# Lab Question bank or Question pattern SQL statements, Hadoop and java map reduce operations

## SQL statements

1. Create the following tables under MCIS_your Regester number database

| branch-name | account-number | balance |
|---|---|---|
| Downtown | A-101 | 500 |
| Mianus | A-215 | 700 |
| Perrryridge | A-102 | 400 |
| Round Hill | A-305 | 350 |
| Brighton | A-201 | 900 |
| Redwood | A-222 | 700 |
| Brighton | A-217 | 750 |
| Account relation | | |

| Customer-name | Customer-street | Customer-city |
|---|---|---|
| Jones | Main | Harrison |
| Smith | North | Rye |
| Hayes | Main | Harrison |
| Curry | North | Rye |
| Lindsay | Park | Pittsfield |
| Turner | Putnam | Stamford |
| Williams | Nassau | Princeton |
| Adams | Spring | Pittsfield |
| Johnson | Alma | Palo Alto |
| Glenn | Sand Hill | Woodside |
| Brooks | Senator | Brooklyn |
| Green | Walnut | Stamford |
| customer relation | | |

| Customer-name | Loan-Number |
|---|---|
| Jones | L-17 |
| Smith | L-23 |
| Hayes | L-15 |
| Jackson | L-14 |
| Currry | L-93 |
| Smith | L-11 |
| Williams | L-17 |
| Adams | L-16 |
| borower relation | |

| branch-name | branch-city | balance |
|---|---|---|
| Downtown | Brooklyn | 9000000 |
| Redwood | Palo Alto | 2100000 |
| Perrryridge | Horseneck | 1700000 |
| Mianus | Horseneck | 400000 |
| Round Hill | Horseneck | 8000000 |
| Pownal | Bennington | 3000000 |
| Northton | Rye | 3700000 |
| Brighton | Brooklyn | 7100000 |
| Branch relation | | |

| Customer-name | Account-Number |
|---|---|
| Jones | A-101 |
| Smith | A-215 |
| Hayes | A-102 |
| Turner | A-305 |
| Johnson | A-201 |
| Jones | A-217 |
| Lindsay | A-222 |
| depositor relation | |

| branch-name | Loan - number | amount |
|---|---|---|
| Downtown | L-17 | 1000 |
| Redwood | L-23 | 2000 |
| Perrryridge | L-15 | 1500 |
| Downtown | L-14 | 1500 |
| Mianus | L-93 | 500 |
| Round Hill | L-11 | 900 |
| Perrryridge | L-17 | 1300 |
| Loan relation | | |

Set 1

1. Create branch table and Declare *branch_name* as the primary key for *branch, and branch_city should not take NULL values*

Ans –

**create table** *branch*
(*branch_name* char(15)**,**
*branch_city* char(30) **not null**,
*assets* integer,
**primary key** (*branch_name*))

2. Add a new tuple to *account with values 'A-9732', 'Perryridge', 1200*

Ans –

**insert into** *account*
**values** ('A-9732', 'Perryridge', 1200)

3. Use The **alter table** command to add new attribute PhoneNumber to an existing relation customer

Ans –

**alter table** *r* **add** *A D*
where *A* is the name of the attribute to be added to relation *r* and *D* is the domain of *A.*
**alter table** customer **add** PhoneNumber *int*

4. Use The **drop table** command to remove the new attribute column PhoneNumber from relation customer
The **alter table** command can also be used to drop attributes of a relation:
**alter table** *r* **drop** *A*
where *A* is the name of an attribute of relation *r*
**alter** customer *r* **drop** PhoneNumber
5. Find the names of all branches in the *loan* relation and remove duplicates.
**select distinct** *branch_name*
**from** *loan*

Set 2

1. Find the names of all branches in the *loan* relation and do not remove duplicates.
**select all** *branch_name*
**from** *loan*
2. Display all the contents of the table without mentioning names of the attributes
**select** * **from** *loan*
3. Multiply amount attribute with value 100 for all the loan numbers in the loan
**select** *loan_number, branch_name, amount* * 100 **from** *loan*
4. Find all loans over $1200
**select** * **from** *loan* ***where amount > 1200***
5. Find the loan number for each loan of an amount > $1200
**select** *loan_number* **from** *loan* ***where amount > 1200***

Set 3

6. Provide as a gift for all loan customers of the Perryridge branch, a $200 savings account. Let the loan number serve as the account number for the new savings account
**insert into** *account*
**select** *loan_number, branch_name,* 200
**from** *loan*
**where** *branch_name* = 'Perryridge'
**insert into** *depositor*

```
select customer_name, loan_number
from loan, borrower
where branch_name = 'Perryridge'
      and loan.account_number = borrower.account_number
```

7. Increase all accounts with balances over $10,000 by 6%, all other accounts receive 5%.
   Write two **update** statements:

```
Update account
        set balance = balance * 1.06
        where balance > 10000
Update account
        set balance = balance * 1.05
        where balance ≤ 10000
```

8. Increase all accounts with balances over $10,000 by 6%, all other accounts receive 5%.(Use case statement)

```
update account
   set balance =  case
                    when balance <= 10000 then balance *1.05
                    else   balance * 1.06
                  end
```

9. Find all customers who have at least two accounts at the Perryridge branch.

```
select distinct T.customer_name

from depositor  as T

where not unique (

    select R.customer_name

    from account, depositor as R

    where T.customer_name = R.customer_name and

          R.account_number = account.account_number  and

          account.branch_name = 'Perryridge')
```

10. Write the SQL queries for the operations below using the relations loan and borrower given below?

- Relation *loan*

| loan-number | branch-name | amount |
|---|---|---|
| L-170 | Downtown | 3000 |
| L-230 | Redwood | 4000 |
| L-260 | Perryridge | 1700 |

- Relation borrower

| customer-name | loan-number |
|---|---|
| Jones | L-170 |
| Smith | L-230 |
| Hayes | L-155 |

i. Natural join
ii. Left Outer join
iii. Right outer join

Natural join

*loan* ***natural right outer join*** *borrower*

Left Outer join

*loan* **left outer join** *borrower* **on**
*loan.loan_number = borrower.loan_number*

Right outer join

*loan* **natural inner join** *borrower*

Full Outer join

*loan* **full outer join** *borrower* **using** (*loan_number*)