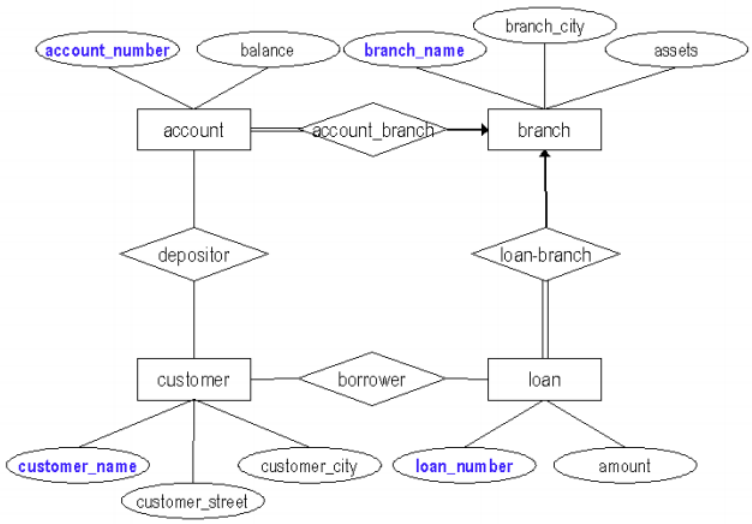
2020 Database System Project #2

[Constructing SQL]

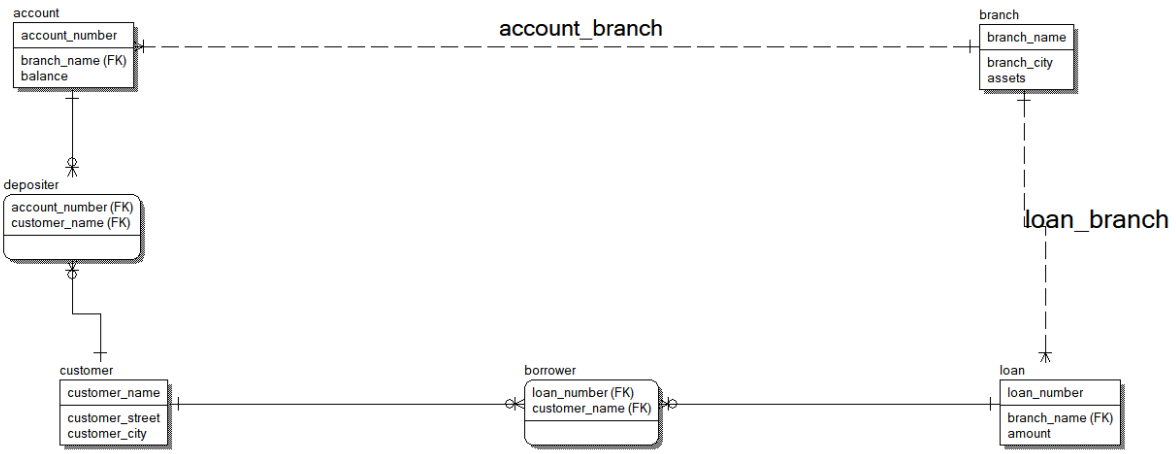
20151741 구본준

1. ER-win을 통한 Modeling

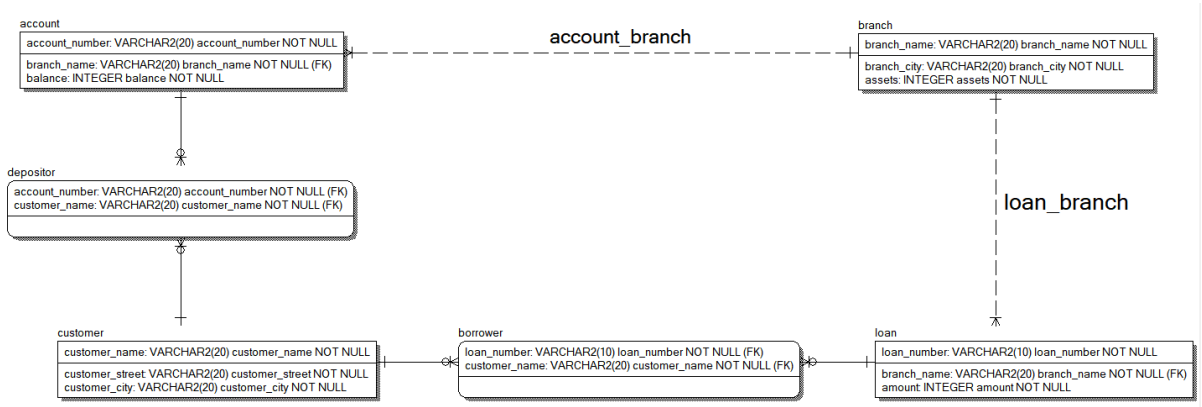
1) ERD (문제에 제시됨)



2) Logical Mode



3) Physical Mode



4) Entity & Relation

Entity: account, branch, customer, loan

Relation: depositor, borrower, account_branch, loan_branch

4.1) account

Account는 account_number, branch_name, balance를 attribute로 가지며, primary key는 account_number이다. 무결성 조건을 만족시키기 위해 앞으로 기술되는 모든 attribute에서 NULL 값을 허용하지 않는다. Primary key가 Not NULL이고 Foreign key가 branch name을 받아오므로 무결성 제약 조건을 만족한다.

[DESCRIPTION & KEY]

ATTRIBUTE	DESCRIPTION	KEY
account_number	계좌 번호	PK
branch_name	계좌가 속한 지점의 이름	FK
balance	계좌의 잔고	

[ATTRIBUTE CLASSIFICATION]

ATTRIBUTE	SIMPLE	COMPOSITE	SINGLE	MULTI	DERIVED	NULL
account_number	O	X	O	X	X	X
branch_name	O	X	O	X	X	X
balance	O	X	O	X	X	X

4.2) branch

Branch는 branch_name, branch_city, assets를 attribute로 가지며, primary key는 branch_name 이다.

[DESCRIPTION & KEY]

ATTRIBUTE	DESCRIPTION	KEY
branch_name	지점의 이름	PK
branch_city	지점이 속한 도시 이름	
assets	지점이 가지고 있는 총 자산	

[ATTRIBUTE CLASSIFICATION]

ATTRIBUTE	SIMPLE	COMPOSITE	SINGLE	MULTI	DERIVED	NULL
branch_name	O	X	O	X	X	X
branch_city	O	X	O	X	X	X
assets	O	X	O	X	X	X

4.3) customer

Customer는 customer_name, customer_street, customer_city를 attribute로 가지며, primary key는 customer_name이다.

[DESCRIPTION & KEY]

ATTRIBUTE	DESCRIPTION	KEY
customer_name	고객의 이름	PK
customer_street	고객이 사는 거리의 이름	
customer_city	고객이 사는 도시의 이름	

[ATTRIBUTE CLASSIFICATION]

ATTRIBUTE	SIMPLE	COMPOSITE	SINGLE	MULTI	DERIVED	NULL
customer_name	O	X	O	X	X	X
customer_street	O	X	O	X	X	X
customer_city	O	X	O	X	X	X

4.4) loan

Loan은 loan_number, branch_name, amount를 attribute로 가지며, primary key는 loan_number이다.

[DESCRIPTION & KEY]

ATTRIBUTE	DESCRIPTION	KEY
loan_number	대출 번호	PK

branch_name	대출을 가지고 있는 지점의 이름	
amount	대출액	

[ATTRIBUTE CLASSIFICATION]

ATTRIBUTE	SIMPLE	COMPOSITE	SINGLE	MULTI	DERIVED	NULL
loan_number	O	X	O	X	X	X
branch_name	O	X	O	X	X	X
amount	O	X	O	X	X	X

4.5) depositor

Depositor는 account_number, customer_name을 attribute로 가지며, account와 customer가 참여하는 relation이다. 따라서 각 entity의 primary key인 account_number, customer_name을 foreign key로 갖는다. 그리고 relation의 PK는 FK의 조합이다.

[DESCRIPTION & KEY]

ATTRIBUTE	DESCRIPTION	KEY
account_number	각 계좌에 해당하는 계좌 번호	PK: (a_n,c_n)
customer_name	각 계좌를 가지고 있는 고객의 이름	FK:a_n,c_n

[ATTRIBUTE CLASSIFICATION]

ATTRIBUTE	SIMPLE	COMPOSITE	SINGLE	MULTI	DERIVED	NULL
account_number	O	X	O	X	X	X
customer_name	O	X	O	X	X	X

[Cardinality]

Depositor RELATIONSHIP	
Participated Entities	Account, Customer
Cardinality	Many(account) to Many(customer)
Total/Partial	Account: Partial Customer: Partial

4.6) borrower

Borrower는 customer_name, loan_number를 attribute로 갖는 relation이다. Foreign key인 customer_name 과 loan_name의 조합을 primary key로 갖는다.

[DESCRIPTION & KEY]

ATTRIBUTE	DESCRIPTION	KEY
customer_name	대출을 가지고 있는 고객의 이름	PK: (c_n,l_n)
loan_number	각 고객이 대출한 대출 번호	FK: c_n, l_n

[ATTRIBUTE CLASSIFICATION]

ATTRIBUTE	SIMPLE	COMPOSITE	SINGLE	MULTI	DERIVED	NULL
customer_name	O	X	O	X	X	X
loan_number	O	X	O	X	X	X

[Cardinality]

Borrower RELATIONSHIP	
Participated Entities	Loan, Customer
Cardinality	Many(loan) to Many(customer)
Total/Partial	Loan: Partial Customer: Partial

4.7) account_branch

Account_branch는 account와 branch가 참여하는 relation이며, 각 entity의 primary key를 foreign key로 받아오고 그 조합을 PK로 갖는다. Borrower, depositor와 달리 본 문제에서 테이블 생성을 요구하지 않기 때문에, Erwin 상에서도 independent table을 만들지 않았다.

[DESCRIPTION & KEY]

ATTRIBUTE	DESCRIPTION	KEY
account_number	각 계좌에 해당하는 계좌 번호	PK: (a_n,b_n)
branch_name	대출을 가지고 있는 지점의 이름	FK: a_n,b_n

[ATTRIBUTE CLASSIFICATION]

ATTRIBUTE	SIMPLE	COMPOSITE	SINGLE	MULTI	DERIVED	NULL
account_number	O	X	O	X	X	X
branch_name	O	X	O	X	X	X

[Cardinality]

Account_Branch RELATIONSHIP	
Participated Entities	Account, Branch
Cardinality	One(Branch) to Many(Account)
Total/Partial	Account: Total(Total 이므로 Zero를 허용하지 않는다) Branch: Partial

4.8) loan_branch

Loan_branch는 branch와 loan이 참여하는 relation이며 각 entity의 primary key를 FK로 받는다. . Borrower, depositor와 달리 본 문제에서 테이블 생성을 요구하지 않기 때문에, Erwin 상에서도 independent table을 만들지 않았다.

[DESCRIPTION & KEY]

ATTRIBUTE	DESCRIPTION	KEY
loan_number	대출 번호	PK: (l_n,b_n)
branch_name	대출을 가지고 있는 지점의 이름	FK: b_n,b_n

[ATTRIBUTE CLASSIFICATION]

ATTRIBUTE	SIMPLE	COMPOSITE	SINGLE	MULTI	DERIVED	NULL
loan_number	O	X	O	X	X	X
branch_name	O	X	O	X	X	X

[Cardinality]

Loan_Branch RELATIONSHIP	
Participated Entities	Loan, Branch
Cardinality	One(Branch) to Many(Loan)
Total/Partial	Loan: Total(Total이므로 Zero를 허용하지 않는다) Branch: Partial

5) Domain Dictionary

무결성을 고려하려면 PRIMARY KEY(개체 무결성)와 FOREIGN KEY(참조 무결성)가 잘 설정되어야 하며 이를 위해 Domain Dictionary를 통해 Data type가 Constraint를 정해주고 이에 위배되면 에러를 출력해야 한다.

Domain	Datatype	Description/Constraint
Account_number	VARCHAR2(20)	-계좌 번호를 위한 Domain으로써 문자, -, 숫자로 이루어져 있기 때문에 문자열로 지정하였다. -계좌마다 계좌 번호는 있어야 하므로 NULL 값을 허용하지 않는다. -계좌 번호의 길이는 20글자로 지정하였다.
Amount	INTEGER	-대출액을 위한 Domain으로써 금액을 나타내는 것

		이기 때문에 integer로 지정하였고 돈의 경우 음수는 없으므로 validation rule을 양수로 지정하였다. -각 대출에 대해 대출액은 반드시 존재하므로 NULL 값을 허용하지 않는다.
Assets	INTEGER	-각 지점이 가지고 있는 자산을 위한 Domain으로써 금액을 나타내는 것이기 때문에 integer로 지정하였고 돈의 경우 음수는 없으므로 validation rule을 양수로 지정하였다. -각 은행의 지점은 반드시 자산을 갖고 있어야 하므로 NULL 값을 허용하지 않는다.
Balance	INTEGER	-각 계좌에 있는 잔고를 위한 Domain으로써 금액을 나타내는 것이기 때문에 integer로 지정하였고 돈의 경우 음수는 없으므로 validation rule을 양수로 지정하였다. -각 계좌에 대해 반드시 잔고는 있어야 하므로 NULL 값을 허용하지 않는다.
Branch_city	VARCHAR2(20)	-각 지점이 속한 도시의 이름을 위한 Domain으로써 도시 이름이기 때문에 문자열로 지정하였다. -각 지점은 반드시 소속된 도시가 있어야 하므로 NULL값을 허용하지 않는다. -도시 이름의 길이는 20글자로 지정하였다.
Branch_name	VARCHAR2(20)	-각 지점의 이름을 위한 Domain으로써 지점 이름이기 때문에 문자열로 지정하였다. -각 지점의 이름은 반드시 있어야 하므로 NULL값을 허용하지 않는다. -각 지점의 이름은 20글자로 지정하였다.
Customer_city	VARCHAR2(20)	-각 고객이 살고 있는 도시의 이름을 위한 Domain으로써 도시 이름이기 때문에 문자열로 지정하였다. -도시는 반드시 이름이 있어야 하므로 NULL값을 허용하지 않는다. -각 도시의 이름은 20글자로 지정하였다.
Customer_name	VARCHAR2(20)	-각 고객의 이름을 위한 Domain으로써 고객 이름이기 때문에 문자열로 지정하였다. -고객의 이름은 반드시 있어야 하므로 NULL 값을 허용하지 않는다. -각 고객의 이름은 20글자로 지정하였다.
Customer_street	VARCHAR2(20)	-각 고객이 살고 있는 도로의 이름을 위한 Domain으로써 도로 이름이기 때문에 문자열로 지정하였다. -각 고객이 살고 있는 도로의 이름은 반드시 있어야

		하므로 NULL값을 허용하지 않는다. -각 도로의 이름은 20글자로 지정하였다.
Loan_number	VARCHAR2(10)	-대출 번호를 위한 Domain으로써 문자, -, 숫자로 이루어져 있으므로 문자열로 지정하였다. -각 대출에는 대출 번호가 반드시 있어야 하므로 NULL값을 허용하지 않는다. -대출 번호의 길이는 10글자로 지정하였다.

결과적으로 각 Attribute와 대응되는 Domain은 아래와 같다.

Attribute	Domain
Account_number	VARCHAR2(20)/account_number/NOT NULL
Amount	INTEGER/amount/NOT NULL
Assets	INTEGER/assets/NOT NULL
Balance	INTEGER/balance/NOT NULL
Branch_city	VARCHAR2(20)/branch_city/NOT NULL
Branch_name	VARCHAR2(20)/branch_name/NOT NULL
Customer_city	VARCHAR2(20)/customer_city/NOT NULL
Customer_name	VARCHAR2(20)/customer_name/NOT NULL
Customer_street	VARCHAR2(20)/customer_street/NOT NULL
Loan_number	VARCHAR2(10)/loan_number/NOT NULL

2. 요구사항 해결

1) 다음 데이터를 입력하시오

1.1) branch

BRANCH_NAME	BRANCH_CITY	ASSETS
Brighton	Brooklyn	7100000
Downtown	Brooklyn	9000000
Mianus	Horseneck	400000
North Town	Rye	3700000
Perryridge	Horseneck	1700000
Pownal	Benninghton	300000
Redwood	Palo Alto	2100000
Round Hill	Horseneck	8000000

<BRANCH relation>

```
SQL> select * from branch;

BRANCH_NAME          BRANCH_CITY                ASSETS
-------------------- -------------------- ----------
Brighton             Brooklyn                7100000
Downtown             Brooklyn                9000000
Mianus               Horseneck                400000
North Town           Rye                     3700000
Perryridge           Horseneck               1700000
Pownal               Benninghton              300000
Redwood              Palo Alto               2100000
Round Hill           Horseneck               8000000

8 개의 행이 선택되었습니다.
```

INSERT INTO branch(branch_name, branch_city, assets) values('Brighton', 'Brooklyn', 7100000);

INSERT INTO branch(branch_name, branch_city, assets) values('Downtown', 'Brooklyn', 9000000);

INSERT INTO branch(branch_name, branch_city, assets) values('Mianus', 'Horseneck', 400000);

INSERT INTO branch(branch_name, branch_city, assets) values('North Town', 'Rye', 3700000);

INSERT INTO branch(branch_name, branch_city, assets) values('Perryridge', 'Horseneck', 1700000);

INSERT INTO branch(branch_name, branch_city, assets) values('Pownal', 'Benninghton', 300000);

INSERT INTO branch(branch_name, branch_city, assets) values('Redwood', 'Palo Alto', 2100000);

INSERT INTO branch(branch_name, branch_city, assets) values('Round Hill', 'Horseneck', 8000000);

1.2) customer

CUSTOMER_NAME	CUSTOMER_STREET	CUSTOMER_CITY
Adams	Spring	Pittsfield
Brooks	Senator	Brooklyn
Curry	North	Rye
Glenn	Sand Hill	Woodside
Green	Walnut	Stamford
Hayes	Main	Harrison
Johnson	Alma	Palo Alto
Jones	Main	Harrison
Lindsay	Park	Pittsfield
Smith	North	Rye
Turner	Putnam	Stamford
Williams	Nassau	Princeton

<CUSTOMER relation>

```
SQL> select * from customer;

CUSTOMER_NAME        CUSTOMER_STREET      CUSTOMER_CITY
-------------------- -------------------- --------------------
Adams                Spring               Pittsfield
Brooks               Senator              Brooklyn
Curry                North                Rye
Glenn                Sand Hill            Woodside
Green                Walnut               Stamford
Hayes                Main                 Harrison
Johnson              Alma                 Palo Alto
Jones                Main                 Harrison
Lindsay              Park                 Pittsfield
Smith                North                Rye
Turner               Putnam               Stamford

CUSTOMER_NAME        CUSTOMER_STREET      CUSTOMER_CITY
-------------------- -------------------- --------------------
Williams             Nassau               Princeton

12 개의 행이 선택되었습니다.
```

INSERT INTO customer(customer_name, customer_street, customer_city) values('Adams', 'Spring', 'Pittsfield');

INSERT INTO customer(customer_name, customer_street, customer_city) values('Brooks', 'Senator', 'Brooklyn');

INSERT INTO customer(customer_name, customer_street, customer_city) values('Curry', 'North', 'Rye');

INSERT INTO customer(customer_name, customer_street, customer_city) values('Glenn', 'Sand Hill', 'Woodside');

INSERT INTO customer(customer_name, customer_street, customer_city) values('Green', 'Walnut', 'Stamford');

INSERT INTO customer(customer_name, customer_street, customer_city) values('Hayes', 'Main', 'Harrison');

INSERT INTO customer(customer_name, customer_street, customer_city) values('Johnson', 'Alma', 'Palo Alto');

INSERT INTO customer(customer_name, customer_street, customer_city) values('Jones', 'Main', 'Harrison');

INSERT INTO customer(customer_name, customer_street, customer_city) values('Lindsay', 'Park', 'Pittsfield');

INSERT INTO customer(customer_name, customer_street, customer_city) values('Smith', 'North', 'Rye');

INSERT INTO customer(customer_name, customer_street, customer_city) values('Turner', 'Putnam', 'Stamford');

INSERT INTO customer(customer_name, customer_street, customer_city) values('Williams', 'Nassau', 'Princeton');

1.3) account

ACCOUNT_NUMBER	BRANCH_NAME	BALANCE
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350

<ACCOUNT relation>

```
SQL> select * from account;

ACCOUNT_NUMBER       BRANCH_NAME            BALANCE
-------------------- -------------------- ----------
A-101                Downtown                    500
A-102                Perryridge                  400
A-201                Brighton                    900
A-215                Mianus                      700
A-217                Brighton                    750
A-222                Redwood                     700
A-305                Round Hill                  350

7 개의 행이 선택되었습니다.
```

INSERT INTO account(account_number, branch_name, balance) values('A-101', 'Downtown', 500);

INSERT INTO account(account_number, branch_name, balance) values('A-102', 'Perryridge', 400);

INSERT INTO account(account_number, branch_name, balance) values('A-201', 'Brighton', 900);

INSERT INTO account(account_number, branch_name, balance) values('A-215', 'Mianus', 700);

INSERT INTO account(account_number, branch_name, balance) values('A-217', 'Brighton', 750);

INSERT INTO account(account_number, branch_name, balance) values('A-222', 'Redwood', 700);

INSERT INTO account(account_number, branch_name, balance) values('A-305', 'Round Hill', 350);

1.4) depositor

CUSTOMER_NAME	ACCOUNT_NUMBER
Hayes	A-102
Johnson	A-101
Johnson	A-201
Jones	A-217
Lindsay	A-222
Smith	A-215
Turner	A-305

<DEPOSITOR relation>

```
SQL> select * from depositor;

ACCOUNT_NUMBER       CUSTOMER_NAME
-------------------- --------------------
A-101                Johnson
A-102                Hayes
A-201                Johnson
A-215                Smith
A-217                Jones
A-222                Lindsay
A-305                Turner

7 개의 행이 선택되었습니다.
```

INSERT INTO depositor(customer_name, account_number) values('Hayes', 'A-102');

INSERT INTO depositor(customer_name, account_number) values('Johnson', 'A-101');

INSERT INTO depositor(customer_name, account_number) values('Johnson', 'A-201');

INSERT INTO depositor(customer_name, account_number) values('Jones', 'A-217');

INSERT INTO depositor(customer_name, account_number) values('Lindsay', 'A-222');

INSERT INTO depositor(customer_name, account_number) values('Smith', 'A-215');

INSERT INTO depositor(customer_name, account_number) values('Turner', 'A-305');

1.5) loan

LOAN_NUMBER	BRANCH_NAME	AMOUNT
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

<LOAN relation>

```
SQL> select * from loan;

LOAN_NUMBE BRANCH_NAME            AMOUNT
---------- -------------------- ----------
L-11       Round Hill                  900
L-14       Downtown                   1500
L-15       Perryridge                 1500
L-16       Perryridge                 1300
L-17       Downtown                   1000
L-23       Redwood                    2000
L-93       Mianus                      500

7 개의 행이 선택되었습니다.
```

INSERT INTO loan(loan_number, branch_name, amount) values('L-11', 'Round Hill', 900);

INSERT INTO loan(loan_number, branch_name, amount) values('L-14', 'Downtown', 1500);

INSERT INTO loan(loan_number, branch_name, amount) values('L-15', 'Perryridge', 1500);

INSERT INTO loan(loan_number, branch_name, amount) values('L-16', 'Perryridge', 1300);

INSERT INTO loan(loan_number, branch_name, amount) values('L-17', 'Downtown', 1000);

INSERT INTO loan(loan_number, branch_name, amount) values('L-23', 'Redwood', 2000);

INSERT INTO loan(loan_number, branch_name, amount) values('L-93', 'Mianus', 500);

1.6) borrower

CUSTOMER_NAME	LOAN_NUMBER
Adams	L-16
Curry	L-93
Hayes	L-15
Johnson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

<BORROWER relation>

```
SQL> select * from borrower;

LOAN_NUMBE CUSTOMER_NAME
---------- --------------------
L-11       Smith
L-14       Johnson
L-15       Hayes
L-16       Adams
L-17       Jones
L-17       Williams
L-23       Smith
L-93       Curry

8 개의 행이 선택되었습니다.

SQL>
```

INSERT INTO borrower(customer_name, loan_number) values('Adams', 'L-16');

INSERT INTO borrower(customer_name, loan_number) values('Curry', 'L-93');

INSERT INTO borrower(customer_name, loan_number) values('Hayes', 'L-15');

INSERT INTO borrower(customer_name, loan_number) values('Johnson', 'L-14');

INSERT INTO borrower(customer_name, loan_number) values('Jones', 'L-17');

INSERT INTO borrower(customer_name, loan_number) values('Smith', 'L-11');

INSERT INTO borrower(customer_name, loan_number) values('Smith', 'L-23');

INSERT INTO borrower(customer_name, loan_number) values('Williams', 'L-17');

2) 중복되지 않은 모든 지점들의 이름을 구하라.

```
SQL> select distinct branch_name
  2  from branch;

BRANCH_NAME
--------------------
Brighton
Downtown
Mianus
North Town
Perryridge
Pownal
Redwood
Round Hill

8 개의 행이 선택되었습니다.
```

3) Brighton지점의 대출을 가진 모든 고객들을 알파벳 역순으로 나열하라.

```
SQL> select customer_name
  2  from loan, borrower
  3  where borrower.loan_number=loan.loan_number
  4  and branch_name='Brighton'
  5  order by customer_name desc;

선택된 레코드가 없습니다.
```

4) 은행에서 대출, 계좌 혹은 둘 다를 가진 모든 고객을 나열하라.

```
SQL> select customer_name from depositor
  2  UNION select customer_name from borrower;

CUSTOMER_NAME
--------------------
Adams
Curry
Hayes
Johnson
Jones
Lindsay
Smith
Turner
Williams

9 개의 행이 선택되었습니다.
```

5) 은행에 대출과 계좌 모두를 가진 모든 고객을 나열하라.

```
SQL> select customer_name from depositor
  2  intersect select customer_name from borrower;

CUSTOMER_NAME
--------------------
Hayes
Johnson
Jones
Smith
```

6) 대출 총액이 가장 작은 고객의 이름과 대출 총액을 구하여라.

```
SQL> select customer_name,sum
  2  from (select customer_name, sum(amount) sum
  3        from borrower, loan
  4        where borrower.loan_number=loan.loan_number
  5        group by customer_name)
  6  where sum=(select min(sum)
  7             from (select customer_name, sum(amount) sum
  8                   from borrower, loan
  9                   where borrower.loan_number=loan.loan_number
 10                   group by customer_name));

CUSTOMER_NAME                SUM
-------------------- ----------
Curry                       500
```

7) Harrison과 Stamford에 살지 않으면서 계좌에 잔고의 합이 1000이하 있는 고객의 이름과 고객이 사는 도 시를 구하라.

```
SQL> select customer_name, customer_city
  2  from customer
  3  where customer_name in (select customer_name
  4                          from depositor, account
  5                          where depositor.account_number=account.account_number
  6                          group by customer_name
  7                          having sum(balance)<=1000)
  8  and customer_city!='Harrison'
  9  and customer_city!='Stamford';

CUSTOMER_NAME        CUSTOMER_CITY
-------------------- --------------------
Lindsay              Pittsfield
Smith                Rye
```

8) 은행에 계좌는 없지만 대출은 가지고 있는 모든 고객들을 나열하라.

```
SQL> select customer_name from borrower
  2  minus select customer_name from depositor;

CUSTOMER_NAME
--------------------
Adams
Curry
Williams
```

9) Perryridge 지점에서 계좌의 평균 잔고를 구하여라.

```
SQL> select avg(balance)
  2  from account
  3  where branch_name='Perryridge';

AVG(BALANCE)
------------
         400
```

10) 각 지점의 예금자들의 수를 구하라.

```
SQL> select branch_name, count(distinct customer_name) count
  2  from (select customer_name, branch_name
  3        from depositor, account
  4        where depositor.account_number=account.account_number)
  5  group by branch_name;

BRANCH_NAME              COUNT
-------------------- ----------
Brighton                      2
Downtown                      1
Mianus                        1
Perryridge                    1
Redwood                       1
Round Hill                    1

6 개의 행이 선택되었습니다.
```

11) 평균 잔고가 $500 이상인 지점 이름과 총 잔고를 나열하라.

```
SQL> select branch_name, sum
  2  from (select branch_name, sum(balance) sum, avg(balance) avg
  3        from account
  4        group by branch_name)
  5  where avg>=500;

BRANCH_NAME                SUM
-------------------- ----------
Mianus                      700
Brighton                   1650
Redwood                     700
Downtown                    500
```

12) Palo Alto에 살고 최소한 두 개의 계좌를 가진 각각의 고객들의 이름과 잔고의 합을 구하라.

```
SQL> select customer.customer_name, sum_balance
  2  from (select customer_name, sum(balance) sum_balance
  3        from account, depositor
  4        where depositor.account_number=account.account_number
  5        group by customer_name
  6        having count(balance)>=2), customer
  7  where customer_city='Palo Alto'
  8  and customer.customer_name in (select customer_name
  9                                 from account, depositor
 10                                 where depositor.account_number=account.account_number
 11                                 group by customer_name
 12                                 having count(balance) >=2);

CUSTOMER_NAME        SUM_BALANCE
-------------------- -----------
Johnson                     1400
```

13) 같은 도시에 사는 고객의 이름의 쌍을 구하여라.

```
SQL> select a.customer_name customerone, b.customer_name customertwo
  2  from customer a, customer b
  3  where a.customer_city=b.customer_city
  4  and a.customer_name<b.customer_name;

CUSTOMERONE          CUSTOMERTWO
-------------------- --------------------
Hayes                Jones
Adams                Lindsay
Curry                Smith
Green                Turner
```

14) 각 도시 별로 가장 높은 대출 총액을 가지고 있는 고객의 이름과 대출 총액을 구하여라. 단, 대출을 가 진 고객이 살지 않는 도시는 표시하지 않는다.

```
SQL> select customer_city,customername,sum
  2  from (select customer_name customername, sum(amount)sum
  3        from loan, borrower
  4        where loan.loan_number=borrower.loan_number
  5        group by customer_name) sum_loan, customer
  6  where customer.customer_name=sum_loan.customername
  7  and(customer_city,sum)
  8  in(select customer_city,max(sum)
  9     from (select customer_name customername, sum(amount) sum
 10           from loan, borrower
 11           where loan.loan_number=borrower.loan_number
 12           group by customer_name) sum_loan, customer
 13     where customer.customer_name=sum_loan.customername
 14     group by customer_city);

CUSTOMER_CITY        CUSTOMERNAME                  SUM
-------------------- -------------------- ----------
Pittsfield           Adams                       1300
Princeton            Williams                    1000
Harrison             Hayes                       1500
Palo Alto            Johnson                     1500
Rye                  Smith                       2900
```

15) Downtown, Perryridge를 제외한 지점에서 $1000 이상의 대출 총액을 지닌 고객들을 전부 구하라.

```
SQL> select customer_name
  2  from(select borrower.loan_number loan_num, amount, customer_name
  3       from loan, borrower
  4       where loan.loan_number=borrower.loan_number
  5       and loan.branch_name not in ('Downtown','Perryridge'))
  6  group by customer_name
  7  having sum(amount) >=1000;

CUSTOMER_NAME
--------------------
Smith
```

16) Downtown, Perryridge를 제외한 지점의 모든 대출에 대해여 고객의 이름과 대출 번호, 대출액을 구하라.

```
SQL> select customer_name, loan.loan_number loan_number,amount
  2  from loan,borrower
  3  where branch_name not in('Downtown','Perryridge')
  4  and loan.loan_number=borrower.loan_number;

CUSTOMER_NAME        LOAN_NUMBE     AMOUNT
-------------------- ---------- ----------
Smith                L-11              900
Smith                L-23             2000
Curry                L-93              500
```

17) 이름에 'or'이라는 부분 문자열이 포함된 거리에 살고 있는 모든 고객들의 이름을 구하여라.

```
SQL> select customer_name
  2  from customer
  3  where customer_street like '%or%';

CUSTOMER_NAME
--------------------
Brooks
Curry
Smith
```

18) 가장 높은 평균 잔고를 가진 고객의 이름과 총잔고를 구하라.

```
SQL> select customer_name, sum_balance
  2  from (select customer_name, sum(balance)sum_balance,avg(balance) avg_balance
  3        from depositor, account
  4        where depositor.account_number=account.account_number
  5        group by customer_name)
  6  where avg_balance in (select max(avg_balance)
  7                        from(select customer_name, avg(balance)avg_balance
  8                             from depositor, account
  9                             where depositor.account_number=account.account_number
 10                             group by customer_name));

CUSTOMER_NAME        SUM_BALANCE
-------------------- -----------
Jones                        750
```

19) 지점 이름과 그 지점에 계좌나 대출 둘 중 하나를 가진 고객 이름으로 구성된 View를 작성하라. (단 View의 이름은 all_customer이다.)

```
SQL> create or replace view all_customer as
  2  select branch_name, customer_name
  3  from depositor, account
  4  where depositor.account_number=account.account_number
  5  union
  6  select branch_name,customer_name
  7  from loan, borrower
  8  where loan.loan_number=borrower.loan_number;

뷰가 생성되었습니다.

SQL> select *from all_customer;

BRANCH_NAME          CUSTOMER_NAME
-------------------- --------------------
Brighton             Johnson
Brighton             Jones
Downtown             Johnson
Downtown             Jones
Downtown             Williams
Mianus               Curry
Mianus               Smith
Perryridge           Adams
Perryridge           Hayes
Redwood              Lindsay
Redwood              Smith

BRANCH_NAME          CUSTOMER_NAME
-------------------- --------------------
Round Hill           Smith
Round Hill           Turner

13 개의 행이 선택되었습니다.
```

20) 19에서 생성된 View를 이용하여 Downtown 지점의 모든 고객 이름을 나열하라.

```
SQL> select customer_name
  2  from all_customer
  3  where branch_name='Downtown';

CUSTOMER_NAME
--------------------
Johnson
Jones
Williams

SQL>
```