

## 2022 데이터 베이스 HLL-Report

숙명매점은 학교안에 있는 작은 매점으로 웹 키오스크에서 고객에게 주문을 받는다. 매점 관리자는 python으로 짜여진 프로그램으로 매점에 대한 관리를 한다.



프로그램의 메인 화면이다. 5개의 버튼으로 구성되어 있다.

재고현황, 매출현황, 재고주문서 버튼을 누르면 각 항목에 대한 테이블을 조회할 수 있다. 재고입고 버튼을 누르면 입고할 재고의 수량을 추가할 수 있다. 신상품입고 버튼은 매점DB에 신상품에 관한 정보를 한 번에 추가할 수 있는 기능을 제공한다.

재고현황				
	상품번호	상품명	재고수량	기본수량
1	1	삼다수	18	40
2	2	삼각김밥	12	12
3	3	진라면	8	20
4	4	왕뚜껑	23	15
5	5	불닭볶음면	0	15
6	6	가나초콜릿	12	10
7	7	파워에이드	20	15
8	8	제티	24	20
9	9	오징어땅콩	9	15
10	10	하리보젤리	26	20
11	11	힐우유	9	10
12	12	바나나우유	1	10
13	13	커피	23	20
14	14	에비앙	15	15
15	15	포켓몬빵	0	30
16	16	초코파이	14	12
17	17	백산수	6	20
18	18	오레오	21	15
19	19	진짬뽕	23	20
20	20	소세지빵	26	25
21	21	크림빵	9	25
22	22	자유시간	31	20
23	23	전주비빔삼각	9	10
24	24	애플잼파이	11	12
25	25	빠빼코	52	30
26	26	더위사냥	37	30
27	27	보석바	26	30

재고현황 버튼을 눌러보았다. 재고현황 탭에서는 각 상품의 상품번호, 상품명, 현재재고, 기본 수량을 테이블로 보여준다. 기본 수량에 현재재고 수량이 못 미칠 시, 재고주문이 필요하므로 이를 편하게 확인할 수 있다. 예를 들면 1번 삼다수는 현재 재고량이 기본 재고량보다 적기 때문에 재고 주문이 필요하다. 추후 다시 나오겠지만 재고주문서 탭에서 자동으로 작성된 재고주문서도 확인할 수 있다. 테이블은 마우스로 스크롤 가능하다.

```
if code=='1':
    sqlCmd = "SELECT * FROM stock_view;"
    message = "재고현황"
```

테이블에서는 재고현황을 볼 수 있는 뷰인 stock\_view를 select한 데이터를 받아 사용한다.

매출현황					
	주문시각	상품번호	상품명	수량	가격
1	2022-05-21/21:07:07	21	크림빵	2	2000
2	2022-05-21/21:07:25	19	진짬뽕	1	1100
3	2022-05-21/21:07:38	14	에비앙	3	6000
4	2022-05-21/22:09:18	5	불닭볶음면	2	2400
5	2022-05-21/22:19:00	17	백산수	3	2400
6	2022-05-21/22:19:12	6	가나초콜릿	1	800
7	2022-05-21/22:39:20	10	하리보젤리	2	2200
8	2022-05-22/10:10:29	22	자유시간	3	4200
9	2022-05-22/10:10:56	13	커피	1	900
10	2022-05-22/10:11:08	16	초코파이	2	4800
11	2022-05-22/10:51:11	20	소세지빵	1	1000
12	2022-05-22/14:57:56	18	오레오	1	1900
13	2022-05-22/15:56:56	10	하리보젤리	2	2200
14	2022-05-22/21:21:40	27	보석바	2	2000
15	2022-05-22/21:40:44	5	불닭볶음면	5	6000

매출현황 버튼을 눌러보았다. 고객들이 웹에서 주문한 매출 기록들이 반영되어 테이블로 보여 진다.

```
elif code=='2':
    sqlCmd = "SELECT ordertime, s.pnum, pname, qty, totalprice FROM
salesSlip s, product p WHERE s.pnum = p.pnum;"
    message = "매출현황"
```

테이블이 사용하는 데이터 SQL연산이다.

매출현황 테이블인 salesSlip 테이블에는 원래 pname이 포함되지 않는다. 하지만 프로그램에서는 상품 이름이 있어야 보기 편하기 때문에 product테이블과 조인연산을 해 pname도 함께 select한다.

재고주문서						
	상품번호	상품명	주문수량	도매가격	공급자이름	공급자전화번호
1	1	삼다수	40	16000	이말숙	01038979493
2	3	진라면	25	10000	강민수	01030589823
3	5	불닭볶음면	20	14000	강민수	01030589823
4	9	오징어땅콩	15	9000	마창진	01003948204
5	11	힘우유	20	8000	박영아	01020398532
6	12	바나나우유	20	12000	박영아	01020398532
7	15	포켓몬빵	20	18000	이민혁	01047502948
8	17	백산수	40	14000	이말숙	01038979493
9	21	크림빵	35	17500	머큐리	01088369920
10	23	전주비빔삼각	10	8000	김정민	01048960395
11	24	애플잼파이	20	11000	이민혁	01047502948
12	27	보석바	40	10000	조자룡	01098038212

재고주문서 버튼을 눌러보았다. 위에서 설명하였듯, 현재 재고량이 기본 재고량보다 작으면 재고주문서가 작성된다. 주문수량은 wholesale테이블에서 도매수량 만큼을 주문하는 것으로 한다.

테이블은 상품번호, 상품명, 주문수량, 도매가격, 공급자이름, 공급자 전화번호를 포함한다. 이 테이블은 프로그램의 테이블 중 가장 많은 4가지의 데이터베이스 릴레이션이 사용된다.

코드 발췌 (indentation때문에 줄 바꿈을 하면 안 되지만 가독성을 위해 해보았다.)

[2점: 테이블/조인 기능 존재 여부 평가]

```
sqlCmd = "SELECT stk.pnum, p.pname, w.qty, w.wprice*w.qty, spl.sname,
spl.phone
FROM stock stk, product p, wholesale w, supplier spl
WHERE stk.pnum = p.pnum
AND stk.pnum = w.pnum
AND p.snum = spl.snum
AND stk.qty_current < stk.qty_reorder;"
```

위 쿼리는 product(상품), stock(재고), wholesale(도매), supplier(공급자) 4가지 테이블을 3중 조인한

다. 앞의 3개 테이블은 pnum으로 조인하고, 공급자테이블은 product의 snum을 사용해 조인한다. 또한 현재 재고수량이 기본 리오더 재고수량보다 작은 것만이 select되게 조건을 작성하였다.

재고주문서 탭에서는 재고를 주문할 때 편의를 위해 테이블을 csv파일로 내보낼 수 있는 기능이 있다. 테이블 하단의 '내보내기' 버튼을 클릭하면 현재 테이블이 "ordersheet.csv"파일로 보내진다. 한 번 클릭해보겠다.

이름	수정한 날짜	유형	크기
.idea	2022-05-18 오전 3:20	파일 폴더	
main.py	2022-05-23 오후 7:02	JetBrains PyChar...	11KB
ordersheet.csv	2022-05-23 오후 7:14	Microsoft Excel ...	1KB
test.py	2022-05-16 오전 3:30	JetBrains PyChar...	0KB

파일이 생성되었다.

	A	B	C	D	E	F	
1	상품번호	상품명	주문수량	도매가격	공급자이름	공급자전화번호	
2	1	삼다수	40	16000	이말숙	01038979493	
3	3	진라면	25	10000	강민수	01030589823	
4	5	불닭볶음면	20	14000	강민수	01030589823	
5	9	오징어땅콩	15	9000	마창진	01003948204	
6	11	흰우유	20	8000	박영아	01020398532	
7	12	바나나우유	20	12000	박영아	01020398532	
8	15	포켓몬빵	20	18000	이민혁	01047502948	
9	17	백산수	40	14000	이말숙	01038979493	
10	21	크림빵	35	17500	머큐리	01088369920	
11	23	전주비빔밥	10	8000	김정민	01048960395	
12	24	애플잼파오	20	11000	이민혁	01047502948	
13	27	보석바	40	10000	조자룡	01098038212	
14							
15							

정상적으로 데이터 파일이 내보내 졌다.

코드 발췌

```
df.to_csv('./ordersheet.csv',index=False, encoding='euc-kr')
```

재고입고

재고입고

상품번호1

입고수량

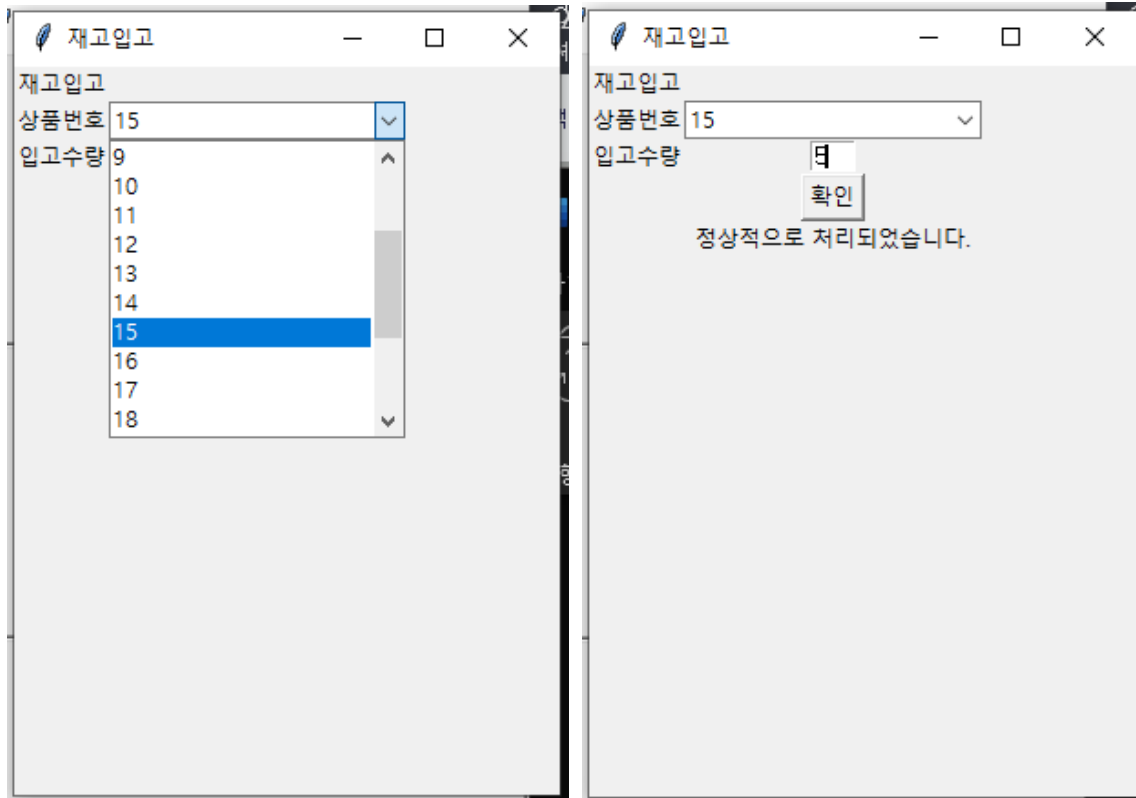
확인

재고입고 버튼을 눌러보았다. 이곳에서는 상품 번호로 재고를 입고시킬 수 있다.

재고현황

	상품번호	상품명	재고수량	기본수량
1	1	삼다수	18	40
2	2	삼각김밥	12	12
3	3	진라면	8	20
4	4	왕뚜껑	23	15
5	5	불닭볶음면	0	15
6	6	가나초콜릿	12	10
7	7	파워에이드	20	15
8	8	제티	24	20
9	9	오징어땅콩	9	15
10	10	하리보젤리	26	20
11	11	흰우유	9	10
12	12	바나나우유	1	10
13	13	커피	23	20
14	14	에비앙	15	15
15	15	포켓몬빵	0	30
16	16	소코파이	14	12
17	17	백산수	6	20
18	18	오레오	21	15
19	19	진짬뽕	23	20
20	20	소세지빵	26	25
21	21	크림빵	9	25
22	22	자유시간	31	20
23	23	전주비빔삼각	9	10
24	24	애플잼파이	11	12
25	25	빠빠코	52	30
26	26	더위사냥	37	30
27	27	보석바	26	30

현재 재고는 이런 상태이다. 한 번 품질인 포켓몬 빵을 입고시켜 보겠다.



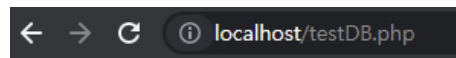
포켓몬빵은 15번이므로 콤보 박스에서 15번을 선택한 후 5개를 입고시켜보았다. 버튼을 누르자 정상적으로 처리되었다는 메시지가 나온다.

	상품번호	상품명	재고수량	기본수량
1	1	삼다수	18	40
2	2	삼각김밥	12	12
3	3	진라면	8	20
4	4	왕뚜껑	23	15
5	5	불닭볶음면	0	15
6	6	가나초콜릿	12	10
7	7	파워에이드	20	15
8	8	제티	24	20
9	9	오징어땅콩	9	15
10	10	하리보젤리	26	20
11	11	흰우유	9	10
12	12	바나나우유	1	10
13	13	커피	23	20
14	14	에비앙	15	15
15	15	포켓몬빵	5	30
16	16	초코파이	14	12
17	17	백산수	6	20
18	18	오레오	21	15
19	19	진짬뽕	23	20
20	20	소세지빵	26	25
21	21	크림빵	9	25
22	22	자유시간	31	20
23	23	전주비빔삼각	9	10
24	24	애플잼파이	11	12
25	25	빠빠코	52	30
26	26	더위사냥	37	30
27	27	보석바	26	30



포켓몬 빵의 재고가 정상적으로 입고되었다. 웹에서도 확인해 보겠다.

[\*\*\* Cross Check 점수 신청: 고급언어-->웹 검증: 1점 신청 \*\*\*]



## 숙명매점

상품번호	상품명	가격	남은수량
1	삼다수	1000	18
2	삼각김밥	1600	12
3	진라면	1000	8
4	왕뚜껑	1000	23
5	불닭볶음면	1200	0
6	가나초콜릿	800	12
7	파워에이드	1900	20
8	제티	700	24
9	오징어땅콩	1300	9
10	하리보젤리	1100	26
11	흰우유	1000	9
12	바나나우유	1200	1
13	커피	900	23
14	에비앙	2000	15
15	포켓몬빵	1500	5
16	초코파이	2400	14
17	백산수	800	6
18	오레오	1900	21
19	진짬뽕	1100	23
20	소세지빵	1000	26
21	크림빵	1000	9
22	자유시간	1400	31
23	전주비빔삼각	1500	9
24	애플잼파이	1100	11
25	빠빠코	1000	52
26	더위사냥	1000	37
27	보석바	1000	26

웹에서도 재고 5개가 정상적으로 들어왔다.

코드발취

```
elif code=='4': #재고현황
```

```
    sqlCmd = "SELECT pnum FROM stock;"
```

먼저 콤보박스를 작성하기 위해 pnum목록을 select한다.

```
#SQL
```

```
    sqlCmd = "UPDATE stock SET qty_current = qty_current+%d WHERE pnum  
= %d;"
```

```
    try:
```

```
        cursor.execute(sqlCmd %(qty, pnum))
```

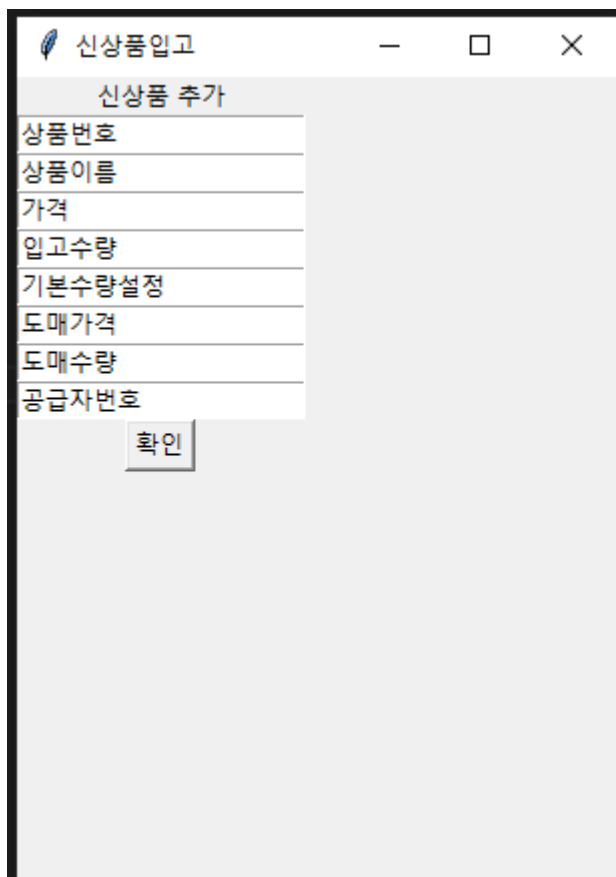
```
        con.commit()
```

```
    except: ttk.Label(self.newWin, text="ERROR").grid(row=4, column=1)
```

```
    else: ttk.Label(self.newWin, text="정상적으로  
처리되었습니다.").grid(row=4, column=1)
```

사용자에게 입력받은 pnum과 qty를 이용해 stock테이블에 업데이트 하는 쿼리를 사용한다. Qty는 원래의 재고수량인 qty\_current와 더하기 연산을 하여 새로운 qty\_current로 DB에 업데이트 된다. 조건은 pnum이 pnum(사용자가 입력한 변수)인 것으로 한다.

연산이 정상적으로 실행되면 화면에 Label을 출력한다. 비정상이면 "ERROR" 라벨을 출력한다.



마지막으로 신상품입고 버튼을 눌러보았다. 여기서는 신상품을 추가할 수 있다. 신상품을 DB에 추가하기 위해서는 입력해야 할 사항이 많다.

신상품입고

— □ ×

신상품 추가

정상 처리 되었습니다.

28
kitkat
1200
30
20
550
20
10

확인

다음과 같이 신상품 kitkat 초콜릿을 입고했다. 웹에서도 보이는지 확인해보겠다.

[\*\*\* Cross Check 점수 신청: 고급언어-->웹 검증 \*\*\*] (2)

23	전주비빔삼각	1500	9
24	애플잼파이	1100	11
25	빠빠코	1000	52
26	더위사냥	1000	37
27	보석바	1000	26
28	kitkat	1200	30

상품명

수량

결제

웹에서도 새로운 상품이 추가된 것이 보인다.

25	25	빠빠코	52	30
26	26	더위사냥	37	30
27	27	보석바	26	30
28	28	kitkat	30	20

고급언어 프로그램의 재고현황 탭에서도 신상품이 업데이트된 것을 확인 가능하다.

그런데 한가지 간과한 상황이 있다. 만약 신상품을 추가할 때 새로운 공급자에게서 제공받게 될 경우도 있다. 이럴 땐 어떻게 될까?

수입과자 Flipz를 신상품으로 입고하려 하는데 이 과자는 새로운 공급자인 11번 Richard에게서 공급받는다. 공급자 번호에 현재 없는 공급자인 11번을 입력하자 새로운 공급자 정보를 저장하기 위해 공급자이름, 공급자 전화번호를 입력하는 칸이 나왔다. 만약 원래부터 공급자이름, 공급자 전화번호를 입력하는 칸이 항상 있었다면, 기존에 있는 공급자를 입력할 때마다 불편했을 것이다.

신상품입고

신상품 추가    정상 처리 되었습니다.

29

Flipz

3000

25

20

1900

30

11

확인

Richard

01039357747

확인

새로운 공급자 정보를 입력하고 확인을 누르자 신상품 추가가 정상 처리되었다는 메시지가 나왔다.

28	28	kitkat	30	20
29	29	Flipz	25	20

25	빠빼코	1000	52
26	더위사냥	1000	37
27	보석바	1000	26
28	kitkat	1200	30
29	Flipz	3000	25

상품명

상품수량

재고현황과 웹에서도 정상적으로 처리되었다.

29	Flipz	3000	25
----	-------	------	----




20개를 모조리 주문해 재고를 5개로 만들었다.

매출현황

	주문시각	상품번호	상품명	수량	가격
1	2022-05-21/21:07:07	21	크림빵	2	2000
2	2022-05-21/21:07:25	19	진짬뽕	1	1100
3	2022-05-21/21:07:38	14	에비앙	3	6000
4	2022-05-21/22:09:18	5	불닭볶음면	2	2400
5	2022-05-21/22:19:00	17	백산수	3	2400
6	2022-05-21/22:19:12	6	가나초콜릿	1	800
7	2022-05-21/22:39:20	10	하리보젤리	2	2200
8	2022-05-22/10:10:29	22	자유시간	3	4200
9	2022-05-22/10:10:56	13	커피	1	900
10	2022-05-22/10:11:08	16	초코파이	2	4800
11	2022-05-22/10:51:11	20	소세지빵	1	1000
12	2022-05-22/14:57:56	18	오레오	1	1900
13	2022-05-22/15:56:56	10	하리보젤리	2	2200
14	2022-05-22/21:21:40	27	보석바	2	2000
15	2022-05-22/21:40:44	5	불닭볶음면	5	6000
16	2022-05-23/11:04:32	29	Flipz	20	60000

재고주문서

	상품번호	상품명	주문수량	도매가격	공급자이름	공급자전화번호
1	1	삼다수	40	16000	이말숙	01038979493
2	3	진라면	25	10000	강민수	01030589823
3	5	불닭볶음면	20	14000	강민수	01030589823
4	9	오징어땅콩	15	9000	마창진	01003948204
5	11	흰우유	20	8000	박영아	01020398532
6	12	바나나우유	20	12000	박영아	01020398532
7	15	포켓몬빵	20	18000	이민혁	01047502948
8	17	백산수	40	14000	이말숙	01038979493
9	21	크림빵	35	17500	머큐리	01088369920
10	23	전주비빔상각	10	8000	김정민	01048960395
11	24	애플잼파이	20	11000	이민혁	01047502948
12	27	보석바	40	10000	조자룡	01098038212
13	29	Flipz	30	57000	Richard	01039357747

내보내기

매출현황과 재고주문서에도 모든 정보가 정상적으로 표기된다.

코드발췌

```
snum = int(self.snum.get())
sqlCmd = "SELECT * FROM supplier WHERE snum=%d;"
cursor.execute(sqlCmd % snum)
rows = cursor.fetchall()
```

먼저 첫번째로 정보를 입력하고 확인을 누르면, 입력받은 snum을 받아 select해 그 snum을 가진 공급자가 있는지를 체크한다.

```
if rows == []:
    self.sname = tk.Entry(self.newWin)
    self.sname.grid(column=0, row=11)
    self.sname.insert(0, "공급자이름")
    self.sphone = tk.Entry(self.newWin)
    self.sphone.grid(column=0, row=12)
    self.sphone.insert(0, "공급자전화번호")
    tk.Button(self.newWin, text="확인",
command=self.add_splr).grid(column=0, row=13)
else : self.add_new()
```

만약 공급자가 없다면 text박스를 추가해 공급자이름, 공급자전화번호를 더 입력 받는다.

공급자가 있다면 바로 데이터 삽입단계로 넘어간다.

```
snum=int(self.snum.get())
sname=self.sname.get()
sphone= str(self.sphone.get())
sqlCmd = "set identity_insert supplier on; INSERT INTO supplier
([snum],[sname],[phone]) VALUES (%d, '%s', '%s'); set identity_insert supplier
off;"

try:
    print(chardet.detect(sname.encode()))
    cursor.execute(sqlCmd % (snum, sname, sphone))
    con.commit()
```

입력 받은 새로운 공급자 정보를 먼저 supplier 테이블에 추가한다.

SET identity insert supplier on/off 구문은 테이블의 snum 칼럼에 identity(1,1)조건으로 번호가 자동 증가하게 해줬었는데 이를 어기고 직접 snum을 지정해 삽입하기 위해 사용했다.

```
sqlCmd1 = "set identity_insert product on; INSERT INTO product ([pnum],
[pname],[price],[snum]) VALUES (%d, '%s', %d, %d); set identity_insert product
off;"
sqlCmd2 = "INSERT INTO stock VALUES (%d, %d, %d);"
sqlCmd3 = "INSERT INTO wholesale VALUES (%d, %d, %d);"
try:
    print(pnum, pname, price, snum)
    #print(chardet.detect(pname.encode()))
    cursor.execute(sqlCmd1 % (pnum, pname, price, snum))
    con.commit()
    cursor.execute(sqlCmd2 % (pnum, qty, dqty))
    con.commit()
    cursor.execute(sqlCmd3 % (pnum, wprice, wqty))
    con.commit()
except:
    ttk.Label(self.newWin, text="ERROR").grid(row=1, column=1)
else:
    ttk.Label(self.newWin, text="정상 처리 되었습니다.").grid(row=0,
column=15)
```

일반 삽입단계로 넘어가 3개의 테이블 product, stock, wholesale에 각각에 맞는 데이터를 모두 insert하였다.

이로써 고급언어 프로그램 설명과 esql구문 설명을 마친다.

<데이터베이스 구조>



ESQL\_project 데이터베이스는 product, supplier, stock, wholesale, saleSlip 다섯개의 테이블과 stock\_view 한 개의 뷰로 구성되어 있다.

[Product]

상품(상품번호, 상품명, 가격, 공급자번호)

	pnum	pname	price	snum
1	1	삼다수	1000	1
2	2	삼각김밥	1600	2
3	3	진라면	1000	3
4	4	왕뚜껑	1000	3
5	5	불닭볶음면	1200	3
6	6	가나초콜릿	800	5
7	7	파워에이드	1900	6
8	8	제티	700	6
9	9	오징어땅콩	1300	5
10	10	하리보젤리	1100	5
11	11	현우유	1000	7
12	12	바나나우유	1200	7
13	13	커피	900	1
14	14	에비앙	2000	8
15	15	포켓몬빵	1500	9
16	16	초코파이	2400	5
17	17	백산수	800	1
18	18	오레오	1900	5
19	19	진잠빵	1100	3
20	20	소세지빵	1000	10
21	21	크림빵	1000	10
22	22	자유시간	1400	5
23	23	전주비빔...	1500	2
24	24	애플잼파이	1100	9
25	25	빠빠코	1000	4
26	26	더위사냥	1000	4
27	27	보석바	1000	4
28	28	kitkat	1200	10
29	29	Flipz	3000	11

[supplier]

공급자(공급자번호, 공급자이름, 전화번호)

	snum	sname	phone
1	1	이말숙	01038979493
2	2	김정민	01048960395
3	3	강민수	01030589823
4	4	조자룡	01098038212
5	5	마창진	01003948204
6	6	이혜준	01074983850
7	7	박영아	01020398532
8	8	기현준	01099482742
9	9	이만혁	01047502948
10	10	머큐리	01088369920
11	11	Richard	01039357747

[stock]

재고(상품번호, 현재재고수량, 기본재고수량)

	pnum	qty_current	qty_reorder
1	1	18	40
2	2	12	12
3	3	8	20
4	4	23	15
5	5	0	15
6	6	12	10
7	7	20	15
8	8	24	20
9	9	9	15
10	10	26	20
11	11	9	10
12	12	1	10
13	13	23	20
14	14	15	15
15	15	5	30
16	16	14	12
17	17	6	20
18	18	21	15
19	19	23	20
20	20	26	25
21	21	9	25
22	22	31	20
23	23	9	10
24	24	11	12
25	25	52	30
26	26	37	30
27	27	26	30
28	28	30	20
29	29	5	20

[wholesale]

도매(상품번호, 도매가격, 도매주문수량)

	pnum	wprice	qty
1	1	400	40
2	2	700	10
3	3	400	25
4	4	350	20
5	5	700	20
6	6	400	18
7	7	1000	20
8	8	250	20
9	9	600	15
10	10	650	20
11	11	400	20
12	12	600	20
13	13	350	30
14	14	1400	20
15	15	900	20
16	16	1100	20
17	17	350	40
18	18	900	15
19	19	600	20
20	20	500	35
21	21	500	35
22	22	600	35
23	23	800	10
24	24	550	20
25	25	250	40
26	26	250	40
27	27	250	40
28	28	550	20
29	29	1900	30

[salesSlip]

매출현황(주문일시, 상품번호, 수량, 총 가격)

	ordertime	pnum	qty	totalprice
1	2022-05-21/21:07:07	21	2	2000
2	2022-05-21/21:07:25	19	1	1100
3	2022-05-21/21:07:38	14	3	6000
4	2022-05-21/22:09:18	5	2	2400
5	2022-05-21/22:19:00	17	3	2400
6	2022-05-21/22:19:12	6	1	800
7	2022-05-21/22:39:20	10	2	2200
8	2022-05-22/10:10:29	22	3	4200
9	2022-05-22/10:10:56	13	1	900
10	2022-05-22/10:11:08	16	2	4800
11	2022-05-22/10:51:11	20	1	1000
12	2022-05-22/14:57:56	18	1	1900
13	2022-05-22/15:56:56	10	2	2200
14	2022-05-22/21:21:40	27	2	2000
15	2022-05-22/21:40:44	5	5	6000
16	2022-05-23/11:04:32	29	20	60000

<전체코드>

```
import chardet
import pymysql
import pandas as pd
import tkinter as tk
from tkinter import ttk

def get_data(code):
    if code=='1': #재고현황
        sqlCmd = "SELECT * FROM stock_view;"
    elif code=='2': #매출현황
        sqlCmd = "SELECT ordertime, s.pnum, pname, qty, totalprice FROM
salesSlip s, product p WHERE s.pnum = p.pnum;"
    elif code=='3': #재고주문서 - 4개 테이블 join
        sqlCmd = "SELECT stk.pnum, p.pname, w.qty, w.wprice*w.qty, spl.sname,
spl.phone FROM stock stk, product p, wholesale w, supplier spl WHERE stk.pnum
= p.pnum AND stk.pnum = w.pnum AND p.snum = spl.snum AND stk.qty_current <
stk.qty_reorder;"
    elif code=='4': #재고현황
        sqlCmd = "SELECT pnum FROM stock;"
    else:
        print("ERROR")

    cursor.execute(sqlCmd)
    rows = cursor.fetchall()
    df = pd.DataFrame(rows)
    data_list = df.values.tolist()
```

```

#print(df)
Make_newWindow(code, data_list)

class Make_newWindow():
    def __init__(self,code, *data):
        if code == '1':
            self.message = "재고현황"
            win_size="700x600"
        elif code == '2':
            self.message = "매출현황"
            win_size="700x600"
        elif code == '3':
            self.message = "재고주문서"
            win_size="700x600"
        elif code == '4':
            self.message = "재고입고"
            win_size="300x400"
        elif code == '5':
            self.message = "신상품입고"
            win_size="300x400"
        #print(message)
        self.newWin = tk.Tk()
        self.newWin.title(self.message)
        self.newWin.geometry(win_size)
        self.newWin.resizable(True,True)
        if code in('1','2','3'): self.make_table(code, data[0])
        elif code == '4': self.get_stock(data[0])
        elif code == '5': self.get_new()
        self.newWin.mainloop()

    def make_table(self, code, treeData):
        ttk.Label(self.newWin, text=self.message).grid(row=0,column=0)

        #self.tree.column("#0", width=10, stretch=True, anchor="w")
        if code == '1':
            treeHeader = ['상품번호', '상품명', '재고수량', '기본수량']
            treeWidth = [30, 20, 10, 10]
            treeHeight = 0.95
        elif code == '2':
            treeHeader = ['주문시각', '상품번호', '상품명', '수량', '가격']
            treeWidth = [40, 10, 20, 10, 12]
            treeHeight = 0.95
        elif code == '3':
            treeHeader = ['상품번호', '상품명', '주문수량', '도매가격',
'공급자이름', '공급자전화번호']
            treeWidth = [10,20,10,12,15,30]
            treeHeight = 0.85
            df=pd.DataFrame(treeData,columns=treeHeader)

```

```

        tk.Button(self.newWin, text="내보내기",
command=lambda:orderSheet_out(df), state=tk.NORMAL).place(relx=0.45,
rely=0.90)

        self.tree = tk.ttk.Treeview(self.newWin, columns=treeHeader,
displaycolumns=treeHeader)
        self.tree.place(relx=0.02, rely=0.03, relwidth=0.96,
relheight=treeHeight)

        self.tree.column("#0", width=2, stretch=True, anchor="w")
        for iCount in range(len(treeHeader)):
            strHdr = treeHeader[iCount]
            self.tree.heading(treeHeader[iCount], text=strHdr.title(),
anchor="w")
            self.tree.column(treeHeader[iCount], width=treeWidth[iCount],
stretch=True, anchor="w")

        for iCount in range(len(treeData)):
            self.tree.insert("", "end", text=iCount + 1,
values=treeData[iCount])

        self.newWin.columnconfigure(4, weight=1)
        self.newWin.rowconfigure(2, weight=1)
        #sb_y = ttk.Scrollbar(self.newWin, orient="vertical",
command=self.tree.yview).place(relx=0.93, rely=0.05, relheight=0.85,
relwidth=0.05)
        #self.tree.configure(yscrollcommand=sb_y.set)

    def get_stock(self, data):
        ttk.Label(self.newWin, text=self.message).grid(row=0, column=0)
        # combobox
        ttk.Label(self.newWin, text="상품번호").grid(row=1, column=0)
        self.combo = tk.ttk.Combobox(self.newWin)
        self.combo['values']=tuple(data)
        self.combo.current(0)
        self.combo.grid(column=1, row=1)
        #text
        ttk.Label(self.newWin, text="입고수량").grid(row=2, column=0)
        self.text = tk.Text(self.newWin, height=1, width=3)
        self.text.grid(column=1, row=2)
        #button
        tk.Button(self.newWin, text="확인",
command=self.cmbtxt_check).grid(column=1, row=3)

    def cmbtxt_check(self):
        pnum = int(self.combo.get())
        qty = int(self.text.get("1.0", "end"))
        #print(pnum)

```

```

        #print(qty)
        #SQL
        sqlCmd = "UPDATE stock SET qty_current = qty_current+%d WHERE pnum
= %d;"

        try:
            cursor.execute(sqlCmd %(qty, pnum))
            con.commit()
        except: ttk.Label(self.newWin, text="ERROR").grid(row=4, column=1)
        else: ttk.Label(self.newWin, text="정상적으로
처리되었습니다.").grid(row=4, column=1)

    def get_new(self):
        ttk.Label(self.newWin, text="신상품 추가").grid(row=0, column=0)
        self.pnum = tk.Entry(self.newWin)
        self.pnum.grid(column=0, row=2)
        self.pnum.insert(0, "상품번호")
        self.pname = tk.Entry(self.newWin)
        self.pname.grid(column=0, row=3)
        self.pname.insert(0, "상품명")
        self.price = tk.Entry(self.newWin)
        self.price.grid(column=0, row=4)
        self.price.insert(0, "가격")
        self.qty = tk.Entry(self.newWin)
        self.qty.grid(column=0, row=5)
        self.qty.insert(0, "입고수량")
        self.dqty = tk.Entry(self.newWin)
        self.dqty.grid(column=0, row=6)
        self.dqty.insert(0, "기본수량설정")
        self.wprice = tk.Entry(self.newWin)
        self.wprice.grid(column=0, row=7)
        self.wprice.insert(0, "도매가격")
        self.wqty = tk.Entry(self.newWin)
        self.wqty.grid(column=0, row=8)
        self.wqty.insert(0, "도매수량")
        self.snum = tk.Entry(self.newWin)
        self.snum.grid(column=0, row=9)
        self.snum.insert(0, "공급자번호")
        tk.Button(self.newWin, text="확인",
command=self.splr_check).grid(column=0, row=10)

    def splr_check(self):
        snum = int(self.snum.get())
        #print(self.snum)
        sqlCmd = "SELECT * FROM supplier WHERE snum=%d;"
        cursor.execute(sqlCmd % snum)
        rows = cursor.fetchall()
        #print(rows)

```



```

        if rows == []:
            self.sname = tk.Entry(self.newWin)
            self.sname.grid(column=0, row=11)
            self.sname.insert(0, "공급자이름")
            self.sphone = tk.Entry(self.newWin)
            self.sphone.grid(column=0, row=12)
            self.sphone.insert(0, "공급자전화번호")
            tk.Button(self.newWin, text="확인",
command=self.add_splr).grid(column=0, row=13)
        else : self.add_new()

    def add_splr(self):
        snum=int(self.snum.get())
        sname=self.sname.get()
        sphone= str(self.sphone.get())
        sqlCmd = "set identity_insert supplier on; INSERT INTO supplier
([snum],[sname],[phone]) VALUES (%d, '%s', '%s'); set identity_insert supplier
off;"

        try:
            print(chardet.detect(sname.encode()))
            cursor.execute(sqlCmd % (snum, sname, sphone))
            con.commit()
        except: ttk.Label(self.newWin, text="ERROR").grid(row=0, column=14)
        else: self.add_new()
        #print(type(snum), type(sname), type(sphone))
        #cursor.execute(sqlCmd % (snum, sname, sphone))
        #con.commit()
        #self.add_new()

    def add_new(self):
        pnum=int(self.pnum.get())
        pname = self.pname.get()
        price = int(self.price.get())
        qty = int(self.qty.get())
        dqty = int(self.dqty.get())
        wprice = int(self.wprice.get())
        wqty = int(self.wqty.get())
        snum = int(self.snum.get())

        sqlCmd1 = "set identity_insert product on; INSERT INTO product
([pnum], [pname],[price],[snum]) VALUES (%d, '%s', %d, %d); set
identity_insert product off;"
        sqlCmd2 = "INSERT INTO stock VALUES (%d, %d, %d);"
        sqlCmd3 = "INSERT INTO wholesale VALUES (%d, %d, %d);"
        try:
            print(pnum, pname, price, snum)
            #print(chardet.detect(pname.encode()))

```

```

        cursor.execute(sqlCmd1 % (pnum, pname, price, snum))
        con.commit()
        cursor.execute(sqlCmd2 % (pnum, qty, dqty))
        con.commit()
        cursor.execute(sqlCmd3 % (pnum, wprice, wqty))
        con.commit()
    except:
        ttk.Label(self.newWin, text="ERROR").grid(row=1, column=1)
    else:
        ttk.Label(self.newWin, text="정상 처리 되었습니다.").grid(row=0,
column=15)

def orderSheet_out(df):
    #print(df)
    df.to_csv('./ordersheet.csv',index=False, encoding='euc-kr')

con = pymssql.connect('DESKTOP-
P5PRM6N','user','0000','ESQL_project',charset='cp949' ) #mssql 접속 EUC-KR
cursor = con.cursor()

window=tk.Tk()

window.title("숙명 매점(관리자용)")
window.geometry("305x485+100+100")
window.resizable(True, True)

tk.Button(window, text="재고현황", overrelief="solid", width=20, height=10,
command=lambda: get_data('1')).grid(row=0,column=0)
tk.Button(window, text="매출현황", overrelief="solid", width=20, height=10,
command=lambda: get_data('2')).grid(row=0,column=1)
tk.Button(window, text="재고주문서", overrelief="solid", width=20, height=10,
command=lambda: get_data('3')).grid(row=1,column=0)
tk.Button(window, text="재고입고", overrelief="solid", width=20, height=10,
command=lambda: get_data('4')).grid(row=1,column=1)
tk.Button(window, text="신상품입고", overrelief="solid", width=42, height=10,
command=lambda: Make_newWindow('5')).grid(row=2,column=0,columnspan=2)

window.mainloop()
con.close()

```

<진행일지>

## 언어 & 주제 정하기

매점관리 프로그램을 만들기로 정하였다. 먼저 웹에서는 데이터베이스에서 데이터를 받아 메뉴를 만들고 주문을 받는 것을 구현한다. 고급언어는 매점 관리자용 프로그램으로, 매출전표와 재고 공급자 등을 확인할 수 있다. 또한 재고나 신상품 입고 기능도 프로그램으로 구현할 수 있을 것 같다. 웹언어는 많이 다뤄보지 않아서 잘 모르겠지만 php나 javascript중에서 정할 것 같고 고급언어로는 python을 사용할 예정이다. python으로는 tkinter 모듈을 사용해 단순한 GUI를 구현할 것이다.

5/14

## 데이터 작성

상품(상품번호, 상품명, 가격, 공급자번호)  
product(pnum,pname,price,snum)

공급자(공급자번호, 공급자이름, 전화번호)  
supplier(snum,sname,phone)

물품재고리스트(상품번호,현재재고수량, 재주문 재고수량)  
stock(pnum,qty\_current,qty\_reorder)

매출전표(상품번호, 수량, 총가격) -> 웹에서 주문 들어오면 상품 수량 업데이트  
salesSlip(pnum,qty,totalprice)

도매가격(상품번호, 도매가격, 도매주문수량)  
wholesale(pnum,wprice,qty)

부 -> 재고주문서(상품번호, 상품명, 주문수량, 도매가격, 공급자이름, 공급자전화번호)

데이터베이스 스키마를 간략히 메모장에 작성하였다. mssql에서 DDL언어로 테이블과 인스턴스를 구현하였다. 나중에 아마 수정이 필요할 것 같지만 일단 매출전표 테이블인 salesSlip을 제외하고는 데이터를 모두 채워두었다.

5/17

Pymssql 모듈을 설치하고 조교님이 올려 주신 자료를 참고해 파이썬에서 데이터를 불러와보았다. 서버 이름 때문에 오류가 많이 났지만 해결했다.

5/18

파이썬에서 메인 페이지를 만들었다. Tkinter를 사용해 간단한 GUI를 구현할 것이다. 파이썬에서는 매점 관리자용 프로그램을 구현할 것이라, 재고현황, 매출현황, 재고주문서, 재고입고, 신상품입고 다섯가지의 버튼을 우선 만들었다. 먼저 메인 윈도우를 만들고 ttk.Button구문을 사용해 버튼을 추가하였다. 아직 연결은 하지 않았다. 재고현황, 매출현황, 재고주문서 버튼을 누르면 테이블이 조회되게 select구문을 쿼리로 날릴 것이다. 재고 입고는 만들어 놓은 stock테이블에 현재 재고를

추가하는 update set 구문을 사용할 것 같고, 신상품입고는 새 튜플을 추가해야되니까 insert into 구문을 사용하게 될 것이다.

5/19

파이썬에서 버튼을 누르면 실행되는 함수 get\_data를 만들었다. 버튼마다 가지는 코드에 따라 sql 쿼리가 달라지게 해 데이터를 받는 함수이다. 재고현황, 매출현황, 재고주문서 3가지 버튼을 누르면 각각의 select 쿼리가 실행된다. 새 창을 여는 클래스 Make\_newWindow도 만들었다. Get\_data 함수에서 데이터를 받아 리스트형태로 바꿔 Make\_newWindow로 보내 tk.treeview를 사용해 테이블을 작성할 것이다. 오늘은 데이터를 받아 클래스로 보내는 것까지 진행하였다.

5/20

전에 재고주문서를 stock\_view로 작성해두고 그걸 get\_data에서 select 하게 했었는데, HLL에서 4개의 테이블 조인을 포함해야 할 것 같아서 뷰 대신 직접 조인하는 select구문이 파이썬에서 실행되게 하였다. Make\_newWindow안에 있는 make\_table함수에서 각각의 테이블이 만들어진다. if 문으로 각 버튼에 맞는 메시지가 조절된다.

5/21

Php와 sqlsrv를 설치했다. Apache도 설치했다. php.ini와 httpd.conf를 수정해 php와 apache를 연결하는 것은 했지만 데이터베이스와 연동이 되지 않아 고생했다. Vscode를 설치하니까 확실히 편리했다. php파일을 다시 수정하고 데이터베이스와 연동을 했지만 이번엔 한글이 깨졌다. 한참동안 시도한 결과 "CharacterSet" => "UTF-8"을 추가하고 한글이 나왔다. 테이블을 만들려면 html이 있어야 해서 또 html 확장을 설치하고 테이블을 만들었다. 상품 주문을 위한 textbox를 넣는 것은 생각보다 간단했다. 데이터를 처음에는 post로 받으려고 했는데 자꾸 오류가 나서 get으로 받는 것으로 변경했다. 상품명과 수량을 입력 받아 상품명으로 select해 key인 pnum을 먼저 찾고 오류사항 처리를 시작했다. 주문수량이 0일 때, 상품번호가 없을 때(상품이름을 잘못 입력했을 때), 재고수량이 부족할 때 등을 염두에 두고 코드를 작성하였다. 올바른 입력일 시 상품번호에 맞는 pnum을 이용해 재고 테이블인 stock에 입력 받은 만큼의 qty를 빼는 update연산을 실시했다. 연산이 정상적으로 시행되면 주문이 정상적으로 완료된 것으로 하고 현재 날짜시간, 상품번호, 구매수량, 총가격을 매출전표인 salesSlip 테이블에 Insert한다. salesSlip테이블은 처음에 작성한 스키마에서 약간 수정해 현재날짜/시간과 주문한 상품번호를 key로 가지게 하였다. 총 가격은 앞서 진행한 select구문에서 상품의 price를 받아 구매수량과 곱해 얻었다. 각종 예외 알림 alert와 echo등을 한 후 주문내역기록을 뜨게 하였다. 상품을 주문하면 재고수량이 줄어드는 것을 확인하고 웹 페이지 구현은 마무리했다.

그 다음 해야 할 것은 파이썬이었다. 4번 버튼인 재고입고 버튼의 기능을 만드는 차례였다. 입고할 재고의 상품번호를 입력하는 방법은 combobox가 좋을 것 같아서 4번버튼도 get\_data함수에서 pnum리스트를 받은 뒤 Make\_newWindow 클래스로 넘겨받아 get\_stock함수가 호출되게 하였

다. 콤보박스과 텍스트박스를 만들고 확인버튼을 만들었다. 오늘은 여기까지.

그리고 만들어진 웹페이지에서 재고와 매출전표가 업데이트 되는 것은 파이썬에서도 확인 가능했다.

5/22

어제 콤보박스과 텍스트에서 입력 받은 pnum, qty로 재고수량이 업데이트 되게 하였다. Update set 구문을 적어 재고 테이블에서 현재재고인 qty\_current에 입력 받은 qty가 더해지게 하였다. 웹 페이지와는 반대되는 연산이다. 파이썬 프로그램에서 재고를 추가해주면 웹 페이지에도 재고가 업데이트 되어 주문할 수 있게 되었다. 이제 마지막 신상품 추가 기능이 문제였다. 신상품을 추가 하면 상품 테이블인 product테이블에 추가가 되어야 한다. 그 뿐만 아니라 연관된 테이블인 stock(재고), wholesale(도매), supplier(공급자) 테이블에도 모두 새로운 정보가 들어가야 한다. 그래서 고민하다가 필요한 모든 정보를 결국 입력을 받기로 하였다. 그런데 또 문제가 공급자는 새로운 공급자일수도 있고, 기존의 공급자일수도 있다는 것이었다. 따라서 먼저 공급자번호를 받아서 앞선 여부를 체크하고, 새로운 공급자이면 나머지 공급자이름, 전화번호 정보를 받고 아니면 그대로 진행되게 하였다. 우려했던 여러 insert 구문들은 잘 실행되는듯 보였다. 하지만 삽입한 데이터에서만 또 한글이 깨지기 시작했다. 이 문제로 네다섯시간을 고민했다. 그리고 아직 진행중이다. - > 생각해보니까 꼭 한글로 insert할 필요는 없는 것 같다.

5/23

보고서를 쓰다가 갑자기 재고주문서를 내보내는 기능을 넣어보고 싶어서 추가해보았다. 데이터 프레임을 내보내는 게 생각보다 더 간단하여서 금방 했다.