

데이터 준비 및 적재

1. 데이터 포맷의 이해

1. 데이터 포맷의 이해

데이터는 그 구조와 형식에 따라 크게 정형, 비정형, 반정형으로 분류할 수 있습니다. 각 데이터 유형은 특정한 특성과 용도를 가지고 있으며, 데이터 분석과 처리 방식도 이에 따라 다르다.

1. 정형 데이터 (Structured Data)

정형 데이터는 고정된 필드와 레코드로 구성된 데이터입니다. 이러한 데이터는 일반적으로 테이블 형식으로 저장되며, 각 필드에는 명확하게 정의된 데이터 유형이 있다.

특징:

- **고정된 구조:** 행과 열로 이루어진 표 형태로 데이터가 정렬된다.
- **명확한 데이터 타입:** 각 열의 데이터 타입이 명확히 정의된다.
- **관계형 데이터베이스에서 사용:** SQL을 통해 데이터베이스에서 관리된다.

예시:

- **스프레드시트:** Excel 파일 (.xlsx)
- **관계형 데이터베이스:** MySQL, PostgreSQL

1. 데이터 포맷의 이해

2. 비정형 데이터 (Unstructured Data)

비정형 데이터는 고정된 구조가 없는 데이터입니다. 텍스트, 이미지, 비디오, 오디오 등 다양한 형식이 포함되며, 분석을 위해 추가적인 처리 및 변환이 필요합니다.

특징:

- **고정된 구조 없음:** 데이터를 저장하는 형식이 일정하지 않음.
- **다양한 형식:** 텍스트 파일, 이미지, 비디오, 오디오 등 다양한 미디어 형식 포함.
- **처리 복잡성:** 분석을 위해서는 자연어 처리(NLP), 이미지 처리 등 고급 기술이 필요.

예시:

- **텍스트 파일:** 로그 파일, 소셜 미디어 포스트
- **멀티미디어 파일:** 이미지 파일 (.jpg, .png), 비디오 파일 (.mp4)

3. 반정형 데이터 (Semi-Structured Data)

반정형 데이터는 정형 데이터와 비정형 데이터의 중간 형태로, 일정한 구조를 가지고 있지만 고정된 스키마는 없는 데이터

특징:

- **유연한 구조:** 데이터의 구조가 존재하지만 고정된 스키마는 없음.
- **계층적 구조:** 중첩된 데이터 구조를 표현할 수 있음.
- **일관성 유지:** 데이터 항목 간의 일관성을 유지하면서도 유연한 확장이 가능.

예시:

- **XML (Extensible Markup Language)**
- **JSON (JavaScript Object Notation)**

2. CSV (Comma-Separated Values) 파일

2. CSV 파일

CSV 파일의 특징

CSV는 **Comma-Separated Values**의 약자로, 각 데이터가 쉼표(,)로 구분되는 텍스트 파일 형식이다. CSV 파일은 보통 아래와 같은 형태를 가진다.

id,name,age,city
10,John,28,New York
11,Jane,32,Los Angeles
12,Doe,22,Chicago



id	name	age	city
10	John	28	New York
11	Jane	32	Los Angeles
12	Doe	22	Chicago

여기서 각 줄은 하나의 레코드를 나타내며, 쉼표로 구분된 각 항목은 필드를 나타냅니다. 첫 번째 줄은 일반적으로 열 제목(header)을 포함하여 각 열의 의미를 명확히 합니다.

장점

- 1. 광범위한 지원:** CSV 포맷은 거의 모든 데이터 분석 도구와 프로그래밍 언어에서 지원됩니다. Python, R, Excel, SQL, 그리고 대부분의 데이터베이스 시스템에서 CSV 파일을 쉽게 읽고 쓸 수 있다.
- 2. 사용 용이성:** CSV 파일은 사람이 쉽게 읽고 이해할 수 있는 형식이다. 텍스트 에디터로 열어볼 수 있으며, 복잡한 소프트웨어 없이도 데이터를 검토할 수 있다.
- 3. 가벼운 파일 크기:** 다른 데이터 포맷에 비해 비교적 단순한 구조로 인해 파일 크기가 작다.

2. CSV 파일

단점

- 1.대용량 데이터 처리의 비효율성:** csv 파일은 구조가 단순하고 인덱스나 메타데이터를 포함하지 않기 때문에 대용량 데이터를 처리할 때 비효율적일 수 있다. 모든 데이터를 메모리에 로드해야 하기 때문에 성능이 저하될 수 있다.
- 2.메타데이터 포함 불가:** csv 포맷은 데이터 외에 추가적인 메타데이터를 포함할 수 없다. 예를 들어, 데이터 유형, 인덱스, 주석 등의 정보를 저장할 수 없다.
- 3.데이터 정렬의 어려움:** csv 파일은 계층적 구조를 지원하지 않기 때문에 복잡한 데이터 관계를 표현하기 어렵다. 예를 들어, 중첩된 데이터 구조를 표현하는 데 한계가 있다.
- 4.데이터 무결성 문제:** 쉼표가 데이터 값의 일부로 포함될 경우 데이터의 무결성이 깨질 수 있다. 이를 방지하기 위해 따옴표로 데이터를 감싸거나 이스케이프 문자를 사용하는 등의 추가적인 처리 작업이 필요하다.

요약

CSV는 각 데이터 항목이 쉼표로 구분되는 텍스트 파일 형식이다. 광범위한 지원과 사용 용이성, 가벼운 파일 크기 등의 장점이 있지만, 대용량 데이터 처리 비효율성, 메타데이터 포함 불가 등의 단점도 있다. CSV는 단순한 데이터 전송과 저장에 유용하다.

3. Excel (XLSX) 파일

3. Excel (XLSX) 파일

특징

Excel 파일 형식(XLSX)은 마이크로소프트 엑셀에서 사용하는 스프레드시트 파일 형식. 데이터가 행과 열로 구성된 표 형태로 저장돼 있어. XLSX 파일은 복잡한 계산, 그래프, 피벗 테이블 등의 기능을 지원하면서 데이터를 관리할 수 있는 강력한 도구

장점

1.데이터 시각적 편집 및 관리: 엑셀은 데이터를 시각적으로 편집하고 관리하는 데 매우 편리하며 셀을 클릭하여 데이터를 입력하거나 수정할 수 있고, 다양한 서식 도구를 사용해 데이터를 강조하거나 체계적으로 정리할 수 있다

1.여러 시트 포함 가능: 하나의 XLSX 파일 안에 여러 개의 시트를 포함할 수 있고 이를 통해 서로 관련된 데이터를 하나의 파일에서 분리하고 체계적으로 관리할 수 있다

2.고급 기능 지원: 엑셀은 수식, 매크로, 그래프, 피벗 테이블 등 다양한 고급 기능을 제공하며 이를 통해 데이터 분석과 보고서를 쉽게 작성할 수 있다.

3.사용자 친화적 인터페이스: 엑셀은 직관적이고 사용자 친화적인 인터페이스를 제공하여 데이터 분석 경험이 적은 사람도 쉽게 사용할 수 있다.

3. Excel (XLSX) 파일

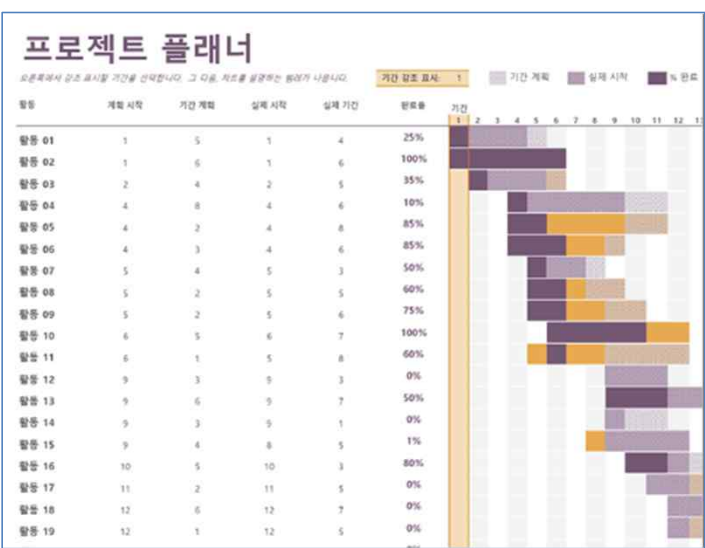
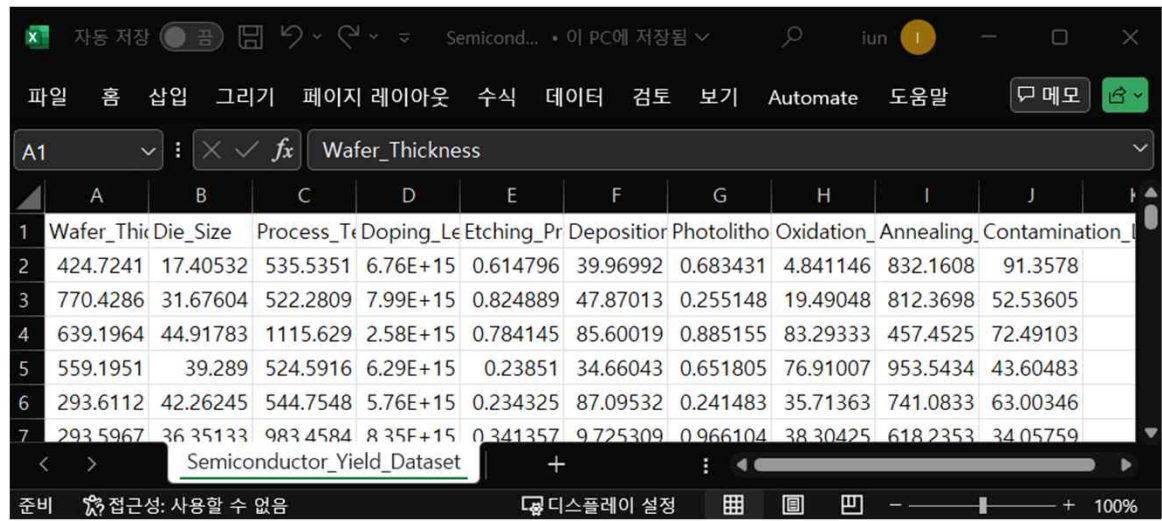
단점

- 1.대용량 데이터 처리 비효율:** 엑셀은 대용량 데이터를 처리할 때 성능이 저하될 수 있고 행과 열의 수가 많아지면 파일 크기가 커지고, 계산 속도가 느려질 수 있다. 대규모 데이터를 다룰 때는 데이터베이스나 빅데이터 도구를 사용하는 것이 더 효율적임.
- 2.호환성 문제:** 엑셀 파일은 다른 시스템이나 소프트웨어와의 호환성이 csv에 비해 낮을 수 있다. 예를 들어, 특정 소프트웨어나 프로그래밍 언어에서 XLSX 파일을 직접 읽거나 쓰기 어려운 경우가 있다. 이럴 때는 csv 같은 단순한 텍스트 형식이 더 유리할 수 있다.
- 3.데이터 무결성 위험:** 엑셀 파일은 수동으로 데이터를 입력하고 수정할 수 있다, 실수로 인해 데이터가 손상되거나 무결성이 깨질 위험이 있고 특히 여러 사람이 동일한 파일을 동시에 편집할 때 이런 문제가 발생할 가능성이 높다.
- 4.보안 취약성:** XLSX 파일은 암호를 설정할 수 있지만, 여전히 보안에 취약할 수 있다. 중요한 데이터는 데이터베이스와 같은 더 안전한 저장소에 보관하는 것이 좋을 수 있다.

3. Excel (XLSX) 파일

요약

엑셀 파일(XLSX)은 데이터를 시각적으로 편집하고 관리하기 좋은 스프레드시트 형식이며 여러 시트를 포함할 수 있고 다양한 고급 기능을 제공하지만, 대용량 데이터 처리에 비효율적이고 호환성 문제가 있을 수 있다. 따라서 엑셀은 소규모 데이터 분석 및 보고서 작성에 적합하며, 대규모 데이터 처리를 위해서는 다른 도구를 고려해야 한다.



4. JSON (JavaScript Object Notation) 포맷

4. JSON (JavaScript Object Notation) 포맷

특징

JSON은 JavaScript Object Notation의 약자로, 속성-값 쌍으로 이루어진 텍스트 형식. 이 형식은 사람이 읽고 쓰기 쉽고, 기계가 해석하고 생성하기도 용이한 데이터 교환 형식이다. JSON 데이터는 중괄호 {}와 대괄호 []를 사용해 계층적 구조를 표현할 수 있다

```
{
  "name": "John Doe",
  "age": 30,
  "is_student": false,
  "courses": ["Math", "Science", "Literature"],
  "address": {
    "street": "123 Main St",
    "city": "Anytown",
    "zipcode": "12345"
  }
}
```

JSON 데이터는 name, age, is_student, courses, address 등의 속성-값 쌍으로 구성돼 있어. 각 속성은 콜론(:)을 사용해 값과 연결되며, 값은 문자열, 숫자, 배열, 객체 등 다양한 데이터 유형이 될 수 있다

4. JSON (JavaScript Object Notation) 포맷

장점

- 1. 구조화된 데이터 표현에 적합:** JSON은 데이터를 계층적이고 구조적으로 표현하는 데 매우 적합하며 중첩된 객체와 배열을 통해 복잡한 데이터 구조를 쉽게 나타낼 수 있다.
- 2. 웹 API와의 호환성:** JSON은 웹 API에서 데이터 교환 형식으로 널리 사용된다. 대부분의 웹 서비스는 JSON 형식을 사용해 데이터를 송수신하며, 이는 JSON이 JavaScript와의 호환성이 뛰어나기 때문이다.
- 3. 언어 독립성:** JSON은 JavaScript 기반으로 설계됐지만, 대부분의 프로그래밍 언어에서 지원되며. Python, Java, C#, PHP 등 다양한 언어에서 JSON 데이터를 쉽게 파싱하고 생성할 수 있다.
- 4. 가벼운 데이터 포맷:** JSON은 텍스트 기반 데이터 포맷으로, 비교적 가벼운 크기를 가지고 있어 네트워크를 통해 데이터를 전송할 때 효율적이다.

4. JSON (JavaScript Object Notation) 포맷

단점

- 1. 대용량 데이터 처리 속도 저하:** JSON은 대용량 데이터의 경우 처리 속도가 느려질 수 있다. JSON 파싱은 메모리 소모가 크기 때문에, 매우 큰 JSON 파일을 처리할 때 성능이 저하될 수 있다. 대규모 데이터를 다룰 때는 바이너리 포맷을 고려할 수 있다.
- 2. 스키마가 없음:** JSON은 정해진 스키마가 없어 데이터의 구조가 유연하지만, 이는 데이터 무결성을 보장하는 데 어려움을 줄 수 있다. 데이터의 일관성을 유지하려면 추가적인 검증 작업이 필요하다.
- 3. 이진 데이터 비효율:** JSON은 텍스트 기반 포맷이기 때문에, 이진 데이터를 저장할 때 비효율적일 수 있다. 이진 데이터를 처리할 때는 Base64 인코딩을 사용해야 하며, 이는 데이터 크기를 증가시키고 처리 속도를 늦출 수 있다.

4. JSON (JavaScript Object Notation) 포맷

요약

JSON은 속성-값 쌍으로 이루어진 텍스트 형식의 데이터 포맷이다. 구조화된 데이터를 표현하는 데 적합하고, 웹 API와의 호환성이 뛰어나며, 다양한 프로그래밍 언어에서 지원된다. 그러나 대용량 데이터를 처리할 때 성능이 저하될 수 있고, 스키마가 없어 데이터 일관성을 유지하기 어렵다. JSON은 가벼운 데이터 교환 포맷으로, 특히 웹 개발에서 널리 사용된다.

5. SQL (Structured Query Language) 포맷

5. SQL (Structured Query Language) 포맷

특징

SQL은 **Structured Query Language**의 약자로, 관계형 데이터베이스에서 데이터를 관리하고 조작하기 위해 사용되는 질의 언어이다. SQL을 통해 데이터베이스에서 데이터를 검색, 삽입, 업데이트, 삭제할 수 있으며, 데이터베이스 스키마를 정의하고 수정할 수도 있다.

데이터베이스 테이블 생성

```
sql

CREATE TABLE students (
  id INT PRIMARY KEY,
  name VARCHAR(50),
  age INT,
  major VARCHAR(50)
);
```

데이터 삽입

```
sql 코드 복사

INSERT INTO students (id, name, age, major) VALUES (1, 'John Doe', 20, 'Computer Science');
INSERT INTO students (id, name, age, major) VALUES (2, 'Jane Smith', 22, 'Mathematics');
INSERT INTO students (id, name, age, major) VALUES (3, 'Michael Brown', 21, 'Physics');
INSERT INTO students (id, name, age, major) VALUES (4, 'Emily Davis', 23, 'Chemistry');
INSERT INTO students (id, name, age, major) VALUES (5, 'Daniel Wilson', 20, 'Biology');
INSERT INTO students (id, name, age, major) VALUES (6, 'Sophia Johnson', 22, 'Economics');
INSERT INTO students (id, name, age, major) VALUES (7, 'David Lee', 21, 'History');
INSERT INTO students (id, name, age, major) VALUES (8, 'Emma Garcia', 23, 'Literature');
INSERT INTO students (id, name, age, major) VALUES (9, 'Olivia Martinez', 20, 'Art');
INSERT INTO students (id, name, age, major) VALUES (10, 'James Anderson', 22, 'Philosophy');
```

5. SQL (Structured Query Language) 포맷

데이터 조회

sql

```
SELECT * FROM students;
```

조회 결과

id	name	age	major
1	John Doe	20	Computer Science
2	Jane Smith	22	Mathematics
3	Michael Brown	21	Physics
4	Emily Davis	23	Chemistry
5	Daniel Wilson	20	Biology
6	Sophia Johnson	22	Economics
7	David Lee	21	History
8	Emma Garcia	23	Literature
9	Olivia Martinez	20	Art
10	James Anderson	22	Philosophy

5. SQL (Structured Query Language) 포맷

장점

- 1. 대규모 데이터 관리에 최적화:** SQL은 대규모 데이터를 효율적으로 관리하는 데 최적화되어 있다. 관계형 데이터베이스는 인덱싱, 조인, 트랜잭션 관리 등의 기능을 제공하여 대용량 데이터를 신속하고 정확하게 처리할 수 있다.
- 2. 강력한 쿼리 기능:** SQL은 매우 강력한 쿼리 기능을 제공한다. SELECT, INSERT, UPDATE, DELETE 등의 기본 명령어를 통해 데이터를 조작할 수 있으며, JOIN, SUBQUERY, AGGREGATE FUNCTIONS 등을 사용하여 복잡한 쿼리도 쉽게 작성할 수 있다.
- 3. 데이터 무결성과 보안:** SQL은 데이터 무결성과 보안을 보장하는 다양한 메커니즘을 제공한다. PRIMARY KEY, FOREIGN KEY, UNIQUE 등의 제약 조건을 통해 데이터의 일관성을 유지할 수 있으며, 권한 관리와 트랜잭션 제어를 통해 데이터베이스의 보안을 강화할 수 있다.
- 4. 표준화된 언어:** SQL은 국제 표준화 기구(ISO)와 미국 표준화 기구(ANSI)에 의해 표준화되어 있다. 이는 SQL이 다양한 데이터베이스 시스템에서 일관되게 사용될 수 있음을 의미하며, 사용자가 특정 데이터베이스 시스템에 종속되지 않고도 SQL을 배워서 활용할 수 있게 한다.

5. SQL (Structured Query Language) 포맷

단점

- 1.비구조화된 데이터 처리에 부적합:** SQL은 관계형 데이터베이스의 테이블 구조에 최적화되어 있기 때문에, 비구조화된 데이터나 반정형 데이터를 처리하는 데 부적합하다. 텍스트 파일, JSON, XML과 같은 데이터 형식은 SQL로 직접 처리하기 어렵다. 이러한 데이터는 NoSQL 데이터베이스나 다른 데이터 처리 도구를 사용하는 것이 더 적합하다.
- 2.복잡한 스키마 변경의 어려움:** SQL 데이터베이스는 스키마가 고정되어 있어, 스키마를 변경하는 작업이 복잡하고 시간이 많이 소요될 수 있다. 특히 대규모 데이터베이스에서 스키마를 변경하려면 데이터베이스를 중지하고 변경 작업을 수행해야 하는 경우도 있어, 운영에 큰 영향을 줄 수 있다.
- 3.확장성의 한계:** SQL 데이터베이스는 수직적 확장(성능 향상을 위해 더 강력한 서버를 사용하는 것)에 최적화되어 있지만, 수평적 확장(여러 서버로 데이터베이스를 분산하는 것)에는 한계가 있다. 반면, NoSQL 데이터베이스는 수평적 확장에 더 적합하다.
- 4.복잡한 쿼리의 성능 저하:** 매우 복잡한 쿼리나 다중 테이블 조인을 사용할 경우, 성능이 저하될 수 있다. 이러한 문제를 해결하기 위해 인덱스 튜닝, 쿼리 최적화 등의 추가 작업이 필요하다.

5. SQL (Structured Query Language) 포맷

요약

SQL은 관계형 데이터베이스에서 데이터를 관리하고 조작하는 데 사용되는 강력한 질의 언어이다. 대규모 데이터 관리와 쿼리에 최적화되어 있으며, 데이터 무결성과 보안을 보장하는 다양한 기능을 제공한다. 하지만 비구조화 된 데이터 처리에 부적합하고, 스키마 변경과 확장성에 한계가 있을 수 있다. SQL은 표준화된 언어로서 다양한 데이터베이스 시스템에서 일관되게 사용할 수 있다.

6. XML (Extensible Markup Language) 포맷

6. XML (Extensible Markup Language) 포맷

특징

XML은 **Extensible Markup Language**의 약자로, 데이터를 구조화하기 위해 태그를 사용하여 정의된 텍스트 기반 데이터 형식이다. XML은 사람이 읽고 쓰기 쉽도록 설계되었으며, 다양한 시스템 간에 데이터를 교환하는 데 사용된다. XML은 자체적으로 태그를 정의할 수 있어 유연하고 확장 가능하다.

```
<students>
  <student>
    <id>1</id>
    <name>John Doe</name>
    <age>20</age>
    <major>Computer Science</major>
  </student>
  <student>
    <id>2</id>
    <name>Jane Smith</name>
    <age>22</age>
    <major>Mathematics</major>
  </student>
</students>
```

이 예제는 students라는 루트 엘리먼트 아래 여러 student 엘리먼트를 포함하고 있다. 각 student 엘리먼트는 id, name, age, major 태그를 통해 학생의 정보를 표현하고 있다.

6. XML (Extensible Markup Language) 포맷

장점

- 1. 유연한 데이터 구조:** XML은 사용자가 직접 태그를 정의할 수 있어, 다양한 형태의 데이터를 표현하는 데 유연하다. 계층적 구조를 쉽게 표현할 수 있어 복잡한 데이터를 잘 나타낼 수 있다.
- 2. 시스템 간 호환성:** XML은 플랫폼과 언어에 독립적이어서, 다양한 시스템 간에 데이터를 교환하는 데 적합하다. 웹 서비스와 데이터 교환 포맷으로 널리 사용된다.
- 3. 표준화된 형식:** XML은 W3C(World Wide Web Consortium)에 의해 표준화되어 있어, 일관된 데이터 표현 방식을 제공한다. 이를 통해 데이터를 교환하는 시스템 간의 호환성을 보장한다.
- 4. 데이터 검증 가능:** XML은 DTD(Document Type Definition)나 XML Schema를 통해 문서의 구조를 정의하고 검증할 수 있다. 이를 통해 데이터의 일관성과 무결성을 유지할 수 있다.

6. XML (Extensible Markup Language) 포맷

단점

- 1. 대용량 데이터의 비효율성:** XML은 태그를 사용하여 데이터를 표현하기 때문에, 데이터 자체보다 태그가 차지하는 공간이 더 클 수 있다. 이는 대용량 데이터를 다룰 때 비효율적일 수 있다.
- 2. 복잡한 파싱:** XML 데이터를 파싱하는 작업은 비교적 복잡하고 리소스를 많이 소모할 수 있다. 특히 큰 XML 문서를 처리할 때 성능이 저하될 수 있다.
- 3. 가독성의 한계:** 사람이 읽을 수 있는 형식으로 설계되었지만, 복잡한 구조의 XML 문서는 가독성이 떨어질 수 있다. 태그가 많아지면 문서가 길어지고 읽기 어려워질 수 있다.
- 4. 구조적 제약:** XML은 엄격한 구조적 규칙을 따르기 때문에, 유연성이 떨어질 수 있다. 비정형 데이터나 매우 동적인 데이터 구조를 표현하는 데는 적합하지 않을 수 있다.

6. XML (Extensible Markup Language) 포맷

요약

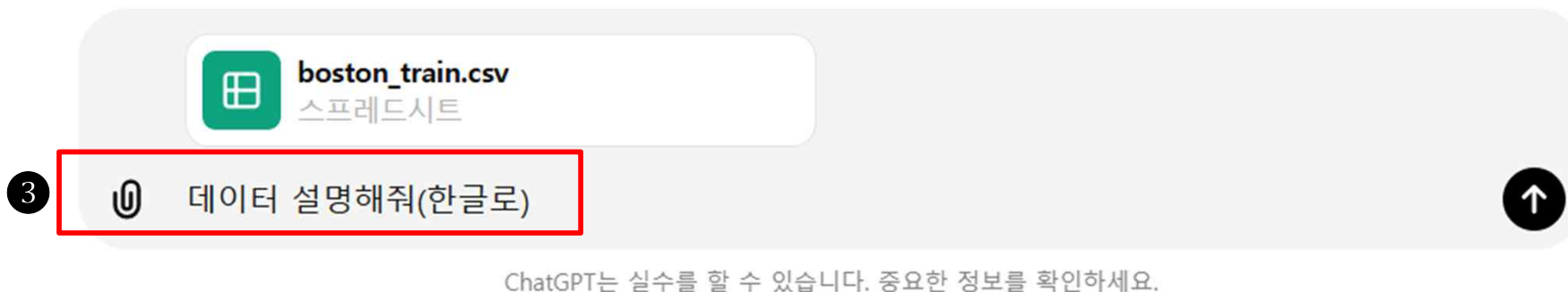
XML은 데이터를 구조화하기 위해 태그를 사용하여 정의된 텍스트 기반 데이터 형식이다. 유연한 데이터 구조와 시스템 간 호환성, 표준화된 형식, 데이터 검증 가능 등의 장점을 가지고 있지만, 대용량 데이터의 비효율성, 복잡한 파싱, 가독성의 한계, 구조적 제약 등의 단점도 있다. XML은 다양한 시스템 간에 데이터를 교환하는 데 유용한 포맷이다.

7. ChatGPT로 데이터 로딩 실습

7. ChatGPT로 데이터 로딩 실습

ChatGPT로 CSV파일 업로드하기

- 데이터셋 로드 및 확인: 데이터셋을 불러오고 데이터셋 정보를 확인.



7. ChatGPT로 데이터 로딩 실습

Boston Train					↓ ↗
	CRIM	ZN	INDUS	NOX	
1	2.3004	0.0	19.58	0.605	
2	13.3598	0.0	18.1	0.693	
3	0.12744	0.0	6.91	0.448	
4	0.15876	0.0	10.81	0.413	
5	0.03768	80.0	1.52	0.404	
6	0.03705	20.0	3.33	0.4429	

7. ChatGPT로 데이터로딩 실습




이 데이터셋은 보스턴 주택 가격 데이터를 포함하고 있으며, 총 400개의 행과 10개의 열로 구성되어 있습니다. 각 열은 다음과 같은 정보를 나타냅니다:


- **CRIM**: 타운별 1인당 범죄율
- **ZN**: 25,000 평방피트를 초과하는 주거 구역의 비율
- **INDUS**: 비소매 상업 지역 비율
- **NOX**: 일산화질소 농도
- **RM**: 주택당 평균 방 수
- **AGE**: 1940년 이전에 건축된 소유주 거주 주택 비율
- **DIS**: 다섯 개의 보스턴 고용 센터까지의 가중 거리
- **TAX**: 10,000달러당 재산세율
- **PTRATIO**: 타운별 학생/교사 비율
- **MEDV**: 소유 주택의 중간 가격 (단위: \$1,000)


이 데이터셋은 주택 가격을 예측하기 위해 사용되며, 각 변수는 주택 가격에 영향을 미치는 요인들입니다. [↗]

7. ChatGPT로 데이터 로딩 실습

ChatGPT로 엑셀(xlsx)파일 업로드하기

 **credit_card.xlsx**
스프레드시트

 데이터 설명해줘(한글로)



ChatGPT는 실수를 할 수 있습니다. 중요한 정보를 확인하세요.

Credit Card Data ▾					↓ ↗
	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	
3	90000	2	2	2	
4	50000	2	2	1	

7. ChatGPT로 데이터로딩 실습



업로드하신 파일은 신용카드 데이터로 보입니다. 각 열의 의미를 설명하겠습니다:

1. **LIMIT_BAL**: 신용 한도 (단위: NT 달러)
2. **SEX**: 성별 (1 = 남성, 2 = 여성)
3. **EDUCATION**: 교육 수준 (1 = 대학원, 2 = 대학, 3 = 고등학교, 4 = 기타/알 수 없음)
4. **MARRIAGE**: 결혼 상태 (1 = 결혼, 2 = 미혼, 3 = 기타)
5. **AGE**: 나이 (년 단위)
6. **PAY_0**: 9월 상환 상태 (-1 = 납부 없음, 0 = 정시 납부, 1 = 1개월 연체, 2 = 2개월 연체, ...)
7. **PAY_2**: 8월 상환 상태 (PAY_0와 동일한 방식)
8. **PAY_3**: 7월 상환 상태 (PAY_0와 동일한 방식)
9. **PAY_4**: 6월 상환 상태 (PAY_0와 동일한 방식)
10. **PAY_5**: 5월 상환 상태 (PAY_0와 동일한 방식)
11. **PAY_6**: 4월 상환 상태 (PAY_0와 동일한 방식)
12. **BILL_AMT1**: 9월 청구 금액 (단위: NT 달러)
13. **BILL_AMT2**: 8월 청구 금액 (단위: NT 달러)
14. **BILL_AMT3**: 7월 청구 금액 (단위: NT 달러)
15. **BILL_AMT4**: 6월 청구 금액 (단위: NT 달러)

7. ChatGPT로 데이터로딩 실습

16. **BILL_AMT5**: 5월 청구 금액 (단위: NT 달러)
17. **BILL_AMT6**: 4월 청구 금액 (단위: NT 달러)
18. **PAY_AMT1**: 9월 납부 금액 (단위: NT 달러)
19. **PAY_AMT2**: 8월 납부 금액 (단위: NT 달러)
20. **PAY_AMT3**: 7월 납부 금액 (단위: NT 달러)
21. **PAY_AMT4**: 6월 납부 금액 (단위: NT 달러)
22. **PAY_AMT5**: 5월 납부 금액 (단위: NT 달러)
23. **PAY_AMT6**: 4월 납부 금액 (단위: NT 달러)
24. **default payment next month**: 다음 달 연체 여부 (1 = 연체, 0 = 연체 아님)

이 데이터는 신용카드 사용자의 다양한 특성과 상환 이력을 포함하고 있으며, 이를 통해 신용 위험을 분석하는 데 사용할 수 있습니다. [↗]



감사합니다