[4-1. 현금흐름 계산1]

1. 필요한 모듈 import 및 기본 설정

1-1. 필요한 모듈 import

- 기존 작성한 모듈 및 필요 파일 확인
  - FCHotel_FSmodeling_assumption.xlsx
  - m00_general_function.py
  - m01_assumption.py
  - m02_index.py
  - m03_funding.py
  - m04_operating_income.py
  - m05_operating_cost.py
  - m06_facility_cost.py

In [ ]:
```
import pandas as pd
pd.set_option('display.max_rows', 30)
pd.set_option('display.max_columns', 100)
pd.set_option('display.max_colwidth', 20)
pd.set_option('display.width', 300)

# DataFrame의 출력을 확장하여 한 줄로 계속 출력되도록 설정
pd.set_option('display.expand_frame_repr', True)

from m01_assumption import assumption
from m02_index import index
from m03_funding import funding
from m04_operating_income import operating_income
from m05_operating_cost import operating_cost
from m06_facility_cost import facility_cost
```

1-2. 기본 Cashflow 데이터프레임 설정

- cashflow :
  - 'date', 'categoryA', 'categoryB', 'categoryC', '입금금액', '출금금액'을 컬럼명으로 하여 빈 DataFrame 설정
  - 날짜별 현금흐름의 내용을 기록
- balance :
  - '기초현금', '입금금액', '출금금액', '기말현금'을 컬럼명으로 하고, 'model index'를 index로 설정하여 DataFrame 설정.
  - 값은 모두 0으로 설정
  - 날짜별로 기초현금, 입금금액 합계, 출금금액 합계, 기말현금을 기록

In [ ]:
```
cashflow = pd.DataFrame(
    columns = ['date', 'categoryA', 'categoryB', 'categoryC', '입금금액', '출금금액']
)

balance = pd.DataFrame({
    '기초현금': [0] * len(index['model']),
    '입금금액': [0] * len(index['model']),
    '출금금액': [0] * len(index['model']),
    '기말현금': [0] * len(index['model']),
},
    index=index['model']
)
```

In [ ]:
```
room_type_list = list(assumption['business_overview']['객실수'].keys())
```

2. 현금흐름: 현금흐름 기록 틀잡기

In [ ]:
```
idx = 0
cash_balance = 0
for dt in index['model']:
    #### 0. 기초현금 계산
    balance.loc[dt, '기초현금'] = cash_balance

    #### Test(입금)
    amount = 2000
    cashflow.loc[idx] = [dt, '입금', 'test', '입금금액', amount, 0]
    balance.loc[dt, '입금금액'] += amount
    idx += 1

    #### Test(출금)
    amount = 1000
    cashflow.loc[idx] = [dt, '출금', 'test', '출금금액', 0, amount]
    balance.loc[dt, '출금금액'] += 1000
    idx += 1

    #### 9. 기말현금 계산
    cash_balance = (
        balance.loc[dt, '기초현금'] + balance.loc[dt, '입금금액'] - balance.loc[dt, '출금금액']
    )
    balance.loc[dt, '기말현금'] = cash_balance
```

In [ ]:
```
cashflow
```

Out[ ]:

	date	categoryA	categoryB	categoryC	입금금액	출금금액
0	2023-12-31	입금	test	입금금액	2000	0
1	2023-12-31	출금	test	출금금액	0	1000
2	2024-01-31	입금	test	입금금액	2000	0
3	2024-01-31	출금	test	출금금액	0	1000
4	2024-02-29	입금	test	입금금액	2000	0
...	...	...	...	...	...	...
71	2026-11-30	출금	test	출금금액	0	1000
72	2026-12-31	입금	test	입금금액	2000	0
73	2026-12-31	출금	test	출금금액	0	1000
74	2027-01-31	입금	test	입금금액	2000	0
75	2027-01-31	출금	test	출금금액	0	1000

76 rows × 6 columns

In [ ]:
```
balance
```

Out[ ]:

	기초현금	입금금액	출금금액	기말현금
2023-12-31	0	2000	1000	1000
2024-01-31	1000	2000	1000	2000
2024-02-29	2000	2000	1000	3000
2024-03-31	3000	2000	1000	4000
2024-04-30	4000	2000	1000	5000
...	...	...	...	...
2026-09-30	33000	2000	1000	34000
2026-10-31	34000	2000	1000	35000
2026-11-30	35000	2000	1000	36000
2026-12-31	36000	2000	1000	37000
2027-01-31	37000	2000	1000	38000

38 rows × 4 columns

3. 자금의 조달 및 소요 현금흐름 작성

3-1. funding 객체 확인

In [ ]:
```
funding.keys()
```

Out[ ]:
```
dict_keys(['자기자본', '차입금', '자산매입'])
```

In [ ]:
```
funding['자기자본']
```

Out[ ]:

	자기자본유입	배당금지급
2023-12-31	10000000000	0
2024-01-31	0	0
2024-02-29	0	0
2024-03-31	0	0
2024-04-30	0	0
...	...	...
2026-09-30	0	0
2026-10-31	0	0
2026-11-30	0	0
2026-12-31	0	0
2027-01-31	0	0

38 rows × 2 columns

3-2. funding 객체의 현금흐름 반영

In [ ]:
```
#### cashflow, balance 객체 초기화 ####
cashflow = pd.DataFrame(
    columns = ['date', 'categoryA', 'categoryB', 'categoryC', '입금금액', '출금금액']
)

balance = pd.DataFrame({
    '기초현금': [0] * len(index['model']),
    '입금금액': [0] * len(index['model']),
    '출금금액': [0] * len(index['model']),
```

```
    '기말현금': [0] * len(index['model']),
},
    index=index['model']
)
room_type_list = list(assumption['business_overview']['객실수'].keys())


#### cashflow, balance 작성 ####
idx = 0
cash_balance = 0
for dt in index['model']:
    #### 0. 기초현금 계산
    balance.loc[dt, '기초현금'] = cash_balance

    #### 1. 자금조달소요
    ## 1-1. 자기자본 유입
    amount = funding['자기자본'].loc[dt, '자기자본유입']
    if amount > 0:
        cashflow.loc[idx] = [dt, '자금조달', '자기자본', '자기자본유입', amount, 0]
        balance.loc[dt, '입금금액'] += amount
        idx += 1

    ## 1-2. 차입금 유입
    amount = funding['차입금'].loc[dt, '차입금유입']
    if amount > 0:
        cashflow.loc[idx] = [dt, '자금조달', '차입금', '차입금유입', amount, 0]
        balance.loc[dt, '입금금액'] += amount
        idx += 1

    ## 1-3. 자산매입
    amount = funding['자산매입'].loc[dt, '자산매입']
    if amount > 0:
        cashflow.loc[idx] = [dt, '자산매입', '자산매입', '매입대금지출', 0, amount]
        balance.loc[dt, '출금금액'] += amount
        idx += 1
```

```
    amount = funding['자산매입'].loc[dt, '매입부수비용']
    if amount > 0:
        cashflow.loc[idx] = [dt, '자산매입', '매입부수비용', '부수비용지출', 0, amount]
        balance.loc[dt, '출금금액'] += amount
        idx += 1


    #### 9. 기말현금 계산
    cash_balance = (
        balance.loc[dt, '기초현금'] + balance.loc[dt, '입금금액'] - balance.loc[dt, '출금금액']
    )
    balance.loc[dt, '기말현금'] = cash_balance
```

3-3. 현금흐름 반영 결과 확인

In [ ]:
```
cashflow
```

Out[ ]:

	date	categoryA	categoryB	categoryC	입금금액	출금금액
0	2023-12-31	자금조달	자기자본	자기자본유입	10000000000	0
1	2023-12-31	자금조달	차입금	차입금유입	10000000000	0
2	2023-12-31	자산매입	자산매입	매입대금지출	0	18000000000
3	2023-12-31	자산매입	매입부수비용	부수비용지출	0	1000000000

In [ ]:
```
balance
```

Out[ ]:

	기초현금	입금금액	출금금액	기말현금
2023-12-31	0	20000000000	19000000000	1000000000
2024-01-31	1000000000	0	0	1000000000
2024-02-29	1000000000	0	0	1000000000
2024-03-31	1000000000	0	0	1000000000
2024-04-30	1000000000	0	0	1000000000
...	...	...	...	...
2026-09-30	1000000000	0	0	1000000000
2026-10-31	1000000000	0	0	1000000000
2026-11-30	1000000000	0	0	1000000000
2026-12-31	1000000000	0	0	1000000000
2027-01-31	1000000000	0	0	1000000000

38 rows × 4 columns

In [ ]: