**Practical 1**

**AIM- To study DDL-create and DML-insert commands.**

**(i) Create tables according to the following definition.**

**Query:**

SQL> CREATE TABLE DEPOSIT (ACTNO VARCHAR(5) ,CNAME VARCHAR(18) ,BNAME VARCHAR(18) , AMOUNT NUMBER(8,2) ,ADATE DATE);

**Query:**

SQL> CREATE TABLE BRANCH(BNAME VARCHAR(18),CITY VARCHAR(18));

**Output**:

Table created.

**Query:**

SQL> CREATE TABLE CUSTOMERS(CNAME VARCHAR(19) ,CITY VARCHAR(18));

**Output:**

Table created.

**Query:**

SQL> CREATE TABLE CUSTOMER(NAME VARCHAR(15),CITY VARCHAR(10));

**Output:**

Table created.

**Query:**

SQL>CREATE TABLE BORROW(LOANNO VARCHAR(5), CNAME VARCHAR(18),BNAME VARCHAR(18), AMOUNT NUMBER (8,2));

**Output:**

Table created.

**From the above given tables perform the following queries:**

1. **Describe deposit, branch**.

**Query1:**

SQL>DESC DEPOSIT;

**Output:**

Name Null? Type

----------------------------------------- -------- ----------------------------

ACTNO VARCHAR2(5)

CNAME VARCHAR2(18)

BNAME VARCHAR2(18)

AMOUNT NUMBER(8,2)

ADATE DATE

**Query2:**

SQL>DESC BRANCH;

**Output:**

Name Null? Type

----------------------------------------- -------- ----------------------------

LOANNO VARCHAR2(5)

CNAME VARCHAR2(18)

BNAME VARCHAR2(18)

AMOUNT NUMBER(8,2)

1. **Describe borrow, customers.**

**Query1:**

SQL> DESC CUSTOMER;

**Output:**

Name Null? Type

----------------------------------------- -------- ----------------------------

NAME VARCHAR2(15)

CITY VARCHAR2(10)

**Query2:**

SQL> DESC BORROW;

**Output:**

Name Null? Type

----------------------------------------- -------- ----------------------------

LOANNO VARCHAR2(5)

CNAME VARCHAR2(18)

BNAME VARCHAR2(18)

AMOUNT NUMBER(8,2)

1. **List all data from table DEPOSIT**.

**Query:**

SQL> SELECT \* FROM DEPOSIT;

Output:

ACTNO CNAME BNAME AMOUNT ADATE

----- ------------------ ------------------ ------- ---------

100 ANIL VRCE 1000 01-MAR-95

101 SUNIL AJNI 5000 04-JAN-96

102 MEHUL KAROLBAGH 3500 17-NOV-95

104 MADHURI CHANDI 1200 17-DEC-95

105 PRMOD M.G.ROAD 3000 27-MAR-96

106 SANDIP ANDHERI 2000 31-MAR-96

107 SHIVANI VIRAR 1000 05-SEP-95

108 KRANTI NEHRU PLACE 5000 02-JUL-95

109 MINU POWAI 7000 10-AUG-95

9 rows selected.

1. **List all data from table BORROW.**

**Query:**

SQL> SELECT \* FROM BORROW;

LOANN CNAME BNAME AMOUNT

----- ------------------ ------------------ ----------

201 ANIL VRCE 1000

206 MEHUL AJNI 5000

311 SUNIL DHARAMPETH 3000

321 MADHURI ANDHERI 2000

375 PRMOD VIRAR 8000

481 KRANTI NEHRU PLACE 3000

6 rows selected.

1. **List all data from table CUSTOMERS.**

**Query:**

SQL> SELECT \* FROM CUSTOMER;

**Output:**

NAME CITY

--------------- ----------

ANIL CALCUTTA

SUNIL DELHI

MEHUL BARODA

MANDAR PATNA

MADHURI NAGPUR

PRAMOD NAGPUR

SANDIP SURAT

SHIVANI BOMBAY

KRANTI BOMBAY

NAREN BOMBAY

10 rows selected.

1. **List all data from table BRANCH.**

**Query:**

SQL> SELECT \* FROM BRANCH;

**Output:**

BNAME CITY

------------------ ------------------

VRCE NAGPUR

AJNI NAGPUR

KAROLBAGH DELHI

CHANDI DELHI

DHARAMPETH NAGPUR

M.G.ROAD BANGLORE

ANDHERI BOMBAY

VIRAR BOMBAY

NEHRU PLACE DELHI

POWAI BOMBAY

10 rows selected.

1. **Give account no and amount of depositors.**

**Query:**

SQL> SELECT ACTNO, AMOUNT FROM DEPOSIT;

**Output:**

ACTNO AMOUNT

----- ----------

100 1000

101 5000

102 3500

104 1200

105 3000

106 2000

107 1000

108 5000

109 7000

9 rows selected.

1. **Give name of depositors having amount greater than 4000.**

**Query:**

SQL> SELECT CNAME FROM DEPOSIT WHERE AMOUNT>4000;

**Output:**

CNAME

------------------

SUNIL

KRANTI

MINU

1. **Give name of customers who opened account after date '1-12 96'.**

**Query:**

SQL> SELECT CNAME FROM DEPOSIT WHERE ADATE >'1-DEC-96';

**Output:**

no rows selected

**(10)Display actno and customer name of those depositor whose amount is greater than 2000**

**Query:**

SQL> SELECT ACTNO,CNAME FROM DEPOSIT WHERE AMOUNT>2000;

**Output:**

ACTNO CNAME

----- ------------------

101 SUNIL

102 MEHUL

105 PRMOD

108 KRANTI

109 MINU

**11.Display living city of anil;**

**Query:**

SQL> SELECT CITY FROM CUSTOMER WHERE NAME='ANIL';

**Output:**

CITY

----------

CALCUTTA

**Practical 2**

**AIM- Create the below given table and insert the data accordingly**

**Query:**

SQL> CREATE TABLE EMPLOYEE(EMP\_NO NUMBER(3), EMP\_NAME VARCHAR(30), EMP\_SAL NUMBER(8,2), EMP\_COMM NUMBER(6,1) , DEPT\_NO NUMBER(3));

**Output:**

Table created.

**Query:**

SQL> Create Table Job (job\_id varchar(15), job\_title varchar(30), min\_sal number(7,2), max\_sal number(7,2));

**Output:**

Table created.

**Perform following queries**

**(1) Retrieve all data from employee, jobs and deposit.**

**Query:**

SQL>SELECT \* FROM EMPLOYEE;

**Output:**

EMP\_NO EMP\_NAME EMP\_SAL EMP\_COMM DEPT\_NO

---------- ------------------------------ ---------- ---------- ----------

101 Smtih 800 20

102 Snehal 1600 300 25

103 Adama 1100 0 20

104 Aman 3000 15

105 Anita 5000 50000 10

106 Sneha 2450 24500 10

107 Anamika 2975 30

**Query:**

SQL>SELECT \* FROM JOBS;

**Output:**

JOB\_ID JOB\_TITLE MIN\_SAL MAX\_SAL

--------------- ------------------------------ ---------- ----------

IT\_PROG Programmer 4000 10000

MK\_MGR Marketing manager 9000 15000

FI\_MGR Finance manager 8200 12000

FI\_ACC Account 4200 17000

COMP\_OP Computer Operator 1500 3000

**Query:**

SQL>SELECT \* FROM DEPOSIT1;

**Output:**

A\_NO CNAME BNAME AMOUNT A\_DATE

----- --------------- ---------- ---------- ---------

101 Anil andheri 7000 07-JAN-06

102 sunil virar 5000 15-JUL-06

103 jay villeparle 6500 15-MAR-06

104 vijay andheri 8000 17-SEP-06

105 keyur dadar 7500 19-NOV-06

106 mayur borivali 5500 21-DEC-06

6 rows selected.

**(2) Give details of account no. and deposited rupees of customers having account opened between dates 01-01-06 and 25-07-06.**

**Query:**

SQL> select a\_no,amount from deposit1 where a\_date between '01-jan-06' and '25-jul-06';

**Output:**

A\_NO AMOUNT

----- ----------

101 7000

102 5000

103 6500

**(3) Display all jobs with minimum salary is greater than 4000.**

**Query:**

SQL> SELECT JOB\_TITLE FROM JOB WHERE MIN\_SAL>4000;

**Output:**

JOB\_TITLE

------------------------------

Marketing manager

Finance manager

Account

**(4) Display name and salary of employee whose department no is 20. Give alias name to name of employee.**

**Query:**

SQL> SELECT EMP\_NAME AS "name\_of\_employee",EMP\_SAL FROM EMPLOYEE WHERE DEPT\_NO=20;

**Output:**

name\_of\_employee EMP\_SAL

------------------------------ ----------

Smtih 800

Adama 1100

**(5) Display employee no,name and department details of those employee whose department lies in(10,20)**

**Query:**

SQL> SELECT EMP\_NO,EMP\_NAME,DEPT\_NO FROM EMPLOYEE WHERE DEPT\_NO IN (10,20);

**Output:**

EMP\_NO EMP\_NAME DEPT\_NO

101 Smtih 20

103 Adama 20

105 Anita 10

106 Sneha 10

**To study various options of LIKE predicate**

**(1) Display all employee whose name starts with ‘A’ and third letter is ‘a’**

**Query:**

SQL> SELECT EMP\_NAME FROM EMPLOYEE WHERE EMP\_NAME LIKE 'A\_a%';

**Output:**

EMP\_NAME

------------------------------

Adama

Aman

Anamika

**(2) Display name, number and salary of those employees whose name is 5 characters long**

**Query:**

SQL> SELECT EMP\_NAME,EMP\_NO,EMP\_SAL FROM EMPLOYEE WHERE EMP\_NAME LIKE '\_\_\_\_\_';

**Output:**

EMP\_NAME EMP\_NO EMP\_SAL

------------------------------ ---------- ----------

Smtih 101 800

Adama 103 1100

Anita 105 5000

Sneha 106 2450

**(3) Display the non-null values of employees and also employee name**

**Query:**

SQL> SELECT \* FROM EMPLOYEE WHERE EMP\_COMM IS NOT NULL;

**Output:**

EMP\_NO EMP\_NAME EMP\_SAL EMP\_COMM DEPT\_NO

---------- ------------------------------ ---------- ---------- ----------

102 Snehal 1600 300 25

103 Adama 1100 0 20

105 Anita 5000 50000 10

106 Sneha 2450 24500 10

**(4)DISPLAY ALL DETAILS OF EMPLOYEE WHOSE NAME IS 5 CHAR LONG AND WITH ‘a’**

**Query:**

SQL> SELECT \* FROM EMPLOYEE WHERE EMP\_NAME LIKE '\_\_\_\_a';

**Output:**

EMP\_NO EMP\_NAME EMP\_SAL EMP\_COMM DEPT\_NO

---------- ------------------------------ ---------- ---------- ----------

103 Adama 1100 0 20

105 Anita 5000 50000 10

106 Sneha 2450 24500 10

**Practical 3**

**AIM- To Perform various data manipulation commands, aggregate functions and**

**sorting concept on all created tables.**

**(1) List total deposit from deposit.**

**Query:**

SQL> select sum(amount) from deposit;

**Output:**

SUM(AMOUNT)

-----------

28700

**(2) List total loan from karolbagh branch**

**Query:**

SQL> select sum(amount) from borrow where bname='KAROLBAGH';

**Output:**

SUM(AMOUNT)

-----------

28700

**(3) Give maximum loan from branch vrce.**

**Query:**

SQL> SELECT MAX(AMOUNT) FROM BORROW WHERE BNAME='VRCE';

**Output:**

MAX(AMOUNT)

-----------

1000

**(4) Count total number of customers**

**Query:**

SQL> select count(name) from customer;

**Output:**

COUNT(NAME)

-----------

10

**(5)count total number of customer’s cities**

**Query:**

SQL> select count(distinct city) from customer;

**Output:**

COUNT(DISTINCTCITY)

-------------------

7

**(6) Create table supplier from employee with all the columns.**

**Query:**

SQL> CREATE TABLE SUPPLIER AS SELECT \* FROM EMPLOYEE;

**Output:**

Table created.

**(7) Create table sup1 from employee with first two columns.**

**Query:**

SQL> CREATE TABLE SUPP1 AS SELECT EMP\_NO,EMP\_NAME FROM EMPLOYEE;

**Output:**

Table created.

**(8) Create table sup2 from employee with no data**

**Query:**

SQL> CREATE TABLE SUPP2 AS SELECT \* FROM EMPLOYEE WHERE 1=2;

**Output:**

Table created.

**(9) insert the data into sup2 from employee whose second charactere should be ‘n’ and string should be 5 characters long in employee name field.**

**Query:**

SQL> INSERT INTO SUPP2(EMP\_NO,EMP\_NAME,EMP\_SAL,EMP\_COMM,DEPT\_NO) SELECT \* FROM EMPLOYEE WHERE EMP\_NAME LIKE '\_n\_\_\_';

**Output:**

2 rows created.

**(10) Delete all the rows from sup1.**

**Query:**

SQL> DELETE FROM SUPP1;

**Output:**

7 rows deleted.

**(11) Delete the detail of supplier whose sup\_no is 103.**

**Query:**

SQL> DELETE FROM SUPPLIER WHERE EMP\_NO=103;

**Output:**

1 row deleted.

**(12) Rename the table sup2.**

**Query:**

SQL> ALTER TABLE SUPPLIER RENAME TO SUPP2;

**Output:**

Table altered.

**(13) Destroy table sup1 with all the data.**

**Query:**

SQL> DROP TABLE SUPP1;

**Output:**

Table dropped.

**(14) Update the value dept\_no to 10 where second character of emp.name is ‘m’**

**Query:**

SQL> UPDATE EMPLOYEE SET DEPT\_NO=10 WHERE EMP\_NAME LIKE '\_m%';

**Output:**

2 rows updated.

**(15) Update the value of employee name whose employee number is 103**

**Query:**

SQL> UPDATE EMPLOYEE SET EMP\_NAME='HIMALAY' WHERE EMP\_NO=103;

**Output:**

1 row updated.

**Practical 4**

**AIM- To study Single-row functions.**

**(1)Write a query to display the current date. Label the column Date .**

**Query:**

SQL> select sysdate as "Date" from dual;

**Output:**

Date

---------

05-JUL-21

**(2)For each employee, display the employee number, job, salary, and salary increased by 15% and expressed as a whole number. Label the column New Salary**

**Query:**

SQL> select emp\_no,emp\_sal,round(emp\_sal+((emp\_sal\*15)/100)) from employee;

**Output:**

EMP\_NO EMP\_SAL ROUND(EMP\_SAL+((EMP\_SAL\*15)/100))

---------- ---------- ---------------------------------

101 800 920

102 1600 1840

103 1100 1265

104 3000 3450

105 5000 5750

106 2450 2818

107 2975 3421

7 rows selected.

**(3) Modify your query no 4.(2) to add a column that subtracts the old salary from the new salary. Label the column Increase**

**Query:**

SQL> select emp\_no,emp\_sal,round(emp\_sal+((emp\_sal\*15)/100)),round(emp\_sal+(emp\_sal\*0.15))-emp\_sal as increase from employee;

**Output:**

EMP\_NO EMP\_SAL ROUND(EMP\_SAL+((EMP\_SAL\*15)/100)) INCREASE

---------- ---------- --------------------------------- ----------

101 800 920 120

102 1600 1840 240

103 1100 1265 165

104 3000 3450 450

105 5000 5750 750

106 2450 2818 368

107 2975 3421 446

7 rows selected.

**(4)Write a query that displays the employee’s names with the first letter capitalised and all other letters lowercase, and the length of the names, for all employees whose name starts with J, A, or M. Give each column an appropriate label. Sort the results by the employees’ last names.**

**Query:**

SQL> select initcap(emp\_name),length(emp\_name) from employee where emp\_name like 'J%' or emp\_name like'A%' or emp\_name like 'M%';

**Output:**

INITCAP(EMP\_NAME) LENGTH(EMP\_NAME)

------------------------------ ----------------

Adama 5

Aman 4

Anita 5

Anamika 7

**(5) Write a query that produces the following for each employee: <employee last name>earns<salary> monthly**

**Query:**

SQL> select concat(emp\_name,concat(' earns ',concat(emp\_sal,' monthly'))) from employee;

**Output:**

CONCAT(EMP\_NAME,CONCAT('EARNS',CONCAT(EMP\_SAL,'MONTHLY')))

--------------------------------------------------------------------------------

Smtih earns 800 monthly

Snehal earns 1600 monthly

Adama earns 1100 monthly

Aman earns 3000 monthly

Anita earns 5000 monthly

Sneha earns 2450 monthly

Anamika earns 2975 monthly

7 rows selected.

**OR**

**Query:**

SQL> select emp\_name||' earns '||emp\_sal||' monthly' from employee;

**Output:**

EMP\_NAME||'EARNS'||EMP\_SAL||'MONTHLY'

--------------------------------------------------------------------------------

Smtih earns 800 monthly

Snehal earns 1600 monthly

Adama earns 1100 monthly

Aman earns 3000 monthly

Anita earns 5000 monthly

Sneha earns 2450 monthly

Anamika earns 2975 monthly

7 rows selected.

**(7)Display the current date of emp in a format that appears as Seventh of June 1994 12:00:00 AM.**

**Query:**

SQL> select to\_char(sysdate,'fm Ddspth') || ' of ' ||to\_char(sysdate,'fm Month yyyy HH:MM:SS AM') from dual;

**Output:**

TO\_CHAR(SYSDATE,'FMDDSPTH')||'OF'||TO\_CHAR(SYS

----------------------------------------------

Twelfth of July 2021 12:7:0 PM

**(8)Write a query to calculate the annual compensation of all employees (sal+comm.).**

**Query:**

SQL> select emp\_sal+nvl(emp\_comm,0) from employee;

**Output:**

EMP\_SAL+NVL(EMP\_COMM,0)

-----------------------

800

1900

1100

3000

55000

26950

2975

7 rows selected.

**Practical 5**

**AIM- Displaying data from Multiple Tables (join)**

**(1) Give details of customers ANIL.**

**Query:**

SQL> select \* from deposit d,customer c,branch b,borrow b1 where d.cname='ANIL' AND d.bname=b.bname AND d.cname=b1.cname AND d.bname=b1.bname AND c.cname=d.cname;

**Output:**

ACTNO CNAME BNAME AMOUNT ADATE CNAME CITY BNAME CITY LOANN CNAME BNAME AMOUNT

100 ANIL VRCE 1000 01-MAR-95 ANIL CALCUTTA VRCE NAGPUR 201 ANIL VRCE 1000

**(2) Give name of customer who are borrowers and depositors and having living city Nagpur**

**Query:**

SQL> select d.cname from deposit d, borrow b,customer c where d.cname=b.cname and b.cname=d.cname and c.cname=d.cname and CITY='NAGPUR';

**Output:**

CNAME

------------------

MADHURI

**(3) Give city as their city name of customers having same living branch.**

**Query:**

SQL> select c.city from deposit d ,customer c ,branch b where d.cname=c.cname and b.bname=d.bname and c.city=b.city;

**Output:**

CITY

----------

BOMBAY

**(4) Write a query to display the name, department number, and department name for all employees.**

**Query: SELECT E.EMP\_NAME,D.DNAME,D.DEPT\_NO FROM EMPLOYEE E INNER JOIN DEPT D ON E.DEPT\_NO=D.DEPT\_NO;  
Output:   
                 EMP\_NAME        DNAME        DEPT\_NO  
                 Smith                    Teaching      20  
                 Snehal                  Marketing    25  
                 Aman                    Computer     15**

**(5) Display Borrower Details Who Is Living In Nagpur**

**Query;**

SQL> select \* from borrow b,customer c where c.cname=b.cname and c.city='NAGPUR' ;

**Output:**

LOANN CNAME BNAME AMOUNT CNAME CITY

321 MADHURI ANDHERI 2000 MADHURI NAGPUR

375 PRMOD VIRAR 8000 PRMOD NAGPUR

**(6) Display Total Deposited Amount Of Vrce Branch**

**Query:**

SQL> SELECT SUM(AMOUNT) FROM DEPOSIT WHERE BNAME='VRCE';

**Output:**

SUM(AMOUNT)

-----------

1000

**(7) Display Total Number Of Customer Whose Branch City Is Bombay**

**Query:**

SQL> SELECT COUNT(CNAME) FROM BRANCH B,DEPOSIT D WHERE B.BNAME=D.BNAME AND CITY='BOMBAY';

**Output:**

COUNT(CNAME)

------------

3

**Practical 6**

**AIM- To apply the concept of Aggregating Data using Group functions.**

**(1) List total deposit of customer having account date after 1-jan-96.**

**Query:​ SELECT SUM(AMOUNT) "AMOUNT" FROM DEPOSIT WHERE ADATE > 'A-JAN-96';  
Output:       
                AMOUNT  
                 10000**

**(2) List total deposit of customers living in city Nagpur.**

**Query:** select sum(amount) from customers c,deposit2 d where c.name=d.cname and city='NAGPUR';

**Output:    
                AMOUNT  
                 4200**

**(3) List maximum deposit of customers living in bombay.**

**Query:**

select max(amount) from customers c,deposit2 d where c.name=d.cname and city='BOMBAY';

**Output:**

**AMOUNT  
                 5000**

**(4) Display the highest, lowest, sum, and average salary of all employees. Label the columns Maximum, Minimum, Sum, and Average, respectively. Round your results to the nearest whole number.**

**Query**:

select max(emp\_sal) as maximum ,min(emp\_sal) as minimum ,sum(emp\_sal) as Sum ,round(avg(emp\_sal)) as average from employee;

**Output:**  
**Average        Maximum        Minimum        Sum  
                  2418               5000                  800                  16925**

**(5) Write a query that displays the difference between the highest and lowest salaries. Label the column DIFFERENCE.**

**Query:**

select max(emp\_sal) - min(emp\_sal) as DIFFERENCE from employee;

**Output:  
                    DIFFERENCE  
                     4200**

**(6) Create a query that will display the total number of employees and, of that total, the number of employees hired in 1995, 1996, 1997, and 1998**

Query: select count(cname) from deposit where to\_char(adate,'fm YYYY') in (1995,1996,1997,1998);

**Output:**

**COUNT  
                  2**

**(7) Find the average salaries for each department without displaying the respective department numbers.**

**Query:**

SQL> select avg(emp\_sal) from employee group by dept\_no;

**Output:**

AVG(EMP\_SAL)

------------

1600

2975

950

3000

3725

**(8) Write a query to display the total salary being paid to each job title, within each department.**

**Query:**

SQL> select sum(min\_sal) from job group by job\_title;

**Output:**

SUM(MIN\_SAL)

------------

4000

4200

1500

8200

9000

**(9) Find the average salaries > 2000 for each department without displaying the respective department numbers.**

**Query:**

SQL> select avg(emp\_sal) from employee having avg(emp\_sal)>2000 group by dept\_no;

**Output:**

AVG(EMP\_SAL)

------------

2975

3000

3725

**(10) Display the job and total salary for each job with a total salary amount exceeding 3000, in which excludes president and sorts the list by the total salary.**

**Query:**

SQL> select sum(min\_sal), job\_title from job group by job\_title having sum(min\_sal)>3000 and job\_title!='president' order by sum(min\_sal) asc;

**Output:**

SUM(MIN\_SAL) JOB\_TITLE

------------ ------------------------------

4000 Programmer

4200 Account

8200 Finance manager

9000 Marketing manager

**(11) List the branches having sum of deposit more than 5000 and located in city bombay**

**Query:**

SQL> select d.bname from branch b ,deposit d WHERE D.BNAME=B.BNAME AND CITY='BOMBAY' GROUP BY D.BNAME HAVING SUM(d.amount)>5000 ;

**Output:**

BNAME

------------------

POWAI

**(12)Display employee name who earns max salary**

**Query:**

SQL> select emp\_name from employee where emp\_sal=(select max(emp\_sal) from employee);

**Output:**

EMP\_NAME

------------------------------

Anita

**(13)Display 2nd highest salary of employee**

**Query:**

SQL> select max(emp\_sal) from employee where emp\_sal!=(select Max(emp\_sal) from employee);

**Output:**

MAX(EMP\_SAL)

------------

3000

**Practical 7**

**AIM- To solve queries using the concept of sub query.**

**(1) Write a query to display the last name and hire date of any employee in the same department as Smtih. Exclude Smith**

SQL> select emp\_name from employee where dept\_no in (select dept\_no from employee where emp\_name='Smtih');

EMP\_NAME

------------------------------

Adama

Smtih

**(2) Give name of customers who are depositors having same branch city of mr. sunil.**

SQL> select d.cname from deposit d,branch b where d.bname=b.bname and city in (select b.city from deposit d,branch b where d.bname=b.bname and d.cname='SUNIL');

CNAME

------------------

ANIL

SUNIL

**3) Give deposit details and loan details of customer in same city where pramod is living.**

SQL> SELECT \* FROM DEPOSIT D FULL OUTER JOIN BORROW B ON B.BNAME=D.BNAME AND B.CNAME=D.CNAME AND B.CNAME IN (SELECT CNAME FROM CUSTOMER WHERE CITY =(SELECT CITY FROM CUSTOMER WHERE CNAME='PRMOD'));

ACTNO CNAME BNAME AMOUNT ADATE LOANN CNAME BNAME AMOUNT

----- ------------------ ------------------ ---------- --------- ----- ------------------ ------------------ ----------

100 ANIL VRCE 1000 01-MAR-95

101 SUNIL AJNI 5000 04-JAN-96

102 MEHUL KAROLBAGH 3500 17-NOV-95

104 MADHURI CHANDI 1200 17-DEC-95

105 PRMOD M.G.ROAD 3000 27-MAR-96

106 SANDIP ANDHERI 2000 31-MAR-96

107 SHIVANI VIRAR 1000 05-SEP-95

108 KRANTI NEHRU PLACE 5000 02-JUL-95

109 MINU POWAI 7000 10-AUG-95

321 MADHURI ANDHERI

2000

206 MEHUL AJNI

5000

ACTNO CNAME BNAME AMOUNT ADATE LOANN CNAME BNAME

AMOUNT

----- ------------------ ------------------ ---------- --------- ----- ------------------ ------------------ ----------

375 PRMOD VIRAR

8000

481 KRANTI NEHRU PLACE

3000

201 ANIL VRCE

1000

311 SUNIL DHARAMPETH

3000

15 rows selected.

**(4) Create a query to display the employee numbers and last names of all employees who earn more than the average salary. Sort the results in ascending order of salary.**

SQL> select emp\_no,emp\_name from employee where emp\_sal>(select avg(emp\_sal) from employee) order by emp\_sal asc;

EMP\_NO EMP\_NAME

---------- ------------------------------

106 Sneha

107 Anamika

104 Aman

105 Anita

**(5) Give names of depositors having same living city as mr. anil and having deposit amount greater than 2000**

SQL> SELECT C.CNAME FROM CUSTOMER C,DEPOSIT D WHERE C.CNAME=D.CNAME AND CITY IN (SELECT CITY FROM CUSTOMER WHERE CNAME='ANIL') AND AMOUNT>2000;

no rows selected

**(6)Display employee name who earns max salary**

SQL> select emp\_name from employee where emp\_sal=(select max(emp\_sal) from employee);

EMP\_NAME

------------------------------

Anita

**(7)Display 2nd highest salary of employee**

SQL> select max(emp\_sal) from employee where emp\_sal!=(select Max(emp\_sal) from employee);

MAX(EMP\_SAL)

------------

3000

**(8) List the name of branch having highest number of depositors.**

select bname from deposit group by bname having count(bname) In (select max(count(bname)) from deposit group by bname);

BNAME

----------

VRCE

AJNI

KAROLBAGH

VIRAR

POWAI

CHANDI

ANDHERI

NEHRUPLACE

N.G.ROAD

**(9) Give the name of cities where in which the maximum numbers of branches are located.**

select city from branch group by city having count(city) = (select max(count(city)) from branch group by city);

CITY

------------------

NAGPUR

DELHI

BOMBAY

**(10) Give name of customers living in same city where maximum depositors are located.**

select cname from customers where city = (select city from customers group by city having count(city) = (select max(count(city)) from customers group by city));

CNAME

--------------------

SHIVANI

KRANTI

NAREN

**Practical 8**

**AIM: Manipulating Data**

**(1) Give 10% interest to all depositors.**

Query: UPDATE DEPOSIT SET AMOUNT = AMOUNT + (AMOUNT\*10/100);

Output: 0 rows updated.

**(2) Give 10% interest to all depositors having branch vrce**

Query: UPDATE DEPOSITE SET AMMOUNT = AMMOUNT + (AMMOUNT\*10/100) WHEREB\_NAME =’vrce’;

Output:1 rows updated.

**(3) Give 10% interest to all depositors living in nagpur and having branch city bombay.**

Query: UPDATE DEPOSITE SET AMMOUNT = AMMOUNT + (AMMOUNT\*10/100) WHERE C\_NAME IN (SELECT CNAME FROM CUSTOMER12 WHERE CITY=’NAGPUR’) AND B\_NAME IN(SELECT BNAME FROM BRANCH WHERE CITY=’BOMBAY’);

Output: 0 rows updated.

**(4) Write a query which changes the department number of all employees with empno 105 job to employee 201 current department number.**

Query: UPDATE EMPLOYEE SET EMP\_NO = 201 WHERE EMP\_NO = 105;

Output:

EMP\_NO EMP\_NAME EMP\_COM

201 Anita 50000

**(5) Transfer 10 Rs from account of anil to sunil if both are having same branch.**

**Query:**UPDATE DEPOSIT SET AMOUNT =AMOUNT -10 WHERE C\_NAME

=’ANIL’ AND B\_NAME IN (SELECT D1.B\_NAME FROM DEPOSIT D1

WHERE D1.C\_NAME = ’SUNIL’ );

**Output:**

0 rows updated.

**Query:**UPDATE DEPOSIT SET AMOUNT=AMOUNT+10 WHERE

C\_NAME=’SUNIL’ AND B\_NAME IN (SELECT D2.B\_NAME FROM

DEPOSIT D2 WHERE D2.C\_NAME=’ANIL’);

**Output:** 0 rows updated.

**(6) Give 100 Rs more to all depositors if they are maximum depositors in their respective branch.**

Query: UPDATE DEPOSITE SET AMMOUNT = AMMOUNT + 100 WHERE C\_NAME IN (SELECT D1.C\_NAME FROM DEPOSITE D1 GROUP BY D1.B\_NAME HAVING AVG(D1.AMMOUNT) > = ALL (SELECT MAX(D2.AMMOUNT) FROM DEPOSITE D2 WHERE D1.B\_NAME = UP BY D2.B\_NAME));

Output: 0 rows updated.

**(7) Delete depositors of branches having number of customers between 1 to 3.**

Query: DELETE FROM DEPOSITE WHERE C\_NAME IN (SELECT D1.C\_NAME D1 GROUP BY D1.B\_NAME HAVING COUNT(D1.C\_NAME) BETWEEN 1 AND 3);

Output:Table dropped.

0.89 seconds

**(8) Delete deposit of ANIL.**

Query: DELETE FROM DEPOSIT WHERE CNAME='ANIL';

Output: 1 row deleted.

**(9) Delete borrower of branches having average loan less than 1001.**

Query: DELETE FROM BORROW WHERE AMOUNT < 1001;

Output: 0 row deleted.

**Practical 9**

**AIM:-To apply the concept of security and privileges.**

This chapter describes PointBase security and privileges. Schemas are an integral part of security in PointBase. When creating a PointBase user, they do not have any access privileges to schemas or other data objects within the database. The PointBase RDBMS only permits the schema owner to grant privileges to the schema and data objects within the schema. The schema owner can grant privileges to the following data objects in the schema:

* Tables
* Columns
* SQL Procedures and Functions

Table 1 describes the privileges that the schema owner can grant users for tables and columns:

|  |  |
| --- | --- |
| Table 1 : User Privileges for Tables and Columns | |
| Privilege  Statements | Privilege  Description |
| DELETE | Allows a user to delete rows from tables within the schema |
| INSERT | Allows a user to insert rows of data into tables within the schema |
| REFERENCES | Allows a user to set up references to primary keys within the schema |
| SELECT | Allows a user to select rows from tables within the schema |
| TRIGGER | Allows a user to create triggers on tables within the schema |
| UPDATE | Allows a user to update rows in tables within the schema |
| EXECUTE | Allows users to execute functions or stored procedures within the schema |

Granting and Revoking Privileges

When a PointBase database is first created the only user is the default user PUBLIC with a password of PUBLIC. The PUBLIC user owns the default PUBLIC schema. For security reasons, PointBase does not recommend using this schema to store sensitive data. Like any PointBase user, PUBLIC must be granted the appropriate privileges to access data objects in schema owned by other users.

The PUBLIC user can be used initially to create new users and new schema. The PUBLIC user will own any new schema that it creates unless otherwise specified during schema creation. New users are then able to create their own new schema and users, and grant appropriate privileges on schema that they own. All new users must be granted privileges to use the PUBLIC schema if this is required.

To grant the ability for a user to pass a privilege on to other users once granted, you must specify the optional WITH GRANT OPTION qualifier when granting the privilege.

GRANT Statement Syntax

GRANT privilege-list

ON object

TO user-list [ WITH GRANT OPTION ]

Use the GRANT statement to grant privileges on a data object. The following describes the GRANT statement syntax. Privilege-list Syntax privilege [ , privilege [ , privilege ]...] | ALL PRIVILEGES

Privilege Syntax

SELECT [ ( column-name [ , column-name ]...)]

| DELETE

| INSERT [ ( column-name [ , column-name ]...)]

| UPDATE [ ( column-name [ , column-name ]...)]

| REFERENCES [ ( column-name [ , column-name ]...)]

| TRIGGER [ ( column-name [ , column-name ]...)]

| EXECUTE

Usage Notes

* If you do not include one or more of these privileges in the GRANT statement, an error will be raised.
* If the optional column-names are not specified for the SELECT, INSERT, UPDATE, REFERENCES and TRIGGER privileges, the GRANT is applied to every column in the table to which the grant is applied.
* If you execute a GRANT statement that contains privileges that you don't have or for which you do not have the right to grant, then PointBase raises an error.

Object Syntax

[ TABLE ] table-name

|SPECIFIC routine\_type specific\_routine-name

|routine\_type routine\_name (parameter\_types\_list)

[ TRIGGER ] trigger-name

Usage Notes

* If you grant a privilege on an SQL Function or Procedure, then the user can only EXECUTE that SQL Function or Procedure. The user cannot access tables that the SQL Function or Procedure uses. User-list Syntax user [ , user ]... [WITH GRANT OPTION] | PUBLIC

Usage Notes

* If you do not specify WITH GRANT OPTION, the user cannot pass the same privilege on to others. However, if you do specify WITH GRANT OPTION, you have given the user permission to pass on the privilege to other users.
* Granting a privilege to the user PUBLIC only grants the privilege to the default PointBase PUBLIC user and is not the same as granting a global privilege to all users.
* If you grant a privilege with the optional WITH GRANT OPTION and then grant the same privilege without this option (without first revoking the original privilege) the user retains the WITH GRANT OPTION.

Examples

* The following statement grants the SELECT privilege on the CUSTOMER\_TBL table to the user MARKETING\_MGR.

GRANT SELECT

ON customer\_tbl

TO marketing\_mgr;

* The following GRANT statement allows the user FINANCIAL\_MGR to delete, insert and update rows from the DISCOUNT\_CODE\_TBL table; it also allows this user to grant the same privileges to others.

GRANT DELETE,INSERT,UPDATE

ON discount\_code\_tbl

TO financial\_mgr

WITH GRANT OPTION;

* The following GRANT statement allows the user HR\_MGR to have ALL PRIVILEGES on the table SALES\_REP\_DATA\_TBL. However, the user HR\_MGR will only be granted privileges that the user granting the privileges has the right to grant. For example, if the user granting the privileges does not have the right to grant DELETE privileges, the HR\_MGR will not have the delete privilege.

GRANT ALL PRIVILEGES

ON sales\_rep\_data\_tbl

TO hr\_mgr

REVOKE Statement Syntax

REVOKE [ GRANT OPTION FOR ] privilege\_list

ON object

FROM user\_name [ RESTRICT | CASCADE ]

The REVOKE statement takes privileges away from users. The arguments are similar to the GRANT statement. The major difference is the additional RESTRICT or CASCADE keyword and the GRANT OPTION FOR clause. The following describes the optional clauses GRANT OPTION FOR and RESTRICT or CASCADE.

NOTE: If none of the privileges that you are trying to revoke actually exist, an error is raised.

RESTRICT | CASCADE

If you use RESTRICT keyword, the privilege will be revoked only from the specified user. If the specified user granted had the WITH GRANT OPTION and granted the same privilege to other users, they will retain the privilege.

If you use CASCADE, it will revoke the privilege and any dependent privileges as a result of your grant. A dependent privilege is one that could exist, if you granted the privilege that you're trying to revoke, which is what you are trying to achieve as a result of your REVOKE statement.

If the optional RESTRICT or CASCADE keywords are not used, PointBase uses RESTRICT by default.

GRANT OPTION FOR

If he optional GRANT OPTION FOR clause is used, the WITH GRANT OPTION right is revoked. The actual privilege itself is not revoked. the GRANT OPTION is revoked.

CASCADE and RESTRICT may be used in the same way as a normal REVOKE statement.

**Practical 10**

**AIM: To study Transaction control commands**.

TCL Commands in SQL- Transaction Control Language Examples: Transaction Control Language can be defined as the portion of a database language used for maintaining consistency of the database and managing transactions in database. A set of SQL statements that are co-related logically and executed on the data stored in the table is known as transaction. In this tutorial, you will learn different TCL Commands in SQL with examples and differences between them.

1. Commit Command

2. Rollback Command

3. Savepoint Command

**TCL Commands in SQL- Transaction Control Language Examples**

The modifications made by the DML commands are managed by using TCL commands. Additionally, it makes the statements to grouped together into logical transactions.

**TCL Commands**

**There are three commands that come under the TCL:**

**1. Commit**

The main use of Commit command is to make the transaction permanent. If there is a need for any transaction to be done in the database that transaction permanent through commit command. Here is the general syntax for the Commit command:

COMMIT;

For Example

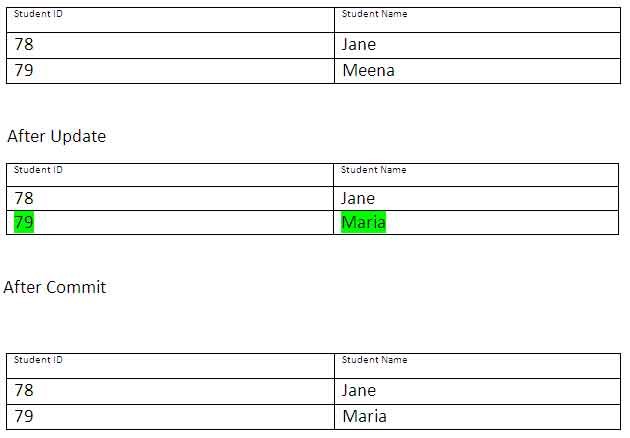
UPDATE STUDENT SET STUDENT\_NAME = ‘Maria’ WHERE STUDENT\_NAME =

‘Meena’;

COMMIT;

By using the above set of instructions, you can update the wrong student name by the correct one and save it permanently in the database. The update transaction gets completed when commit is used. If commit is not used, then there will be lock on ‘Meena’ record till the rollback or commit is issued.

Now have a look at the below diagram where ‘Meena’ is updated and there is a lock on her record. The updated value is permanently saved in the database after the use of commit and lock is released.



**2. Rollback**

Using this command, the database can be restored to the last committed state. Additionally, it is also used with savepoint command for jumping to a savepoint in a transaction.

The general syntax for the Rollback command is mentioned below:

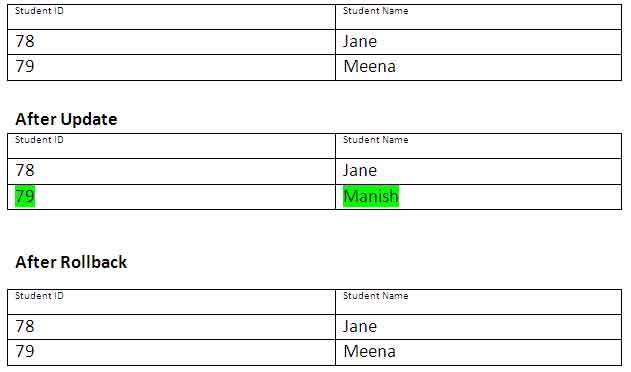
Rollback to savepoint-name;

For example

UPDATE STUDENT SET STUDENT\_NAME = ‘Manish’ WHERE STUDENT\_NAME

= ‘Meena’;

ROLLBACK;

This command is used when the user realizes that he/she has updated the wrong information after the student name and wants to undo this update. The users can issues ROLLBACK command and then undo the update. Have a look at the below tables to know better about the implementation of this command. 

**3. Savepoint**

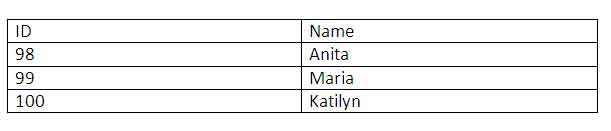
The main use of the Savepoint command is to save a transaction temporarily. This way users can rollback to the point whenever it is needed.

The general syntax for the savepoint command is mentioned below:

savepoint savepoint-name;

For Example

Following is the table of a school class



Use some SQL queries on the above table and then watch the results

INSERT into CLASS VALUES (101, ‘Rahul);

Commit;

UPDATE CLASS SET NAME= ‘Tyler’ where id= 101

SAVEPOINT A;

INSERT INTO CLASS VALUES (102, ‘Zack’);

Savepoint B;

INSERT INTO CLASS VALUES (103, ‘Bruno’)

Savepoint C;

Select \* from Class;

The result will look like



Now rollback to savepoint B

Rollback to B;

SELECT \* from Class;



Now rollback to savepoint A rollback to A;

SELECT \* from class;

