DATABASE CREATION AND SIMPLE QUERIES

1. CREATE TABLE "EMPLOYEE" WITH FIELDS ENO,ENAME, ADDDRESS, SALARY, JOB, MANAGER NUMBER, DEPARTMENT, COMMISSION.

QUERY:

Create table employee (no number(10), name varchar(20), addr varchar(20), salary number(6), job varchar(10), managno number(10), departmentno number(5), comm. Number(6));

---

2. ALTER THE TABLE "EMPLOYEE" BY ADDING A NEW FIELD GENDER.

QUERY:

Alter table employee add (gender char);

---

3. CREATE TABLE "DEPARTMENT" WITH FIELDS DNO, DDNAME, LOCATION.

QUERY:

Create table department (dno number(5), dname varchar(10), loc varchar(15));

---

4. CHANGE THE FIELD NAME
   'SALARY' IN "EMPLOYEE" INTO 'BASIC'
   'LOC' IN "DEPARTMENT" INTO 'LOCATION'.

QUERY:

Alter table employee rename column salary to basic;
Alter table department rename column loc to location;

---

5. INCREASE THE WIDTH OF THE FIELD ADDRESS IN THE TABLE EMPLOYEE.

QUERY:

Alter table employee modify (addr varchar(15));

---

6. DELETE A COLUMN COMM FROM THE EMPLOYEE TABLE.

QUERY:

Alter table employee drop column comm.;

---

7. INSERT VALUES INTO ABOVE TABLES.

QUERY:

Insert into employee values (&no, '&name','&addr',
&basic,'&job','&dob',&manno,&dno,'&gender');

Insert into department values (&dno,'&dname','&loc');

---

8. LIST THE DETAILS OF EMPLOYEE FROM CHENNAI.

QUERY:

```
Select * from employee Where addr =’employee’;
```

---

9. LIST ALL DEPARTMANT DETAILS.

QUERY:

```
Select * from department;
```

---

10. LIST THE NAMES OF EMPLOYEES WHOSE SALARY IS GREATER THAN 5000.

QUERY:

```
Select * from employee  Where basic >= 5000;
```

---

11. LIST ENAME,ADDR & ENO OF FEMALE EMPLOYEES WORKING IN DEPARTMANT NO. 45.

QUERY:

```
Select no,name,addr from employee Where gender=’f’ and dno=45;
```

---

12. LIST THE DETAILS OF ASSISTANT PROFESSOR.

QUERY:

```
Select * from employee Where job=’assistant professor’;
```

---

13. LIST ENAME, ENO, ADDR & DNO WHOSE SALARY IS IN THE RANGE OF 10000 & 20000.

QUERY:

```
Select no,name,addr,dno from employee  Where basic >= 10000 and basic <= 20000;
```

---

14. LIST THE DETAILS OF PROFESSOR AND ASSISTANT PROFESSOR.

QUERY:

```
Select * from employee  Where job=’professor’ or job=’assistant professor’;
```

---

15. CHANGE THE LOCATION OF DEPARTMENT SCIENCE INTO MUMBAI.

QUERY:

Update department set loc='mumbai' Where dno=101;

---

16. LIST ENO & ADDR OF EMPLOYEE WHOSE SALARY IS ABONE 10,000 & NOT TO BE A PROFESSROR.

QUERY:

Select no, addr from employee Where basic>=10000 and job!=(<>) 'professor';

---

17. DELETE A TABLE INCLUDING ITS STRUCTURE.

QUERY:

Drop table department;

---

18. DELETE A TABLE EXCEPT ITS STRUCTURE.

QUERY:

Truncate table department;

NESTED QUERIES

1.List the details of employee whose salary is greater than the salary of John.

QUERY:

```
Select * from employee
Where basic > (select basic from employee
Where ename='John');
```

2. List the details of employee whose basic salary is greater than any one of the salary of employee working in department number 20.

QUERY:

```
Select * from employee
Where basic > any(select basic from employee
Where dno=20);
```

3.List the details of employee who are assigned for atleast one project

QUERY:

```
Select * from employee
Where pno in (select pno from
 (select pno,count(*) from worksin
group by pno
having count(*) > 1));
```

4.List the details of male employee who are getting more salary then all female employees
.

QUERY:

```
Select * from employee
Where basic> all(select basic from
Employee where gender='f');
```

5.List the details of employee who are assigned for morethan 3 projects.

QUERY:

```
select * from employee
Where pno in (select pno from
(select pno,count(*) from worksin
group by pno
having count(*) > 3));
```

6.List the details of employee whose address ano department number is as same as John.

QUERY:

```
select * from employee
Where (addr,dno) = (select addr,dno
From employee where name='John');
```

7.Get name and salary with eno lesser then 110 whose basic is more than the basic of at least on employee with eno greater than 150.

QUERY:

```
select name, basic from employee
Where eno > 110 and basic  = any(select
Basic from employee where eno > 150);
```

8. List name and address of employee whose basic is greater then highest salary number of employee belong in dno=45.

QUERY:

```
select name,addr from employee
Where basic < (select max(basic) from
Employee where dno=45);
```

9.Get details of employee who are not assigned for any project.

QUERY:

```
Select * from employee
Where pno in(select pno from (select pno,
Count(*) from worksin
Group by pno
Having count(*)>2));
```

9. List name and address of employee whose salary is greater than highest salary number of employee belong in dno=45.

QUERY

```
Select name,addr from employee
Where basic>(select max(basic) from
Employee where dno=45);
```

FIBONACCI SERIES

```
declare
   f1 number:=0;
   f2 number:=1;
   f3 number;
   i number;
   n number:=&n;
begin
   dbms_output.put_line(f1);
   dbms_output.put_line(f2);
   for i in 1..n-2 loop
      f3:=f1+f2;
     dbms_output.put_line(f3);
      f1:=f2;
      f2:=f3;
   end loop;
end;
```

OUTPUT

```
Enter value for n: 5
old   6: n number:=&n;
new   6: n number:=5;
0
1
1
2
3
```

Multiplication table

```
declare
a number:=5;
b number(4);
c number(4);
begin
for c in 1..10 loop
b:=a*c;
dbms_output.put_line(a || ' * '|| c || ' = ' || b);
end loop;
end;
```

output

```
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

using while loop

```
declare
a number:=5;
b number(4);
c number:=1;
begin
while(c<10) loop
b:=a*c;
dbms_output.put_line(a || ' * '|| c || ' = ' || b);
c:=c+1;
end loop;
end;
```

output

5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45

using if else construct

```
declare
a number:=5;
b number;
c number:=1;
begin
a:=&a;
b:=&b;
if (a<b) then
dbms_output.put_line('b is greater');
else
dbms_output.put_line('a is greater');
end if;
end;
```

```
Enter value for a: 10
old   6: a:=&a;
new   6: a:=10;
Enter value for b: 3
old   7: b:=&b;
new   7: b:=3;
a is greater

PL/SQL procedure successfully completed.
```

Select case

```
SQL>  declare
  2   a number(4);
  3   begin
  4   a:=&a;
  5  case
  6  when (a=1) then
  7  dbms_output.put_line('good mrn');
  8  when (a=2) then
  9  dbms_output.put_line('good noon');
 10  when (a=3) then
 11   dbms_output.put_line('good eve');
 12  else
 13  dbms_output.put_line('good day');
 14  end case;
 15  end;
 16  /
Enter value for a: 2
old   4:  a:=&a;
new   4:  a:=2;


PL/SQL procedure successfully completed.
```

SQL> set serveroutput on

```
SQL> /
Enter value for a: 2
old   4:  a:=&a;
new   4:  a:=2;
good noon
```

ELECTRICITY BILL

```
 declare
   cursor eb is select * from bill;
   cost1 number;
   unit1 number;
 eb1 bill%rowtype;
  begin
    dbms_output.put_line('...cusno cus_name c_reading p_reading unit totalcost...');
    dbms_output.put_line('***************');
  open eb;
  loop
    fetch eb into eb1;
exit when eb%notfound;
unit1:=eb1.c_reading-eb1.p_reading;
  if unit1 >=50 and unit1 <=100 then
      cost1:=unit1*1.50;
elsif  unit1>101 and unit1<=200 then
     cost1:=unit1*2.50;
 else
      cost1:=unit1*3.50;
 end if;
dbms_output.put_line(' '||eb1.cusno||'    '||rpad(eb1.cus_name,15,' ')||'   '||eb1.c_reading||'
'||eb1.p_reading||'   '||ltrim(unit1,' ')||'    '||lpad(cost1,8,' '));
update bill set unit=unit1,totalcost=cost1 where cusno=eb1.cusno;
  end loop;
  dbms_output.put_line('**************');
  close eb;
end;
/
```

OUTPUT

```
  CUS_NO    CUS_NAME    C_READING    P_READING   UNIT   COST
 ***********************************************************************
      1              tom              2000             1578        422       1477
      2              sam              4046             3785        261        913.5
      3              ram              7855             6454       1401       4903.5
      4              pop               4567             3456       1111       3888.5
      5              roy               2345             1234       1111       3888.5
**************************************************************************
```

MASTER-SLAVE RELATIONSHIP

```
Create or replace trigger check_budget_emp
After insert or update of sal,deptno on emp002
Declare
        Cursor dept_cur is
        Select deptno,budget from dept002;
        Dno dept002.deptno%type;
        Allsal dept002 .budget%type;
        Dept _sal number;
Begin
        Open dept_cur;
Loop
        Fetch dept_cur into dno,allsal;
        Exit when dept_cur%notfound;
        Select sum(sal) into dept_sal from emp002 where deptno=dno;
If dept_sal>allsal then
        Raise_application_error(-20001,'total of salaries in department' || to_char(dno) ||
'exceeds budget');
        End if;
End loop;
Close dept_cur;
End;
/
```

OUTPUT

```
SQL> insert into slav_emp values(187,20,300000);
insert into slav_emp values(187,20,300000)
        *
ERROR at line 1:
ORA-20201: total salary exceeds the budget
ORA-06512: at "ABIRAMI.MASTSLAVE", line 12
ORA-04088: error during execution of trigger 'ABIRAMI.MASTSLAVE'
```

INSERTING, MODIFYING AND DELETING RECORDS IN A DATABASE USING TRIGGER

```
Create or replace trigger bookshelf_bef_upd_ins_row
Before insert or update or delete of rating on bookshelf
For each row
Begin
        If  INSERTING then
                Insert into bookshelf_audit
        `       (title,publisher,catergoryname,new_rating,audit_date)
                values
                (:new.title,:new.publilsher,:new.categoryname,:new.rating,sysdate);
elsif UPDATING then
                insert into bookshelf_audit
                (title,publisher,catergoryname,old_rating,new_rating,audit_date)
                values

        (:old.title,:old.publisher,:old.categoryname,:old,rating,:new.rating,sysdate);
else
                insert into bookshelffold(title,publisher,categoryname,rating)
                values
                (:old.tilte,:old.publisher,:old.categoryname,:old.rating);
end if;
end;
/
```

OUTPUT

```
SQL> insert into emp values(103,'lali',12000);
1 row created.

SQL> select *from temp;

EID        ENAME      OSAL       NSAL
---------- ------------     ---------- ----------
104        chitra          10000       19999
103        lali            12000

SQL> update emp set sal =18000
2  where eid=103;

1 row updated.

SQL>  select *from temp;

EID        ENAME    OSAL       NSAL
---------- ------------ ---------- ----------
104        chitra       10000      19999
103        lali         12000      18000

SQL> delete from emp
  2  where eid=103;

1 row deleted.

SQL> select *from demp;
USERID         OPRN          SDATE
-------------- --------------      ---------
ABIRAMI        deleting       17-OCT-10
```

QUERIES ON VIEW & INDEX

1. Create separate views for issued and received items.

QUERY1:

```
Create view vitrans as
Select * from trans
Where ttype='i';

Create view vrtrans as
Select * from trans
Where ttype='r';
```

2.Create a view for items which belongs to class 'a'.

QUERY2:

```
Create view vaclass as
Select * from item
Where icode='a';
```

3.Create a view called ITCLASS which contains minimum,maximum,average rate of items
And quantity in hand.

QUERY3:

```
Create view itclass as
Select icode,count(price) as "QOH", min(price) as "MIN", max(price) as
"MAX",avg(price) as "AVG"
from item group by icode;
```

4.Delete the transactions of item number 100.

QUERY4:

```
Delete from trans
Where ino=100;
```

5.Create index for Date Of Transactions from transaction table.

QUERY5:

```
Create index idot on trans (dot);
```

1. TABLE CONSTRAINTS

```
SQL> create table dept1
  2   (deptno number(2) constraint pk_deptno primary key,
  3   dname  varchar2(10) not null,
  4   loc  varchar2(20));

Table created.

SQL> create table emp1
2  (empno number(5) constraint pk_emp1 primary key,
3  ename varchar2(20) constraint emp1_nn not null,
4  job varchar2(20),
5  mgr number(5),
6  hiredate date,
7  sal number(8,2),
8  comm number(4),
9  deptno number(2) references dept1(deptno) on delete cascade);

Table created.
```

2. DESCRIBE COMMAND

```
SQL> describe dept1

Name    Null?           Type
------------------------------------------------------------------------------
 DEPTNO              NOT NULL        NUMBER(2)
 DNAME               NOT NULL        VARCHAR2(10)
 LOC                                 VARCHAR2(20)
SQL>desc emp1
 Name                              Null?         Type
 ----------------------------------------- -------- ---------------------
 EMPNO                          NOT NULL  NUMBER(5)
 ENAME                          NOT NULL   VARCHAR2(20)
 JOB                                        VARCHAR2(20)
 MGR                                        NUMBER(5)
 HIREDATE                                   DATE
 SAL                                        NUMBER(8,2)
 COMM                                       NUMBER(4)
 DEPTNO                                     NUMBER(2)
```

3. ALTER TABLE COMMANDS

```
SQL> alter table emp1 add(child  number(1));
Table altered.

SQL> alter table emp1 modify (enamel varchar2(30));
```

Table altered.

4. TRUNCATE

```
SQL> truncate table emp1;
Table truncated.
```

5. DROP

```
SQL> drop table emp1;
Table dropped.
```

DATA MANIPULATION LANGUAGE
1. INSERT:

```
 SQL> insert into dept1 values(&deptno,'&dname','&loc');
Enter value for deptno: 10
Enter value for dname: ACCOUNTING
Enter value for loc: NEW YORK
old   2: (&deptno,'&dname','&loc')
new   2: (10,'ACCOUNTING','NEW YORK')
1 row created.

SQL> INSERT INTO DEPT1 VALUES (20,'RESEARCH','DALLAS');
1 row created.
```

2. SIMPLE SELECT STATEMENTS

```
SQL> SELECT * FROM EMP;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30

4. DELETE

```
SQL> DELETE FROM DEPT1 WHERE DNAME='ACCOUNTING';
1 row deleted.
```

VIEWING CONSTRAINTS

```
SQL> SELECT CONSTRAINT_NAME,CONSTRAINT_TYPE,SEARCH_CONDITION
FROM USER_CONSTRAINTS WHERE TABLE_NAME='EMP';

CONSTRAINT_NAME              C SEARCH_CONDITION
---------------------------- - ------------------------------------------------------------------------------
PK_EMP                       P
FK_DEPTNO                    R
NT_ENAME                     C "ENAME" IS NOT NULL
SAL_CHK                      C SAL>500
```

DISABLE AND DROPPING CONSTRAINTS

```
SQL> ALTER TABLE EMP DISABLE CONSTRAINT PK_EMP;
Table altered.

SQL>ALTER TABLE EMP ENABLE CONSTRAINT PK_EMP
    Table altered.
```

DROPPING CONSTRIANTS:

```
SQL> ALTER TABLE EMP ADD(CHILD NUMBER(2));
    Table altered.

SQL> ALTER TABLE EMP DROP COLUMN CHILD;
    Table altered.
```

ADDING THE PRIMARY KEY CONSTRAINTS TO THE EXISTING TABLE

```
SQL> ALTER TABLE EMP ADD CONSTRAINT PK_EMP_EMPNO PRIMARY
KEY(EMPNO);
    Table altered.
```

ADDING REFERENCE BETWEEN THE EXITSING TABLE (FOREIGN KEY CONSTRAINT)

```
SQL>ALTER TABLE EMP ADD CONSTRAINT FK_DEPTNO FOREIGN
KEY(DEPTNO) REFERENCES DEPT(DEPTNO) ON DELETE  CASCADE;
    Table altered.
```

ADDING THE UNIQUE KEY CONSTRAINT TO THE EXISTING TABLE

```
SQL> ALTER TABLE EMP ADD CONSTRAINT EMP1_EMPNO_UNIQ
UNIQUE(ENAME);
        Table altered.
```

ADDING THE CHECK CONSTRAINT TO THE EXISTING TABLE

```
SQL> ALTER TABLE EMP ADD CONSTRAINT SAL_CHK CHECK(SAL>500);
        Table altered.
```

ADDING THE NOT NULL CONSTRAINT TO THE EXISTING TABLE

```
SQL> ALTER TABLE EMP MODIFY (ENAME CONSTRAINT NT_ENAME NOT
NULL);
        Table altered.
```

2.AND , OR , NOT OPERATIONS

THE SELECT STATEMENT WITH ADDITIONAL CLAUSES:

1. SQL > SELECT ENAME,SAL,DEPTNO FROM EMP WHERE JOB='SALESMAN' AND DEPTNO=30;

ENAME	SAL	DEPTNO
ALLEN	1600	30
WARD	1250	30
MARTIN	1250	30
TURNER	1500	30

2. SQL> SELECT EMPNO,ENAME,JOB,SAL,DEPTNO FROM EMP WHERE DEPTNO=30 OR HIREDATE='07-DEC-80';

EMPNO	ENAME	JOB	SAL	DEPTNO
7499	ALLEN	SALESMAN	1600	30
7521	WARD	SALESMAN	1250	30
7654	MARTIN	SALESMAN	1250	30
7698	BLAKE	MANAGER	2850	30
7844	TURNER	SALESMAN	1500	30
7900	JAMES	CLERK	950	30

3. SQL> SELECT ENAME,JOB FROM EMP WHERE JOB IN('MANAGER','PRESIDENT');

ENAME	JOB
JONES	MANAGER
BLAKE	MANAGER
CLARK	MANAGER
KING	PRESIDENT

4. SQL> SELECT * FROM EMP WHERE SAL BETWEEN 1000 AND 2000;

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO

7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

5. SQL> SELECT DISTINCT STATE FROM CUSTOMERS;

```
ST
--
NJ
CA
WY
MA
GA
IL
MI
NY
FL
ID
WA
TX
```

6. SQL> SELECT UNIQUE SAL FROM EMP;

```
       SAL
----------
      2450
      5000
      1300
      1250
      2850
      2975
      1100
      3000
       800
      1600
      1500
       950
```

USING CONCATENATION OPERATOR:

1. SQL> SELECT FIRSTNAME ||' '||LASTNAME FROM CUSTOMERS;

FIRSTNAME||"||LASTNAM E

BONITA MORALES
RYAN THOMPSON
LEILA SMITH
THOMAS PIERSON
CINDY GIRARD
MESHIA CRUZ
TAMMY GIANA
USING COLUMN ALIAS:

1. SQL> SELECT TITLE AS "TITLE OF THE BOOKS",CATEGORY FROM BOOKS;

TITLE OF THE BOOKS	CATEGORY
BODYBUILD IN 10 MINUTES A DAY	FITNESS
REVENGE OF MICKEY	FAMILY LIFE
BUILDING A CAR WITH TOOTHPICKS	CHILDREN
DATABASE IMPLEMENTATION	COMPUTER
COOKING WITH MUSHROOMS	COOKING
HOLY GRAIL OF ORACLE	COMPUTER
HANDCRANKED COMPUTERS	COMPUTER
E-BUSINESS THE EASY WAY	COMPUTER
PAINLESS CHILD REARING	FAMILY LIFE
BIG BEAR AND LITTLE DOVE	CHILDREN
HOW TO GET FASTER PIZZA	SELF HELP
HOW TO MANAGE THE MANAGER	BUSINESS
SHORTEST POEMS	LITERATURE
THE WOK WAY TO COOK	COOKING

2. SQL> SELECT FIRSTNAME||' '||LASTNAME "CUSTOMER NAME" FROM CUSTOMERS;

CUSTOMER NAME

BONITA MORALES
RYAN THOMPSON
LEILA SMITH
THOMAS PIERSON
CINDY GIRARD
MESHIA CRUZ
TAMMY GIANA
KENNETH JONES

USING ARITHMATIC OPERATOR:

1. SQL> SELECT TITLE,RETAIL-COST"PROFIT" FROM BOOKS;

TITLE	PROFIT
BODYBUILD IN 10 MINUTES A DAY	12.2
REVENGE OF MICKEY	7.8
BUILDING A CAR WITH TOOTHPICKS	22.15
DATABASE IMPLEMENTATION	24.55
COOKING WITH MUSHROOMS	7.45
HOLY GRAIL OF ORACLE	28.7
HANDCRANKED COMPUTERS	3.2

PATTERN MATCHING OPERATOR (LIKE)

1. SQL> SELECT ENAME,JOB,DEPTNO FROM EMP WHERE ENAME LIKE 'M%';

ENAME	JOB	DEPTNO
MARTIN	SALESMAN	30
MILLER	CLERK	10

2. SQL> SELECT * FROM EMP WHERE ENAME LIKE 'ALL_N';

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30

3. SQL> SELECT CUSTOMER#,LASTNAME,FIRSTNAME FROM CUSTOMERS
WHERE CUSTOMER# LIKE '10_9';

CUSTOMER#	LASTNAME	FIRSTNAME
1009	PAREZ	JORGE
1019	SMITH	JENNIFER

3. UNION , INTERSECTION , MINUS AND JOIN OPERATIONS

1. UNION

1. SQL>select deptno from emp union select deptno from dept;

```
   DEPTNO
----------
        10
        20
        30
        40
```

2. UNION ALL

1. SQL> select deptno from emp union all select deptno from dept;

```
   DEPTNO
----------
        20
        30
        30
        20
        30
        30
        10
        20
        20
        10
        10
        20
        30
        40
```

3. INTERSECT

1. SQL> select deptno from emp intersect select deptno from dept;

```
   DEPTNO
----------
        10
        20
        30
```

4. MINUS

1. SQL> select deptno from dept minus select deptno from emp;

```
   DEPTNO
----------
        40
```

4. EQUI JOIN

1. SQL> SELECT ENAME,EMPNO,DNAME,DEPT.DEPTNO,LOC FROM EMP,DEPT WHERE EMP.DEPTNO=DEPT.DEPTNO;

ENAME	EMPNO	DNAME	DEPTNO	LOC
SMITH	7369	RESEARCH	20	DALLAS
ALLEN	7499	SALES	30	CHICAGO
WARD	7521	SALES	30	CHICAGO
JONES	7566	RESEARCH	20	DALLAS
MARTIN	7654	SALES	30	CHICAGO
BLAKE	7698	SALES	30	CHICAGO
CLARK	7782	ACCOUNTING	10	NEW YORK
SCOTT	7788	RESEARCH	20	DALLAS
KING	7839	ACCOUNTING	10	NEW YORK
TURNER	7844	SALES	30	CHICAGO
ADAMS	7876	RESEARCH	20	DALLAS
JAMES	7900	SALES	30	CHICAGO
FORD	7902	RESEARCH	20	DALLAS
MILLER	7934	ACCOUNTING	10	NEW YORK

14 rows selected.

COLUMN OUALIFIER

SQL> SELECT TITLE,BOOKS.PUBID,NAME FROM PUBLISHER,BOOKS WHERE PUBLISHER.PUBID=BOOKS.PUBID AND PUBLISHER.PUBID=4;

TITLE	PUBID	NAME
BODYBUILD IN 10 MINUTES A DAY	4	READING MATERIALS INC
COOKING WITH MUSHROOMS	4	READING MATERIALS INC
HOW TO GET FASTER PIZZA	4	READING MATERIALS INC
THE WOK WAY TO COOK	4	READING MATERIALS INC

TABLE ALIAS

SQL> SELECT TITLE,P.PUBID,NAME FROM PUBLISHER P,BOOKS B WHERE P.PUBID=B.PUBID;

TITLE	PUBID	NAME
BODYBUILD IN 10 MINUTES A DAY	4	READING MATERIALS INC
REVENGE OF MICKEY	1	PRINTING IS US
BUILDING A CAR WITH TOOTHPICKS	2	PUBLISH OUR WAY
DATABASE IMPLEMENTATION	3	AMERICAN PUBLISHING
COOKING WITH MUSHROOMS	4	READING MATERIALS INC
HOLY GRAIL OF ORACLE	3	AMERICAN PUBLISHING
HANDCRANKED COMPUTERS	3	AMERICAN PUBLISHING
E-BUSINESS THE EASY WAY	2	PUBLISH OUR WAY
PAINLESS CHILD REARING	5	REED-N-RITE
BIG BEAR AND LITTLE DOVE	5	REED-N-RITE
HOW TO GET FASTER PIZZA	4	READING MATERIALS INC
HOW TO MANAGE THE MANAGER	1	PRINTING IS US
SHORTEST POEMS	5	REED-N-RITE
THE WOK WAY TO COOK	4	READING MATERIALS INC

EQUALITY JOIN-JOIN METHODS
NATURAL JOIN:

1. SQL> SELECT EMPNO,ENAME,SAL,DEPTNO FROM EMP NATURAL JOIN DEPT;

EMPNO	ENAME	SAL	DEPTNO
7369	SMITH	800	20
7499	ALLEN	1600	30
7521	WARD	1250	30
7566	JONES	2975	20
7654	MARTIN	1250	30
7698	BLAKE	2850	30
7782	CLARK	2450	10
7788	SCOTT	3000	20
7839	KING	5000	10
7844	TURNER	1500	30
7876	ADAMS	1100	20
7900	JAMES	950	30
7902	FORD	3000	20
7934	MILLER	1300	10

JOIN USING:

1. SQL> SELECT EMPNO,ENAME,SAL,DEPTNO FROM EMP JOIN DEPT USING(DEPTNO);

EMPNO	ENAME	SAL	DEPTNO
7369	SMITH	800	20
7499	ALLEN	1600	30
7521	WARD	1250	30
7566	JONES	2975	20
7654	MARTIN	1250	30
7698	BLAKE	2850	30
7782	CLARK	2450	10
7788	SCOTT	3000	20
7839	KING	5000	10
7844	TURNER	1500	30
7876	ADAMS	1100	20
7900	JAMES	950	30
7902	FORD	3000	20
7934	MILLER	1300	10

JOIN ON:

1. SQL> SELECT EMPNO,ENAME,SAL,DEPT.DEPTNO FROM EMP JOIN DEPT ON EMP.DEPTNO=DEPT.DEPTNO;

EMPNO	ENAME	SAL	DEPTNO
7369	SMITH	800	20
7499	ALLEN	1600	30
7521	WARD	1250	30
7566	JONES	2975	20
7654	MARTIN	1250	30
7698	BLAKE	2850	30
7782	CLARK	2450	10
7788	SCOTT	3000	20
7839	KING	5000	10
7844	TURNER	1500	30

7. SELF JOIN

1. SQL> SELECT X.ENAME,X.SAL,X.JOB,Y.ENAME,Y.SAL,Y.JOB FROM EMP X,EMP Y WHERE X.SAL > Y.SAL AND Y.ENAME='JONES';

ENAME	SAL	JOB	ENAME	SAL	JOB
SCOTT	3000	ANALYST	JONES	2975	MANAGER
KING	5000	PRESIDENT	JONES	2975	MANAGER
FORD	3000	ANALYST	JONES	2975	MANAGER

2. SQL> SELECT R.FIRSTNAME,R.LASTNAME,C.LASTNAME,C.REFERRED FROM CUSTOMERS C,CUSTOMERS R WHERE C.REFERRED=R.CUSTOMER#;

FIRSTNAME	LASTNAME	LASTNAME	REFERRED
LEILA	SMITH	GIANA	1003
LEILA	SMITH	PAREZ	1003
MESHIA	CRUZ	NGUYEN	1006
JAKE	LUCAS	DAUM	1010
LEILA	SMITH	SMITH	1003

8. NON-EQUI JOIN

1. SQL> SELECT TITLE,GIFT FROM BOOKS,PROMOTION WHERE RETAIL BETWEEN MINRETAIL AND MAXRETAIL;

TITLE	GIFT
BIG BEAR AND LITTLE DOVE	BOOKMARKER
REVENGE OF MICKEY	BOOKLABELS
COOKING WITH MUSHROOMS	BOOKLABELS
HANDCRANKED COMPUTERS	BOOKLABELS
BODYBUILD IN 10 MINUTES A DAY	BOOKCOVER
DATABASE IMPLEMENTATION	BOOKCOVER
E-BUSINESS THE EASY WAY	BOOKCOVER
HOW TO GET FASTER PIZZA	BOOKCOVER
HOW TO MANAGE THE MANAGER	BOOKCOVER
SHORTEST POEMS	BOOKCOVER
THE WOK WAY TO COOK	BOOKCOVER
BUILDING A CAR WITH TOOTHPICKS	FREE SHIPPING
HOLY GRAIL OF ORACLE	FREE SHIPPING
PAINLESS CHILD REARING	FREE SHIPPING

9. OUTER JOIN (TRADITIONAL METHOD)

1. SQL> SELECT D.*,E.JOB,E.ENAME FROM DEPT D,EMP E WHERE D.DEPTNO=E.DEPTNO(+);

DEPTNO	DNAME	LOC	JOB	ENAME
10	ACCOUNTING	NEW YORK	MANAGER	CLARK
10	ACCOUNTING	NEW YORK	PRESIDENT	KING
10	ACCOUNTING	NEW YORK	CLERK	MILLER
20	RESEARCH	DALLAS	MANAGER	JONES
20	RESEARCH	DALLAS	ANALYST	FORD
20	RESEARCH	DALLAS	CLERK	ADAMS
20	RESEARCH	DALLAS	CLERK	SMITH
20	RESEARCH	DALLAS	ANALYST	SCOTT
30	SALES	CHICAGO	SALESMAN	WARD
30	SALES	CHICAGO	SALESMAN	TURNER
30	SALES	CHICAGO	SALESMAN	ALLEN
30	SALES	CHICAGO	CLERK	JAMES
30	SALES	CHICAGO	MANAGER	BLAKE
30	SALES	CHICAGO	SALESMAN	MARTIN
40	OPERATIONS	BOSTON		

10. OUTER JOIN-JOIN METHOD

1. SQL> SELECT LASTNAME,FIRSTNAME,ORDER# FROM CUSTOMERS C LEFT OUTER JOIN ORDERS O ON C.CUSTOMER#=O.CUSTOMER# ORDER BY C.CUSTOMER#

LASTNAME	FIRSTNAME	ORDER#
MORALES	BONITA	1018
MORALES	BONITA	1003
THOMPSON	RYAN	
PIERSON	THOMAS	1008
GIRARD	CINDY	1009
GIRARD	CINDY	1000
CRUZ	MESHIA	
GIANA	TAMMY	1014
JONES	KENNETH	1020
PAREZ	JORGE	
LUCAS	JAKE	1001
MCGOVERN	REESE	1002
MCKENZIE	WILLIAM	
NGUYEN	NICHOLOS	
LEE	JASMINE	1013
SCHELL	STEVE	1017
DAUM	MICHELL	

2. SQL>SELECT LASTNAME,FIRSTNAME,ORDER# FROM CUSTOMERS C RIGHT OUTER JOIN ORDERS O ON C.CUSTOMER#=O.CUSTOMER# ORDER BY C.CUSTOMER#

LASTNAME	FIRSTNAME	ORDER#
MORALES	BONITA	1018
MORALES	BONITA	1003
SMITH	LEILA	1016
SMITH	LEILA	1006
PIERSON	THOMAS	1008
GIRARD	CINDY	1009
GIRARD	CINDY	1000
GIANA	TAMMY	1007
GIANA	TAMMY	1014
JONES	KENNETH	1020
LUCAS	JAKE	1001
LUCAS	JAKE	1011
MCGOVERN	REESE	1002

LEE	JASMINE	1013
SCHELL	STEVE	1017
NELSON	BECCA	1012
MONTISSA	GREG	1005
MONTISSA	GREG	1019
SMITH	JENNIFER	1010
FALAH	KENNETH	1004
FALAH	KENNETH	1015

3. SQL> SELECT LASTNAME,FIRSTNAME,ORDER# FROM CUSTOMERS C FULL OUTER JOIN ORDERS O ON C.CUSTOMER#=O.CUSTOMER# ORDER BY C.CUSTOMER#

LASTNAME	FIRSTNAME	ORDER#
MORALES	BONITA	1018
MORALES	BONITA	1003
THOMPSON	RYAN	
SMITH	LEILA	1016
PIERSON	THOMAS	1008
GIRARD	CINDY	1009
CRUZ	MESHIA	
GIANA	TAMMY	1007
JONES	KENNETH	1020
PAREZ	JORGE	
LUCAS	JAKE	1001

11. CARTESIAN JOIN

SQL> SELECT * FROM EMP;

EMPNO	ENAME	JOB	SAL	DNO
1000	THIYAGU	SALESMAN	2500	10
1001	VIGNESH	ANALYST	3000	10
1002	DHANUSH	CLERK	2000	20
1003	PRAVEEN	PRESIDENT	5000	30
1004	SIRAJ	MANAGER	4000	20

SQL> SELECT * FROM DEPT;

DEPTNO	DNAME	DLOC
10	ACCOUNTING	BOSTON
20	RESEARCH	DALLAS

30 HEADQUATERS HOUSTON

SQL> SELECT EMPNO,ENAME,DNAME FROM EMP,DEPT;

EMPNO	ENAME	DNAME
1000	THIYAGU	ACCOUNTING
1001	VIGNESH	ACCOUNTING
1002	DHANUSH	ACCOUNTING
1003	PRAVEEN	ACCOUNTING
1004	SIRAJ	ACCOUNTING
1000	THIYAGU	RESEARCH
1001	VIGNESH	RESEARCH
1002	DHANUSH	RESEARCH
1003	PRAVEEN	RESEARCH
1004	SIRAJ	RESEARCH
1000	THIYAGU	HEADQUATERS
1001	VIGNESH	HEADQUATERS
1002	DHANUSH	HEADQUATERS
1003	PRAVEEN	HEADQUATERS
1004	SIRAJ	HEADQUATERS

SQL>SELECT EMPNO,ENAME,DNAME FROM EMP CROSS JOIN DEPT

EMPNO	ENAME	DNAME
1000	THIYAGU	ACCOUNTING
1001	VIGNESH	ACCOUNTING
1002	DHANUSH	ACCOUNTING
1003	PRAVEEN	ACCOUNTING
1004	SIRAJ	ACCOUNTING
1000	THIYAGU	RESEARCH
1001	VIGNESH	RESEARCH
1002	DHANUSH	RESEARCH
1003	PRAVEEN	RESEARCH
1004	SIRAJ	RESEARCH
1000	THIYAGU	HEADQUATERS
1001	VIGNESH	HEADQUATERS

1002	DHANUSH	HEADQUATERS
1003	PRAVEEN	HEADQUATERS
1004	SIRAJ	HEADQUATERS

15 rows selected.

4. SORTING AND GROUPING

1. ORDER BY(ASCENDING AND DESCENDING ORDER)

1. SQL>select ename,sal from emp order by enameasc;

ENAME	SAL
ADAMS	1120
ALLEN	1600
BLAKE	2850
CLARK	2450
FORD	3000
JAMES	970

JONES 2975
KING 5000
MARTIN 1250
MILLER 1320

2. SQL> SELECT ENAME,COMM,SAL FROM EMP ORDER BY ENAME;

ENAME	COMM	SAL
ADAMS		1120
ALLEN	300	1600
BLAKE		2850
JONES		2975
KING		5000
MARTIN	1400	1250
MILLER		1320
SMITH		820
TURNER	0	1500
WARD	500	1250

3. SQL> SELECT COMM FROM EMP WHERE COMM IS NOT NULL ORDER BY COMM DESC;

COMM
1400
500
300
0

4. SQL>SELECT ENAME,SAL,COMM FROM EMP ORDER BY COMM DESC NULLS LAST;

ENAME	SAL	COMM
MARTIN	1250	1400
WARD	1250	500
ALLEN	1600	300
TURNER	1500	0
SCOTT	3000	
KING	5000	
ADAMS	1120	
JAMES	970	

5. SQL> SELECT DISTINCT SAL FROM EMP ORDER BY SAL ASC;

```
       SAL
----------
       820
       970
      1120
      1250
      1320
      1500
      1600
      2450
      2850
      2975
      3000
      5000
```

POSITIONAL SORTING:

1. SQL> SELECT ENAME,SAL,HIREDATE FROM EMP ORDER BY 3;

```
ENAME           SAL    HIREDATE
-------------------------------------------------
SMITH           820        17-DEC-80
ALLEN          1600       20-FEB-81
WARD           1250       22-FEB-81
BLAKE          2850       01-MAY-81
CLARK          2450       09-JUN-81
TURNER          1500       08-SEP-81
MARTIN         1250       28-SEP-81
KING           5000       17-NOV-81
```

2. SQL> SELECT ENAME,SAL,HIREDATE FROM EMP ORDER BY 2 DESC;

```
ENAME         SAL     HIREDATE
-----------------------------------------------------
KING           5000      17-NOV-81
FORD           3000       03-DEC-81
SCOTT          3000      19-APR-87
JONES          2975       02-APR-81
BLAKE         2850      01-MAY-81
CLARK         2450      09-JUN-81
ALLEN         1600      20-FEB-81
TURNER       1500      08-SEP-81
MILLER        1320      23-JAN-82
```

COMPOSITE SORTING
SQL>select job,ename,sal,hiredate from emp order by job desc,ename,3 desc;

JOB	ENAME	SAL	HIREDATE
SALESMAN	ALLEN	1600	20-FEB-81
SALESMAN	MARTIN	1250	28-SEP-81
PRESIDENT	KING	5000	17-NOV-81
MANAGER	BLAKE	2850	01-MAY-81
CLERK	MILLER	1320	23-JAN-82
CLERK	SMITH	820	17-DEC-80
ANALYST	FORD	3000	03-DEC-81
ANALYST	SCOTT	3000	19-APR-87

2. GROUP BY
1. SQL> SELECT DEPTNO,AVG(SAL) FROM EMP GROUP BY DEPTNO;

DEPTNO	AVG(SAL)
30	1570
20	2183
10	2923.33333

2. SQL> SELECT DEPTNO FROM EMP WHERE JOB='CLERK' GROUP BY DEPTNO;

DEPTNO
30
20
10

3. HAVING CLAUSE

1. SQL> SELECT DEPTNO FROM EMP WHERE JOB='ANALYST' GROUP BY DEPTNO HAVING COUNT(*)>=2;

DEPTNO
20

5. NESTED QUERIES USING SQL

1. SINGLE ROW SUBQUERY:

1. Display the names of employee who working with JONES with same job

SQL> Select ename,job from emp where job = (SELECT job from emp where ename = 'JONES');

ENAME	JOB

JONES MANAGER
BLAKE MANAGER
CLARK MANAGER

2. List all computer books that have a higher cost than DATABASE IMPLEMENTATION

SQL> Select title,cost from books where cost > (select cost from books where title = 'DATABASE IMPLEMENTATION') and category = 'COMPUTER';

TITLE	COST
HOLY GRAIL OF ORACLE	47.25
E-BUSINESS THE EASY WAY	37.9

3. To retrieve the title of most expensive books. The subquery return highest price book and the outer query return results of title

SQL> Select title from books where retail = (Select max(retail) from books);

TITLE
PAINLESS CHILD REARING

a) Multiple single row subquries in select statement:

SQL> Select isbn,title from BOOKS where pubid = (select pubid from BOOKS where title ='BIG BEAR AND LITTLE DOVE') and retail-cost >(select avg(retail-cost) from BOOKS);

ISBN	TITLE
2491748320	PAINLESS CHILD REARING
2147428890	SHORTEST POEMS

b) Single row subquery in a having clause:

selectcategory,avg(retail-cost)"average profit" from books group by category having avg(retail-cost)>(select avg(retail-cost) from books where category='LITERATURE');

CATEGORY	average profit
COMPUTER	18.2625
FAMILY LIFE	24.875

c) Single row subquery in a select clause

1. Suppose that management like to compare the price of each book in inventory against the average price of all books in inventory

SQL> SELECT TITLE,RETAIL,(SELECT AVG(RETAIL) FROM BOOKS)"OVERALL AVERAGE" FROM BOOKS;

TITLE	RETAIL	OVERALL AVERAGE
BODYBUILD IN 10 MINUTES A DAY	30.95	40.9821429
REVENGE OF MICKEY	22	40.9821429
BUILDING A CAR WITH TOOTHPICKS	59.95	40.9821429
DATABASE IMPLEMENTATION	55.95	40.9821429
COOKING WITH MUSHROOMS	19.95	40.9821429
HOLY GRAIL OF ORACLE	75.95	40.9821429
HANDCRANKED COMPUTERS	25	40.9821429
E-BUSINESS THE EASY WAY	54.5	40.9821429
PAINLESS CHILD REARING	89.95	40.9821429
BIG BEAR AND LITTLE DOVE	8.95	40.9821429

d) Single-row subquery in a FROM clause

1. Display top three highest rated books from JUSTLEE database.

SQL> SELECT RETAIL,TITLE FROM (SELECT RETAIL,TITLE FROM BOOKS ORDER BY RETAIL DESC)WHERE ROWNUM<4;

RETAIL	TITLE
89.95	PAINLESS CHILD REARING
75.95	HOLY GRAIL OF ORACLE
59.95	BUILDING A CAR WITH TOOTHPICKS

SQL> Select title from books where retail = (Select max(retail) from books);

TITLE
PAINLESS CHILD REARING

2. MULTIPLE ROW SUB-QUERY

IN- The commonly used operator

1. SQL> SELECT EMPNO,ename,job from emp where job in (SELECT job from emp where deptno = 10);

EMPNO	ENAME	JOB
7782	CLARK	MANAGER
7698	BLAKE	MANAGER
7566	JONES	MANAGER
7839	KING	PRESIDENT
7934	MILLER	CLERK
7900	JAMES	CLERK
7876	ADAMS	CLERK
7369	SMITH	CLERK

NOT IN

1. SQL> SELECT EMPNO,ename,job from emp where deptno<>30 and job not in (SELECT job from emp where deptno = 10);

EMPNO	ENAME	JOB
7788	SCOTT	ANALYST
7902	FORD	ANALYST

ANY AND ALL OPERATOR:

1. SQL> SELECT EMPNO,ename, job from emp where deptno<>30 and job = any (SELECT job from emp where deptno = 10);

EMPNO	ENAME	JOB
7782	CLARK	MANAGER
7566	JONES	MANAGER
7839	KING	PRESIDENT
7934	MILLER	CLERK
7876	ADAMS	CLERK
7369	SMITH	CLERK

2. Display the titles of all books having a retail price greater than the most expensive in the cooking category
SQL> Select title,retail from books where retail > ALL(Select retail from books where category = 'COOKING');

TITLE	RETAIL

BODYBUILD IN 10 MINUTES A DAY	30.95
BUILDING A CAR WITH TOOTHPICKS	59.95
DATABASE IMPLEMENTATION	55.95
HOLY GRAIL OF ORACLE	75.95
E-BUSINESS THE EASY WAY	54.5
PAINLESS CHILD REARING	89.95
HOW TO GET FASTER PIZZA	29.95
HOW TO MANAGE THE MANAGER	31.95
SHORTEST POEMS	39.95

CORRELATED SUBQUERIES

1. List all employees whose salary is less than average salary of department

SQL>select p.ename,p.deptno,P.SAL from emp p where p.sal<(select avg(s.sal) from emp s where s.deptno =p.DEPTNO);

ENAME	DEPTNO	SAL
SMITH	20	900
WARD	30	1250
MARTIN	30	1250
CLARK	10	2450
TURNER	30	1500
ADAMS	20	1200
JAMES	30	1050
MILLER	10	1400

2. Display each the book ordered in the orderitem table using correlated sub query

SQL> SELECT TITLE FROM BOOKS WHERE EXISTS (SELECT ISBN FROM ORDERITEMS WHERE BOOKS.ISBN=ORDERITEMS.ISBN);

TITLE

REVENGE OF MICKEY
SHORTEST POEMS
PAINLESS CHILD REARING
COOKING WITH MUSHROOMS
HOLY GRAIL OF ORACLE
BIG BEAR AND LITTLE DOVE
DATABASE IMPLEMENTATION
HOW TO MANAGE THE MANAGER
NESTED SUB QUERY

1. DISPLAY THE SECOND LARGEST SALARY FROM THE EMPLOYEES USING EMP TABLE

SQL> Select ename,sal from emp where sal in (select max(sal) from emp where sal<(select max(sal) from emp));

```
ENAME           SAL
------------------------------
KING            5000
```

2. To find the name of the customer who was ordeed the most books from Justlee books

SQL> SELECT CUSTOMER#,LASTNAME,FIRSTNAME FROM CUSTOMERS NATURAL JOIN ORDERS WHERE ORDER# IN(SELECT ORDER# FROM ORDERITEMS NATURAL JOIN ORDERS GROUP BY ORDER# HAVING COUNT(*) IN(SELECT MAX(COUNT(*)) FROM ORDERITEMS GROUP BY ORDER#));

```
 CUSTOMER#       LASTNAME      FIRSTNAME
--------------------------------------------------------------------------------
      1007              GIANA            TAMMY
      1017              NELSON           BECCA
```

3. Display the third highest retail price from the books using books table
SQL> SELECT TITLE,RETAIL FROM BOOKS WHERE RETAIL IN(SELECT MAX(RETAIL) FROM BOOKS WHERE RETAIL IN(SELECT MAX(RETAIL) FROM BOOKS WHERE RETAIL IN(SELECT MAX(RETAIL) FROM BOOKS)));

```
TITLE                                    RETAIL
-------------------------------------------------------------
PAINLESS CHILD REARING             89.95
```

6. BUILT IN FUNCTIONS

1. CHARACTER FUNCTION
A) ASCII
SQL> select ascii('A') from dual;

ASCII('A')

65

B) CHR(C)
SQL> select chr(75) from dual;

C

K

C) CONCAT(C1,C2)
1. SQL> select concat('ram','chandran') from dual;

CONCAT('RAM
---
ramchandran

2. SQL> SELECT ENAME,CONCAT('EMPLOYEE NUMBER:',EMPNO)"NUMBER"
FROM EMP;

ENAME NUMBER

SMITH EMPLOYEE NUMBER:7369
ALLEN EMPLOYEE NUMBER:7499
WARD EMPLOYEE NUMBER:7521
JONES EMPLOYEE NUMBER:7566
MARTIN EMPLOYEE NUMBER:7654

D) UPPER
SQL> SELECT UPPER('sastra') from dual;

UPPER(

SASTRA

E) LOWER
SQL> select lower('SASTRA') FROM DUAL;

LOWER(
---
sastra

F) INITCAP
SQL> SELECT INITCAP('india is our country') from dual;

```
INITCAP('INDIAISOURC
------------------------------------
India Is Our Country
```

G) LENGTH
SQL> select length('sastra') from dual;

```
LENGTH('SASTRA')
-----------------------------
             6
```

H) LPAD
SQL> select lpad(firstname,12,'*') from customers

```
LPAD(FIRSTNA
--------------------
******BONITA
********RYAN
*******LEILA
******THOMAS
*******CINDY
```

I) RPAD
SQL>select rpad(firstname,12,'*') from customers;

```
RPAD(FIRSTNA
---------------------
BONITA******
RYAN********
LEILA*******
THOMAS******
CINDY*******
MESHIA******
```

J) LTRIM
SQL> select firstname,lastname,ltrim(address,'p.o box') from customers;

```
FIRSTNAME      LASTNAME            LTRIM(ADDRESS,'P.OBO
-------------------------------------------------------------------------------
BONITA         MORALES                P.O BOX 651
RYAN           THOMPSON                P.O BOX 9835
LEILA          SMITH                   P.O BOX 66
THOMAS         PIERSON                 69821 SOUTHAVENUE
CINDY          GIRARD                  P.O BOX 851
```

MESHIA CRUZ 82 DIRT ROAD

K) REPLACE

SQL> select replace(address,'po','post office') from customers;

REPLACE(ADDRESS,'PO','POSTOFFICE')

P.O BOX 651
P.O BOX 9835
P.O BOX 66
69821 SOUTHAVENUE
P.O BOX 851
82 DIRT ROAD
9153 MAIN STREET
P.O BOX 137
P.O BOX 8564
114 EAST SAVANNAH
P.O BOX 18

L) SUBSTR

1.SQL>select distinct zip,substr(zip,1,3) from customers;

ZIP SUB

32328 323
2110 211
82414 824
83707 837
12211 122
82003 820
91510 915
33111 331

2.SQL>select distinct substr(zip,1,3),substr(zip,-3,2),zip from customers

SUB SU ZIP
---
323 30 32306
837 70 83707
820 00 82003
796 96 7962
211 11 2110
981 11 98115
606 60 60606
347 71 34711

NUMERIC FUNCTION

A) ABS(N)

```
SQL> select abs(-15) from dual;

ABS(-15)
----------
        15
```

B) CEIL(N)

```
SQL> select ceil(15.7) from dual;
CEIL(15.7)
----------
        16
```

C) FLOOR

```
SQL> select floor(15.7) from dual;

FLOOR(15.7)
----------
        15
```

D) POWER

```
SQL> select power(3,3) from dual;

POWER(3,3)
----------
        27
```

E) SQRT

```
SQL> select sqrt(729) from dual;

SQRT(729)
----------
        27
```

F) ROUND

```
SQL> select title,retail,round(retail) from books;

TITLE                                    RETAIL     ROUND(RETAIL)
---------------------------------------- ---------- -------------
BODYBUILD IN 10 MINUTES A DAY            30.95              31
REVENGE OF MICKEY                        22                 22
BUILDING A CAR WITH TOOTHPICKS           59.95              60
```

DATABASE IMPLEMENTATION	55.95	56
COOKING WITH MUSHROOMS	19.95	20
HOLY GRAIL OF ORACLE	75.95	76
HANDCRANKED COMPUTERS	25	25
E-BUSINESS THE EASY WAY	54.5	55

G) TRUNCATE

SQL>select title,retail,trunc(retail,1) from books

TITLE	RETAIL	TRUNC(RETAIL,1)
BODYBUILD IN 10 MINUTES A DAY	30.95	30.9
REVENGE OF MICKEY	22	22
BUILDING A CAR WITH TOOTHPICKS	59.95	59.9
DATABASE IMPLEMENTATION	55.95	55.9
COOKING WITH MUSHROOMS	19.95	19.9
HOLY GRAIL OF ORACLE	75.95	75.9
HANDCRANKED COMPUTERS	25	25

DATE FUNCTIONS

A) MONTHS_BETWEEN

SQL> select title,months_between(orderdate,pubdate) from books natural join orders natural join orderitems where order#=1009;

TITLE	MONTHS_BETWEEN(ORDERDATE,PUBDATE)
BODYBUILD IN 10 MINUTES A DAY	26.4193548
BODYBUILD IN 10 MINUTES A DAY	26.4193548
REVENGE OF MICKEY	15.6451613
REVENGE OF MICKEY	15.6451613
BUILDING A CAR WITH TOOTHPICKS	12.516129
BUILDING A CAR WITH TOOTHPICKS	12.516129
DATABASE IMPLEMENTATION	45.9677419
DATABASE IMPLEMENTATION	45.9677419
COOKING WITH MUSHROOMS	37.1935484
COOKING WITH MUSHROOMS	37.1935484
HOLY GRAIL OF ORACLE	15.0967742
HOLY GRAIL OF ORACLE	15.0967742

B) ADD_MONTHS

SQL> select title,pubdate,add_months(pubdate,60)"drop date" from books;

TITLE	PUBDATE	drop date
BODYBUILD IN 10 MINUTES A DAY	21-JAN-01	21-JAN-06
REVENGE OF MICKEY	14-DEC-01	14-DEC-06
BUILDING A CAR WITH TOOTHPICKS	18-MAR-02	18-MAR-07
DATABASE IMPLEMENTATION	04-JUN-99	04-JUN-04

COOKING WITH MUSHROOMS	28-FEB-00	28-FEB-05
HOLY GRAIL OF ORACLE	31-DEC-01	31-DEC-06
HANDCRANKED COMPUTERS	21-JAN-01	21-JAN-06

C) NEXT DAY

```
SQL> select next_day(orderdate,'MONDAY') FROM ORDERS;

NEXT_DAY(
-----------------------
07-APR-03
07-APR-03
07-APR-03
07-APR-03
07-APR-03
07-APR-03
```

D) TO_DATE

```
SQL> SELECT ORDER#,ORDERDATE,SHIPDATE FROM ORDERS WHERE
ORDERDATE=TO_DATE('MARCH 31,2003',’MONTH DD,YY');
```

ORDER#	ORDERDATE	SHIPDATE
1000	31-MAR-03	02-APR-03
1001	31-MAR-03	01-APR-03
1002	31-MAR-03	01-APR-03

E) TO_CHAR

```
SQL> SELECT TITLE,TO_CHAR(PUBDATE,'MONTH DD,YYYY')"PUBLICATION
DATE",TO_CHAR(RETAIL,'$999.99')"RETAIL PRICE" FROM BOOKS;
```

TITLE	PUBLICATION DATE	RETAIL
BODYBUILD IN 10 MINUTES A DAY	JANUARY 21,2001	$30.95
REVENGE OF MICKEY	DECEMBER 14,2001	$22.00
BUILDING A CAR WITH TOOTHPICKS	MARCH 18,2002	$59.95
DATABASE IMPLEMENTATION	JUNE 04,1999	$55.95
COOKING WITH MUSHROOMS	FEBRUARY 28,2000	$19.95
HOLY GRAIL OF ORACLE	DECEMBER 31,2001	$75.95
HANDCRANKED COMPUTERS	JANUARY 21,2001	$25.00
E-BUSINESS THE EASY WAY	MARCH 01,2002	$54.50
PAINLESS CHILD REARING	JULY 17,2000	$89.95

F) DECODE

```
SQL> SELECT CUSTOMER#,STATE,DECODE(STATE,'CA',.08,'FL',.07,0)"SALES TAX
RATE" FROM CUSTOMERS;

 CUSTOMER#     ST SALES        TAX RATE
-----------------------------------------------------------------
      1001                    FL          .07
      1002                    CA          .08
      1003                    FL          .07
      1004                    ID          0
      1005                    WA           0
      1006                    NY           0
      1007                    TX           0
      1008                    WY           0
      1009                    CA          .08
```

G) NVL FUNCTION

```
SQL> SELECT ENAME,EMPNO,SAL+NVL(COMM,0) FROM EMP;

ENAME       EMPNO      SAL+NVL(COMM,0)
-----------------------------------------------------------------
SMITH           7369            820
ALLEN           7499            1900
WARD            7521            1750
JONES           7566           2975
MARTIN          7654            2650
BLAKE           7698            2850
CLARK           7782            2450
SCOTT          7788           3000
KING            7839            5000
```

GROUP FUNCTION

A) AVG([DISTINCT/ALL])

```
SQL> SELECT AVG(SAL) FROM EMP;

AVG(SAL)
------------------
```

2078.92857

B) COUNT({*/[DISTINCT/ALL]EXPR})
SQL> SELECT COUNT(*) FROM EMP;

COUNT(*)

14

C) SUM({[DISTINCT/ALL]EXPR})
SQL> SELECT SUM(SAL) FROM EMP;

SUM(SAL)

29105

D) MIN({[DISTINCT/ALL]EXPR})
SQL> SELECT MIN(SAL) FROM EMP;

MIN(SAL)

820

E) MAX({[DISTINCT/ALL]EXPR})
SQL> SELECT MAX(SAL) FROM EMP;

MAX(SAL)

5000

7. UPDATE OPERATIONS IN SQL

UPDATE COMMAND

1. Update emp set job = 'SALESMAN' ,hiredate = sysdate, sal = 1.1*sal+SAL where ename = 'ALLEN';

1 row updated.

SQL> SELECT * FROM EMP WHERE ENAME='ALLEN';

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7499	ALLEN	SALESMAN	7698	17-MAR-17	7056	300	30

2. SQL> Update emp set sal = sal +20 where job = 'CLERK';

4 rows updated.

SQL> SELECT * FROM EMP WHERE JOB='CLERK';

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	900		20
7876	ADAMS	CLERK	7788	23-MAY-87	1200		20
7900	JAMES	CLERK	7698	03-DEC-81	1050		30
7934	MILLER	CLERK	7782	23-JAN-82	1400		10

SUBSTITUTION METHOD

SQL>UPDATE EMP50 SET ENAME='&ENAME' WHERE EMPNO=&EMPNO;
Enter value for ename: VICKY
Enter value for empno: 7369
old 1: UPDATE EMP50 SET ENAME='&ENAME' WHERE EMPNO=&EMPNO
new 1: UPDATE EMP50 SET ENAME='VICKY' WHERE EMPNO=7369

1 row updated.

SQL> SELECT ENAME FROM EMP50 WHERE EMPNO=7369;

ENAME

VICKY

3. SQL> Update emp set ename = Lower(ename);

14 rows updated.

SQL> SELECT ENAME FROM EMP;
ENAME
----
adams
allen
blake
clark
ford
james
jones
king
martin
miller
scott
smith
turner
ward

8. CREATE SYNONYMS,VIEWS,INDEX AND GENERATE SET REPORT

SYNONYMS:

SQL>CREATE SYNONYM DEPT FOR BCA2B50.DEPT50;
Synonym created.

SQL> select * from dept50;

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

SQL>DROP SYNONYM DEPT;
Synonym dropped.

View:

SQL>CREATE VIEW V1 AS SELECT ENAME, EMPNO, SAL FROM EMP WHERE SAL>1000;
View created.

SQL>SELECT * FROM V1;

ENAME	EMPNO	SAL
SMITH	7369	820
ALLEN	7499	1600
WARD	7521	1250
JONES	7566	2975

```
MARTIN          7654          1250
BLAKE           7698          2850
CLARK           7782          2450
SCOTT           7788          3000
KING            7839          5000
TURNER          7844          1500
ADAMS           7876          1120
JAMES           7900           970
FORD            7902          3000
MILLER          7934          1320
```

```
SQL>DROP VIEW V1;
View dropped.
```

Index:

```
SQL> create index  deptind  on emp46(deptno);
Index created.
```

```
SQL>SELECT  EMPNO,ENAME  FROM  EMP46  WHERE  DEPTNO=10;
EMPNO       ENAME
-------------------------
 7782          CLARK
 7839          KING
 7934          MILLER
```

```
SQL>DROP DEPTIND;
Index dropped.
```

COMPUTE COMMAND

```
SQL> SET LINESIZE 32
SQL> SET PAGESIZE 27
SQL> TTITLE CENTER 'AMOUNT DUE PER ORDER'SKIP 2
SQL> BTITLE LEFT 'RUN BY:'SQL.USER FORMAT A5
SQL> COLUMN TOTAL FORMAT 999999.99
SQL> BREAK ON CUSTOMER# SKIP 1
SQL> SELECT CUSTOMER#,ORDER#,SUM(RETAIL*QUANTITY)TOTAL FROM
CUSTOMERS NATURAL JOIN ORDERS NATURAL JOIN ORDERITEMS NATURAL
JOIN BOOKS GROUP BY CUSTOMER#,ORDER# ORDER BY CUSTOMER#;
     AMOUNT DUE PER ORDER

 CUSTOMER#        ORDER/NUMBER       TOTAL
========== ============  ========== ============
  1001                1003                1147.50
  1003                1006                573.75
  1004                1008                1147.50
  1005                1000                573.75
  1009                                    1147.50
```

```
RUN BY:SCOTT
     AMOUNT DUE PER ORDER
  CUSTOMER#        ORDER/NUMBER       TOTAL
========== ============  ========== ============
  1007                 1007                 3442.50
  1010                 1001                 1147.50
  1011                                      573.75
  1011                 1002                1147.50
  1014                 1013                 573.75

SQL> COMPUTE SUM OF TOTAL ON CUSTOMER#
SQL> CLEAR BREAK
breaks cleared
SQL> CLEAR COLUMN
columns cleared
```

9. COMMIT, ROLLBACK AND SAVEPOINT

EXAMPLE OF COMMIT,ROLLBACK AND SAVEPOINT

```
CREATE TABLE CLASS(
2  ID NUMBER(2),
3  NAME VARCHAR(15));

Table created.
SQL> INSERT INTO CLASS VALUES(1,'ABHI');
1 row created.
SQL> INSERT INTO CLASS VALUES(2,'ADAM');
1 row created.
SQL> INSERT INTO CLASS VALUES(4,'ALEX');
1 row created.
SQL> INSERT INTO CLASS VALUES(5,'RAHUL');
1 row created.

SQL> SELECT * FROM CLASS;

        ID       NAME
----------------------------
         1       ABHI
         2       ADAM
         4       ALEX
         5       RAHUL

SQL> COMMIT;
Commit complete.
SQL> UPDATE CLASS SET NAME='ABHIJIT' WHERE ID=5;
1 row updated.
```

```
SQL> SAVEPOINT  A;
Savepoint created.
SQL> INSERT INTO CLASS VALUES(6,'CHRIS');
1 row created.
SQL> SAVEPOINT  B;
Savepoint created.
SQL> INSERT INTO CLASS VALUES(7,'BRAVO');
1 row created.
SQL> SAVEPOINT  C;
Savepoint created.
```

```
SQL> SELECT * FROM CLASS;

        ID     NAME
-----------------------------------
         1      ABHI
         2      ADAM
         4      ALEX
         5     ABHIJIT
         6     CHRIS
         7     BRAVO

6 rows selected.
SQL> ROLLBACK   TO   B;
Rollback complete.

SQL> SELECT * FROM CLASS;

        ID    NAME
---------------------------------
         1    ABHI
         2    ADAM
         4    ALEX
         5    ABHIJIT
         6    CHRIS

SQL> ROLLBACK   TO   A;
Rollback complete.

SQL> SELECT  *  FROM  CLASS;
```

```
        ID     NAME
------------------------------
         1     ABHI
         2     ADAM
         4     ALEX
         5     ABHIJIT
```

SQL> ROLLBACK;
Rollback complete.

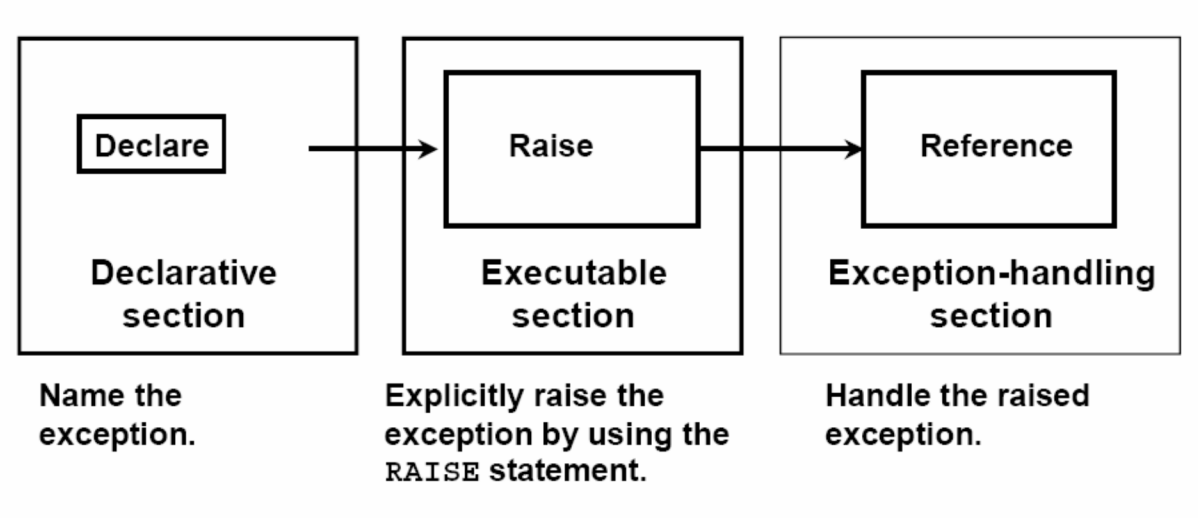
SQL> SELECT * FROM CLASS;

```
        ID      NAME
----------------------------
         1      ABHI
         2      ADAM
         4      ALEX
         5      RAHUL
```

17. Write a PL/SQL block to handle built-in exception like NO_DATA_FOUND, TOO_MANY_ROWS

```
DECLARE
  v_order#  NUMBER := &sv_orderno;
  v_ordered VARCHAR2(3) := 'NO';
BEGIN
       DBMS_OUTPUT.PUT_LINE('Check if the order has items');
       SELECT 'YES' INTO v_ordered FROM orderitems WHERE order# = v_order#;
        DBMS_OUTPUT.PUT_LINE ('The order has one item');
EXCEPTION
  WHEN NO_DATA_FOUND THEN
     DBMS_OUTPUT.PUT_LINE ('The order is not found');
  WHEN TOO_MANY_ROWS THEN
     DBMS_OUTPUT.PUT_LINE('The order has too many items');
END;
```

18. Write a PL/SQL block to handle user-defined exceptions.




```
DECLARE
-- Exception to indicate an error condition
  high_retail  EXCEPTION;
  v_error_code number;
  v_error_message varchar2(255);
  v_retail  books.retail%type;
BEGIN
SELECT retail into v_retail from books WHERE title = 'SHORTEST POEMS';
/* Ensure that there are no duplicates */
IF  v_retail > 20  THEN
    RAISE high_retail;
END IF;
EXCEPTION
    WHEN high_retail  THEN
    v_error_code := SQLCODE ;
    v_error_message := SQLERRM ;
    DBMS_OUTPUT.PUT_LINE ('An error has occurred'|| 'error code is '|| v_error_code || ' error
     message is '|| v_error_message);
INSERT INTO log_table (loginfo)
VALUES ('shortest poems retail greater than $20');
END;
```

19. Write a database trigger before insert/update/delete for each row not allowing any of these operations on the table instructor between 4 pm and 10 am.

```
Create or replace trigger idudeptmca before insert or delete or update  on deptmca for each
row
Begin
      If  not(to_char(sysdate,'hh PM')>='4 PM' or to_char(sysdate,'hh PM')<='10 AM') then
         dbms_output.put_line('trigger operation');
      else
           raise_application_error(-20111,'Insertion not allowed between 4 PM and 10PM');
            --dbms_output.put_line('insertion not allowed');
      end if;
end;
/
```

20. Write a package with functions isprime, ispositive, isnegative to check whether the given number is prime, positive, negative respectively.

Package Specification:

```
CREATE OR REPLACE PACKAGE prnegpos
IS
FUNCTION negorpos (p_no IN NUMBER) RETURN  varchar2;
FUNCTIoN primeornot(p_no in number) return varchar2;
END prnegpos;
```

Package Body

```
CREATE OR REPLACE PACKAGE BODY  prnegpos
 IS
 function negorpos(p_no In number) return varchar2
 is
 begin
  if p_no <0 then
   return 'Negative';
   else
    return 'Positive';
  end if;
 end negorpos;
 function primeornot(p_no In number) return varchar2
 is
 a number;
 begin
  for i in 2..p_no-1
  loop
  a:=p_no MOD i;
  --dbms_output.put_line(a);
  if a=0 then
   GOTO out;
   end if;
   end loop;
   <<out>>
   if a=1  then
  return(p_no||'is a prime');
  else
  return(p_no||'is not a prime');
  end if;
 end primeornot;
 end  prnegpos;
```

15. Write a PL/SQL block that uses date function to find the age of a person by accepting his DOB.

(Here use PL/SQL Cursor concepts and Table empmca created in SQL session used to calculate the age of employee using dob(Data of Birth) column to calculate the age of each employee and display the result on the screen)

```
SQL> desc empmca
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EMPNO                                     NOT NULL NUMBER(5)
 ENAME                                              VARCHAR2(10)
 JOB                                       NOT NULL VARCHAR2(9)
 MGR                                                NUMBER(4)
 DOB                                                DATE
 SAL                                                NUMBER(8,2)
 COMM                                               NUMBER(7,2)
 DEPTNO                                             NUMBER(2)
 CHILD                                              CHAR(1)
```

Program:-

```
declare
    empage int;
    empname emp.ename%type;
    cursor ageemp is
       SELECT ename, MONTHS_BETWEEN(sysdate, dob) / 12 age FROM empmca;
 begin
    open ageemp;
    loop
       fetch ageemp into empname, empage;
       exit when ageemp%notfound;
       dbms_output.put_line('Employee Age = ' || empage || ' || EmpName = '
||empname);
    end loop;
    close ageemp;
 end;
 /
```

```
Employee Age = 42 || EmpName = Smith
Employee Age = 42 || EmpName = Allen
Employee Age = 42 || EmpName = Ward
Employee Age = 42 || EmpName = Jones
Employee Age = 41 || EmpName = Martin
Employee Age = 42 || EmpName = Blake
Employee Age = 36 || EmpName = Scott
Employee Age = 41 || EmpName = Turner
Employee Age = 36 || EmpName = Adams
Employee Age = 41 || EmpName = James
Employee Age = 41 || EmpName = Ford
Employee Age = 1 || EmpName = Ramu
PL/SQL procedure successfully completed.
```

14. Write a PL/SQL block to handle string manipulations.

Strings in PL/SQL

Here learn about the strings in PL/SQL and it's some of the important functions which may help us to manipulate strings in PL/SQL in a better way.

Program for ASCII and CHR string functions.

```
declare
character char(1);
i integer;
j integer;
ch1 char;
ch2 char;

begin

i:=67;
ch1:=CHR(i);
ch2:=CHR(97);
dbms_output.put_line(ch1);
dbms_output.put_line(ch2);

character:='t';
i:=ASCII('A');
j:=ASCII(character);

dbms_output.put_line(i);
dbms_output.put_line(j);
end;
```

```
Output:
C
a
65
116

PL/SQL procedure successfully completed.
```

Program for Concate,Initcap and Length functions for string handling in SQL.

```
declare
 ch1 varchar2(10);
 ch2 varchar2(10);
 ch3 varchar2(20);
 i  integer;
 begin
 ch1:='include';
 ch2:='help';
 ch3:=concat(ch1,ch2);
 dbms_output.put_line('Concatenation of Two strings');
 dbms_output.put_line('your first string : '||ch1);
 dbms_output.put_line('your second string : '||ch2);
 dbms_output.put_line('your result :   '||ch3);
 ch1:= 'include';
 ch2:=INITCAP(ch1);
```

```
 dbms_output.put_line('Initcap of the first string and result');
 dbms_output.put_line('your first string : '||ch1);
 dbms_output.put_line('Initcap of first sting is: '||ch2);
 dbms_output.put_line(' Length of the string');
 i:=length(ch1);
 dbms_output.put_line('length of your first string :'||i);
 end;
```

SQL> /
Concatenation of Two strings
your first string : include
your second string : help
your result : includehelp
Initcap of the first string and result
your first string : include
Initcap of first sting is: Include
Length of the string
length of your first string :7

PL/SQL procedure successfully completed.

SQL>

Program for Upper, Lower and Substr functions for string handling in SQL.
(using empmca table with PL/SQL Cursor the demo)

```
declare
   empno1 int;
   ename1 emp.ename%type;
   enameupper emp.ename%type;
   enamelower emp.ename%type;
   empcode varchar2(20);
   cursor empdetails is SELECT empno,ename,concat(substr(empno,1,3),substr(ename,1,2)) empcode from empmca;
 begin
   open empdetails;

dbms_output.put_line('*************************************************************************************');
  dbms_output.put_line('     Empno          Ename            Ename Uppper                Ename Lower         Empcode');

dbms_output.put_line('*************************************************************************************');
   loop
      fetch empdetails into empno1,ename1,empcode;
      exit when empdetails%notfound;
      enameupper :=upper(ename1);
      enamelower:=lower(ename1);
      dbms_output.put_line(empno1||'          '||ename1||'              '||enameupper||'                        '||enamelower||'       '||empcode);
   end loop;
 dbms_output.put_line('*************************************************************************************');
   close empdetails;
 end;
/
```

SQL> /

Empno	Ename	Ename Uppper	Ename Lowerr	Empcode
7369	Smith	SMITH	smith	736Sm
7499	Allen	ALLEN	allen	749Al
7521	Ward	WARD	ward	752Wa
7566	Jones	JONES	jones	756Jo
7654	Martin	MARTIN	martin	765Ma
7698	Blake	BLAKE	blake	769Bl
7788	Scott	SCOTT	scott	778Sc
7844	Turner	TURNER	turner	784Tu
7876	Adams	ADAMS	adams	787Ad
7900	James	JAMES	james	790Ja
7902	Ford	FORD	ford	790Fo
7936	Ramu	RAMU	ramu	793Ra

Ex 8. Write a PL/SQL block for inserting rows into EMPDET table with the following calculations:
a. HRA=50% of BASIC
b. DA=20% of BASIC
c. PF=7% of BASIC
d. NETPAY=BASIC + DA+HRA-PF

Before execute the program to create table EMPDET(EMPNO, ENAME,BASIC,DA,HRD,PF and NETSALARY)

Name	Null?	Type
EMPNO	NOT NULL	NUMBER(4)
ENAME	NOT NULL	VARCHAR2(20)
BASIC		NUMBER(8,3)
DA		NUMBER
HRA		NUMBER
PF		NUMBER
NETSALARY		NUMBER

```
declare
        empno number;
        ename varchar2(15);
        basic number;
        da number;
        hra number;
        pf number;
        netsalary number;
  begin
         empno :=&empno;
         ename := '&ename';
         basic := &basic;
        da := basic*(20/100);
        hra:= basic*(50/100);
        pf := basic * (7/100);
        netsalary:=basic + da + hra - pf;
        dbms_output.put_line('Employee Name:'||ename);
        dbms_output.put_line('Providend Fund:'||pf);
        dbms_output.put_line('Net Salary:'||netsalary);
        insert into empdet values(empno,'ename',basic,da,hra,pf,netsalary);
 end;
```

Ex 9(a). Write a PL/SQL block to generate Fibonacci series for 10 numbers

Example : 0,1,1,2,3,5,8,13,21…….

```
declare
        a number;
        b number;
        c number;
        i number;
        n number;
begin
        n := &n;
        a := -1;
        b := 1;
        dbms_output.put_line('Fibonacci series for 10 numbers');
        for i in 1..n loop
        c := a+b;
        a := b;
        b := c;
        dbms_output.put_line( c || '  ');
        end loop;
end;
```

Ex 9(b) : Write a PL/SQL block to generate Armstrong number

A number is said to be an armstrong number if sum of its digits raised to the power n is equal to number itself, where n is total digits in number.

For example 407 is armstrong number as $4^3 + 0^3 + 7^3 = 64 + 0 + 343 = 407$.

```
declare
            n number;
            s number:=0;
            r number;
            len number;
            m number;
begin
         for a in 1..500 loop
              m:=a;
              n:=a;
              s:=0;                          -- Reset s for each loop
              len:=length(to_char(n));
                    while(n>0)  loop
                    r:=mod(n,10);
                    s:=s+power(r,len);
                    n:=trunc(n/10);
                    end loop;
                   If  m=s then
                  dbms_output.put_line(‘ The given number is Armstrong number’);
                  end if;
                  dbms_output.put_line(a || '=' || s); -- Output values for every loop.
             end loop;
end;
```

Ex 9(b) : Write a PL/SQL block to generate Armstrong number

A number is said to be an armstrong number if sum of its digits raised to the power n is equal to number itself, where n is total digits in number.

For example 407 is armstrong number as $4^3 + 0^3 + 7^3 = 64 + 0 + 343 = 407$.

```
declare
            n number;
            s number:=0;
            r number;
            len number;
            m number;
begin
         for a in 1..500 loop
              m:=a;
              n:=a;
              s:=0;                          -- Reset s for each loop
              len:=length(to_char(n));
                    while(n>0)  loop
                    r:=mod(n,10);
                    s:=s+power(r,len);
                    n:=trunc(n/10);
                    end loop;
                   If  m=s then
                   dbms_output.put_line(‘ The given number is Armstrong number’);
                   end if;
                   dbms_output.put_line(a || '=' || s); -- Output values for every loop.
             end loop;
end;
```

10. Write a program to update the database using cursors based on the following conditions
a. If basic >Rs.15000.00 then give increment of Rs.200.00
b. If basic is between Rs.1501.00 and Rs.3000.00 then give increment of Rs.500.00.
c. If basic > Rs.3000.00 then give increment of Rs.750.00.

Note:- Use the EX:8 table EMPDET to update the basic based on the condition.

Program:-

```
set serveroutput on
Declare
 Cursor ed is select empno,basic from empdet;
 ed1  ed%rowtype;
 begin
dbms_output.put_line(' The employee salary updation problem');
dbms_output.put_line('*********************************');
            open ed;
            loop
                  fetch ed into ed1;
                  exit when ed%notfound;
                        if (ed1.basic >15000 ) then
                              update empdet set basic = basic +200 where empno=ed1.empno;
                         elsif (ed1.basic >1501  and ed1.basic <=3000) then
                              update empdet set basic = basic+500 where empno = ed1.empno;
                        elsif( ed1.basic > 3000) then
                              update empdet set basic = basic+750 where empno=ed1.empno;
                        end if;
          end  loop;
          dbms_output.put_line(' The basic updated successfully');
end;
/
```

11. Write a function to get the factorial of given number.

Syntax of the PL/SQL function

CREATE [OR REPLACE] FUNCTION function_name (parameter_list)
RETURN return_type
IS

[declarative section]

BEGIN
[executable section]
[EXCEPTION]
[exception-handling section]
END;

Program:

```
Create or replace function fact( n number)
Return number
is
f  integer :=1;
i integer;
   begin
       for i in 1.. n
          loop
               f := f * i;
          end loop;
return f;
end;
/
```

6. Write queries using nested queries.

AIM: To execute nested queries using SQL.

SUB QUERY:

Sometimes, getting an answer to a query requires a multi-step operation. First a query be created to determine a value that is unknown to the user but is contained within the database. That first query is sub-query. The results of the subquery are passed as input to the parent query or outer query. The parent query incorporates that value into its calculation to determine the final output. The sub-query appears within the WHERE or HAVING clause of SQL statement, there may be times when it is appropriate to use a subquery in the SELECT or Using subquery in a FROM clause has a specific purpose.

Rules for Subquery of Any Type:

- A Subquery must be a complete query in itself (i.e It must have atleast a SELECT and a FROM clause)
- A Subquery cannot have an ORDER BY clause. If the displaye output needs to be presented in a specific order , an ORDER BY clause should be listed as the last clause of the outer query.
- A subquery must be enclosed within a set of parentheses to separate it from the outer query.
- If the subquery is placed in the WHERE or HAVING clause of an outer query , the subquery can be placed only on the right side of the comparison operator.

1) a) Single Row Sub-Query:

A single sub-query is used when the results of the outer query are based on a single unknown value. A single-row subquery can return to the outer query only one row results that consists of only one column. Therefore a single row subquery as a single value.

Example:-

Display the names of the employee who working with JONES with same job

1) Select ename, job from emp1 where job = (SELECT job from emp1 where ename = ‘jones’)

Note - Operators indicate to oracle whether a user is creating a single-row subquery or multi-row subquery. The single-row operators are =,>,<,>=,<= and <>. Although other operators such as IN are allowed single –row operators to instruct oracle to take one value is expected from the subquery.

2) **To retrieve the employee who get the maximum salary from list of employees.**

Select ename from empmca where sal = (Select max (sal) from empmca);

b) Single Row Subquery in a HAVING Clause:

A subquery can also be included in a HAVING clause. A HAVING clause is used when the group results of a query need to be restricted based on some condition. If the result returned from a subquery must be compared to a group function then the inner query must be nested in the outer query's HAVING clause.

3) To list of all department that return a higher average salary than the Dept no 20

The multiple steps given below

1. Determine the average salary of all employees of dept no 20.
2. Calculate the average salary of each department.
3. Compare average salary of each department with average salary of deptno 20.

Select deptno, avg(sal) "Average salary" from empmca Group By deptno Having AVG(sal) > (Select AVG(sal) from empmca where deptno=20);

c) Single-Row Subquery in a SELECT Clause

A single-row subquery can also be nested in the SELECT clause of an outer query. However this approach is rarely used because when the subquery is listed in the SELECT clause, this means the value returned by the subquery will be displayed for every row of output generated by the parent query.

To compare the salary of each employee against the average salary of all employees in the empmca table.

4) Select ename,sal,(select avg(sal) from empmca) "Overall Average" from empmca;

d) Single-Row Subquery in a FROM Clause

Subqueries can also be used in the FROM clause, where they are sometimes referred to as *inline views*.

Display top three highest salary received by employees in the empmca table.

5) Select ename,sal from (select ename, sal from empmca order by sal desc) where rownum <4;

2) Multiple Row Sub-Query:

Multiple Row Sub-Queries are nested queries that can return more than one row of results to the parent query. Multiple-row subqueries are commonly used in WHERE and HAVIGNG clause.

IN: The commonly used operator.

Display maximum salary from each department using IN operator as a multiple row subquery.

6) Select ename,empno,deptno,sal from empmca where sal in(select max(sal) from empmca group by deptno) order by deptno;

2) ANY and ALL Operator:

The ALL and ANY operators can be combined with other comparison operators to treat the results of a subquery as a set of values, rather than as individual values

Operator	Description
>ALL	More than the highest value returned by the subquery
<ALL	Less than the lowest value returned by the subquery
<ANY	Less than the highest value returned by the subquery
>ANY	More than the lowest value returned by the subquery
=ANY	Equal to any value returned by the subquery (same as in)

Display the employees names who is receiving salary greater than the all salary in the dept no 10.

(This example used to demonstrate the above operator changing the operator)

7) Select empno,ename,sal from empmca where sal > ALL(Select sal from empmca where deptno =10)

3) **Correlated Subqueries:**

Uncorrelated Subqueries : - Previous queries with example are uncorrelated because the subquery is executed first , the results of the subquery are passed to the outer query and then outer query is executed.

Correlated Subqueries:- A correlated subquery is subquery that is processed or executed Once for each row in the outer query.

Difference between Uncorrelated and a Correlated subquery:-

If a subquery references a column from the outer query then it is a correlated subquery otherwise uncorrelated.

Example :

List all employees whose salary is less than the average salary of their department

8) select p.ename,p.deptno,P.SAL from emp p where p.sal <(select avg(s.sal) from emp s where s.deptno =p.DEPTNO);

4) Nested Subqueries:-

Subqueries can be nested inside the FROM,WHERE Or HAVING clauses of other subqueries. Subqueries in WHERE clause can be nested to a depth of 255 subqueries, and there is no depth limit when the subqueries are nested in a FROM clause.

When nesting use following strategy

- Determine exactly what you are trying to find.
- Write innermost subquery first.

From innermost query look at the value able to pass back to the outer query and check for its suitability to outer query .Sometimes need to create several layers of subqueries to link the value returned by the innermost subquery to the value needed by the outer query.

Example:-

1). **Display the second largest salary from the employees using emp table.**

Select ename,sal from emp where sal in (select max(sal) from emp where sal <(select max(sal) from emp));

RESULT:

Thus the all types of subquery executed successfully and results verified.

13. Write a PL/SQL block to check whether the given String is palindrome or not.

```
DECLARE
   -- Declared variables are s, l, t .
   -- These variables are of same data type VARCHAR.
   s VARCHAR2(10) :='abccba';
   l VARCHAR2(20);
   t VARCHAR2(16);
BEGIN
   FOR i IN REVERSE 1..Length(s) LOOP
      l := Substr(s, i, 1);
      -- here || are used for concatenation of string.
      t := t ||''||l;
   END LOOP;
   IF t = s THEN
    dbms_output.Put_line(t ||''||' is palindrome');
   ELSE
    dbms_output.Put_line(t||''||' is not palindrome');
   END IF;
END;
```

7. Develop queries using functions and views

CHARACTER FUNCTIONS : -

a) ASCII- It returns the ascii value of the specified character.

Ex: SELECT ascii (‘A’) from dual;

b) CHR(C) – returns binary equivalent to c in the database character set.

Ex: select chr(75) from dual;

Return value ‘K’;

c)CONCAT(C1,C2)-returns c1 concatenated with c2.

Ex: select concat(‘ram’, ‘chandran’) from dual;

select concat(‘empname’,’john smith’) from dual;

select ename,concat('Employee number:',empno) "number" from empmca;

d)UPPER – converts the given string in to uppercase.

Ex: select upper(‘sastra’) from dual;

e) LOWER – converts the given string into lowercase.

Ex: SELECT lower (‘sastra’) from dual;

f) INITCAP – converts the first character of each word of the string into capital.

Ex: SELECT initcap (‘India is my country’) from dual;

g)LENGTH – Returns the length of the string.

Ex: SELECT length(‘SASTRA’) from dual;

h) LPAD – Used to pad or fill in the area left of a character string with a specific character or blank space.

Ex: select lpad(firstname,12,'*') from customers;

i) RPAD - Used to pad or fill in the area right of a character string with a specific character or blank space.

Ex: - select rpad(firstname,12,'*') from customers;

j) LTRIM- To remove a specific string of characters from the left side of a set of data.

Ex: select firstname,lastname,ltrim(address,'P.O. BOX') from customers;

k) REPLACE – Search a particular occurrence of string of characters and if found, substitutes it with another set of characters.

Ex:- select replace(address,'P.O','Post Office') from customers;

l) SUBSTR - It is used to return a substring or portion of a string.

Ex:- select distinct substr(zip,1,3) from customers;

Select distinct zip,substr(zip,1,3),substr(zip,-3,2) from customers;

NUMERIC FUNCTIONS :

Accepts numeric input and return numerical values

a) ABS(n) – Returns the absolute value of n

Ex: SELECT abs(-15) from dual;

Return value - 15

b) CEIL(n) – Returns smallest integer greater than or equal to n

Ex: SELECT ceil(15.7) from dual;

Return value -16

c) FLOOR – Returns largest integer equal to or less than n

Ex: SELECT floor(15.7) from dual;

Return value -15

d) POWER – Returns m raised to the n th power, n must be an integer

Ex : SELECT power(3,2) from dual;

Return value – 9

e) SQRT – Returns the square root of n.

Ex: SELECT sqrt(26) from dual;

Return value – 5.099

f) ROUND- Used to round numeric fields to the started precision. The syntax of the Round f function is ROUND(n,p).

Ex: SELECT title,retail, round(retail,1) from books;

g) TRUNCATE – Used to truncate numeric data.

Ex: SELECT title,retail, trunc(retail,1) from books;

DATE FUNCTIONS

Oracle's DATE function displays date values in a DD-MON-YY format that represents a two-digit day, a three-letter month abbreviation and two-digit year(e.g 02-FEB-04).Users reference a date as a non-numeric field but actually stored internally in a numeric format.

a) MONTHS_BETWEEN – It determines the number of months between two dates

Ex: - select title,months_between(orderdate,pubdate) from books natural join orders natural join orderitems where order#=1009

b) ADD_MONTHS –used add months to given date and determine the date information. Sytax ADD_MONTHS(d,m), Where d represents the beginning date for the calculation and m represents the number of months to add to the date.

Ex:- Select title,pubdate,add_months(pubdate,60) “Drop Date” from books;

c) NEXT _DAY - It determines the next occurrence of a specific day of the week after a given date. The syntax NEXT_DAY(d,DAY), Where d represents the starting date and DAY represents the day of the week to be identified.

Ex:- Select next_day(orderdate,’MONDAY’) from orders;

d) TO_DATE- The function allows users to enter a date in any format and then it converts the entry into oracle default format.The syntax TO_DATE(d,f) , Where d represents the date being entered by the user and f is the format for the date that was entered.

Ex:- select order#,orderdate,shipdate from orders where orderdate=to_date('MARCH 31,2003','MONTH DD,YYYY');

e) TO_CHAR - It is used to convert dates and numbers to a formatted character string. It is used to display date in a particular format. The syntax of the TO_CHAR(n.’f’), where n is the date or number to be formatted and f is the format model to be used.

Ex:- select title,to_char(pubdate,'MONTH DD YYYY') "PUBLICATION DATE", TO_CHAR(RETAIL,'$999.99') "RETAIL Price" from books

f) DECODE – It takes a specified value and compares it to values in a list. If a match is found the specified result is returned . If no match is found then a default result is returned. If no default result is defined, a NULL is returned as the result.The syntax for the DECODE function is DECODE(v,L1,R1,L2,R2,…….,D) , Where V is the value are searching for , L1 the first value in the list , R1 represents the result to be returned if L1 and V are equivalentect and D is the default result to return if no match is found.

Ex:- Select customer#,state,decode(state,’CA’,.08,’FL’,.07,0) “SALES TAX RATE” from customers;

g) NVL Function – The NVL function is used to substitute a value for the existing NULL. The Syntax for the NVL function is NVL(x,y) , Where y represents the value to be substituted if x is NULL . In many cases, the substitute for a NULL value in a calculation is zero(0).

Ex:- select ename,empno,sal+nvl(comm,0) from emp;

Group functions:

Group function returns results based on groups of rows,rather than single row.

Distinct – Returns only distinct values of the argument expression.

All - Retunrs all values including all duplicates.

a) Avg([distinct/all]n) – Returns average value of n.

Ex : SELECT avg(sal) from empmca;

Return value - 2074.21

b) Count({*/[distinct/all]expr}) - Returns the number of rows in the query.

Ex: Select count(*) from empmca;

Return value – 14

c) sum({[distinct/all]expr}) – Returns sum value of expr.

Ex: Select sum(sal) from empmca;

Return value – 29025

d) min({[distinct/all]expr})}) – Returns minimum value of expr

Ex: Select min(sal) from empmca;

e) max({[distinct/all]expr})}) – Returns maximum value of expr

Ex: Select max(sal) from empmca;

VIEW

1) A VIEW is a virtual table based on the result-set of a SELECT statement.

2) A view contains rows and columns, just like a real table.

Syntax

CREATE VIEW view_name AS SELECT column_name(s)FROM table_name

WHERE condition

Example:

Create view v1 as select ename, address from empmca where sal>1000;

Querying the view

Select * from v1;

Drop view

Drop view v1;

RESULT: Library function character, numeric ,date group functions and view created and query executed on the view successfully.

5. Write queries using set operations.

Aim : Use Set operators to combine the results of two (or more) SELECT operations.

1.UNION

UNION merges the output of two or more queries. It suppresses duplicates.

Example:

Select deptno from empmca Union select deptno from deptmca;

2.UNION ALL

It is same, as UNION but it does not filter the duplicate records.

Example:

Seclect deptno from empmca **Union all** select deptno from deptmca;

3.INTERSECT

It returns the records that are common to both the queries.

Example:

Select deptno from empmca **Intersect** Select deptno from deptmca;

4. MINUS

It removes the results of the second query that are also found in the first query and only displays the rows that were uniquely returned by only the first query.

Example:

SELECT deptno from deptmca **Minus** select deptno from empmca;

Result: All the set operations executed successfully.

4(b) Writing Queries using Join Operations

Aim : To execute queries using Join Operations

JOIN

To join information from any number of tables. Sometimes information need to retrieve are stored in more than two table in such situations traditional join using WHERE clause to identify the access path to relate various tables . From Oracle9i onwards Oracle Corporation introduced JOIN methods.

The general form of the traditional Oracle-proprietary syntax relevant to joins is as follows:
SELECT table1.column, table2.column FROM table1, table2
[WHERE (table1.column_name = table2.column_name)] |
[WHERE (table1.column_name(+)= table2.column_name)] |
[WHERE (table1.column_name)= table2.column_name) (+)] ;

6.EQUI JOIN / SIMPLE JOIN (=)

The most common type of join that will use in the workplace is based upon two(or more) tables having equivalent data stored in a common column. Such joins are called equality joins. They may also be referred to as equi joins, inner joins or simple joins.

A common column is a column with equivalent data that exists in two or more tables.
Example : The EMPMCA and DEPTMCA tables both have a common column called deptno. Equi Join combines records when common values are found in a record from each of the join tables.

Example:

1) Select ename,loc from empmca,deptmca where ename='ALLEN' and empmca.deptno = deptmca.deptno;

2) Select ename,empno,dname,loc from empmca,deptmca where empmca.deptno = deptmca.deptno;

Column Qualifier:

Any time oracle references multiple tables having the same column name, the column name must be prefixed with the table name. You will receive an error message if query is ambiguous and does not specify exactly which column is the common column

Example :

2) Select ename,empno,dname,deptmca.deptno,loc from empmca,deptmca where empmca.deptno = deptmca.deptno;

Table Alias: Table Alias provides nick(short) name to table name and easy way to handle table names more than once in select statement. If a table alias is assigned in the FROM clause, it must be used any time the table is referenced in that SQL statement.

Select ename,empno,dname,d.deptno,loc from empmca e,deptmca d where e.deptno = d.deptno;

SELF JOIN and Non-Equi Join :

Joints a table itself. Sometimes data in a table referencing other data stored within the same table.

In join condition other than = such as { >,<,<= >= etc} to be used then it is called non- equijoin.

Example:

Display the employees who earn more than JONES of the same company.

3) Select x.ename,x.sal,x.job,y.ename,y.sal,y.job from empmca x, empmca y where x.sal > y.sal and y.ename = 'JONES';

OUTER JOIN:

Includes records of a table in output when there is no matching record in the other table. Outer join operator: '+'. It is used to indicate the table containing the deficient rows. This operator is placed next to the table that should have null rows added to create a match.

Example:

4) Select d.*,e.job,e.ename from dept d, emp e where d.deptno=e.deptno(+);

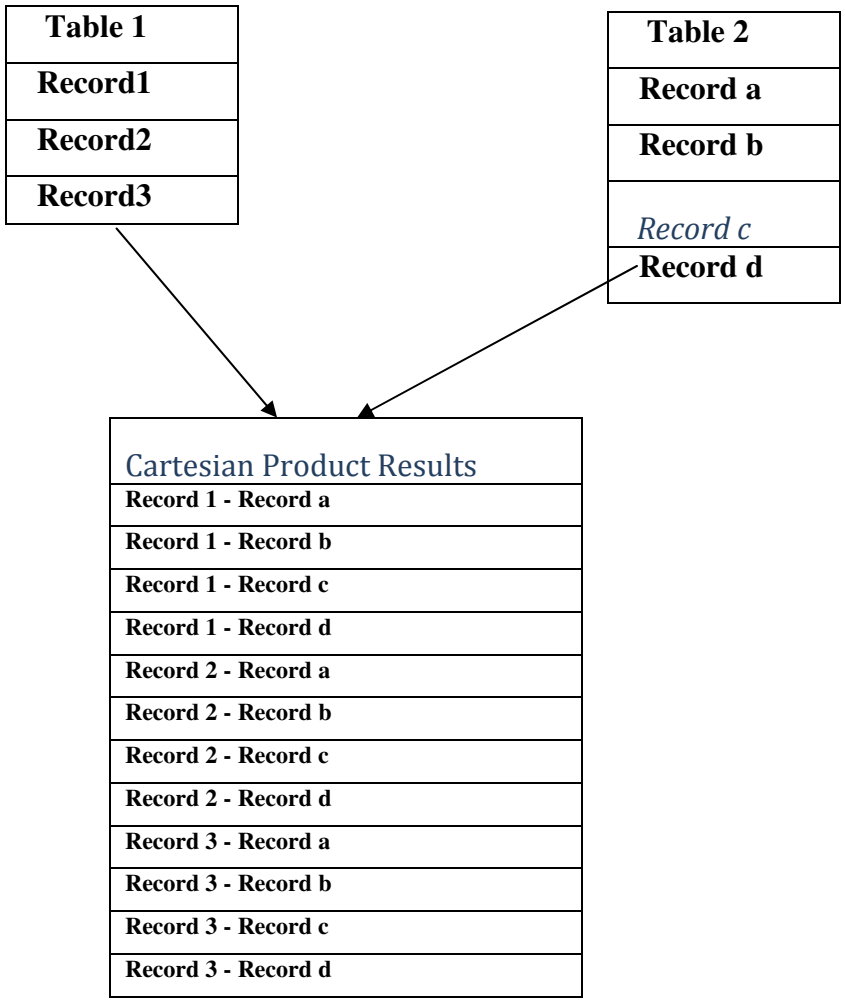
Explanation for the two examples:-

The department number (Deptno) in the DEPT table available but no employees in the department then EMP is lack of rows so the (+) placed after the table in the WHERE condition.

CARTESIAN JOIN:

It is also called Cartesian product or cross join, each record in the first table matched with each record in the second table. This type of join is useful when performing certain statistical procedures for data analysis. If the first table contain m rows and second table n rows then m * n rows on the result.

CARTESIAN PRODUCT RESULTS:-



Example:

Cartesian Join – Traditional Method

5) Select empno,ename,sal,dname,dloc from empmca,deptmca;

RESULT:

Ex 4. (a) SORTING AND GROUPING

1. ORDER BY:

The format of the ORDER BY clause in the context of the SQL SELECT statement is as follows:
SELECT *|{[DISTINCT] *column|expression* [*alias*],…}
FROM *table*
[WHERE *condition(s)*]
[ORDER BY {*col(s)|expr|numeric_pos*} [ASC|DESC] [NULLS FIRST|LAST]];

Ascending and Descending Sorting

The order by class is used for displaying the results of a query in a sorted order. The order by class is listed at the end of the select statement. Ascending sort order is natural for most types of data and is therefore the default sort order used whenever the ORDER BY clause is specified. An ascending sort order for numbers is lowest to highest, while it is earliest to latest for dates and alphabetically for characters. The first form of the ORDER BY clause shows that results of a query may be sorted by one or more columns or expressions.

ORDER BY *col(s)|expr;*

Example :

1) Select ename,sal from emp order by ename asc;
2) Select ename,comm.,sal from emp1 order by ename ;
3) Select comm from emp1 where comm is not null order by comm desc;
4) Select ename,sal,comm from emp order by comm desc nulls last
5) Select distinct ename from emp1 order by sal asc;

Several implicit default options are selected when you use the ORDER BY clause. The most important of these is that unless DESC is specified, the sort order is assumed to be ascending. If null values occur in the sort column, the default sort order is assumed to be NULLS LAST for ascending sorts and NULLS FIRST for descending sorts

Positional Sorting:

Oracle offers an alternate and shorter way to specify the sort column or expression. Instead of specifying the column name, the position of the column as it occurs in the SELECT list is appended to the ORDER BY clause.

Consider the following example:

1) select ename,sal,hiredate from emp order by 3;
2) select ename,sal,hiredate from emp order by 2 desc;

Composite Sorting:-

Results of a query may be sorted by more than one column using *composite sorting.*Two or more columns may be specified (either literally or positionally) as the composite sort key by commas separating them in the ORDER BY clause.

1) select job,ename,sal,hiredate from emp order by job desc,ename,3 desc;

Consider the requirement to fetch the JOB, ENAME, SAL, and HIRE_DATE values from the EMPLOYEES table. The further requirements are that the results must be sorted in reverse alphabetical order by JOB first, then in ascending alphabetical order by ENAME, and finally in numerically descending order based on the SALARY column.

2.GROUP BY

Using group by clause summarize the results at a subtotal level. The group by clause of the select statement provide this capability.

Ex:

SELECT deptno, Avg(sal) from emp1 group by deptno;
SELECT deptno from emp where job = 'CLERK' group by deptno;

3.HAVING CLAUSE:

It is used to restrict the groups return by a query.

EX:

Select deptno from emp1 where job='ANALYST' group by deptno
m having count(*)>=2;

Result :

All the condition operators join operators with join method and orderby and group by executed successfully.

Sample Queries:

1. ORDER BY(ASCENDING AND DESCENDING ORDER)

1. SQL>select ename,sal from emp order by enameasc;

ENAME	SAL
ADAMS	1120
ALLEN	1600
BLAKE	2850
CLARK	2450
FORD	3000
JAMES	970
JONES	2975
KING	5000
MARTIN	1250
MILLER	1320

2. SQL> SELECT ENAME,COMM,SAL FROM EMP ORDER BY ENAME;

ENAME	COMM	SAL
ADAMS		1120
ALLEN	300	1600
BLAKE		2850
JONES		2975
KING		5000
MARTIN	1400	1250
MILLER		1320
SMITH		820
TURNER	0	1500
WARD	500	1250

3. SQL> SELECT COMM FROM EMP WHERE COMM IS NOT NULL ORDER BY COMM DESC;

COMM
1400

500
300
0

4. SQL>SELECT ENAME,SAL,COMM FROM EMP ORDER BY COMM DESC NULLS LAST;

ENAME	SAL	COMM
MARTIN	1250	1400
WARD	1250	500
ALLEN	1600	300
TURNER	1500	0
SCOTT	3000	
KING	5000	
ADAMS	1120	
JAMES	970	

5. SQL> SELECT DISTINCT SAL FROM EMP ORDER BY SAL ASC;

SAL
820
970
1120
1250
1320
1500
1600
2450
2850
2975
3000
5000

POSITIONAL SORTING:

1. SQL> SELECT ENAME,SAL,HIREDATE FROM EMP ORDER BY 3;

ENAME	SAL	HIREDATE
SMITH	820	17-DEC-80
ALLEN	1600	20-FEB-81
WARD	1250	22-FEB-81
BLAKE	2850	01-MAY-81
CLARK	2450	09-JUN-81
TURNER	1500	08-SEP-81
MARTIN	1250	28-SEP-81
KING	5000	17-NOV-81

2. SQL> SELECT ENAME,SAL,HIREDATE FROM EMP ORDER BY 2 DESC;

ENAME	SAL	HIREDATE
KING	5000	17-NOV-81
FORD	3000	03-DEC-81
SCOTT	3000	19-APR-87
JONES	2975	02-APR-81
BLAKE	2850	01-MAY-81
CLARK	2450	09-JUN-81
ALLEN	1600	20-FEB-81
TURNER	1500	08-SEP-81
MILLER	1320	23-JAN-82

COMPOSITE SORTING

SQL>select job,ename,sal,hiredate from emp order by job desc,ename,3 desc;

JOB	ENAME	SAL	HIREDATE
SALESMAN	ALLEN	1600	20-FEB-81
SALESMAN	MARTIN	1250	28-SEP-81
PRESIDENT	KING	5000	17-NOV-81
MANAGER	BLAKE	2850	01-MAY-81
CLERK	MILLER	1320	23-JAN-82

```
CLERK          SMITH           820          17-DEC-80
ANALYST        FORD           3000          03-DEC-81
ANALYST        SCOTT           3000         19-APR-87
```

2. GROUP BY

1. SQL> SELECT DEPTNO,AVG(SAL) FROM EMP GROUP BY DEPTNO;

```
   DEPTNO    AVG(SAL)
------------------------------------
       30          1570
       20          2183
       10          2923.33333
```

2. SQL> SELECT DEPTNO FROM EMP WHERE JOB='CLERK' GROUP BY DEPTNO;

```
   DEPTNO
-------------------
       30
       20
       10
```

3. HAVING CLAUSE

1. SQL> SELECT DEPTNO FROM EMP WHERE JOB='ANALYST' GROUP BY DEPTNO HAVING COUNT(*)>=2;

```
   DEPTNO
--------------------------
       20
```

INTRODUCTION TO SQL:

SQL stands for structured query language, which is used to communicate with the relational database, which are in turn a set of related information stored in the form of tables. SQL is a non-procedural language because it process sets of records rather than just one data at a time and also provides automatic navigation to the data.

The SQL language is subdivided according to their functions as follows:

Data Definition Language (DDL)
Or
Schema Definition Language.
Data Manipulation Language (DML)
Data Control Language (DCL)

Tables

In relational database systems (DBS) data are represented using tables (relations). A query issued against the DBS also results in a table.

A table has the following structure:

	Column 1	Column2	…… …..	Column n
Tuple (or Record->)				
Tuple (or Record->)				
Tuple (or Record->)	…..	…..	…..	……..
Tuple (or Record->)	……	……..	…….	………..

A table is uniquely identified by its name and consists of rows that contain the stored information, each row containing exactly one tuple (or record). A table can have one or more columns.A column is made up of a column name and a data type, and it describes an attribute of the tuples. The structure of a table, also called relation schema, thus is defined by its attributes.The type of information to be stored in a table is defined by the data types of the attributes at table creation time.

SQL uses the terms table, row, and column for relation, tuple, and attribute, respectively.

A table can have up to 254 columns which may have different or same data types and sets of values (domains), respectively. Possible domains are alphanumeric data (strings), numbers and date formats. Oracle offers the following basic data types:

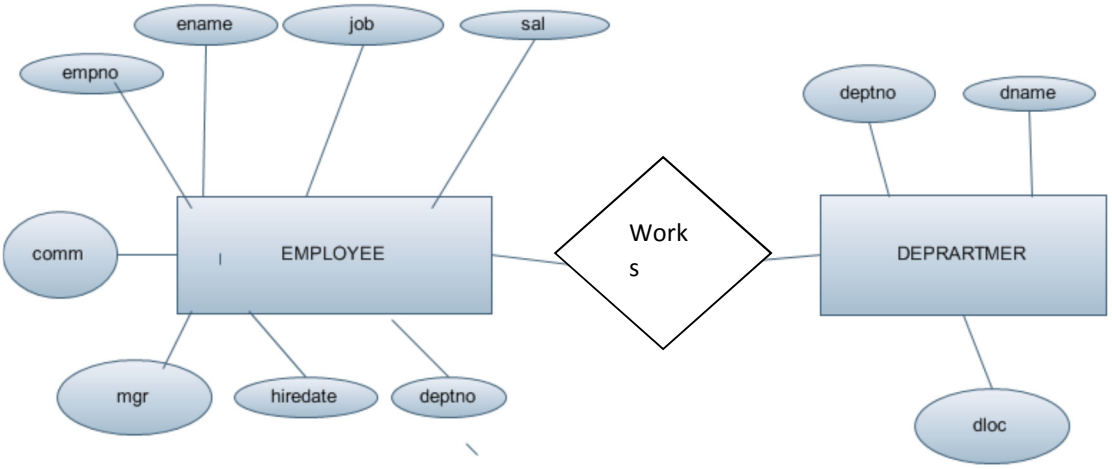
DATA TYPES

S.NO	Data Types	Description
1	Char(n)	Character String . n is the size of variable. Maximum size is 255 characters. The default size is 1
2.	Varchar2(n)	Character string . n is the size of the variable
3.	Number	Defines Numeric data type with space for 40 digit and space for sign and decimal point
4.	Number(n)	Numeric variable.n is the size of the variable

5.	Number(n,d)	Numeric variable.n is the size of variable and d is the size if the decimal point.
6.	Raw(size)	Raw Binary data of length size bytes .Maximum size is 32767 bytes.
7.	Integer	It is same as number data type but values will be whole numbers. Columns defined with this format not accept decimal values.
8.	Integer(n)	Specifies an integer data type of length n.
9.	Long	Defines a character data type upto 32760bytes. One one long column may be defined for table. This type of column may not be used in sub queries, Where clauses or indexes.
10.	Long Raw	Same as LONG except it contains binary data or byte strings and not interpreted by PL/SQL
11.	LOB Type	LOB variables can used interchangeably with LONG and LONG RAW variables. It consists of BFILE,BLOB,CLOB and NLOB
12.	BFILE	It is used to store large binary objects in operating system files outside the database
13.	BLOB	The BLOB datat ype to store large binary objects in the database, in-line or out-of-line. Every BLOB variable stores a locator, which points to a large binary object. The size of a BLOB cannot exceed four gigabytes.
14.	CLOB	The CLOB datatype to store large blocks of character data in the database, in-line or out-of-line. Both fixed-width and variable-width character sets are supported. Every CLOB variable stores a locator, which points to a large block of character data. The size of a CLOB cannot exceed four gigabytes
15.	NCLOB	The NCLOB datatype to store large blocks of NCHAR data in the database, in-line or out-of-line. Both fixed-width and variable-width character sets are supported. Every NCLOB variable stores a locator, which points to a large block of NCHAR data. The size of an NCLOB cannot exceed four gigabytes.
16.	DATE	The DATE datatype to store fixed-length datetimes, which include the time of day in seconds since midnight. The date portion defaults to the first day of the current month; the time portion defaults to midnight. The date function SYSDATE returns the current date and time.

Sample Databases used for illustration of SQL Commands is given below with ER Diagram and corresponding Relational Model with suitable data entered in the tables.

Example 1: EMPLOYEE and DEPARTMENT Tables.



- - Relation between Employee and Department is Works is many –to-one .

DATABASE for EMPLOYEE and DEPARTMENT Entities

EMP Table given with sample Data

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

DEPT Table given with Sample Data

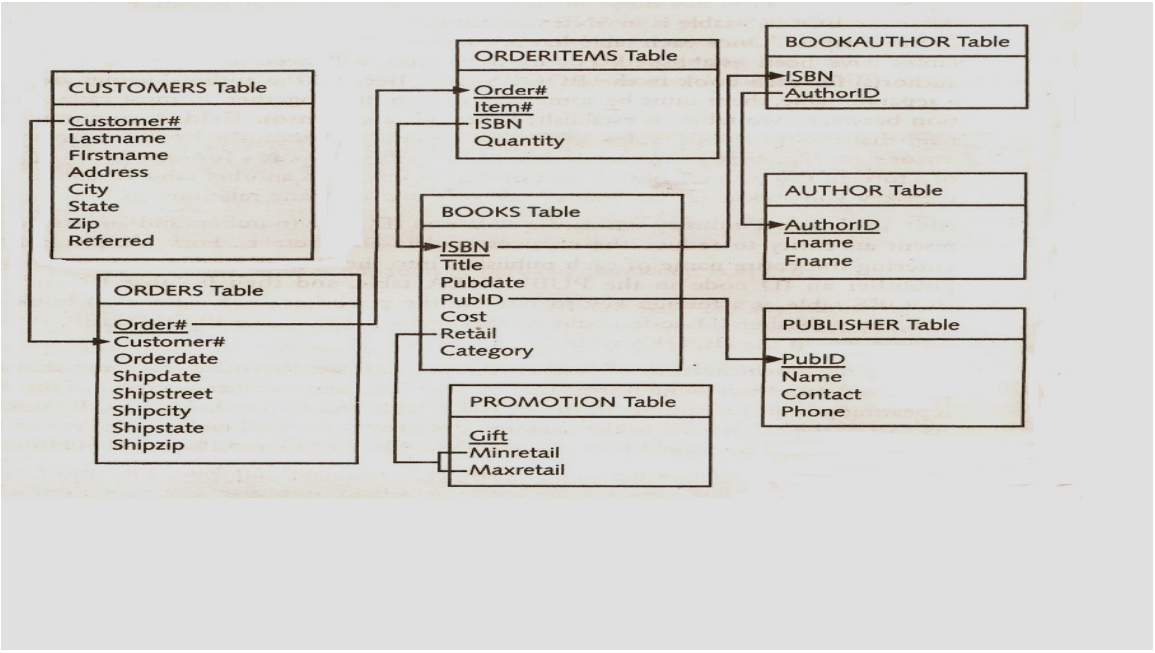
DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Example 2: JUSTLEE BOOK'S DATABASE DESCRIPTION

The initial organization of the database structure for JustLee Books – the Case study for this RDBMS lab. The database is used first to record customer's orders. Customers and JustLee's employee can identify a book by its ISBN, title or author's name(s). Employees can also determine when a particular order was placed and when or if the order was shipped. The database also stores the publisher contact information so the bookseller can reorder a book.

Basic Assumptions

Three assumptions made when designing the database are as follows:

1. An order is not shipped until all items are available (i.e there are no back orders or partial order shipments).
2. All address are within the United States ; Otherwise , the Address /Zip Code fields would need to be altered.
3. Only orders for the current month or orders from previous months that have not yet shipped are stored in the ORDERS table. The following figure illustrates the structure of JustLee database.
4. 

Customers Table

CUSTOMER#	LASTNAME	FIRSTNAME	ADDRESS	CITY	STATE	ZIP	REFERRED
1001	MORALES	BONITA	P.O. BOX 651	EASTPOINT	FL	32328	
1002	THOMPSON	RYAN	P.O. BOX 9835	SANTA MONICA	CA	90404	
1003	SMITH	LEILA	P.O. BOX 66	TALLAHASSEE	FL	32306	
1004	PIERSON	THOMAS	69821 SOUTH AVENUE	BOISE	ID	83707	
1005	GIRARD	CINDY	P.O. BOX 851	SEATTLE	WA	98115	
1006	CRUZ	MESHIA	82 DIRT ROAD	ALBANY	NY	12211	
1007	GIANA	TAMMY	9153 MAIN STREET	AUSTIN	TX	78710	1003
1008	JONES	KENNETH	P.O. BOX 137	CHEYENNE	WY	82003	
1009	PEREZ	JORGE	P.O. BOX 8564	BURBANK	CA	91510	1003
1010	LUCAS	JAKE	114 EAST SAVANNAH	ATLANTA	GA	30314	
1011	MCGOVERN	REESE	P.O. BOX 18	CHICAGO	IL	60606	
1012	MCKENZIE	WILLIAM	P.O. BOX 971	BOSTON	MA	02110	
1013	NGUYEN	NICHOLAS	357 WHITE EAGLE AVE.	CLERMONT	FL	34711	1006
1014	LEE	JASMINE	P.O. BOX 2947	CODY	WY	82414	
1015	SCHELL	STEVE	P.O. BOX 677	MIAMI	FL	33111	
1016	DAUM	MICHELL	9851231 LONG ROAD	BURBANK	CA	91508	1010
1017	NELSON	BECCA	P.O. BOX 563	KALMAZOO	MI	49006	
1018	MONTIASA	GREG	1008 GRAND AVENUE	MACON	GA	31206	
1019	SMITH	JENNIFER	P.O. BOX 1151	MORRISTOWN	NJ	07962	1003
1020	FALAH	KENNETH	P.O. BOX 335	TRENTON	NJ	08607	

AUTHOR table

AUTHORID	LNAME	FNAME
S100	SMITH	SAM
J100	JONES	JANICE
A100	AUSTIN	JAMES
M100	MARTINEZ	SHEILA
K100	KZOCHSKY	TAMARA
P100	PORTER	LISA
A105	ADAMS	JUAN
B100	BAKER	JACK
P105	PETERSON	TINA
W100	WHITE	WILLIAM
W105	WHITE	LISA

R100	ROBINSON	ROBERT
F100	FIELDS	OSCAR
W110	WILKINSON	ANTHONY

BOOKS table

ISBN	TITLE	PUBDATE	PUBID	COST	RETAIL	CATEGORY
1059831198	BODYBUILD IN 10 MINUTES A DAY	21-JAN-01	4	18.75	30.95	FITNESS
0401140733	REVENGE OF MICKEY	14-DEC-01	1	14.2	22	FAMILY LIFE
4981341710	BUILDING A CAR WITH TOOTHPICKS	18-MAR-02	2	37.8	59.95	CHILDREN
8843172113	DATABASE IMPLEMENTATION	04-JUN-99	3	31.4	55.95	COMPUTER
3437212490	COOKING WITH MUSHROOMS	28-FEB-00	4	12.5	19.95	COOKING
3957136468	HOLY GRAIL OF ORACLE	31-DEC-01	3	47.25	75.95	COMPUTER
1915762492	HANDCRANKED COMPUTERS	21-JAN-01	3	21.8	25	COMPUTER
9959789321	E-BUSINESS THE EASY WAY	01-MAR-02	2	37.9	54.5	COMPUTER
2491748320	PAINLESS CHILD-REARING	17-JUL-00	5	48	89.95	FAMILY LIFE
0299282519	THE WOK WAY TO COOK	11-SEP-00	4	19	28.75	COOKING
8117949391	BIG BEAR AND LITTLE DOVE	08-NOV-01	5	5.32	8.95	CHILDREN
0132149871	HOW TO GET FASTER PIZZA	11-NOV-02	4	17.85	29.95	SELF HELP
9247381001	HOW TO MANAGE THE MANAGER	09-MAY-99	1	15.4	31.95	BUSINESS
2147428890	SHORTEST POEMS	01-MAY-01	5	21.85	39.95	LITERATURE

BOOKAUTHOR table

ISBN	AUTHORID
0132149871	S100
0299282519	S100
0401140733	J100
1059831198	P100
1059831198	S100
1915762492	W100
1915762492	W105
2147428890	W105
ISBN	**AUTHORID**
2491748320	B100
2491748320	F100
2491748320	R100

3437212490	B100
3957136468	A100
4981341710	K100
8117949391	R100
8843172113	A100
8843172113	A105
8843172113	P105
9247381001	W100
9959789321	J100

Orders table

ORDER#	CUSTOMER#	ORDERDATE	SHIPDATE	SHIPSTREET	SHIPCITY	SHIPST	SHIPZIP
1000	1005	31-MAR-03	02-APR-03	1201 ORANGE AVE	SEATTLE	WA	98114
1001	1010	31-MAR-03	01-APR-03	114 EAST SAVANNAH	ATLANTA	GA	30314
1002	1011	31-MAR-03	01-APR-03	58 TILA CIRCLE	CHICAGO	IL	60605
1003	1001	01-APR-03	01-APR-03	958 MAGNOLIA LANE	EASTPOINT	FL	32328
1004	1020	01-APR-03	05-APR-03	561 ROUNDABOUT WAY	TRENTON	NJ	08601
1005	1018	01-APR-03	02-APR-03	1008 GRAND AVENUE	MACON	GA	31206
1006	1003	01-APR-03	02-APR-03	558A CAPITOL HWY.	TALLAHASSEE	FL	32307
1007	1007	02-APR-03	04-APR-03	9153 MAIN STREET	AUSTIN	TX	78710
1008	1004	02-APR-03	03-APR-03	69821 SOUTH AVENUE	BOISE	ID	83707
1009	1005	03-APR-03	05-APR-03	9 LIGHTENING RD.	SEATTLE	WA	98110
1010	1019	03-APR-03	04-APR-03	384 WRONG WAY HOME	MORRISTOWN	NJ	07960
1011	1010	03-APR-03	05-APR-03	102 WEST LAFAYETTE	ATLANTA	GA	30311
1012	1017	03-APR-03		1295 WINDY AVENUE	KALMAZOO	MI	49002
1013	1014	03-APR-03	04-APR-03	7618 MOUNTAIN RD.	CODY	WY	82414
1014	1007	04-APR-03	05-APR-03	9153 MAIN STREET	AUSTIN	TX	78710
ORDER#	**CUSTOMER#**	**ORDERDATE**	**SHIPDATE**	**SHIPSTREET**	**SHIPCITY**	**SHIPST**	**SHIPZIP**
1015	1020	04-APR-03		557 GLITTER ST.	TRENTON	NJ	08606
1016	1003	04-APR-03		9901 SEMINOLE WAY	TALLAHASSEE	FL	32307

1017	1015	04-APR-03	05-APR-03	887 HOT ASPHALT ST	MIAMI	FL	33112
1018	1001	05-APR-03		95812 HIGHWAY 98	EASTPOINT	FL	32328
1019	1018	05-APR-03		1008 GRAND AVENUE	MACON	GA	31206
1020	1008	05-APR-03		195 JAMISON LANE	CHEYENNE	WY	82003

ORDERITEMS TABLE

ORDER#	ITEM#	ISBN	QUANTITY
1000	1	3437212490	1
1001	1	9247381001	1
1001	2	2491748320	1
1002	1	8843172113	2
1003	1	8843172113	1
1003	2	1059831198	1
1003	3	3437212490	1
1004	1	2491748320	2
1005	1	2147428890	1
1006	1	9959789321	1
1007	1	3957136468	3
1007	2	9959789321	1
1007	3	8117949391	1
1007	4	8843172113	1
1008	1	3437212490	2
1009	1	3437212490	1
1009	2	0401140733	1
1010	1	8843172113	1
1011	1	2491748320	1
1012	1	8117949391	1
1012	2	1915762492	2
1012	3	2491748320	1
1012	4	0401140733	1

1013	1	8843172113	1

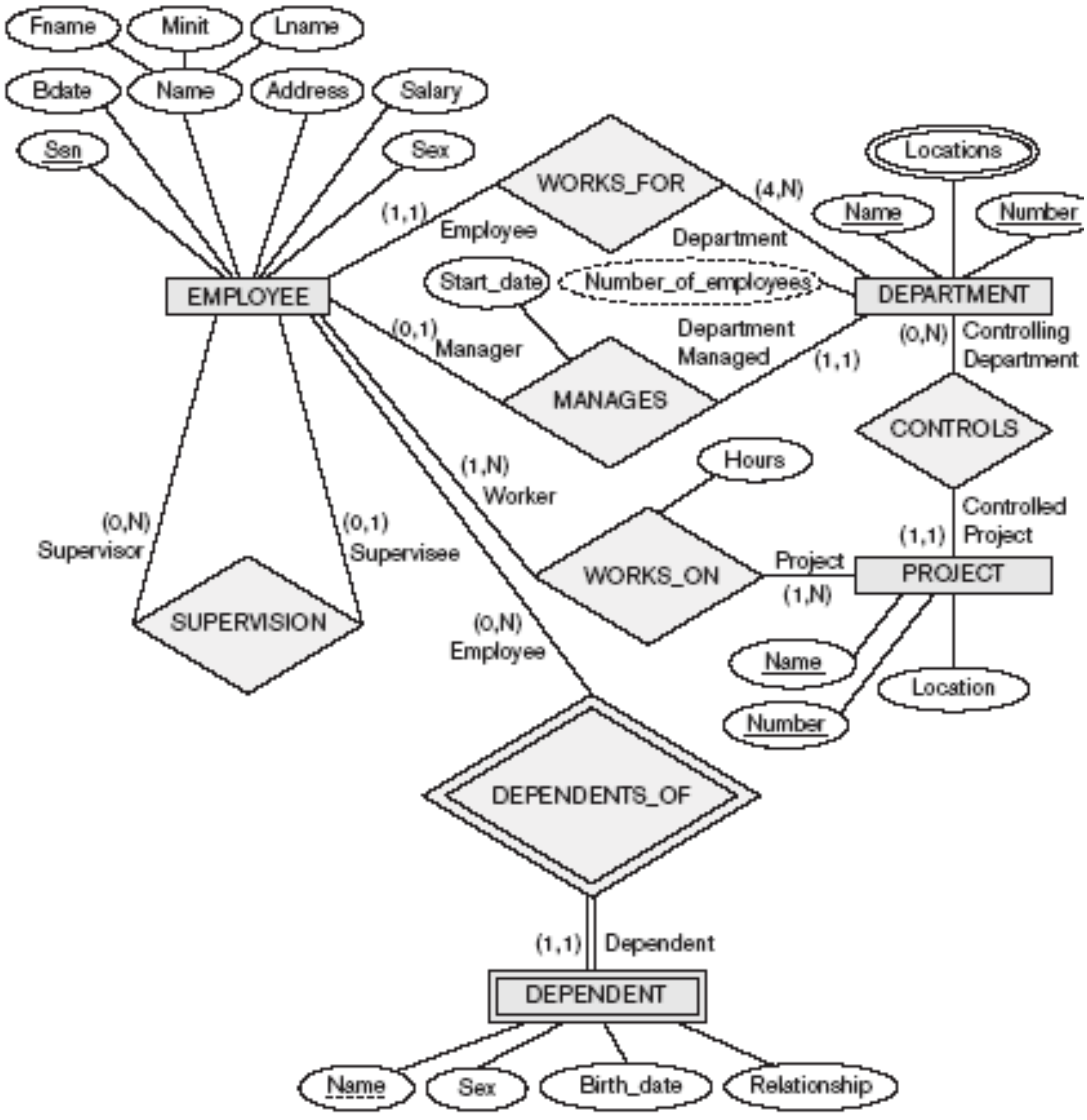
Promotion table

GIFT	MINRETAIL	MAXRETAIL
BOOKMARKER	0	12
BOOK LABELS	12.01	25
BOOK COVER	25.01	56
FREE SHIPPING	56.01	999.99

Table used to create a Justlee Database

Create table Customers (Customer# NUMBER(4) PRIMARY KEY, LastName VARCHAR2(10), FirstName VARCHAR2(10), Address VARCHAR2(20), City VARCHAR2(12), State VARCHAR2(2), Zip VARCHAR2(5), Referred NUMBER(4));	Create Table Orders (Order# NUMBER(4) PRIMARY KEY, Customer# NUMBER(4), OrderDate DATE, ShipDate DATE, ShipStreet VARCHAR2(18), ShipCity VARCHAR2(15), ShipState VARCHAR2(2), ShipZip VARCHAR2(5));
Create Table Publisher (PubID NUMBER(2) PRIMARY KEY, Name VarCHAR2(23), Contact VARCHAR2(15), Phone VARCHAR2(12));	Create Table Author (AuthorID Varchar2(4) PRIMARY KEY, Lname VARCHAR2(10), Fname VARCHAR2(10));
Create table Books (ISBN VARCHAR2(10) PRIMARY KEY, Title VARCHAR2(30), PubDate DATE, PubID NUMBER (2), Cost NUMBER (5,2), Retail NUMBER (5,2), Category VARCHAR2(12));	CREATE TABLE ORDERITEMS (ORDER# NUMBER(4) NOT NULL, ITEM# NUMBER(2) NOT NULL, ISBN VARCHAR2(10), QUANTITY NUMBER(3), constraint pk_orderitems PRIMARY KEY (order#, item#));
CREATE TABLE BOOKAUTHOR (ISBN VARCHAR2(10), AUTHORid VARCHAR2(4), CONSTRAINT pk_bookauthor PRIMARY KEY (isbn,authorid));	create table promotion (gift varchar2(15), minretail number(5,2), **maxretail number(5,2));**

ER Diagram for Company Data base:



Schema diagram for the COMPANY database:

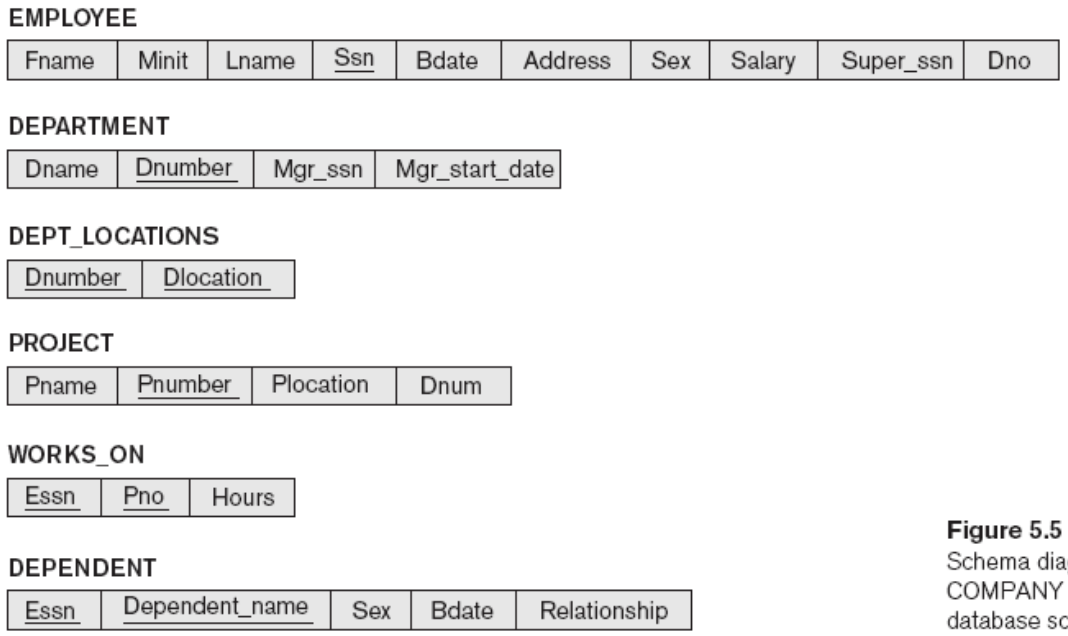
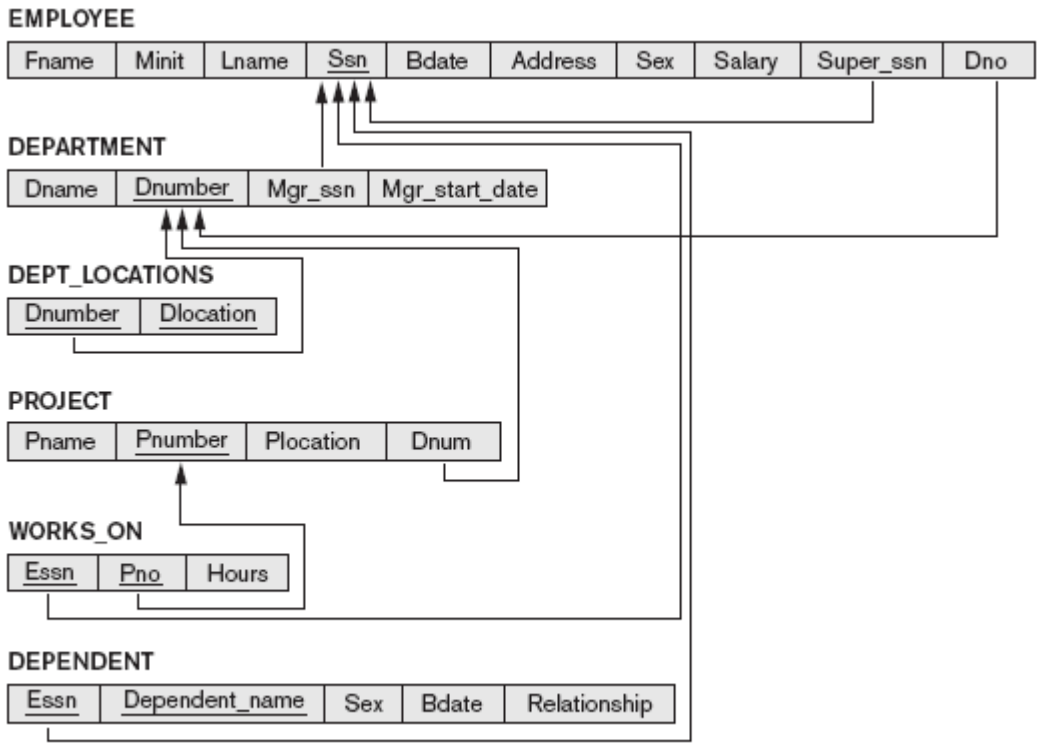


Figure 5.5
Schema diagram for the COMPANY relational database schema.

Entity and Referential integrity constraints on COMPANY relational database schema:



Possible Tuples (database state) on COMPANY relational database:

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Data Definition Statements for COMPANY database relation:

```
CREATE TABLE EMPLOYEE
        ( Fname                 VARCHAR(15)             NOT NULL,
          Minit                 CHAR,
          Lname                 VARCHAR(15)             NOT NULL,
          Ssn                   CHAR(9)                 NOT NULL,
          Bdate                 DATE,
          Address               VARCHAR(30),
          Sex                   CHAR,
          Salary                DECIMAL(10,2),
          Super_ssn             CHAR(9),
          Dno                   INT                     NOT NULL,
        PRIMARY KEY (Ssn),
CREATE TABLE DEPARTMENT
        ( Dname                 VARCHAR(15)             NOT NULL,
          Dnumber               INT                     NOT NULL,
          Mgr_ssn               CHAR(9)                 NOT NULL,
          Mgr_start_date        DATE,
        PRIMARY KEY (Dnumber),
        UNIQUE (Dname),
        FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn) );
CREATE TABLE DEPT_LOCATIONS
        ( Dnumber               INT                     NOT NULL,
          Dlocation             VARCHAR(15)             NOT NULL,
        PRIMARY KEY (Dnumber, Dlocation),
        FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber) );
CREATE TABLE PROJECT
        ( Pname                 VARCHAR(15)             NOT NULL,
          Pnumber               INT                     NOT NULL,
          Plocation             VARCHAR(15),
          Dnum                  INT                     NOT NULL,
        PRIMARY KEY (Pnumber),
        UNIQUE (Pname),
        FOREIGN KEY (Dnum) REFERENCES DEPARTMENT(Dnumber) );
CREATE TABLE WORKS_ON
        ( Essn                  CHAR(9)                 NOT NULL,
          Pno                   INT                     NOT NULL,
          Hours                 DECIMAL(3,1)            NOT NULL,
        PRIMARY KEY (Essn, Pno),
        FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn),
        FOREIGN KEY (Pno) REFERENCES PROJECT(Pnumber) );
CREATE TABLE DEPENDENT
        ( Essn                  CHAR(9)                 NOT NULL,
          Dependent_name        VARCHAR(15)             NOT NULL,
          Sex                   CHAR,
          Bdate                 DATE,
          Relationship          VARCHAR(8),
        PRIMARY KEY (Essn, Dependent_name),
        FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn) );
```

Ex.2 Data Manipulation Commands on SQL

DATA MANIPULATION LANGUAGE (DML)

It describes a group of commands with which you can modify the contents of the table

The Select statement Syntax With Additional Clauses:-

Syntax:

```
Select  [ Distinct / Unique ] ( *columnname [ As alias}, ….]
        From    tablename
        [ where  condition ]
         [ Group BY group _by_expression ]
        [Having  group_condition ]
       [ORDER BY {col(s)|expr|numeric_pos} [ASC|DESC] [NULLS
FIRST|LAST]];
```

Relational Operator

Operators	**Functions**
=	Equal to
<> or !=	Not equal to
>	Greater than
<	Less than
>=	Greater than or Equal to
<=	Less than or Equal to
||	Concatenation operator

Logical Operators

Operators	**Functions**
NOT	Reverse of result of logical expression
AND	Check two conditions are true
OR	Check any of the two conditions are true

Other Operators

Operators	Function
IN	Equal to any member of a given set
NOT IN	Not equal to any member of a given set
BETWEEN	In between values
NOT BETWEEN	Not in between values
IS NULL	True if value is null
IS NOT NULL	True if value is not null

1. INSERT:

Insert command insert one or more rows into a table.

Syntax:

INSERT INTO<table name> values (value1, value2, value3….);

Example:

```
SQL>insert into emp1 values (7954,'SMITH','CLERK', 7902,'17-DEC-1980'
      ,800,NULL,20);
SQL> insert into emp1 values
     (&empno,'&ename','&job',&mgr,'&hiredate',&sal, comm);
```

2. SIMPLE SELECT STATEMENT:

To view records from a table

Syntax:

SELECT * from <tablename>

SELECT column1, column2 from <tablename>

Example

```
SQL>select * from emp1
SQL>select eno,ename from emp1
```

3.SELECT STATEMENT WITH CONDITION:

Retrieve records from a table with conditions.

Syntax:

SELECT * from <tablename> where <condition>

SELECT column1, column2 from <tablename> where <condition> AND | OR <condition>

Example

```
SQL>select * from emp1 where eno=1234;
SQL>select eno,ename from emp1   where dno=10 and job='clerk';
```

4. DELETE:

To delete particular record from a table.

Syntax:

Delete from <tablename> where <condition>

Example:

```
SQL> Delete from emp1 where ename='john';
```

Sample Queries:

1.INSERTION:

```
SQL> insert into empo values(&eno,'&ename','&job',
&mgr,'&hiredate',&salary,&comm,&dno,&child);

SQL> insert into empo values(103,'ccc','manager',null,'21-jan-1989',50000,null,30,1);

SQL> insert into depo values(&dno,'&dname','&location');
```

2.SELECT STATEMENT:

```
SQL> select * from empo;

ENO   ENAME   JOB     MGR   HIREDATE   SALARY  COMM  DNO  CHILD
---- -------- -------- ----- ---------  ------  ----  ----- ----
101     aaa   salesman 7890  21-JAN-90  20000         10     1
102     bbb    clerk   7891  21-FEB-91  25000        20     2
103     ccc   manager  7892  21-JAN-89  50000         30     1
104     ddd   salesman 7893  21-MAR-92  20000         40     1 105     eee   salesman 7894  20-
JUL-94  10000         10     1
```

3.DELETE OPERATION:

```
SQL>delete from depo where dno=40;

1 row deleted.

SQL> select * from depo;

      DNO DNAME      LOCATION
      --- ---------- ----------
       10 sales      trichy
       20 management
       30 account    chennai
```

SELECT Statements with AND, OR, NOT,IN,BETWEEN, LIKE, DISTINCT, UNIQUE operators:

1.AND OPERATION:

```
SQL> select ename,salary,dno,from empo where job='salesman' AND dno=10;
    ENAME    SALARY   DNO
  --------  -------  -----                          aaa      20000     10
    eee      10000     10
```

2.OR OPERATION:

```
SQL> select * from empo where dno=10 or hiredate='7-dec-80';
```

```
ENO   ENAME  JOB      MGR  HIREDATE   SALARY  COMM  DNO  CHILD
---- -------- -------- ----- ---------  ------  ----  ----- ----
101    aaa   salesman 7890  21-JAN-90  20000         10    1
105    eee   salesman 7894  20-JUL-94  10000         10    1
```

3.IN OPERATION:

```
SQL> select ename,job from empo where job  in ('manager','salesman');
    ENAME   JOB
        ------- ---------
     aaa    salesman
     ccc    manager
     eee    salesman
```

4.BETWEEN OPERATION:

```
SQL> select * from empo where salary between 10000 and 20000;
ENO ENAME  JOB   MGR  HIREDATE  SALARY  COMM  DNO  CHILD
---- -------- -------- ----- ---------  ----  ----- ----
101 aaa  salesman 7890 21-JAN-90  20000       10     1 104 ddd  salesman 7893 21-MAR-92
20000       40     1 105 eee  salesman 7894 20-JUL-94  10000       10     1
```

5.DISTINCT:

```
SQL> select distinct dno from empo;
           DNO
                 ------
               10
                    20
               30
                    40
```

6.UNIQUE:

```
SQL> select unique salary from empo;
                SALARY
                ------
                20000
                25000
                50000
                20000
                10000
```

7.USING CONCATENATION OPERATOR:

```
SQL> select fname||lname from empo;
                          fname||lnameCHILD
                          ------------------
                           johnsmith
                           franklinwong
                           aiciazeyala
                        jenniferwallace
```

8.USING COLUMN ALAISES:

```
SQL> select fname||lname "ename" from emp114;
                        fname||lnameCHILD
                        ------------------
                         johnsmith
                         franklinwong
                         aiciazeyala
                      jenniferwallace
```

9.USING ARITHMETIC OPERATORS:

```
SQL> select ename,salary,salary+10,salary-10,salary*10,salary/10 from empo;
ENAME   SALARY   SALARY+10   SALARY-10   SALARY*10   SALARY/10
------- -------- ---------   ---------   ---------   ---------
 aaa    20000    20010       19990       200000      2000
 bbb    25000    25010       24990       250000      2500
 ccc    50000    50010       49990       500000      5000
 ddd    20000    20010       19990       200000      2000
 eee    10000    10010       9990        100000      1000
```

10.MATCHING A CHARACTER PATTERN(LIKE OPERATOR):

```
SQL> select eno,ename,job from empo where ename like 'a%';
ENO    ENAME   JOB
---- -------- --------
101     aaa   salesman
SQL> select eno,ename,job from empo where ename like 'b_b';
ENO    ENAME   JOB
---- -------- --------
101     bbb    clerk
```

INTRODUCTION TO SQL:

SQL stands for structured query language, which is used to communicate with the relational database, which are in turn a set of related information stored in the form of tables. SQL is a non-procedural language because it process sets of records rather than just one data at a time and also provides automatic navigation to the data.

The SQL language is subdivided according to their functions as follows:

Data Definition Language (DDL)
Or
Schema Definition Language.
Data Manipulation Language (DML)
Data Control Language (DCL)

Tables

In relational database systems (DBS) data are represented using tables (relations). A query issued against the DBS also results in a table.

A table has the following structure:

	Column 1	Column2	…… …..	Column n
Tuple (or Record->)				
Tuple (or Record->)				
Tuple (or Record->)	…..	…..	…..	……..
Tuple (or Record->)	……	……..	…….	………..

A table is uniquely identified by its name and consists of rows that contain the stored information, each row containing exactly one tuple (or record). A table can have one or more columns.A column is made up of a column name and a data type, and it describes an attribute of the tuples. The structure of a table, also called relation schema, thus is defined by its attributes.The type of information to be stored in a table is defined by the data types of the attributes at table creation time.

SQL uses the terms table, row, and column for relation, tuple, and attribute, respectively.

A table can have up to 254 columns which may have different or same data types and sets of values (domains), respectively. Possible domains are alphanumeric data (strings), numbers and date formats. Oracle offers the following basic data types:

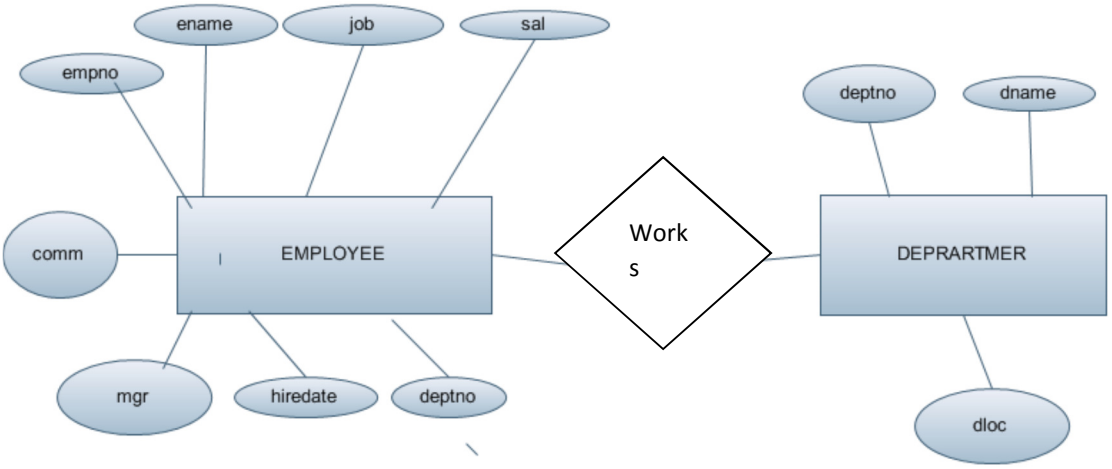
DATA TYPES

S.NO	Data Types	Description
1	Char(n)	Character String . n is the size of variable. Maximum size is 255 characters. The default size is 1
2.	Varchar2(n)	Character string . n is the size of the variable
3.	Number	Defines Numeric data type with space for 40 digit and space for sign and decimal point
4.	Number(n)	Numeric variable.n is the size of the variable

5.	Number(n,d)	Numeric variable.n is the size of variable and d is the size if the decimal point.
6.	Raw(size)	Raw Binary data of length size bytes .Maximum size is 32767 bytes.
7.	Integer	It is same as number data type but values will be whole numbers. Columns defined with this format not accept decimal values.
8.	Integer(n)	Specifies an integer data type of length n.
9.	Long	Defines a character data type upto 32760bytes. One one long column may be defined for table. This type of column may not be used in sub queries, Where clauses or indexes.
10.	Long Raw	Same as LONG except it contains binary data or byte strings and not interpreted by PL/SQL
11.	LOB Type	LOB variables can used interchangeably with LONG and LONG RAW variables. It consists of BFILE,BLOB,CLOB and NLOB
12.	BFILE	It is used to store large binary objects in operating system files outside the database
13.	BLOB	The BLOB datat ype to store large binary objects in the database, in-line or out-of-line. Every BLOB variable stores a locator, which points to a large binary object. The size of a BLOB cannot exceed four gigabytes.
14.	CLOB	The CLOB datatype to store large blocks of character data in the database, in-line or out-of-line. Both fixed-width and variable-width character sets are supported. Every CLOB variable stores a locator, which points to a large block of character data. The size of a CLOB cannot exceed four gigabytes
15.	NCLOB	The NCLOB datatype to store large blocks of NCHAR data in the database, in-line or out-of-line. Both fixed-width and variable-width character sets are supported. Every NCLOB variable stores a locator, which points to a large block of NCHAR data. The size of an NCLOB cannot exceed four gigabytes.
16.	DATE	The DATE datatype to store fixed-length datetimes, which include the time of day in seconds since midnight. The date portion defaults to the first day of the current month; the time portion defaults to midnight. The date function SYSDATE returns the current date and time.

Sample Databases used for illustration of SQL Commands is given below with ER Diagram and corresponding Relational Model with suitable data entered in the tables.

Example 1: EMPLOYEE and DEPARTMENT Tables.



- - Relation between Employee and Department is Works is many –to-one .

DATABASE for EMPLOYEE and DEPARTMENT Entities

EMP Table given with sample Data

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

DEPT Table given with Sample Data

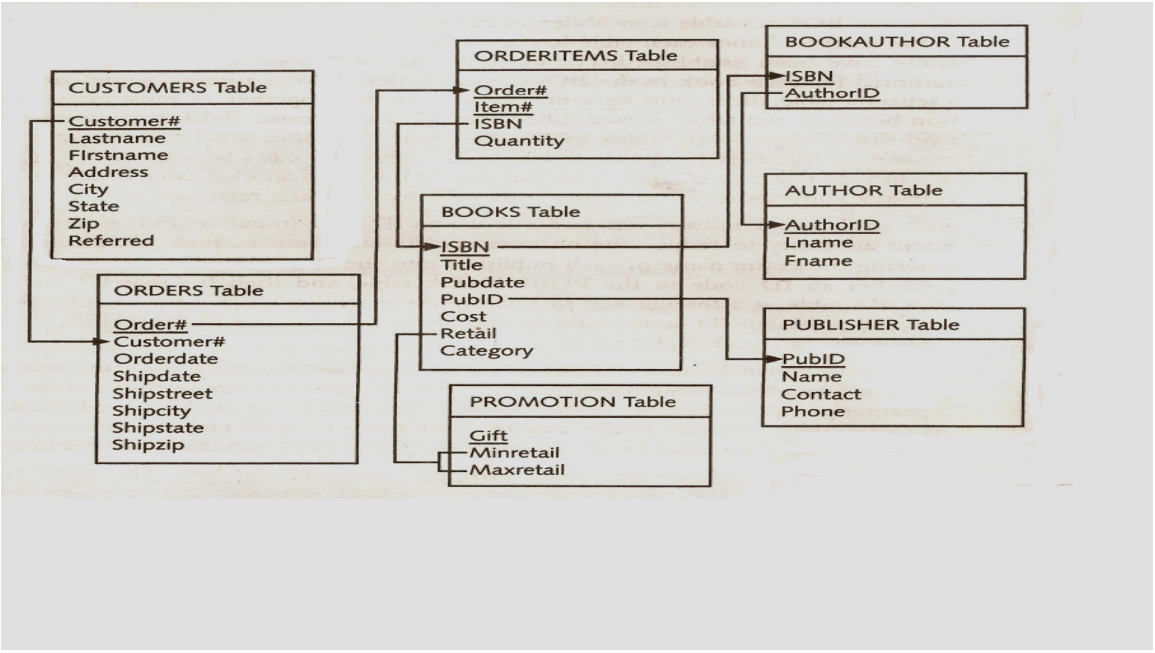
DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

Example 2: JUSTLEE BOOK'S DATABASE DESCRIPTION

The initial organization of the database structure for JustLee Books – the Case study for this RDBMS lab. The database is used first to record customer's orders. Customers and JustLee's employee can identify a book by its ISBN, title or author's name(s). Employees can also determine when a particular order was placed and when or if the order was shipped. The database also stores the publisher contact information so the bookseller can reorder a book.

Basic Assumptions

Three assumptions made when designing the database are as follows:

1. An order is not shipped until all items are available (i.e there are no back orders or partial order shipments).
2. All address are within the United States ; Otherwise , the Address /Zip Code fields would need to be altered.
3. Only orders for the current month or orders from previous months that have not yet shipped are stored in the ORDERS table. The following figure illustrates the structure of JustLee database.
4. 

Customers Table

CUSTOMER#	LASTNAME	FIRSTNAME	ADDRESS	CITY	STATE	ZIP	REFERRED
1001	MORALES	BONITA	P.O. BOX 651	EASTPOINT	FL	32328	
1002	THOMPSON	RYAN	P.O. BOX 9835	SANTA MONICA	CA	90404	
1003	SMITH	LEILA	P.O. BOX 66	TALLAHASSEE	FL	32306	
1004	PIERSON	THOMAS	69821 SOUTH AVENUE	BOISE	ID	83707	
1005	GIRARD	CINDY	P.O. BOX 851	SEATTLE	WA	98115	
1006	CRUZ	MESHIA	82 DIRT ROAD	ALBANY	NY	12211	
1007	GIANA	TAMMY	9153 MAIN STREET	AUSTIN	TX	78710	1003
1008	JONES	KENNETH	P.O. BOX 137	CHEYENNE	WY	82003	
1009	PEREZ	JORGE	P.O. BOX 8564	BURBANK	CA	91510	1003
1010	LUCAS	JAKE	114 EAST SAVANNAH	ATLANTA	GA	30314	
1011	MCGOVERN	REESE	P.O. BOX 18	CHICAGO	IL	60606	
1012	MCKENZIE	WILLIAM	P.O. BOX 971	BOSTON	MA	02110	
1013	NGUYEN	NICHOLAS	357 WHITE EAGLE AVE.	CLERMONT	FL	34711	1006
1014	LEE	JASMINE	P.O. BOX 2947	CODY	WY	82414	
1015	SCHELL	STEVE	P.O. BOX 677	MIAMI	FL	33111	
1016	DAUM	MICHELL	9851231 LONG ROAD	BURBANK	CA	91508	1010
1017	NELSON	BECCA	P.O. BOX 563	KALMAZOO	MI	49006	
1018	MONTIASA	GREG	1008 GRAND AVENUE	MACON	GA	31206	
1019	SMITH	JENNIFER	P.O. BOX 1151	MORRISTOWN	NJ	07962	1003
1020	FALAH	KENNETH	P.O. BOX 335	TRENTON	NJ	08607	

AUTHOR table

AUTHORID	LNAME	FNAME
S100	SMITH	SAM
J100	JONES	JANICE
A100	AUSTIN	JAMES
M100	MARTINEZ	SHEILA
K100	KZOCHSKY	TAMARA
P100	PORTER	LISA
A105	ADAMS	JUAN
B100	BAKER	JACK
P105	PETERSON	TINA
W100	WHITE	WILLIAM
W105	WHITE	LISA

R100	ROBINSON	ROBERT
F100	FIELDS	OSCAR
W110	WILKINSON	ANTHONY

BOOKS table

ISBN	TITLE	PUBDATE	PUBID	COST	RETAIL	CATEGORY
1059831198	BODYBUILD IN 10 MINUTES A DAY	21-JAN-01	4	18.75	30.95	FITNESS
0401140733	REVENGE OF MICKEY	14-DEC-01	1	14.2	22	FAMILY LIFE
4981341710	BUILDING A CAR WITH TOOTHPICKS	18-MAR-02	2	37.8	59.95	CHILDREN
8843172113	DATABASE IMPLEMENTATION	04-JUN-99	3	31.4	55.95	COMPUTER
3437212490	COOKING WITH MUSHROOMS	28-FEB-00	4	12.5	19.95	COOKING
3957136468	HOLY GRAIL OF ORACLE	31-DEC-01	3	47.25	75.95	COMPUTER
1915762492	HANDCRANKED COMPUTERS	21-JAN-01	3	21.8	25	COMPUTER
9959789321	E-BUSINESS THE EASY WAY	01-MAR-02	2	37.9	54.5	COMPUTER
2491748320	PAINLESS CHILD-REARING	17-JUL-00	5	48	89.95	FAMILY LIFE
0299282519	THE WOK WAY TO COOK	11-SEP-00	4	19	28.75	COOKING
8117949391	BIG BEAR AND LITTLE DOVE	08-NOV-01	5	5.32	8.95	CHILDREN
0132149871	HOW TO GET FASTER PIZZA	11-NOV-02	4	17.85	29.95	SELF HELP
9247381001	HOW TO MANAGE THE MANAGER	09-MAY-99	1	15.4	31.95	BUSINESS
2147428890	SHORTEST POEMS	01-MAY-01	5	21.85	39.95	LITERATURE

BOOKAUTHOR table

ISBN	AUTHORID
0132149871	S100
0299282519	S100
0401140733	J100
1059831198	P100
1059831198	S100
1915762492	W100
1915762492	W105
2147428890	W105
ISBN	**AUTHORID**
2491748320	B100
2491748320	F100
2491748320	R100

3437212490	B100
3957136468	A100
4981341710	K100
8117949391	R100
8843172113	A100
8843172113	A105
8843172113	P105
9247381001	W100
9959789321	J100

Orders table

ORDER#	CUSTOMER#	ORDERDATE	SHIPDATE	SHIPSTREET	SHIPCITY	SHIPST	SHIPZIP
1000	1005	31-MAR-03	02-APR-03	1201 ORANGE AVE	SEATTLE	WA	98114
1001	1010	31-MAR-03	01-APR-03	114 EAST SAVANNAH	ATLANTA	GA	30314
1002	1011	31-MAR-03	01-APR-03	58 TILA CIRCLE	CHICAGO	IL	60605
1003	1001	01-APR-03	01-APR-03	958 MAGNOLIA LANE	EASTPOINT	FL	32328
1004	1020	01-APR-03	05-APR-03	561 ROUNDABOUT WAY	TRENTON	NJ	08601
1005	1018	01-APR-03	02-APR-03	1008 GRAND AVENUE	MACON	GA	31206
1006	1003	01-APR-03	02-APR-03	558A CAPITOL HWY.	TALLAHASSEE	FL	32307
1007	1007	02-APR-03	04-APR-03	9153 MAIN STREET	AUSTIN	TX	78710
1008	1004	02-APR-03	03-APR-03	69821 SOUTH AVENUE	BOISE	ID	83707
1009	1005	03-APR-03	05-APR-03	9 LIGHTENING RD.	SEATTLE	WA	98110
1010	1019	03-APR-03	04-APR-03	384 WRONG WAY HOME	MORRISTOWN	NJ	07960
1011	1010	03-APR-03	05-APR-03	102 WEST LAFAYETTE	ATLANTA	GA	30311
1012	1017	03-APR-03		1295 WINDY AVENUE	KALMAZOO	MI	49002
1013	1014	03-APR-03	04-APR-03	7618 MOUNTAIN RD.	CODY	WY	82414
1014	1007	04-APR-03	05-APR-03	9153 MAIN STREET	AUSTIN	TX	78710
ORDER#	**CUSTOMER#**	**ORDERDATE**	**SHIPDATE**	**SHIPSTREET**	**SHIPCITY**	**SHIPST**	**SHIPZIP**
1015	1020	04-APR-03		557 GLITTER ST.	TRENTON	NJ	08606
1016	1003	04-APR-03		9901 SEMINOLE WAY	TALLAHASSEE	FL	32307

1017	1015	04-APR-03	05-APR-03	887 HOT ASPHALT ST	MIAMI	FL	33112
1018	1001	05-APR-03		95812 HIGHWAY 98	EASTPOINT	FL	32328
1019	1018	05-APR-03		1008 GRAND AVENUE	MACON	GA	31206
1020	1008	05-APR-03		195 JAMISON LANE	CHEYENNE	WY	82003

ORDERITEMS TABLE

ORDER#	ITEM#	ISBN	QUANTITY
1000	1	3437212490	1
1001	1	9247381001	1
1001	2	2491748320	1
1002	1	8843172113	2
1003	1	8843172113	1
1003	2	1059831198	1
1003	3	3437212490	1
1004	1	2491748320	2
1005	1	2147428890	1
1006	1	9959789321	1
1007	1	3957136468	3
1007	2	9959789321	1
1007	3	8117949391	1
1007	4	8843172113	1
1008	1	3437212490	2
1009	1	3437212490	1
1009	2	0401140733	1
1010	1	8843172113	1
1011	1	2491748320	1
1012	1	8117949391	1
1012	2	1915762492	2
1012	3	2491748320	1
1012	4	0401140733	1

1013	1	8843172113	1

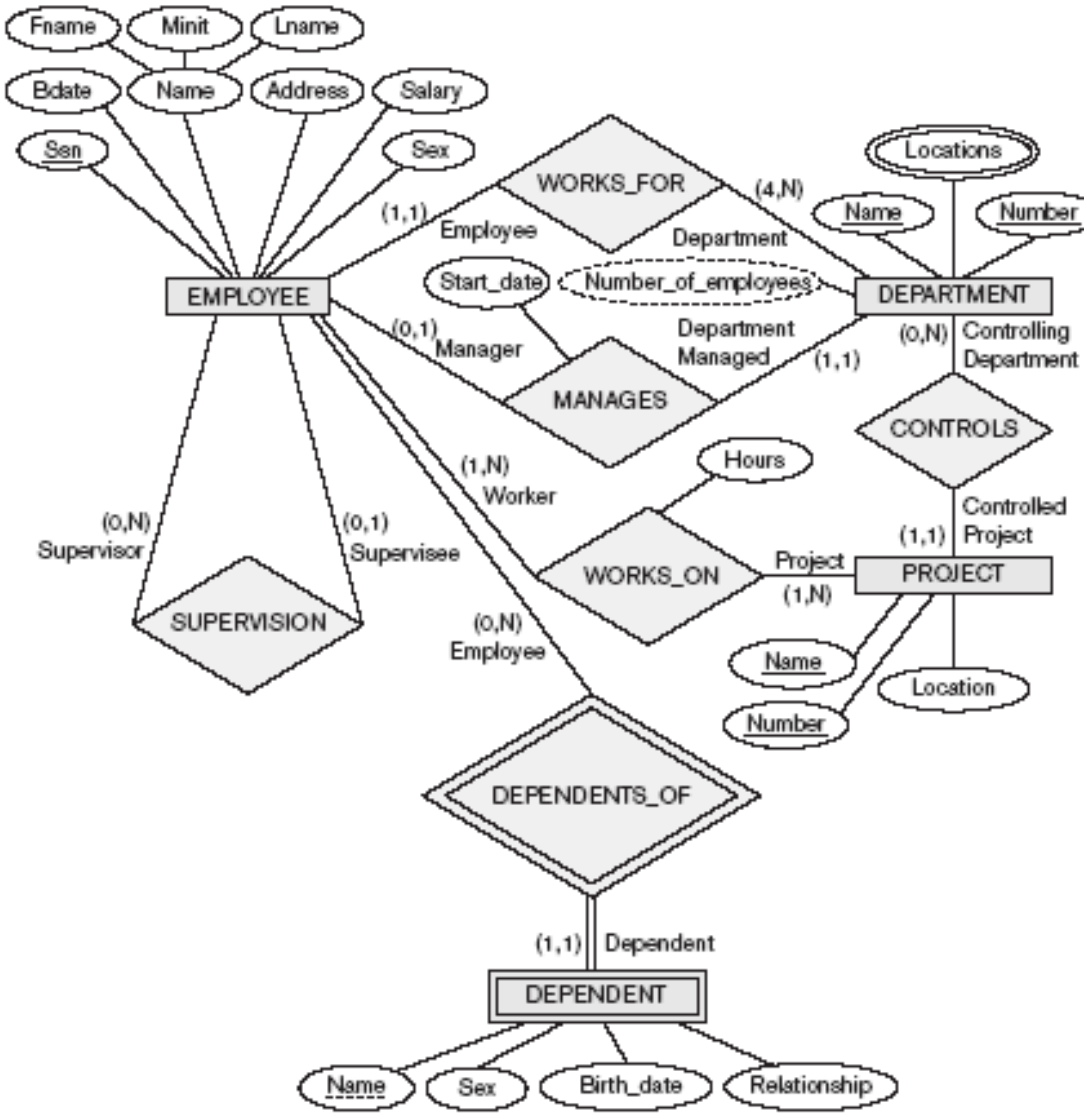
Promotion table

GIFT	MINRETAIL	MAXRETAIL
BOOKMARKER	0	12
BOOK LABELS	12.01	25
BOOK COVER	25.01	56
FREE SHIPPING	56.01	999.99

Table used to create a Justlee Database

Create table Customers (Customer# NUMBER(4) PRIMARY KEY, LastName VARCHAR2(10), FirstName VARCHAR2(10), Address VARCHAR2(20), City VARCHAR2(12), State VARCHAR2(2), Zip VARCHAR2(5), Referred NUMBER(4));	Create Table Orders (Order# NUMBER(4) PRIMARY KEY, Customer# NUMBER(4), OrderDate DATE, ShipDate DATE, ShipStreet VARCHAR2(18), ShipCity VARCHAR2(15), ShipState VARCHAR2(2), ShipZip VARCHAR2(5));
Create Table Publisher (PubID NUMBER(2) PRIMARY KEY, Name VarCHAR2(23), Contact VARCHAR2(15), Phone VARCHAR2(12));	Create Table Author (AuthorID Varchar2(4) PRIMARY KEY, Lname VARCHAR2(10), Fname VARCHAR2(10));
Create table Books (ISBN VARCHAR2(10) PRIMARY KEY, Title VARCHAR2(30), PubDate DATE, PubID NUMBER (2), Cost NUMBER (5,2), Retail NUMBER (5,2), Category VARCHAR2(12));	CREATE TABLE ORDERITEMS (ORDER# NUMBER(4) NOT NULL, ITEM# NUMBER(2) NOT NULL, ISBN VARCHAR2(10), QUANTITY NUMBER(3), constraint pk_orderitems PRIMARY KEY (order#, item#));
CREATE TABLE BOOKAUTHOR (ISBN VARCHAR2(10), AUTHORid VARCHAR2(4), CONSTRAINT pk_bookauthor PRIMARY KEY (isbn,authorid));	create table promotion (gift varchar2(15), minretail number(5,2), **maxretail number(5,2));**

ER Diagram for Company Data base:



Schema diagram for the COMPANY database:

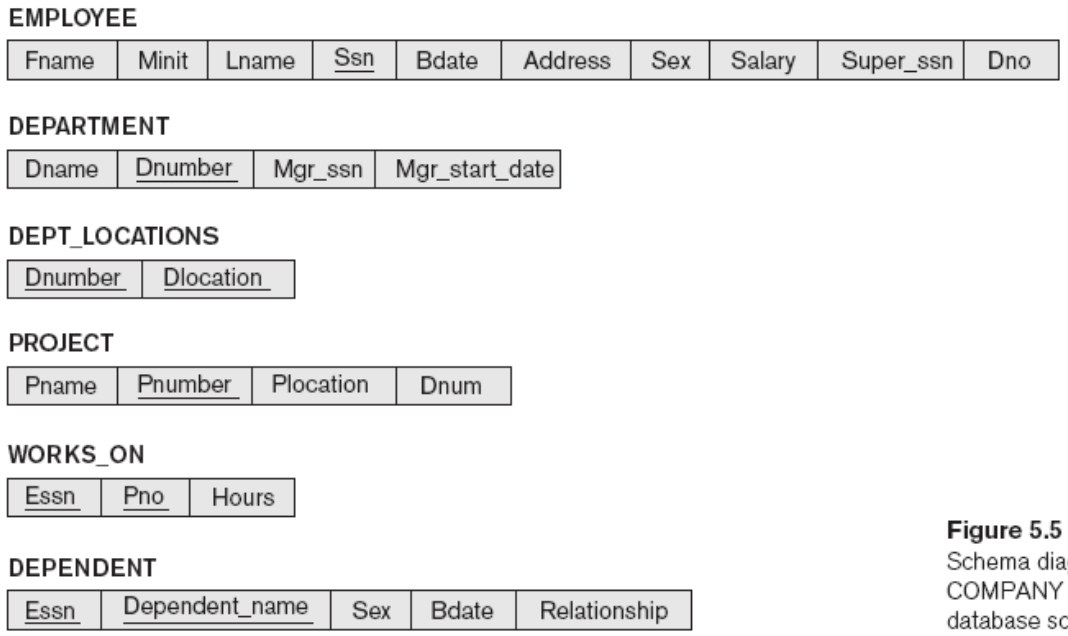
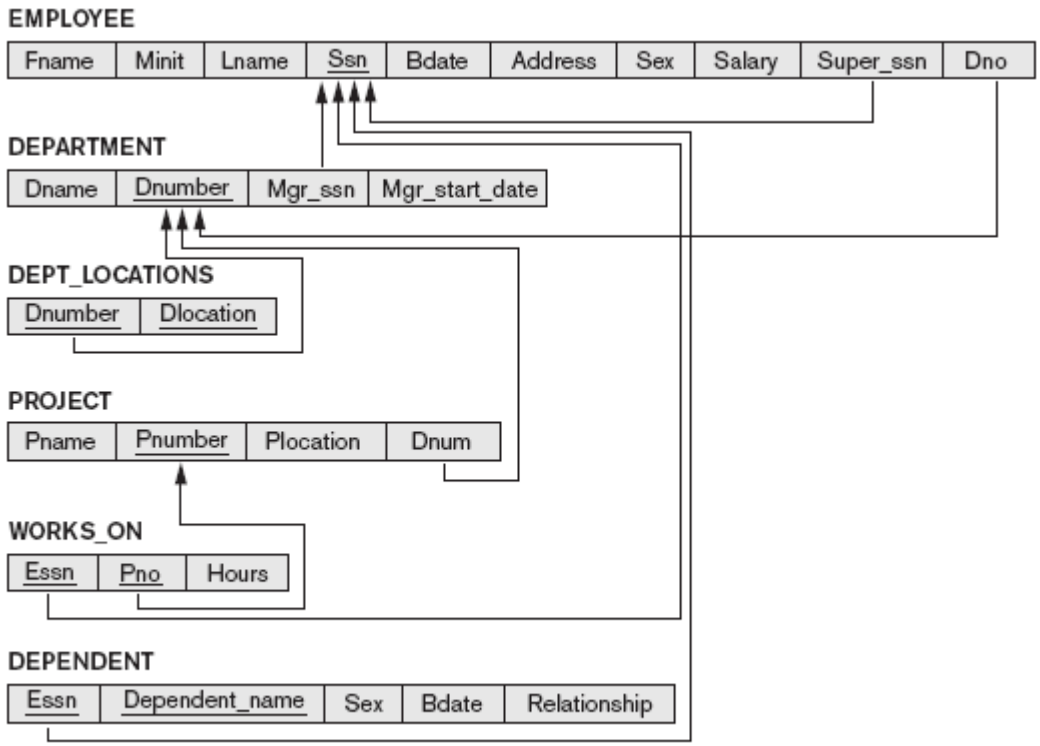


Figure 5.5
Schema diagram for the COMPANY relational database schema.

Entity and Referential integrity constraints on COMPANY relational database schema:



Possible Tuples (database state) on COMPANY relational database:

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Data Definition Statements for COMPANY database relation:

```
CREATE TABLE EMPLOYEE
        ( Fname                 VARCHAR(15)             NOT NULL,
          Minit                 CHAR,
          Lname                 VARCHAR(15)             NOT NULL,
          Ssn                   CHAR(9)                 NOT NULL,
          Bdate                 DATE,
          Address               VARCHAR(30),
          Sex                   CHAR,
          Salary                DECIMAL(10,2),
          Super_ssn             CHAR(9),
          Dno                   INT                     NOT NULL,
        PRIMARY KEY (Ssn),
CREATE TABLE DEPARTMENT
        ( Dname                 VARCHAR(15)             NOT NULL,
          Dnumber               INT                     NOT NULL,
          Mgr_ssn               CHAR(9)                 NOT NULL,
          Mgr_start_date        DATE,
        PRIMARY KEY (Dnumber),
        UNIQUE (Dname),
        FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn) );
CREATE TABLE DEPT_LOCATIONS
        ( Dnumber               INT                     NOT NULL,
          Dlocation             VARCHAR(15)             NOT NULL,
        PRIMARY KEY (Dnumber, Dlocation),
        FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber) );
CREATE TABLE PROJECT
        ( Pname                 VARCHAR(15)             NOT NULL,
          Pnumber               INT                     NOT NULL,
          Plocation             VARCHAR(15),
          Dnum                  INT                     NOT NULL,
        PRIMARY KEY (Pnumber),
        UNIQUE (Pname),
        FOREIGN KEY (Dnum) REFERENCES DEPARTMENT(Dnumber) );
CREATE TABLE WORKS_ON
        ( Essn                  CHAR(9)                 NOT NULL,
          Pno                   INT                     NOT NULL,
          Hours                 DECIMAL(3,1)            NOT NULL,
        PRIMARY KEY (Essn, Pno),
        FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn),
        FOREIGN KEY (Pno) REFERENCES PROJECT(Pnumber) );
CREATE TABLE DEPENDENT
        ( Essn                  CHAR(9)                 NOT NULL,
          Dependent_name        VARCHAR(15)             NOT NULL,
          Sex                   CHAR,
          Bdate                 DATE,
          Relationship          VARCHAR(8),
        PRIMARY KEY (Essn, Dependent_name),
        FOREIGN KEY (Essn) REFERENCES EMPLOYEE(Ssn) );
```

EX.NO: 1 DATABASE CREATION AND SIMPLE QUERIES

TABLE CREATION:

Syntax:

```
CREATE TABLE <TABLENAME>
                    (Column name 1 datatype,
                     Column name 2 datatype,
                   Column name 3 data type…);
CREATE TABLE<TABLENAME>
              (Column name1 datatype Column Constraint,
               Column name 2 datatype Column Constraint,
                                                    :
                                                    :
          Column name n data type Column Constraint  n)
```

Constraints in Table Creation:

Constraints are condition for the data item to be stored into a database. There are two types of Constraints viz., Column Constraints and Table Constraints.

Syntax:

```
[CONSTRAINT constraint name]
{[NOT] NULL / UNIQUE / PRIMARY KEY}(Column[,column]..)
FOREIGN KEY (column [, colum]…)
REFERENCES table(columnname)
[ON DELETE CASCADE]
[CHECK (condition)]
```

These are the following constraints:

PRIMARY KEY
FOREIGN KEY
UNIQUE KEY
NOT NULL
DEFAULT

PRIMARY KEY:

Used to uniquely identify the records in a table. It does not accept NULL and duplicate values. Its value never changes.

FOREIGN KEY:

Primary Key of the parent table becomes the Foreign Key of the child table.

UNIQUE KEY:

It ensures that the particular value has distinct values. It accepts NULL value and can be changed.

NOT NULL:

Used to prevent the user from skipping any values.

DEFAULT:

It is used to assign default value to the field when it is not supplied.

ON DELETE CASCADE:

Allows deletion of REFERENCED key values in the parent table that has dependent rows in the child table and the dependent rows of the child table are automatically deleted.

CHECK:

Each row in the table must satisfy the specified check condition.

Column Constraints:

These are applied only to the individual columns of the table. It defines the restrictions placed on the column.

Table Constraints:

Table constraints are applied to group of one or more columns. It is identical to column constraints except that it can reference multiple columns with a single constraint. For example, in declaring three columns as a Primary Key or Foreign Key.

1.CREATE command:

```
SQL>create table emp
      (empno          number(5)  constraint pk_emp primary key,
       ename          varchar2(20)  constraint emp_nn  not null,
       hiredate       date,
       job            varchar2(20),
       sal            number(8,2),
       comm           number(4),
       deptno         number(2) references dept(deptno) on delete cascade);
SQL>create table dept1
  (deptno     number(2) constraint pk_dept primary key,
   dname varchar2(10) not null,
    loc  varchar2(20));
```

2.Describe command:

It is used to view the table structure to confirm whether the table was created correctly.

```
SQL> describe dept1
SQL> desc emp1
```

3.ALTER TABLE COMMAND:

To modify structure of an already existing table to add one more columns and also modify the existing columns.

Syntax:

```
ALTER TABLE <tablename>
ADD / MODIFY (columnname datatype(width));
```

Example
SQL>Alter table emp1 add(child number(1));
SQL>Alter table emp1 modify (empname varchar2(30));

4. DROP:

To remove a table along with its structure and data.

Syntax:

Drop table<table name>;

Example
SQL> Create table employee as Select * from emp;
employee table created successfully.
SQL> drop table employee;

5. TRUNCATE:

Delete all the records in the table retaining its structure.

Syntax: Truncate table <table name>;

Example
SQL> truncate table employee;

Sample SQL Queries:

1.TABLE CREATION:

SQL> create table empo(eno number(5),ename varchar(25),job varchar(25),mgr number(10),hiredate date,salary number(7,2),comm number(10),dno number(10));

Table created.

2.CONSTRAINTS IN TABLE CREATION:

SQL> create table depo(dno number(10)constraint pke_dept primary key,dname varchar(10)not null,location number(10));

Table created.

3.DESCRIBE COMMAND:

SQL> desc empo;

Name	Null?	Type
ENO		NUMBER(5)
ENAME		VARCHAR2(25)
JOB		VARCHAR2(25)
MGR		NUMBER(10)
HIREDATE		DATE
SALARY		NUMBER(7,2)
COMM		NUMBER(10)
DNO		NUMBER(10)

SQL> desc depo;

Name	Null?	Type
DNO	NOT NULL	NUMBER(10)
DNAME	NOT NULL	VARCHAR2(10)
LOCATION		NUMBER(10)

4.ALTER TABLE COMMAND:

SQL> alter table empo add(child number(1));
Table altered.

SQL> alter table empo modify(ename varchar(30));
Table altered.
SQL> desc empo;

Name	Null?	Type
ENO		NUMBER(5)
ENAME		VARCHAR2(30)
JOB		VARCHAR2(25)
MGR		NUMBER(10)
HIREDATE		DATE
SALARY		NUMBER(7,2)
COMM		NUMBER(10)
DNO		NUMBER(10)
CHILD		NUMBER(1)

5.DROP:

SQL> drop table depo;

Table dropped.

6.TRUNCATE:

SQL> truncate table depo;

Table truncated.

7.VIEWING CONSTRAINTS:

SQL> select constraint_name,constraint_type from
user_constraints;

CONSTRAINT_NAME	C
SYS_C005492	C
BIN$wUn12XlBTT+EI4Ro93Payg==$0	C
BIN$kC9vKLPLT0yRaelNR0OHgg==$0	C

```
BIN$OTvtVHrQQ8C9U/RXYONkzA==$0 C
SYS_C005468                    C
FK_DEPTNO                      R
PK_DEPT                        P
PK_EMP                         P
SRC_PK                         P
BIN$qwiYoLflRJ+9yOLVdQAMgA==$0 P
BIN$ZeZGvR5LSv6TlLzoVcW5Lw==$0 P

11 rows selected.
```

8.DISABLING CONSTRAINTS:

```
SQL> alter table depo disable constraint pke_dept;

Table altered.
```

9.ENABLING CONSTRAINTS:

```
SQL> alter table depo enable constraint pke_dept;

Table altered.
```

10.DROPPING CONSTRAINTS:

```
SQL> alter table depo drop constraints pke_dept;

Table altered.
```

11.ALTER TABLE DROP COLUMN COMMAND:

```
SQL> alter table empo drop column child;

Table altered.
```

12.ADDING THE PRIMARY KEY CONSTRAINT TO EXISTING TABLE:

```
SQL> alter table empo add constraint pkey_empo_eno primary key (eno);

Table altered.
```

13.ADDING REFERENCE BETWEEN EXISTING TABLE(FOREIGN CONSTRAINT):

```
SQL> alter table empo add constraint empo_depo_fkey foreign key(dno) references depo(dno);

Table altered.
```

14.USE OF ON DELETE CASCADE:

SQL> alter table empo add constraint empo_depo_fkey foreign key(dno) references depo(dno) on delete cascade;

Table altered.

15.ADDING THE UNIQUE KEY CONSTRAINT TO EXISTING TABLE:

SQL> alter table empo add constraint empo_eno_uni unique(ename);

Table altered.

16.ADDING THE CHECK CONSTRAINT TO EXISTING TABLE:

SQL> alter table empo add constraint sal_chk check (salary>5000);

Table altered.

17.ADDING THE NOT NULL CONSTRAINT TO EXISTING TABLE:

SQL> alter table empo modify(ename constraint nt_ename not null);

Table altered.