0	0	689	0	0	0	0	1
2	2	2	26	0	2	0	0	...	3272	3455	3261	0	1000	1000	1000	0	2000	1
2	2	2	34	0	0	0	0	...	14331	14948	15549	1518	1500	1000	1000	1000	5000	0
2	2	1	37	0	0	0	0	...	28314	28959	29547	2000	2019	1200	1100	1069	1000	0
1	2	1	57	0	0	0	0	...	20940	19146	19131	2000	36681	10000	9000	689	679	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1	3	1	39	0	0	0	0	...	88004	31237	15980	8500	20000	5003	3047	5000	1000	0

In [14]:

```
#Calculating the total bill amount, total pay amount, total outstanding amount after 6 months for each row
df1['TOT_BILL_AMT'] = df1[['BILL_AMT1', 'BILL_AMT2', 'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6']].sum(axis=1)
df1['TOT_PAY_AMT'] = df1[['PAY_AMT1', 'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6']].sum(axis=1)
df1['OUTS_AMT'] = df1['TOT_BILL_AMT'] - df1['TOT_PAY_AMT']
df1
```

Out[14]:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	...	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	d pay r
0	1	20000	2	2	1	24	2	2	0	0	...	0	689	0	0	0	0	
1	2	120000	2	2	2	26	0	2	0	0	...	0	1000	1000	1000	0	2000	
2	3	90000	2	2	2	34	0	0	0	0	...	1518	1500	1000	1000	1000	5000	
3	4	50000	2	2	1	37	0	0	0	0	...	2000	2019	1200	1100	1069	1000	
4	5	50000	1	2	1	57	0	0	0	0	...	2000	36681	10000	9000	689	679	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
29995	29996	220000	1	3	1	39	0	0	0	0	...	8500	20000	5003	3047	5000	1000	
29996	29997	150000	1	3	2	43	0	0	0	0	...	1837	3526	8998	129	0	0	
29997	29998	30000	1	2	2	37	4	3	2	0	...	0	0	22000	4200	2000	3100	
29998	29999	80000	1	3	1	41	1	0	0	0	...	85900	3409	1178	1926	52964	1804	
29999	30000	50000	1	2	1	46	0	0	0	0	...	2078	1800	1430	1000	1000	1000	

29601 rows × 28 columns

In [15]:

```
#Using the describe() function for the specific columns
spec_cols = ['LIMIT_BAL', 'BILL_AMT1', 'BILL_AMT2', 'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6', 'PAY_AMT1',
             'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6', 'TOT_BILL_AMT', 'TOT_PAY_AMT', 'OUTS_AMT']
description = df1[spec_cols].describe().round(decimals = 2)
print(description)
```

```
        LIMIT_BAL  BILL_AMT1  BILL_AMT2   BILL_AMT3  BILL_AMT4  BILL_AMT5  \
count    29601.00   29601.00   29601.00    29601.00   29601.00   29601.00   
mean    167550.54   50957.43   48942.19    46803.20   43122.55   40235.55   
std     129944.02   73370.24   70923.99    69123.89   64196.38   60699.34   
min      10000.00 -165580.00  -69777.00  -157264.00 -170000.00  -81334.00   
25%      50000.00    3528.00    2970.00     2652.00    2329.00    1780.00   
50%     140000.00   22259.00   21050.00    20035.00   19005.00   18091.00   
75%     240000.00   66623.00   63497.00    59830.00   54271.00   50072.00   
max    1000000.00  964511.00  983931.00  1664089.00  891586.00  927171.00   

       BILL_AMT6   PAY_AMT1    PAY_AMT2   PAY_AMT3   PAY_AMT4   PAY_AMT5  \
count   29601.00   29601.00    29601.00   29601.00   29601.00   29601.00   
mean    38858.45    5649.56     5894.79    5198.42    4828.66    4795.03   
std     59519.89   16568.26    23089.19   17580.91   15711.06   15244.22   
min   -339603.00       0.00        0.00       0.00       0.00       0.00   
25%      1278.00    1000.00      825.00     390.00     298.00     259.00   
50%     17118.00    2100.00     2007.00    1800.00    1500.00    1500.00   
75%     49121.00    5005.00     5000.00    4500.00    4014.00    4042.00   
max    961664.00  873552.00  1684259.00  896040.00  621000.00  426529.00   

        PAY_AMT6  TOT_BILL_AMT  TOT_PAY_AMT    OUTS_AMT  
count   29601.00      29601.00     29601.00    29601.00  
mean     5181.33     268919.37     31547.78   237371.59  
std     17657.26     378782.13     60817.85   362462.36  
min         0.00    -336259.00         0.00 -2671514.00  
25%       138.00      28568.00      6689.00     4463.00  
50%      1500.00     125476.00     14353.00   101445.00  
75%      4000.00     341268.00     33424.00   303599.00  
max    528666.00    5263883.00   3764066.00  4116080.00  
```

In [27]:

```
df1['default payment next month'].value_counts()
```

Out[27]:

```
default payment next month
0    22996
1     6605
Name: count, dtype: int64
```

In [13]:

```
#Preparing a dataframe contatining the bill_amt and pay_amt columns:
bill_sums = df1.filter(like='BILL_AMT').sum().tolist() #total bill statement in list format
pay_sums = df1.filter(like='PAY_AMT').sum().tolist() #total previous payment in list format

print("Bill Amount Sums:")
print(bill_sums)
print("\nPayment Amount Sums:")
print(pay_sums)


#Converting the above result into two separate dataframes:
bill_sums = pd.DataFrame(bill_sums, columns = ['BILL_SUMS'])
pay_sums = pd.DataFrame(pay_sums, columns = ['PAY_SUMS'])

#Joining two above mentioned dataframes:
final_table = pd.concat([bill_sums, pay_sums], axis = 1)

print('-'*50)
print('Sums Table:')
print(final_table)
```

```
Bill Amount Sums:
[1508390945, 1448737753, 1385421620, 1276470727, 1191012373, 1150248973]

Payment Amount Sums:
[167232635, 174491631, 153878309, 142933143, 141937764, 153372442]
--------------------------------------------------
Sums Table:
    BILL_SUMS   PAY_SUMS
0  1508390945  167232635
1  1448737753  174491631
2  1385421620  153878309
3  1276470727  142933143
4  1191012373  141937764
5  1150248973  153372442
```

In [45]:
```
#Calculating the outstanding sums
final_table['OUT_SUMS'] = final_table['BILL_SUMS'] - final_table['PAY_SUMS']
final_table

#Entering the month column
new_column_data = ['September', 'August', 'July', 'June', 'May', 'April']
final_table['Month (2005)'] = new_column_data
final_table
```

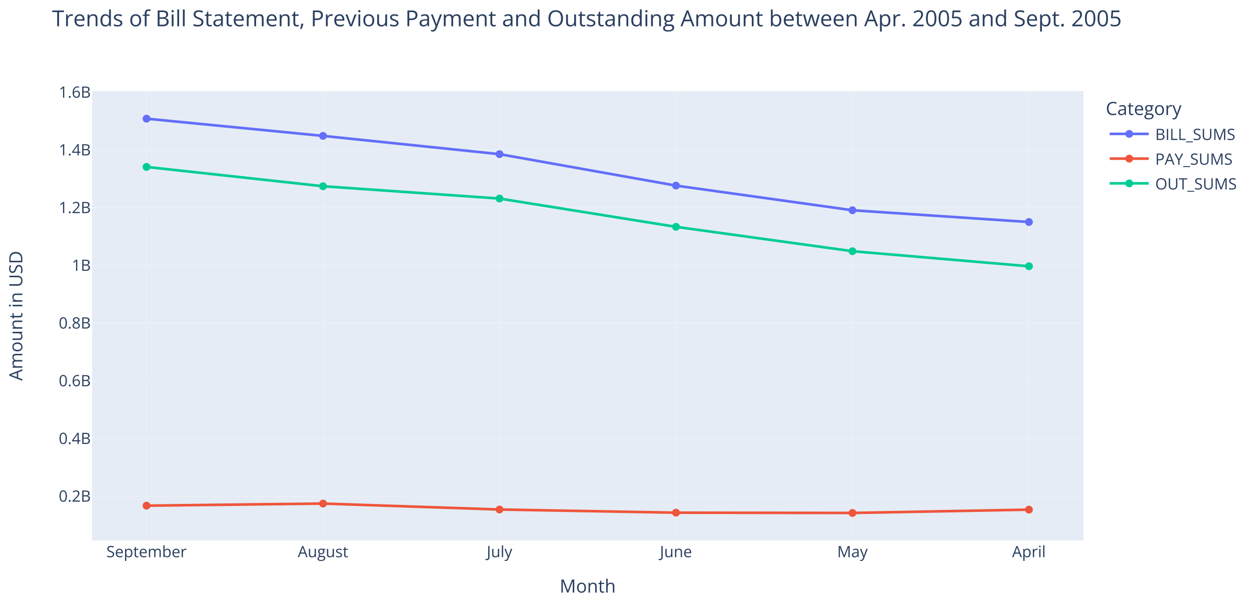
Out[45]:

	BILL_SUMS	PAY_SUMS	OUT_SUMS	Month (2005)
0	1508390945	167232635	1341158310	September
1	1448737753	174491631	1274246122	August
2	1385421620	153878309	1231543311	July
3	1276470727	142933143	1133537584	June
4	1191012373	141937764	1049074609	May
5	1150248973	153372442	996876531	April

Analysis

Compare the trends of bill statement, previous payment, Outstanding amount over the months.

In [106]:
```
fig = px.line(final_table2, x = 'Month (2005)', y = ['BILL_SUMS', 'PAY_SUMS', 'OUT_SUMS'],
             labels = {'Month (2005)' : 'Month', 'value' : 'Amount in USD', 'variable' : 'Category'},
             title = 'Trends of Bill Statement, Previous Payment and Outstanding Amount between Apr. 2005 and Sept. 2005',
             markers = True)
fig.show()
```



In [18]:
```
fig = px.line(final_table, x = 'Month (2005)', y = ['BILL_SUMS', 'PAY_SUMS', 'OUT_SUMS'],
              labels = {'Month (2005)' : 'Month', 'value' : 'Amount in USD', 'variable' : 'Category'},
              title = 'Trends of Bill Statement, Previous Payment and Outstanding Amount between Apr. 2005 and Sept. 2005',
              markers = True)
fig.show()df1
```

Out[18]:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	...	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	d pay r
0	1	20000	2	2	1	24	2	2	0	0	...	0	689	0	0	0	0	
1	2	120000	2	2	2	26	0	2	0	0	...	0	1000	1000	1000	0	2000	
2	3	90000	2	2	2	34	0	0	0	0	...	1518	1500	1000	1000	1000	5000	
3	4	50000	2	2	1	37	0	0	0	0	...	2000	2019	1200	1100	1069	1000	
4	5	50000	1	2	1	57	0	0	0	0	...	2000	36681	10000	9000	689	679	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
29995	29996	220000	1	3	1	39	0	0	0	0	...	8500	20000	5003	3047	5000	1000	
29996	29997	150000	1	3	2	43	0	0	0	0	...	1837	3526	8998	129	0	0	
29997	29998	30000	1	2	2	37	4	3	2	0	...	0	0	22000	4200	2000	3100	
29998	29999	80000	1	3	1	41	1	0	0	0	...	85900	3409	1178	1926	52964	1804	
29999	30000	50000	1	2	1	46	0	0	0	0	...	2078	1800	1430	1000	1000	1000	

29601 rows × 28 columns

In [19]:
```
#Filtering those outstanding amounts which are more than 0:
df2 = df1[df1['OUTS_AMT'] > 0]
df2
df2.describe().round(decimals = 2)
```

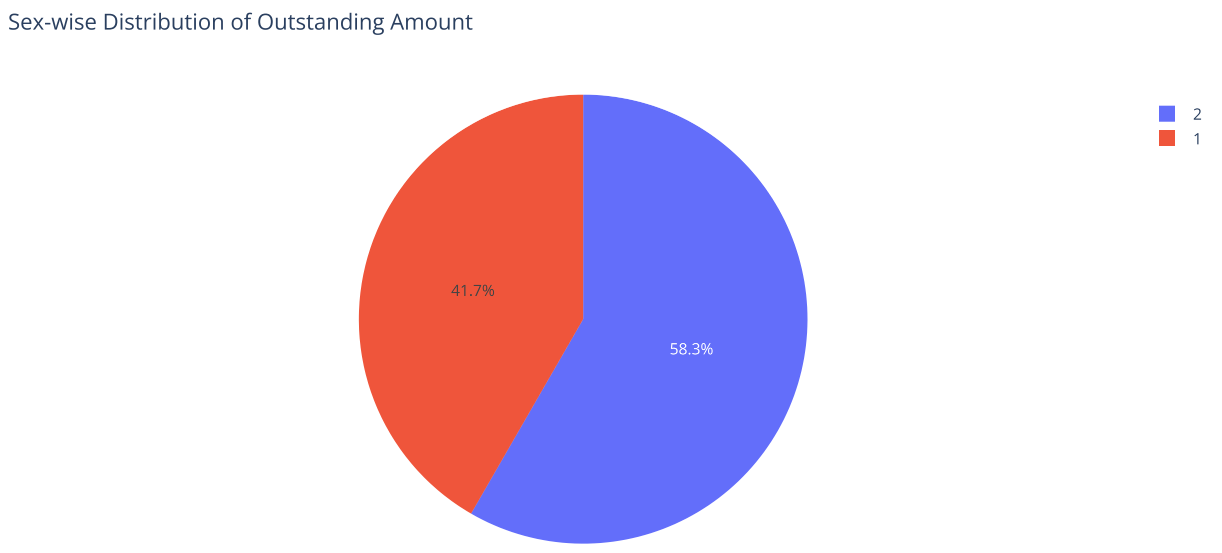
Out[19]:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	...	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AI
count	25049.00	25049.00	25049.00	25049.00	25049.00	25049.00	25049.00	25049.00	25049.00	25049.00	...	25049.0	25049.00	25049.00	25049.00	2504(
mean	14939.89	157637.98	1.59	1.85	1.57	35.22	0.36	0.37	0.35	0.30	...	5845.8	5944.49	5298.36	4927.11	469
std	8619.49	128403.18	0.49	0.70	0.52	9.24	0.80	0.86	0.84	0.82	...	15430.7	17629.18	17175.79	15708.51	1481(
min	1.00	10000.00	1.00	1.00	1.00	21.00	0.00	0.00	0.00	0.00	...	0.0	0.00	0.00	0.00	(
25%	7504.00	50000.00	1.00	1.00	1.00	28.00	0.00	0.00	0.00	0.00	...	1300.0	1200.00	780.00	500.00	43
50%	14968.00	120000.00	2.00	2.00	2.00	34.00	0.00	0.00	0.00	0.00	...	2500.0	2252.00	2000.00	1728.00	175
75%	22293.00	230000.00	2.00	2.00	2.00	41.00	0.00	0.00	0.00	0.00	...	5260.0	5006.00	4874.00	4267.00	416
max	30000.00	1000000.00	2.00	4.00	3.00	79.00	8.00	8.00	8.00	8.00	...	423903.0	580464.00	896040.00	528897.00	42652(

8 rows × 28 columns

Is there any relationship between outstanding amount and sex?

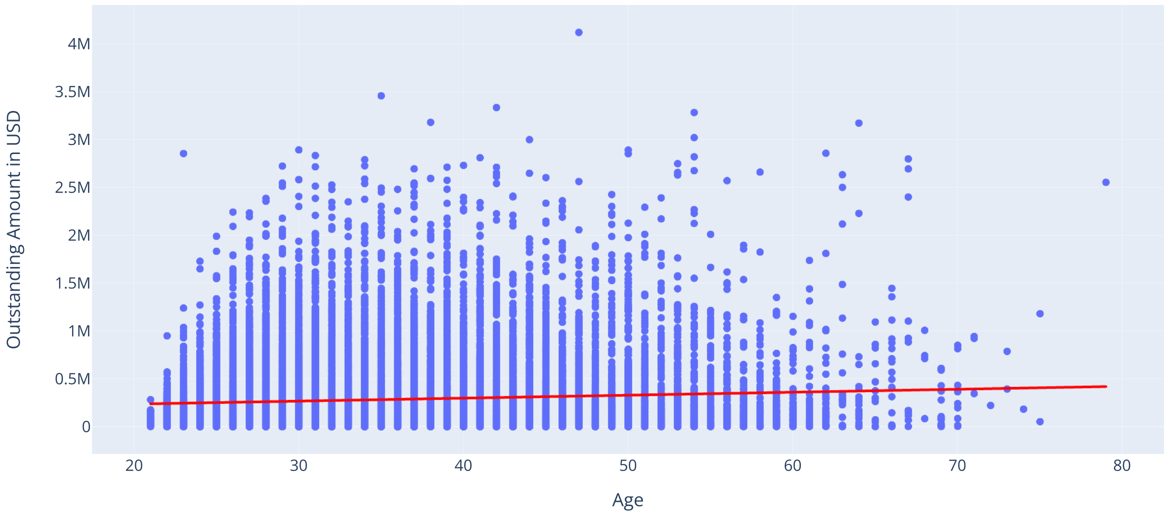
In [20]:
```
fig = px.pie(df2, names = 'SEX', values = 'OUTS_AMT', labels = df2['SEX'],
             title = 'Sex-wise Distribution of Outstanding Amount')
fig.show()
```



Is there any relationship between outstanding amount and age?

```
In [21]: fig = px.scatter(df2, x = 'AGE', y = 'OUTS_AMT',
                          labels = {'OUTS_AMT' : 'Outstanding Amount in USD', 'AGE' : 'Age'}, trendline = 'ols',
                          trendline_color_override = 'red',
                          title = 'Relationship between Age and Outstanding Amount')
         fig.show()

         correlation = df2['AGE'].corr(df2['OUTS_AMT'])
         print(f"Correlation between age and outstanding amount: {correlation}")
```

Relationship between Age and Outstanding Amount

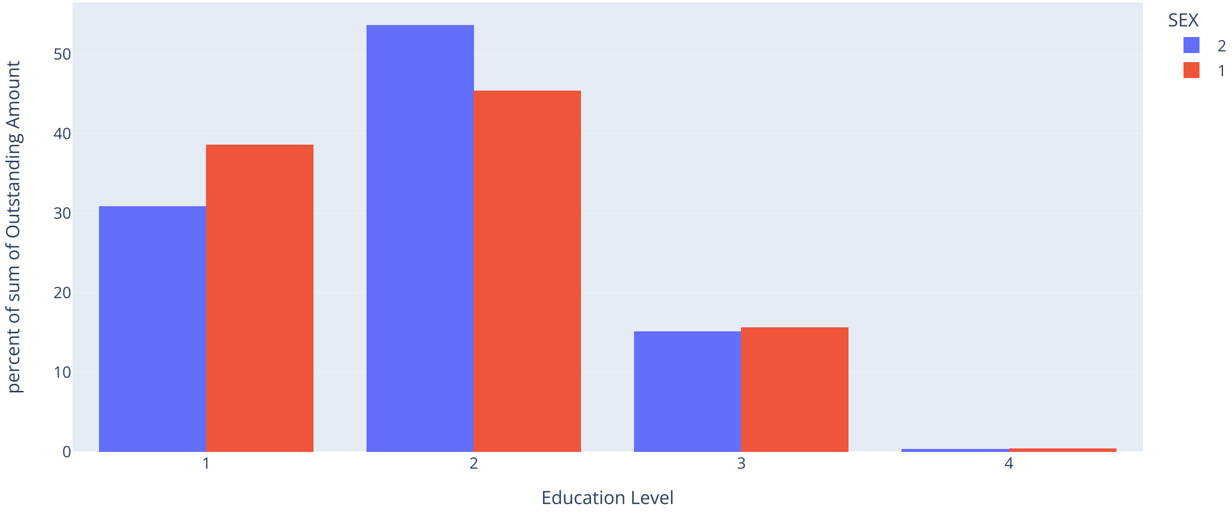


```
Correlation between age and outstanding amount: 0.07599954984480116
```

Is there any relationship between outstanding amount and education?

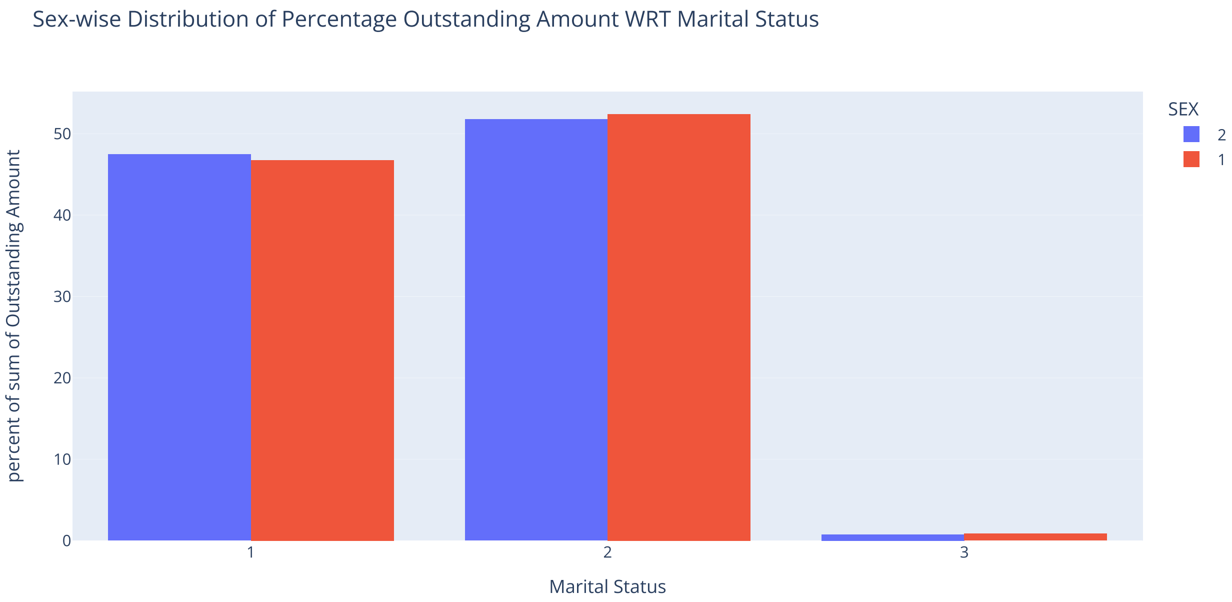
```
In [22]: fig = px.histogram(df2, x = 'EDUCATION', histnorm='percent', y = 'OUTS_AMT', color = 'SEX', barmode = 'group',
                    labels = {'OUTS_AMT' : 'Outstanding Amount', 'EDUCATION' : 'Education Level'},
                          title = 'Sex-wise Distribution of Percentage Outstanding Amount WRT Education Level')
         fig.show()
```

Sex-wise Distribution of Percentage Outstanding Amount WRT Education Level



Is there any relationship between outstanding amount and marital status?

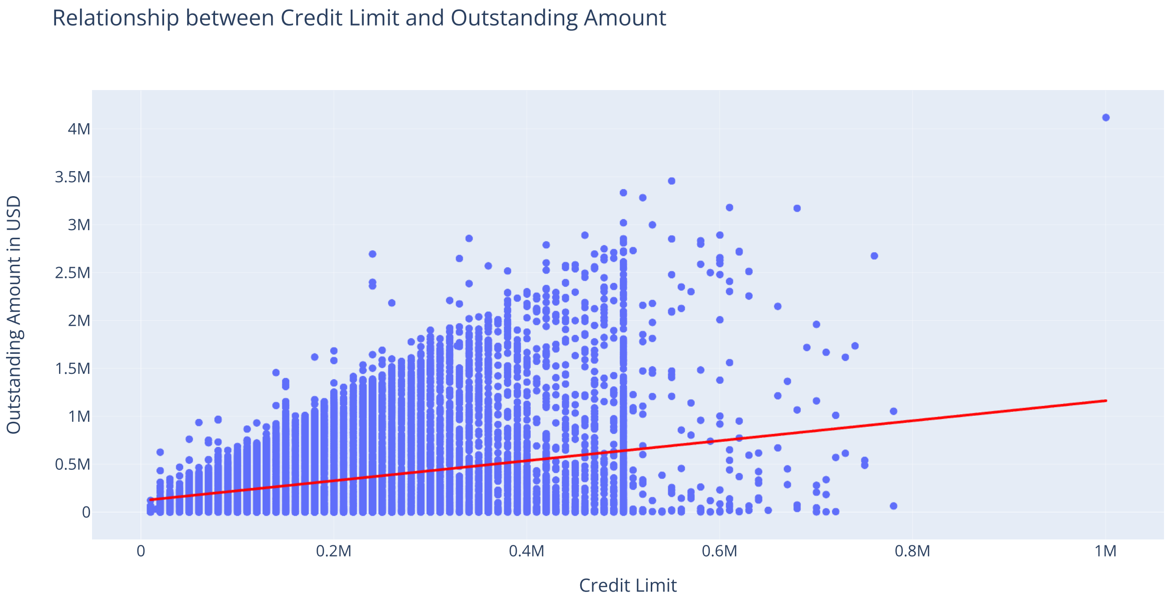
```
In [23]: fig = px.histogram(df2, x = 'MARRIAGE', histnorm='percent', y = 'OUTS_AMT', color = 'SEX', barmode = 'group',
                 labels = {'OUTS_AMT' : 'Outstanding Amount', 'MARRIAGE' : 'Marital Status'},
                           title = 'Sex-wise Distribution of Percentage Outstanding Amount WRT Marital Status')
         fig.show()
```



Is there any correlation between credit limit and outstanding amount?

In [24]:

```
fig = px.scatter(df2, x = 'LIMIT_BAL', y = 'OUTS_AMT',
                labels = {'OUTS_AMT' : 'Outstanding Amount in USD', 'LIMIT_BAL' : 'Credit Limit'}, trendline = 'ols',
                 trendline_color_override = 'red',
                title = 'Relationship between Credit Limit and Outstanding Amount')
fig.show()

correlation = df2['LIMIT_BAL'].corr(df2['OUTS_AMT'])
print(f"Correlation between credit limit and outstanding amount: {correlation}")

#Linear Regression:
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

X = df2[['LIMIT_BAL']]  # X should be a 2D array
y = df2['OUTS_AMT']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create the model
model = LinearRegression()

# Fit the model
model.fit(X_train, y_train)

# Print the coefficients
print('-'*100)
print(f'Intercept: {model.intercept_}')
print(f'Coefficient: {model.coef_[0]}')

# Predict the values
y_pred = model.predict(X_test)

# Calculate mean squared error and R^2 score
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R^2 Score: {r2}')
```

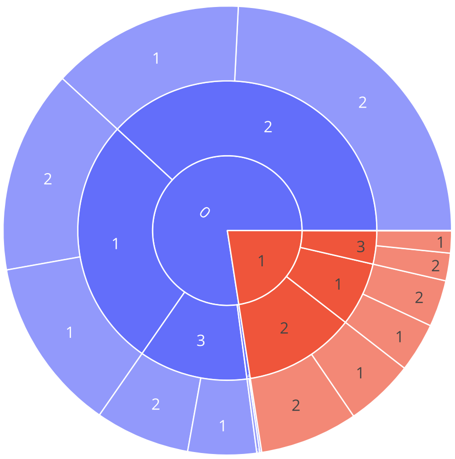


```
Correlation between credit limit and outstanding amount: 0.35608887997880156
----------------------------------------------------------------------------------------------------
Intercept: 116611.29656273013
Coefficient: 1.0635590363558187
Mean Squared Error: 116747915724.87791
R^2 Score: 0.11416334330516864
```

Is there any relationship between outstanding amount and default payment next month?

In [25]:
```
fig = px.sunburst(df2, path = ['default payment next month', 'EDUCATION', 'SEX'], values = 'OUTS_AMT',
                  title = 'Analysis of Default Payment Next Month based on Education and Sex')
fig.show()
```

Analysis of Default Payment Next Month based on Education and Sex



In [46]:
```
final_table
```

Out[46]:

	BILL_SUMS	PAY_SUMS	OUT_SUMS	Month (2005)
0	1508390945	167232635	1341158310	September
1	1448737753	174491631	1274246122	August
2	1385421620	153878309	1231543311	July
3	1276470727	142933143	1133537584	June
4	1191012373	141937764	1049074609	May
5	1150248973	153372442	996876531	April

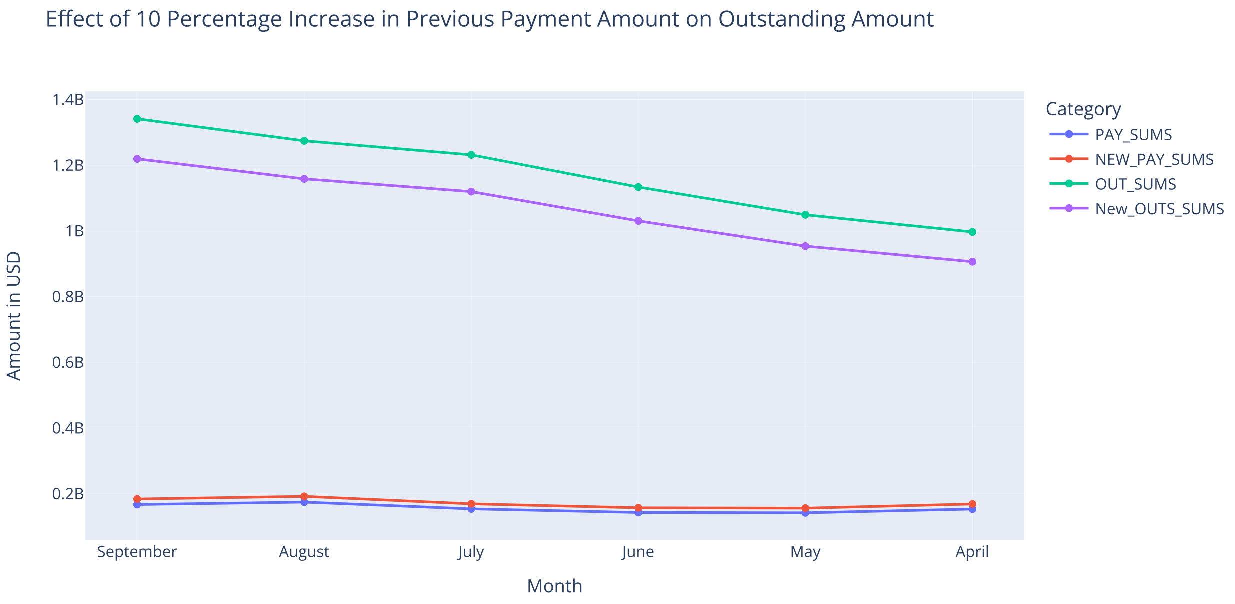
If previous payment amount increased by 10% for each month, what changes would have been there wrt outstanding amount?

In [111]:
```
final_table2 = final_table
final_table2['NEW_PAY_SUMS'] = final_table['PAY_SUMS']*1.1
final_table2['NEW_OUTS_SUMS'] = final_table2['OUT_SUMS']*final_table2['PAY_SUMS']/final_table2['NEW_PAY_SUMS']
final_table2['%Change_OUTS_SUMS'] = (1- final_table2['NEW_OUTS_SUMS']/final_table2['OUT_SUMS'])*100
final_table2
```

Out[111]:

	BILL_SUMS	PAY_SUMS	OUT_SUMS	Month (2005)	NEW_PAY_SUMS	New_OUTS_SUMS	%Change_OUTS_SUMS	NEW_OUTS_SUMS
0	1508390945	167232635	1341158310	September	183955898.5	1.219235e+09	9.090909	1.219235e+09
1	1448737753	174491631	1274246122	August	191940794.1	1.158406e+09	9.090909	1.158406e+09
2	1385421620	153878309	1231543311	July	169266139.9	1.119585e+09	9.090909	1.119585e+09
3	1276470727	142933143	1133537584	June	157226457.3	1.030489e+09	9.090909	1.030489e+09
4	1191012373	141937764	1049074609	May	156131540.4	9.537042e+08	9.090909	9.537042e+08
5	1150248973	153372442	996876531	April	168709686.2	9.062514e+08	9.090909	9.062514e+08

```
In [114]: fig = px.line(final_table2, x = 'Month (2005)', y = ['PAY_SUMS', 'NEW_PAY_SUMS', 'OUT_SUMS', 'New_OUTS_SUMS'],
                        labels = {'Month (2005)' : 'Month', 'value' : 'Amount in USD', 'variable' : 'Category'},
                        title = 'Effect of 10 Percentage Increase in Previous Payment Amount on Outstanding Amount',
                        markers = True)
          fig.show()
```



Effect of 10 Percentage Increase in Previous Payment Amount on Outstanding Amount