

[1-3. raw 데이터 INSERT]

1. 데이터베이스 연결 모듈 작성

1-1. ChatGPT 질의

1-1-1. 프롬프트 작성

mysql-connector 라이브러리를 이용해서 mysql DB와 연결하는 Connector 클래스를 만들고 싶어. Context Manager를 이용해서 작성된 Connector 클래스를 모듈처럼 Import해서 DB를 연결하는데 사용할 수 있도록 파이썬 코드 작성해줘.

DB 연결 속성 값은 아래와 같아.

```
host = "localhost",  
user = "root",  
port = "3306",  
database = "daily_sales"
```

1-1-2. 작성 내용 확인



아래는 `mysql-connector-python` 라이브러리를 사용하여 MySQL 데이터베이스와 연결하는 `Connector` 클래스를 정의한 Python 코드입니다. 이 클래스는 Context Manager를 사용하여 데이터베이스 연결을 관리할 수 있도록 구현되었습니다.

python

코드 복사

```
import mysql.connector  
from mysql.connector import Error  
  
class Connector:  
    def __init__(self, host="localhost", user="root", port="3306", database="daily_sales"):  
        self.host = host  
        self.user = user  
        self.port = port  
        self.database = database  
        self.connection = None  
  
    def __enter__(self):  
        try:  
            self.connection = mysql.connector.connect(  
                host=self.host,  
                user=self.user,  
                port=self.port,  
                database=self.database  
            )  
            if self.connection.is_connected():  
                print("Successfully connected to the database")
```

```

        print("Successfully connected to the database.")
    return self.connection
except Error as e:
    print(f"Error: {e}")
    raise

```

```

def __exit__(self, exc_type, exc_val, exc_tb):
    if self.connection.is_connected():
        self.connection.close()
    print("Database connection closed")

# 모듈처럼 사용할 수 있도록 별도의 파일에 저장할 수 있습니다.
# 예를 들어, 이 코드를 `db_connector.py` 파일에 저장하고 import 할 수 있습니다.

# 사용 예제
if __name__ == "__main__":
    with Connector() as connection:
        if connection.is_connected():
            cursor = connection.cursor()
            cursor.execute("SELECT DATABASE();")
            record = cursor.fetchone()
            print(f"You're connected to the database: {record}")

```

이 코드는 다음과 같은 기능을 제공합니다:

1. `__init__` 메서드: 데이터베이스 연결 속성을 초기화합니다.
2. `__enter__` 메서드: Context Manager 진입 시 호출되며 데이터베이스에 연결합니다.
3. `__exit__` 메서드: Context Manager 종료 시 호출되며 데이터베이스 연결을 닫습니다.

사용 방법

1. 위 코드를 예를 들어 `db_connector.py`라는 파일에 저장합니다.
2. 다른 Python 스크립트에서 이 모듈을 import하여 사용할 수 있습니다.

```

python
from db_connector import Connector

with Connector() as connection:
    if connection.is_connected():
        cursor = connection.cursor()
        cursor.execute("SELECT DATABASE();")
        record = cursor.fetchone()
        print(f"You're connected to the database: {record}")

```

이렇게 하면 `Connector` 클래스를 Context Manager로 사용하여 데이터베이스 연결을 간편하게 관리할 수 있습니다.

1-2. 상황에 맞게 파이썬 파일 작성

```
connector.py
# connecor.py

import mysql.connector

class Connector:
    def __enter__(self):
        self.connection = mysql.connector.connect(
            host="localhost",
            user="root",
            port="3306",
            password="fastcampus1!",
            database="daily_sales"
        )
        self.cursor =
self.connection.cursor(dictionary=True)
        return self

    def __exit__(self, exc_type, exc_val, exc_tb):
        self.cursor.close()
        self.connection.close()
```

2. raw data 데이터베이스에 입력

2-1. 엑셀파일에서 raw data 읽어오기

daily_sales_raw_data.xlsx 라는 이름의 엑셀파일이 있고, 엑셀 파일에는 "daily_sales_raw" sheet와 "original_price" sheet가 있어.
pandas 라이브러리를 이용해서 엑셀 각각의 sheet 데이터를 읽어온 후 sheet 이름의 변수에 DataFrame 형태로 저장하고 싶어.



아래는 `pandas` 라이브러리를 사용하여 `daily_sales_raw_data.xlsx` 파일의 각 시트를 읽어와 `DataFrame`으로 저장하는 Python 코드입니다.

python

코드 복사

```
import pandas as pd

# 엑셀 파일 경로
file_path = 'daily_sales_raw_data.xlsx'

# 각 시트를 읽어서 DataFrame으로 저장
daily_sales_raw_df = pd.read_excel(file_path, sheet_name='daily_sales_raw')
original_price_df = pd.read_excel(file_path, sheet_name='original_price')

# DataFrame 출력 (확인용)
print(daily_sales_raw_df.head())
print(original_price_df.head())
```

위 코드는 다음과 같은 기능을 제공합니다:

1. `pandas` 라이브러리를 사용하여 `daily_sales_raw_data.xlsx` 파일을 읽어옵니다.
2. `pd.read_excel` 함수를 사용하여 각 시트를 읽어와 `DataFrame`으로 저장합니다.
3. 각 시트의 `DataFrame`을 `daily_sales_raw_df`와 `original_price_df` 변수에 저장합니다.

```
In [ ]: import pandas as pd

file_path = 'daily_sales_raw_data.xlsx'

daily_sales_raw_df = pd.read_excel(
    file_path,
    sheet_name='daily_sales_raw'
)
original_price_df = pd.read_excel(
    file_path,
    sheet_name='original_price'
)
```

```
In [ ]: daily_sales_raw_df
```

```
Out[ ]:
```

	no	payment_date	stay_days	checkin_date	checkout_date	original_price	payi
0	0	2024-01-03	1	2024-01-13	2024-01-14	700000	
1	1	2024-01-03	2	2024-01-10	2024-01-12	1000000	
2	2	2024-01-03	1	2024-01-03	2024-01-04	500000	
3	3	2024-01-03	5	2024-01-07	2024-01-12	2700000	
4	4	2024-01-03	4	2024-01-09	2024-01-13	2200000	
...
2227	2227	2024-05-30	1	2024-05-31	2024-06-01	700000	
2228	2228	2024-05-30	6	2024-06-01	2024-06-07	3400000	
2229	2229	2024-05-30	2	2024-06-01	2024-06-03	1400000	
2230	2230	2024-05-30	6	2024-06-03	2024-06-09	3400000	
2231	2231	2024-05-30	1	2024-06-08	2024-06-09	700000	

2232 rows × 7 columns

```
In [ ]: original_price_df
```

```
Out[ ]:
```

	date	weekday	original_price
0	2024-01-01	1	500000
1	2024-01-02	2	500000
2	2024-01-03	3	500000
3	2024-01-04	4	500000
4	2024-01-05	5	700000
...
177	2024-06-26	3	500000
178	2024-06-27	4	500000
179	2024-06-28	5	700000
180	2024-06-29	6	700000
181	2024-06-30	7	700000

182 rows × 3 columns

2-2. 읽어온 데이터, 데이터베이스에 INSERT

2-2-1. 프롬프트 작성

읽어온 DataFrame 데이터를 먼저 만들어 놓은 Connector 모듈을 이용해서 데이터베이스에 INSERT 하고 싶어.

Connector 모듈은 아래와 같이 작성해 놓았어.

connecor.py

```
import mysql.connector

class Connector:
    def __enter__(self):
        self.connection = mysql.connector.connect(
            host="localhost",
            user="root",
            port="3306",
            password="fastcampus1!",
            database="daily_sales"
        )
        self.cursor = self.connection.cursor(dictionary=True)
        return self

    def __exit__(self, exc_type, exc_val, exc_tb):
        self.cursor.close()
        self.connection.close()
```

2-2-2. ChatGPT 답변 1



Connector 클래스를 이용하여 pandas DataFrame 데이터를 MySQL 데이터베이스에 삽입하는 방법을 아래에 설명합니다. daily_sales_raw와 original_price 테이블에 각각 데이터를 삽입하는 코드를 작성하겠습니다.

먼저, Connector 클래스를 수정할 필요가 있습니다. 현재 커서가 닫히지 않고 connection 객체만 반환되고 있습니다. 이를 수정하고, 데이터 삽입을 위한 함수를 작성합니다.

python

코드 복사

```
# connector.py
import mysql.connector

class Connector:
    def __enter__(self):
        self.connection = mysql.connector.connect(
            host="localhost",
            user="root",
            port="3306",
            password="fastcampus1!",
            database="daily_sales"
        )
        self.cursor = self.connection.cursor(dictionary=True)
        return self

    def __exit__(self, exc_type, exc_val, exc_tb):
        self.cursor.close()
        self.connection.close()

    def insert_data(self, query, data):
        self.cursor.executemany(query, data)
        self.connection.commit()
```



2-2-3. ChatGPT 답변 2

이제, `pandas` DataFrame 데이터를 읽어와서 데이터베이스에 삽입하는 코드를 작성합니다.

python

코드 복사

```
import pandas as pd
from connector import Connector

# 엑셀 파일 경로
file_path = 'daily_sales_raw_data.xlsx'

# 각 시트를 읽어서 DataFrame으로 저장
daily_sales_raw_df = pd.read_excel(file_path, sheet_name='daily_sales_raw')
original_price_df = pd.read_excel(file_path, sheet_name='original_price')

# daily_sales_raw 테이블에 데이터 삽입
daily_sales_raw_query = """
INSERT INTO daily_sales_raw (
    payment_date, stay_days, checkin_date, checkout_date, original_price, payment_a
) VALUES (%s, %s, %s, %s, %s, %s)
"""

daily_sales_raw_data = daily_sales_raw_df[['payment_date', 'stay_days', 'checkin_date', 'checkout_date', 'original_price', 'payment_amount']]

# original_price 테이블에 데이터 삽입
original_price_query = """
INSERT INTO original_price (
    date, weekday, original_price
) VALUES (%s, %s, %s)
"""

original_price_data = original_price_df[['date', 'weekday', 'original_price']].values.tolist()

# 데이터베이스에 삽입
with Connector() as db:
    db.insert_data(daily_sales_raw_query, daily_sales_raw_data)
    db.insert_data(original_price_query, original_price_data)

print("Data has been inserted successfully")
```

이 코드는 다음과 같은 기능을 제공합니다:

1. `pandas`를 사용하여 엑셀 파일을 읽어와 `DataFrame`으로 저장합니다.
2. `Connector` 클래스를 사용하여 MySQL 데이터베이스에 연결합니다.
3. 각 `DataFrame` 데이터를 데이터베이스에 삽입하는 쿼리를 작성하고, 데이터를 리스트 형태로 변환합니다.
4. `insert_data` 메서드를 사용하여 데이터를 데이터베이스에 삽입합니다.

2-2-4. 코드 실행


```
In [ ]: import pandas as pd
from connector2 import Connector

# 엑셀 파일 경로
file_path = 'daily_sales_raw_data.xlsx'

# 각 시트를 읽어서 DataFrame으로 저장
daily_sales_raw_df = pd.read_excel(
    file_path,
    sheet_name='daily_sales_raw'
)
original_price_df = pd.read_excel(
    file_path,
    sheet_name='original_price'
)

# daily_sales_raw 테이블에 데이터 삽입
daily_sales_raw_query = """
INSERT INTO daily_sales_raw (
    payment_date, stay_days, checkin_date, checkout_date,
    original_price, payment_amount
) VALUES (%s, %s, %s, %s, %s, %s)
"""

daily_sales_raw_data = daily_sales_raw_df[
    ['payment_date', 'stay_days', 'checkin_date',
     'checkout_date', 'original_price', 'payment_amount']
].values.tolist()

# original_price 테이블에 데이터 삽입
original_price_query = """
INSERT INTO original_price (
    date, weekday, original_price
) VALUES (%s, %s, %s)
"""

original_price_data = original_price_df[
    ['date', 'weekday', 'original_price']
].values.tolist()

# 데이터베이스에 삽입
with Connector() as db:
    db.insert_data(daily_sales_raw_query, daily_sales_raw_data)
    db.insert_data(original_price_query, original_price_data)

print("Data has been inserted successfully")

Data has been inserted successfully
```

2-2-5. 워크벤치에서 INSERT 된 데이터 확인

[daily_sales_raw 테이블]

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree with 'daily_sales' expanded, showing 'daily_sales_raw' under 'Tables'. The main query editor contains the query: `1 • SELECT * FROM daily_sales.daily_sales_raw;`. The 'Result Grid' shows 20 rows of data. The bottom 'Action Output' pane shows the execution log for the query.

id	payment_date	stay_days	checkin_date	checkout_date	original_price	payment_amount
1	2024-01-03	1	2024-01-13	2024-01-14	700000	770000
2	2024-01-03	2	2024-01-10	2024-01-12	1000000	980000
3	2024-01-03	1	2024-01-03	2024-01-04	500000	540000
4	2024-01-03	5	2024-01-07	2024-01-12	2700000	2280000
5	2024-01-03	4	2024-01-09	2024-01-13	2200000	2210000
6	2024-01-03	6	2024-01-04	2024-01-10	3600000	3180000
7	2024-01-03	7	2024-01-06	2024-01-13	4100000	3370000
8	2024-01-03	2	2024-01-10	2024-01-12	1000000	930000
9	2024-01-03	3	2024-01-09	2024-01-12	1500000	1550000
10	2024-01-03	4	2024-01-04	2024-01-08	2600000	2220000
11	2024-01-03	4	2024-01-05	2024-01-09	2600000	2510000
12	2024-01-03	6	2024-01-03	2024-01-09	3600000	3430000
13	2024-01-03	4	2024-01-12	2024-01-16	2600000	2190000
14	2024-01-03	1	2024-01-06	2024-01-07	700000	780000
15	2024-01-03	7	2024-01-05	2024-01-12	4100000	3860000
16	2024-01-03	2	2024-01-06	2024-01-08	1400000	1460000
17	2024-01-03	1	2024-01-07	2024-01-08	700000	740000
18	2024-01-04	7	2024-01-09	2024-01-16	4100000	3490000
19	2024-01-04	4	2024-01-09	2024-01-13	2200000	1850000
20	2024-01-04	6	2024-01-09	2024-01-15	3600000	3110000

	Time	Action	Response	Duration / Fetch Time
2	02:08:30	USE daily_sales	0 row(s) affected	0.00059 sec
3	02:08:30	CREATE TABLE IF NOT EXISTS daily_sales_raw (id BIGINT UNSI...	0 row(s) affected	0.016 sec
4	02:08:30	CREATE TABLE IF NOT EXISTS original_price (date DATE NOT N...	0 row(s) affected	0.0038 sec
5	07:55:59	SELECT * FROM daily_sales.daily_sales_raw	2232 row(s) returned	0.0017 sec / 0.0021 s...

[original_price 테이블]

MySQL Workbench

Local instance 3306

AdministrationSchemasQuery 1daily_sales_raworiginal_price

1 • SELECT * FROM daily_sales.original_price;

100%1:1

Result GridFilter Rows: SearchEdit:Export/Import:

date	weekday	original_price
2024-01-02	2	500000
2024-01-03	3	500000
2024-01-04	4	500000
2024-01-05	5	700000
2024-01-06	6	700000
2024-01-07	7	700000
2024-01-08	1	500000
2024-01-09	2	500000
2024-01-10	3	500000
2024-01-11	4	500000
2024-01-12	5	700000
2024-01-13	6	700000
2024-01-14	7	700000
2024-01-15	1	500000
2024-01-16	2	500000
2024-01-17	3	500000
2024-01-18	4	500000
2024-01-19	5	700000
2024-01-20	6	700000
2024-01-21	7	700000

original_price 1ApplyRevert

Action Output

	Time	Action	Response	Duration / Fetch Time
3	02:08:30	CREATE TABLE IF NOT EXISTS daily_sales_raw (id BIGINT UNSI...	0 row(s) affected	0.016 sec
4	02:08:30	CREATE TABLE IF NOT EXISTS original_price (date DATE NOT N...	0 row(s) affected	0.0038 sec
5	07:55:59	SELECT * FROM daily_sales.daily_sales_raw	2232 row(s) returned	0.0017 sec / 0.0021 s...
6	08:00:33	SELECT * FROM daily_sales.original_price	182 row(s) returned	0.0017 sec / 0.00003...

Query Completed

SCHEMAS

Filter objects

cashflow

daily_sales

Tables

daily_sales_raw

original_price

Views

Stored Procedures

Functions

room_sales

sys

Object InfoSession

Schema: daily_sales

Result GridForm EditorField TypesQuery StatsExecution Plan