Database Notes by Rakib

Table Of Contents

[**Database Part**](#_xjntm5szr38a) **3**

[Show Databases](#_gl986yhhx2mf) 3

[Create Database:](#_d8smh5if5t8o) 3

[Retrive Database:](#_u3flcp5riywm) 3

[Update Database:](#_sp7n6c7ig15n) 3

[Delete Database:](#_afc66fqzl7gz) 3

[**Database Table Part**](#_3of2hz5we2nd) **4**

[Create Table:](#_kn4s3d2hf4rc) 4

[Retrive Table:](#_6rednf8gtpy9) 4

[Update Table:](#_a8ovai8lwnp3) 4

[Delete Table:](#_9yxrtqchge59) 4

[**Database Table Column**](#_3a5fhu489t14) **5**

[Create Column(add column):](#_62ytumka6npo) 5

[Retrive Column(Retrive Data):](#_3f1vgvtiugss) 5

[Update Column(Rename/Modify Name):](#_y1dyqfvqidjl) 5

[Delete Column:](#_qs11byj255dm) 5

[**Database Table Row**](#_uoegq7owdjcz) **6**

[Create Row(Insert Data):](#_r3in43o58pxn) 6

[Retrive Row(Retrive Data):](#_418p5q5c1p1e) 6

[Update Row(Update Data):](#_fji3qd9jy1s7) 6

[Delete Row:](#_fdx110ro9lkq) 6

[**Retrieval Queries(Select \*)**](#_1vby91rzyj89) **7**

[Simple Select:](#_pqdyr5a39b3f) 7

[Conditional Select:](#_t0jjvxckeqh8) 7

[Order By Select:](#_revtcyi1pq13) 7

[Distinct Select:](#_gu3xz2wuv4fl) 7

[Distinct == Group By Select:](#_6q8rqkmh24xy) 8

[Date Select:](#_cnfj38vjwosp) 8

[Like Varchar Select:](#_w5ccnzlk1crz) 8

[**Aggregation**](#_txq1diav66o5) **10**

[Count:](#_aca3qn8jogik) 10

[Min:](#_8uv462h6kx90) 10

[Max:](#_5vvfmi7rnkrh) 10

[Group By:](#_lumph8xxsa0x) 10

[Having:](#_yhkxln32nr7k) 10

[Proper SQL Query: SFWGH](#_rqt35lnfez91) 11

[**Subquery:**](#_gg5bddaaea1b) **12**

[IN:](#_yn8f4gj4jnwy) 12

[ANY:](#_4gsaen6hbdzp) 12

[ALL:](#_850wcdu8nrsf) 12

[**Join Operation:**](#_x228z4zd1hxw) **13**

[Inner Join: joins common](#_402njvbnnghq) 13

[Left Join: O><O => O>](#_sz61cn59vsgz) 13

[Right Join: O><O => <O](#_5522f7uwnfpt) 13

[Full Outer Join: joins all content of tables](#_irhwdcrju4cn) 13

[**Quiz 2**](#_hf3lkdnd7a8i) **14**

[**Worksheet 4**](#_vwepm0lgvy5i) **17**

[CUSTOMER](#_1w667ajn7vwg) 18

[DEPOSITOR](#_53u9rcoh7qbf) 18

[BORROWER](#_tfkzr7vw6n05) 18

[ACCOUNT](#_801jc97kf6n2) 19

[BRANCH](#_otohjnu84tqq) 19

[LOAN](#_pwv3lavfqnt1) 19

[Task 1](#_9fs50ca01rn8) 20

[Task 2](#_wsbwr4s3swdv) 20

[Task 3](#_1h7m9fjdjehd) 20

Database Notes by Me

### 

## **Database Part**

### **Show Databases**

-> show database;

### **Create Database**:

-> create database "name";

-> create database seu;

### **Retrive Database**:

-> USE 'DB\_NAME';

// show all tables od the used DB

-> show tables;

### **Update Database**:

-> CAN'T RENAME DATABASE

### **Delete Database**:

-> drop database 'DB\_NAME';

***N:B:*** Tried to note with sequence of CRUD.

Where **CRUD** = **C**reate **R**etrive **U**pdate **D**elete.

CRUD operations are used in different applications.

Sometimes those application is called as CRUD application.

## **Database Table Part**

### **Create Table**:

-> create table "table name"("Attribute/Column name" variable type, Attribute name" variable type);

-> create table studentinfo(uid int(10) primary key, uname varchar(50), uaddress varchar(11), uphone int(10));

### **Retrive Table**:

-> Desc 'Table\_NAME';

// show all tables od the used DB

-> Desc studentinfo

### **Update Table**:

-> ALTER TABLE `TableName`

RENAME TO `UpdatedTableName` ;

-> ALTER TABLE 'STUDENTINFO'

RENAME TO 'STD\_INFO'

### **Delete Table**:

-> drop table "tablename";

## 

## 

## **Database Table Column**

### **Create Column(add column)**:

-> ALTER TABLE 'TABLE\_NAME'

ADD 'COLUMN\_NAME' 'COL\_TYPE(X)';

-> ALTER TABLE std\_info

ADD dept varchar(10);

### **Retrive Column(Retrive Data)**:

-> SELECT 'COLUMN\_NAME' FROM 'TABLE\_NAME';

-> SELECT \* FROM STD\_INFO;

-> SELECT DEPT, UNAME FROM STD\_INFO;

### **Update Column(Rename/Modify Name)**:

// Modify Attribute Type

-> ALTER TABLE `TableName`

MODIFY 'std\_no' char(8);

// Rename Column Name

ALTER TABLE `TableName`

Change Column 'std\_no' 'std\_no\_updated' char(7);

### **Delete Column**:

-> ALTER TABLE `TableName`

Drop Column 'ColumnName';

-> ALTER TABLE `std\_info`

Drop Column 'std\_no';

## **Database Table Row**

### **Create Row(Insert Data)**:

// insert into table columns

-> insert into 'TableName' values(data, data, data, data);

-> insert into 'std\_info' values(01, "Rakib", "Dhaka, Bangladesh", 019999);

-> insert into 'std\_info'(uid, uname, phone) values(02, "Abdul", "Abc, Def");

### **Retrive Row(Retrive Data)**:

-> SELECT 'COLUMN\_NAME' FROM 'TABLE\_NAME' Where 'column\_name' = 'rowData;

-> SELECT \* FROM STD\_INFO where uid = 01;

-> SELECT DEPT, UNAME FROM STD\_INFO where uid = 01;

### **Update Row(Update Data)**:

// Modify Row Data

-> update 'Table\_name' set 'Col\_Name' = "Data" where 'Col\_Name' = "Data";

-> update Std\_Info set DEPT = "CSE" where uid = "01";

-> update Std\_Info set DEPT = "CSE", uName = "Rakibul" where uid = "01";

### **Delete Row**:

-> Delete from 'Table\_name' where 'Col\_Name' = "Data";

-> Delete from Std\_Info where uid = "01";

-> Delete from Std\_Info where uid >= "01" and uid <= "03";

## **Retrieval Queries(Select \*)**

### **Simple Select:**

1) select \* from STUDENT;

2) select STD\_NO, NAME, CGPA from STUDENT;

3) select NAME, CGPA\*200+200 as MARKS from STUDENT;

4) select STD\_NO, NAME, upper(DEPT) from STUDENT;

### **Conditional Select:**

11) select NAME, DEPT, CGPA from student where CGPA > 3.95;

12) select NAME, DEPT, CGPA from student where CGPA >= 3.50 && CGPA <= 3.90;

or,

select NAME, DEPT, CGPA from student where CGPA Between 3.50 and 3.90;

13) select NAME from STUDENT where DEPT = "CSE" || DEPT = "EEE";

### **Order By Select:**

7) select DEPT from STUDENT order by dept asc;

8) select NAME from STUDENT order by GRAD\_DATE asc;

select NAME from STUDENT order by NAME DESC;

9) select STD\_NO, NAME, CGPA from STUDENT order by CGPA desc;

select STD\_NO, NAME, CGPA from STUDENT order by NAME asc;

0) select STD\_NO, NAME, CGPA from STUDENT order by CGPA desc, NAME asc;

### **Distinct Select:**

6) select distinct DEPT from STUDENT;

### 

### **Distinct == Group By Select:**

6) select distinct DEPT from STUDENT;

or,

6) select DEPT from STUDENT GROUP BY DEPT;

### **Date Select:**

14) select NAME from STUDENT where year(GRAD\_DATE) = "2008" && CGPA > 3.70;

15) select \* from STUDENT where year(GRAD\_DATE) != "2008" && year(GRAD\_DATE) != "2007";

### **Like Varchar Select:**

16) select \* from STUDENT where name like "s%"; //starts with s

-> select \* from STUDENT where name like "s%a";//starts with s, ends with a

17) select \* from STUDENT where NAME like "%a%a%"; // at least two 'a' in string

| LIKE 'a%' | Finds any values that start with "a" |
| --- | --- |
| LIKE '%a' | Finds any values that end with "a" |
| LIKE '%abc%' | Finds any values that have "abc" in any position |
| LIKE '\_r%' | Finds any values that have "r" in the second position |
| LIKE 'a\_%' | Finds any values that start with "a" and are at least 2 characters in length |
| LIKE 'a\_\_%' | Finds any values that start with "a" and are at least 3 characters in length |
| LIKE 's%a' | Finds any values that start with "s" and ends with "a" |

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## **Aggregation**

### **Count:**

c) select count(std\_no) from student;

or

select count(\*) from student;

### **Min:**

a)select min(cgpa) from student;

### **Max:**

b)select max(grad\_date) from student;

i) select max(cgpa) from student where year(grad\_date) >= 2004;

0) select name, dept, cgpa from student where dept = "CSE" && cgpa > (select max(cgpa) from student where dept = "EEE");

### **Group By:**

d)select max(cgpa), min(cgpa), dept from student group by dept;

e)select max(grad\_date), min(grad\_date), dept from student group by dept;

f)select dept, count(std\_no) from student group by dept;

g)select dept, max(cgpa), min(cgpa) from student group by dept having count(std\_no) >= 3;

h) select dept, max(cgpa), min(cgpa), grad\_date from student where year(grad\_date) >= 2004 group by dept;

### **Having:**

//Having is used to check info of any aggregation operation and specially after group by

-> select max(cgpa), min(cgpa), dept from student group by dept having count(\*) >= 2;

### **Proper SQL Query: SFWGH**

**S**elect \* **F**rom student **W**here id = xx and name in(select name from student where dept = xx) **G**roup by dept **H**aving count(id) >= 1;

## **Subquery:**

a) select name from student where cgpa= (select max(cgpa) from student);

### **IN:**

b) select name, dept, cgpa from student where (dept, cgpa) **in**(select dept, min(cgpa) from student group by dept);

### **ANY:**

select name, dept, cgpa from student where dept = "CSE" && cgpa > **any**(select cgpa from student where dept = "EEE");

### **ALL:**

select name, dept, cgpa from student where dept = "CSE" && cgpa > **all**(select cgpa from student where dept = "EEE");

## **Join Operation:**

### **Inner Join: joins common**

* select \* from customerinfo c, checkinoutinfo cio where c.NID = cio.NID;
* select \* from customerinfo c inner join checkinoutinfo cio on where c.NID = cio.NID;
* SELECT NAME AS COURSE\_NAME, GRADE AS STUDENT\_GRADE FROM GRADE G, COURSE C WHERE G.ID = C.ID AND G.STUDENT\_ID = 234;
* SELECT C.NAME AS COURSE\_NAME, ROW\_NUMBER() OVER (ORDER BY C.NAME) AS CourseTally FROM TEACHER T, COURSE C WHERE T.ID = C.TEACHER\_ID AND T.ID = 3578;

### **Left Join: O><O => O>**

* select \* from customerinfo c left join checkinoutinfo cio on c.NID = cio.NID;

### **Right Join: O><O => <O**

* select \* from customerinfo c Right join checkinoutinfo cio on c.NID = cio.NID;

### **Full Outer Join: joins all content of tables**

* select \* from customerinfo c FULL OUTER JOIN checkinoutinfo cio on c.NID = cio.NID;

## **Quiz 2**

| For questions involving the formula **r = (n%x)+1**, you should be aware of the following:   * **x** will be a given number, which will typically be the **total number of rows** in a table * **n** is the **last two digits of your ID** * **r** is the **row number** of the table that you should use * **%** sign denotes **modulus** operation |
| --- |

**Instructions:**

* Insert your name and ID at the top
* Rename your file with only your ID
* **You cannot attempt this exam using pen and paper**
* Do not modify this file in any way other than where it is explicitly stated that you should replace a line
* Final file must be converted to PDF
* The **last 15 minutes** must be reserved solely for file **submission**

Given below is partial database state of an educational institution.

| r | RELATION |
| --- | --- |
| 1 | STUDENT   | ID | Name | Email | Password | | --- | --- | --- | --- | | 121 | Istiaq | 121@seu.edu.bd | Fsa7f6 | | 234 | Ahmed | 234@seu.edu.bd | Sf6saf | | …….. | | | | |
| 2 | GRADE   | ID | Student\_ID | Grade | | --- | --- | --- | | CSE383 | 121 | A+ | | CSE384 | 121 | A+ | | CSE384 | 234 | B- | | …….. | | | |
| 3 | COURSE   | ID | Name | Teacher\_ID | | --- | --- | --- | | CSE383 | Database Design | 3578 | | CSE384 | Database Design Lab | 7651 | | …….. | | | |
| 4 | TEACHER   | ID | Name | | --- | --- | | 3578 | Iqbal | | 7651 | Hossain | | …….. | | |

Additionally, the following information is given regarding the relations:

* STUDENT: ID (PK)
* GRADE: ID and Student\_ID (PK), Student\_ID (FK references STUDENT’s ID)
* Teacher: ID (PK)
* Course: ID (PK), Teacher\_ID (FK references TEACHER’s ID)

1. For RELATION, r = (n%3)+1, write down the following:

1. SQL statements for **creating** the relation. Ensure that you use sensible data types for the attributes. [3]
2. SQL statements needed to **insert** two new rows of your own, using relevant data from your own imagination. [2]
3. Insert a single **screenshot** showing SQL statements from both a) and b) after they have been executed in a DBMS. If you see an error, **explain** the reasoning behind it. If you do not see an error, explain what could’ve caused one even if there were no syntax errors. [2]

Note that your marks for answers a) and b) are dependent upon you uploading an appropriate screenshot in c).

|  |  |
| --- | --- |

1. CREATE TABLE course (

ID VARCHAR(7) NOT NULL PRIMARY KEY,

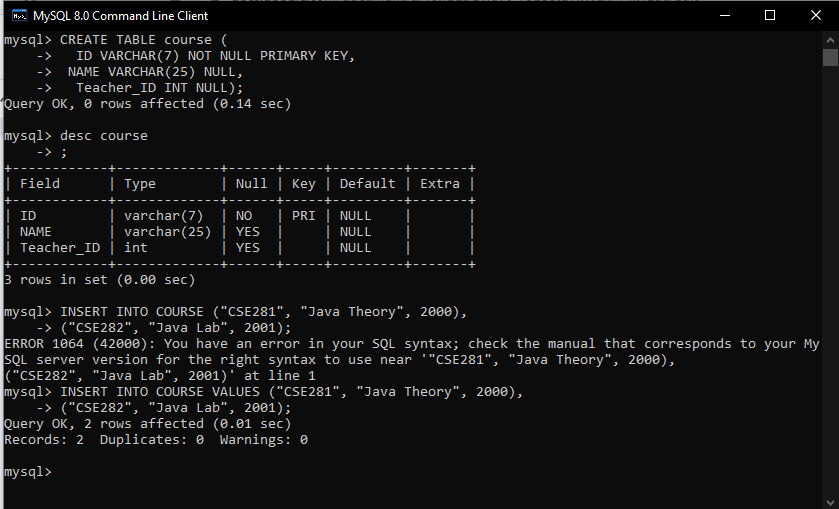
NAME VARCHAR(25) NULL,

Teacher\_ID INT NULL);

1. INSERT INTO COURSE VALUES ("CSE281", "Java Theory", 2000),

("CSE282", "Java Lab", 2001);

1. There is an error because I didn't use the "VALUES" SQL keyword during inserting corresponding columns data.



| For questions 2 and 3 below, you have to **write SQL statements** to satisfy the given requirements and **explain** them (the statements) in your own words. Correct statements without valid explanations will not be eligible for any marks. |
| --- |

2. Retrieve the course names and course grades obtained by the student with the ID 234.

|  |  |
| --- | --- |

SELECT NAME AS COURSE\_NAME, GRADE AS STUDENT\_GRADE FROM GRADE G, COURSE C WHERE G.ID = C.ID AND G.STUDENT\_ID = 234;

3. Retrieve the number of courses taught by the teacher whose ID is 3578. Your table should have a single column named CourseTally containing the number of courses.

|  |  |
| --- | --- |

SELECT C.NAME AS COURSE\_NAME, ROW\_NUMBER() OVER (ORDER BY C.NAME) AS CourseTally FROM TEACHER T, COURSE C WHERE T.ID = C.TEACHER\_ID AND T.ID = 3578;

## **Worksheet 4**

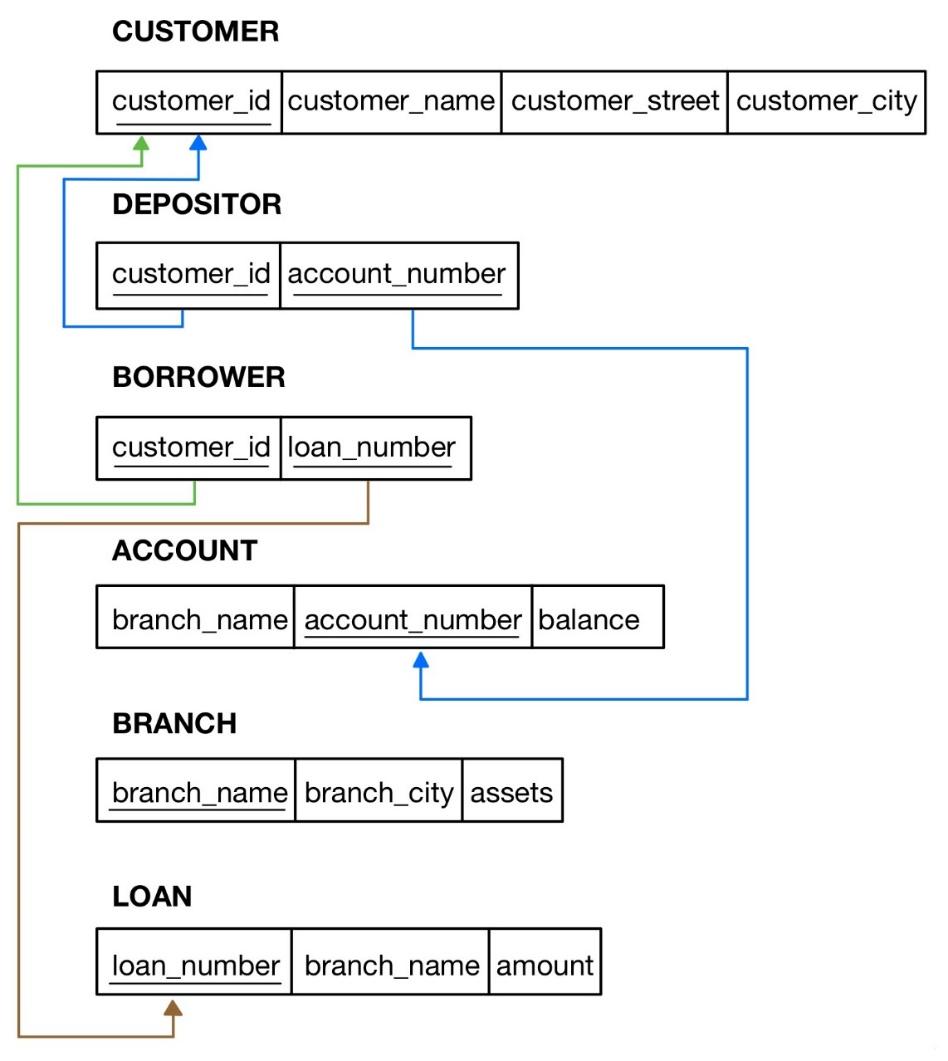
CSE384/CSE3012: Database Design Lab

**Worksheet 4**: Constraints, Joins



Given below is the schema diagram of a bank database. Apart from the primary key and foreign key constraints, it the following two constraints:

* Assets must always be greater than or equal to zero in the BRANCH table
* Balance must always be greater than or equal to zero in the ACCOUNT table



The aforementioned database is populated with the data given below.

### **CUSTOMER**

+-------------+---------------+-----------------+---------------+

| **customer\_id | customer\_name | customer\_street | customer\_city** |

+-------------+---------------+-----------------+---------------+

| C-101 | Jones | Main | Harrison |

| C-201 | Smith | North | Rye |

| C-211 | Hayes | Main | Harrison |

| C-212 | Curry | North | Rye |

| C-215 | Lindsay | Park | Pittsfield |

| C-220 | Turner | Putnam | Stamford |

| C-222 | Williams | Nassau | Princeton |

| C-225 | Adams | Spring | Pittsfield |

| C-226 | Johnson | Alma | Palo Alto |

| C-233 | Glenn | Sand Hill | Woodside |

| C-234 | Brooks | Senator | Brooklyn |

| C-255 | Green | Walnut | Stamford |

+-------------+---------------+-----------------+---------------+

### **DEPOSITOR**

+-------------+----------------+

| **customer\_id | account\_number** |

+-------------+----------------+

| C-101 | A-217 |

| C-201 | A-215 |

| C-211 | A-102 |

| C-215 | A-222 |

| C-220 | A-305 |

| C-226 | A-101 |

| C-226 | A-201 |

+-------------+----------------+

### **BORROWER**

+-------------+-------------+

| **customer\_id | loan\_number** |

+-------------+-------------+

| C-101 | L-17 |

| C-201 | L-11 |

| C-201 | L-23 |

| C-211 | L-15 |

| C-212 | L-93 |

| C-222 | L-17 |

| C-225 | L-16 |

| C-226 | L-14 |

+-------------+-------------+

### **ACCOUNT**

+-------------+----------------+---------+

| **branch\_name | account\_number | balance** |

+-------------+----------------+---------+

| Downtown | A-101 | 500 |

| Perryridge | A-102 | 400 |

| Brighton | A-201 | 900 |

| Mianus | A-215 | 700 |

| Brighton | A-217 | 750 |

| Redwood | A-222 | 700 |

| Round Hill | A-305 | 350 |

+-------------+----------------+---------+

### **BRANCH**

+-------------+-------------+---------+

| **branch\_name | branch\_city | assets** |

+-------------+-------------+---------+

| Brighton | Brooklyn | 7100000 |

| Downtown | Brooklyn | 9000000 |

| Mianus | Horseneck | 400000 |

| North Town | Rye | 3700000 |

| Perryridge | Horseneck | 1700000 |

| Pownal | Bennington | 300000 |

| Redwood | Palo Alto | 2100000 |

| Round Hill | Horseneck | 8000000 |

+-------------+-------------+---------+

### **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 |

+-------------+-------------+--------+

### **Task 1**

Create a bank database. Then, write SQL statements to create the relations in that database by referring to the schema. Ensure that you use sensible data types for the attributes by referring to both the database schema and state.

### **Task 2**

Populate the database with the data given above.

### **Task 3**

1. Find the names and cities of customers who have a loan at Perryridge branch

SELECT CUSTOMER\_NAME, CUSTOMER\_CITY FROM CUSTOMER C, BORROWER B, LOAN L

WHERE C.CUSTOMER\_ID = B.CUSTOMER\_ID

AND L.LOAN\_NUMBER = B.LOAN\_NUMBER

AND L.BRANCH\_NAME = 'PERRYRIDGE';

1. Find which accounts have balances between 700 and 900.

select a.account\_number, a.balance from depositor d, customer c, account a where d.customer\_id = c.customer\_id and

d.account\_number = a.account\_number and a.balance >= 700 && a.balance <= 900;

1. Find the names of customers on streets with names ending in "Hill".

SELECT C.CUSTOMER\_STREET FROM CUSTOMER AS C WHERE CUSTOMER\_STREET LIKE '%HILL';

1. Find the names of customers with accounts at a branch where Johnson has an account.

SELECT CUSTOMER\_NAME FROM CUSTOMER C, DEPOSITOR D, ACCOUNT A

WHERE C.CUSTOMER\_ID = D.CUSTOMER\_ID

AND D.ACCOUNT\_NUMBER = A.ACCOUNT\_NUMBER

AND BRANCH\_NAME IN (SELECT BRANCH\_NAME FROM CUSTOMER C, DEPOSITOR D,

ACCOUNT A

WHERE C.CUSTOMER\_ID = D.CUSTOMER\_ID

AND D.ACCOUNT\_NUMBER = A.ACCOUNT\_NUMBER

AND CUSTOMER\_NAME = 'JOHNSON')

AND CUSTOMER\_NAME != 'JOHNSON';

1. Find the names of customers with an account but not a loan at Mianus branch.

SELECT C.CUSTOMER\_NAME FROM ACCOUNT A, CUSTOMER C, DEPOSITOR D

WHERE A.ACCOUNT\_NUMBER = D.ACCOUNT\_NUMBER

AND C.CUSTOMER\_ID = D.CUSTOMER\_ID

AND A.BRANCH\_NAME = 'MIANUS'

AND C.CUSTOMER\_NAME NOT IN(SELECT C.CUSTOMER\_NAME FROM LOAN L, CUSTOMER C, BORROWER B

WHERE C.CUSTOMER\_ID = B.CUSTOMER\_ID

AND B.LOAN\_NUMBER = L.LOAN\_NUMBER

AND L.BRANCH\_NAME = 'MIANUS');

1. Find the names of branches whose assets are greater than the assets of some branch in Brooklyn.

SELECT BRANCH\_NAME, BRANCH\_CITY FROM BRANCH B WHERE ASSESTS > any(SELECT ASSESTS FROM BRANCH B WHERE B.BRANCH\_CITY = 'BROOKLYN')

B.BRANCH\_CITY != 'BROOKLYN';

or,

SELECT BRANCH\_NAME, BRANCH\_CITY FROM BRANCH B WHERE ASSETS > any(SELECT ASSETS FROM BRANCH B

WHERE B.BRANCH\_NAME IN (SELECT branch\_name FROM branch WHERE branch\_city = "brooklyn"))

AND B.BRANCH\_NAME NOT IN (SELECT branch\_name FROM branch WHERE branch\_city = "brooklyn");

1. Find the set of names of branches whose assets are greater than the assets of all branches in Horseneck.

SELECT BRANCH\_NAME, BRANCH\_CITY FROM BRANCH B

WHERE ASSETS > ALL(SELECT ASSETS FROM BRANCH B WHERE B.BRANCH\_CITY = 'HORSENECK')

AND B.BRANCH\_CITY != 'Horseneck';

or,

SELECT BRANCH\_NAME, BRANCH\_CITY FROM BRANCH B

WHERE ASSETS > (SELECT MAX(ASSETS) FROM BRANCH B WHERE B.BRANCH\_CITY = 'HORSENECK')

AND B.BRANCH\_CITY != 'Horseneck';

1. Find the set of names of customers at Brighton branch, in alphabetical order.

SELECT CUSTOMER\_NAME FROM CUSTOMER C, DEPOSITOR D, ACCOUNT A

WHERE C.CUSTOMER\_ID = D.CUSTOMER\_ID

AND D.ACCOUNT\_NUMBER = A.ACCOUNT\_NUMBER

AND A.BRANCH\_NAME = 'BRIGHTON' ORDER BY CUSTOMER\_NAME ASC;

1. Show the loan data, ordered by decreasing amounts and increasing loan numbers.

SELECT \* FROM LOAN ORDER BY AMOUNT DESC, LOAN\_NUMBER ASC;

1. Find the names of each branch and the number of customers having at least one account at that branch.

SELECT A.BRANCH\_NAME, COUNT(C.CUSTOMER\_ID) FROM ACCOUNT A, CUSTOMER C, DEPOSITOR D

WHERE A.ACCOUNT\_NUMBER = D.ACCOUNT\_NUMBER

AND D.CUSTOMER\_ID = C.CUSTOMER\_ID

GROUP BY A.BRANCH\_NAME

HAVING COUNT(C.CUSTOMER\_ID) >= 1;

1. Find the average balance of all customers in ‘Palo Alto’ having at least 2 accounts.

SELECT A.ACCOUNT\_NUMBER, C.CUSTOMER\_CITY, C.CUSTOMER\_ID, AVG(A.BALANCE) FROM ACCOUNT A, CUSTOMER C, DEPOSITOR D

WHERE A.ACCOUNT\_NUMBER = D.ACCOUNT\_NUMBER

AND D.CUSTOMER\_ID = C.CUSTOMER\_ID

AND C.CUSTOMER\_CITY = 'PaloAlto'

HAVING COUNT(A.ACCOUNT\_NUMBER) >= 2;