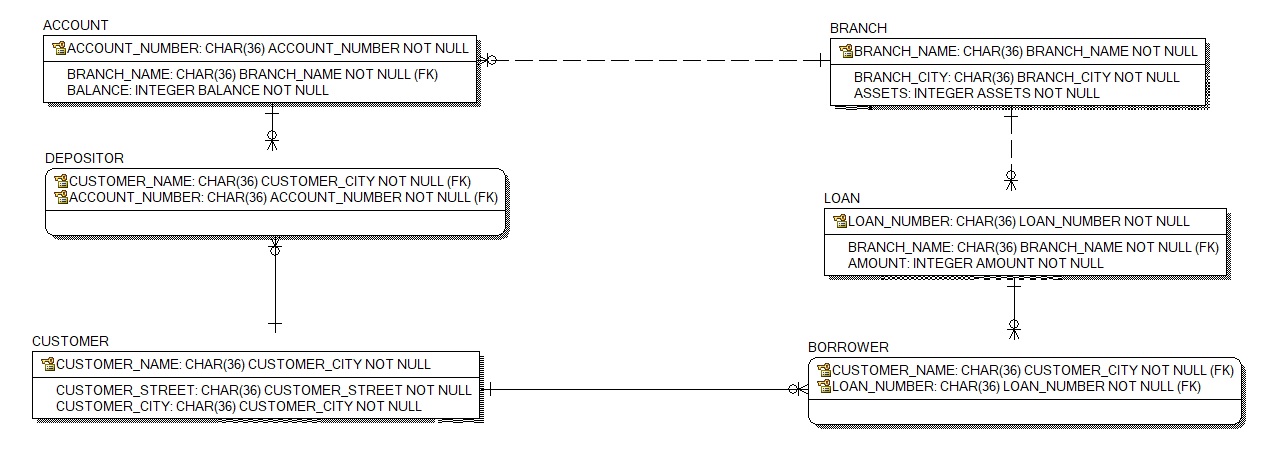
**2019학년도 1학기 데이터베이스 Project 2 보고서**

**담당교수: 서강대학교 컴퓨터공학과 박석**

**1. Physical Mode**

프로젝트 2의 명세서에 안내된 ER 다이어그램을 바탕으로 relation schema를 완성하고 각 relation의 attribute 별로 무결성 제약과 type을 지정했다. [그림 1]은 설계한 Physical Mode의 완성본이다.

[그림 1]



**◎ Domain · Integrity Rule of Attributes**

|  |  |  |  |
| --- | --- | --- | --- |
| **Domain Name** | **Data Type** | **Null 허용** | **설명** |
| **ACCOUNT\_NUMBER** | CHAR(36) | Not Null | 개설된 계좌의 고유번호를 저장하기 위한 ACCOUNT\_NUMBER attribute의 도메인이며, 계좌의 고유번호는 앞자리에 오는 character인 ‘-’와 숫자의 나열로 되어 있어서 이를 충족하고자 36자리의 CHAR를 data type으로 가진다. 계좌 고유번호에 관한 데이터는 ACCOUNT relation의 PK이므로 Not Null로 설정한다. |
| **BRANCH\_NAME** | CHAR(36) | Not Null | 지점의 이름을 저장하기 위한 BRANCH\_NAME attribute의 도메인이며, 지점명은 character로 이루어진 string이므로 36자리의 CHAR을 data type으로 지정했다. 지점명에 관한 데이터는 BRANCH relation의 PK이므로 Not Null로 설정한다. |
| **BRANCH\_CITY** | CHAR(36) | Not Null | 지점이 소재하고 있는 도시의 이름을 저장하기 위한 BRANCH\_CITY attribute의 도메인이며, 지점이 소재하는 도시명은 character로 이루어진 string이므로 36자리의 CHAR을 data type으로 지정했다. 모든 지점에 관해 반드시 소재하고 있는 도시의 이름을 갖고 있다고 가정하여 Not Null로 설정한다. |
| **ASSETS** | INTEGER | Not Null | 지점의 자산의 금액을 저장하기 위한 ASSETS attribute의 도메인이며, 지점의 자산 금액은 정수로 이루어져 있으므로 10자리의 INTEGER를 data type으로 지정했다. 모든 지점은 최소한 자산 금액에 관한 데이터를 가지고 있으며, 실제로 갖고 있지 않다 하더라도 최소한 자산을 0으로 갖는다고 가정하여 Not Null로 설정한다. |
| **CUSTOER\_NAME** | CHAR(36) | Not Null | 손님의 이름을 저장하기 위한 CUSTOMER\_NAME attribute의 도메인이며, 손님의 이름은 character로 이루어진 string이므로 36자리의 CHAR을 data type으로 지정했다. 손님의 이름에 관한 데이터는 CUSTOMER relation의 PK이므로 Not Null로 설정한다. |
| **CUSTOMER\_STREET** | CHAR(36) | Not Null | 손님이 살고 있는 거리의 이름을 저장하기 위한 CUSTOMER\_STREET attribute의 도메인이며, 손님이 살고 있는 거리명은 character로 이루어진 string이므로 36자리의 CHAR을 data type으로 지정했다. 모든 손님에 관해 반드시 살고 있는 거리가 존재한다고 가정하여 Not Null로 설정한다. |
| **CUSTOMER\_CITY** | CHAR(36) | Not Null | 손님이 살고 있는 도시의 이름을 저장하기 위한 CUSTOMER\_CITY attribute의 도메인이며, 손님이 살고 있는 도시명은 character로 이루어진 string이므로 36자리의 CHAR을 data type으로 지정했다. 모든 손님에 관해 반드시 살고 있는 도시가 존재한다고 가정하여 Not Null로 설정한다. |
| **LOAN\_NUMBER** | CHAR(36) | Not Null | 대출 상품의 고유번호 저장하기 위한 LOAN\_NUMBER attribute의 도메인이며, 대출 상품의 고유번호는 앞자리에 오는 character인 ‘-’와 숫자의 나열로 되어 있어서 이를 충족하고자 36자리의 CHAR를 data type으로 가진다. 대출 상품에 관한 데이터는 LOAN relation의 PK이므로 Not Null로 설정한다. |
| **AMOUNT** | INTEGER | Not Null | 대출 상품에서 손님에게 대출해주는 금액을 저장하기 위한 AMOUNT attribute의 도메인이며, 대출 상품의 금액은 정수로 이루어져 있으므로 10자리의 INTEGER를 data type으로 가진다. 모든 대출 상품은 최소한 대출해주는 금액에 관한 데이터를 가지고 있으며, 실제로 갖고 있지 않다 하더라도 최소한 자산을 0으로 갖는다고 가정하여 Not Null로 설정한다. |

[표 1 - 1]

**2. SQL 질의와 실행 결과**

ORACLE 데이터베이스 서버에 접속해 다음과 같은 SQL 질의를 수행하고 결과 화면을 각 SQL 질의마다 붙이고 사용한 SQL 질의를 텍스트로 적었다.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1) | 프로젝트 2 명세서에 주어진 데이터를 relation에 입력하시오.     |  | | --- | | INSERT ALL  INTO BRANCH (BRANCH\_NAME, BRANCH\_CITY, ASSETS)  VALUES ('Brighton', 'Brooklyn', 7100000)  INTO BRANCH (BRANCH\_NAME, BRANCH\_CITY, ASSETS)  VALUES ('Downtown', 'Brooklyn', 9000000)  INTO BRANCH (BRANCH\_NAME, BRANCH\_CITY, ASSETS)  VALUES ('Mianus', 'Horseneck', 400000)  INTO BRANCH (BRANCH\_NAME, BRANCH\_CITY, ASSETS)  VALUES ('North Town', 'Rye', 3700000)  INTO BRANCH (BRANCH\_NAME, BRANCH\_CITY, ASSETS)  VALUES ('Perryridge', 'Horseneck', 1700000)  INTO BRANCH (BRANCH\_NAME, BRANCH\_CITY, ASSETS)  VALUES ('Pownal', 'Benninghton', 300000)  INTO BRANCH (BRANCH\_NAME, BRANCH\_CITY, ASSETS)  VALUES ('Redwood', 'Palo Alto', 2100000)  INTO BRANCH (BRANCH\_NAME, BRANCH\_CITY, ASSETS)  VALUES ('Round Hill', 'Horseneck', 8000000)  SELECT \* FROM DUAL; |      |  | | --- | | INSERT ALL  INTO CUSTOMER (CUSTOMER\_NAME, CUSTOMER\_STREET, CUSTOMER\_CITY)  VALUES ('Adams', 'Spring', 'Pittsfield')  INTO CUSTOMER (CUSTOMER\_NAME, CUSTOMER\_STREET, CUSTOMER\_CITY)  VALUES ('Brooks', 'Senator', 'Brooklyn')  INTO CUSTOMER (CUSTOMER\_NAME, CUSTOMER\_STREET, CUSTOMER\_CITY)  VALUES ('Curry', 'North', 'Rye')  INTO CUSTOMER (CUSTOMER\_NAME, CUSTOMER\_STREET, CUSTOMER\_CITY)  VALUES ('Glenn', 'Sand Hill', 'Woodside')  INTO CUSTOMER (CUSTOMER\_NAME, CUSTOMER\_STREET, CUSTOMER\_CITY)  VALUES ('Green', 'Walnut', 'Stamford')  INTO CUSTOMER (CUSTOMER\_NAME, CUSTOMER\_STREET, CUSTOMER\_CITY)  VALUES ('Hayes', 'Main', 'Harrison')  INTO CUSTOMER (CUSTOMER\_NAME, CUSTOMER\_STREET, CUSTOMER\_CITY)  VALUES ('Johnson', 'Alma', 'Palo Alto')  INTO CUSTOMER (CUSTOMER\_NAME, CUSTOMER\_STREET, CUSTOMER\_CITY)  VALUES ('Jones', 'Main', 'Harrison')  INTO CUSTOMER (CUSTOMER\_NAME, CUSTOMER\_STREET, CUSTOMER\_CITY)  VALUES ('Lindsay', 'Park', 'Pittsfield')  INTO CUSTOMER (CUSTOMER\_NAME, CUSTOMER\_STREET, CUSTOMER\_CITY)  VALUES ('Smith', 'North', 'Rye')  INTO CUSTOMER (CUSTOMER\_NAME, CUSTOMER\_STREET, CUSTOMER\_CITY)  VALUES ('Turner', 'Putnam', 'Stamford')  INTO CUSTOMER (CUSTOMER\_NAME, CUSTOMER\_STREET, CUSTOMER\_CITY)  VALUES ('Williams', 'Nassau', 'Princeton')  SELECT \* FROM DUAL; |      |  | | --- | | INSERT ALL  INTO ACCOUNT (ACCOUNT\_NUMBER, BRANCH\_NAME, BALANCE)  VALUES ('A-101', 'Downtown', 500)  INTO ACCOUNT (ACCOUNT\_NUMBER, BRANCH\_NAME, BALANCE)  VALUES ('A-102', 'Perryridge', 400)  INTO ACCOUNT (ACCOUNT\_NUMBER, BRANCH\_NAME, BALANCE)  VALUES ('A-201', 'Brighton', 900)  INTO ACCOUNT (ACCOUNT\_NUMBER, BRANCH\_NAME, BALANCE)  VALUES ('A-215', 'Mianus', 700)  INTO ACCOUNT (ACCOUNT\_NUMBER, BRANCH\_NAME, BALANCE)  VALUES ('A-217', 'Brighton', 750)  INTO ACCOUNT (ACCOUNT\_NUMBER, BRANCH\_NAME, BALANCE)  VALUES ('A-222', 'Redwood', 700)  INTO ACCOUNT (ACCOUNT\_NUMBER, BRANCH\_NAME, BALANCE)  VALUES ('A-305', 'Round Hill', 350)  SELECT \* FROM DUAL; |      |  | | --- | | INSERT ALL  INTO LOAN (LOAN\_NUMBER, BRANCH\_NAME, AMOUNT)  VALUES ('L-11', 'Round Hill', 900)  INTO LOAN (LOAN\_NUMBER, BRANCH\_NAME, AMOUNT)  VALUES ('L-14', 'Downtown', 1500)  INTO LOAN (LOAN\_NUMBER, BRANCH\_NAME, AMOUNT)  VALUES ('L-15', 'Perryridge', 1500)  INTO LOAN (LOAN\_NUMBER, BRANCH\_NAME, AMOUNT)  VALUES ('L-16', 'Perryridge', 1300)  INTO LOAN (LOAN\_NUMBER, BRANCH\_NAME, AMOUNT)  VALUES ('L-17', 'Downtown', 1000)  INTO LOAN (LOAN\_NUMBER, BRANCH\_NAME, AMOUNT)  VALUES ('L-23', 'Redwood', 2000)  INTO LOAN (LOAN\_NUMBER, BRANCH\_NAME, AMOUNT)  VALUES ('L-93', 'Mianus', 500)  SELECT \* FROM DUAL; |      |  | | --- | | INSERT ALL  INTO DEPOSITOR (CUSTOMER\_NAME, ACCOUNT\_NUMBER)  VALUES ('Hayes', 'A-102')  INTO DEPOSITOR (CUSTOMER\_NAME, ACCOUNT\_NUMBER)  VALUES ('Johnson', 'A-101')  INTO DEPOSITOR (CUSTOMER\_NAME, ACCOUNT\_NUMBER)  VALUES ('Johnson', 'A-201')  INTO DEPOSITOR (CUSTOMER\_NAME, ACCOUNT\_NUMBER)  VALUES ('Jones', 'A-217')  INTO DEPOSITOR (CUSTOMER\_NAME, ACCOUNT\_NUMBER)  VALUES ('Lindsay', 'A-222')  INTO DEPOSITOR (CUSTOMER\_NAME, ACCOUNT\_NUMBER)  VALUES ('Smith', 'A-215')  INTO DEPOSITOR (CUSTOMER\_NAME, ACCOUNT\_NUMBER)  VALUES ('Turner', 'A-305')  SELECT \* FROM DUAL; |      |  | | --- | | INSERT ALL  INTO BORROWER (CUSTOMER\_NAME, LOAN\_NUMBER)  VALUES ('Adams', 'L-16')  INTO BORROWER (CUSTOMER\_NAME, LOAN\_NUMBER)  VALUES ('Curry', 'L-93')  INTO BORROWER (CUSTOMER\_NAME, LOAN\_NUMBER)  VALUES ('Hayes', 'L-15')  INTO BORROWER (CUSTOMER\_NAME, LOAN\_NUMBER)  VALUES ('Johnson', 'L-14')  INTO BORROWER (CUSTOMER\_NAME, LOAN\_NUMBER)  VALUES ('Jones', 'L-17')  INTO BORROWER (CUSTOMER\_NAME, LOAN\_NUMBER)  VALUES ('Smith', 'L-11')  INTO BORROWER (CUSTOMER\_NAME, LOAN\_NUMBER)  VALUES ('Smith', 'L-23')  INTO BORROWER (CUSTOMER\_NAME, LOAN\_NUMBER)  VALUES ('Williams', 'L-17')  SELECT \* FROM DUAL; | |
| 2) | 중복되지 않은 모든 지점들의 이름을 구하라.     |  | | --- | | SELECT DISTINCT BRANCH\_NAME FROM BRANCH; | |
|  |  |
| 3) | Redwood 지점의 대출을 가진 모든 고객들을 알파벳 역순으로 나열하라.     |  | | --- | | SELECT \* FROM CUSTOMER  WHERE CUSTOMER.CUSTOMER\_NAME IN (SELECT BORROWER.CUSTOMER\_NAME FROM BORROWER  WHERE BORROWER.LOAN\_NUMBER = (SELECT LOAN.LOAN\_NUMBER FROM LOAN  WHERE LOAN.LOAN\_NUMBER = BORROWER.LOAN\_NUMBER  AND LOAN.BRANCH\_NAME = 'Redwood'))  ORDER BY CUSTOMER.CUSTOMER\_NAME DESC; | |
| 4) | 은행에서 대출, 계좌 혹은 둘 다를 가진 모든 고객을 나열하라.     |  | | --- | | SELECT \* FROM CUSTOMER  WHERE CUSTOMER.CUSTOMER\_NAME IN (SELECT DISTINCT DEPOSITOR.CUSTOMER\_NAME FROM ACCOUNT, BRANCH, DEPOSITOR WHERE ACCOUNT.ACCOUNT\_NUMBER = DEPOSITOR.ACCOUNT\_NUMBER  AND BRANCH.BRANCH\_NAME = ACCOUNT.BRANCH\_NAME) OR  CUSTOMER.CUSTOMER\_NAME IN (SELECT DISTINCT BORROWER.CUSTOMER\_NAME FROM LOAN, BRANCH, BORROWER WHERE BORROWER.LOAN\_NUMBER = LOAN.LOAN\_NUMBER  AND BRANCH.BRANCH\_NAME = LOAN.BRANCH\_NAME); | |
| 5) | 은행에 대출과 계좌 모두를 가진 모든 고객을 나열하라.     |  | | --- | | SELECT \* FROM CUSTOMER  WHERE CUSTOMER.CUSTOMER\_NAME IN (SELECT DISTINCT DEPOSITOR.CUSTOMER\_NAME FROM ACCOUNT, BRANCH, DEPOSITOR WHERE ACCOUNT.ACCOUNT\_NUMBER = DEPOSITOR.ACCOUNT\_NUMBER  AND BRANCH.BRANCH\_NAME = ACCOUNT.BRANCH\_NAME) AND  CUSTOMER.CUSTOMER\_NAME IN (SELECT DISTINCT BORROWER.CUSTOMER\_NAME FROM LOAN, BRANCH, BORROWER WHERE BORROWER.LOAN\_NUMBER = LOAN.LOAN\_NUMBER  AND BRANCH.BRANCH\_NAME = LOAN.BRANCH\_NAME); | |
| 6) | 대출 총액이 가장 작은 고객의 이름과 대출 총액을 구하여라.     |  | | --- | | WITH TEMPDATA AS (  SELECT CUSTOMER\_NAME, SUM(AMOUNT) AS TOTMAT FROM BORROWER JOIN LOAN USING(LOAN\_NUMBER)  JOIN BRANCH USING(BRANCH\_NAME) GROUP BY CUSTOMER\_NAME)  SELECT \* FROM TEMPDATA WHERE TOTMAT = (SELECT MIN(TOTMAT) FROM TEMPDATA)  AND CUSTOMER\_NAME IN (SELECT CUSTOMER.CUSTOMER\_NAME FROM CUSTOMER); | |
| 7) | Harrison과 Stamford에 살지 않으면서 계좌에 잔고의 합이 1000 이하 있는 고객의 이름과 고객이 사는 도시를 구하여라.     |  | | --- | | SELECT CUSTOMER\_NAME, CUSTOMER\_CITY  FROM ((SELECT \* FROM (SELECT CUSTOMER\_NAME, SUM(BALANCE) AS BALANCE\_SUM  FROM CUSTOMER JOIN DEPOSITOR USING(CUSTOMER\_NAME) JOIN ACCOUNT USING(ACCOUNT\_NUMBER)  GROUP BY CUSTOMER\_NAME) WHERE BALANCE\_SUM <= 1000) JOIN (SELECT \* FROM CUSTOMER  WHERE CUSTOMER\_CITY NOT IN (SELECT CUSTOMER\_CITY FROM CUSTOMER  WHERE CUSTOMER\_CITY = 'Harrison' OR CUSTOMER\_CITY = 'Stamford')) USING(CUSTOMER\_NAME)); | |
| 8) | 은행에 계좌는 없지만 대출은 가지고 있는 모든 고객들을 나열하라.     |  | | --- | | SELECT \* FROM CUSTOMER  WHERE CUSTOMER.CUSTOMER\_NAME NOT IN (SELECT DISTINCT CUSTOMER\_NAME  FROM ACCOUNT JOIN DEPOSITOR USING(ACCOUNT\_NUMBER) JOIN CUSTOMER USING(CUSTOMER\_NAME)) AND CUSTOMER.CUSTOMER\_NAME IN (SELECT DISTINCT CUSTOMER\_NAME  FROM LOAN JOIN BORROWER USING(LOAN\_NUMBER) JOIN CUSTOMER USING(CUSTOMER\_NAME)); | |
| 9) | Perryridge 지점에서 계좌의 평균 잔고를 구하여라.     |  | | --- | | SELECT AVG(BALANCE) AS BALANCE\_AVERAGE  FROM ACCOUNT JOIN DEPOSITOR USING(ACCOUNT\_NUMBER) JOIN BRANCH USING(BRANCH\_NAME)  WHERE BRANCH\_NAME = 'Perryridge'; | |
| 10) | 각 지점의 대출자들의 수를 구하라.     |  | | --- | | SELECT BRANCH\_NAME, COUNT(CUSTOMER\_NAME) FROM (SELECT DISTINCT BRANCH\_NAME, CUSTOMER\_NAME  FROM BORROWER JOIN LOAN USING(LOAN\_NUMBER) JOIN BRANCH USING(BRANCH\_NAME)) GROUP BY BRANCH\_NAME; | |
| 11) | 평균 잔고가 $500 이상인 지점 이름과 총 잔고를 나열하라.     |  | | --- | | SELECT BRANCH\_NAME, BALANCE\_SUM FROM  (SELECT BRANCH\_NAME, SUM(BALANCE) AS BALANCE\_SUM, AVG(BALANCE) AS BALANCE\_AVERAGE  FROM ACCOUNT JOIN DEPOSITOR USING(ACCOUNT\_NUMBER) JOIN BRANCH USING(BRANCH\_NAME) GROUP BY BRANCH\_NAME) WHERE BALANCE\_AVERAGE >= 500; | |
| 12) | Palo Alto에 살고 최소한 두 개의 계좌를 가진 각각의 고객들의 이름과 잔고의 합을 구하라.     |  | | --- | | SELECT CUSTOMER\_NAME, BALANCE\_SUM FROM (  (SELECT CUSTOMER\_NAME, SUM(BALANCE) AS BALANCE\_SUM, COUNT(ACCOUNT\_NUMBER) AS  ACCOUNT\_COUNTER FROM CUSTOMER JOIN DEPOSITOR USING(CUSTOMER\_NAME) JOIN ACCOUNT USING(ACCOUNT\_NUMBER) GROUP BY CUSTOMER\_NAME) JOIN CUSTOMER USING(CUSTOMER\_NAME)) WHERE CUSTOMER\_CITY = 'Palo Alto' AND ACCOUNT\_COUNTER >= 2; | |
| 14) | 각 도시 별로 가장 높은 대출 총액을 가지고 있는 고객의 이름과 대출 총액을 구하여라. 단, 대출을 가진 고객이 살지 않는 도시는 표시하지 않는다.     |  | | --- | | WITH TEMPDATA AS(SELECT CUSTOMER\_NAME, SUM(AMOUNT) AS LOAN\_AMOUNT\_SUM  FROM BORROWER JOIN LOAN USING(LOAN\_NUMBER) JOIN CUSTOMER USING(CUSTOMER\_NAME)  GROUP BY CUSTOMER\_NAME)  SELECT CUSTOMER\_CITY, CUSTOMER\_NAME, LOAN\_AMOUNT\_SUM FROM TEMPDATA JOIN CUSTOMER USING(CUSTOMER\_NAME) JOIN (SELECT CUSTOMER\_CITY, MAX(LOAN\_AMOUNT\_SUM) AS LOAN\_AMOUNT\_SUM  FROM TEMPDATA JOIN CUSTOMER USING(CUSTOMER\_NAME)  GROUP BY CUSTOMER\_CITY) USING(CUSTOMER\_CITY, LOAN\_AMOUNT\_SUM); | |
| 15) | Horseneck에 있는 지점들의 자산 중 최소값보다 작거나 같은 자산 값을 갖는 모든 지점들의 이름을 나열하라.     |  | | --- | | SELECT BRANCH\_NAME FROM BRANCH WHERE ASSETS <= (SELECT MIN(ASSETS)  FROM BRANCH WHERE BRANCH\_CITY = 'Horseneck'); | |
| 16) | Downtown 지점에서 $1000 이상의 대출 총액을 지닌 고객들을 전부 구하라.     |  | | --- | | WITH TEMPDATA AS  (SELECT CUSTOMER\_NAME, SUM(AMOUNT) AS LOAN\_AMOUNT FROM  CUSTOMER JOIN BORROWER USING(CUSTOMER\_NAME) JOIN LOAN USING(LOAN\_NUMBER) JOIN BRANCH USING(BRANCH\_NAME)  WHERE BRANCH\_NAME = 'Downtown' GROUP BY CUSTOMER\_NAME)  SELECT CUSTOMER\_NAME FROM TEMPDATA WHERE LOAN\_AMOUNT >= 1000; | |
| 17) | 은행에 대출을 가지고 있는 모든 고객들에 대해 그들의 이름과 대출 총액을 구해라.     |  | | --- | | SELECT CUSTOMER\_NAME, SUM(AMOUNT) AS LOAN\_AMOUNT\_SUM  FROM BORROWER JOIN LOAN USING(LOAN\_NUMBER) JOIN CUSTOMER USING(CUSTOMER\_NAME)  GROUP BY CUSTOMER\_NAME; | |
| 18) | Downtown 지점의 모든 대출에 대하여 고객의 이름과 대출 번호, 대출액을 구하라.     |  | | --- | | SELECT CUSTOMER\_NAME, LOAN\_NUMBER, AMOUNT  FROM BORROWER JOIN LOAN USING(LOAN\_NUMBER) JOIN CUSTOMER USING(CUSTOMER\_NAME)  WHERE BRANCH\_NAME = 'Downtown'; | |
| 19) | 이름에 ‘ai’이라는 부분 문자열이 포함된 거리에 살고 있는 모든 고객들의 이름을 구하여라.     |  | | --- | | SELECT DISTINCT CUSTOMER\_NAME FROM CUSTOMER WHERE CUSTOMER\_STREET LIKE '%ai%'; | |
| 20) | 가장 낮은 평균 잔고를 가진 고객의 이름과 총 잔고를 구하라.     |  | | --- | | WITH TEMPDATA AS  (SELECT CUSTOMER\_NAME, SUM(BALANCE) AS BALANCE\_SUM, AVG(BALANCE) AS BALANCE\_AVERAGE  FROM ACCOUNT JOIN DEPOSITOR USING(ACCOUNT\_NUMBER) JOIN CUSTOMER USING(CUSTOMER\_NAME) GROUP BY CUSTOMER\_NAME)  SELECT CUSTOMER\_NAME, BALANCE\_SUM FROM TEMPDATA  WHERE BALANCE\_AVERAGE = (SELECT MIN(BALANCE\_AVERAGE) FROM TEMPDATA); | |
| 21) | 지점 이름과 그 지점에 계좌나 대출 둘 중 하나 이상을 가진 고객 이름으로 구성된 View를 작성하라. (단 View의 이름은 all\_customer이다.)     |  | | --- | | CREATE OR REPLACE VIEW all\_customer AS  SELECT DISTINCT \* FROM(  SELECT DISTINCT BRANCH\_NAME, CUSTOMER\_NAME  FROM DEPOSITOR JOIN ACCOUNT USING(ACCOUNT\_NUMBER) JOIN BRANCH USING(BRANCH\_NAME))  UNION (SELECT DISTINCT BRANCH\_NAME, CUSTOMER\_NAME  FROM BORROWER JOIN LOAN USING(LOAN\_NUMBER) JOIN BRANCH USING(BRANCH\_NAME)); | |
| 22) | 21에서 생성된 View를 이용하여 Downtown 지점의 모든 고객 이름을 나열하라.     |  | | --- | | SELECT DISTINCT CUSTOMER\_NAME FROM ALL\_CUSTOMER WHERE BRANCH\_NAME = 'Downtown'; | |
| 23) | 각 지점마다 잔고의 최댓값을 나열하라.     |  | | --- | | SELECT BRANCH\_NAME, MAX(BALANCE)  FROM ACCOUNT JOIN DEPOSITOR USING (ACCOUNT\_NUMBER) JOIN BRANCH USING (BRANCH\_NAME)  GROUP BY BRANCH\_NAME; | |
| 24) | 모든 지점의 총 계좌 예금의 평균보다 많은 총 계좌 예금을 갖는 모든 지점을 나열하라. (지점에 계좌가 없으면 해당 지점은 평균 계산에서 제외한다.)     |  | | --- | | WITH TEMPDATA AS (  SELECT BRANCH\_NAME, SUM(BALANCE) AS BALANCE\_SUM FROM  DEPOSITOR JOIN ACCOUNT USING (ACCOUNT\_NUMBER) JOIN BRANCH USING (BRANCH\_NAME) GROUP BY BRANCH\_NAME)  SELECT BRANCH\_NAME, BRANCH\_CITY, ASSETS FROM(  SELECT \* FROM TEMPDATA WHERE BALANCE\_SUM >= (SELECT AVG(BALANCE\_SUM) FROM TEMPDATA)  ) JOIN BRANCH USING(BRANCH\_NAME); | |
| 25) | 고객들의 평균 대출 총액보다 많은 대출 총액을 가지고 있는 고객의 이름과 대출 총액을 구하라.     |  | | --- | | WITH TEMPDATA AS (  SELECT CUSTOMER\_NAME, SUM(AMOUNT) AS LOAN\_AMOUNT FROM  CUSTOMER JOIN BORROWER USING(CUSTOMER\_NAME) JOIN LOAN USING(LOAN\_NUMBER) GROUP BY CUSTOMER\_NAME)  SELECT CUSTOMER\_NAME, LOAN\_AMOUNT FROM TEMPDATA  WHERE LOAN\_AMOUNT >= (SELECT AVG(LOAN\_AMOUNT) FROM TEMPDATA); | |