

01 PJT

관통 프로젝트

INDEX

- 관통 프로젝트 소개
- Requests
- CSV
- 01 PJT

관통 프로젝트 소개

| 관통 프로젝트란?

- 추가 기술 습득
- 정규 커리큘럼에서 학습한 내용과 추가 기술을 모두 포함한 소규모 프로젝트

프로젝트 목표

- 1학기 마지막 월말평가 (최종 관통 프로젝트)를 위한 준비
- 개인 포트폴리오
- 협업 과정 학습

Requests

Requests

Requests is a simple, yet elegant, HTTP library.

Requests 설치와 사용

- pip 명령어로 requests 설치

```
$ pip install requests
```

- 외부 라이브러리 사용을 위해 `import`

```
# 01_requests.py
```

```
import requests
```

Requests 기본 예시 (1/3)

- jsonplaceholder에 데이터 get 요청 보내기

```
import requests

response = requests.get('https://jsonplaceholder.typicode.com/todos/')
print(response)
```

```
$ python 01_requests.py
<Response [200]>
```

➤ 정상적으로 응답 받음을 의미하는 200 status code를 출력

Requests 기본 예시 (2/3)

- json 메서드로 응답 데이터만 출력

```
import requests

response = requests.get('https://jsonplaceholder.typicode.com/todos/').json()
print(response)
```

```
$ python 01_requests.py
[{'userId': 1, 'id': 1, 'title': 'delectus aut autem', 'completed': False}, {'userId': 1, 'id': 2, 'title': 'quis ut nam facilis et officia qui', 'completed': False}, {'userId': 1, 'id': 3, 'title': 'fugiat veniam minus', 'completed': False}, {'userId': 1, 'id': 4, 'title': 'et porro tempora', 'completed': True}, {'userId': 1, 'id': 5, 'title': 'laboriosam mollitia et enim quasi adipisci quia provident illum', 'completed': False}, {'userId': 1, 'id': 6, 'title': 'qui ullam ratione quibusdam voluptatem quia omnis', 'completed': False}, ...]
```

Requests 기본 예시 (3/3)

- 응답 데이터 중, 필요한 결과만 출력 (completed == True)

```
import requests

response = requests.get('https://jsonplaceholder.typicode.com/todos/').json()

for item in response:
    if item['completed']:
        print(item)
```

```
$ python 01_requests.py
{'userId': 1, 'id': 4, 'title': 'et porro tempora', 'completed': True}
{'userId': 1, 'id': 8, 'title': 'quo adipisci enim quam ut ab', 'completed': True}
{'userId': 1, 'id': 11, 'title': 'vero rerum temporibus dolor', 'completed': True}
```

데이터 전처리 (1/2)

- jsonplaceholder가 제공하는 데이터 중, 내 서비스에 필요 없는 데이터가 있다면?
- 필요 데이터 양식에 맞춰 전처리 과정 필요
- completed가 True인 값들 중, id, title 필드만 필요하다면?

```
{  
  'userId': 1,  
  'id': 4,  
  'title': 'et porro tempora',  
  'completed': True  
}
```



```
{  
  'id': 4,  
  'title': 'et porro tempora',  
}
```

데이터 전처리 (2/2)

- 응답 데이터 중, 필요한 값만 추출

```
response = requests.get('https://jsonplaceholder.typicode.com/todos/').json()
completed_items = []
fields = ['id', 'title']
for item in response:
    if item['completed']:
        temp_item = {}
        for key in fields:
            temp_item[key] = item[key]
        completed_items.append(temp_item)

print(completed_items)
```

```
$ python 01_requests.py
[{'id': 4, 'title': 'et porro tempora'}, {'id': 8, 'title': 'quo adipisci enim quam ut ab'},
{'id': 10, 'title': 'illo est ratione doloremque quia maiores aut'}, ...]
```

데이터 전처리 아이디어 (1/3)

- jsonplaceholder는 다양한 정보를 제공

Resources

JSONPlaceholder comes with a set of 6 common resources:

<u>/posts</u>	100 posts
<u>/comments</u>	500 comments
<u>/albums</u>	100 albums
<u>/photos</u>	5000 photos
<u>/todos</u>	200 todos
<u>/users</u>	10 users

데이터 전처리 아이디어 (2/3)

- 작성자 정보를 포함한 데이터로 수정하려면?

- todos 제공 데이터

```
{
  'userId': 1,
  'id': 4,
  'title': 'et porro tempora',
  'completed': True
}
```

- users 제공 데이터

```
{
  'id': 1,
  'name': 'Leanne Graham',
  'username': 'Bret',
  'email': 'Sincere@april.biz',
  ...
}
```

➤ todo의 userId와 user의 id 값이 일치하는 경우

➤ user 정보를 저장할 새로운 key에 필요 데이터만 삽입하여 처리

데이터 전처리 아이디어 (3/3)

- 각 todo별 userId가 일치하는 user 정보 중, 필요 데이터만 추출하여 삽입

```
[
  {'id': 4,
   'title': 'et porro tempora',
   'user': {'id': 1,
            'email': 'Sincere@april.biz',
            'name': 'Leanne Graham',
            'username': 'Bret'}
  },
  ...
]
```

➤ 필요한 user 정보를 얻기 위해 몇 번의 요청이 필요할까?

참고

requests 제공사항

- 다양한 parameters를 활용 할 수 있음
- method 또한, get 외 다양한 method 지원

Main Interface

All of Requests' functionality can be accessed by these 7 methods. They all return an instance of the **Response** object.

`requests.request(method, url, **kwargs)` [\[source\]](#)

Constructs and sends a **Request**.

- Parameters:**
- **method** – method for the new **Request** object: GET, OPTIONS, HEAD, POST, PUT, PATCH, or DELETE.
 - **url** – URL for the new **Request** object.
 - **params** – (optional) Dictionary, list of tuples or bytes to send in the query string for the **Request**.
 - **data** – (optional) Dictionary, list of tuples, bytes, or file-like object to send in the body of the **Request**.

pprint (1/2)

- pprint 모듈은 파이썬 데이터 구조를 인터프리터의 입력으로 사용할 수 있는 형태로 “예쁘게 인쇄” 할 수 있는 기능을 제공

```
import requests
from pprint import pprint

response = requests.get('https://jsonplaceholder.typicode.com/todos/').json()

completed_items = []
fields = ['id', 'title']
for item in response:
    ...

pprint(completed_items)
```

pprint (2/2)

- 딕셔너리는 디스플레이를 계산하기 전에 키로 정렬되어 출력

```
[{'id': 4, 'title': 'et porro tempora',  
'user': {'id': 1, 'name': 'Leanne Graham',  
'username': 'Bret', 'email':  
'Sincere@april.biz'}}, {'id': 8, 'title':  
'quo adipisci enim quam ut ab', 'user':  
{ 'id': 1, 'name': 'Leanne Graham',  
'username': 'Bret', 'email':  
'Sincere@april.biz'}}, {'id': 10, 'title':  
'illo est ratione doloremque quia maiores  
aut', ... ]
```

➤ print 출력 결과

```
[  
  {'id': 4,  
   'title': 'et porro tempora',  
   'user': {'email': 'Sincere@april.biz',  
            'id': 1,  
            'name': 'Leanne Graham',  
            'username': 'Bret'}  
  },  
  ...  
]
```

➤ pprint 출력 결과

이어서..

삼성 청년 SW 아카데미

CSV

CSV

comma-separated values

몇 가지 필드를 쉼표(,)로 구분한 텍스트 데이터 및 텍스트 파일

CSV

1. 단순한 형식

- 사람이 읽기 쉽고 작성하기 쉬운 형식

2. 호환성

- 거의 모든 스프레드시트 프로그램(예: Excel) 및 데이터베이스 시스템에서 지원

3. 텍스트 기반

- 텍스트 편집기를 사용하여 쉽게 편집 가능

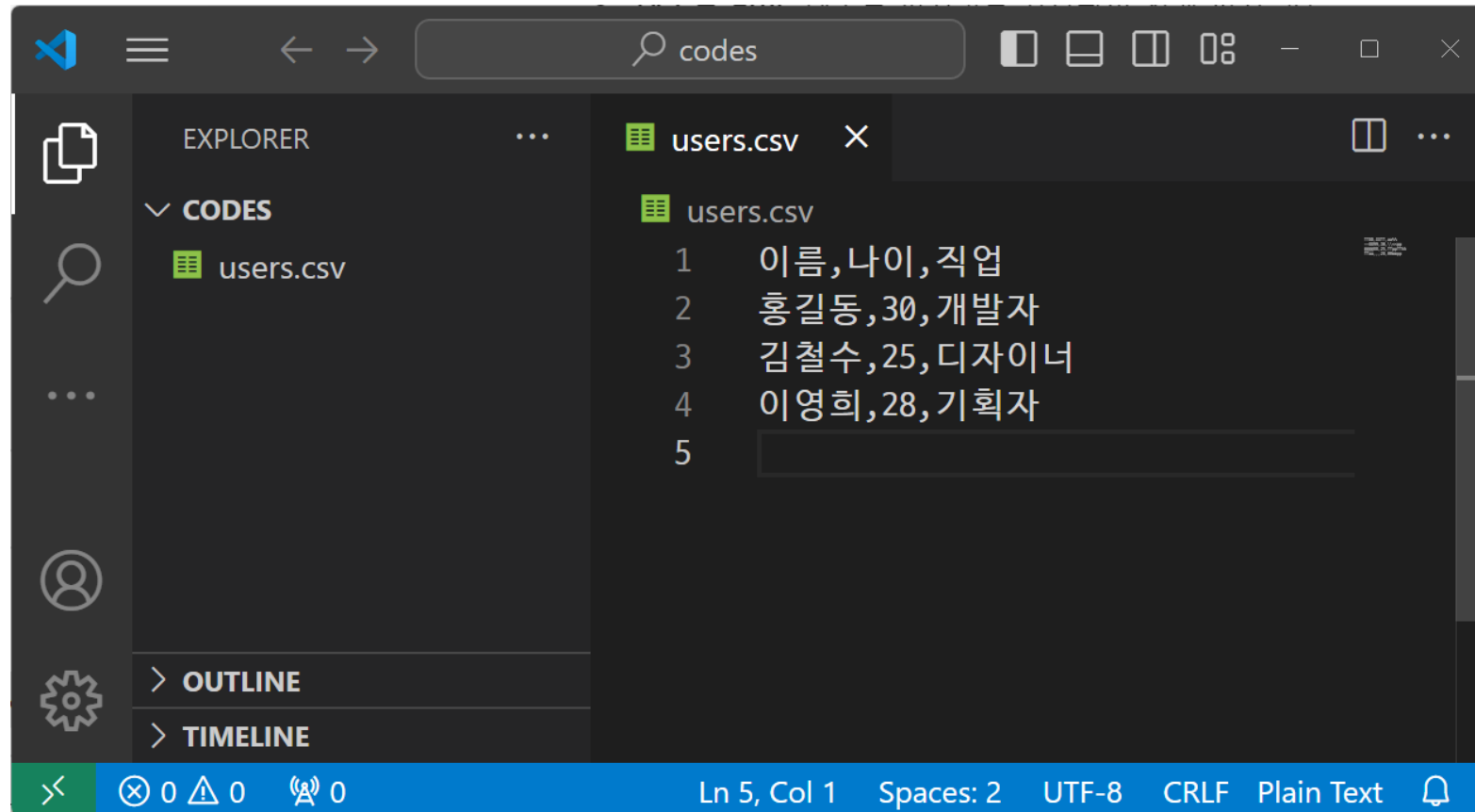
CSV 예시 (1/2)

- 사용자의 이름, 나이, 직업을 포함한 데이터

이름	나이	직업
홍길동	30	개발자
김철수	25	디자이너
이영희	28	기획자

CSV 예시 (2/2)

- .csv 확장자 파일을 VSCode에서 확인



The screenshot shows the Visual Studio Code interface with a file named 'users.csv' open. The Explorer sidebar on the left shows the file structure. The main editor area displays the contents of the file, which is a CSV with 5 lines of data. The status bar at the bottom indicates the file is at Line 5, Column 1, using UTF-8 encoding with CRLF line endings.

Line	Name	Age	Job
1	이름,나이,직업		
2	홍길동	30	개발자
3	김철수	25	디자이너
4	이영희	28	기획자
5			

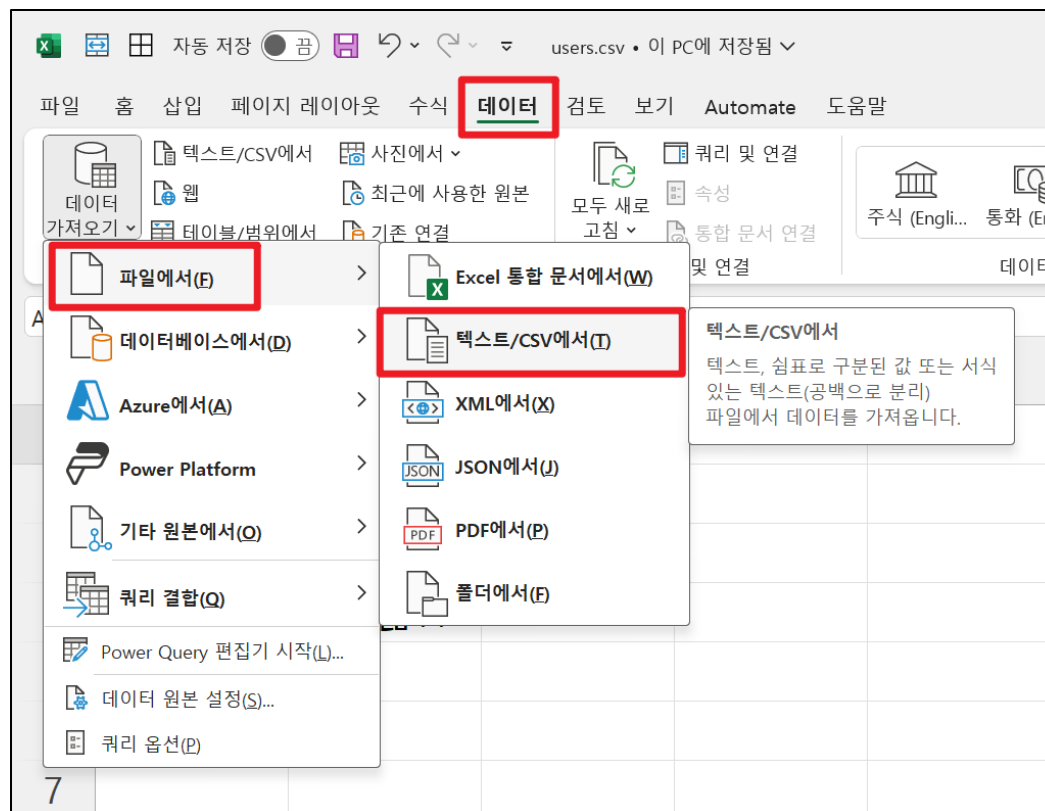
CSV 예시 with excel (1/4)

- .csv 확장자 파일을 Excel에서 확인
- encoding 문제 발생

	A	B	C	D
1	?대짖	?썰썩	吏곡뽕	
2	?땡만??30	媛씽긔??		
3	源泥쭈늬	25	?뵐역?대꺽	
4	?덱뽕??28	媛고썩??		
5				
6				
7				

CSV 예시 with excel (2/4)

- 데이터 > 데이터 가져오기 > 파일에서 > 텍스트/CSV에서



CSV 예시 with excel (3/4)

- 데이터에 따라 적절한 설정
- 원본 파일 작성 시 쉼표(,)를 기준으로 구분하였음

users.csv

파일 원본 구분 기호 데이터 형식 검색

65001: 유니코드(UTF-8) 쉼표 첫 200개 행 기준

이름	나이	직업
홍길동	30	개발자
김철수	25	디자이너
이영희	28	기획자

CSV 예시 with excel (4/4)

- 연결된 데이터 확인

	A	B	C	D	E
1	이름	나이	직업		
2	홍길동	30	개발자		
3	김철수	25	디자이너		
4	이영희	28	기획자		
5					
6					
7					

with statement

| open 함수

- 파일을 열고 파일 객체를 반환하는 함수
- 기본 문법: `open(filename, mode)`
 - `filename`: 열고자 하는 파일의 이름과 경로
 - `mode`: 파일을 여는 모드 (읽기, 쓰기 등)
 - `encoding`: 파일을 읽고 쓸 때 사용할 문자 인코딩
- `with` 문을 사용하면 파일을 자동으로 닫을 수 있음

| with 문이란?

- 리소스를 자동으로 해제하는 문법
- 파일 작업, 네트워크 연결, 쓰레드 락 등에 사용
- 코드의 간결성 및 가독성 향상

with 문 예제 (1/2)

- 기본 사용 방법
- with문 종료시 사용한 리소스가 자동으로 해제 됨

```
with 표현식 as 변수: # with 문을 시작하고, 표현식의 결과를 변수에 할당
    코드 블록        # with 블록 내부에서 실행할 코드
# with문 종료 시 리소스 해제
```

with 문 예제 (2/2)

- 예제: 파일 읽기
- 'example.txt' 파일을 열고 내용을 읽고
- 'with' 블록이 끝나면 **file** 객체가 자동으로 닫힘

```
with open('example.txt', 'r') as file: # 'example.txt' 파일을 읽기 모드로 열기
    content = file.read()              # 파일 내용 읽기
    print(content)                     # 파일 내용 출력
# with 블록이 끝나면 파일이 자동으로 닫힘
```

```
$ python 02_with_statement.py
예제 샘플
```

CSV file read

CSV 파일 읽기

- csv 모듈 import

```
import csv

# CSV 파일 읽기
with open('users.csv', 'r', encoding='utf-8') as file:
    csv_reader = csv.reader(file)
    for row in csv_reader:
        print(row)
```

```
$ python 03_csv_read.py
['이름', '나이', '직업']
['홍길동', '30', '개발자']
['김철수', '25', '디자이너']
['이영희', '28', '기획자']
```

DictReader로 CSV 파일 읽기

- 각 행을 딕셔너리로 변환하여 읽기
- 첫 번째 행을 열 이름으로 사용

```
# CSV 파일 읽기 (DictReader)
with open('users.csv', 'r', encoding='utf-8') as file:
    csv_reader = csv.DictReader(file)
    for row in csv_reader:
        print(row)
```

```
$ python 03_csv_read.py
{'이름': '홍길동', '나이': '30', '직업': '개발자'}
{'이름': '김철수', '나이': '25', '직업': '디자이너'}
{'이름': '이영희', '나이': '28', '직업': '기획자'}
```

CSV file write

newline 매개변수

- open 함수의 매개변수
- 파일에서 새로운 줄을 처리하는 방식을 지정
 - None (기본값): 모든 줄바꿈 문자 (\n, \r\n, \r)를 기본 줄바꿈 문자로 변환
 - '': 줄바꿈 문자를 변환하지 않음 (raw 모드)
 - '\n': 줄바꿈 문자를 LF (Line Feed)로 변환
 - '\r\n': 줄바꿈 문자를 CRLF (Carriage Return + Line Feed)로 변환
 - '\r': 줄바꿈 문자를 CR (Carriage Return)로 변환
- 다양한 운영체제 간의 줄바꿈 문자 차이를 처리

CSV 파일 쓰기 (1/2)

- `newline`을 지정하지 않은 경우

```
with open('data.csv', 'w', encoding='utf-8') as file:  
    csv_writer = csv.writer(file)  
    csv_writer.writerow(['이름', '나이', '직업'])  
    csv_writer.writerow(['홍길동', 30, '개발자'])  
    csv_writer.writerow(['김철수', 25, '디자이너'])  
    csv_writer.writerow(['이영희', 28, '기획자'])
```

이름,나이,직업

홍길동,30,개발자

김철수,25,디자이너

이영희,28,기획자

CSV 파일 쓰기 (1/2)

- newline을 지정

```
with open('data.csv', 'w', newline='', encoding='utf-8') as file:
    csv_writer = csv.writer(file)
    csv_writer.writerow(['이름', '나이', '직업'])
    csv_writer.writerow(['홍길동', 30, '개발자'])
    csv_writer.writerow(['김철수', 25, '디자이너'])
    csv_writer.writerow(['이영희', 28, '기획자'])
```

```
이름,나이,직업
홍길동,30,개발자
김철수,25,디자이너
이영희,28,기획자
```

DictWriter로 CSV 파일 쓰기

- fieldnames 인자로 열 이름 지정 가능

```
with open('data.csv', 'w', newline='', encoding='utf-8') as file:
    fieldnames = ['이름', '나이', '직업']
    csv_writer = csv.DictWriter(file, fieldnames=fieldnames)

    csv_writer.writeheader()
    csv_writer.writerow({'이름': '홍길동', '나이': 30, '직업': '개발자'})
    csv_writer.writerow({'이름': '김철수', '나이': 25, '직업': '디자이너'})
    csv_writer.writerow({'이름': '이영희', '나이': 28, '직업': '기획자'})
```

```
이름,나이,직업
홍길동,30,개발자
김철수,25,디자이너
이영희,28,기획자
```

참고

writeheader 메서드

- `fieldnames`에 지정된 대로 CSV 파일의 첫 번째 행에 헤더가 작성
- CSV 파일의 헤더를 정의
- 데이터가 어떤 필드에 매핑되는지 명확히 하는 데 도움

완료된 todos data만 csv에 추출하기

- 01_requests.py 코드 활용

```
with open('completed_todos.csv', 'w', newline='') as csv_file:
    fieldnames = ['id', 'title']
    writer = csv.DictWriter(csv_file, fieldnames=fieldnames)

    writer.writeheader()
    for item in completed_items:
        writer.writerow({'id': item['id'], 'title': item['title']})
```

```
id,title
4,et porro tempora
8,quo adipisci enim quam ut ab
10,illo est ratione doloremque quia maiores aut
...
```

Import CSV from MySQL

데이터 삽입용 테이블 생성

1. pjt01 DB 생성
2. todos table 생성

The screenshot displays a database management tool interface. On the left sidebar, the tree view shows the following structure:

- localhost 8.0.37
 - pjt01 16k
 - Query
 - create_todos_table
 - Tables
 - todos
 - Columns
 - id int
 - title varchar(150)
 - Index
 - Partitions

The central panel shows the SQL query:

```
SELECT * FROM todos LIMIT 100
```

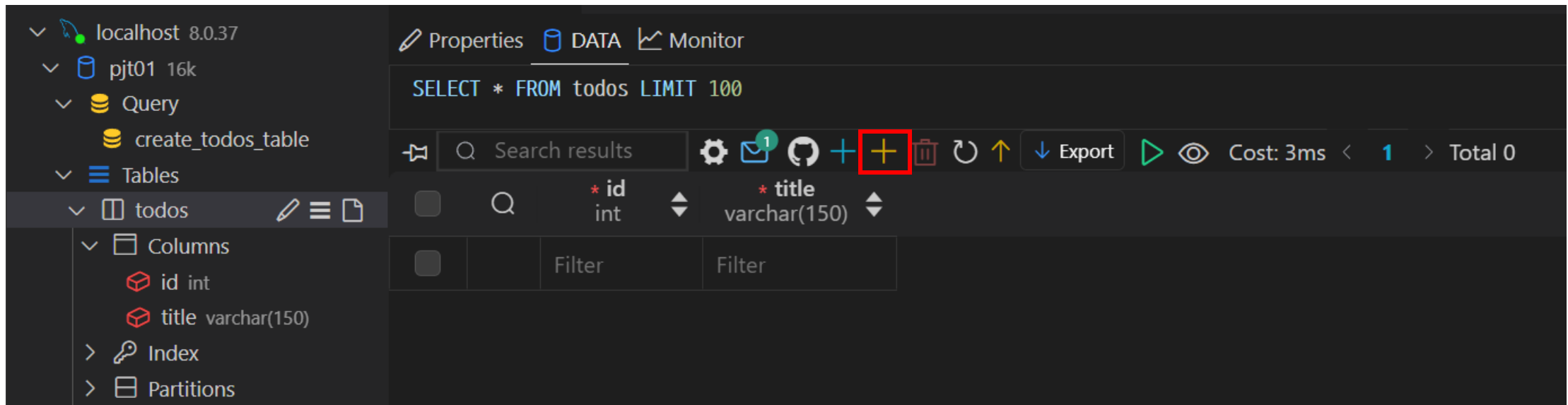
Below the query editor, there is a toolbar with icons for search, settings, refresh, and export. The results pane shows the table schema:

* id	* title
int	varchar(150)

The bottom of the results pane shows filter boxes for each column.

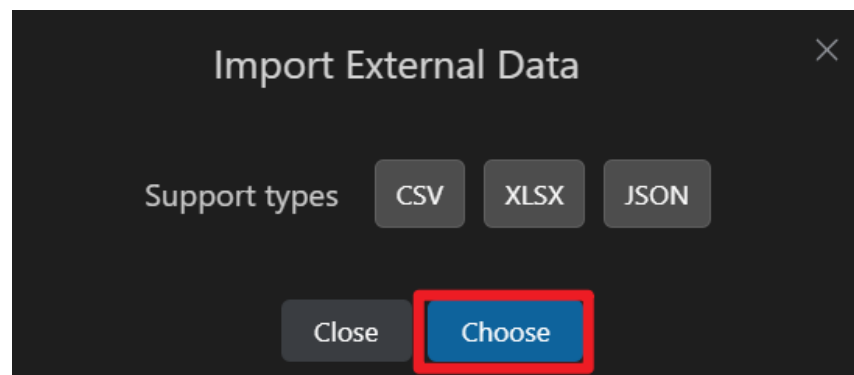
csv 파일 import (1/3)

1. file import 버튼 클릭






csv 파일 import (1/3)

2. 지원 파일 타입 확인 후, Choose 선택

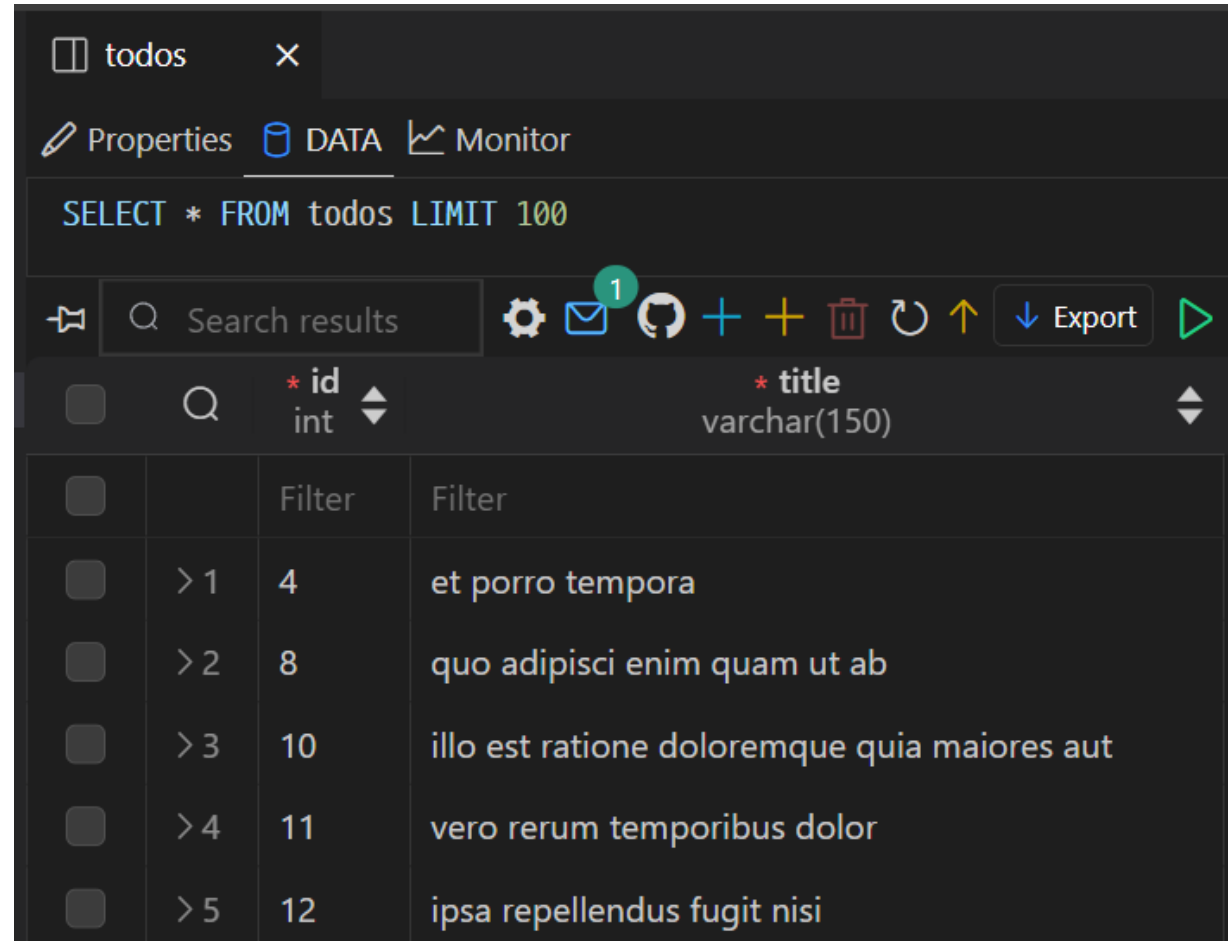


3. 삽입 할 파일 선택 후 삽입

이름	상태	수정한 날짜	유형	크기
 completed_todos.csv		2024-06-16 오전 12:30	Comma Separate...	5KB
 data.csv		2024-06-16 오전 12:19	Comma Separate...	1KB
 users.csv		2024-06-15 오후 10:24	Comma Separate...	1KB

csv 파일 import (1/3)

4. 결과 확인



The screenshot shows a database client window with a tab labeled 'todos'. The 'DATA' tab is selected, displaying the results of the query 'SELECT * FROM todos LIMIT 100'. The interface includes a search bar, a toolbar with icons for settings, email, refresh, and export, and a table of results. The table has columns for 'id' (integer) and 'title' (varchar(150)). The results show five rows of data, each with a checkbox, a greater-than symbol, and the values for 'id' and 'title'.

		* id int	* title varchar(150)
<input type="checkbox"/>		Filter	Filter
<input type="checkbox"/>	> 1	4	et porro tempora
<input type="checkbox"/>	> 2	8	quo adipisci enim quam ut ab
<input type="checkbox"/>	> 3	10	illo est ratione doloremque quia maiores aut
<input type="checkbox"/>	> 4	11	vero rerum temporibus dolor
<input type="checkbox"/>	> 5	12	ipsa repellendus fugit nisi

이어서..

삼성 청년 SW 아카데미

01 PJT

프로젝트 진행 방법

프로젝트 생성 및 관리

- 관통 프로젝트 제출 가이드
- 관통 프로젝트 README.md 작성 가이드

Python을 활용한 데이터 수집

01 PJT 도전 과제

- 목표
 - pjt01 DB 생성
 - 요구 조건에 따른 테이블 생성
 - 파이썬으로 영화 데이터 수집 및 가공
 - 요구 조건에 따른 데이터 전처리
 - 각 테이블에 삽입 가능한 CSV 파일 export
 - 데이터 삽입

다음 시간에
만나요!

삼성 청년 SW 아카데미