# A Helpful Hand

DEPARTMENT OF INFORMATION TECHNOLOGY,
INDIRA GANDHI INSTITUTE OF TECHNOLOGY (IGIT)

## WEEK-1

## How to Write and execute sql, pl/sql commands/programs:

1). Open your oracle application by the following navigation Start->all programs->Oracle Database ->Run SQL.
2). You will be asked for user name and password.
3). Upon successful login you will get SQL prompt (SQL>).
In two ways you can write your programs:
a) directly at SQL prompt (or)
b) in sql editor.

If you type your programs at sql prompt then screen will look like follow:
SQL> SELECT ename,empno,
2 sal from
3 emp;
where 2 and 3 are the line numbers and rest is the command.

To execute above program/command you have to press '/' then enter. Here editing the program is somewhat difficult; if you want to edit the previous command then you have to open sql editor (by default it displays the sql buffer contents). By giving 'ed' at sql prompt.(this is what I mentioned as a second method to type/enter the program).

To execute the program which is saved; do the following
SQL> @ programname.sql (or)
SQL> Run programname.sql

To save the day`s session; do the following
SQL>commit;

### Different types of commands in SQL:

A).**DDL commands: -** To create a database objects
B).**DML commands: -** To manipulate data of a database objects
C).**DQL command: -** To retrieve the data from a database.
D).**DCL/DTL commands: -** To control the data of a database…

### DDL commands:

**1. The Create Table Command: - I**t defines each column of the table uniquely. Each column has minimum of three attributes, a name, data type and size.

**Syntax: Create table** <table name> (<col1> <datatype> (<size>), <col2> <datatype><size>));
**Ex:** create table emp(empno number(4) primary key, ename char(10));

**2. Modifying the structure of tables.**
a) add new columns

**Syntax: Alter table** <tablename> add(<new col><datatype(size),<new col>datatype(size));
**Ex:** alter table emp add(sal number(7,2));

**3. Dropping a column from a table.**

**Syntax:** Alter table <tablename> drop column <col>;
**Ex:** alter table emp drop column sal;

**4. Modifying existing columns.**

**Syntax:** Alter table <tablename> modify(<col><newdatatype>(<newsize>));
**Ex:** alter table emp modify(ename varchar2(15));

**5. Renaming the tables**

**Syntax: Rename** <oldtable> to <new table>;
**Ex:** rename emp to emp1;

**6. Truncating the tables.**

**Syntax: Truncate table** <tablename>;
**Ex:** trunc table emp1;

**7. Destroying tables.**

**Syntax: Drop table** <tablename>;
**Ex:** drop table emp;

**DML commands:**

**8. Inserting Data into Tables: -** Once a table is created the most natural thing to do is load this table with data to be manipulated later.

**Syntax 1:** insert into <tablename> (<col1>,<col2>…..<col n>) values(<val 1>, <val 2>…….<val n>);
**Syntax 2:** insert into <tablename> values(&<col1>,&<col2>……,&<col n>);
**Syntax 3:** insert into <tablename> values(<val 1>,<val 2>……..,<val n>);

**Ex 1:** Insert into skc (sname,rollno,class,dob,fee_paid) values('sri','104B','cse','27-feb-05',10000.00);

**Ex 2:** insert into skc values(&sname,&roll no,&class); enter sname:'sri' enter roll no:'104B' enter class:'cse' 1 row created.

**Ex 3:** insert into skc values('sri','104B',cse','27-feb-05',10000.00);

**9. Delete operations.**

**a)** remove all rows

**Syntax:** delete from <tablename>;

**b)** removal of a specified row/s

**Syntax:** delete from <tablename> where <condition>;

**10. Updating the contents of a table.**

**a)** updating all rows

**Syntax:** Update <tablename> set <col>=<exp>,<col>=<exp>;

**b)** updating seleted records.

**Syntax:** Update <tablename> set <col>=<exp>,<col>=<exp> where <condition>;

**11. Types of data constrains.**

**a)** not null constraint at column level.

**Syntax:** <col><datatype>(size)not null

**b)** unique constraint

**Syntax:** Unique constraint at column level.
<col><datatype>(size)unique;

**c)** unique constraint at table level:

**Syntax:** Create table
tablename(col=format,col=format,unique(<col1>,<col2>);

**d)** primary key constraint at column level

**Syntax:**
<col><datatype>(size)primary key;

**e)** primary key constraint at table level.

**Syntax:** Create table tablename(col=format,col=format
primary key(col1>,<col2>);

**f)** foreign key constraint at column level.

**Syntax:** <col><datatype>(size>) references <tablename>[<col>];

**g)** foreign key constraint at table level

**Syntax:** foreign key(<col>[,<col>]) references <tablename>[(<col>,<col>)

**h)** check constraint

check constraint constraint at column level.

**Syntax:** <col><datatype>(size) check(<logical expression>)

**i)** check constraint constraint at table level.

**Syntax:** check(<logical expression>)

**DQL Commands:**

**12. Viewing data in the tables**: - once data has been inserted into a table, the next most logical operation would be to view what has been inserted.

**a)** all rows and all columns

**Syntax:** Select <col> to <col n> from tablename;
Select * from tablename;

**13. Filtering table data**: - while viewing data from a table, it is rare that all the data from table will be required each time. Hence, sql must give us a method of filtering out data that is not required data.

**a)** Selected columns and all rows:
**Syntax:** select <col1>,<col2> from <tablename>;

**b)** selected rows and all columns:
**Syntax:** select * from <tablename> where <condition>;

**c)** selected columns and selected rows
**Syntax:** select <col1>,<col2> from <tablename> where<condition>;

**14. Sorting data in a table.**

**Syntax:** Select * from <tablename> order by <col1>,<col2> <[sortorder]>;

**DCL commands:**

Oracle provides extensive feature in order to safeguard information stored in its tables from unauthoraised viewing and damage.The rights that allow the user of some or all oracle resources on the server are called privileges.

**a)** Grant privileges using the GRANT statement. The grant statement provides various types of access to database objects such as tables, views and sequences and so on.

**Syntax:** GRANT <object privileges> ON <objectname> TO<username> [WITH GRANT OPTION];

**b)** Reoke permissions using the REVOKE statement:
The REVOKE statement is used to deny the Grant given on an object.

**Syntax:** REVOKE<object privilege> ON FROM<user name>;

**EXAMPLE 1:**

**CREATING A STUDENT RELATION TABLE WITH ALL DATATYPES:**

SQL> create table student252(sid number(5), sname varchar(20), sbranch char(5), dob date, spercent number(3,2));

Table created.

SQL> desc student252;

| Name | Null? | Type |
|---|---|---|
| SID | | NUMBER(5) |
| SNAME | | VARCHAR2(20) |
| SBRANCH | | CHAR(5) |
| DOB | | DATE |
| SPERCENT | | NUMBER(5,2) |

**INSERT THE RECORDS INTO STUDENT RELATION:**

**METHOD 1:** SQL>Insert into Student252(sid,sname,sbranch,dob,spercent) values(104,'sri',,'cse','27- feb-05',70);
1 row created.

**METHOD 2:** SQL>Insert into Student252 values(104,'sri',,'cse','27-feb-05',70);
1 row created.

**METHOD 3:** SQL>Insert into Student252(sid,sname,sbranch,dob,spercent) values(&sid, &sname,&sbranch,&dob,&spercent);
1 row created.

**METHOD 4:** SQL>Insert into Student252(sid,sname,sbranch,dob,spercent) values(&sid, '&sname','&sbranch','&dob',&spercent);
1 row created.

**QUERY THE TABLE VALUES:**

SQL> select * from student252;

| SID | SNAME | SBRANCH | DOB | SPERCENT |
|---|---|---|---|---|
| 130 | ravi | it | 30-1-95 | 60 |
| 131 | teja | cse | 21-07-87 | 55 |
| 129 | kiran | mech | 12-05-92 | 60 |
| 104 | sri | cse | 30-07-90 | 70 |
| 133 | sajith | eee | 12-06-89 | 55 |
| 137 | ram | ece | 07-07-85 | 40 |

**MODIFYING THE STRUCTURE OF TABLE**

**ADDING A NEW COLUMN**

SQL> ALTER TABLE Emp252 ADD (age number(3), phno number(10));

Table altered.

**MODIFYING EXISTING COLUMN**

SQL> ALTER TABLE Emp252 MODIFY (phno varchar(20));

Table altered.

**DROPING A COLUMN**

SQL> ALTER TABLE Emp252 DROP COLUMN phno;

Table altered.

**QUERY FOR THE TABLE VALUES**
SQL> SELECT * FROM Emp252;

**UPDATING ENTIRE COLUMN**

SQL> UPDATE Emp252 SET age=18;

**RENAMING THE TABLE:**

SQL> RENAME Emp252 TO Emp252;

Table renamed.

**SELECTING THE TABLE VALUES**
SQL> SELECT * FROM Emp252;

## WEEK 2

**CREATING A TABLE WITH KEY CONSTRAINTS**

**Example 1**

**CREATING A TABLE WITH 'UNIQUE ', 'NOT NULL', 'CHECK' AND 'DEFAULT' CONSTRAINT:**

SQL> CREATE TABLE emp252 (eid NUMBER(5) UNIQUE, ename VARCHAR(10) DEFAULT('UNKNOWN'), age NUMBER(3) NOT NULL, esal NUMBER(7) CHECK(esal > 1000));

Table created.

***INSERTING RECORDS INTO TABLE:***

SQL> INSERT INTO emp252 VALUES (&eid, &ename, &age, &esal);
Enter value for eid: 1
Enter value for ename: 'ravi'
Enter value for age: 18
Enter value for esal: 10000
old 1: INSERT INTO emp252 VALUES (&eid, &ename, &age, &esal)
new 1: INSERT INTO emp252 VALUES (1, 'ravi', 18, 10000)

1 row created.

SQL> /
Enter value for eid: 2
Enter value for ename: 'teja'
Enter value for age: 18
Enter value for esal: 20000
old 1: INSERT INTO emp252 VALUES (&eid, &ename, &age, &esal)
new 1: INSERT INTO emp252 VALUES (2, 'teja', 18, 20000)

1 row created.

SQL> /
Enter value for eid: 3
Enter value for ename: 'kiran'
Enter value for age: 19
Enter value for esal: 25000
old 1: INSERT INTO emp252 VALUES (&eid, &ename, &age, &esal)
new 1: INSERT INTO emp252 VALUES (3, 'kiran', 19, 25000)

1 row created.

SQL> /
Enter value for eid: 4
Enter value for ename: 'srinivas'
Enter value for age: 19

```
Enter value for esal: 30000
old   1: INSERT INTO emp252 VALUES (&eid, &ename, &age, &esal)
new   1: INSERT INTO emp252 VALUES (4, 'srinivas', 19, 30000)

1 row created.

SQL> /
Enter value for eid: 1
Enter value for ename: 'alan'
Enter value for age: 19
Enter value for esal: 29000
old   1: INSERT INTO emp252 VALUES (&eid, &ename, &age, &esal)
new   1: INSERT INTO emp252 VALUES (1, 'alan', 19, 29000)
INSERT INTO emp252 VALUES (1, 'alan', 19, 29000)
[SHOWING AN ERROR WHILE VIOLATING UNIQUE KEY CONSTRAINT]
*
ERROR at line 1:
ORA-00001: unique constraint (SYSTEM.SYS_C003875) violated

SQL> /
Enter value for eid: 7
Enter value for ename: 'dravid'
Enter value for age: null
Enter value for esal: 100000
old   1: INSERT INTO emp252 VALUES (&eid, &ename, &age, &esal)
new   1: INSERT INTO emp252 VALUES (7, 'dravid', null, 100000)
INSERT INTO emp252 VALUES (7, 'dravid', null, 100000)
 [SHOWING AN ERROR AS NOT NULL KEY CONSTRAINT IS VIOLATED]
*
ERROR at line 1:
ORA-01400: cannot insert NULL into ("SYSTEM"."EMP230"."AGE")


SQL> /
Enter value for eid: 8
Enter value for ename: 'sachin'
Enter value for age: 35
Enter value for esal: 100
old   1: INSERT INTO emp252 VALUES (&eid, &ename, &age, &esal)
new   1: INSERT INTO emp252 VALUES (8, 'sachin', 35, 100)
INSERT INTO emp252 VALUES (8, 'sachin', 35, 100)
*
```

***[NOT ALLOWING AS IT VOILATES CHECK CONSTRAINT FOR esal > 1000 VALUE]***

```
ERROR at line 1:
ORA-02290: check constraint (SYSTEM.SYS_C003874) violated
```

**Example 2**

***CREATING A TABLE WITH 'PRIMARY KEY' CONSTRAINT:***

```
SQL> CREATE TABLE mdept252 (dno NUMBER(5),  dname CHAR(10),  dloc
VARCHAR(10),  PRIMARY KEY (dno));

Table created.

SQL> desc mdept252;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DNO                                       NOT NULL NUMBER(5)
 DNAME                                              CHAR(10)
 DLOC                                               VARCHAR2(10)
```

***INSERTING RECORDS INTO MASTER DEPARTMENT TABLE:***

```
SQL> INSERT INTO mdept252 VALUES (&dno, &dname, &dloc);
Enter value for dno: 1
Enter value for dname: 'ravi'
Enter value for dloc: 'hyd'
old   1: INSERT INTO mdept252 VALUES (&dno, &dname, &dloc)
new   1: INSERT INTO mdept252 VALUES (1, 'ravi', 'hyd')

1 row created.

SQL> /
Enter value for dno: 1
Enter value for dname: 'teja'
Enter value for dloc: 'sec'
old   1: INSERT INTO mdept252 VALUES (&dno, &dname, &dloc)
new   1: INSERT INTO mdept252 VALUES (1, 'teja', 'sec')
INSERT INTO mdept252 VALUES (1, 'teja', 'sec')
*
ERROR at line 1:
ORA-00001: unique constraint (SYSTEM.SYS_C003876) violated
```

```
SQL> /
Enter value for dno: null
Enter value for dname: 'sajithulhuq'
Enter value for dloc: 'kmm'
old   1: INSERT INTO mdept252 VALUES (&dno, &dname, &dloc)
new   1: INSERT INTO mdept252 VALUES (null, 'sajithulhuq', 'kmm')
INSERT INTO mdept252 VALUES (null, 'sajithulhuq', 'kmm')
                             *
ERROR at line 1:
ORA-01400: cannot insert NULL into ("SYSTEM"."MDEPT230"."DNO")
```

***ADDING A PRIMARY KEY TO AN EXISTING TABLE:***

```
SQL> ALTER TABLE student252 ADD PRIMARY KEY (sid);

Table altered.

SQL> ALTER TABLE emp252 ADD PRIMARY KEY (eid);
ALTER TABLE emp252 ADD PRIMARY KEY (eid)
                       *
```

***[ GIVING AN ERROR AS ONE TABLE CAN HAVE A SINGLE PRIMARY KEY AT COLUMN LAVEL]***

```
ERROR at line 1:
ORA-02261: such unique or primary key already exists in the table
```

**Example 3**

***CREATING A TABLE WITH 'FORIEGN KEY' CONSTRAINT:***

```
SQL> CREATE TABLE detailemp252 (eid NUMBER(5) REFERENCES mdept252
(dno),  ename VARCHAR(10),  esal NUMBER(7));

Table created.
```

**INSERING RECORDS INTO DETAIL EMPLOYEE TABLE:**

```
SQL> INSERT INTO detailemp252 VALUES (2, 'ravi', 50000);
INSERT INTO detailemp252 VALUES (2, 'ravi', 50000)
*
ERROR at line 1:
ORA-02291: integrity constraint (SYSTEM.SYS_C003877) violated - parent key not
found
```

SQL> INSERT INTO detailemp252 VALUES (1, 'teja', 60000);

1 row created.

SQL> DELETE FROM mdept252 where dno=1;
DELETE FROM mdept252 where dno=1
*
ERROR at line 1:
ORA-02292: integrity constraint (SYSTEM.SYS_C003877) violated - child record found

SQL> SELECT * FROM detailemp252;

EID ENAME ESAL
---------- ---------- ----------
1 teja 60000

SQL> SELECT * FROM mdept252;

DNO DNAME DLOC
---------- ---------- ----------
1 ravi hyd

**2) Queries (along with subqueries) using ANY, ALL, IN, EXISTS, NOT EXISTS, UNIQUE, INTERSECT, Constraints.**

**TABLE DEFINITIONS**

SQL> CREATE TABLE Customer (cust_no NUMBER(4) PRIMARY KEY,

| CUST NO | LAST NAME | FIRST NAME | ADDRESS1 | ADDRESS2 | CITY | STATE | PIN | BIRTH DATE | STAT |
|---|---|---|---|---|---|---|---|---|---|
| *1001* | *UDUPI* | *RAJ* | *UPENDRABAUG* | *NEAR KALPANA* | *UDPP* | *KARNARATA* | *576101* | *12-DEC-62* | *A* |
| *1002* | *KUMAR* | *RAJ* | | | | | | | *A* |
| *1003* | *BAHADUR* | *RAJ* | *SHANTHI VILLA* | *NEAR MALLIKA* | *UDP* | *KARNATAKA* | *576101* | *1-AUG-70* | *V* |
| *1004* | *SIMON* | *FELIX* | *M-J-56* | *ALTOBETIM* | *PJM* | *GOA* | *403002* | *12-FEB-71* | *A* |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| *1005* | *KUTTY* | *RAJAN* | *A1 TRADERS* | *NEAR RLY STATION* | *KNR* | *KERALA* | *67001* | *9-JUN-71* | *A* |
| *1006* | *PAI* | *SHILPA* | *12/4B* | *POLICE QUARTERS* | *MNG* | *KARNATAKA* | *574154* | *11-DEC-70* | *I* |
| *1007* | *JAIN* | *RAKSHIT* | *BOSCO* | *R.K PLAZA* | *BNG* | *KARNATAKA* | *576201* | *1-JAN-71* | *A* |

last_name VARCHAR2(20), first_name VARCJHAR2(20) NOT NULL, address1 VARCHAR2(20), address2 VARCHAR2(20), city VARCHAR2(3), state VARCHAR2(20), pin VARCHAR2(6), birth_date DATE, status VARCHAR2(1), CHECK (status IN ('V', 'I', 'A')));

Table created.

Insert the following data:

**QUERIES**

1) To list all the fields from the table Customer.

SEELCT * FROM Customer;

2) To list the first name, last name.

SELECT first_name, last_name FROM Customer;

3) To list the first name and last name of persons in Karnataka.

SELECT first_name, last_name FROM Customer WHERE state = 'KARNATAKA';

4) To list all the columns for invalid persons.

SELECT * FROM Customer WHERE status = 'I';

5) To list the names of active customers.

SELECT first_name, last_name FROM Customer WHERE status = 'A';

6) To list the name and address using concatenation.

SELECT first_name || ' ' || last_name, address1 || ',' || address2 || ',' || city || ',' || state || '-' || pin FROM Customer;

7) To select records where the pin code has not been entered.

SELECT * FROM Customer WHERE pin IS NULL;

8) To select the single occurrence of any value from the table.

SELECT DISTINCT state FROM Customer;

9) To select rows of valid customers from Karnataka.

```
SELECT * FROM Customer WHERE state = 'KARNATAKA' AND status = 'V';
```

10) To select rows of customers from Karnataka or Kerala.

```
SELECT * FROM Customer WHERE state = KARNATAKA' OR state = 'KERALA';
```

11) To sort the customer data in the alphabetic order of state.

```
SELECT state, first_name, last_name, pin FROM Customer ORDER BY state;
```

12) To sort in the descending order.

```
SELECT state, first_name, last_name, pin FROM Customer ORDER BY state DESC;
```

13) To sort the customer data, state wise and within state by the last name.

```
SELECT state, first_name, last_name, pin FROM Customer ORDER BY state,
last_name;
```

14) To retrieve records of Karnataka customers who are valid.

```
SELECT * FROM Customer WHERE UPPER(state) = 'KARNATAKA' AND
UPPER(status) = 'V';
```

15) To retrieve records of Karnataka/Kerala customers.

```
SELECT * FROM Customer WHERE UPPER(state) = 'KARNATAKA' OR
UPPER(state) = 'KERALA';
```

16) To retrieve records of Karnataka/Kerala customers who are active.

```
SELECT *  FROM Customer WHERE (UPPER(state) = 'KARNATAKA'  OR
UPPER(state) = 'KERALA') AND UPPER(status) = 'A';
```

17) To retrieve records of Karnataka customers with pin code 576101.

```
SELECT * FROM Customer WHERE LOWER(state) = 'karnataka' AND pin =
'576101';
```

18) To retrieve rows where the state name begins with K and followed by any other character.

```
SELECT first_name, last_name, state FROM Customer WHERE state LIKE 'K%';
```

19) To retrieve rows where the first name contains the word RAJ embedded in it.

```
SELECT first_name, last_name, state FROM Customer WHERE first_name LIKE
'%RAJ%';
```

20) To retrieve rows where the address2 contains the word UDUPI or UDIPI in which the 3rd character may be anything.

SELECT first_name, last_name, state FROM Customer WHERE address2 LIKE 'UD_PI';

21) To retrieve rows where the cust_no has data representing any value between 1003 and 1005, both numbers included.

SELECT * FROM Customer WHERE cust_no BETWEEN 1003 AND 1005;

22) To retrieve rows of persons born after 9-JAN-70 and before 1-AUG-96.

SELECT * FROM Customer WHERE birth_date BETWEEN '10-JAN-70' AND '31-JUL-96';

23) To retrieve rows where the city has data which is equal to UDP or MNG or BNG or PJM or MAR.

SELECT * FROM Customer WHERE city IN ('UDP', 'MNG', 'BNG', 'PJM', 'MAR');

**Excercise**

**TABLE DEFINITIONS**

SQL> CREATE TABLE Emp ( emp_no NUMBER, emp_name VARCHAR(20), join_date DATE, join_basic NUMBER(7, 2), PRIMARY KEY (emp_no));

Table created.

Insert the following data:

| EMP NO | EMP NAME | JOIN DATE | JOIN BASIC |
|---|---|---|---|
| *1001* | *Subhas bose* | *01-JUN-96* | *3000* |
| *1002* | *Nadeem shah* | *01-JUN-96* | *2500* |
| *1003* | *Charles babbage* | *01-JUN-96* | *3000* |
| *1004* | *Shreyas kumar* | *01-JUL-96* | *2500* |
| *1005* | *George boole* | *01-JUL-96* | *2800* |

SQL> CREATE TABLE Salary (emp_no NUMBER, basic NUMBER(7, 2), commission NUMBER(7, 2), deduction NUMBER(7, 2), salary_date DATE, FOREIGN KEY (emp_no) REFERENCES Emp);

Table created.

Insert the following data:

| EMP NO | BASIC | COMMISSION | DEDUCTION | SALARY DATE |
|---|---|---|---|---|
| *1001* | *3000* | *200* | *250* | *30-JUN-96* |
| *1002* | *2500* | *120* | *200* | *30-JUN-96* |
| *1003* | *3000* | *500* | *290* | *30-JUN-96* |
| *1004* | *2500* | *200* | *300* | *30-JUN-96* |
| *1005* | *2800* | *100* | *250* | *30-JUN-96* |
| *1001* | *3000* | *200* | *250* | *31-JUL-96* |
| *1002* | *2500* | *120* | *200* | *31-JUL-96* |
| *1003* | *3000* | *500* | *290* | *31-JUL-96* |
| *1004* | *2500* | *200* | *300* | *31-JUL-96* |
| *1005* | *2800* | *100* | *150* | *31-JUL-96* |

**QUERIES**

1) To sum the salary of each employee.
2) To sum the salary of each employee and sort it on the sum of basic.
3) To sum the salary of each employee and sort it in descending order on the sum of basic.
4) To sum the salary of each employee and sort it in descending order on the sum of basic. Display name also
5) To group the data by average salary of each employee.
6) To group the basic by month.
7) To group the data by average salary of each employee and display where average basic is more than 2000..

**SUBQUERIES**

8) To list the employees who earn less than the average salary.
9) To list the employees whose deduction is 150.
10) To list the names of employees and salary details, whose basic is less than the average salary.

## WEEK 3

2) **Queries (along with subqueries) using ANY, ALL, IN, EXISTS, NOT EXISTS, UNIQUE, INTERSECT, Constraints.**
3) **Queries using Aggregate functions (COUNT, SUM, AVG, MAX and MIN), GROUP BY, HAVING and Creation and Dropping of Views.**
4) **Queries using Conversions, functions (to_char, to_num, and to_date), string function (Conactenation, lpad, rpad, ltrim, rtrim, lower, upper, initcap, length, substr, and instr), date functions (sysdate, next_day, add_months, last_day, months_between, least, greatest, trunk, round, to_char, to_date).**

### TABLE DEFINITIONS

Use above tables.

### QUERIES

11) To sum the salary of each employee.

```
SELECT emp_no, SUM(basic) FROM salary GROUP BY emp_no;
```

12) To sum the salary of each employee and sort it on the sum of basic.

```
SELECT emp_no, SUM(basic) FROM salary GROUP BY emp_no ORDER BY
SUM(basic);
```

13) To sum the salary of each employee and sort it in descending order on the sum of basic.

```
SELECT emp_no, SUM(basic) FROM salary GROUP BY emp_no ORDER BY
SUM(basic) DESC;
```

14) To sum the salary of each employee and sort it in descending order on the sum of basic. Display name also

```
SELECT s.emp_no, e.emp_name, SUM(s.basic) FROM salary s, emp e WHERE
s.emp_no = e.emp_no GROUP BY s.emp_no, e.emp_name ORDER BY SUM(s.basic)
DESC;
```

15) To group the data by average salary of each employee.

```
SELECT s.emp_no, INITCAP(e.emp_name), AVG(s.basic) FROM salary s, emp e
WHERE s.emp_no = e.emp_no GROUP BY s.emp_no, e.emp_no ORDER BY
AVG(s.basic);
```

16) To group the basic by month.

SELECT TO_CHAR(salary_date, 'MONTH') "MONTH", SUM(basic) "TOTAL BASIC" FROM salary GROUP BY TO_CHAR(salary_date, 'MONTH');

17) To group the data by average salary of each employee and display where average basic is more than 2000..

SELECT s.emp_no, INITCAP(e.emp_name), AVG(s.basic) FROM salary s, emp e WHERE s.emp_no = e.emp_no GROUP BY s.emp_no, e.emp_no HAVING AVG(s.basic) >= 2000 ORDER BY AVG(s.basic);

**SUBQUERIES**

18) To list the employees who earn less than the average salary.

SELECT * FROM salary WHERE basic < (SELECT AVG(basic) FROM salary);

19) To list the employees whose deduction is 150.

SELECT * FROM salary WHERE emp_no IN (SELECT emp_no FROM salary WHERE deduction = 150);

20) To list the names of employees and salary details, whose basic is less than the average salary.

SELECT s.*, e.emp_name FROM salary s, emp e WHERE s.emp_no = e.emp_no AND s.basic < (SELECT AVG(basic)
FROM salary);

## WEEK 4

**TABLE DEFINITIONS**

Branch Schema <branch-name, branch-city, assets>
Customer Schema <customer-name, customer-street, customer-city>
Loan Schema <loan-number, branch-name, amount>
Borrower Schema <customer-name, loan-number>
Account Scheme <account-number, branch-name, balance>
Depositor Scheme <customer-name, account-number>

BRANCH TABLE

| 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 | 800000 |

CUSTOMER TABLE

| 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 |

LOAN TABLE

| 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 |

BORROWER TABLE

| Customer Name | Loan Number |
|---|---|
| Adams | l-16 |
| Curry | L-93 |
| Hayes | L-15 |
| Jackson | L-14 |
| Jones | L-17 |
| Smith | L-11 |
| Smith | L-23 |

| | |
|---|---|
| Williams | L-17 |

ACCOUNT TABLE

| 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 |

DEPOSITOR TABLE

| Customer Name | Account Number |
|---|---|
| Hayes | A102 |
| Johnson | A-101 |
| Johnson | A-201 |
| Jones | A-217 |
| Lindsay | A-222 |
| Smith | A-215 |
| Turner | A-305 |

**QUERIES**

1) To list all the fields from the table Customer.

SELECT branch_name FROM Loan;

2) To list rows after eliminating duplicates.

SELECT distinct branch_name FROM Loan;

3) To explicitly list rows, including duplicates.

SELECT all branch_name FROM Loan;

4) To list fields after applying arithmetic operations.

SELECT loan_number, branch_name, amount *100 FROM Loan;

5) Find all loan numbers for loans made at the Perryridge branch with loan amounts greater than Rs1200.

SELECT loan_number FROM Loan WHERE branch_name = 'Perryridge' AND amount > 1200;

6) Find all loan numbers for loans with loan amounts between Rs90,000 and Rs100,000.

SELECT loan_number FROM Loan WHERE amount BETWEEN 90000 AND 100000;

Or

SELECT loan_number FROM Loan WHERE amount <= 100000 AND amount >= 90000;

7) Find all loan numbers for loans with loan amounts not between Rs90,000 and Rs100,000.

SELECT loan_number FROM Loan WHERE amount NOT BETWEEN 90000 AND 100000;

8) For all customers who have a loan from the bank, find their names, loan numbers and loan amounts.

SELECT customer_name, Borrower.loan_number, amount FROM Borrower, Loan WHERE Borrower.loan_number = Loan.loan_number;

Or

SELECT customer_name, Borrower.loan_number AS loan_id, amount FROM Loan WHERE Borrower.loan_number = Loan.loan_number;

9) Find the customer names, loan numbers and loan amounts for all loans at the Perryridge branch.

SELECT customer_name, Borrower.loan_number, amount FROM Borrower, Loan WHERE Borrower.loan_number = Loan.loan_number AND branch_name = Perryridge';

Or

SELECT customer_name, T.loan_number, S.amount FROM Borrower AS T, Loan AS S WHERE T.loan_number = S.loan_number AND branch_name = 'Perryridge';

10) Find the names of all branches that have assets greater than atleast one branch located in Brooklyn.

SELECT DISTINCT T.branch_name FROM Branch as T, Branch as S WHERE T.assets > S.assets AND S.branch_city = 'Brooklyn';

11) Find the names of all customers whose street address includes the substring 'Main'.

SELECT customer_name FROM Customer WHERE customer_street LIKE '%Main%';

12) To list in alphabetic order all customers who have a loan at the Perryridge branch.

SELECT DISTINCT customer_name FROM Borrower B, Loan L WHERE B.loan_number = L.loan_number AND branch_name = 'Perryridge' **ORDER BY** customer_name;

13) To list the entire loan info in descending order of amont.

SELECT * FROM Loan ORDER BY amount DESC, loan_number ASC;

14) To find all customers having a loan, an account or both at the bank, without duplicates.

(SELECT customer_name FROM Depositor) UNION (SELECT customer_name FROM Borrower);

15) To find all customers having a loan, an account or both at the bank, with duplicates.

(SELECT customer_name FROM Depositor) UNION ALL (SELECT customer_name FROM Borrower);

16) To find all customers having both a loan and an account at the bank, without duplicates.

(SELECT customer_name FROM Depositor) INTERSECT (SELECT customer_name FROM Borrower);

17) To find all customers having a loan, an account or both at the bank, with duplicates.

(SELECT customer_name FROM Depositor) INTERSECT ALL (SELECT customer_name
FROM Borrower);

18) To find all customers who have an account but no loan at the bank, without duplicates.

(SELECT DISTINCT customer_name FROM Depositor) EXCEPT (SELECT customer_name FROM Borrower);

19) To find all customers who have an account but no loan at the bank, with duplicates.

(SELECT DISTINCT customer_name FROM Depositor) EXCEPT ALL (SELECT customer_name FROM Borrower);

20) Find the average account balance at the Perryridge branch

SELECT branch_name, AVG(balance) FROM Account WHERE branch_name = 'Perryridge';

21) Find the average account balance at the each branch SELECT AVG(balance) FROM Account GROUP BY branch_name;

22) Find the number of depositors for each branch .

SELECT branch_name, COUNT(DISTINCT customer_name) FROM Depositor D, Account A WHERE D.account_number = A.account_number GROUP BY branch_name;

23) Find the number of depositors for each branch where average account balance is more than Rs 1200.

SELECT branch_name, COUNT(DISTINCT customer_name) FROM Depositor D, Account A WHERE D.account_number = A.account_number GROUP BY branch_name HAVING AVG(balance) > 1200;

24) Find the average balance for all accounts.

SELECT AVG(balance) FROM Account;

25) Find the number of tuples in the customer relation.

SELECT COUNT(*) FROM Customer;

26) Find the average balance for each customer who lives in Harrision and has at least three accounts.

SELECT D.customer_name, AVG(balance) FROM Depositor D, Account A, Customer C WHERE D.account_number = A.account_number AND D.customer_name = C.customer_name AND C.customer_city = 'Harrison' GROUP BY D.customer_name HAVING COUNT(DISTINCT D.account_number) >= 3;

27) Find all the loan number that appear in loan relation with null amount values.

SELECT loan_number FROM Loan WHERE amount IS NULL;

28) Find all customers who have both a loan and an account at the bank.

SELECT customer_name FROM Borrower WHERE customer_street IN (SELECT customer_name FROM Depositor);

29) Find all customers who have both an account and a loan at the Perryridge branch

SELECT DISTINCT B.customer_name FROM Borrower B, Loan L WHERE B.loan_number L.loan_number AND branch_name = 'Perryridge' AND (branch_name, customer_name) IN (SELECT branch_name, customer_name FROM Depositor D, Account A WHERE D.account_number = A.account_number);

or

SELECT customer_name FROM Borrower B WHERE EXISTS (SELECT * FROM Depositor D WHERE D.customer_name = B.customer_name);

30) Find all customers who do not have a loan at the bank, but do not have an account the bank.

SELECT DISTINCT customer_name FROM Borrower WHERE customer_name NOT IN (SELECT customer_name FROM Depositor);

31) Find the names of customers who do have a loan at the bank, and whose names are neither Smith nor Jones.

SELECT DISTINCT customer_name FROM Borrower WHERE customer_name NOT IN ('Smith', 'Jones');

32) Find the names of all branches that have assets greater than those of at least one branch located in Brooklyn.

SELECT DISTINCT T.branch_name FROM Branch AS T, Branch AS S WHERE T.assets > S.assets AND S.branch_city = 'Brooklyn';

33) Find the names of all branches that have assets greater than that of each branch located in Brooklyn.

SELECT branch_name FROM Account GROUP BY branch_name HAVING AVG(balance) >= ALL (SELECT AVG(balance) FROM Account GROUP BY branch_name);

34) Find all customers who have an account at all the branches located in Brooklyn.

SELECT DISTINCT S.customer_name FROM Depositor AS D WHERE NOT EXISTS ((SELECT branch_name FROM Branch WHERE branch_city = 'Brroklyn) EXCEPT (SELECT R.branch_name FROM Depositor AS T, Account AS R WHERE T.account_number = R.account_number AND D.customer_name = t.customer_name));

35) Find all customers who have at most one account at the Perryridge branch.

SELECT T.customer_name FROM Depositor AS T WHERE UNIQUE (SELECT R.customer_name FROM Depositor AS R, Account AS A WHERE T.customer_name = R.customer_name AND R.account_number = A.account_number AND A.branch_name = 'Perryridge');

36) 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 Depositor AS R, Account AS A

WHERE T.customer_name = R.customer_name AND R.account_number = A.account_number AND A.branch_name = 'Perryridge');

37) Find the average account balance of those branches where the average account balance is greater than 1200.

SELECT branch_name, avg_balance FROM (SELECT branch_name, AVG(balance) FROM Account GROUP BY branch_name) AS Branch_avg(branch_name, avg_balance) WHERE avg_balance > 1200;

38) Find the maximum across all branches of the total balance at each branch.

SELECT MAX(tot_balance) FROM (SELECT branch_name, SUM(balance) FROM Account GROUP BY branch_name) AS Branch_total(branch_name, tot_balance);

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

SELECT d-CN FROM (Depositor LEFT OUTER JOIN Borrower ON Depositor.customer_name = Borrower.customer_name) AS db1(d-CN, account_number, b-CN, loan_number) WHERE b-CN is null;

40) Find the all customers who have either an account or a loan (but not both) at the bank.

SELECT customer_name FROM (Depositor NATURAL FULL OUTER JOIN Borrower) WHERE account_number IS NULL OR loan_number IS NULL;

## WEEK 5

### DUAL (ORACLE WORK TABLE):

1) To display system date.
   SELECT SYSDATE FROM DUAL;

2) To display arithmetic calculations.
   SELECT 2*2 FROM DUAL;

3) To display the logged user.
   SELECT USER FROM DUAL;

4) To display system time.
   SELECT TO_CHAR(SYSDATE, 'HH:MI:SS') FROM DUAL;

5) To display current month.
SELECT TO_CHAR(SYSDATE, 'MONTH') FROM DUAL;

6) To display system date in specified format.
SELECT TO_CHAR(SYSDATE, 'DD/MM/YY') FROM DUAL;

7) To display system date in specified format.
SELECT TO_CHAR(SYSDATE, 'MM') FROM DUAL;

8) To display date arithmetic.
SELECT ADD_MONTHS(SYSDATE, 5) FROM DUAL;

9) To display date arithmetic.
SELECT LAST_DAY(SYSDATE) FROM DUAL;

10) To display date arithmetic.
SELECT MONTHS_BETWEEN(SYSDATE, '01-APR-09') FROM DUAL;

11) To display date arithmetic.
SELECT NEXT_DAY(SYSDATE, 'MON') FROM DUAL;

**GROUP FUNCTIONS:**

12) To display average basic salary of the employees.
SELECT SUM(basic) FROM salary;

13) To display minimum basic salary of the employees.
SELECT MIN(basic) FROM salary;

14) To display maximum basic salary of the employees.
SELECT MAX(basic) FROM salary;

15) To display sum of basic salaries of all the employees.
SELECT SUM(basic) FROM salary;

16) To display the number of records in salary table.
SELECT COUNT(*) FROM salary;

**STRING FUNCTIONS:**

17) To display a field value after left padding.

SELECT LPAD('PAGE-1', 10, '*') FROM DUAL;

18) To display a field value after left padding.
SELECT RPAD('PAGE-1', 10, '*') FROM DUAL;

19) To display a field value after converting to lower case.
SELECT LOWER('A') FROM DUAL;

20) To display a field value after converting to upper case.
SELECT LOWER('a') FROM DUAL;

21) To display a field value after converting to initial capital case.
SELECT INITCAP('HOW ARE YOU?') FROM DUAL;

22) To display a substring of a field value.
SELECT SUBSTR('CSE2A', 4, 2) FROM DUAL;

23) To display the length of a field value.
SELECT LENGTH('HOW LONG AM I?') FROM DUAL;

24) To display a field value after trimming the right side.
SELECT RTRIM('CSE2A', '2A') FROM DUAL;

25) To display a field value after trimming the left side.
SELECT LTRIM('CSE2A', 'CSE') FROM DUAL;

## WEEK 6

1. Create a table - use name "Software" with the fields and insert the values:

| Field name | Field type | Field size |
|---|---|---|
| Programmer name | character | 15 |
| Title | character | 20 |
| Language used | character | 15 |
| Software cost | number | 10 with 2 decimal places |
| Development cost | number | 10 |
| Software sold | number | 3 |

Queries:
a) Display the details of software developed by "PRAKASH".
b) Display the details of the packages whose software cost exceeds "2000".
c) Display the details of the software that are developed in "C++".

d) What is the price of costliest software developed in "C".
e) Display the details of the programmer whose language used is same as "Suresh".

2. Create a table "Company" with the following fields and insert the values:

| Field name | Field type | Field size |
|---|---|---|
| Company name | character | 15 |
| Proprietor | character | 15 |
| Address | character | 25 |
| Supplier name | character | 15 |
| No of employees | number | 4 |
| GP percent | number | 6 with 2 decimal places |

Queries:
a) Display all the records of the company which are in the ascending order of GP percent
b) Display the name of the company whose supplier name is "Telco".
c) Display the details of the company whose GP percent is greater than 20 and order by GP percent.
d) Display the detail of the company having the employee ranging from 300 to1000
e) Display the name of the company whose supplier is same as like Tata's.

3. Create a table named "Employee" with the following fields and insert the values:

| Field name | Field type | Field size |
|---|---|---|
| Employee Name | character | 15 |
| Employee Code | number | 6 |
| Address | character | 25 |
| Designation | character | 15 |
| Grade | character | 1 |
| Date of Joining | Date | - |
| Salary | number | 10 with 2 decimal places |

Queries:
a) Display name of the employees whose salary is greater than "10,000".
b) Display the details of employees in ascending order according to Employee Code
c) Display the total salary of the employees whose grade is "A".
d) Display the details of the employee earning the highest salary.
e) Display the names of the employees who earn more than "Ravi"

4. Create a table named "Student" with the following fields and insert the values:

| Field name | Field type | Field size |
|---|---|---|
| Student Name | character | 15 |
| Gender | character | 6 |
| Roll No. | character | 10 |
| Department Name | character | 15 |
| Address | character | 25 |
| Percentage | number | 4 with 2 decimal places |

Queries:
a) Calculate the average percentage of the students.
b) Display the names of the students whose percentage is greater than 80
c) Display the details of the student who got the highest percentage.
d) Display the details of the students whose percentage is between 50 and 70.
e) Display the details of the students whose percentage is greater than the percentage of Roll No = 12CA01

## WEEK 7

5. Create the table "PRODUCT" with the following fields and insert the values:

| Field name | Field type | Field size |
|---|---|---|
| Product no | number | 6 |
| Product name | character | 15 |
| Unit of measure | character | 15 |
| Quantity | number | 6with 2 decimal places |
| Total amount | number | 8 with 2 decimal places |

Queries:
a) Using update statements calculate the total amount and then select the record.
b) Select the records whose unit of measure is "Kg"
c) Select the records whose quantity is greater than 10 and less than or equal to 20
d) Calculate the entire total amount by using sum operation
e) Calculate the number of records whose unit price is greater than 50 with count operation

6. Create the table PAYROLL with the following fields and insert the values:

| Field name | Field type | Field size |
|---|---|---|
| Employee no | number | 8 |
| Employee name | character | 8 |
| Department | character | 10 |

| | | |
|---|---|---|
| Basic pay | number | 8 with 2 decimal places |
| HRA | number | 6 with 2 decimal places |
| DA | number | 6 with 2 decimal places |
| PF | number | 6 with 2 decimal places |
| Net pay | number | 8 with 2 decimal places |

Queries;
a) Update the records to calculate the net pay.
b) Arrange the records of employees in ascending order of their net pay
c) Display the details of the employees whose department is: sales"
d) Select the details of employees whose HRA>=1000 and DA<=900
e) Select the records in descending order

## WEEK 8

1. Consider the supplier relations and show output for each of the queries given below:

**(S)**

| S# (Supplier No.) | SNAME (Supplier Name ) | STATUS | CITY |
|---|---|---|---|
| S 1 | Smith | 20 | London |
| S2 | Jones | 10 | Paris |
| S 3 | Blake | 30 | Paris |
| S4 | Adams | 30 | Athens |
| S5 | Clark | 20 | London |

**(SP)**

| S# (Supplier No.) | P#(Part No.) | Quantity |
|---|---|---|
| S1 | P1 | 300 |
| S1 | P2 | 200 |
| S2 | P1 | 100 |
| S2 | P2 | 400 |
| S3 | P2 | 200 |
| S4 | P2 | 200 |

Question 1.Get supplier numbers for suppliers with status > 20 and city is Paris.
Question 2.Get Supplier Numbers and status for suppliers in Paris, in descending order of
status.

Question 3.Get all pairs of supplier numbers such that the two suppliers are located in the same city. (Hint: It is retrieval involving join of a table with itself.)
Question 4.Get unique supplier names for suppliers who supply part P2.
Question 5.Give the same query above by using the operator IN.
Question 6.Get part numbers supplied by more than one supplier. (Hint: It is retrieval with a sub-query, with interblock reference and same table involved in both blocks).
Question 7.Get supplier numbers for suppliers who are located in the same city as supplier S 1. (Hint: Retrieval with sub query and unqualified comparison operator).
Question 8.Get supplier names for suppliers who supply part P1. (Hint: Retrieval using
EXISTS)
Question 9. Get part numbers for parts whose quantity is greater than 200 or are currently
supplied by S2. (Hint: Use a retrieval using union).
Question 10. Suppose for the supplier S5 the value for status is NULL instead of 20. Get supplier numbers for suppliers greater than 25 or is NULL. (Hint: Retrieval using NULL ).
Question 11. Get the number of Suppliers, who are supplying at least one part. (Hint: This query is using the built-in function count).
Question 12. For each part supplied, get the part no. and the total quantity supplied for that part. (Hint: The query using GROUP BY).
Question 13. Get part numbers for all parts supplied by more than one supplier. (Hint: It is GROUP BY with HAVING).
Question 14. For all parts such that the total quantity supplied is greater than 300 (exclude from the total all shipments for which quantity is less than or equal to 200), get the part no. and the maximum quantity of the part supplied, and order tlie result by
descending part no. within those maximum quantity values which are in ascending order.
Question 15. Double the status of all suppliers in London. (Hint: UPDATE Operation).
Question 16. Let us consider the table TEMP has one column, called P#. Enter into TEMP part numbers for all parts supplied by S2.
Question 17. Add part P7.
Question 18. Delete all the suppliers in London and also the supplies concerned.

2. Consider the employee database. Give an expression in SQL and show the output for each of the following queries:
EMPLOYEE (Employee-Name, Street, City)
WORKS (Employee-Name, Company-Name, Salary)

COMPANY (Company-Name, City)
MANAGES (Employee-Name, Manager-Name)

1) Find the names of all employees who work for First Bank Corporation.
2) Find the names and cities of residence of all employees who work for the First Bank Corporation.
3) Find the names, street address, and cities of residence of all employees who work for First Bank Corporation and earn more than Rs. 10,000.
4) Find the employees in the database who live in the same cities as the companies for which they work.
5) Find all employees in the database who live in the same cities and on the same streets as do their managers.
6) Find all employees in the database who do not work for First Bank Corporation.
7) Find all employees in the database who earn more than every employee of Small Bank Corporation.
8) Assume that the compa~iies may be located in several cities. Find all companies located in every city in which Small Bank Corporation is located
9) Find all employees who earn more than the average salary of all employees of their company.
10) Find the company that has the most employees.
11) Find the company that has the smallest payroll.
12) Find those companies whose employees earn a higher salary, on average, than the average salary at First Bank Corporation.

## WEEK 9 (PL/SQL)

### Syntax to write a sql program

Declare
<declaration stmts>
Begin
<executable stmts>
[exception <exceptional stmts>]----- optional
End;
/---end of buffer

***Example: 1***
***Create a file DBFOR.SQL, to execute the FOR loop and display the variable.***

At SQL Prompt type, ed dbfor to open notepad and type the below program:
**Program**

declare

```
cnt number;
begin
dbms_output.put_line('This is a demo of FOR loop ');
for cnt in 1..5 loop
                dbms_output.put_line('loop number ' || cnt);
end loop;
end;
/
set serveroutput off
```

Save the file and at SQL prompt run as:

**Execution**

```
SQL>set serveroutput on
SQL> start dbfor      (press enter) OR
SQL> @dbfor
```

**OUTPUT:-**

This is a demo of FOR loop

loop number 1

loop number 2

loop number 3

loop number 4

loop number 5

**PS:**

For syntax:

```
For <var> in <start_num> .. <endnum> loop
        <statement(s);>
End loop;
```

***Example: 2***

***Create a file DBREVFOR.SQL, to execute the REVERSE FOR loop and display the variable.***

**Program**

```
begin
```

```
        dbms_ouput.put_line('This is a demo of REVERSE FOR loop');
        for cnt in reverse 1..10 loop
                if mod(cnt, 2) = 0 then
                        dbms_output.put_line('loop counter ' || cnt);
                end if;
        end loop;
end;
/
```

**OUTPUT:-**

This is a demo of REVERSE FOR loop

| | | |
|---|---|---|
| loop | counter | 10 |
| loop | counter | 8 |
| loop | counter | 6 |
| loop | counter | 4 |
| loop | counter | 2 |

**PS:**

Reverse For syntax:

```
For <var> in reverse <start_num> .. <endnum> loop
        <statement(s);>
End loop;
```

Other forms of if syntax are:

```
If <condition> then
        <action(s);>
End if;

If <condition> then
        <action(s);>
Else
        <action(s);>
End if;

If <condition> then
        <action(s);>
Elsif <condition> then
        <action(s);>
else
```

```
        <action(s);>
End if;
```

***Example: 3***

***Create a file DBLOOP.SQL, to execute the LOOP loop and display the variable.***

**Program**

```
set serveroutput on
declare
        cnt number(2) := 0;
begin
        dbms_ouput.put_line('This is a demo of LOOP loop');
        loop
                cnt := cnt + 1;
                exit when cnt > 10;
                dbms_output.put_line('loop counter ' || cnt);
        end loop;
end;
/
set serveroutput off
```

**OUTPUT**:-

```
This is the demo of LOOP loop

loop counter 1

loop counter 2

loop                    counter         3
loop                    counter         4
loop                    counter         5
loop                    counter         6
loop                    counter         7
loop                    counter         8
loop                    counter         9
loop                    counter         10
```

**PS:**

Loop syntax:

```
loop
        <statement(s);>
```

```
        Exit when <condition>;
End loop;
```

***Example: 4***

***Create a file DBWHILE.SQL, to execute the WHILE loop and display the variable.***

**Program**

```
set serveroutput on
declare
        cnt number(2) := 1;
begin
        dbms_ouput.put_line('This is a demo of WHILE loop');
        while cnt <= 10 loop        dbms_output.put_line('loop    counter:    '    || to_char(cnt, '999'));
                cnt := cnt + 1;
        end loop;
end;
/
set serveroutput off
```

**OUTPUT:-**

```
This is a demo of WHILE loop

loop counter : 1

loop counter : 2

loop                    counter                 :                   3
loop                    counter                 :                   4
loop                    counter                 :                   5
loop                    counter                 :                   6
loop                    counter                 :                   7
loop                    counter                 :                   8
loop                    counter                 :                   9
loop counter : 10
```

**PS:**

```
while syntax:
while <condition> loop
        <statement(s);>
End loop;
```

## WEEK 10 (PL/SQL)

***Example: 4***

***Write a program EMPDATA.SQL, to retrieve the employee details of an employee whose number is input by the user .***

**Program**

```
-- PROGRAM TO RETRIEVE EMP DETAILS
set serveroutput on

prompt Enter Employee Number:
accept n
declare
        dname emp.emp_name%type;
        dbasic emp.emp_basic%type;
        ddesig emp.desig%type;
begin
        select emp_name, basic, design
        into dname, dbasic, ddesig
        from emp
        where emp_no = &n;
        dbms_ouput.put_line('Employee Details:);
        dbms_output.put_line('Name:     ' || dname);
        dbms_output.put_line('Basic:      ' || dbasic);
        dbms_output.put_line('Designation:      ' || ddesig);
end;
/
```

**OUTPUT:-**

**enter employee number:**

**13**

```
old 9:where eno =&n;
new 9:where eno=13;
employee details
Name:allen
basic:9500
desig:mechs
set serveroutput off
```

**PS:**

Similarly you can use other SQL statements in the PL/SQL block

*Exercises:*

**1) Write a PL/SQL code, EX_INVNO.SQL, block for inverting a number using all forms of loops.**

**ANSWER:-**

```
declare
n number(20):=123;
s number(13):=0;
d number(3):=1;
r number(3):=10;
begin
dbms_output.put_line('the number is :' || n);
while n>0 loop
d:=mod(n,10);
s:=(s*r)+d;
n:=n/r;
end loop;
dbms_output.put_line('inverted values' || s);
end;
/
```

***OUTPUT:-***

```
the number is:123
inverted value is:321
```

**2) Write a PL/SQL code, EX_SUMNO.SQL that prints the sum of 'n' natural numbers.**

ANSWER:-

```
prompt enter number:
accept number n
declare
isum number(2):=0;
i number;
n number:=&n;
begin
for i in 1..n loop
isum:=isum+i;
end loop;
dbms_output.put_line('sum is ' || isum);
end;
/
```

**OUTPUT:-**

enter the number:7
sum is 28

**3) Write a PL/SQL code, EX_AREA.SQL, of block to calculate the area of the circle for the values of radius varying from 3 to 7. Store the radius and the corresponding values of calculated area in the table AREA_VALUES.**

ANSWER:-

```
set serveroutput on
declare
area number(5);
rad number(3);
pi number(4):=3.14;
begin
for rad in 3..7 loop
area:=pi*rad*rad;
dbms_output.put_line('area is' || area);
insert into area_values values(area,rad);
end loop;
end;
/
```

OUTPUT:-

```
area                    is                    :27
area                    is                    :48
area                    is                    :75
area                    is                    :108
area is :147
```

## WEEK-11

1. Construct an E-R diagram for a car-insurance company that has a set of customers, each of whom owns one or more cars. Each car has associated with it zero to any number of recorded accidents.
(ii) Construct appropriate tables.

2. Consider an E-R diagram in which the same entity set appears several times. Why is allowing this redundancy a bad practice that one should avoid whenever possible?

3. The room booking side of a small hotel is to be computerized. The hotel has a number of rooms. Each room has a basic (double) price and a supplementary price for extra children. These prices also depend on the time of year- it is more expensive at Christmas, during the summer and around bank holidays, for example. There are 3 seasonal bands. The system must enable the hotel proprietor to answer phone calls from prospective clients (for example, rooms available now and in the future, with costs), make provisional bookings, do mailings of previous clients, prepare clients' bill (ignore extras such as papers, drinks etc). Recognise various entities and relationship among them. Construct an E-R diagram for the above.

4. A student uses a particular computing system to do the computations for a given course using a limited hour account code. Find the appropriate relationship. Draw the E-R diagram.

5. Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examinations conducted.

**REFERENCES:**


1. Oracle 9i Release 2 (9.2) SQL Reference, www.cs.ncl.ac.uk/teaching/facilities/swdoc/oracle9i/server.920/a96540/toc.htm.

2. Oracle 9i Release 1 (9.0.1) SQL Reference, http://download-east.oracle.com/docs/cd/A91202_01/901_doc/server.901/a90125/toc.htm.

3. An A-Z Index of Oracle SQL Commands (version 9.2) http://www.ss64.com/ora/.

4. Database Systems Instructor: Prof. Samuel Madden Source: MIT Open Courseware (http://ocw.mit.edu).

5. RDBMS Lab Guide, www.campusconnect.infosys.com userid:demo@infosys and passwork:infosys.

6. Orelly PL/SQL Pocket Reference, http://www.unix.org.ua/orelly/oracle/langpkt/index.htm

7. PL/SQL User's Guide and Reference, Release 2 (9.2)

http://www.lc.leidenuniv.nl/awcourse/oracle/appdev.920/a96624/toc.htm