1. DATA TYPES

a. NUMBER

- It allows only numeric values
- Max size is 38 digits
- Synt: X Number (p, (s));

  P=>it allows how many digits to store

  S=>size

  Ex: X Number (5, 2)

b. CHAR

- It allows alpha numeric characters
- Max size 2000 bytes/character
- Synt: X char(s);

c. VARCHAR or VARCHAR2

- It allows alpha numeric characters
- Max size 4000 bytes/character
- Memory allocation is dynamic
- Synt: X Varchar2 (s);

d. DATE

- It is used to store date values
- Max size is 7 bytes
- Synt: X date;

e. TIMESTAMP

- It is used to store date along with fraction of seconds
- Synt: X Timestamp;

f. LONG

- It is used to store information
- Max size is 2GB
- Only once we have to use it in entire table
- Synt: X long;

g. RAW

- It is used to store images
- Max size is 2000 bytes
- Synt: X Raw;

h. LONG RAW

- It is used to store information as well as images

- Max size is 2 GB
- Synt: X Longraw;

i. LOB (Large Object Bytes) (CLOB, BLOB, BFILE & NCLOB)

i. CLOB
- It is used to store huge information
- Max size is 4GB
- Synt: X Clob;

ii. BLOB:
- It is used to store images but in form of binary format
- Max size is 4 Gb
- Synt: X Blob;

iii. BFILE:
- It is used to store files
- Max size is 4Gb
- Synt: X Bfile;

iv. NCLOB:
- It is used to store multiple languages (Unicode format)

INTERVIEW QUESTIONS

1. DIFFERENCE BETWEEN CHAR AND NCHAR

   a. NCHAR: it used to store fixed length Unicode data. It is often used to store data in different languages. Char on the other hand is store fixed length character data

2. DIFFERENCE BETWEEN CHAR AND VARCHAR2

3. WHAT IS COMPOSITE DATATYPE?

   a. collections are usually referred as composite datatypes .Composite type is one that has components in it.

4. WHICH DATATYPE IS USED FOR STORING GRAPHICS AND IMAGE

2. SQL STATEMENTS:

   a. DDL(DATA DEFINITION LANGUAGE)
      - These are used to define data base objects
        - These are auto commit commands
        - These are session independent

i. CREATE
   - It is used to create data base object
   - Synt: create table tablename (col-1 Datatype(size), col-2 Datatype(size)….);

ii. ALTER
   - It is used to alter the structure of the table

     a. ADD
        - It is used to add the column in a table
        - Synt: alter table tablename add colname Datatype (s);

     b. RENAME
        - It is used to rename a column in table
        - Synt: Alter table tablename rename column oldcol to newcol;

     c. MODIFY:
        - It is used to modify the column in a table
        - Synt: Alter table tablename modify colname Datatype(s);

     d. DROP
        - It is used to drop a column in a table
        - Synt: Alter table tablename drop column colname
        - Synt: Alter table tablename drop column (Col-1, Col-2……);

iii. RENAME:
   - It is used to rename a table name
   - Synt: Rename oldtablename to newtablename

iv. DROP
   - It is used to drop the table from database
   - Synt: drop table tablename;

b. DML (DATA MANIPULATION LANGUAGE)

- It is used to handle the data in the database object
- These are non-auto commit commands
- These are session dependent

i. INSERT

- It is used to insert data into the table
- We can insert the data into table in 2 methods
  a. Direct method
     - It is used to insert the data directly into table
     - Synt: insert into tablename (col-1,col-2) Values (Val-1,Val-2);
  b. Reference method
     - It is used to insert the data into table through prompt
     - Insert into tablename values (&col-1,&col-2);

ii. UPDATE

- It is used to update data in the table
- Synt: Update tablename set col-1=val-1, col2=val-2….. Where condition.

iii. DELETE

- It is used to delete the data in the table
- Synt: Delete from tablename where condition

c. DQL (DATA QUERY LANGUAGE)

i. SELECT

- It is used the retrieve the data from the table
  - Synt: select * from tablename

d. TCL (TRANSACTION CONTROL LANGUAGE)

- It is used to save the transactions on the table

ii. COMMIT

- COMMIT command is used to make changes permanent save to a database during the current transactions
- Implicit commit
  a. It is applied by the system
- Explicit commit
  a. It is applied by user

iii. ROLLBACK

- It is execute at the end of the transaction and undo any changes made since the begin transactions

iv. **SAVE POINT**

- This command save the current point with the unique name in the processing of a transaction
- It is only for temporary purpose
- Synt: Savepoint S1;
- Rollback to s1 or rollback to Savepoint s1

v. **TRUNCATE**

- It works like a Delete+commit
- Synt: Truncate table tablename

e. **DCL (data control language)**

- It is used to provide the access to users

i. **GRANT**

- It is used to provoke the permission to users
- Grant <privileges> on <object_name> to <user_name>[with grant options]
- EX: grant select on student to sudha
- Ex: grant select, insert on student to sudha;
- EX: grant all on student to sudha

ii. **REVOKE**

- It is used to cancel the permissions to users
- Revoke <privileges> on <object_name> from <user_name>
- EX: revoke select on student from sudha;
- EX: revoke all on student from sudha;

INTERVIEW QUESTIONS:

1. WHAT ARE THE VARIOUS PRIVILEGES THAT A USER CAN GRANT TO ANOTHER USER?

A: ALL DML, DRL, DDL AND TCL COMMANDS

2. DIFFERENCE BETWEEN TRUNCATE AND DELETE, WHICH IS FASTER

A: TRUNCATE WORKS FASTER

BECAUSE DELETE HAS TO PERFORM 2 OPERATIONS

1. DELETE FROM THE DATABSE

2. WRITE THE DELETE RECORD INTO ROLLBACK SEGMENTS

3. DIFFERENCE BETWEEN TRUNCATE AND DROP

4. DIFFERENCE BETWEEN DELETE AND DROP

5. COMMIT, SAVEPOINT, ROLLBACK, AUTOCOMMIT

A: set AUTOCOMMIT ON to execute COMMIT statement automatically

3. CLAUSES

a. SELECT CLAUSE
   - It is used to retrieve the data from table

b. FROM CLAUSE
   - It is used to retrieve the data from which table

c. WHERE CLAUSE
   - It is used provide the conditions
   - It is used to filter the data from records based on condition
   - It won't allow group functions and alias names
   - Synt: Select * from emp where empno=10;

d. GROUP BY
   - It is used to make the data into group records
   - It is not possible to provide the group functions along with the normal columns in a select statement without using group by clause
   - Select deptno from emp group by deptno;
   - Columns used in select must be used with group by, otherwise it was not a group by expression
   - Select deptno, sum (sal) from emp group by deptno
   - Select deptno, job, sum (sal) from emp group by deptno, job

e. HAVING CLASUE
   - It is used to provide the conditions
   - It is used to filter the data from grouped data base on conditions

- Select deptno, job, sum (sal) from emp group by deptno, job having sum(sal)>3000 ;

f. **ORDER BY CLAUSE**
   - It is used to make the data in order
   - Synt: select * from emp order by sal;

g. **DISTINCT CLAUSE**
   - It is used restrict the duplicate records
   - Synt: select distinct(empno) from emp;

INTERVIEW QUESTIONS:

1. WRITE A QUERY TO FIND THE MAX SAL?

A. SELECT * FROM (SELECT DISTINCT (SAL) FROM EMP ORDER BY SAL DESC) WHERE ROWNUM = &ENTER_NTH_VALUE

2. USE OF GROUP BY AND HAVING CLAUSE?

3. DIFF B/W WHERE CLAUSE AND HAVING CLAUSE

A. IN WHERE CLAUSE WE CAN'T USE AGGREGATES LIKE—SUM, AVG, MAX (USED IN HAVING CLAUSE). HAVING CLAUSE IS AN ADDITIONAL FILTER TO THE WHERE CLAUSE

4. OPERATIONS

a. **ARTHEMATIC OPERATERS (+, -, *,/)**
   - It is used to do the mathematical functions
   - EX: select 2+2,2*2 from dual;

b. **LOGICAL OPERATERS (AND , OR , NOT)**
   - Select * from emp where deptno=10 and sal>2000;
   - Select * from emp where deptno=10 or deptno=20;
   - Select * from emp where not deptno=20;

c. **RELATIONAL OPERATERS (<, >, =, <=, >=, !=)**
   - Select * from emp where sal>2000;

d. **SPECIAL OPERATERS (IS, IN, LIKE, BETWEEN, NOT IN)**
   - Select * from emp where comm is null;
   - Select * from emp where dept in (10,20);
   - Select * from emp where sal between 1000 and 2000;
   - Select * from emp where ename like 'K%';

e. SET OPERATERS

- By using set operators we can join more than one query such queries are called compound queries
- In each of the select statement there must be same number of columns and same Datatype but must not of same size

i. UNION ALL

- It displays the all values along the duplicate values also
- Two queries must have equal number of columns
- Select * from query-1 union all select * from query-2;

ii. UNION

- It is similar to unionall but it won't display the duplicate values
- Select * from query-1 union select * from query-2

iii. INTERSECT

- It displays common values from two queries
- Select * from query-1 intersect select * from query-2

iv. MINUS

- It display first query records, which are not found in the second query
- Select * from query1 minus select * from query2

INTERVIEW QUESTIONS

1. DIFFERENCE B/W UNION AND UNION ALL

2. DIFFERENCE B/W UNION, MINUS, INTERSECT

3. IN VS EXISTS OPERATERS … GIVE ME ONE SCENARIO?

A: EXISTS IS A COMPARISION OPERATER,WHICH IS USED TO CHECK AND MATCH RECORDS B/W TWO QUERIES ON CORRELATION BASIS AND RETURNS A BOOLEAN OUTPUT .THE TWO QUERIES ARE DESIGNED AS THE PARENT QUERY AND SUB QUERY

NOT EXISTS IS THE NEGATION FORMAT OF EXISTS,

SELECT * FROM EMP E WHERE EXISTS (SELECT * FROM DEPT WHERE DEPTNO=E.DEPTNO)

In	Exists
The inner query is executed first and the list of values obtained as its result is used by the outer query. The inner query is executed for only once.	The first row from the outer query is selected, then the inner query is executed and, the outer query output uses this result for checking. This process of inner query

	execution repeats as many no.of times as there are outer query rows. That is, if there are ten rows that can result from outer query, the inner query is executed that many no.of times.
WHEN YOU USE 'IN', WHILE CHECKING FOR WHERE CONDITION SQL SERVER ENGINE DOES WHOLE TABLE SCAN.	IF YOU USE 'EXISTS' AS SOON AS ENGINE FINDS THE REQUIRED ROW IT WILL STOP EXECUTING QUERY AND GOING FURTHER SCANNING TABLE.
Using the IN clause, you're telling the rule-based optimizer that you want the inner query to drive the outer query (think: IN = inside to outside).	When you write EXISTS in a where clause, you're telling the optimizer that you want the outer query to be run first, using each value to fetch a value from the inner query (think: EXISTS = outside to inside).
Returns true if a specified value matches any value in a sub-query or a list.	Returns true if a sub-query contains any rows.

5. FUNCTIONS

a. NUMBER FUNCTIONS

i. POWER (M,N)

- **Synt: select power (25, 2) from dual;**

DUAL:

- **It is a dummy table provided by oracle engine**
  - **a. It has only one column associated with varchar datatype**

ii. SQRT (M)

- **Select sqrt (625) from dual;**

iii. MOD (M, N)

- **Select mod (5, 2) from dual;**

iv. ASCII (C)

- **Select ascii (‘a’) from dual;**

v. CEIL (M)

- **It displays the next highest value**
- **Synt: select ceil(12.45) from dual;**

vi. FLOOR (M)

- **It displays the next lowest value**
- **Synt: select floor(13.65) from dual;**

vii. ROUND (M, N)

- It rounds the value up to given no.of position. That is if last eliminating value is >=5 then it simply add one value to the left adjacent value.
- It check the condition
- Synt: select round(15.2345, 2) from dual;

viii. TRUNC (M, N)

- It is similar to round ,but it doesn't check the condition
- Select trunk(12.562, 2) from dual;

QUESTIONS

1. DIFF CEIL & FLOOR

2. DIFF B/W ROUND AND TRUNC

b. STRING FUNCTIONS

i. LENGTH (S)

- It is used to display the number of characters in a given string
- Synt: select length('ebs') from dual;

ii. REVERSE (S)

- It is use to reverse the given string
- Synt: select reverse('ebs') from dual;

iii. UPPER (S)

- It is used to convert the string to upper characters
- Synt: select upper('ebs') from dual;

iv. LOWER (S)

- It is used to convert the string to lower characters
- Synt: select lower('ebs') from dual;

v. INITCAP

- It is used to convert the first character into upper character in a given string
- Synt: select initcap('ebs') from dual;

vi. CONCAT (s1, s2)

- It is used to merge the two strings ,and we have to use the '||' symbol
- Synt: select concat('ebs','solutions') from dual;
- Synt: select 'ebs'||'business'||'solution' from dual

vii. LTRIM (S, C)

- It is used to remove the character from the left end of the given string, if the character is found
- Select ltrim(‘ebs’,’e’) from dual;

viii. RTRIM (S, C)

- It is used to remove the character from the right end of the given string, if the character is found
- Select rtrim(‘ebs’,’s’) from dual;

ix. TRIM

- It is used to remove the character from both sides of the given string,
- Synt: select trim(‘e’ from ‘ebse’) from dual;

x. LPAD

- It is used to add character from left end
- Select lpad(‘ebs’,5,’&’) from dual;

xi. RPAD

- It is used to add character from right end
- Select rpad(‘ebs’,7,’&’) from dual;

xii. TRANSLATE (S, C, C)

- It is used to translate the string character wise
- It is not possible to translate the entire string
- Select translate (‘welcome’, ’w’, ’t’) from dual;
- Select translate (‘india’,’id’,’xy’) from dual

xiii. REPLACE (S, S, S)

- It is used to replace entire string
- It is not possible to replace more than one string
- Synt: select replace(‘e business solutions’, ’business’, ’b’) from dual;
- Select replace (‘india’,’id’,’xy’) from dual

xiv. DECODE (column, condition, do),……………column)

- It is used to replace more than one string
- It works like ‘if condition’ but is doesn t allow the relational operaters
- Select job,
  ```
  decode (job, ’manager’, ’mgr’, ’clerk’,’ clk’)
  from emp
  ```
- Select job,
  ```
  decode (job, ’manager’, ’mgr’, ’clerk’,’ clk’,job)
  from emp
  ```

xv. **CASE(when condition then result else default value)**

- It is used to replace more than one string by using relational operater
- Select job,
```
case job
when ‘manager’ then ‘mgr’
else ‘job’
end case
from emp;
```

xvi. **SUBSTR (S, C, N)**

- It is used to display the set of characters from a given string
- S= string
- C=cursor position @
- N= no.of characters
- Synt: select substr(‘welcome’,1,3) from dual;

xvii. **INSTR (S, C, C, N)**

- It is used to display the position of a given character
- S= string
- C= character
- C=cursor position @
- N= occurance
- Synt: select instr(‘welcome’,’e’,1,1) from dual;

xviii. **COALESCE:**

- This will give the first non-null string
- EX: select coalesce (‘a’,’b’,’c’),coalesce(‘null’,’a’,’null’) from dual
- Coalesce   coalesce
-     a   a

QUESTIONS

1. DIFF B/W TRANSLATE AND REPLACE

Translate: cleaning up data

Translate (‘s1, s2, s3, s4’, ‘,’)

And other common differences

---

2. DIFF B/W DECODE AND CASE

Everything DECODE can do, CASE can. There is a lot more that you can do with CASE, though, which DECODE cannot. Following is the list of differences -

1. DECODE can work with only scaler values but CASE can work with logical operators, predicates and searchable subqueries.

```
Select ename,
        case grade
        When sal<1000 then ‘grade1’
        When sal>2000 and ssal<10000then ‘grade2’
        Else ‘grade3’
        End case
From emp;



Select ename, case
                ---predictlable with “in”
                ---set the category based on ename list
        When e.ename in (‘king’, ‘smith’, ‘ward’);
        then ‘top bosses’
                ---searchable subquery
                ---identify if emp has a reportee
        When exists (select 1 from emp emp1 where emp1.mgr=e.empno)
        Then ‘mangers’
        Else ‘general employees’;
        End case
From emp;
```

2. CASE can work as a PL/SQL construct but DECODE is used only in SQL statement. CASE can be used as parameter of a function/procedure.

```
Var a varchar2(5);
Exec :a:=’three’;
Pl/sql procedure successfully completed

Create or replace pro_test (I number)
Is
```

```
Beign
        Dbms_output.put_line('output='||i);
End;
Procedure created
```

```
exec pro_test(decode (:a, 'three', '3','0'));
```

function or pseudo-column 'DECODE' may be used inside sql statement only

```
exec pro_test(case :a, when 'three' then 3 else 0 end);
```

output=3

3. CASE expects datatype consistency, DECODE does not.

```
Select decode (2, 1, 1,
                    '2', '2'
                    '3') t
From dual;

        T
------------
        2
```

```
Select case 2 when 1 then 1
                When '2' then '2'------------error
                Else '3';
End case;
End;
```

Inconsistent datatype: expected number got char.

4. CASE executes faster in the optimizer than does DECODE.
5. CASE is a statement while DECODE is a function.

c. DATE FUNCTIONS

i. SYSDATE

- **It is used to display the system date**
- **Select sysdate from dual**

ii. CURRENT_DATE

- It is used to current date in session's time zone
- Select current_date from dual;

iii. ADD_MONTHS

- It is used to add or subtract no.of months from a given date
- Select add_months(sysdate,1) from dual

iv. MONTHS_BETWEEN (DATE1, DATE2)

- It is used to display no.of month b/w two dates
- Select months_between(sysdate, hiredate) from emp;

v. NEXT_DAY (DATE,FORMAT)

- It is used to display the next day date, based on the format
- Select next_day(sysdate, 'sun') from dual;

vi. LAST_DAY (DATE)

- It is used to display the last day of the given month
- Select last_day(sysdate) from dual;

DATE FORMATS:

DATE FORMATS
D -- No of days in week
DD -- No of days in month
DDD -- No of days in year
MM -- No of month
MON -- Three letter abbreviation of month
MONTH -- Fully spelled out month
RM -- Roman numeral month
DY -- Three letter abbreviated day
DAY -- Fully spelled out day
Y -- Last one digit of the year
YY -- Last two digits of the year
YYY -- Last three digits of the year
YYYY -- Full four digit year
SYYYY -- Signed year
I -- One digit year from ISO standard

IY -- Two digit year from ISO standard
IYY -- Three digit year from ISO standard
IYYY -- Four digit year from ISO standard

Y, YYY -- Year with comma
YEAR -- Fully spelled out year
CC -- Century
Q -- No of quarters
W -- No of weeks in month
WW -- No of weeks in year
IW -- No of weeks in year from ISO standard
HH -- Hours
MI -- Minutes
SS -- Seconds
FF -- Fractional seconds
AM or PM -- Displays AM or PM depending upon time of day
A.M or P.M -- Displays A.M or P.M depending upon time of day
AD or BC -- Displays AD or BC depending upon the date
A.D or B.C -- Displays AD or BC depending upon the date
FM -- Prefix to month or day, suppresses padding of month or day
TH -- Suffix to a number
SP -- suffix to a number to be spelled out
SPTH -- Suffix combination of TH and SP to be both spelled out
THSP -- same as SPTH

d. CONVERSION FUNCTIONS

i. TO_CHAR (DATE, FORMAT)

- It is used to system format in to user format
- Synt: select to_char(sysdate, 'day') from dual;

ii. TO_DATE (C, format)

- It is used to convert user format to system format
- Synt: select to_date('21','DD') from dual
- Select to_date('december', 'MM') from dual

iii. TO_NUMBER

- It is used to translate a value of char or varchar datatype to number format
- Select to_number(20) from dual;

QUESTIONS:

1. WHAT ARE THE CONVERSION FUNCTIONS?

2. HOW TO DISPLAY THE EMPLOYEE WHO JOINED AFTER 15 TH OF ANY MONTH

A: SELECT * FROM EMP WHERE TO_CHAR (HIREDATE,'DD')>15

e. GENERAL FUNCTIONS

   i. USER & UID
      - Select user, uid(integer value corresponding to the user) from dual

   ii. GREATEST & LEAST
      - Select greatest(1,2,3), least(1,2,3) from dual;

   iii. NVL (COL, VAL)
      - It is used to handle the null values
      - It works like a if condition
      - Synt: select comm, nvl(comm,0) from dual

   iv. NVL2 (COL, VAL1,VAL2)
      - It is advance of nvl
      - It works like if then else condtion
      - Select comm,nvl2(comm, 0, 100) from dual

   v. dRANK()
      - Select rank(2975) within group(order by sal desc)

   vi. DENSE_RANK()
      - Select dense_rank(2975) withingroup (order by sal desc)

   vii. ROW_NUMBER() OVER(partition by deptno order by empno)
      -

QUESTIONS

1. HOW TO GENERATE ROWNUMBER WITH OUT USING ROWNUM

A: SELECT ENAME, ROW_NUMBER() OVER (PARTITION BY DEPTNO ORDERBY EMPNO) RN FROM EMP

2. DIFF B/W ROWNUM AND ROW_NUMBER

A: ROWNUM IS A PSEUDO COLUMN THAT ASSIGN A NUMBER TO EACH ROW RETURNED BY A QUERY

ROW_NUMBER IS A ANALTICLA FUNCTION THAT ASSIGNS A NUMBER TO EACH ROW ACCORDING TO ITS ORDERING WITHIN A GROUP OF ROWS

3. DIFF B/W ROWNUM, ROW_NUMBER (), RANK, DENSE_RANK, NTILE

ROW_NUMBER(): RETURNS THE SEQUENTIAL NO OF ROW,WITH IN A PARTITION OF A RESULT SET,WITHOUT ANY GAPS IN THE RANKING,THE RANK OF A ROW IS ONE PLUS THE NUMBER OF DISTINCT RANKS THAT COME BEFORE THE ROW IN QUESTION

RANK: RETURNS THE RANK OF EACH ROW WITHIN THE PARTITION OF A RESULT SET .THE RANK OF A ROW IS ONE PLUS THE NUMBER OF RANKS THAT COME BEFORE THE ROW IN QUESTION

DENSE-RANK: RETURNS THE RANK OF ROWS WITHIN THE PARTITION OF A RESULT SET, WITH OUT ANY GAPS IN THE RANKING.THE RANK OF A ROW IS ONE PLUS THE NUMBER OF DISTINCT RANKS THAT COME BEFORE THE ROW IN QUESTION

NTILE: DISTRIBUTES THE ROWS IN AN ORDERED PARTITION INTO A SPECIFIED NUMBER OF GROUPS.THE GROUPS ARE NUMBERED,STARTING AT ONE.FOR EACH , NTILE TETURNS THE NUMBER OF THE GROUP TO WHICH IT BELONGS

QUERY THIS

```
SELECT ENAME,

ROW_NUMBER () OVER (ORDER BY SAL DESC) ROW_NUM,

RANK () OVER (ORDER BY SAL DESC) RANK,

DENSE_RANK () OVER (ORDER BY SAL DESC) DESN_RANK,

NTILE (3) OVER (ORDER BY SAL DESC) NTILE

FROM EMP
```

4. WHATS IS THE DIFFERENCE BEWTWEEN ROWNUM AND ROWID

ROWNUM	ROWID
ROWNUM STARTS WITH 1AND INCREMENT WITH 1	ROWID ARE HEXA DECIMAL VALUES
ROWNUM VALUES ARE TEMPORARY	ROWID ARE PERMANENT
ROWNUM VALUES ARE GENERATED WHEN QUERY IS EXECUTED	ROWID IS GENERATED WHEN THE ROW IS CREATED OR INSERTED

5. WHAT ARE ANALYTICAL FUNCTIONS

A: GENERAL FUNCTIONS ARE ALSO CALLED ANLYTICAL FUNCTINS, WHICH ARE GENERALLY USED TO IMPROVE THE PERFORMANCE

ADVANTAGES

1. THESE ARE MORE FLEXIBLE TO USE

2. NO NEED TO TUNE THE QUERY PERFORMANCE

6. LAG AND LEAD FUNCTIONS

A. SEE PDF

7. HOW TO DELETE DUPLICATE RECORDS

A: delete from emp where rowid in (select lead(rowid) over (partition by empno order by null from emp);

8. NULLIF, NVL, NVL2?

A: NULLIF: COMPARES TWO EXPRESSINS, AND RETURNS NULL IF THEY ARE EQUAL, RETURNS THE FIRST EXPRESSION IF THEY ARE NOT EQUAL

f. AGGRIGATE FUNCTIONS

i. COUNT(*)
- **It is used to count of all the record from the table**
- **Synt: select count(*) from emp**

ii. COUNT(COLUMN)
- **It is used to count the given column values**
- **Synt: select count(empno) from dual;**

iii. MIN
- **Select min(sal) from emp**

iv. MAX
- **Select max(sal) from emp**

v. AVG
- **Select avg(sal) from emp**

vi. SUM
- **Select sum(sal) from emp**

QUESTIONS

1. DIFF B/W COUNT (*) AND COUNT (1)

A: BOTH ARE SAME, NO.OF RECORDS IN THE TABLE REGARDLESS OF NULL VALUES AND DUPLICATES

2. DIFF B/W COUNT (*) AND COUNT (COLUMN NAME)

A: count(col_name):IT COUNTS A SPECIFIED COLUMN RECORDS, HERE NULL VALUES ARE NOT COUNTED (DUPLICATED ARE COUNTED)

3. WHO EARN MORE THAN AVERAGE SALARY OF THEIR DEPARTMENT

A: SELECT * FROM EMP WHERE SAL> (SELECT AVG(SAL) FROM EMP)

4. NTH MAX SALARY

A: REFER DISTINCT CLAUSE QUESTIONS

5. QUERY TO DISPLAY MAX SAL FORM EACH DEPT

A: SELECT DETPNO, MAX (SAL) FROM EMP GROUP BY DEPTNO

6. HOW TO DISPLAY COLUMN COUNT

A: SELECT COUTN (COLUMN_NAME) FROM TABLE_NAME

7. IF A STRING IS THERE LIKE S1, S2, S3, S4 .HOW TO FIND COUNT OF COMMAS IN THIS?

A: SELECT LENGTH (S1, S2, S3, S4) – LENGTH (REPLACE (‘S1, S2, S3, S4’, ’,’)) AS COUNT_COMMA FROM DUAL

6. CONSTRAINTS

Constrain are rules which are used to allow the valid data

a. UNIQUE

- This is used avoid duplicate record, but it has null values
- <u>Column level</u>: create table tablename(sno number(5) unique, rollnumber number(5) unique, marks number(5))
- <u>Table level</u>: create table tablename(sno number(5), rollnumber number(5), marks number (5),unique(no, rollnumber))
- <u>Alter level</u>: alter table tablename add unique (no, rollnumber)

b. NOT NULL

- This is used to avoid null values
- We can add only in column level
- Create table tablename (sno number (5) not null, rollnumber number (5) not null, marks numbers (5))

c. **PRIMARY KEY**

- This is combination of unique and not null
- We can add this in three levels

d. **FOREIGN KEY**

- This is used to reference the parent table primary key column
- Foreign key always attached to the child table
- We can add this constraint in table and alter levels only
- Table level: create table emp (empno number (5), ename varchar, deptno (5), foreign key(deptno) references dept (deptno))
- Alter level: alter table emp add foreign key (deptno) references dept (deptno)

NOTE: ONCE THE PRIMARY KEY AND FOREIGN KEY RELATIONSHIP HAS BEEN CREATED THEN YOU CANNOT REMOVE ANY PARENT REOCRD IF THE DEPENDENT CHILD EXISTS

USING DELETE CASCADE:

By using this clause you can remove the parent record even its child exists

This is because, oracle will also delete dependent records from child table, if this clause is present while creating foreign key constraint.

Table level: create table emp (empno number, ename varchar, deptno number, foreign key (deptno) references dept(deptno) on delete cascade ;

Alter level: alter table emp add foreign key (deptno) references dept (deptno) on delete cascade

e. **CHECK CONSTRAINT**

- This is used to insert the values based on specified condition
- We can add it in three levels
- Column level: create table student(marks number(5) check (marks>300))

f. **DEFAULT VALUE**

- When you define a column with the default keyword followed by a value, you are actually telling the database that, o insert if a row was not assigned a value for this column, use the default value that you have specified
- We can add in column level
- Create table student (no number, marks number default 30)

g. **OPERATIONS WITH CONSTRAINTS**

i. **Enable**

- This will enable the constraint. Before enable, the constraint will check the existing data
- Alter table student enable constraint unique key;

ii. **Disable**

- This will disable the constraint
- Alter table student disable constraint unique key;

iii. **Enforce**

- This will enforce the constraint rather than enable for future inserts or updates. This will not check for existing data while enforcing data.
- Alter table student enforce constraint unique key;

iv. **Drop**

- This will remove the constraint
- Alter table student drop constraint unique key;

v. Once the table is dropped ,constraints will automatically will drop

QUESTIONS:

1. HOW TO ENABLE AND DISABLE CONSTRAINT

2. DIFF B/W PRIMARY AND FOREIGN KEY

A:

PRIMAY KEY	FOREIGN KEY
PRIMAY KEY UNIQUELY IDNTIFIES A RECORD IN THE TABLE	FOREIGN IS FIELD IN THE TABLE ,THAT IS PRIMARY KEY IN OTHER TABLE
PRIMARY CAN NOT ACCEPT NULL VALUES	CAN ACCEPT MULTIPLE NULL VALUES
BY DEFALUT, IT IS CULSERED INDEX AND DATA IN THE DATA BASE TABLE IS	FOREIGN KEY DO NOT AUTOMATICCLY CREATE AN INDEX, CLUSTERED OR NON-

PHYSICALLY ORGANIZED IN THE SEQUENCE OF CLUSTERED INDEX.	CLUSTERED .YOU CAN CREATE MANUALLY AN INDEX ON FOREIGN KEY
WE CAN HAVE ONLY ONE PRIMARY KEY IN A TABLE	WE CAN HAVE MORE THAN ONE FOREIGN KEY IN A TABLE

3. PRIMAR KEY AND UNIQUE KEY?

4. IN A TABLE HAVE ONE COLUMN PRIMARY KEY. IT WILL NOT ALLOWS NULL VALUES AND DUPLICATE VALUES, INSTEAD OF PRIMARY KEY WHY CANT WE USE UNIQUE AND NOT NULL KEYS? WHAT THE DIFF?

A: 1. PRIMARY KEY WILL CREATE CLUSTER INDEX AUTOMATICALLY INCASE OF UNIQUE+NOT NULL CONSTRIANT WILL CREATE NON-CLUSTER INDEX

2 .A TABLE HAVE ONLY ONE PRIMAY KEY. INCASE OF UNIQUE+NOT NULL CONSTRAINT NOT RESTRICTION. i.e., WE CAN GIVE IT TO MORE DIFFERENT COLUMNS

7. **JOINS**
   a. **SIMPLE JOIN / INNER JOIN /NATURAL JOIN**
      i. **EQUI JOIN**
         - **It is used to join two tables based on equal condition**
         - **Select * from emp, dept where emp.deptno=dept.deptno**
      ii. **NON EQUI JOIN**
         - **It is used to join two tables based on not equal condition**
         - **Synt: select * from emp, dept where emp.deptno!= dept.deptno**
   b. **OUTER JOIN**
      i. **LEFT OUTER JOIN**
         - **It is used to display the full details of the left table and matched record of the right table**
         - **Synt: select * from emp, dept where emp.deptno=dept.deptno(+)**
      ii. **RIGHT OUTER JOIN**
         - **It is used to display the full details of the right table and matched records of the left table**
         - **Synt: select * from emp, dept where emp.deptno(+)=dept.deptno**
      iii. **FULL OUTER JOIN**
         - **If you join left and right outer join with union then we get full outer join**

- This will display matched and non-matched records from both tables
- Synt: select * from emp, dept where emp full outer join dept on (emp.deptno=dept.deptno)
- Select * from emp, dept where emp.deptno = dept.deptno(+)

  Union

  Select * from emp, dept where dept.deptno (+) = emp.deptno

c. SELF JOIN

- It is used to join the table itself
- Synt: select * from emp e1, emp e2 where e1.empno=e2.mgr

b. CROSS JOIN

- This will give the cross product
- Select * from emp cross join dept
- Select * from emp, dept

QUESTIONS:

1. WHAT ARE JOINS? HOW MANY TYPES OF JOINS?

2. WHAT IS DIFF B/W JOIN AND SET OPERATER?

A:

join

IN JOIN, DATA MUST BE IN TABLE

o/p: deptno dname loc deptno1 name addr

Set operator:

IN SET OPERATER, DATA MUST MATCH

same number of columns and same Datatype but must not of same size

o/p:	deptno	dname	loc
	10	accountig	newyork
	10	kiran	hyd
	20	manager	London
	20	chanu	sri

We can say that set operaters are like vertical joins

3. WHAT IS SELF JOIN? WHY IS IT REQUIRED?

8. SYNONYMS

- It is used to hide the owner of the table
- It works like a mirror image of the table
- It does not have a own structure
- It is dependent on the table
- We can possible to create synonym on table but we can't create directly synonym
- We can synonym on synonym
- All synonyms are stored in ALL_synonyms table

a. PRIVATE SYNONYM

- Private synonym is available to the particular user who creates
- Create synonym synonym_name for table_name

b. PUBLIC SYNONYM

- Public synonym is created by DBA which is available to all users
- Create public synonym synonym_name for table_name
- Drop synonym s1;

QUESTIONS:

1. TELL ME SOMETHING ABOUT SYNONYMS

9. VIEWS

- These are advance of the synonyms
- It is a virtual table to hide the base table and it works like a mirror image of the table
- It doesn't have own structure
- It is not possible to modify the structure of a table by using views
- We can define view on synonym and synonym on views
- We can define view on particular column only
- All views are stored in all_views

a. **SIMPLE VIEW**
   - It is used define a view on single table that view are called single views
   - Create view view_name as select * from emp

b. **COMPLEX VIEW**
   - It is used to define view on multiple table that view is called complex views
   - Create view view_name as select * from emp, dept where emp.deptno=dept.deptno

c. **FORCE VIEW**
   - It is used to define a view without base table
   - Create force_view view_name as select * from emp

d. **VERTICAL VIEW**
   - It is used to define view on specified columns
   - Create view view_name as select empno, ename, job from emp;

e. **HORIZONTAL VIEW**
   - It is used to define view on specified records
   - Create view view_name as select * from emp where deptno=10

f. **PARTITION VIEW**
   - It is used to define view on compound queries(set operators like union)
   - Create view view_name as query1 union query2

g. **MATERALIZED VIEW**
   - It is one of the view which is having the own structure
   - It doesn't allow the DML operations on view
   - It is used to store the historical data
   - We can define this view on table, which is having the primary key
   - Create materialized view view_name as select * from emp

How to refresh materalized view?

```
CREATE OR REPLACE
PROCEDURE MAT_VIEW_FOO_TBL
IS
BEGIN
   DBMS_MVIEW.REFRESH('v_materialized_foo_tbl')
END MAT_VIEW_FOO_TBL IS;
```

Exec mat_view_foo_tbl

h. INLINE VIEW

- It works a "QUERY" ,which is having the query in from clause instead of table
- SYNT: select * from (select * from emp);
- First 5 records
- Select * from (select emp.*, rownum r from emp) where r<=5
- Last 5 records (r>max-n)
- Select * from (select emp.*, rownum r from emp) where r> (select max(rownum)-&n from emp);
- Even no. records
- Select * from (select emp.* , rownum r from emp) where mod(r,2)=0
- Last record
- Select * from (select emp.* ,rownum r from emp) where r=(select count(*) from emp)

QUESTIONS:

1. WHAT IS VIEW? TELL ME DIFFERENT VIEWS?

2. WHAT IS DIFF B/W VIEW AND MATERIALIZED VIEW?

MATERIALLIZED VIEW	VIEW
M.V ARE DISK BASED AND UPDATED PERIODICALLY BASED ON THE QUERY	VIEWS ARE VIRTUAL ONLY AND RUN THE QUERY DEFINITION EACH TIME THEY ARE ACCESSED
M.V PERFORMANC IS BETTER THAN VIEW	PERFORMANCE IS SLOW THAN M.V
IT HAS ITS OWN STRUCTURE	IT DOESN'T HAVE THE STRUCTURE
INDEXING, JOINING IS DONE AT TIME OF CREATION, SO NO NEED TO FIRE EVERY TIME JOIN STATEMENT	FOR BETTER PERFORMANCE DON'T USE JOIN HERE, IF U WANT TO USED INDEXED BASED COLUMNS

3. WHAT IS INLINE VIEW?

10. INDEXES

- It is one of the objects, to retrieve the data fastly from the database
- It is used to increase performance while retrieve the data from the database
- It will make the use of user_ids
- All indexes are stored in all_indexes

a. **SIMPLE INDEX/ BTREE /ASCENDING**
   - **It is used to create an index on single column of a table**
   - **Create index index_name on table_name (column_name)**

b. **COMPLEX INDEX**
   - **It is used to create index on multiple columns of a table**
   - **Create index index_name on table_name (col1, col2)**

c. **UNIQUE INDEX**
   - **It is used to create index on columns which are having unique data**
   - **Create unique index index_name on table_name (col1);**

d. **FUNCTIONAL INDEX**
   - **It is used to create a index on columns while making use of functions**
   - **Create index index_name on emp (length(ename));**

e. **BITMAP INDEX**
   - **It is used to create bitmap index on column (indexs on images), used on low cardinality columns**
   - **Create bitmap index index_name on voter_card (photo)**
   - **Create bitmap index stud_ind on student (sex);**

f. **BTREE OR ASCENDING INDEX**
   i. **A BTREE index is designed to provide both rapid access to individual rows and quick access to groups of rows within a range. The BTREE index does this by performing a succession of value comparisons. Each comparison eliminates many of the rows.**

g. **DESCENDING INDEX**
   i. **This is reverse of ascending index, you can categorize data in b-tree index in descending order as well**
   ii. **Create index stud_in on student (sno desc)**

QUESTIONS

1. DIFF BTREE AND BITMAP INDEX?

A: 1. B-TREE HAS LOW CARDINALITY VALUES, WHERE AS BITMPA

11.CLUSTERS

- **It is a logical boundary which is used to improve the overall performance of the database**
- **We can create cluster on tables but we can’t on columns**

- We can possible to create index on cluster
- All clusters are stored in all_clusters
- <u>Synt:</u>
    i. Create cluster clst8 (sno number(5))
    ii. Create table clusttable8 (sno number(5)) cluster clst8(sno)
    iii. Create index clstindex8 on cluster clst8

12.SEQUENCES

- It is used to create sequence on columns in a table
- While insert the data into tables we use the sequence
- All sequences are stored in all_sequences

a. <u>Synt:</u>
    i. Create sequence sequence_name
    ii. Create sequence sequence_name increment by 1 start with 1;

b. It contains two functions
    i. <u>CURRVAL</u>: it is used to insert current value
    ii. <u>NEXTVAL</u>: it is used to insert next value
        - Create table t1 (sno number(5), cno number(5));
        - Insert into t1 values (seql.currval, seql.nextval)
    iii. Step1: create sequence seq1
    iv. Sterp2:create table stu (sno number(5), cno number(5))

13.SUB QUERIES

- Query with in the query is called sub query

a. SIMPLE SUB QUERIES
    - In simple subquery first inner query is executed independently, based on inner query value outer query is executed
    - Outer query is dependent on inner query but inner query doesn't depend on outer query
    - Inner query is executed only once
    - Synt: select * from emp where empno= (select * from emp)

Ex1: display the employees who are working in research department

A: select * from emp where deptno= (select deptno from dept where dname='RESEARCH')

EX2: Display the employee details who are getting max salary

A: select * from emp where sal= max (sal)--group functions are not allowed

Select * from emp where sal= (Select max (sal) from emp)

Select * from emp where sal= (select max (sal) from emp group by deptno)—single row sub query return more than one row

Select * from emp where sal in (select max (sal) from emp group by deptno)

EX3: display the employee details who are getting 2nd highest salary

A: select * from emp where sal= (select max (sal) from emp where sal<(select max(sal) from emp))

EX4: display the employees who are reporting to king

A: select * from emp where empno= (select empno from emp where ename='king')

EX5: select department details which are having more than 5 employees

A: select * from dept where deptno= (select deptno from emp group by deptno having count (*)>=5)

EX6: display the duplicate records in the table?

A: select * from emp where rowid not in (select max (rowid) from emp group by empno)

b. CO RELATED SUB QUERIES

- The outer query is executed first and for every row of outer query, inner query will get executed.
- So the inner query will get executed as many times as no.of rows in result of outer query.
- The outer query output can use the inner query output for comparison. This means inner query and outer query are dependent on each other

See pdf

FINAL QUESTIONS:

1. ADVANTAGES OF SQL?

a. SQL is used by all vendors who develop DBMS
b. SQL is mainly used for relational database
c. SQL can be used to communicate with the database and get answers to complex question in seconds
d. SQL can do both jobs as programming as well as interactive language
e. SQL is used for linking front end computers and back end database, thus providing client server architecture.
f. SQL supports the latest object based programming and highly flexible
g. SQL is the database language which is used by business and enterprises throughout the globe. For an enterprise application it is a perfect language for a database

2. WHAT IS EXTERNAL TABLE?

a. Prior to oracle database 10g , external tables were read only, however as of oracle database 10g, external tables can also be written to.
b. It enables you to access data from external sources as if it were in a table in the database, like flat file
c. Note that SQL*loader may be the better choice in data loading situations that require additional indexing of the staging.

3. HOW TO TRANSPOSE ROWS TO COLUMNS AND COLUMNS TO ROWS?

Marks	Chanu	Ravi	Ajay
Mat	10	20	30
Phy	40	50	60
che	70	80	90

Marks	Mat	Phy	Che
Chanu	10	40	70
Ravi	20	50	60
Ajay	30	60	90

4. HOW TO DISPLAY FIRST 5 RECORDS FROM A TABLE?

a. Select * from emp where rownum<6

5. PSEUDO COLUMNS?

a. Pseudo columns behave like a column, but it not actually stored in the table.

b. You can select from pseudo columns, but you cannot insert, update, delete their values
c. Ex: rownum, rowid, currval, nextval, user, sysdate etc.,

6. CUBE AND ROLLUP OPERATERS?

a. Rollup: this will give the salaries in each department in each job category along with the total salary for individual departments and the total salary of all the department

Ex:

select deptno, job, sum(sal) from emp group by rollup(deptno, job) order by deptno

Deptno	Job	Sum(sal)
10	Clerk	1300
10	Manager	2450
10	President	5000
10		8750
20	Analyst	6000
20	Clerk	1900
20	Manager	2975
20		10875
30	Clerk	950
30	Manager	2850
30	Salesman	5600
30		9400
		29025

Cube: this will give the salaries in each department in each job category, the total salary for individual departments, the total salary of all the departments and the salaries In each job category.

Ex: select deptno, job, sum (sal) from emp group by cube (deptno, job) order by Deptno;

Deptno	Job	Sum(sal)
10	Clerk	1300

10	Manager	2450
10	President	5000
10		8750
20	Analyst	6000
20	Clerk	1900
20	Manager	2975
20		10875
30	Clerk	950
30	Manager	2850
30	Salesman	5600
30		9400
	Analyst	6000
	Clerk	4150
	Manager	8275
	President	5000
	Salesman	5600
		29025

7. WHAT IS MERGE?

a. You can use merge command to perform insert and update in single command

Ex:

```
Merge into student1 s1
Using (select * from student2) s2
On (s1.no=s2.no)
When matched then
Update set marks=s2.marks
When not matched then
Insert (s1.no, s1.name, s1.marks)
Values (s2.no, s2.name, s2.marks);
```

In the above the two tables are of same structure but we can merge different structured tables also but the datatype of the columns should match

Assume student1 has column like no, name, marks and student2 has no, name, hno, city

Ex:
```
merge into student1 s1
Using (select * from student2) s2
On (s1.no=s2.no)
When matched then
Update set marks=s2.hno
When not matched then
Insert values (s1.no, s1.name, s1.marks)
Values (s2.no, s2.name, s2.hno);
```

8. WHAT IS THE GLOBAL TEMPORATY TABLE?

a. GTT is a database table or plsql table. It stores data temporarily means GTT table stores data for current session. After closing of session the data will be truncated automatically, even if it ends abnormally
b. This table can be used by others. There is no effect on the data of other sessions
c. Indexes can be created on GTT. The content and scope is same as database session

9. WHAT IS INDEX ORGANIZED TABLE

a. An index organized table has a storage organization that is variant of a primary b-tree. Unlike a ordinary table whose data is stored as an unordered collection, data for an index-organized table is stored in a b-tree index structure in a primary key sorted manner