Phase2 대비 ERD 와 Relational schema 가 일부 수정되었습니다. 로그인을 가정하고 Customer 와 Manager 엔티티에 ID 속성이 있었으나 customer_id 와 manager_id 가 그 역할을 대체할 수 있다고 생각하여 두 엔티티에서 ID 속성을 삭제하였습니다. 수정된 사항이 포함된 DDL 및 Insert 문은 각각 Team5-Phase2-1-modified.sql, Team5-Phase2-2-modified.sql 로 첨부하였습니다.

Eclipse IDE 를 사용하였고 Java 11, ojdbc10.jar 를 사용했습니다. Database 접속을 위한 USER_NAME 및 USER_PASSWD 는 각각 teamproject, comp322 입니다.

Main.java 에서 실행을 하면

```
0. Exit
1. Query
2. DML
Select Type:
```

위와 같은 메뉴가 나올 것이고 0 은 종료, 1 은 쿼리문 출력을 위한 옵션 선택, 2 는 DML 출력을 위한 옵션 선택이 출력됩니다.

1 입력 시 Phase2 의 쿼리 번호를 선택하도록 아래와 같이 출력됩니다.

```
Query: 2, 3, 4, 6, 7, 9, 12, 13, 17, 18, 19, 20
Select Query Type:
```

쿼리 번호를 입력할 시 각 쿼리별 입력받을 attribute 의 이름이 출력되고 입력받습니다.

쿼리 설명

Phase2 제출물 중 Q7, Q9, Q18, Q20 쿼리문을 수정하였습니다.

Q2: payment_type 을 입력받고 customer 의 이름과 휴대폰 번호를 출력합니다. "Cash" 또는 "Credit Card"를 입력받습니다.

Q3: 몇 개 이하의 재료를 사용한 menu_item 의 이름과 재료 개수를 출력합니다. 정수를 입력받습니다.

Q4: 품절 여부에 따른 menu_item 을 주문한 customer 의 customer_id 와 이름을 출력합니다. 품절 여부를 "T" 또는 "F"로 입력받습니다.

Q6: 특정 기간에 게시된 menu 에 등록됐던 menu_item 의 이름을 출력합니다. 게시 시작 날짜와 종료 날짜를 yymmdd 형식으로 입력받습니다.

Q7: menu_item 의 이름을 입력 받고 menu_item 의 이름과 menu_item 의 수량, menu_item 에 필요한 재료의 총 수량을 출력합니다. menu_item 의 이름은 "Spicy Paneer Wrap", "American Veg Burger", "Veg Maharaja Mac", "Green Chilli Aloo Naan", "Pizza Puff" 등이 있습니다.

Q9: menu_item 의 이름을 입력 받고 그 menu_item 을 관리하는 manager 의 이름과 그 menu_item 이 몇 번 주문되었는지의 총 수량을 출력합니다. menu_item 의 이름 종류는 Q7 과 같습니다.
Q12: customer_id 를 입력받고 그 사람이 지금까지 주문한 메뉴 항목의 총 가격을 계산하고, 그 결과를 order 별로 그룹화하여 출력합니다. customer_id 는 "CU" + 숫자 6 개로 구성되어있습니다.
Q13: menu_item 의 category 를 입력받고, 그 카테고리에 속하는 menu_item 의 단일가격 평균보다 총합 가격이 더 큰 payment 의 order_id, customer_id, 그리고 총합 가격을 출력합니다. category 는 "Gourmet Menu", "Breakfast Menu", "Beverages Menu", "Regular Menu", "Desserts Menu", "McCafe Menu", "Condiments Menu"가 있습니다.
Q17: 정수 n 을 입력받아 각 menu_item 중 가장 많이 주문된 상위 n 개를 출력합니다.
Q18: 정수 n 을 입력받아 총 가격이 n 이상인 payment 에서 customer 의 이름, 주문한 menu_item 의 이름 그리고 해당 menu_item 을 관리하는 manager 의 이름을 출력합니다.
Q19: 정수 n 을 입력받아 동일한 카테고리에 속하는 menu_item 을 n 개 이상 관리하는 manager 의 이름과 휴대폰 번호를 이름의 오름차순으로 출력합니다.
Q20: customer_id 를 입력받고 customer 가 주문한 menu_item 의 이름과 해당 menu_item 을 만드는 chef 의 이름을 출력합니다.

쿼리문의 수정사항은 아래와 같습니다.
Q7.
각 menu_item 의 이름과 menu_item 의 수량, menu_item 에 필요한 재료의 총 수량을 출력합니다.
-> 이름이 "Spicy Paneer Wrap"인 menu_item 의 이름과 menu_item 의 수량, menu_item 에 필요한 재료의 총 수량을 출력합니다.

Q9.
각 menu_item 에 대해 그것을 관리하는 manager 의 이름과 그 menu_item 이 몇 번 주문되었는지의 총 수량을 출력합니다
-> 이름이 "Spicy Paneer Wrap"인 menu_item 에 대해 그것을 관리하는 manager 의 이름과 그 menu_item 이 몇 번 주문되었는지의 총 수량을 출력합니다.

Q18.
payment 에서 customer 의 이름, 주문한 menu_item 의 이름 그리고 해당 menu_item 을 관리하는 manager 의 이름을 출력합니다.
-> 총 가격이 100000 이상인 payment 에서 customer 의 이름, 주문한 menu_item 의 이름 그리고 해당 menu_item 을 관리하는 manager 의 이름을 출력합니다.

Q20.
customer 가 주문한 menu_item 의 이름과 해당 menu_item 을 만드는 chef 의 이름을 출력합니다. 결과는 chef 의 이름 순으로 정렬합니다.
-> customer_id 가 "CU000001" 인 customer 가 주문한 menu_item 의 이름과 해당 menu_item 을 만드는 chef 의 이름을 출력합니다. 결과는 chef 의 이름 순으로 정렬합니다.

수정된 쿼리를 포함하여 사용된 모든 쿼리문은 첨부된 using_query.sql 에 있습니다.

메인에서 2 입력시 DML 클래스가 실행됩니다.
DML: Customer, Manager, Chef 중 어떤 엔티티에 대한 DML 을 선택할 것인지 입력받은 후 각각의 정보 입력, 수정, 삭제를 하는 기능을 선택합니다.

1. Customer: Customer 에 대한 정보 입력, 수정, 삭제를 수행합니다.
    1. Customer 정보 입력 : Customer 로부터 이름과 비밀번호 그리고 휴대폰 번호를 입력받는다. 신규 Customer id 는 기존에 순차적으로 저장되어 있는 Customer_id 에서 가장 큰 id 번호에 1 을 더해 생성한다. 최종적으로 customer 의 id, 이름, 비밀번호, 휴대폰 번호를 추가하는 insert 를 실행한다.
    2. Customer 정보 수정 : Customer 로부터 id 를 입력 받고, 이름과 비밀번호 그리고 휴대폰 번호 중 수정할 항목을 선택하게 한다. customer 가 선택한 수정 항목에 대해 새로운 값을 입력 받게 하고 Update 를 실행한다.
    3. Customer 정보 삭제 : Customer 로부터 id 를 입력 받고, customer id 에 해당하는 정보를 삭제하는 delete 를 실행한다.

2. Manager: Manager 에 대한 정보 입력, 수정, 삭제를 수행합니다.
    1. Manager 정보 입력 : Manager 로부터 이름과 비밀번호 그리고 휴대폰 번호를 입력받는다. 신규 Manager id 는 기존에 순차적으로 저장되어 있는 Manager_id 에서 가장 큰 id 번호에 1 을 더해 생성한다. 최종적으로 manager 의 id, 이름, 비밀번호, 휴대폰 번호를 추가하는 insert 를 실행한다.
    2. Manager 정보 수정 : Manager 로부터 id 를 입력 받고, 이름과 비밀번호 그리고 휴대폰 번호 중 수정할 항목을 선택하게 한다. manager 가 선택한 수정 항목에 대해 새로운 값을 입력 받게 하고 Update 를 실행한다.
    3. Manager 정보 삭제 : Manager 로부터 id 를 입력 받고, manager id 에 해당하는 정보를 삭제하는 delete 를 실행한다.

3. Chef: Chef 에 대한 정보 입력, 수정, 삭제를 수행합니다.

1. Chef 정보 입력 : chef 로부터 이름과 휴대폰 번호를 입력받는다. 신규 Chef id 는 기존에 순차적으로 저장되어 있는 Chef_id 에서 가장 큰 id 번호에 1 을 더해 생성한다. 최종적으로 chef 의 id, 이름, 휴대폰 번호를 추가하는 insert 를 실행한다.

2. Chef 정보 수정 : chef 로부터 id 를 입력 받고, 이름과 휴대폰번호 중 수정할 항목을 선택하게 한다. chef 가 선택한 수정 항목에 대해 새로운 값을 입력 받게 하고 Update 를 실행한다.

3. Chef 정보 삭제 : Chef 로 부터 id 를 입력 받고, Chef id 에 해당하는 정보를 삭제하는 delete 를 실행한다.

아래는 입출력 예제입니다.

```
Success!
Connected.

0. Exit
1. Query
2. DML
Select Type: 2

1. Customer
2. Manager
3. Chef
Select Entity: 1

1. INSERT
2. UPDATE
3. DELETE
Select DML type: 1

Name: Customer1
Password: comp322
Phone_number: 01012345678
CU000201 customer insert
0. Exit
1. Query
2. DML
Select Type: 2

1. Customer
2. Manager
3. Chef
Select Entity: 1

1. INSERT
2. UPDATE
3. DELETE
Select DML type: 2

Customer id: CU000201
```

```
Attribute: Name Password Phone_number
Select Attribute: Name
Name: Customer 201
CU000201 customer update
Name: Customer 201
0. Exit
1. Query
2. DML
Select Type: 2

1. Customer
2. Manager
3. Chef
Select Entity: 1

1. INSERT
2. UPDATE
3. DELETE
Select DML type: 3

Customer id: CU000201
CU000201 customer delete
0. Exit
1. Query
2. DML
Select Type: 0

Exit
```