Solutions for Oracle Statements

USE ORACLE SQL Command LINE :

First Command

```
SQL> Connect sys as sysdba
Password: system
```

Problem Statement

1. Account(Acc_no, branch_name,balance) branch(branch_name,branch_city,assets)

customer(cust_name,cust_street,cust_city) Depositor(cust_name,acc_no) Loan(loan_no,branch_name,amount)

Borrower(cust_name,loan_no)

Solve following query:

Create above tables with appropriate constraints like primary key, foreign key, check constrains, not null etc

Q1. Find the names of all branches in loan relation.

Q2. Find all loan numbers for loans made at Akurdi Branch with loan amount > 12000.

Q3. Find all customers who have a loan from bank. Find their names, loan_no and loan amount. Q4. List all customers in

alphabetical order who have loan from Akurdi branch.

Q5. Find all customers who have an account or loan or both at bank.

Q6. Find all customers who have both account and loan at bank.

Solution:

```
CREATE TABLE Account (
   Acc_no INT PRIMARY KEY,
   branch_name VARCHAR(255) NOT NULL,
   balance DECIMAL(10, 2) NOT NULL);
```

create the branch table

```
CREATE TABLE branch (
   branch_name VARCHAR(255) PRIMARY KEY,
   branch_city VARCHAR(255) NOT NULL,
   assets DECIMAL(15, 2) NOT NULL
);

-- Create the customer table
CREATE TABLE customer (
   cust_name VARCHAR(255) PRIMARY KEY,
   cust_street VARCHAR(255) NOT NULL,
   cust_city VARCHAR(255) NOT NULL
);

-- Create the Depositor table
CREATE TABLE Depositor (
   cust_name VARCHAR(255) NOT NULL,
   acc_no INT NOT NULL,
   PRIMARY KEY (cust_name, acc_no),
   FOREIGN KEY (cust_name) REFERENCES customer(cust_name),
   FOREIGN KEY (acc_no) REFERENCES Account(Acc_no)
);
```

```
-- Create the Loan table

CREATE TABLE Loan (

   loan_no INT PRIMARY KEY,

   branch_name VARCHAR(255) NOT NULL,

   amount DECIMAL(10, 2) NOT NULL,

   FOREIGN KEY (branch_name) REFERENCES branch(branch_name)

);


-- Create the Borrower table

CREATE TABLE Borrower (

   cust_name VARCHAR(255) NOT NULL,

   loan_no INT NOT NULL,

   PRIMARY KEY (cust_name, loan_no),

   FOREIGN KEY (cust_name) REFERENCES customer(cust_name),

   FOREIGN KEY (loan_no) REFERENCES Loan(loan_no)

);
```

Q 1 Find the names of all branches in the loan relation.

```
SELECT DISTINCT branch_name

FROM Loan;
```

Q2 Find all loan numbers for loans made at Akurdi Branch with a loan amount greater than 12,000.

```
SELECT loan_no

FROM Loan

WHERE branch_name = 'Akurdi' AND amount > 12000;
```

Q3 Find all customers who have a loan from the bank. Find their names, loan_no, and loan amount.

```
SELECT C.cust_name, L.loan_no, L.amount

FROM customer C

INNER JOIN Borrower B ON C.cust_name = B.cust_name

INNER JOIN Loan L ON B.loan_no = L.loan_no;
```

Q 4 List all customers in alphabetical order who have a loan from the Akurdi branch.

```
SELECT DISTINCT C.cust_name

FROM customer C

INNER JOIN Borrower B ON C.cust_name = B.cust_name

INNER JOIN Loan L ON B.loan_no = L.loan_no

WHERE L.branch_name = 'Akurdi'

ORDER BY C.cust_name;
```

Q5 Find all customers who have an account or loan or both at the bank.

```
SELECT DISTINCT C.cust_name

FROM customer C

LEFT JOIN Depositor D ON C.cust_name = D.cust_name

LEFT JOIN Borrower B ON C.cust_name = B.cust_name

WHERE D.cust_name IS NOT NULL OR B.cust_name IS NOT NULL;
```

Q 6 Find all customers who have both an account and a loan at the bank.

```
SELECT C.cust_name

FROM customer C

INNER JOIN Depositor D ON C.cust_name = D.cust_name
```

INNER JOIN Borrower B ON C.cust_name = B.cust_name;

Problem Statement 2: Account(Acc_no, branch_name,balance)
branch(branch_name,branch_city,assets)

customer(cust_name,cust_street,cust_city) Depositor(cust_name,acc_no)
Loan(loan_no,branch_name,amount)

Borrower(cust_name,loan_no)

Solve following query:

Create above tables with appropriate constraints like primary key, foreign key, check constrains, not null etc

Q1. Find all customers who have both account and loan at bank.

Q2. Find all customer who have account but no loan at the bank.

Q3. Find average account balance at Akurdi branch.

Q4. Find the average account balance at each branch

Q5. Find no. of depositors at each branch

<u>Table Creation same as above</u>

Q 1 Find all customers who have both an account and a loan at the bank.

SELECT DISTINCT D.cust_name

FROM Depositor D

INNER JOIN Borrower B ON D.cust_name = B.cust_name;

Q2. Find all customers who have an account but no loan at the bank.

SELECT DISTINCT D.cust_name

FROM Depositor D

WHERE D.cust_name NOT IN (SELECT cust_name FROM Borrower);

Q3. Find the average account balance at the Akurdi branch.

SELECT AVG(balance) AS avg_balance

FROM Account

WHERE branch_name = 'Akurdi';

Q4. Find the average account balance at each branch.

SELECT branch_name, AVG(balance) AS avg_balance

FROM Account

GROUP BY branch_name;

Q5. Find the number of depositors at each branch.

SELECT branch_name, COUNT(*) AS num_depositors

FROM Depositor

GROUP BY branch_name;

Problem Statement 3 :

Account(Acc_no, branch_name,balance) branch(branch_name,branch_city,assets)

customer(cust_name,cust_street,cust_city) Depositor(cust_name,acc_no)
Loan(loan_no,branch_name,amount)

Borrower(cust_name,loan_no)

Solve following query:

Create above tables with appropriate constraints like primary key, foreign key, check constrains, not null etc

Q1. Find the branches where average account balance > 12000.

Q2. Find number of tuples in customer relation.

Q3. Calculate total loan amount given by bank.

Q4. Delete all loans with loan amount between 1300 and 1500.

Q5. Delete all tuples at every branch located in Nigdi.

Solution :

Create tables as above

Q1. Find the branches where the average account balance is greater than 12,000.

```
SELECT branch_name

FROM Account

GROUP BY branch_name

HAVING AVG(balance) > 12000;
```

Q2. Find the number of tuples in the customer relation.

```
SELECT COUNT(*) AS num_tuples

FROM customer;
```

Q3. Calculate the total loan amount given by the bank.

```
SELECT SUM(amount) AS total_loan_amount

FROM Loan;
```

Q4. Delete all loans with a loan amount between 1300 and 1500.

```
DELETE FROM Loan

WHERE amount BETWEEN 1300 AND 1500;
```

Q5. Delete all tuples at every branch located in Nigdi.

```
DELETE FROM branch

WHERE branch_city = 'Nigdi';
```

Problem Statement 4:

solve following join operations.

a. Create following Tables

cust_mstr(cust_no,fname,lname)

add_dets(code_no,add1,add2,state,city,pincode)

Retrieve the address of customer Fname as 'Ramesh' and Lname as 'Shinde'

b.Create following Tables

cust_mstr(custno,fname,lname)

acc_fd_cust_dets(codeno,acc_fd_no)

fd_dets(fd_sr_no,amt)

List the customer holding fixed deposit of amount more than 5000

c. Create following Tables

emp_mstr(e_mpno,f_name,l_name,m_name,dept,desg,branch_no)

branch_mstr(name,b_no)

List the employee details along with branch names to which they belong

<u>Solution :</u>

a. Create the tables and retrieve the address of the customer with Fname as 'Ramesh' and Lname as 'Shinde'.

```
-- Create the cust_mstr table

CREATE TABLE cust_mstr (

   cust_no INT PRIMARY KEY,

   fname VARCHAR(255),
    lname VARCHAR(255)

);


-- Create the add_dets table

CREATE TABLE add_dets (

    code_no INT PRIMARY KEY,

    add1 VARCHAR(255),

    add2 VARCHAR(255),

    state VARCHAR(255),

    city VARCHAR(255),

    pincode INT

);


-- Retrieve the address of the customer

SELECT add1, add2, state, city, pincode

FROM cust_mstr

JOIN add_dets ON cust_mstr.cust_no = add_dets.code_no

WHERE fname = 'Ramesh' AND lname = 'Shinde';
```

b. Create the tables and list the customers holding fixed deposits of more than 5000.

```
-- Create the cust_mstr table

CREATE TABLE cust_mstr (

    custno INT PRIMARY KEY,

    fname VARCHAR(255),

    lname VARCHAR(255)

);
```

```
-- Create the acc_fd_cust_dets table

CREATE TABLE acc_fd_cust_dets (

   codeno INT,

   acc_fd_no INT

);

-- Create the fd_dets table

CREATE TABLE fd_dets (

   fd_sr_no INT PRIMARY KEY,

   amt DECIMAL(10, 2)

);


-- List the customers holding fixed deposits of more than 5000

SELECT fname, lname

FROM cust_mstr

JOIN acc_fd_cust_dets ON cust_mstr.custno = acc_fd_cust_dets.codeno

JOIN fd_dets ON acc_fd_cust_dets.acc_fd_no = fd_dets.fd_sr_no

WHERE amt > 5000;
```

c. Create the tables and list the employee details along with the branch names to which they belong.

```
-- Create the emp_mstr table

CREATE TABLE emp_mstr (

   e_mpno INT PRIMARY KEY,

   f_name VARCHAR(255),

   l_name VARCHAR(255),

   m_name VARCHAR(255),

   dept VARCHAR(255),
    desg VARCHAR(255),
    branch_no INT
);
-- Create the branch_mstr table
CREATE TABLE branch_mstr (
    name VARCHAR(255),
    b_no INT PRIMARY KEY
);

-- List the employee details along with branch names
SELECT e_mpno, f_name, l_name, m_name, dept, desg, name AS branch_name
FROM emp_mstr
JOIN branch_mstr ON emp_mstr.branch_no = branch_mstr.b_no;
```

Problem Statement 5 :

Create Tables , Account(Acc_no, branch_name,balance) branch(branch_name,branch_city,assets) and solve following Queries.

a) Create View on Account table by selecting any two columns and perform insert update delete operations

b) Create view on Account and Branch table by selecting any one column from each table perform insert update delete operations

c) create updateable view on Account table by selecting any two columns and perform insert update delete operations

<u>Create table cust_mstr</u>

```
SQL> create table cust_mstr(cust_no numeric, fname varchar(20), lname varchar(20),
cust_pin numeric); Table created.
```

Create Table add_dets

```
SQL> create table add_dets(code_no numeric,pincode numeric);

Table created.
```

Insert into add_dets

```
insert into add_dets values (1,411041);

insert into add_dets values (2,411051);
insert into add_dets values (3,411071);
```

Insert data in cust_mstr

```
insert into cust_mstr values (1,'Sarika','Joshi',411051);

insert into cust_mstr values (2,'sanika','kulkarni',411071);

insert into cust_mstr values (3,'aniket','shinde',411091);

 insert into cust_mstr values (4,'kedar','pawar',411011);

insert into cust_mstr values (5,'Rahul','pandit',411031);
```

See the data in the tables:

```
SQL> select * from cust_mstr;
SQL> select * from add_dets;
```

1. **Query to get customers who do not have bank in their location:**

```
SQL> select * from cust_mstr where cust_pin not in (select distinct pincode from
add_dets) order by  cust_no;
```

Create View:

```
SQL> create view cust_mstr_vw as (select * from cust_mstr);

View created.
```

2. **Insert in View:**

```
SQL> insert into cust_mstr_vw values (6,'Sachin','tendulkar',411021);

1 row created.

SQL> insert into cust_mstr_vw values (7,'Virat','Kohli',411011);

1 row created.
```

3. **See the data in the view:**

```
SQL> select * from cust_mstr_vw;
```

4. **Update view :**

```
SQL> update cust_mstr_vw set cust_no=9, fname='VIRAT' where cust_no=7 and

lname='Kohli' ; 1 row updated.
```

5. **See the updated view:**

```
SQL> select * from cust_mstr_vw;
```

6. **Delete from view:**

```
SQL> delete from cust_mstr_vw where fname='VIRAT' ;

1 row deleted.
```

7. **See the deleted Data:**

```
SQL> select * from cust_mstr_vw;
```

Problem Statement 6 :

Consider table Stud(Roll, Att, Status)

Write a PL/SQL block for following requirement and handle the exceptions.

Roll no. of student will be entered by user. Attendance of roll no. entered by user will be checked in Stud table. If attendance is less than 75% then display the message “Term not granted” and set the status in stud table as “D”. Otherwise display message “Term granted” and set the status in stud table as “ND”

Solution :

Don’t Forget to write

Set serveroutput on

Create a Table:

create table stud1(roll_no number(5),attendance number(5),status varchar(7)); **Insert values in Table:**

```
        Insert into stud1(roll_no, attendance) values(101, 80);

        Insert into stud1(roll_no, attendance) values(102, 65);

        Insert into stud1(roll_no, attendance) values(103, 92);

        Insert into stud1(roll_no, attendance) values(104, 55);

        Insert into stud1(roll_no, attendance) values(105, 98);
```

```
SQL> Declare
 roll number(10);
 att number(10);
 Begin
 roll:=&roll;
 select attendance into att from stud1 where roll_no = roll;
 if att<75 then
 dbms_output.put_line(roll||'is detained');
 update stud1 set status='D' where roll_no=roll;
 else dbms_output.put_line(roll||'is not detained');
 update stud1 set status='ND' where roll_no=roll;
end if;
exception
when no_data_found then dbms_output.put_line(roll||'not found');
end;
/
```

Problem Statement 7 :

Write a PL/SQL stored Procedure for following requirements and call the procedure in appropriate PL/SQL block.

a. Borrower(Rollin, Name, DateofIssue, NameofBook, Status)

b. Fine(Roll_no,Date,Amt)

Accept roll_no & name of book from user.

Check the number of days (from date of issue), if days are between 15 to 30 then fine

amount will be Rs 5per day.

If no. of days>30, per day fine will be Rs 50 per day & for days less than 30, Rs. 5 per

day.

Solution :

```
sql> create table borrower(rollin int primary key,name varchar(20),dateofissue

date,nameofbook varchar(20),status varchar(20));

sql> create table fine(rollno int,foreign key(rollno) references borrower(rollin),returndate
date,amount  int);

SQL> insert into borrower values(1,'abc','01-AUG-2017','SEPM','PEN');

SQL> insert into borrower values(2,'xyz','01-JULY-2017','DBMS','PEN');

SQL> insert into borrower values(3,'pqr','08-AUG-2015','DBMS','PEN');
PROCEDURE calc_fine_lib() AS

DECLARE

roll number;

fine1 number;

noofdays int;

issuedate date;

BEGIN

roll:=&roll;

select dateofissue into issuedate from borrower where rollin = roll;

SELECT TO_DATE(sysdate,'DD-MON-YYYY') - TO_DATE(issuedate,'DD-MON-YY') INTO
noofdays  FROM dual;
```

```
if noofdays>15 and noofdays<=30 then

fine1:=noofdays*5;

insert into fine values(roll,'30-AUG-2017',fine1);

elsif noofdays>30 then

fine1:=((noofdays-30)*50) + 15*5;

else

insert into fine values(roll,'30-AUG-2017',fine1);

end if;

insert into fine values(roll,'30-AUG-2017',0);

update borrower set status='return' where rollin=roll;

end ;

/
```