Future cash flow analysis

A large global online lending platform that provides loans to both consumers and merchants has hired us to make sure the balance sheet values are correct. We must ensure correct valuation of client portfolio.

In [38]:
```
#obtaining dataset

import pandas as pd
df = pd.read_csv('C:\\Users\\sujoydutta\\Downloads\\Data.csv')


df.head()
```

Out[38]:

	Dates	Origination Amount	31.05.2019	30.06.2019	31.07.2019	31.08.2019	30.09.2019	31.10.2019	30.11.2019
0	31.05.2019	10018746.17	1443069.08	3332200.33	1328138.75	928085.74	736418.27	539403.31	427557.86
1	30.06.2019	10868379.04	0.00	1392751.60	3011884.91	1237868.70	970929.28	892351.83	668767.02
2	31.07.2019	10733932.61	0.00	0.00	1537650.24	2953335.55	1208316.08	879375.19	711016.84
3	31.08.2019	12558727.02	0.00	0.00	0.00	1617681.94	4082016.00	1387474.94	1247623.59
4	30.09.2019	14505071.44	0.00	0.00	0.00	0.00	1992242.84	3930445.60	1394620.78

5 rows × 22 columns

In [39]:
```
# Formula to calculate repayment percentages
def calculate_repayment_percentages(df):

    df_rep = df.drop(columns=['Dates', 'Origination Amount'])


    origination_amount = df['Origination Amount']
    repayment_percentages = df_rep.div(origination_amount, axis=0)

    return repayment_percentages
```

In [40]:
```
#calculating repayment percentages
repayment_percentages = calculate_repayment_percentages(df)
print(repayment_percentages.head())
```

```
   31.05.2019  30.06.2019  31.07.2019  31.08.2019  30.09.2019  31.10.2019  \
0    0.144037    0.332597    0.132565    0.092635    0.073504    0.053839
1    0.000000    0.128147    0.277124    0.113896    0.089335    0.082105
2    0.000000    0.000000    0.143251    0.275140    0.112570    0.081925
3    0.000000    0.000000    0.000000    0.128809    0.325034    0.110479
4    0.000000    0.000000    0.000000    0.000000    0.137348    0.270970

   30.11.2019  31.12.2019  31.01.2020  29.02.2020  31.03.2020  30.04.2020  \
0    0.042676    0.032385    0.023661    0.016805    0.011647    0.009253
1    0.061533    0.046521    0.038607    0.030295    0.023483    0.018295
2    0.066240    0.061324    0.046904    0.039412    0.028189    0.024097
3    0.099343    0.070572    0.055288    0.045468    0.033222    0.026809
4    0.096147    0.084654    0.064765    0.055351    0.043325    0.040654

   31.05.2020  30.06.2020  31.07.2020  31.08.2020  30.09.2020  31.10.2020  \
0    0.006328    0.005317    0.003705    0.002973    0.002448    0.001805
1    0.014905    0.012740    0.008497    0.007328    0.005839    0.004819
```

```
2    0.017868    0.015840    0.011885    0.010276    0.008363    0.006032
3    0.020190    0.015930    0.012092    0.008757    0.007184    0.005626
4    0.031527    0.022321    0.019866    0.016537    0.013254    0.011827

   30.11.2020  31.12.2020
0    0.001655    0.001142
1    0.003991    0.003442
2    0.005721    0.004687
3    0.004228    0.003748
4    0.009829    0.008056
```

In [64]:

```python
# Function to compute expected repayment percentages
def compute_expected_repayment_percentages(repayment_percentages):
    expected_percentages = {}

    for vintage in repayment_percentages.index:
        repayments = repayment_percentages.loc[vintage]
        p = []


        num_months = len(repayments)

        for i in range(num_months):
            if i == 0:
                p.append(repayments.iloc[0])
            elif i == 1:
                if vintage == '31.12.2020':
                    p.append(2 * p[0])
                else:
                    p.append(repayments.iloc[1])
            else:
                previous_sum = sum(p[:i])
                if previous_sum < 1:
                    p_i = max(repayments.iloc[1] * np.log(1 + (1 - (i) / num_months) / (
                else:
                    p_i = max(repayments.iloc[1] * (1 - (i) / num_months), 0)
                p.append(p_i)

        expected_percentages[vintage] = p


    expected_df = pd.DataFrame(expected_percentages).T


    expected_df.columns = repayment_percentages.columns


    expected_df = expected_df.dropna()

    return expected_df
```

In [65]:

```python
#calculating expected repayment percentages
import numpy as np
expected_repayment_percentages = compute_expected_repayment_percentages(repayment_percen
print(expected_repayment_percentages.head())
```

```
   31.05.2019  30.06.2019  31.07.2019  31.08.2019  30.09.2019  31.10.2019  \
0    0.144037    0.332597    0.332762    0.564535    0.266077    0.249447
1    0.000000    0.128147    0.090877    0.094366    0.098992    0.105410
2    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
3    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
4    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000

   30.11.2019  31.12.2019  31.01.2020  29.02.2020  31.03.2020  30.04.2020  \
0    0.232818    0.216188    0.199558    0.182928    0.166298    0.149668
```

```
1    0.114918    0.130549    0.161792    0.271782    0.064074    0.057666
2    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
3    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
4    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000

   31.05.2020  30.06.2020  31.07.2020  31.08.2020  30.09.2020  31.10.2020  \
0    0.133039    0.116409    0.099779    0.083149    0.066519    0.049889
1    0.051259    0.044851    0.038444    0.032037    0.025629    0.019222
2    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
3    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
4    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000

   30.11.2020  31.12.2020
0    0.033260    0.016630
1    0.012815    0.006407
2    0.000000    0.000000
3    0.000000    0.000000
4    0.000000    0.000000
```

In [78]:

```python
#Function to calculate cash flows

def calculate_forecasted_cash_flows(df, expected_repayment_percentages):
    cash_flows = {}


    expected_repayment_percentages.columns = [str(col) for col in expected_repayment_per


    for index, row in df.iterrows():
        vintage = str(row['Dates'])
        origination_amount = row['Origination Amount']


        if vintage in expected_repayment_percentages.columns:
            expected_percentages = expected_repayment_percentages[vintage]


            forecasted_flows = expected_percentages.multiply(origination_amount)
            cash_flows[vintage] = forecasted_flows
        else:
            print(f"Warning: {vintage} not found in expected_repayment_percentages colum


    forecasted_df = pd.DataFrame(cash_flows).T


    print("Forecasted Cash Flows DataFrame:")
    print(forecasted_df.head())

    return forecasted_df
```

In [81]:

```python
# Recalculate the forecasted cash flows
forecasted_cash_flows = calculate_forecasted_cash_flows(df, expected_repayment_percentag
forecasted_cash_flows
```

```
Forecasted Cash Flows DataFrame:
                     0             1    2    3    4    5    6    7    8   \
31.05.2019  1.443069e+06  0.000000e+00  0.0  0.0  0.0  0.0  0.0  0.0  0.0
30.06.2019  3.614785e+06  1.392752e+06  0.0  0.0  0.0  0.0  0.0  0.0  0.0
31.07.2019  3.571847e+06  9.754662e+05  0.0  0.0  0.0  0.0  0.0  0.0  0.0
31.08.2019  7.089842e+06  1.185119e+06  0.0  0.0  0.0  0.0  0.0  0.0  0.0
30.09.2019  3.859469e+06  1.435887e+06  0.0  0.0  0.0  0.0  0.0  0.0  0.0

             9    10   11   12   13   14   15   16   17   18   19
31.05.2019  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0
```

30.06.2019 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
31.07.2019 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
31.08.2019 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
30.09.2019 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

Out[81]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
31.05.2019	1.443069e+06	0.000000e+00	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30.06.2019	3.614785e+06	1.392752e+06	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
31.07.2019	3.571847e+06	9.754662e+05	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
31.08.2019	7.089842e+06	1.185119e+06	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30.09.2019	3.859469e+06	1.435887e+06	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
31.10.2019	3.904588e+06	1.649975e+06	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30.11.2019	3.517341e+06	1.736144e+06	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
31.12.2019	3.676218e+06	2.219958e+06	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
31.01.2020	3.351452e+06	2.717195e+06	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
29.02.2020	3.515367e+06	5.222898e+06	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
31.03.2020	3.596715e+06	1.385789e+06	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30.04.2020	3.342804e+06	1.287959e+06	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
31.05.2020	2.979180e+06	1.147857e+06	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30.06.2020	2.871998e+06	1.106561e+06	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
31.07.2020	2.542192e+06	9.794887e+05	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
31.08.2020	2.151802e+06	8.290742e+05	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30.09.2020	1.721996e+06	6.634731e+05	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
31.10.2020	1.381918e+06	5.324434e+05	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30.11.2020	9.935620e+05	3.828125e+05	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
31.12.2020	5.069267e+05	1.953153e+05	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

In [82]:
```
#writing function to calculate discount rate
def discount_cash_flows(cash_flows, annual_rate):
    monthly_rate = annual_rate / 12 / 100
    discounted_cash_flows = {}
    for vintage, flows in cash_flows.items():
        discounted_flows = [cf / (1 + monthly_rate) ** i for i, cf in enumerate(flows, s
        discounted_cash_flows[vintage] = discounted_flows
    return discounted_cash_flows
```

In [83]:
```
# Computing present value of the portfolio
annual_discount_rate = 2.5


discounted_cash_flows = discount_cash_flows(forecasted_cash_flows, annual_discount_rate)

portfolio_value = sum([sum(flows) for flows in discounted_cash_flows.values()])


print(f"Calculated Portfolio Value: CHF {portfolio_value:.2f}")
```

```
Calculated Portfolio Value: CHF 85076306.30
```

In [84]:
```
#comparing with client estimate
```

```
client_estimate = 84993122.67


difference = portfolio_value - client_estimate
relative_difference = difference / client_estimate


acceptable = abs(difference) < 500000

print(f"Client's Estimate: CHF {client_estimate:.2f}")
print(f"Difference: CHF {difference:.2f}")
print(f"Relative Difference: {relative_difference:.4%}")
print(f"Difference is acceptable: {acceptable}")
```

```
Client's Estimate: CHF 84993122.67
Difference: CHF 83183.63
Relative Difference: 0.0979%
Difference is acceptable: True
```