# [ 3-4. 데이터 조회 모듈 확장 ]

## 1. 기존 코드 리뷰

### 1-1. 엑셀파일 로딩 코드

In [ ]:
```
import pandas as pd

pd.set_option('display.max_rows', 20)
pd.set_option('display.max_columns', 100)
pd.set_option('display.max_colwidth', 50)
pd.set_option('display.width', 300)
pd.set_option('display.expand_frame_repr', True)

from pandas import DataFrame, Series
from datetime import datetime, date
from openpyxl import load_workbook


colname_row_number = 4
start_row_number = 5
last_row_value = "합 계"


filename = "분개장_샘플.xlsx"

wb = load_workbook(filename)
ws = wb.active

colnames = [x.value for x in ws[colname_row_number]]

datalist = []
for row in ws.iter_rows(
        min_row=start_row_number,
        max_col=len(colnames),
        values_only=True
    ):
    if row[0] == last_row_value:
        break
    row = list(row)

    if row[0] is not None:
        일자 = row[0].split('/')
        일자 = date(2024, int(일자[0]), int(일자[1]))
        전표번호 = row[1]

    row[0] = 일자
    row[1] = 전표번호
    if pd.isna(row[5]):
        row[5] = 0
```

```
    if pd.isna(row[6]):
        row[6] = 0

    datalist.append(row)
journal = DataFrame(datalist, columns=colnames)

category_table = pd.DataFrame([
        ['자본금', '자본', '자본금', '자본금'],
        ['보통예금', '유동자산', '현금및현금성자산', '보통예금'],
        ['객실수입', '매출', '매출', '객실수입'],
        ['고객용품비', '판매비와관리비', '고객용품비', '고객용품비'],
        ['기타영업비용', '판매비와관리비', '판매비용', '기타영업비용'],
        ['판매수수료', '판매비와관리비', '판매비용', '판매수수료'],
        ['세금과공과', '판매비와관리비', '기타판관비', '세금과공과'],
        ['수도광열비', '판매비와관리비', '기타판관비', '수도광열비'],
        ['급여', '판매비와관리비', '인건비', '급여'],
        ['노무용역비', '판매비와관리비', '인건비', '노무용역비'],
        ['이자수입', '영업외수익', '영업외수익', '이자수입']
    ],
    columns=['구분', '대분류', '중분류', '소분류']
)
category_table.set_index('구분', inplace=True)

journal['대분류'] = None
journal['중분류'] = None
journal['소분류'] = None

for idx in journal.index:
    # journal 데이터프레임의 "계정과목" 컬럼 값 추출
    key = journal.loc[idx, '계정과목']

    # category_table에서 추출된 키값에 해당하는 "대분류", "중분류", "소분류" 값을
    # journal에 대체
    journal.loc[idx, '대분류'] = category_table.loc[key, '대분류']
    journal.loc[idx, '중분류'] = category_table.loc[key, '중분류']
    journal.loc[idx, '소분류'] = category_table.loc[key, '소분류']

journal = journal[
    [
        '일자', '전표번호', '대분류', '중분류', '소분류', '계정코드',
        '계정과목', '적요', '차변', '대변', '구분', '거래처명'
        ]
    ]
```

## 1-2. 데이터 조회 모듈 코드

In [ ]:

```
class Journal:
    def __init__(self, journal_df):
        self.df = journal_df
        self.전표번호 = self.전표번호(self)
        self.대분류 = self.대분류(self)
        self.중분류 = self.중분류(self)
        self.소분류 = self.소분류(self)
        self.계정코드 = self.계정코드(self)
        self.계정과목 = self.계정과목(self)
```

```
        self.거래처명 = self.거래처명(self)

    class 전표번호:
        def __init__(self, parent_instance):
            self.parent_instance = parent_instance
            self.attr_name = "전표번호"

        def __call__(self, lookup_value):
            df = self.parent_instance.df
            filtered_df = df[df[self.attr_name] == lookup_value]
            return Journal(filtered_df)

    class 대분류:
        def __init__(self, parent_instance):
            self.parent_instance = parent_instance
            self.attr_name = "대분류"

        def __call__(self, lookup_value):
            df = self.parent_instance.df
            filtered_df = df[df[self.attr_name] == lookup_value]
            return Journal(filtered_df)

    class 중분류:
        def __init__(self, parent_instance):
            self.parent_instance = parent_instance
            self.attr_name = "중분류"

        def __call__(self, lookup_value):
            df = self.parent_instance.df
            filtered_df = df[df[self.attr_name] == lookup_value]
            return Journal(filtered_df)

    class 소분류:
        def __init__(self, parent_instance):
            self.parent_instance = parent_instance
            self.attr_name = "소분류"

        def __call__(self, lookup_value):
            df = self.parent_instance.df
            filtered_df = df[df[self.attr_name] == lookup_value]
            return Journal(filtered_df)

    class 계정코드:
        def __init__(self, parent_instance):
            self.parent_instance = parent_instance
            self.attr_name = "계정코드"

        def __call__(self, lookup_value):
            df = self.parent_instance.df
            filtered_df = df[df[self.attr_name] == lookup_value]
            return Journal(filtered_df)

    class 계정과목:
        def __init__(self, parent_instance):
            self.parent_instance = parent_instance
            self.attr_name = "계정과목"
```

```
        def __call__(self, lookup_value):
            df = self.parent_instance.df
            filtered_df = df[df[self.attr_name] == lookup_value]
            return Journal(filtered_df)

    class 거래처명:
        def __init__(self, parent_instance):
            self.parent_instance = parent_instance
            self.attr_name = "거래처명"

        def __call__(self, lookup_value):
            df = self.parent_instance.df
            filtered_df = df[df[self.attr_name] == lookup_value]
            return Journal(filtered_df)
```

# 2. 데이터조회 모듈 반복 코드 수정

## 2-1. ChatGPT 질의 회신

- 요청 프롬프트 : 이 코드에 반복되는 코드를 축약해서 간결하게 다시 작성해줘.

반복되는 코드를 축약하여 보다 간결하게 작성할 수 있습니다. 공통된 기능을 하는 부분을 하나의 메서드로 통합하면 코드의 재사용성이 높아지고 유지보수가 용이해집니다. 아래는 이를 반영한 코드입니다.

python 코드 복사

```python
class Journal:
    def __init__(self, journal_df):
        self.df = journal_df
        self.전표번호 = self.Filter(self, "전표번호")
        self.대분류 = self.Filter(self, "대분류")
        self.중분류 = self.Filter(self, "중분류")
        self.소분류 = self.Filter(self, "소분류")
        self.계정코드 = self.Filter(self, "계정코드")
        self.계정과목 = self.Filter(self, "계정과목")
        self.거래처명 = self.Filter(self, "거래처명")

    class Filter:
        def __init__(self, parent_instance, attr_name):
            self.parent_instance = parent_instance
            self.attr_name = attr_name

        def __call__(self, lookup_value):
            df = self.parent_instance.df
            filtered_df = df[df[self.attr_name] == lookup_value]
            return Journal(filtered_df)
```

### 설명

1. **Filter 클래스**:
   - `Filter` 클래스는 필터링 기능을 담당하며, `parent_instance`와 `attr_name`을 초기화합니다.
   - `__call__` 메서드를 통해 특정 열(`attr_name`)에서 `lookup_value`와 일치하는 행들을 필터링하고 새로운 `Journal` 인스턴스를 반환합니다.
2. **Journal 클래스**:
   - `Journal` 클래스는 입력 데이터프레임(`journal_df`)을 저장합니다.
   - 다양한 필터링 속성(`전표번호`, `대분류`, `중분류`, `소분류`, `계정코드`, `계정과목`, `거래처명`)을 `Filter` 클래스를 사용하여 초기화합니다.
   - 각 필터링 속성은 해당 열 이름을 인수로 받아 `Filter` 클래스의 인스턴스를 생성합니다.

In [ ]:
```python
class Journal:
    def __init__(self, journal_df):
        self.df = journal_df
        self.전표번호 = self.Filter(self, "전표번호")
        self.대분류 = self.Filter(self, "대분류")
        self.중분류 = self.Filter(self, "중분류")
        self.소분류 = self.Filter(self, "소분류")
        self.계정코드 = self.Filter(self, "계정코드")
        self.계정과목 = self.Filter(self, "계정과목")
        self.거래처명 = self.Filter(self, "거래처명")

    class Filter:
        def __init__(self, parent_instance, attr_name):
            self.parent_instance = parent_instance
            self.attr_name = attr_name

        def __call__(self, lookup_value):
            df = self.parent_instance.df
            filtered_df = df[df[self.attr_name] == lookup_value]
            return Journal(filtered_df)
```

In [ ]:
```python
# 사용 예시
jnl = Journal(journal)
jnl.대분류("매출").거래처명("호텔스닷컴").df
```

Out[ ]:

| | 일자 | 전표번호 | 대분류 | 중분류 | 소분류 | 계정코드 | 계정과목 | 적요 | 차변 | 대변 | 구분 | 거래처명 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **3** | 2024-01-03 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 33000000 | 대변 | 호텔스닷컴 |
| **11** | 2024-01-05 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 30000000 | 대변 | 호텔스닷컴 |
| **15** | 2024-01-07 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 31000000 | 대변 | 호텔스닷컴 |
| **23** | 2024-01-09 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 31000000 | 대변 | 호텔스닷컴 |
| **33** | 2024-01-11 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 25000000 | 대변 | 호텔스닷컴 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **515** | 2024-05-23 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 27000000 | 대변 | 호텔스닷컴 |
| **525** | 2024-05-25 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 25000000 | 대변 | 호텔스닷컴 |
| **533** | 2024-05-27 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 39000000 | 대변 | 호텔스닷컴 |
| **537** | 2024-05-29 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 25000000 | 대변 | 호텔스닷컴 |
| **547** | 2024-05-30 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 29000000 | 대변 | 호텔스닷컴 |

79 rows × 12 columns

# 3. 엑셀파일 로딩 기능 추가

## 3-1. 엑셀파일 로딩 기능을 함수로 변경

In [ ]:

```
import pandas as pd

pd.set_option('display.max_rows', 20)
pd.set_option('display.max_columns', 100)
pd.set_option('display.max_colwidth', 50)
pd.set_option('display.width', 300)
pd.set_option('display.expand_frame_repr', True)
```

```
from pandas import DataFrame, Series
from datetime import datetime, date
from openpyxl import load_workbook


def load_xlfile(filename):
    colname_row_number = 4
    start_row_number = 5
    last_row_value = "합 계"


    wb = load_workbook(filename)
    ws = wb.active

    colnames = [x.value for x in ws[colname_row_number]]

    datalist = []
    for row in ws.iter_rows(
            min_row=start_row_number,
            max_col=len(colnames),
            values_only=True
        ):
        if row[0] == last_row_value:
            break
        row = list(row)

        if row[0] is not None:
            일자 = row[0].split('/')
            일자 = date(2024, int(일자[0]), int(일자[1]))
            전표번호 = row[1]

        row[0] = 일자
        row[1] = 전표번호
        if pd.isna(row[5]):
            row[5] = 0
        if pd.isna(row[6]):
            row[6] = 0

        datalist.append(row)
    journal = DataFrame(datalist, columns=colnames)

    category_table = pd.DataFrame([
            ['자본금', '자본', '자본금', '자본금'],
            ['보통예금', '유동자산', '현금및현금성자산', '보통예금'],
            ['객실수입', '매출', '매출', '객실수입'],
            ['고객용품비', '판매비와관리비', '고객용품비', '고객용품비'],
            ['기타영업비용', '판매비와관리비', '판매비용', '기타영업비용'],
            ['판매수수료', '판매비와관리비', '판매비용', '판매수수료'],
            ['세금과공과', '판매비와관리비', '기타판관비', '세금과공과'],
            ['수도광열비', '판매비와관리비', '기타판관비', '수도광열비'],
            ['급여', '판매비와관리비', '인건비', '급여'],
            ['노무용역비', '판매비와관리비', '인건비', '노무용역비'],
            ['이자수입', '영업외수익', '영업외수익', '이자수입']
        ],
        columns=['구분', '대분류', '중분류', '소분류']
    )
```

```
    category_table.set_index('구분', inplace=True)

    journal['대분류'] = None
    journal['중분류'] = None
    journal['소분류'] = None

    for idx in journal.index:
        # journal 데이터프레임의 "계정과목" 컬럼 값 추출
        key = journal.loc[idx, '계정과목']

        # category_table에서 추출된 키값에 해당하는 "대분류", "중분류", "소분류"
        # 값을 journal에 대체
        journal.loc[idx, '대분류'] = category_table.loc[key, '대분류']
        journal.loc[idx, '중분류'] = category_table.loc[key, '중분류']
        journal.loc[idx, '소분류'] = category_table.loc[key, '소분류']

    journal = journal[
        [
            '일자', '전표번호', '대분류', '중분류', '소분류', '계정코드',
            '계정과목', '적요', '차변', '대변', '구분', '거래처명'
            ]
        ]

    return journal
```

In [ ]:
```
# 함수 테스트
journal = load_xlfile("분개장_샘플.xlsx")
print(journal)
```

```
           일자    전표번호      대분류        중분류      소분류    계정코드     계정과
목         적요          차변          대변   구분     거래처명
0    2024-01-02  00001       자본        자본금      자본금  00001     자본금
자본금 입금           0  2000000000  차변     None
1    2024-01-02  00001     유동자산   현금및현금성자산     보통예금  10300     보통예금
자본금 입금  2000000000           0  대변     하나은행
2    2024-01-03  00001     유동자산   현금및현금성자산     보통예금  10300     보통예금
객실 매출    33000000           0  차변     하나은행
3    2024-01-03  00001       매출         매출     객실수입  40100     객실수입
객실 매출           0    33000000  대변    호텔스닷컴
4    2024-01-03  00002  판매비와관리비      고객용품비    고객용품비  80100    고객용품
비  고객용품비 지출    20000000           0  차변   ABC용품사
..          ...    ...      ...        ...      ...    ...      ...
...         ...         ...  ..      ...
543  2024-05-29  00004     유동자산   현금및현금성자산     보통예금  10300     보통예금
고객용품비 지출           0    16100000  대변     하나은행
544  2024-05-29  00005  판매비와관리비       판매비용   기타영업비용  80200   기타영업비
용      영업비용E    15000000           0  차변       C사
545  2024-05-29  00005     유동자산   현금및현금성자산     보통예금  10300     보통예금
영업비용E           0    15000000  대변     하나은행
546  2024-05-30  00001     유동자산   현금및현금성자산     보통예금  10300     보통예금
객실 매출    29000000           0  차변     하나은행
547  2024-05-30  00001       매출         매출     객실수입  40100     객실수입
객실 매출           0    29000000  대변    호텔스닷컴

[548 rows x 12 columns]
```

## 3-2. Journal 모듈 init 함수 수정

- 입력되는 값이 데이터프레임인 경우 바로 잔여 코드 실행.
- 입력되는 값이 엑셀파일의 이름인 경우, 엑셀파일로부터 데이터 로딩 후 잔여 코드 실행

In [ ]:
```
class Journal:
    def __init__(self, input_value):
        if type(input_value) is DataFrame:
            # input_value의 형식이 DataFrame인 경우 바로 df 변수에 저장
            self.df = input_value
        elif type(input_value) is str and input_value[-4:] == "xlsx":
            # input_value의 형식이 string이고, 마지막 4개의 값이 "xlsx"인 경우
            # load_xlfile 함수를 이용하여 엑셀 파일의 데이터를 가져와 df 변수에 저장
            self.df = load_xlfile(input_value)
        else:
            raise ValueError("Error: 데이터 프레임 또는 엑셀파일명을 입력하시요.")

        self.전표번호 = self.Filter(self, "전표번호")
        self.대분류 = self.Filter(self, "대분류")
        self.중분류 = self.Filter(self, "중분류")
        self.소분류 = self.Filter(self, "소분류")
        self.계정코드 = self.Filter(self, "계정코드")
        self.계정과목 = self.Filter(self, "계정과목")
        self.거래처명 = self.Filter(self, "거래처명")

    class Filter:
        def __init__(self, parent_instance, attr_name):
            self.parent_instance = parent_instance
            self.attr_name = attr_name

        def __call__(self, lookup_value):
            df = self.parent_instance.df
            filtered_df = df[df[self.attr_name] == lookup_value]
            return Journal(filtered_df)
```

In [ ]:
```
# 모듈 테스트
jnl = Journal("분개장_샘플.xlsx")
jnl.대분류("매출").거래처명("호텔스닷컴").df
```

Out[ ]:

| | 일자 | 전표번호 | 대분류 | 중분류 | 소분류 | 계정코드 | 계정과목 | 적요 | 차변 | 대변 | 구분 | 거래처명 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **3** | 2024-01-03 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 33000000 | 대변 | 호텔스닷컴 |
| **11** | 2024-01-05 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 30000000 | 대변 | 호텔스닷컴 |
| **15** | 2024-01-07 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 31000000 | 대변 | 호텔스닷컴 |
| **23** | 2024-01-09 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 31000000 | 대변 | 호텔스닷컴 |
| **33** | 2024-01-11 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 25000000 | 대변 | 호텔스닷컴 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **515** | 2024-05-23 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 27000000 | 대변 | 호텔스닷컴 |
| **525** | 2024-05-25 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 25000000 | 대변 | 호텔스닷컴 |
| **533** | 2024-05-27 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 39000000 | 대변 | 호텔스닷컴 |
| **537** | 2024-05-29 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 25000000 | 대변 | 호텔스닷컴 |
| **547** | 2024-05-30 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 29000000 | 대변 | 호텔스닷컴 |

79 rows × 12 columns

# 4. journal_file.py 파일 작성

- journal_file.py 파일 신규 생성
- 엑셀파일 로딩 함수 코드 복사
- Journal 모듈 코드 복사

In [ ]:
```
# journal.py 파일 테스트
from journal_file import Journal
```

In [ ]:
```
jnl = Journal("분개장_샘플.xlsx")
jnl.대분류("매출").거래처명("호텔스닷컴").df
```

Out[ ]:

| | 일자 | 전표번호 | 대분류 | 중분류 | 소분류 | 계정코드 | 계정과목 | 적요 | 차변 | 대변 | 구분 | 거래처명 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **3** | 2024-01-03 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 33000000 | 대변 | 호텔스닷컴 |
| **11** | 2024-01-05 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 30000000 | 대변 | 호텔스닷컴 |
| **15** | 2024-01-07 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 31000000 | 대변 | 호텔스닷컴 |
| **23** | 2024-01-09 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 31000000 | 대변 | 호텔스닷컴 |
| **33** | 2024-01-11 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 25000000 | 대변 | 호텔스닷컴 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **515** | 2024-05-23 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 27000000 | 대변 | 호텔스닷컴 |
| **525** | 2024-05-25 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 25000000 | 대변 | 호텔스닷컴 |
| **533** | 2024-05-27 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 39000000 | 대변 | 호텔스닷컴 |
| **537** | 2024-05-29 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 25000000 | 대변 | 호텔스닷컴 |
| **547** | 2024-05-30 | 00001 | 매출 | 매출 | 객실수입 | 40100 | 객실수입 | 객실매출 | 0 | 29000000 | 대변 | 호텔스닷컴 |

79 rows × 12 columns

In [ ]:
```
```