AMMINI COLLEGE OF ENGINEERING

CS09 607(P)

SYSTEMS LAB-DBMS

LAB MANUAL

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CONTENTS

Sl.No	Name of the Experiment	Page No
1	BASIC SQL QUERIES	2
2	SELECT STATEMENTS	7
3	UNION,INTERSECTION,JOIN OPERATION	14
4	SORTING AND GROUPING	19
5	NESTED QUERIES	22
6	SQL BUILT- IN FUNCTIONS	25
7	VIEWS	27
8	DATABASE APPLICATION	29
9	FUNCTIONS	38
10	PROCEDURES	46
11	IMPLICIT CURSORS	54
12	EXPLICIT CURSORS	57
13	TRIGGERS	60
14	OBJECTS	64
15	EXCEPTION HANDLING	66
16	PACKAGES	69
17	VIVA – VOICE	74

LAB 1:

BASIC SQL QURIES

- DDL COMMANDS
- DML COMMANDS

DDL COMMANDS

- CREATE
- ALTER
- DROP
- TRUNCATE

DML COMMANDS

- INSERT
- UPDATE
- DELETE
- SELET

DDL COMMANDS

CREATE

SQL> create table student(sno number(3),name varchar(20),marks number(3),dep varchar(2));

Table created

ALTER

SQL> alter table student add(age number(3));

Table altered

*Sql> select * from student;*

SNO	NAME	MARKS	DEP	AGE
1	Manoj	99	It	

MODIFY

SQL> alter tables student modify(dep varchar(3));

Table altered

*SQL> select * from student*

SNO	NAME	MARKS	DEP	AGE
1	Manoj	99	It	18

TRUNCATE

SQL> truncate table student

Table truncated

*SQL> select * from student*

No rows selected

DROP

SQL> alter table student drop column age

Table altered

*SQL> select * from student*

SNO	NAME	MARKS	DEP
1	Manoj	99	It

DML COMMANDS

INSERT

SQL> insert student values(&sno,'&name',&marks,'&dep');

Enter value for sno: 01

Enter value for name: Manoj

Enter value for marks: 99

Enter value for dep: IT

Old 1: insert into student values(&sno,'&name','&marks','&dep')

New 1: insert into student values(01,'Manoj',99,'IT')

1 row created

SQL> insert student values(&sno,'&name',&marks,'&dep');

Enter value for sno: 02

Enter value for name: Ramana

Enter value for marks: 98

Enter value for dep: IT

Old 1: insert into student values(&sno,'&name','&marks','&dep')

New 1: insert into student values(02,'Ramana',98,'IT')

1 row created

SQL> insert student values(&sno,'&name',&marks,'&dep');

Enter value for sno: 03

Enter value for name: Samy

Enter value for marks: 90

Enter value for dep: IT

Old 1: insert into student values(&sno,'&name','&marks','&dep')

New 1: insert into student values(03,'Samy',90,'IT')

1 row created

SQL> insert student values(&sno,'&name',&marks,'&dep');

Enter value for sno: 04

Enter value for name: Prabha

Enter value for marks: 99

Enter value for dep: IT

Old 1: insert into student values(&sno,'&name','&marks','&dep')

New 1: insert into student values(04,'Prabha',99,'IT')

1 row created

SELECT

*SQL> select * from student*

SNO	NAME	MARKS	DEP
1	Manoj	99	IT
2	Ramana	98	IT
1	Samy	90	IT
1	Prabha	99	IT

DELETE

SQL> delete from student

4 rows deleted

UPDATE

SQL> insert into student values(&sno,'&name',&marks,'&dep');

Enter value for sno: 01

Enter value for name: Manoj

Enter value for marks: 99

Enter value for dep: IT

Old 1: insert into student values(&sno,'&name','&marks','&dep')

New 1: insert into student values(01,'Manoj',99,'IT')

1 row created

SQL>update student set age=18;

1 row updated

*SQL>select * from student;*

SNO	NAME	MARKS	DEP	AGE
1	Manoj	99	It	18

LAB 2:

SELECT STATEMENTS

CREATE TABLE

SQL> create table student(id number, name varchar2(20),dept varchar2(4),grade va number);

Table created

INSERT

SQL>insert into student values(001,'RAGAV','IT','S',95);

1 row created

SQL> insert into student values(002,'KISHORE','CSE','S',94);

1 row is created

SQL> insert into student values(003,'KANNA','MECH','S',92);

1 row is created

SQL> insert into student values(004,'VASANTHA','DME','A',89);

1 row is created

SQL> insert into student values(005,'RAMU','CA','A',88);

1 row is created

SQL> insert into student values(006,'RATHA','WIPO','F',49);

1 row is created

SQL> insert into student values(007,'SANJAY','PET','B',78);

1 row is created

SQL> select id, name, dept from student

7 rows created.

1. SELECT ALL FROM A STUDENT TABLE

*SQL> select * from student*

ID	NAME	DEPT	GR	PERCENTAGE
1	RAGAV	IT	S	95
2	KISHORE	CSE	S	94
3	KANNA	MECH	S	92
4	VASANTHA	DME	A	88
5	RAMU	CA	A	88
6	RADHA	WIPO	F	49
7	SANJAY	PET	B	78

7 rows selected

2. SELECT ID, NAME, GRADE FROM A STUDENT TABLE

SQL>select id, name, grade from student

ID	NAME	GR
1	RAGAV	S
2	KISHORE	S
3	KANNA	S
4	VASANTHA	A

5 RAMU A

6 RADHA F

7 SANJAY B

3. SELECT ALL ROWS FROM A STUDENT TABLE WHERE ID=001

*SQL> select * from student where id=001*

ID	NAME	DEPT	GR	PERCENTAGE
1	RAGAV	IT	S	95

USING RELATIONAL OPERATORS

1. SELECT ROWS FROM STUDENT TABLE WHOSE PERCENTAGE > 90

*SQL> select * from student where percentage>90*

ID	NAME	DEPT	GRADE	PERCENTAGE
1	RAGAV	IT	S	95
2	KISHORE	CSE	S	94
3	KANNA	MECH	S	92

2. SELECT ROWS FROM STUDENT WHOSE PERCENTAGE>=50

*SQL> select * from student where percentage>=50*

ID	NAME	DEPT	GR	PERCENTAGE
1	RAGAV	IT	S	95
2	KISHORE	CSE	S	94
3	KANNA	MECH	S	92

4	VASANTHA	DME	A	88
5	RAMU	CA	A	88
7	SANJAY	PET	B	78

3. SELECT ROWS FROM STUDENT WHOSE PERCENTAGE<50

*SQL>select * from student where percentage<50*

ID	NAME	DEPT	GR	PERCENTAGE
6	RADHA	WIPO	F	49

4. SELECT ROWS FROM STUDENT WHOSE PERCENTAGE>=80 AND DEPT=IT

*SQL>select * from student where percentage>=80 and dept='IT'*

ID	NAME	DEPT	GR	PERCENTAGE
1	RAGAV	IT	S	95

5. SELECT ROWS FROM STUDENT WHOSE PERCENTAGE<50 AND GRADE='F'

*SQL>select * from student where percentage<50 and gr='f'*

ID	NAME	DEPT	GR	PERCENTAGE
6	RADHA	WIPO	F	49

6. SELECT ROWS FROM STUDENT WHOSE PERCENTAGE IS WHETHER 95 OR 92 OR 89

*SQL>select * from student where percentage=95 or percentage=92 or percentage=88*

ID	NAME	DEPT	GR	PERCENTAGE
1	RAGAV	IT	S	95
3	KANNA	MECH	S	92
4	VASANTHA	DME	A	88
5	RAMU	CA	A	88

7. SELECT ROWS FROM STUDENT WHOSE PERCENTAGE IS BETWEEN 80 AND 90

*SQL>select * from student where percentage between 80 and 90*

ID	NAME	DEPT	GR	PERCENTAGE
4	VASANTHA	DME	A	88
5	RAMU	CA	A	88

8. SELECT PERCENTAGE FROM STUDENT WHOSE NAME HAS SUBSTREAM RAG

SQL> select percentage from student where name='RAG'

PERCENTAGE

95

9. INSERT DATA FROM ANOTHER TABLE

SQL> insert in to selva select id,name,dept from student

Rows 7 created

*SQL> select * from selva*

ID	NAME	DEPT	GR	PERCENTAGE
1	RAGAV	IT	S	95
2	KISHORE	CSE	S	94
3	KANNA	MECH	S	92
4	VASANTHA	DME	A	88
5	RAMU	CA	A	88
6	RADHA	WIPO	F	49
7	SANJAY	PET	B	78

10.INSERTION OF A PART OF COLUMN IN A ROW

SQL> insert into selva (id, name)values(9,'PRABU');

1 row created

*SQL> select * from selva*

ID	NAME	DEPT	GR	PERCENTAGE
1	RAGAV	IT	S	95
2	KISHORE	CSE	S	94
3	KANNA	MECH	S	92
4	VASANTHA	DME	A	88
5	RAMU	CA	A	88
6	RADHA	WIPO	F	49
7	SANJAY	PET	B	78
9	PRABU			

LAB 3:

UNION, INTERSECTION AND JOIN OPERATIONS

SQL> create table depositor (name varchar2(20),accountno number(4));

Table created

SQL> create table borrower (name varchar2(20), loanno number(4));

Table created

DEPOSITER TABLE

*SQL> select * from depositor*

NAME	ACCOUNTNO
Kishore	2525
Ratha	3535
Ramu	4545
Vasantha	6565
Kanna	7575

BORROWER TABLE

*SQL> select * from borrower*

NAME	LOANNO
Ragav	1122
Santhia	2233
Rengaran	3344
Jeganathan	4455

Manikandan 5566

Ratha 6677

Ramu 7788

7 rows selected

UNION

SQL> select name from depositor union (select name from borrower)

NAME

Jeganathan

Kanna

Kishor

Manikandan

Ragav

Ramu

Ratha

Rengaraj

Santhia

Vasantha

10 rows selected

SQL> select name from depositor union all(select name from borrower)

NAME

Kishor

Ratha

Ramu

Vasantha

Kanna

Ragav

Santhia

Rengaraj

Jeganathan

Manikandan

Ratha

Ramu

12 rows selected

INTERSECT

SQL> select name from depositor intersect (select name from borrower)

NAME

Ramu

Ratha

MINUS

SQL> select name from depositor minus(select name from borrower)

NAME

Kanna

Kishor

Vasantha

STUDENT TABLE

SQL> create table student5 (name varchar2(20),idno number(5))

Table created

*SQL> select * from student5;*

NAME	IDNO	DEPTCODE
Gomathi	1111	1
Mohan	2222	2
Raju	3333	1
Roja	4444	3
Santha	5555	1
Dinesh	6666	4
Sanjay	7777	5

7 rows selected

DEPARTMENT TABLE

SQL> create table department (department code number(5),department name varchar2(20));

Table created

*SQL> select * from department*

DEPARTMENT CODE	DEPARTMENTNAME
1	it
2	bt
3	ece
4	eee
5	at

JOIN

INNER JOIN

SQL> select name, department name from student5, department where student5.deptcode=department.depart

NAME	DEPARTMENT NAME
Gomathi	it
Raju	it
Santha	it
Mohan	bt
Roja	ece
Dinesh	eee
Sanjay	at

7 rows selected

OUTER JOIN

SQL> select name, department name from student5,department where student5.deptcode=department.departmentcode(+) order by name

NAME	DEPARTMENT NAME
Dinesh	eee
Gomathi	it
Mohan	bt
Raju	it
Roja	ece
Sanjay	at
Santha	it

7 rows selected

LAB 4:

SORTING AND GROUPING

Order by id

*SQL>select * from student order by id*

NAME	ID	AGE	DEPT
Manoj	1234	19	it
Vijay	2345	18	pt
Ahok	3456	19	bt
Dinesh	3456	21	pt
Karthi	5678	19	at
Vivek	45455	20	cse
Dharani	541014	21	it

7 rows selected

Order by name

SQL> select name from student order by name

NAME

Ahok

Dharani

Dinesh

Karthi

Manoj

Vijay

Vivek

7 rows selected

Order the name by descending order

*SQL> select * from student order by name desc;*

NAME	ID	AGE	DEPT
Vivek	45455	20	cse
Vijay	2345	18	pt
Manoj	1234	19	it
Karthi	5678	19	at
Dinesh	3456	21	pt
Dharani	541014	21	it
Ahok	3456	19	bt

7 rows selected

Group by

SQL> select dept,count(id) "total student" from student group bt dept

DEPT	total student
At	1
Bt	1
Cse	1
It	2
Pt	2

SQL> select dept, count(id), 'total student' from student group by dept having dept='it' or dept='pt'

DEPT	Total student
It	2
Pt	2

LAB 5:

NESTED QUERIES

ACCOUNT TABLE

*SQL> select * from account*

NAME	ACCOUNTNO	BALANCE
Ragav	1010	1000
Santha	2020	2000
Kishor	3030	3000
Ramu	4040	4000
Ratha	5050	5000
Santhia	6060	6000

6 rows selected

SQL> select name from account where balance=(select min(balance) from account);

NAME

Ragav

SQL> select name from account where balance=(select max(balance0from account);

NAME

Santha

SQL> select name from account where balance=(select min(balance0 from account where balance>(select min(balance) from account));

NAME

Santha

SQL> Select name from account where balance=(select min(balance) from account where min(balance0 from account));

NAME

Santha

BORROWER TABLE

*SQL> select * from borrow*

NAME	LOANNO
Krishna	1111
Parthiban	2222
Sathish	3333
Vijayaraja	4444
Abdul	5555
Karthickraja	6666

SQL> create table deposit (name varchar2(20), accountno number(5));

Table created

DEPOSITER TABLE

*SQL> select * from deposit*

NAME	ACCOUNTNO
Parthiban	1100
Krishna	6600
Vijayaraja	1108
Pooja	6608
Raja	7708
Suresh	8808

6 rows selected

SQL> select distinct name deposit where name not in (select name from borrow);

NAME

Pooja

Raja

Suresh

SQL> select distinct name from borrow where name in (select name from deposit)

NAME

Krishna

Parthiban

Vijayaraja

LAB 6

SQL BUILT IN FUNCTIONS

- STRING FUNCTIONS
- MATHEMATICAL FUNCTIONS
- DATE FUNCTIONS
- CONVERSION FUNCTIONS
- GROUP FUNCTIONS
- MISCELLANEOUS FUNCTIONS

MATHEMATICAL FUNCTIONS

ABSOLUTE
SQL> select abs(-33) absolute from dual
33
CELL
SQL>select cell984.345)from dual;
85
SQUARE ROOT
SQL>select sqrt(169) from dual
13
FLOOR
SQL>select floor(23.7) from dual
23
POWER
SQL> select power(2.3) from dual
8
MODULE
SQL> select mod(235,8) from dual
3
EXPONENT
SQL> select exp(5) from dual
148413159

CHARACTER FUNCTIONS

INITCAP
SQL> select initcap('computer') from dual
Computer
UPPER
SQL> select upper('computer'0 from dual
COMPUTER
LOWER
SQL> select lower('DINESH') FROM DUAL
dinesh

SUBSTR()

SQL> select substr('hardware',2,4) from dual

ardv

LENGTH

Sql> select length('hardware') from dual

8

REPLACE

SQL> select replace('mouse','o','y') from dual

Myuse

DATE FUNCTIONS

LASTDAY

SQL> select last_day('3-jan-07') from dual

31-jan-07

NEXT DAY

SQL> select next_day('5-feb-07','monday') from dual

12-feb-07

LAB 7

VIEWS

*SQL> select * from stu*

REGNO	NAME	DEPT
1007	ragava	it
1008	ramu	it
1009	krishna	it
2345	arun	at
6789	parthi	cs
1011	dhoni	eee

6 rows selected

*SQL> create view information as select * from stu where dept='it';*

View created

*SQL> select * from information*

REGNO	NAME	DEPT
1007	ragava	it
1008	ramu	it
1009	krishna	it

SQL> insert into information values(1004,'dhoni', 'it');

1 row created

*SQL> select * from information*

REGNO NAME DEPT

1007	ragava	it
1008	ramu	it
1009	krishna	it
1004	dhoni	it

UPDATE

SQL> update information set regno=1001 where dept='it';

4 rows updated

*SQL> select * from information*

REGNO	NAME	DEPT
1001	ragava	it
1001	ramu	it
1001	krishna	it
1001	dhoni	it

LAB 8

OLYMPIC GAME DATABASE

1.Creating an Olympic database with needed attributes

SQL> create oly_game(comp varchar2(10),comp_id , number,comp_name varchar2(10),country varchar2(10), medal varchar2(3), location varchar2(10),comp_date date, time varchar2(10));

Table created

*SQL> select * from oly_game*

COMP	COMP_ ID	COMP_NAME	COUNTRY	MED	LOCATION	COMP_D ATE	TIME
Tennis	1	Sania	India	Gol	Nehru	15-jan-07	15.00
Tennis	2	Roaddick	Spain	sill	Nehru	15-jan-07	15.00
Tennis	3	John	USA	brol	Nehru	15-jan-07	15.00
Tennis	4	philip	Italy	Nil	Nehru	15-jan-07	15.00
Tennis	5	rosy	germany	nil	Nehru	15-jan-07	15.00
100 m run	6	Shanthi	India	Nil	Anna	20-jan-07	13:30
100 m run	4	philip	Italy	goll	Anna	20-jan-07	13:30
100 m run	7	Umar	Spain	Sil	Anna	20-jan-07	13:30
100 m run	8	Shawn	USA	bro	Anna	20-jan-07	13:30
100 m run	9	Peter	germany	Nil	Anna	20-jan-07	13:30
Shooting	10	Jack	USA	Gol	Rajiv	19-jan-07	05:00
Shooting	11	George	Spain	Sil	Rajiv	19-jan-07	05:00
Shooting	5	Rosy	Germany	Bro	Rajiv	19-jan-07	05:00

Shooting	12	Sanjiv	India	Nil	Rajiv	19-jan-07	05:00
Shooting	13	nancy	italy	Nill	Rajiv	19-jan-07	05:00

2. List all the sports men from India

*SQL> select * from oly_game where country=india*

COMP	COMP_ ID	COMP_NAME	COUNTRY	MED	LOCATION	COMP_D ATE	TIME
Tennis	1	Sania	India	Gol	Nehru	15-jan-07	15.00
100 m run	6	Shanthi	India	Nil	Anna	20-jan-07	13:30
Shooting	12	Sanjiv	India	Nil	Rajiv	19-jan-07	05:00

3. Making a copy of that table

SQL> create table duplicate(comp varchar2(10),comp_id number, comp_name varchar2(10),country varchar2(10),medal varchar2(30),location varchar2(10),comp_date date,time varchar2(10);

Table created

*SQL> insert into duplicate(select * from oly_game);*

15 rows created

*SQL> select * from duplicate*

COMP	COMP _ID	COMP_NAME	COUNTRY	ME D	LOCATIO N	COMP_D ATE	TIME
Tennis	1	Sania	India	Gol	Nehru	15-jan-07	15.00
Tennis	2	Roaddick	Spain	sill	Nehru	15-jan-07	15.00
Tennis	3	John	USA	brol	Nehru	15-jan-07	15.00

Tennis	4	Philip	Italy	Nil	Nehru	15-jan-07	15.00
Tennis	5	Rosy	germany	nil	Nehru	15-jan-07	15.00
100 m run	6	Shanthi	India	Nil	Anna	20-jan-07	13:30
100 m run	4	philip	Italy	goll	Anna	20-jan-07	13:30
100 m run	7	Umar	Spain	Sil	Anna	20-jan-07	13:30
100 m run	8	Shawn	USA	bro	Anna	20-jan-07	13:30
100 m run	9	Peter	germany	Nil	Anna	20-jan-07	13:30
Shooting	10	Jack	USA	Gol	Rajiv	19-jan-07	05:00
Shooting	11	George	Spain	Sil	Rajiv	19-jan-07	05:00
Shooting	5	Rosy	Germany	Bro	Rajiv	19-jan-07	05:00
Shooting	12	Sanjiv	India	Nil	Rajiv	19-jan-07	05:00
Shooting	13	nancy	italy	Nill	Rajiv	19-jan-07	05:00

4. Altering the table duplicate

SQL> alter table duplicate add (r varchar2(1),s varchar2(1));

Table altered

*SQL> select * from duplicate;*

COMP	COMP_ID	COMP_NAME	COUNTRY	MED	LOCATION	COMP_DATE	TIME	R	S
Tennis	1	Sania	India	Gol	Nehru	15-jan-07	15.00	N	F
Tennis	2	Roaddick	Spain	sill	Nehru	15-jan-07	15.00	N	M
Tennis	3	John	USA	brol	Nehru	15-jan-07	15.00	N	M
Tennis	4	philip	Italy	Nil	Nehru	15-jan-07	15.00	Y	M
Tennis	5	rosy	germany	nil	Nehru	15-jan-07	15.00	Y	F
100 m run	6	Shanthi	India	Nil	Anna	20-jan-07	13:30	Y	F

100 m run	4	philip	Italy	goll	Anna	20-jan-07	13:30	N	M
100 m run	7	Umar	Spain	Sil	Anna	20-jan-07	13:30	N	M
100 m run	8	Shawn	USA	bro	Anna	20-jan-07	13:30	N	M
100 m run	9	Peter	germany	Nil	Anna	20-jan-07	13:30	Y	M
Shooting	10	Jack	USA	Gol	Rajiv	19-jan-07	05:00	N	M
Shooting	11	George	Spain	Sil	Rajiv	19-jan-07	05:00	N	M
Shooting	5	Rosy	Germany	Bro	Rajiv	19-jan-07	05:00	N	F
Shooting	12	Sanjiv	India	Nil	Rajiv	19-jan-07	05:00	Y	M
Shooting	13	nancy	italy	Nill	Rajiv	19-jan-07	05:00	y	F

5. Displaying the players having remarks='yes'

*SQL> select * from duplicate where r='y';*

COMP	COMP_ID	COMP_NAME	COUNTRY	MED	LOCATION	COMP_DATE	TIME	R	S
Tennis	4	philip	Italy	Nil	Nehru	15-jan-07	15.00	Y	M

Tennis	5	rosy	germany	nil	Nehru	15-jan-07	15.00	Y	F
100 m run	6	Shanthi	India	Nil	Anna	20-jan-07	13:30	Y	F
100 m run	9	Peter	germany	Nil	Anna	20-jan-07	13:30	Y	M
Shooting	13	nancy	italy	Nill	Rajiv	19-jan-07	05:00	y	F

6. Creating a table for players with remarks =’yes’

SQL> create table remar(comp varchar2910),comp_id number,comp_name varchar2(10),country varchar2(8),medal varchar2(3), location varchar2(80, comp_date, time varchar2(10),r varchar2910,s varchar291));

Table created

*SQL> insert into remar(select * from duplicate where r=’y’);*

12 rows created

*SQL> select * from remar;*

COMP	COMP_ID	COMP_NAME	COUNTRY	MED	LOCATION	COMP_DATE	TIME	R	S
Tennis	4	Philip	Italy	Nil	Nehru	15-jan-07	15.00	Y	M
Tennis	5	Rosy	germany	nil	Nehru	15-jan-07	15.00	Y	F
100 m run	6	Shanthi	India	Nil	Anna	20-jan-07	13:30	Y	F
100 m run	9	Peter	germany	Nil	Anna	20-jan-07	13:30	Y	M
Shooti	13	Nancy	italy	Nill	Rajiv	19-jan-07	05:00	y	F

ng									

SQL> alter table duplicate drop column r;

Table altered

*SQL> select * from duplicate*

COMP	COMP_ID	COMP_NAME	COUNTRY	MED	LOCATION	COMP_DATE	TIME	S
Tennis	1	Sania	India	Gol	Nehru	15-jan-07	15.00	F
Tennis	2	Roaddick	Spain	sill	Nehru	15-jan-07	15.00	M
Tennis	3	John	USA	brol	Nehru	15-jan-07	15.00	M
Tennis	4	philip	Italy	Nil	Nehru	15-jan-07	15.00	M
Tennis	5	rosy	Germany	nil	Nehru	15-jan-07	15.00	F
100 m run	6	Shanthi	India	Nil	Anna	20-jan-07	13:30	F
100 m run	4	philip	Italy	Gol	Anna	20-jan-07	13:30	M
100 m run	7	Umar	Spain	Sil	Anna	20-jan-07	13:30	M
100 m run	8	Shawn	USA	Bro	Anna	20-jan-07	13:30	M
100 m run	9	Peter	Germany	Nil	Anna	20-jan-07	13:30	M
Shooti	10	Jack	USA	Gol	Rajiv	19-jan-07	05:00	M

ng								
Shooting	11	George	Spain	Sil	Rajiv	19-jan-07	05:00	M
Shooting	5	Rosy	Germany	Bro	Rajiv	19-jan-07	05:00	F
Shooting	12	Sanjiv	India	Nil	Rajiv	19-jan-07	05:00	M
Shooting	13	nancy	Italy	Nill	Rajiv	19-jan-07	05:00	F

7. Creating a view for each competition

*SQL> create view tennis_game as select * from duplicate where comp='tennis'*;

View created

*SQL> select * from tennis_game*

COMP	COMP_ID	COMP_NAME	COUNTRY	MED	LOCATION	COMP_DATE	TIME	S
Tennis	1	Sania	India	Gol	Nehru	15-jan-07	15.00	F
Tennis	2	Roaddick	Spain	sill	Nehru	15-jan-07	15.00	M
Tennis	3	John	USA	brol	Nehru	15-jan-07	15.00	M
Tennis	4	philip	Italy	Nil	Nehru	15-jan-07	15.00	M
Tennis	5	rosy	germany	nil	Nehru	15-jan-07	15.00	F

*SQL> create view run_games as select * from duplicate where comp='100m run';*

View created

*SQL> select * from run_games*

COMP	COMP_ID	COMP_ NAME	COUNTRY	MED	LOCATION	COMP_ DATE	TIME	S
100 m run	6	Shanthi	India	Nil	Anna	20-jan-07	13:30	F
100 m run	4	philip	Italy	goll	Anna	20-jan-07	13:30	M
100 m run	7	Umar	Spain	Sil	Anna	20-jan-07	13:30	M
100 m run	8	Shawn	USA	bro	Anna	20-jan-07	13:30	M
100 m run	9	Peter	germany	Nil	Anna	20-jan-07	13:30	M

*SQL> create view shoot_game as select * from duplicate where comp='shooting';*

View created

*SQL> select * from shoot_game*

COMP	COMP_ID	COMP_ NAME	COUNTRY	MED	LOCATION	COMP_ DATE	TIME	S
Shooting	10	Jack	USA	Gol	Rajiv	19-jan-07	05:00	M
Shooting	11	George	Spain	Sil	Rajiv	19-jan-07	05:00	M
Shooting	5	Rosy	Germany	Bro	Rajiv	19-jan-07	05:00	F

Shooting	12	Sanjiv	India	Nil	Rajiv	19-jan-07	05:00	M
Shooting	13	nancy	italy	Nill	Rajiv	19-jan-07	05:00	F

8. Sort the country by the number of medals they got

SQL> select country, count(medal)"no of medal" from duplicate group by country

COUNRY	no of medal
USA	3
Spain	3
India	1
Italy	1
Germany	1

9. Select country having the maximum number of medals

*SQL> select * from maxcountry where medal=(select max(medal) from maxcountry);*

COUNTRY	MEDAL
USA	3
Spain	3

.

LAB 9:

FUNCTIONS

SYNTAX

Create [or replace] function function name[(argument1,argument2,....., argument n)]

Return function- datatype is

[local- variable-declarations]

Begin

Executable-section

[exception-section]

Return-function value

End[function-name];

Exercise

Create a function for withdrawing money from an account in a bank management system which uses bank table

SQL> create or replace function withdraw (n in number, amt in number)

2 return number is

3 b number

4 begin

5 select balance into b from bank where acc-no=n;

6 if b-500>amt then

7 b:=b-amt

8 update bank set balance=b where acc-no=n;

```
9 else
10 dbms_output.put_line('can not withdraw');
11 endif
12 return b;
13 End;
14 /
```

Function created

*SQL> select * from bank*

ACC-NO	B_NAME	BALANCE
101	SBI	25000
102	SBT	5000
103	FEDERAL	10000
104	AXIS	15000
105	CANARA	50000

Calling function from a PL\SQL block:

```
SQL> declare
2 n number
3 begin
4 n:=withdraw(101,20000);
5 end
6 /
```

PL/SQL procedure successfully completed.

*SQL> select *from bank;*

ACC-NO	B_NAME	BALANCE
101	SBI	5000
102	SBT	5000
103	FEDERAL	10000
104	AXIS	15000
105	CANARA	50000

Create a function for depositing money to an account in a bank management system which uses bank table

SQL> create or replace function deposit(n in number, amt in number)

2 return number is

3 b number

4 Begin

5 select balance into b from bank where acc_no=n;

6 b:=b+amt;

7 update bank set balance=b where acc_no=n;

8 return b;

9 end

10 /

Function created

Calling function from a PL/SQL block

SQL> declare

2 n number

3 begin

4 n:=deposit(104,5000);

5 end

6 /

PL/SQL procedure successfully completed.

*SQL> select * from bank*

ACC-NO	B_NAME	BALANCE
101	SBI	5000
102	SBT	5000
103	FEDERAL	10000
104	AXIS	20000
105	CANARA	50000

Function with implicit Cursors:

Create a function for calculating the total marks and percentage of the student in a student management system which uses a student table:

SQL> create or replace function stud_update

2 return number is

3 cursor c is select * from student;

4 ctot number

5 cper number

6 begin

7 for i in c

8 loop

9 ctot:=i.m1+i.m2+i.m3;

```
10 cper:=ctot/3;
11 update student set total=ctot where id_no=i.id_no;
12 update student set per=cper where id_no=id_no;
13 end loop;
14 Return 1;
15 End;
16 /
Function created
```

*SQL> select * from student;*

ID_NO	NAME	GR	DE	M1	M2	M3	TOTAL	PER
10	Gouri	a	it	89	56	74	0	0
11	Akashaj	s	it	95	91	93	0	0
23	Lithik	b	bt	78	67	71	0	0
27	Pallavi	s	bt	90	98	96	0	0
30	Navaj	a	ph	88	81	89	0	0

```
SQL> declare
2 n number
3 begin
4 n:=stud_update;
5 end;
```

6 /

PL/sql Procedure successfully completed

*SQL> select * from student*

ID_NO	NAME	GR	DE	M1	M2	M3	TOTAL	PER
10	Gouri	a	it	89	56	74	219	73
11	Akashaj	s	it	95	91	93	279	93
23	Lithik	b	bt	78	67	71	216	72
27	Pallavi	s	bt	90	98	96	284	94.6667
30	Navaj	a	ph	88	81	89	258	86

Function with explicit Cursors:

Creating a function for calculating net salary for all employees in an organization using an employee table

SQL> create or replace function salary

2 return number is

3 cursor c is select * from employee;

4 i employee%rowtype;

5 netsalary number;

6 begin

7 open c;

8 loop

9 fetch c into i;

10 If c%notfound then exit

11 end if

```
12 netsalary=i.basic+i.hra+i.da-i.pf;
13 update employee set netsal=netsalary where e_no=i.e_no;
14 end loop;
15 return netsalary;
16 close c;
17 end
18 /
```

*SQL> select * from employee*

E_NO	E_NAME	HRA	DA	PF	BASIC	NETSAL
100	Adithya	500	200	350	10000	0
101	Anusha	250	300	410	12000	0
102	Sara	250	300	100	20000	0
103	Ragul	295	600	480	8500	0
104	Pooja	100	100	200	7500	0

Function created

SQL> declare

```
2 n number
3 begin
4 n:=salary;
5 end
6 /
```

PL/SQL procedure successfully completed

*SQL> select * from employee*

E_NO	E_NAME	HRA	DA	PF	BASIC	NETSAL
100	Adithya	500	200	350	10000	10350
101	Anusha	250	300	410	12000	12140
102	Sara	250	300	100	20000	20450
103	Ragul	295	600	480	8500	8915
104	Pooja	100	100	200	7500	7500

LAB 10:

PROCEDURES

SYNTAX

SQL> create[or replace] procedure procedure_name

[(argument1, argument2,…(argument n)] is

[local –variable - declarations]

begin

executable-section

[exception- section]

End[procedure-name];

SQL> create table bank(acc_no number, br_name varchar2(10), bal number);

Table created.

1. Create a procedure for deposit and withdrawal of money in an account in a Bank Management System which uses bank table.

SQL> create or replace procedure bank_up(opt_number, amount number, n number) from bank where accno= n;

2 as

3 balance number

```
4     begin
5     select bal into balance from bank where accno = n;
6     if opt=1 then
7     balance:= balance + amount;
8     update bank set bal = balance where accno= n;
9     commit;
10    dbms_output.put_line('balance after deposition is ' || balance);
11    elsif opt=2 then
12    balance:= balance-amount;
13    if balance <1000 then
14    dbms_output.put_line('cannot withdraw… balance low…!('|| balance||')');
15    else
16    update bank set bal = balance where accno= n;
17    commit; dbms_output.put_line ('balance after withdrawal is '||balance);
18    end if;
19    end if;
20    end;
21    /
Procedure created.
SQL> select * from bank;
ACC_NO    NAME          BAL
100       Anil          50000
101       Abi           10000
102       Bavi          2500
```

103 Chandru 1000

104 Divakar 20000

SQL>exec bank_up(1, 40000, 100);

balance after deposition is 80900

PL/ SQL procedure successfully completed.

*SQL> select * from bank;*

ACC_NO	NAME	BAL
100	Anil	90000
101	Abi	10000
102	Bavi	2500
103	Chandru	1000
104	Divakar	20000

SQL >exec bank_up(2,1500,102);

Cannot withdraw… balance low..!(500)

PL/ SQL procedure successfully completed.

*SQL> select * from bank;*

ACC_NO	NAME	BAL
100	Anil	90000
101	Abi	10000
102	Bavi	500
103	Chandru	1000

104 Divakar 20000

PROCEDURES WITH IMPLICIT CURSORS:

SQL> create table emp1(empno number, empname varcgar2(10), deptno number, sal number exp number);

Table created.

*SQL> select * from emp1;*

EMPNO	EMPNAME	DEPTNO	SAL	EXP
1000	Akshay	250	22000	3
1010	Akshay	230	18000	2
1012	Abhirami	250	20000	2
1015	Bavi	240	34000	6
1200	Akshay	250	32000	4
1017	Lakshmi	200	28000	3
1019	Sundar	200	30000	3
1045	Sreeram	260	42000	4

7 rows selected.

2. Create a procedure for incrementing rupees 1500 for those with experience less than 4 years and 4000 for rest of all employees in an organization which uses employee table.

```
SQL > create or replace procedure emp1_bonus
2       is
3       cursor c is select * from emp1;
4       bsal number;
5       begin
6       for I in c
7       loop
8       if i.exp<4 then
9       bsal:=sal+1500;
10      update emp1 set sal = bsal where empno =i.empno;
11      else
12      bsal:=i.sal+5000;
13      update emp1 set sal = bsal where empno =i.empno;
14      end if;
15      end loop;
16      end;
17      /
Procedure created.
```

PL/ SQL procedure successfully completed.

*SQL> select * from emp1;*

EMPNO	EMPNAME	DEPTNO	SAL	EXP
1000	Akshay	250	23500	3
1010	Akshay	230	19500	2
1012	Abhirami	250	21500	2
1015	Bavi	240	39000	6
1200	Akshay	250	37000	4
1017	Lakshmi	200	29500	3
1019	Sundar	200	31500	3
1045	Sreeram	260	47000	4

8 rows selected.

PROCEDURES WITH EXPLICIT CURSORS:

SQL> create table invent1(itcode number, itname varchar2(10), no_of_indiv number);

Table created.

*SQL> select * from invent1;*

ITCODE	ITNAME	NO_OF_INDIV
1001	FLOPPY	10
1067	DVD ROM	100
500	CD ROM	500
1000	PEN DRIVE	5
249	ZIP DRIVE	3

3. Create a procedure for checking the stock of an agency to order items which uses inventory and orderit tables.

SQL > create or replace procedure invent_check

```
2       is
3       cursor c select * from invent1;
4       j invent% rowtype;
5       che number;
6       begin
7       open c
8       loop
9       fetch c into j;
10      if c%notfoun then exit;
11      end if;
12      che:= j.no_of_indiv;
13      if che<=5 then
```

```
14    insert into orderit values(j.itname, j.no_of_indiv);
15    end if;
16    end loop;
17    close c;
18    end;
19    /
```

Procedure created.

SQL> exec invent_check

PL/ SQL procedure successfully completed.

*SQL> select * from orderit;*

ITNAME	NO_OF_INDIV
PEN DRIVE	5
ZIP DRIVE	3

LAB 11

IMPLICIT CURSORS

SYNTAX:

Exercise:

Write a PL/ SQL code for calculating total mark, percentage, grade for all the students in a student management system using implicit cursors.

SQL> create table student(id number, name varchar2(10), dept varchar2(10), percent number,m1 number,m2 number, m3 number, tot number, g varchar2(1));

Table created.

*SQL> select * from student;*

ID	NAME	DEP	PERCENT	M1	M2	M3	TOT	G
1	Anu	it	0	90	89	80	0	
2	Beena	cse	0	98	91	95	0	
3	Bindhu	it	0	87	67	86	0	
4	Varun	it	0	67	46	50	0	
5	Rahul	cse	0	81	82	83	0	

SQL> declare

2 cursor c is select * from student;

3 ctot number;

4 cgra varchar2(1);

```
5     cper number;
6     begin
7     for I in c
8     loop
9     ctot= i.m1+i.m2+i.m3;
10    cper :=ctot/3;
11    update student set tot = ctot where id =i.id;
12    update student set percent = cper where id =i.id;
13    if(cper between 91 and 100)then
14    cgra:= 'S'
15    elsif(cper between 81 and 90)then
16    cgra:= 'A'
17    elsif(cper between 71 and 80)then
18    cgra:= 'B'
19    elsif(cper between 61 and 70)then
20    cgra:= 'C'
21    elsif(cper between 56 and 60)then
22    cgra:= 'D'
23    elsif(cper between 50 and 55)then
24    cgra:= 'E'
25    else
26    cgra:= 'F'
27    end if;
28    update student set g = cgra where id =i.id;
```

```
29    end loop;
30    end;
31    /
```

PL/ SQL procedure successfully completed.

*SQL> select * from student;*

ID	NAME	DEP	PERCENT	M1	M2	M3	TOT	G
1	Anu	it	86.3333333	90	89	80	259	A
2	Beena	cse	94.6666667	98	91	95	284	S
3	Bindhu	it	80	87	67	86	240	B
4	Varun	it	54.3333333	67	46	50	163	E
5	Rahul	cse	82	81	82	83	246	A

LAB 12:

EXPLICIT CURSORS

SYNTAX:

cursor cursor_name is select * from table name;

To open the cursor:

open cursor_name;

To close the cursor:

close cursor_name;

Exercise:

Write PL/ SQL code for calculating hra , da, netsalary for all the employees in the Payroll Processing using Explicit cursor(uses employee table).

*SQL> select * from employee;*

EMPNO	NAME	HRA	DA	PF	NETSAL	BASICPAY
101	AAA	0	0	0	0	15000
102	BBB	0	0	0	0	18000
103	CCC	0	0	0	0	20000
104	DDD	0	0	0	0	10000
105	EEE	0	0	0	0	25000

```
SQL> declare
2       cursor c is select * from employee;
3       i employee% rowtype;
4       hrasal number;
5       dasal number;
6       pfsal number;
7       netsalary number;
8       begin
9       open c;
10      loop;
11      fetch c into i;
12      if c% notfound ten exit;
13      endif;
14      hrasal:=i.basicpay*0.1;
15      dasal:=i.basicpay*0.08;
16      pfsal:=i.basicpay*0.12;
17      netsalaray:= i.basicpay + hrasal + dasal + pfsal;
18      update employee set hra = hrasal, da= dasal, pf= pfsal, netsal= netsalaray
where empno=i.empno;
19      end loop;
20      close c;
21      end;
22      /
PL/ SQL procedure successfully completed.
```

*SQL> select * from employee;*

EMPNO	NAME	HRA	DA	PF	NETSAL	BASICPAY
101	AAA	1500	1200	1800	15900	15000
102	BBB	1800	1440	2160	19080	18000
103	CCC	2000	1600	2400	21200	20000
104	DDD	1000	800	1200	10600	10000
105	EEE	2500	2000	3000	26500	25000

LAB 13:

TRIGGERS

SYNTAX

Create or replace trigger {schema]trigger name

{BEFORE, AFTER}

{DELETE, INSERT, UPDATE [OF column.....]}

ON [schema] table name

[REFERENCING {OLD AS old, NEW AS new}]

{FOR EACH ROW[WHEN condition]}

DECLARE

    Variable declaration;

    Constant declaration;

BEGIN

If inserting then

<PL/SQL block>

end if;

If updating then

<PL/SQL block>

end if;

if deleting then

<PL/SQL block>

end if;

EXCEPTION

Exception<PL/SQL block>

End;

1. Create a trigger for invent table which triggers while updating operations are performed on the invent table

SQL> create or replace triggers inve before update on invent

2 for each row

3 declare

4 begin

5 if: new.no_of_indiv<=10 then

6 Insert into orderit values(:new.itname,:new.no_of_indiv);

7 else

8 delete from orderit where itname=:new.itname;

9 end if;

10 end;

11 /

Trigger created

SQL>select from orderit;*

ITNAME	NO_OF_INDIV
hard disk	3
cd rom	5

ipod 2

2. Create a trigger while insert or update or delete operations are performed on the table employ.

*SQL> select * from employ*

No rows selected

*SQL> select * from emp_log;*

No rows selected

SQL> create or replace trigger LOG_EMP

2 after insert or update or delete on employ

3 begin

4 if inserting then

5 insert into EMP_LOG values(user,'INSERT',sysdate);

6 end if;

7 if updating then

8 insert into EMP_LOG values(user,'UPDATE',sysdate);

9 end if;

10 if deleting then

11 insert into EMO_LOG values(user,'DELETE',sysdate);

12 end if;

13 end;

14/

Trigger created

SQL>insert into employ values('sachin',101);

1 row created

*SQL>select *from emp_log;*

USEDBY	OPERATION	SYSDATE
SCOTT	INSERT	25-MAR-13

SQL>update employ set id=103 where id=101;

1 row updated.

*SQL>select * from emp_log;*

USEDBY	OPERATION	SYSDATE
SCOTT	INSERT	25-MAR-13
SCOTT	UPDATE	25-MAR-13

SQL> delete from employ where id=103;

1 row deleted

*SQL>select *from emp_log;*

USEDBY	OPERATION	SYSDATE
SCOTT	INSERT	25-MAR-13
SCOTT	UPDATE	25-MAR-13
SCOTT	DELETE	25-MAR-13

LAB 14:

OBJECTS

SYNTAX:

CREATE OR REPLACE TYPE typename AS OBJECT

(<column1><datatype>,.....<column n><datatype>);

Exercise:

SQL> create or replace type stu_addr as object(streat varchar2(10), city varchar2(10),state varchar2(20),pincode number);

Type created.

Using objects to create table:

Syntax:

Create Table tablename(<column 1<datatype>.....<column n><object name>);

Exercise:

SQL>create table stu(name varchar2(10),address stu_add);

Table created.

Inserting values into table:

Syntax:

Insert into tablename values (column 1 value, column2 value,.....objectname(column1 value, column2 value,.....column n value),.....column n value);

Example:

SQL> insert into stu
values('AAA',stu_addr('Donaldst','mankara',kerala',620024));

1 row created

SQL>insert into stu values('BBB',stu_addr('
Gandgist','chennai',tamilnadu',600087));

1 row created

*SQL> select * from stu;*

NAME

ADDRESS(STREAT,CITY,STATE,PINCODE)

AAA

STU_ADDR('Donald st','mankara',620024)

BBB

STU_ADDR('Gandhi st','chennai',Tamilnadu',600087)

LAB 15:

EXCEPTION HANDLING

SYNTAX:

DECLARE

<declaration section>

BEGIN

<executable commands>

EXECPTION

<exception handling>

End:

1. Create an exception if balance of an account number is less than 500.

*SQL>select * from bank;*

ACCNO	DNAME	BAL
101	trichy	37000
102	Deini	450
103	Mumbai	6000
104	chennai	2000

SQL>set serveroutput on;

SQL>declare

```
2 b number
3 n number
4 balanlow exception
5 balanok exception
6 begin
7 n:=&n;
8 select bal into b from bank where accno=n;
9 if (b<500)then
10 raise balanlow;
11 else
12 raise balanok;
13 end if;
14 exception
15 when balanlow then
16 dbms_output.put_line('YOUR balace is low'||b);
17 when balanok then
18 Dbms_output.put_line(YOUR balance is'||b);
19 end;
/

Enter value for n:102
Old 7: n:=&n;
New 7: n:=102;
YOUR balance is low 450
```

PL/SQL procedure successfully completed

SQL>

Enter value for n:103

Old 7: n:=&n;

New 7: n:=103;

YOUR balance is low 6000

PL/SQL procedure successfully completed

LAB 16:

PACKAGES

SYNTAX:

Creating Package:

Create [or replace] package packagename

{Is/as}

PL/SQL package specification;

Creating package body:

Create [or replace] package body packagename

{is/as}

PL/SQL package body;

Exercise:

Create a package for bank management system with Enquire, Deposit, withdraw options

Creating package:

SQL>create or replace package banking

2 is

3 procedure enquire(n number);

4 function withdraw(n number,amount number)return number;

5 function deposit (n number,amount number)return number;

6 end banking;

7 /

Package created.

Before ececution:

*SQL>select *from bank;*

ACCNO	BNAME	BAL
101	trichy	37000
102	Deini	450
103	Mumbai	6000
104	chennai	2000

Creating package body:

SQL>create or replace package body Banking

```
2 is
3 procedure enquire(n in number) is
4 s number;
5 begin
6 select bal into s from bank where accno=n;
7 dbms_output.put_line('***Baqlance for given account number
'||n||'is'||s||'***');
8 end;
9 function withdraw(n in number,amount in number)return number
10 is
11 s number;
12 begin
13 select bal into a from bank where accno=n;
14 if s-500>amount then
```

```
15 s:s-amount;
 16 update bank set bal=s where accno=n;
17 dbms_output.put_line('Your balance after withdrawn'||s);
18 else
19 dbms_output.put_line(YOU cannt withdraw!!!');
20 dbms_output.put_line('*** Balance for given account
number'||n||'is'||s||'***');
21 end if;
22 return s;
23 end;
24 function deposit( n number, amount number)return number
25 is
26 s number
27 begin
28 select bal into a from bank where accno=n;
29 s:=s+amount;
30 update bank set bal=s where accno=n;
31 dbme_output.put_line('Your balance after deposit '||s);
32 return s;
33 end;
34 end Banking;
35/
Package body created.
Executing package (calling deposit function);
SQL>declare
```

2 p number;

3 begin

4 p:=banking.deposit(104,1000);

5 end;

6 /

Your balance after deposit 3000

PL/SQL procedure successfully completed

After execution:

*SQL>select *from bank;*

ACCNO	BNAME	BAL
101	trichy	37000
102	Deini	450
103	Mumbai	6000
104	chennai	3000

Executing packages(Calling enquire procedure);

SQL>declare

2 p number;

3 begin

4 p:=banking.enquire(103);

5 end;

6 /

*** Balance for given account number 103 is 6000***

PL/SQL procedure successfully completed

*SQL>select *from bank;*

ACCNO	BNAME	BAL
101	trichy	37000
102	Deini	450
103	Mumbai	6000
104	chennai	3000

Executing packages(Calling withdraw function);

SQL>declare

2 p number;

3 begin

4 p:=banking. Withdraw (101,2000);

5 end;

6 /

Your balance after withdrawn 35000

PL/SQL procedure successfully completed

After execution:

*SQL>select *from bank;*

ACCNO	BNAME	BAL
101	trichy	35000
102	Deini	450
103	Mumbai	6000
104	chennai	3000

Expected Viva Questions

1. What is database?

A database is a logically coherent collection of data with some inherent meaning, representing some aspect of real world and which is designed, built and populated with data for a specific purpose.

2. What is DBMS?

It is a collection of programs that enables user to create and maintain a database. In other words it is general-purpose software that provides the users with the processes of defining, constructing and manipulating the database for various applications.

3. What is a Database system?

The database and DBMS software together is called as Database system.

4. What are the advantages of DBMS?

1. Redundancy is controlled.
2. Unauthorised access is restricted.
3. Providing multiple user interfaces.
4. Enforcing integrity constraints.
5. Providing backup and recovery.

5. What are the disadvantage in File Processing System?

1. Data redundancy and inconsistency.
2. Difficult in accessing data.

3. Data isolation.
4. Data integrity.
5. Concurrent access is not possible.
6. Security Problems.

6. Describe the three levels of data abstraction?

The are three levels of abstraction:

1. **Physical level:** The lowest level of abstraction describes how data are stored.
2. **Logical level:** The next higher level of abstraction, describes what data are stored in database and what relationship among those data.
3. **View level:** The highest level of abstraction describes only part of entire database.

7. Define the "integrity rules"?

There are two Integrity rules.

1. **Entity Integrity:** States that "Primary key cannot have NULL value"
2. **Referential Integrity:** States that "Foreign Key can be either a NULL value or should be Primary Key value of other relation.

8. What is extension and intension?

1. **Extension:** It is the number of tuples present in a table at any instance. This is time dependent.
2. **Intension:** It is a constant value that gives the name, structure of table and the constraints laid on it.

9. What is System R? What are its two major subsystems?

System R was designed and developed over a period of 1974-79 at IBM San Jose Research Center. It is a prototype and its purpose was to demonstrate that it is possible to build a Relational System that can be used in a real life environment to solve real life problems, with performance at least comparable to that of existing system.

Its two subsystems are

1. Research Storage
2. System Relational Data System.

10. How is the data structure of System R different from the relational structure?

Unlike Relational systems in System R

1. Domains are not supported
2. Enforcement of candidate key uniqueness is optional
3. Enforcement of entity integrity is optional
4. Referential integrity is not enforced

11. What is Data Independence?

Data independence means that "the application is independent of the storage structure and access strategy of data". In other words, The ability to modify the schema definition in one level should not affect the schema definition in the next higher level.

Two types of Data Independence:

1. **Physical Data Independence:** Modification in physical level should not affect the logical level.
2. **Logical Data Independence:** Modification in logical level should affect the view level.

NOTE: Logical Data Independence is more difficult to achieve

12. What is a view? How it is related to data independence?

A view may be thought of as a virtual table, that is, a table that does not really exist in its own right but is instead derived from one or more underlying base table. In other words, there is no stored file that direct represents the view instead a definition of view is stored in data dictionary.

Growth and restructuring of base tables is not reflected in views. Thus the view can insulate users from the effects of restructuring and growth in the database. Hence accounts for logical data independence.

13. What is Data Model?

A collection of conceptual tools for describing data, data relationships data semantics and constraints.

14. What is E-R model?

This data model is based on real world that consists of basic objects called entities and of relationship among these objects. Entities are described in a database by a set of attributes.

15. What is Object Oriented model?

This model is based on collection of objects. An object contains values stored in instance variables with in the object. An object also contains bodies of code that operate on the object. These bodies of code are called methods. Objects that contain same types of values and the same methods are grouped together into classes.

16. What is an Entity?

It is a 'thing' in the real world with an independent existence.

17. What is an Entity type?

It is a collection (set) of entities that have same attributes.

18. What is an Entity set?

It is a collection of all entities of particular entity type in the database.

19. What is an Extension of entity type?

The collections of entities of a particular entity type are grouped together into an entity set.

20. What is Weak Entity set?

An entity set may not have sufficient attributes to form a primary key, and its primary key compromises of its partial key and primary key of its parent entity, then it is said to be Weak Entity set.

21. What is an attribute?

It is a particular property, which describes the entity.

22. What is a Relation Schema and a Relation?

A relation Schema denoted by R(A1, A2, ..., An) is made up of the relation name R and the list of attributes Ai that it contains. A relation is defined as a set of tuples. Let r be the relation which contains set tuples (t1, t2, t3, ..., tn). Each tuple is an ordered list of n-values t=(v1,v2, ..., vn).

23. What is degree of a Relation?

It is the number of attribute of its relation schema.

24. What is Relationship?

It is an association among two or more entities.

25. What is Relationship set?

The collection (or set) of similar relationships.

26. What is Relationship type?

Relationship type defines a set of associations or a relationship set among a given set of entity types.

27. What is degree of Relationship type?

It is the number of entity type participating.

28. What is DDL (Data Definition Language)?

A data base schema is specifies by a set of definitions expressed by a special language called DDL.

29. What is VDL (View Definition Language)?

It specifies user views and their mappings to the conceptual schema.

30. What is SDL (Storage Definition Language)?

This language is to specify the internal schema. This language may specify the mapping between two schemas.

31. What is Data Storage - Definition Language?

The storage structures and access methods used by database system are specified by a set of definition in a special type of DDL called data storage-definition language.

32. What is DML (Data Manipulation Language)?

This language that enable user to access or manipulate data as organised by appropriate data model.

1. **Procedural DML or Low level:** DML requires a user to specify what data are needed and how to get those data.
2. **Non-Procedural DML or High level:** DML requires a user to specify what data are needed without specifying how to get those data.

33. What is DML Compiler?

It translates DML statements in a query language into low-level instruction that the query evaluation engine can understand.

34. What is Query evaluation engine?

It executes low-level instruction generated by compiler.

35. What is DDL Interpreter?

It interprets DDL statements and record them in tables containing metadata.

36. What is Record-at-a-time?

The Low level or Procedural DML can specify and retrieve each record from a set of records. This retrieve of a record is said to be Record-at-a-time.

7. What is Set-at-a-time or Set-oriented?

The High level or Non-procedural DML can specify and retrieve many records in a single DML statement. This retrieve of a record is said to be Set-at-a-time or Set-oriented.

38. What is Relational Algebra?

It is procedural query language. It consists of a set of operations that take one or two relations as input and produce a new relation.

39. What is Relational Calculus?

It is an applied predicate calculus specifically tailored for relational databases proposed by E.F. Codd. E.g. of languages based on it are DSL ALPHA, QUEL.

40. How does Tuple-oriented relational calculus differ from domain-oriented relational calculus?

1. The **tuple-oriented calculus** uses a tuple variables i.e., variable whose only permitted values are tuples of that relation. E.g. QUEL
2. The **domain-oriented calculus** has domain variables i.e., variables that range over the underlying domains instead of over relation. E.g. ILL, DEDUCE.

41. What is normalization?

It is a process of analysing the given relation schemas based on their Functional Dependencies (FDs) and primary key to achieve the properties
(1).Minimizing redundancy, (2). Minimizing insertion, deletion and update anomalies.

42. What is Functional Dependency?

A Functional dependency is denoted by X Y between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuple that can form a relation state r of R. The constraint is for any two tuples t1 and t2 in r if t1[X] = t2[X] then they have t1[Y] = t2[Y]. This means the value of X component of a tuple uniquely determines the value of component Y.

43. What is Lossless join property?

It guarantees that the spurious tuple generation does not occur with respect to relation schemas after decomposition.

44. What is 1 NF (Normal Form)?

The domain of attribute must include only atomic (simple, indivisible) values.

45. What is Fully Functional dependency?

It is based on concept of full functional dependency. A functional dependency X Y is full functional dependency if removal of any attribute A from X means that the dependency does not hold any more.

46. What is 2NF?

A relation schema R is in 2NF if it is in 1NF and every non-prime attribute A in R is fully functionally dependent on primary key.

47. What is 3NF?

A relation schema R is in 3NF if it is in 2NF and for every FD X A either of the following is true

1. X is a Super-key of R.
2. A is a prime attribute of R.

In other words, if every non prime attribute is non-transitively dependent on primary key.

48. What is BCNF (Boyce-Codd Normal Form)?

A relation schema R is in BCNF if it is in 3NF and satisfies an additional constraint that for every FD X A, X must be a candidate key.

49. What is 4NF?

A relation schema R is said to be in 4NF if for every Multivalued dependency X Y that holds over R, one of following is true.
1.) X is subset or equal to (or) XY = R.
2.) X is a super key.

50. What is 5NF?

A Relation schema R is said to be 5NF if for every join dependency {R1, R2, ..., Rn} that holds R, one the following is true 1.) Ri = R for some i.

2.) The join dependency is implied by the set of FD, over R in which the left side is key of R.

51. What is Domain-Key Normal Form?

A relation is said to be in DKNF if all constraints and dependencies that should hold on the the constraint can be enforced by simply enforcing the domain constraint and key constraint on the relation.

52. What are partial, alternate,, artificial, compound and natural key?

1. **Partial Key:** It is a set of attributes that can uniquely identify weak entities and that are related to same owner entity. It is sometime called as Discriminator.
2. **Alternate Key:** All Candidate Keys excluding the Primary Key are known as Alternate Keys.
3. **Artificial Key:** If no obvious key, either stand alone or compound is available, then the last resort is to simply create a key, by assigning a unique number to each record or occurrence. Then this is known as developing an artificial key.
4. **Compound Key:** If no single data element uniquely identifies occurrences within a construct, then combining multiple elements to create a unique identifier for the construct is known as creating a compound key.
5. **Natural Key:** When one of the data elements stored within a construct is utilized as the primary key, then it is called the natural key.

53. What is indexing and what are the different kinds of indexing?

Indexing is a technique for determining how quickly specific data can be found.
Types:

1. Binary search style indexing

2. B-Tree indexing
3. Inverted list indexing
4. Memory resident table
5. Table indexing

54. What is system catalog or catalog relation? How is better known as?

A RDBMS maintains a description of all the data that it contains, information about every relation and index that it contains. This information is stored in a collection of relations maintained by the system called metadata. It is also called data dictionary.

55. What is meant by query optimization?

The phase that identifies an efficient execution plan for evaluating a query that has the least estimated cost is referred to as query optimization.

56. What is durability in DBMS?

Once the DBMS informs the user that a transaction has successfully completed, its effects should persist even if the system crashes before all its changes are reflected on disk. This property is called durability.

57. What do you mean by atomicity and aggregation?

1. **Atomicity:** Either all actions are carried out or none are. Users should not have to worry about the effect of incomplete transactions. DBMS ensures this by undoing the actions of incomplete transactions.
2. **Aggregation:** A concept which is used to model a relationship between a collection of entities and relationships. It is used when we need to express a relationship among relationships.

58. What is a Phantom Deadlock?

In distributed deadlock detection, the delay in propagating local information might cause the deadlock detection algorithms to identify deadlocks that do not really exist. Such situations are called phantom deadlocks and they lead to unnecessary aborts.

59. What is a checkpoint and When does it occur?

A Checkpoint is like a snapshot of the DBMS state. By taking checkpoints, the DBMS can reduce the amount of work to be done during restart in the event of subsequent crashes.

60. What are the different phases of transaction?

Different phases are
1.) Analysis phase,
2.) Redo Phase,
3.) Undo phase.

61. What do you mean by flat file database?

It is a database in which there are no programs or user access languages. It has no cross-file capabilities but is user-friendly and provides user-interface management.

62. What is "transparent DBMS"?

It is one, which keeps its Physical Structure hidden from user.

63. What is a query?

A query with respect to DBMS relates to user commands that are used to interact with a data base. The query language can be classified into data definition language and data manipulation language.

64. What do you mean by Correlated subquery?

Subqueries, or nested queries, are used to bring back a set of rows to be used by the parent query. Depending on how the subquery is written, it can be executed once for the parent query or it can be executed once for each row returned by the parent query. If the subquery is executed for each row of the parent, this is called a correlated subquery.

A correlated subquery can be easily identified if it contains any references to the parent subquery columns in its WHERE clause. Columns from the subquery cannot be referenced anywhere else in the parent query. The following example demonstrates a non-correlated subquery.

Example: Select * From CUST Where '10/03/1990' IN (Select ODATE From ORDER Where CUST.CNUM = ORDER.CNUM)

65. What are the primitive operations common to all record management systems?

Addition, deletion and modification.

66. Name the buffer in which all the commands that are typed in are stored?

'Edit' Buffer.

67. What are the unary operations in Relational Algebra?

PROJECTION and SELECTION.

68. Are the resulting relations of PRODUCT and JOIN operation the same?

No.

PRODUCT: Concatenation of every row in one relation with every row in another.

JOIN: Concatenation of rows from one relation and related rows from another.

69. What is RDBMS KERNEL?

Two important pieces of RDBMS architecture are the kernel, which is the software, and the data dictionary, which consists of the system-level data structures used by the kernel to manage the database You might think of an RDBMS as an operating system (or set of subsystems), designed specifically for controlling data access; its primary functions are storing, retrieving, and securing data. An RDBMS maintains its own list of authorized users and their associated privileges; manages memory caches and paging; controls locking for concurrent resource usage; dispatches and schedules user requests; and manages space usage within its table-space structures.

70. Name the sub-systems of a RDBMS.

I/O, Security, Language Processing, Process Control, Storage Management, Logging and Recovery, Distribution Control, Transaction Control, Memory Management, Lock Management.

71. Which part of the RDBMS takes care of the data dictionary? How?

Data dictionary is a set of tables and database objects that is stored in a special area of the database and maintained exclusively by the kernel.

72. What is the job of the information stored in data-dictionary?

The information in the data dictionary validates the existence of the objects, provides access to them, and maps the actual physical storage location.

73. Which TCP/IP port does SQL Server run on? How can it be changed?

SQL Server runs on port 1433. It can be changed from the Network Utility TCP/IP properties.

74. What are the difference between clustered and a non-clustered index?

1. **A clustered index** is a special type of index that reorders the way records in the table are physically stored. Therefore table can have only one clustered index. The leaf nodes of a clustered index contain the data pages.
2. **A non clustered index** is a special type of index in which the logical order of the index does not match the physical stored order of the rows on disk. The leaf node of a non clustered index does not consist of the data pages. Instead, the leaf nodes contain index rows.

75. What are the different index configurations a table can have?

A table can have one of the following index configurations:

1. No indexes
2. A clustered index
3. A clustered index and many nonclustered indexes
4. A nonclustered index
5. Many nonclustered indexes

76. What are different types of Collation Sensitivity?

1. **Case sensitivity** - A and a, B and b, etc.

2. **Accent sensitivity**
3. **Kana Sensitivity** - When Japanese kana characters Hiragana and Katakana are treated differently, it is called Kana sensitive.
4. **Width sensitivity** - A single-byte character (half-width) and the same character represented as a double-byte character (full-width) are treated differently than it is width sensitive.

77. What is OLTP (Online Transaction Processing)?

In OLTP - online transaction processing systems relational database design use the discipline of data modeling and generally follow the Codd rules of data normalization in order to ensure absolute data integrity. Using these rules complex information is broken down into its most simple structures (a table) where all of the individual atomic level elements relate to each other and satisfy the normalization rules.

78. What's the difference between a primary key and a unique key?

Both primary key and unique key enforces uniqueness of the column on which they are defined. But by default primary key creates a clustered index on the column, where are unique creates a nonclustered index by default. Another major difference is that, primary key doesn't allow NULLs, but unique key allows one NULL only.

79. What is difference between DELETE and TRUNCATE commands?

Delete command removes the rows from a table based on the condition that we provide with a WHERE clause. Truncate will actually remove all the rows from a table and there will be no data in the table after we run the truncate command.

1. **TRUNCATE**:

1. TRUNCATE is faster and uses fewer system and transaction log resources than DELETE.
2. TRUNCATE removes the data by deallocating the data pages used to store the table's data, and only the page deallocations are recorded in the transaction log.
3. TRUNCATE removes all rows from a table, but the table structure, its columns, constraints, indexes and so on, remains. The counter used by an identity for new rows is reset to the seed for the column.
4. You cannot use TRUNCATE TABLE on a table referenced by a FOREIGN KEY constraint. Because TRUNCATE TABLE is not logged, it cannot activate a trigger.
5. TRUNCATE cannot be rolled back.
6. TRUNCATE is DDL Command.
7. TRUNCATE Resets identity of the table

2. **DELETE**:
    1. DELETE removes rows one at a time and records an entry in the transaction log for each deleted row.
    2. If you want to retain the identity counter, use DELETE instead. If you want to remove table definition and its data, use the DROP TABLE statement.
    3. DELETE Can be used with or without a WHERE clause
    4. DELETE Activates Triggers.
    5. DELETE can be rolled back.
    6. DELETE is DML Command.
    7. DELETE does not reset identity of the table.

Note: DELETE and TRUNCATE both can be rolled back when surrounded by TRANSACTION if the current session is not closed. If TRUNCATE is written in

Query Editor surrounded by TRANSACTION and if session is closed, it can not be rolled back but DELETE can be rolled back.

80. When is the use of UPDATE_STATISTICS command?

This command is basically used when a large processing of data has occurred. If a large amount of deletions any modification or Bulk Copy into the tables has occurred, it has to update the indexes to take these changes into account. UPDATE_STATISTICS updates the indexes on these tables accordingly.

81. What is the difference between a HAVING CLAUSE and a WHERE CLAUSE?

They specify a search condition for a group or an aggregate. But the difference is that HAVING can be used only with the SELECT statement. HAVING is typically used in a GROUP BY clause. When GROUP BY is not used, HAVING behaves like a WHERE clause. Having Clause is basically used only with the GROUP BY function in a query whereas WHERE Clause is applied to each row before they are part of the GROUP BY function in a query.

82. What are the properties and different Types of Sub-Queries?

1. **Properties of Sub-Query**
    1. A sub-query must be enclosed in the parenthesis.
    2. A sub-query must be put in the right hand of the comparison operator, and
    3. A sub-query cannot contain an ORDER-BY clause.
    4. A query can contain more than one sub-query.
2. **Types of Sub-Query**
    1. Single-row sub-query, where the sub-query returns only one row.

2. Multiple-row sub-query, where the sub-query returns multiple rows,. and
3. Multiple column sub-query, where the sub-query returns multiple columns

83. What is SQL Profiler?

SQL Profiler is a graphical tool that allows system administrators to monitor events in an instance of Microsoft SQL Server. You can capture and save data about each event to a file or SQL Server table to analyze later. For example, you can monitor a production environment to see which stored procedures are hampering performances by executing too slowly.

Use SQL Profiler to monitor only the events in which you are interested. If traces are becoming too large, you can filter them based on the information you want, so that only a subset of the event data is collected. Monitoring too many events adds overhead to the server and the monitoring process and can cause the trace file or trace table to grow very large, especially when the monitoring process takes place over a long period of time.

84. What are the authentication modes in SQL Server? How can it be changed?

Windows mode and Mixed Mode - SQL and Windows. To change authentication mode in SQL Server click Start, Programs, Microsoft SQL Server and click SQL Enterprise Manager to run SQL Enterprise Manager from the Microsoft SQL Server program group. Select the server then from the Tools menu select SQL Server Configuration Properties, and choose the Security page.

****