소규모 쇼핑몰을 위한

재고관리 프로그램

"인벤토리메니저"

조은겸

기업 맞춤형 빅데이터 분석가 양성 과정 파이썬 개인 프로젝트

목차

프로젝트 목적

FOR 소규모 쇼핑몰 사업자

이 프로젝트는 소규모 쇼핑몰 사업자들이 **상품의 재고를 효과적으로 관리하기 위한 목적**으로 실행되었습니다. 창고의 재고상품을 신속하게 관리하고, 매입이나 매출 발생의 기록 뿐만 아니라 재고 및 원가의 증감도 자동적으로 계산되어 효율적인 쇼핑몰 운영에 기여할 수 있도록 프로젝트를 진행했습니다.

프로젝트의 필요성

01 재고관리의 어려움

- 복잡성과 오류

수작업으로 재고를 관리하는 것은 많은 복잡성과 오류 발생 가능성을 수반. 인간의 실수로 잘못된 데이터를 입력하게 되면 큰 손실이 발생할 수 있음.

- 시간 낭비

수작업 입력은 시간 소모적임.

재고관리의 효율성

02 데이터 수집의 어려움

- 매입 및 매출 데이터 정확성

매입 및 매출 데이터를 체계적으로 기록하지 않으면 데이터 정확성에 대한 의문이 생길 수 있음
어떤 상품이 언제 어떻게 매입되었는지 정확하게 추적할 필요가 있음.

- 예측 및 분석

데이터의 신뢰성을 확보해야 비즈니스 분석이 가능함.

자료의 무결성 획득

03 업무 효율 증진 필요

- 자동화된 재고 관리

자동화를 통한 업무 효율성 향상
실시간 업데이트

- 데이터 신속성

필요한 정보를 데이터베이스에서 신속하게 검색하고 업데이트 할 수 있음. 즉, 의사결정 속도가 높아지고 업무 프로세스가 개선될 것으로 전망

비즈니스 개선

프로젝트 개요

재고관리프로그램
인벤토리매니저

01

주요 기능 개요

1. 자동화된 재고 관리
2. 매입 및 매출의 기록 관리
3. 데이터베이스 활용

02

대상 이용자

1. 소규모 쇼핑몰 운영자
2. 매입 및 매출 관련 실무자
3. 재고관리 담당자

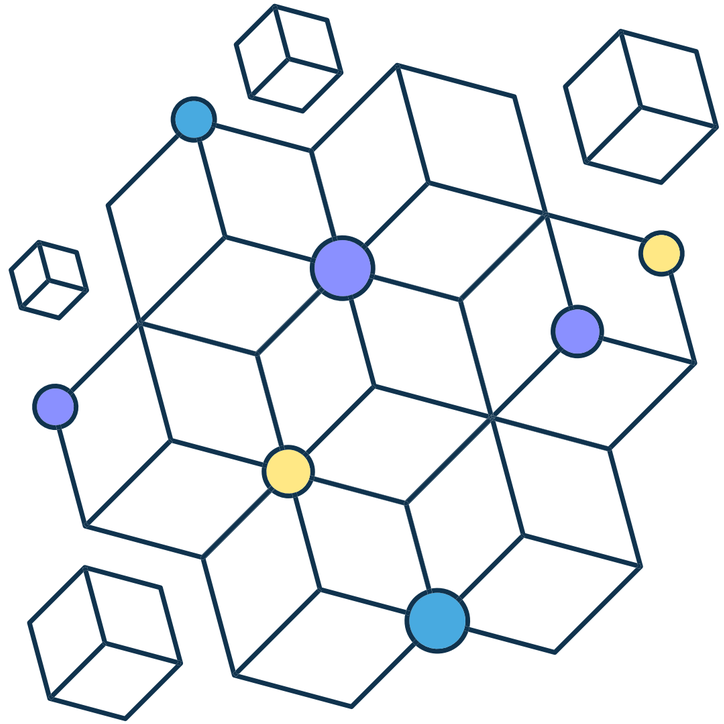
03

효과

1. 의사결정을 위한 무결성 데이터 획득
2. 쇼핑몰의 재고 및 기타 상품관리의 효율성 향상

기술 스택

1. 프로그래밍 언어 : 파이썬
2. 데이터베이스 : 오라클
3. 데이터베이스 연결 라이브러리 : cx.Oracle
4. 데이터 시각화 : PrettyTable
5. 주피터노트북



기술 스택

파이썬 

강점

1. 가독성과 간결성
2. 다양한 라이브러리와 프레임워크
3. 크로스 플랫폼 지원
4. 커뮤니티와 생태계

약점

1. 성능
2. GIL
3. 리소스 사용
4. 모바일 앱 개발
5. 컴파일 언어와의 상호 운용성

기술 스택

오라클

오라클데이터베이스는 오라클이 개발하고 유지 관리하는 관계형 데이터베이스 관리 시스템(RDBMS)

엔터프라이즈급

대규모 엔터프라이즈급 환경에서도 사용 가능한 고성능 데이터베이스 시스템

스케일 아웃 및 스케일 업

확장 가능한 아키텍처

높은 가용성

고가용성을 제공하는 클러스터링 및 복제 기능 지원

보안

강력한 보안 기능 제공

데이터 무결성

데이터 무결성을 보장하며 ACID 트랜잭션 지원

기술 스택

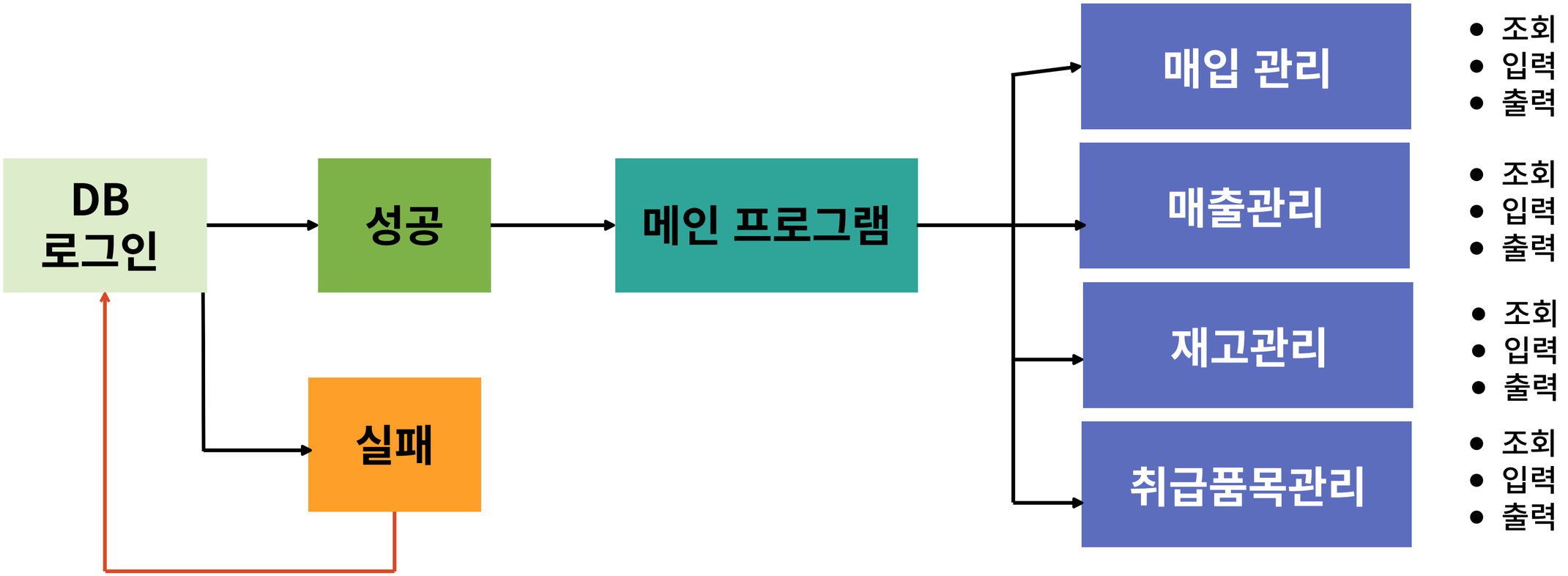
모듈s

cx_Oracle	파이썬 프로그램에서 오라클 데이터베이스와 상호작용하기 위한 라이브러리 연결 : cx_Oracle.connect() 연결 닫기 : connection.close() 커서(cursor) : sql 쿼리를 실행하는 중요한 객체. 커서 객체를 이용해서 오라클과 파이썬의 연결 프로그램을 실행하는 것을 지원. 파이썬 => 오라클 : cursor.execute() (오라클에서 쿼리를 실행) 오라클 => 파이썬 : cursor.fetchall(), one이나 many도 있음
PrettyTable	텍스트 형식의 표를 출력 다양한 출력 형식 지원 정렬 기능 지원

개발 환경

1. 파이썬IDE : 주피터 노트북
2. 버전 관리 : Git(예정)

프로그램 구조


DB
로그인
성공
실패
메인 프로그램
매입 관리
매출관리
재고관리
취급품목관리
조회
입력
출력
조회
입력
출력
조회
입력
출력
조회
입력
출력

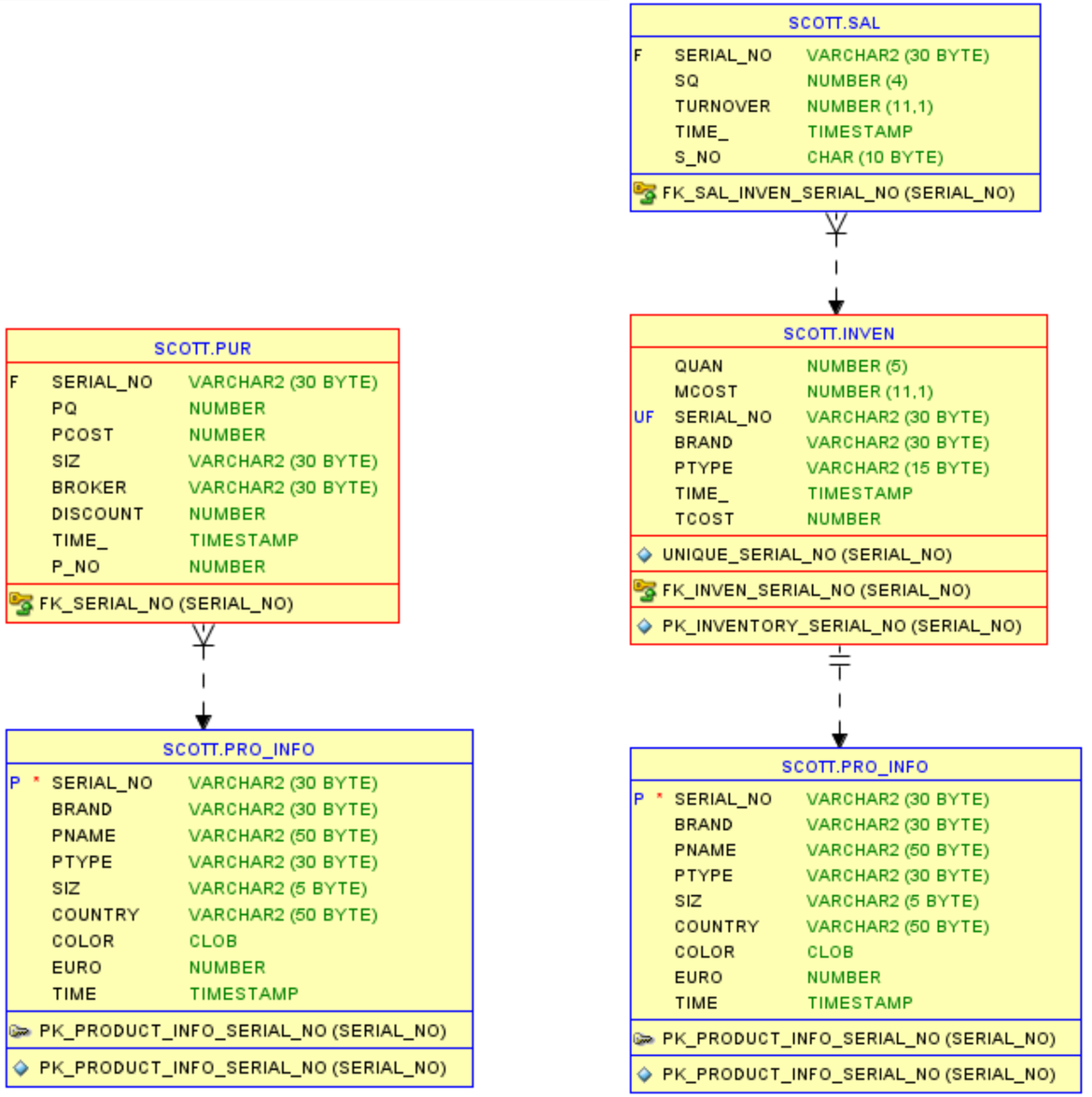
프로그램 구조

1. 데이터베이스 구조

PRO_INFO	INVEN	PUR	SAL
시리얼넘버(serial_no)	시리얼넘버(serial_no)	시리얼넘버(serial_no)	시리얼넘버(serial_no)
브랜드(brand)	재고수량(quan)	매입수량(pq)	매출수량(sq)
상품명(pname)	평균원가(mcost)	총매입가(pcost)	매출액(turnober)
상품종류(ptype)	총매입원가(tcost)	거래처(broker)	사이즈(siz)
사이즈(siz)	마지막업데이트시각(time)	할인률(discount)	마지막업데이트시각(time)
컬러(color)		마지막업데이트시각(time)	
생산국(country)			
원가(euro)			

프로그램 구조

1. 데이터베이스 구조



프로그램 구조

2. 프로그램 구조

- **class DBconnection 구조**

```
def __init__(self, user, password, tab=None, value=None): # 기본 어트리뷰트
def __enter__(self): # with 블록이 시작될 때 호출
def __exit__(self, exc_type, exc_value, traceback): # with 블록이 끝날 대 호출

def choice(self, selec_cols): # 메뉴에서 조회, 삽입, 수정 메서드를 호출하는 메서드

def look(self): # 사용자로부터 입력받은 검색 조건으로 테이블을 조회
def insert(self): # 시리얼 넘버를 기준으로 레코드를 추가
def update(self): # 시리얼 넘버를 기준으로 기존 레코드를 수정
```

프로그램 구조

2. 프로그램 구조

- **class DBconnection 구조**

테이블명 작업번호(1. 조회, 2. 삽입, 3. 수정)

```
with DBconnection(user, password, tab="inven", value=value) as inventory:
    inventory.choice(selec_cols=["serial_no", "quan", "tcost", "brand", "ptype"])
```

입력이나 수정 가능한 열

프로그램 구조

2. 프로그램 구조

- class DBconnection 구조

```python
import cx_Oracle as cx
from prettytable import PrettyTable

class DBconnection:

    def __init__(self, user, password, tab=None, value=None):
        self.user = user
        self.password = password
        self.dns = "localhost:1521/xe"
        self.tab = tab
        self.value = value
        self.conn = None
        self.cursor = None
        self.tab_col_names = []
        self.conditions = {}
        self.ch = None
        self.tab_col_names = None

        # 유저권한 확인하고 진입
        try:
            self.conn = cx.connect(self.user, self.password, self.dns)
            self.cursor = self.conn.cursor()
            self.cursor.execute(f"SELECT * FROM {self.tab}")
            self.tab_col_names = [desc[0] for desc in self.cursor.description]

        except Exception:
            print("에러입니다.")
        except cx.DatabaseError as e1:
            print("\nDB 접속에 실패했습니다.")
            print("아이디와 비밀번호를 다시 확인해 주세요.")
            self.conn = None
            self.cursor = None
            print("self.conn")
            exit(1)

    def __enter__(self):
        return self

    def __exit__(self, exc_type, exc_value, traceback):
        if self.conn:
            self.conn.close()
```

프로그램 구조

2. 프로그램 구조

- class DBconnection 구조

```
    ## 메인메뉴 선택 메서드 -------------
    def choice(self, selec_cols):
        self.selec_cols=selec_cols
        if self.value == "1":
            self.look()
        elif self.value == "2":
            self.insert()
        elif self.value == "3":
            self.update()
        else:
            pass
    ## 조건별 조회 메서드 -------------
    def look(self):
        print("검색 조건 입력\n(공백입력시 전체 테이블 조회)")
        valid_conditions = {}
        for col_name in self.tab_col_names:
            valu = input(f"{col_name} (없으면 Enter): ").strip()
            if valu:
                valid_conditions[col_name] = valu

        if valid_conditions:
            condition_str = " AND ".join([f"{col_name} LIKE :{col_name}" for col_name in valid_conditions])
            query = f"SELECT * FROM {self.tab} WHERE {condition_str}"
        else:
            query = f"SELECT * FROM {self.tab}"
        try:
            self.cursor.execute(query, valid_conditions)
            rows = self.cursor.fetchall()

            if not rows:
                print("검색 결과가 없습니다.")
            else:
                table = PrettyTable()
                table.field_names = self.tab_col_names
                for row in rows:
                    table.add_row(row)
                print(table)

        except cx.DatabaseError as e:
            print("데이터베이스 오류:", e)
        except Exception as e:
            print("오류 발생:", e)
```

프로그램 구조

2. 프로그램 구조

- class DBconnection 구조

```
    ## 입력 메서드 -------------
    def insert(self):
        print(self.selec_cols)

        input_values = {}
        valid_columns = []

        for col in self.selec_cols:
            val = input(f"{col}: ")
            if val:
                input_values[col] = val
                valid_columns.append(col)

        if not valid_columns:
            print("입력된 데이터가 없습니다.")
            return

        columns = ', '.join(valid_columns)

        set_statement = ', '.join([f":{col}" for col in valid_columns])
        query = f"INSERT INTO {self.tab} ({columns}) VALUES ({set_statement})"

        try:
            self.cursor.execute(query, input_values)
            self.cursor.connection.commit()
            print(f"{self.tab}에 레코드가 추가되었습니다.")
        except Exception as e:
            print("데이터 삽입 중 오류 발생", e)

    ## 수정하기 메소드 -------------
    def update(self):
        print(self.selec_cols)
        serial_no = input("상품 시리얼 넘버 : ")

        input_values = {}
        valid_columns = []

        for col in self.selec_cols:
            val = input(f"{col}: ")
            if val:
                input_values[col] = val
                valid_columns.append(col)

        set_statement = ', '.join([f":{col}" for col in input_values])
        valid_columns_str = ', '.join(valid_columns)  # 유효한 컬럼들을 문자열로 변환
        query = f"UPDATE {self.tab} SET {valid_columns_str} = {set_statement} WHERE serial_no = :serial_no"

        input_values["serial_no"] = serial_no

        try:
            self.cursor.execute(query, input_values)
            self.cursor.connection.commit()
            print(f"{serial_no}의 정보가 변경되었습니다.")
        except Exception as e5:
            print("잘못된 입력입니다.")
```

프로그램 구조

2. 프로그램 구조

- main() 구조

```
    ## 메인 프로그램 -------------
def main():
    print("\n\n\n----------------재고관리 프로그램----------------")

    user = input("DB 아이디: ")
    password = input("DB 비밀번호: ")
    while True:
        prompt = '''
            ---------------- 메인 프로그램  ----------------
            1.매입::2.매출::3.재고상품::4.취급상품::5.종료
            : '''
        ch = input(prompt)

        if ch == "1":
            prompt = '''
            -------------------- 매입관리창 --------------------
            1. 매입기록 조회 :: 2. 매입기록 입력 :: 3. 매입기록 수정
            : '''
            value = input(prompt)
            with DBconnection(user, password, tab="pur", value=value) as purchase:
                purchase.choice(selec_cols=["serial_no", "pq", "pcost", "siz", "broker", "discount"])
        elif ch == "2":
            prompt = '''
            -------------------- 매출관리창 --------------------
            1. 매출기록 조회 :: 2. 매출기록 입력 :: 3. 매출기록 수정
            : '''
            value = input(prompt)
            with DBconnection(user, password, tab="sal",  value=value) as sales:
                sales.choice(selec_cols=["serial_no", "sq", "turnover"])
        elif ch == "3":
            prompt = '''
            -------------------- 재고관리창 --------------------
            1. 재고상품 조회 :: 2. 재고상품 입력 :: 3. 재고상품 수정
            : '''
            value = input(prompt)
            with DBconnection(user, password, tab="inven", value=value) as inventory:
                inventory.choice(selec_cols=["serial_no", "quan", "tcost", "brand", "ptype"])
        elif ch == "4":
            prompt = '''
            -------------------- 취급상품창 --------------------
            1. 취급상품 조회 :: 2. 취급상품 입력 :: 3. 취급상품 수정
            : '''
            value = input(prompt)
            with DBconnection(user, password, tab="pro_info", value=value) as product_info:
                product_info.choice(selec_cols=["serial_no",
                                                "brand",
                                                "pname",
                                                "ptype", "siz", "country", "color", "euro"])
        elif ch == "5":
            print("프로그램 종료")
            break
        else:
            print("1, 2, 3, 4, 5 중에서 선택하세요.")


if __name__ == "__main__":
    main()
```

프로그램 기능

1. 매입 기록 관리

- 매입기록 조회 : 조건을 입력하여 매입 기록 조회
- 매입기록 입력 : 새로운 매입 기록을 데이터베이스에 입력
- 매입기록 수정 : 기존의 매입 기록을 수정

2. 매출 기록 관리

- 매출기록 조회 : 조건을 입력하여 매출 기록 조회
- 매출기록 입력 : 새로운 매출 기록을 데이터베이스에 입력
- 매출기록 수정 : 기존의 매출 기록을 수정

3. 재고 상품 관리

- 매입기록 조회 : 조건을 입력하여 매입 기록 조회
- 매입기록 입력 : 새로운 매입 기록을 데이터베이스에 입력
- 매입기록 수정 : 기존의 매입 기록을 수정

4. 취급 상품 정보 관리

- 취급 상품 조회 : 사용자가 취급 상품 정보를 조회
- 취급 상품 입력 : 사용자가 새로운 취급 상품 정보를 데이터베이스에 입력
- 취급 상품 수정 : 사용자가 기존 취급 상품 정보를 수정

프로그램 인터페이스

초기 실행 화면

```
---------------재고관리 프로그램---------------
DB 아이디: scott
DB 비밀번호: 1234
```

로그인 성공

```
---------------   메인 프로그램   ---------------

1.매입::2.매출::3.재고상품::4.취급상품::5.종료

입력 :
```

로그인 실패

```
 로그인 실패! 아이디와 비밀번호를 다시 확인해 주세요.
```

프로그램 인터페이스

1. 매입

```
---------------- 매입관리창 ----------------
1.매입기록 조회::2.매입기록 입력::3.매입기록 수정
 :
```

프로그램 인터페이스

2. 매출

```
---------------- 매출관리창 ----------------
1.매출기록 조회::2.매출기록 입력::3.매출기록 수정
 :
```

프로그램 인터페이스

3. 재고상품

```
---------------- 재고관리창 ----------------
1.재고상품 조회::2.재고상품 입력::3.재고상품 수정
 :
```

프로그램 인터페이스

4. 취급상품

```
---------------- 취급상품창 ----------------
1.취급상품 조회::2.취급상품 입력::3.취급상품 수정
 :
```

프로그램 인터페이스

4. 취급상품 -> 1 (취급상품 조회)

```
--------------- 취급상품창 ---------------
1.취급상품 조회::2.취급상품 입력::3.취급상품 수정
 : 1
검색 조건 입력
(공백입력시 전체 테이블 조회)
SERIAL_NO (없으면 Enter): 730828VCPP07671
BRAND (없으면 Enter):
PNAME (없으면 Enter):
PTYPE (없으면 Enter):
SIZ (없으면 Enter):
COUNTRY (없으면 Enter):
COLOR (없으면 Enter):
EURO (없으면 Enter):
TIME (없으면 Enter):
+----------------+-------------+----------+-------+-----+---------+-------+------+------+
|    SERIAL_NO   |    BRAND    |   PNAME  | PTYPE | SIZ | COUNTRY | COLOR | EURO | TIME |
+----------------+-------------+----------+-------+-----+---------+-------+------+------+
| 730828VCPP07671 | 보테가베네타 | 캔디 조디 |  가방 | None | 이탈리아 |  핑크 | 1300 | None |
+----------------+-------------+----------+-------+-----+---------+-------+------+------+
```

프로그램 인터페이스

4. 취급상품 -> 2
(취급상품 입력)

```
---------------- 취급상품창 ----------------
1.취급상품 조회::2.취급상품 입력::3.취급상품 수정
 : 2
['serial_no', 'brand', 'pname', 'ptype', 'siz', 'country', 'color', 'euro']
serial_no: 755031VCQC49009
brand: BottegaVeneta
pname: 미니 카세트 숄더백
ptype: 가방
siz:
country: 이탈리아
color: 화이트
euro: 1300
pro_info에 레코드가 추가되었습니다.
---------------   메인 프로그램   ---------------
```

프로그램 인터페이스

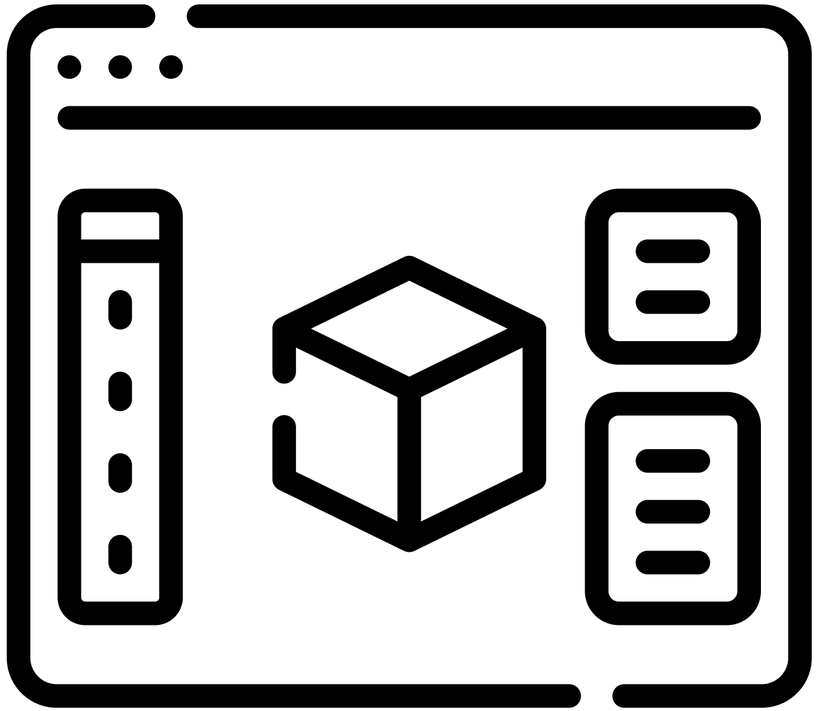
4. 취급상품 -> 3
(취급상품 수정)

```
---------------- 취급상품창 ----------------
1.취급상품 조회::2.취급상품 입력::3.취급상품 수정
 : 3
['serial_no', 'brand', 'pname', 'ptype', 'siz', 'country', 'color', 'euro']
상품 시리얼 넘버 : 730828VCPP07671
serial_no:
brand: BottegaVeneta
pname:
ptype:
siz:
country:
color:
euro:
730828VCPP07671의 정보가 변경되었습니다.
```

4. 취급상품 -> 1
(취급상품 조회)

```
| 730828VCPP07671 | BottegaVeneta |          캔디 조디          | 가방 | None
+-----------------+---------------+-----------------------------+------+-----
```

프로젝트 성과 및 시연



재고관리 프로그램(베타) 완성

DB + 파이썬 프로그램

취급상품정보 테이블 DBconnection 클래스
재고상품 테이블 choice 메서드
매입 기록 look 메서드
매출 기록 insert 메서드
update 메서드

남은 과제 및 업그레이드 계획

데이터베이스	추가 사용자 권한 관리	몇 일 내로
데이터베이스	데이터베이스 아키텍처 더 견고하게	입력 넘버링, 시간 입력, 순서 정렬
데이터베이스	백업 시스템	몇 일 내로
시스템	시리얼 넘버, 원가 웹에서 크롤링으로 참조할 수 있도록	2023년 안으로 크롤링 모듈 제작
시스템	DB 접속시 보안 관리 업그레이드	-
파이썬	코드 안정성	2023년 안으로 베타버전1 마무리
파이썬	메뉴에 분석 추가	분석 수업 후 모듈 제작
기타	...	

Q&A

감사합니다.