

[2-4. 미결제정 데이터 리포팅]

1. 샘플 데이터 추가

```
In [ ]: import pandas as pd

        from pending_account import TransactionManager, CashflowManager
```

```
In [ ]: tm = TransactionManager()
        cm = CashflowManager()
```

```
In [ ]: tm.create(
        account_name = "미지급금",
        client_name = "AA사",
        transaction_date = "2024-01-05",
        amount_receivable = 0,
        amount_payable = 2_000_000,
        settlement_status = None,
        notes = None
    )
    tm.create(
        account_name = "미지급금",
        client_name = "BB사",
        transaction_date = "2024-01-25",
        amount_receivable = 0,
        amount_payable = 1_000_000,
        settlement_status = None,
        notes = None
    )
```

```
Out[ ]: 4
```

```
In [ ]: cm.create(
        transaction_id = 3,
        dw_date = "2024-01-20",
        deposit_amount = 0,
        withdrawal_amount = 1_000_000,
        notes = None
    )
    cm.create(
        transaction_id = 3,
        dw_date = "2024-01-30",
        deposit_amount = 0,
        withdrawal_amount = 500_000,
        notes = None
    )
```

Out[]: 8

```
In [ ]: cm.create(
    transaction_id = 4,
    dw_date = "2024-01-30",
    deposit_amount = 0,
    withdrawal_amount = 300_000,
    notes = None
)
```

Out[]: 9

2. 미결제정 잔액 계산

2-1. 데이터 로딩

```
In [ ]: tmdf = pd.DataFrame(tm.read_all())
        cmdf = pd.DataFrame(cm.read_all())
```

2-1-1. Transaction records table

```
In [ ]: print(tmdf)
```

	id	account_name	client_name	transaction_date	amount_receivable	\
0	1	외상매출금	ABC전자	2024-01-10	1000000	
1	2	외상매출금	AG상사	2024-01-10	2000000	
2	3	미지급금	AA사	2024-01-05	0	
3	4	미지급금	BB사	2024-01-25	0	

	amount_payable	settlement_status	notes
0	0	None	None
1	0	None	None
2	2000000	None	None
3	1000000	None	None

2-1-2. Cashflow records table

```
In [ ]: print(cmdf)
```

	id	transaction_id	dw_date	deposit_amount	withdrawal_amount	note
s						
0	1	1	2024-01-20	500000	0	Non
e						
1	2	1	2024-01-30	500000	0	Non
e						
2	3	2	2024-01-15	1000000	0	Non
e						
3	4	2	2024-01-25	500000	0	Non
e						
4	7	3	2024-01-20	0	1000000	Non
e						
5	8	3	2024-01-30	0	500000	Non
e						
6	9	4	2024-01-30	0	300000	Non
e						

2-2. 미결제잔액 조회 코드 작성

2-2-1. 프롬프트 작성

tmdf는 아래의 내용을 저장하고 있는 DataFrame이야.

```
id account_name client_name transaction_date
amount_receivable \
0 1 외상매출금 ABC전자 2024-01-10 1000000
1 2 외상매출금 AG상사 2024-01-10 2000000
```

```
amount_payable settlement_status notes
0 0 None None
1 0 None None
```

cmdf는 아래의 내용을 저장하고 있는 DataFrame이야.

```
id transaction_id dw_date deposit_amount withdrawal_amount
notes
0 1 1 2024-01-20 500000 0 None
1 2 1 2024-01-30 500000 0 None
2 3 2 2024-01-15 1000000 0 None
3 4 2 2024-01-25 500000 0 None
```

이때, 아래 컬럼 구조로 amount_receivable 잔액과 amount_payable 잔액을 확인할 수 있는 DataFrame을 만들고 싶어.

```
id, account_name, client_name, transaction_date,
amount_receivable, amount_received, receivable_remained,
amount_payable, amount_paid, payable_remained,
settlement_status, notes
```

이렇게 DataFrame을 만들도록 python 코드 작성해줘.

2-2-2. ChatGPT 답변



주어진 `tmdf`와 `cmdf`를 사용하여 새로운 DataFrame을 만드는 코드는 다음과 같습니다:

python

코드 복사

```
import pandas as pd

# 초기 데이터
tmdf = pd.DataFrame({
    'id': [1, 2],
    'account_name': ['외상매출금', '외상매출금'],
    'client_name': ['ABC전자', 'AG상사'],
    'transaction_date': ['2024-01-10', '2024-01-10'],
    'amount_receivable': [1000000, 2000000],
    'amount_payable': [0, 0],
    'settlement_status': [None, None],
    'notes': [None, None]
})

cmdf = pd.DataFrame({
    'id': [1, 2, 3, 4],
    'transaction_id': [1, 1, 2, 2],
    'dw_date': ['2024-01-20', '2024-01-30', '2024-01-15', '2024-01-25'],
    'deposit_amount': [500000, 500000, 1000000, 500000],
    'withdrawal_amount': [0, 0, 0, 0],
    'notes': [None, None, None, None]
})

# 받은 금액을 계산
amount_received = cmdf.groupby('transaction_id')['deposit_amount'].sum().reset_index()
amount_received.columns = ['id', 'amount_received']

# 거래 DataFrame에 받은 금액을 병합
result_df = tmdf.merge(amount_received, on='id', how='left')
```

```

# 지급된 금액을 계산
amount_paid = cmdf.groupby('transaction_id')['withdrawal_amount'].sum().reset_index()
amount_paid.columns = ['id', 'amount_paid']

# 거래 DataFrame에 지급된 금액을 병합
result_df = result_df.merge(amount_paid, on='id', how='left')

# 결산된 금액 계산
result_df['amount_received'] = result_df['amount_received'].fillna(0).astype(int)
result_df['amount_paid'] = result_df['amount_paid'].fillna(0).astype(int)

# 미수/미지급 잔액 계산
result_df['receivable_remained'] = result_df['amount_receivable'] - result_df['amount_paid']
result_df['payable_remained'] = result_df['amount_payable'] - result_df['amount_paid']

# 원하는 컬럼 순서로 재배열
result_df = result_df[['id', 'account_name', 'client_name', 'transaction_date',
                        'amount_receivable', 'amount_received', 'receivable_remained',
                        'amount_payable', 'amount_paid', 'payable_remained',
                        'settlement_status', 'notes']]

print(result_df)

```

이 코드는 `tmddf`와 `cmdf`에서 데이터를 병합하여 새로운 DataFrame을 생성합니다. 여기서는 `amount_received`와 `amount_paid`를 계산하고, 이를 기반으로 `receivable_remained`와 `payable_remained`를 계산한 후, 원하는 컬럼 순서로 정리합니다.

2-2-3. ChatGPT 코드

```

import pandas as pd

# 초기 데이터
tmddf = pd.DataFrame({
    'id': [1, 2],
    'account_name': ['외상매출금', '외상매출금'],
    'client_name': ['ABC전자', 'AG상사'],
    'transaction_date': ['2024-01-10', '2024-01-10'],
    'amount_receivable': [1000000, 2000000],
    'amount_payable': [0, 0],
    'settlement_status': [None, None],
    'notes': [None, None]
})

cmdf = pd.DataFrame({
    'id': [1, 2, 3, 4],
    'transaction_id': [1, 1, 2, 2],
    'dw_date': [
        '2024-01-20', '2024-01-30', '2024-01-15', '2024-
01-25'
    ],

```

```
        'deposit_amount': [500000, 500000, 1000000, 500000],
        'withdrawal_amount': [0, 0, 0, 0],
        'notes': [None, None, None, None]
    })

# 받은 금액을 계산
amount_received = cmdf.groupby('transaction_id')[
    'deposit_amount'
].sum().reset_index()
amount_received.columns = ['id', 'amount_received']

# 거래 DataFrame에 받은 금액을 병합
result_df = tmdf.merge(amount_received, on='id',
    how='left')

# 지급된 금액을 계산
amount_paid = cmdf.groupby('transaction_id')[
    'withdrawal_amount'
].sum().reset_index()
amount_paid.columns = ['id', 'amount_paid']

# 거래 DataFrame에 지급된 금액을 병합
result_df = result_df.merge(amount_paid, on='id',
    how='left')

# 결산된 금액 계산
result_df['amount_received'] = result_df[
    'amount_received'
].fillna(0).astype(int)
result_df['amount_paid'] = result_df[
    'amount_paid'
].fillna(0).astype(int)

# 미수/미지급 잔액 계산
result_df['receivable_remained'] = (
    result_df['amount_receivable'] -
    result_df['amount_received']
)
result_df['payable_remained'] = (
    result_df['amount_payable'] -
    result_df['amount_paid']
)

# 원하는 컬럼 순서로 재배열
result_df = result_df[[
    'id', 'account_name', 'client_name',
    'transaction_date',
    'amount_receivable', 'amount_received',
    'receivable_remained',
    'amount_payable', 'amount_paid',
    'payable_remained',
```

```

        'settlement_status', 'notes'
    ]]

    print(result_df)

```

```

In [ ]: # 받은 금액을 계산
amount_received = cmdf.groupby('transaction_id')[
    'deposit_amount'
].sum().reset_index()
amount_received.columns = ['id', 'amount_received']

# 거래 DataFrame에 받은 금액을 병합
result_df = tmdf.merge(amount_received, on='id', how='left')

# 지급된 금액을 계산
amount_paid = cmdf.groupby('transaction_id')[
    'withdrawal_amount'
].sum().reset_index()
amount_paid.columns = ['id', 'amount_paid']

# 거래 DataFrame에 지급된 금액을 병합
result_df = result_df.merge(amount_paid, on='id', how='left')

# 결산된 금액 계산
result_df['amount_received'] = result_df[
    'amount_received'
].fillna(0).astype(int)
result_df['amount_paid'] = result_df[
    'amount_paid'
].fillna(0).astype(int)

# 미수/미지급 잔액 계산
result_df['receivable_remained'] = (
    result_df['amount_receivable'] - result_df['amount_received']
)
result_df['payable_remained'] = (
    result_df['amount_payable'] - result_df['amount_paid']
)

# 원하는 컬럼 순서로 재배열
result_df = result_df[[
    'id', 'account_name', 'client_name', 'transaction_date',
    'amount_receivable', 'amount_received', 'receivable_remained',
    'amount_payable', 'amount_paid', 'payable_remained',
    'settlement_status', 'notes'
]]

print(result_df)

```

	id	account_name	client_name	transaction_date	amount_receivable	\
0	1	외상매출금	ABC전자	2024-01-10	1000000	
1	2	외상매출금	AG상사	2024-01-10	2000000	
2	3	미지급금	AA사	2024-01-05	0	
3	4	미지급금	BB사	2024-01-25	0	

	amount_received	receivable_remained	amount_payable	amount_paid	\
0	1000000	0	0	0	
1	1500000	500000	0	0	
2	0	0	2000000	1500000	
3	0	0	1000000	300000	

	payable_remained	settlement_status	notes
0	0	None	None
1	0	None	None
2	500000	None	None
3	700000	None	None

```
In [ ]: result_df.to_excel("미결제정 잔액.xlsx")
```

3. 데이터 리포팅 모듈 작성

3-1. 데이터 로딩 및 미결제잔액 계산 함수 작성

```
In [ ]: import pandas as pd
from connector import Connector

with Connector() as db:
    query = "SELECT * FROM transaction_records"
    db.cursor.execute(query)
    tmdf = pd.DataFrame(db.cursor.fetchall())

    query = "SELECT * FROM cashflow_records"
    db.cursor.execute(query)
    cmdf = pd.DataFrame(db.cursor.fetchall())

    # 받은 금액을 계산
    amount_received = cmdf.groupby('transaction_id')[
        'deposit_amount'
    ].sum().reset_index()
    amount_received.columns = ['id', 'amount_received']

    # 거래 DataFrame에 받은 금액을 병합
    result_df = tmdf.merge(amount_received, on='id', how='left')

    # 지급된 금액을 계산
    amount_paid = cmdf.groupby('transaction_id')[
        'withdrawal_amount'
    ].sum().reset_index()
    amount_paid.columns = ['id', 'amount_paid']
```



```

# 거래 DataFrame에 지급된 금액을 병합
result_df = result_df.merge(amount_paid, on='id', how='left')

# 결산된 금액 계산
result_df['amount_received'] = result_df[
    'amount_received'
].fillna(0).astype(int)
result_df['amount_paid'] = result_df[
    'amount_paid'
].fillna(0).astype(int)

# 미수/미지급 잔액 계산
result_df['receivable_remained'] = (
    result_df['amount_receivable'] - result_df['amount_received']
)
result_df['payable_remained'] = (
    result_df['amount_payable'] - result_df['amount_paid']
)

# 원하는 컬럼 순서로 재배열
result_df = result_df[[
    'id', 'account_name', 'client_name', 'transaction_date',
    'amount_receivable', 'amount_received', 'receivable_remained',
    'amount_payable', 'amount_paid', 'payable_remained',
    'settlement_status', 'notes'
]]

```

3-2. "status" 설정 코드 작성

```

In [ ]: for idx in result_df.index:
        if result_df.loc[idx, 'receivable_remained'] > 0:
            result_df.loc[idx, 'settlement_status'] = '미수'
        elif result_df.loc[idx, 'payable_remained'] > 0:
            result_df.loc[idx, 'settlement_status'] = '미지급'
        else:
            result_df.loc[idx, 'settlement_status'] = '완료'

```

```

In [ ]: result_df

```

```

Out[ ]:
   id  account_name  client_name  transaction_date  amount_receivable  amount_receive
0   1      외상매출금      ABC전자      2024-01-10           1000000           100000
1   2      외상매출금      AG상사      2024-01-10           2000000           150000
2   3      미지급금        AA사      2024-01-05                0
3   4      미지급금        BB사      2024-01-25                0

```

3-3. 클래스 작성

```

In [ ]: import pandas as pd

```

```

from connector import Connector

class ReportingManager:
    def pending_report(self):
        with Connector() as db:
            query = "SELECT * FROM transaction_records"
            db.cursor.execute(query)
            tmdf = pd.DataFrame(db.cursor.fetchall())

            query = "SELECT * FROM cashflow_records"
            db.cursor.execute(query)
            cmdf = pd.DataFrame(db.cursor.fetchall())

            # 받은 금액을 계산
            amount_received = cmdf.groupby('transaction_id')[
                'deposit_amount'
            ].sum().reset_index()
            amount_received.columns = ['id', 'amount_received']

            # 거래 DataFrame에 받은 금액을 병합
            result_df = tmdf.merge(amount_received, on='id', how='left')

            # 지급된 금액을 계산
            amount_paid = cmdf.groupby('transaction_id')[
                'withdrawal_amount'
            ].sum().reset_index()
            amount_paid.columns = ['id', 'amount_paid']

            # 거래 DataFrame에 지급된 금액을 병합
            result_df = result_df.merge(amount_paid, on='id', how='left')

            # 결산된 금액 계산
            result_df['amount_received'] = result_df[
                'amount_received'
            ].fillna(0).astype(int)
            result_df['amount_paid'] = result_df[
                'amount_paid'
            ].fillna(0).astype(int)

            # 미수/미지급 잔액 계산
            result_df['receivable_remained'] = (
                result_df['amount_receivable'] - result_df['amount_receiv
            )
            result_df['payable_remained'] = (
                result_df['amount_payable'] - result_df['amount_paid']
            )

            # 원하는 컬럼 순서로 재배열
            result_df = result_df[[
                'id', 'account_name', 'client_name', 'transaction_date',
                'amount_receivable', 'amount_received', 'receivable_remai
                'amount_payable', 'amount_paid', 'payable_remained',
                'settlement_status', 'notes'
            ]]

```

```
# settlement_status 설정
for idx in result_df.index:
    if result_df.loc[idx, 'receivable_remained'] > 0:
        result_df.loc[idx, 'settlement_status'] = '미수'
    elif result_df.loc[idx, 'payable_remained'] > 0:
        result_df.loc[idx, 'settlement_status'] = '미지급'
    else:
        result_df.loc[idx, 'settlement_status'] = '완료'

return result_df
```

```
In [ ]: rm = ReportingManager()
```

```
In [ ]: rm.pending_report()
```

```
Out [ ]:
```

	id	account_name	client_name	transaction_date	amount_receivable	amount_receive
0	1	외상매출금	ABC전자	2024-01-10	1000000	100000
1	2	외상매출금	AG상사	2024-01-10	2000000	150000
2	3	미지급금	AA사	2024-01-05	0	
3	4	미지급금	BB사	2024-01-25	0	

3-4. 모듈에 클래스 추가

- pending_account 모듈에 ReportingManager 클래스 추가
- pending_account2.py 생성 후 위에서 작성한 코드 추가

```
In [ ]: from pending_account2 import ReportingManager as RM
rm2 = RM()
```

3-4. 결과데이터 엑셀 출력

```
In [ ]: rm2.pending_report()
```

```
Out [ ]:
```

	id	account_name	client_name	transaction_date	amount_receivable	amount_receive
0	1	외상매출금	ABC전자	2024-01-10	1000000	100000
1	2	외상매출금	AG상사	2024-01-10	2000000	150000
2	3	미지급금	AA사	2024-01-05	0	
3	4	미지급금	BB사	2024-01-25	0	

```
In [ ]: rm2.pending_report().to_excel("미결제정 잔액2.xlsx")
```

In []: