ORACLE NOTES

Database: Collection of Rows & Columns having related data.

Popular DBMS Products are: Dbase, Clipper, Paradox, Foxbase, Foxpro and MS Access

Popular RDBMS Products are: Oracle, Sybase, SQLSERVER, Informics, Ingrace and DB2

RDBMS (Relational Database Management System)

RDBMS: A set of related tables in which one table is related to another by using a common column and data stored is valid according to the integrity constraints is called a RDBMS.

Integrity Constraints- A set of rules.

Integrity Constraints: A set of rules and any RDBMS including ORACLE has three basic rules.

Domain-Column; Tuple-Row

(i) Domain Integrity: It means according to this rule the data stored under every particular domain (or) column must reflect proper information compared to its headings (or) column name. This rule is implemented practically in any RDBMS by using different data types.

(ii) Entity Integrity: According to this constraint, every row stored under a table must be uniquely identifiable. It means there should be at least column in a table, which will not have repetitive value (or) Null Value. Practically, it is implemented in any RDBMS by using “Primary key”.

(iii) Referential Integrity: When values of one column in a table are dependent on values of another table column having common data type.

Self Referential Integrity: When values of one column in a table are dependent on values of another column within the same table, then it is called as “Self Referential Integrity”. Practically, it is implemented by using Reference.

Per Table- 1 Primary key is allowed.

SQL: Structured Query Language.

It is a common language used by any RDBMS to do basic operations on their RDBMS. Following are the features of SQL: -

1. It is simple English like language and uses commands such as Select, Create, Drop etc., in its commands.

2. It is not having If, Loop, Variables and most of the commands are Single line commands.

3. To implement application Logics, SQL has got extension Language popularly called as PL/SQL (Procedural Language of SQL).

4. The entire SQL has been divided into 4 major categories. They are-
    - Data Manipulation Language (SELECT, INSERT, UPDATE, DELETE)
    - Data Definition Language (CREATE, ALTER, DROP, TRUNCATE)
    - Transaction Control Language (COMMIT, ROLLBACK)
    - Data Control Language (GRANT, REOVE)

COMMIT=SAVE; ROLLBACK=UNDO

Database Management System Vs. Relational Database Management System

Most of the DBMS were developed for supporting DOS; whereas most of the RDBMS work on Windows/Unix/Linux Systems/Operating Systems.

1024GB= 1TB (Maximum storage capacity of Oracle 8i= 110TB)
1024TB= 1PB (Maximum storage capacity of Oracle 9i= 530PB)

TB= Terra Byte;
PB= Peta Byte

(i). Most of the DBMS used to store data in terms of Giga Byte (GB); whereas most of the RDBMS store data in terms of Terra Bytes (Or) Peta Bytes.

(ii). Oracle 9i can store data up to 530 PB.

(iii). Most of the DBMs were developed to work on single machine; whereas most of RDBMS were on Network and Oracle 9i has a capacity of pulling more than 10,000 clients per minute provided Hardware and Network supports that.

(iv). Survival of Data: Oracle Corporation claims that Oracle 8i can survive the data for 320 years and Oracle 9i can survive the data for 510 years, if it is maintained properly.

(v). Security: Three levels of Security by way of identification: User Name; User Password, String Code and permissions. Etc.

Triggers= Security Based programs.

Security: The main difference between a DBMS & RDBMS is that of Security.
DBMS were not having any security. Even if they were there, it was very min.

In case of RDBMS, there are minimum three levels of securities:-

a) User has to compulsorily give a user ID and a password to connect to the database.
b) Even after getting connected to database, what work user can do depends on the permissions given by the Database Administrator.
c) There are certain securities programs like Triggers, which will take care of the database even when Administrator is not personally present on the machine.

User Name: Scott (Dummy account given by Oracle for the practice)
Password: Tiger

Every command in Oracle should end with "; "

To see the list of the Tables present, the following command is used: -
```
SQL> SELECT * FROM TAB;
```

For clearing the screen:
```
SQL> CL SCR;
```

TO ALTER THE SESSION:
```
SQL> ALTER SESSION SET NLS_DATE_FORMAT='DD/MM/YYYY';
```

CREATE: This command is used to create any database object including Tables, Views, and Sequence etc.,

Following is an example of creating a Table called STUDENT with the help of 4 columns (STNO, STNAME, DOA, FEES)

```
SQL> CREATE TABLE STUDENT
        (STNO NUMBER(3) PRIMARY KEY,
        STNAME VARCHAR2(30),
        DOA DATE,
        FEES NUMBER(6));
```

For any Numerical Field, Maximum 18 Nos. is allowed.
For any Character Field, Maximum 2000 characters are allowed.

For any Date Column, 8 Nos. is allowed.

The important data types of SQL are:-
CHAR, VARCHAR2, NUMBER, DATE

VARCHAR2 is used to save Bytes according to the Characters of the Name.
To see the columns present in a Table, DESCRIBE command is used

```
SQL> DESCRIBE STUDENT;
```

INSERT: In Oracle to Add or New Records, INSERT command is used.
There are four different types of INSERTS in Oracle, they are: -

i. INSERT ALL FIELDS ONE RECORD
ii. INSERT FEW FIELDS ONE RECORD
iii. INSERT ALL FIELDS MANY RECORD
iv. INSERT FEW FIELDS MANY RECORD

Note: in Oracle for CHARACTER, DATE field, information is inserted within the Single quotes. Where as Numerical fields are entered directly. Entering DATE, in the month field, only first three characters should be used.

I. INSERT ALL FIELDS ONE RECORD

```
SQL>INSER INTO STUDENT VALUES (‘KRISHNA’,’07-JUL-06’, 3000);
```

II. INSERT FEW FIELD ONE RECORD

```
SQL>INSERT INTO STUDENT (STNO, STNAME) VALUES (2, ‘RAVI’);
```

III. INSERT ALL FIELDS MANY RECORDS

```
SQL>INSER INTO STUDENT VALUES (&STNO, ‘&STNAME’, ‘&DOA’, &FEES);
```

IV. INSERT FEW FIELDS MANY RECORDS

```
SQL>INSER INTO STUDENT (STNAME, DOA) VALUES (‘&STNAME’, ‘&DOA’);
```

Note: In Oracle, whenever you type a command, if you make small mistakes, instead of retyping the entire command, you can use option called as EDITOR by typing ED. Whenever you are working in EDITOR, you should not use Semicolon (;)

Note: In Oracle, the immediately/previously-executed command can be re-executed by typing slash (/) and pressing Enter key.

SELECT: To see the Records present in a Table, SELECT command is used as show below: -

```
SQL> SELECT * FROM STUDENT;
```

UPDATE: To Change the existing record present in a Table, we use UPDATE command with the SET option.

SQL>UPDATE STUDENT SET FEES=11000 WHERE STNO=5;

SQL>UPDATE STUDENT SET STNO=6, FEES=15000 WHERE STNAME =' RAVI';

DELETE: When we want to delete ONE, FEW, ALL THE RECORDS present in a Table, following respective commands are used:-

SQL>DELETE FROM STUDENT WHERE STNO=4;

SQL>DELETE FROM STUDENT WHERE FEES>=10000;

DELETING ALL THE RECORDS:

SQL>DELETE FROM STUDENT;

To remove the Table itself, DROP Command is used: -

SQL>DROP TABLE STUDENT;

PROJECTION: When we want to see all the rows, but only few columns of a table, then instead of using *, we use the column names in the SELECT statement, which is called as Projection.

SQL<SELECT EMPNO,ENAME, JOB, SAL FROM EMP;

RESTRICTION: When we want to filter the rows of a Table in the query result by giving one or more condition with the help of WHERE clause, then it is called as RESTRICTION.

i. **QUERY WITH SINGLE CONDITION**
SQL>SELECT * FROM EMP WHERE SAL>=3000;
SQL>SELECT * FROM EMP WHERE JOB='MANAGER';

ii. **QUERIES BASED ON TWO CONDITIONS**
SQL>SELECT * FROM EMP WHERE JOB='CLERL' AND SAL>1200;

iii. **QUERY BASED ON THREE CONDITIONS**
SQL>SELECT * FROM EMP WHERE DEPTNO=10 AND JOB='SALESMAN' AND SAL>=1000;
SQL>SELECT * FROM EMP WHERE DEPTNO=10 OR DEPTNO=20;
SQL>SELECT * FROM EMP WHERE JOB! ='CLERK'
SQL>SELECT * FROM EMP WHERE NOT JOB='CLERK';

COLUMN ALIAS: Whenever query results are displayed, they will have the some heading as their column alias (or) the Expression given by you, which sometime may not look proper. In such case, column alias can be used, which has to be a single (or) it may contain an underscore, but it cannot be split in words.

```
SQL>SELECT ENAME, SAL*12  ANNUAL_SAL FROM EMP;
```

CONCATINATION: Whenever query results are displayed, there will be some gap between one column to another, which can be removed by using double pipe symb0ols called CONCATINATION.

```
SELECT EMPNAME || ‘WORKS AS’ || JOB REPORT FROM EMP;
SQL>SELECT ENAME || JOB FROM EMP;
```

LITERLS: When we want to replace certain information in every row of the result, but it is neither a column name, nor a column alias, then it is called as LITERAL.

```
SQL>SELECT ENAME || ‘WORKS AS’ || JOB || ‘IN DEPARTMENT’||
DEPTNO EMPLOYEE_DETAILS FROM EMP;
```

DISTINCT: When we want to remove the repetitive rows from the query result, we use DISTINCT.

```
SQL> SELECT DISTINCT DEPTNO FROM EMP;
```

NULL VALUE HANDLING: In oracle, whenever a value gets added to Null, it also becomes Null. To overcome this problem, we use NVL function as shown below:-

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

ASCENDING/ DESCENDING ORDER: When we want to display the query results in the Ascending Order; we used ORDER BY, in the Descending order by ‘DSESC’.

Note: The option order by should always be at the end of the SQL statement. Order by can be implemented on One or More columns at the same time.

```
SQL> SELECT * FROM EMP ORDER BY ENAME;
SQL> SELECT * FROM EMP ORDER BY SAL DESC;
SQL> SELECT DEPTNO, ENAME FROM EMP ORDER BY DEPTNO, ENAME;
```

SET OPTION: Following command sets the date in a different style for that entire session only.

SQL> ALTER SESSION SET NLS_DATE_FORMAT = ‘DD/MM/YYYY’.
SQL>SELECT * FROM EMP WHERE HIREDATE>=‘01/01/1981’ AND HIREDATE<=’31/12/1981’;

SQL OPERATORS: Following four options are called as SQL operators. They are:-

i. IN
ii. BETWEEN AND
iii. LIKE
iv. IS NULL

IN: When we want to select a few rows randomly from a large Table instead of using too many or operators, we should use one IN operator with a series of values given the brackets.

SQL>SELECT * FROM EMP WHERE EMP IN (7766, 7788, 7900);

BETWEEN AND: When we want to select few records inclusive of values given in the multiple conditions, then use operator BETWEEN AND.

SQL>SELECT * FROM EMP WHERE SAL BETWEEN 3000 AND 5000;

LIKE: When we want to search few records based on Character field with Wild Characters like *, then LIKE operator is used.

SQL>SELECT * FROM EMP WHERE ENAME LIKE ‘A%’;

Name to be displayed starting with ‘A’

SQL>SELECT * FROM EMP WHERE ENAME LIKE ‘&%’;
SQL>SELECT * FROM EMP WHERE ENAME LIKE ‘%S’;
SQL>SELECT * FROM EMP WHERE ENAME LIKE ‘%A%’;

IS NULL: Using this operator, we can fetch those rows where information is ‘Blank’

SQL>SELECT * FROM EMP WHERE COMM IS NULL;

DATA EXTRACTION FROM MORE THAN ONE TABLE:

JOINS: To link Table to Table in the query results, we use JOIN operator. They are four different types of JOINS in Oracle-

(i). EQUI JOIN
(ii). NON EQUI JOIN
(iii). OUTER JOIN

(iv). SELF JOIN
(v). NATURAL JOIN
(vi). CROSS JOIN
(vii). CARTESIAN JOIN

(i).**EQUI JOIN:** This join can be used only when there is at least One Common column between the Tables from where data is being extracted.

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

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

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

(ii).**NON EQUI JOIN:** When we want to fetch the data from more than one Table, but they don't have any common column between then, then we use NON EQUI JOIN with the help of operator NON EQUI JOIN.

SQL>SELECT EMPNO, ENAME, SAL, GRADE FROM EMP, SALGRADE WHERE SAL BETWEEN LOSAL AND HISAL;

Three Table Joining:
SQL>SELECT EMPNO, ENAME, DNAME, LOC, SAL, GRADE FROM EMP, DEPT, SALGRADE WHERE EMP.DEPTNO=DEPT.DEPTNO AND SAL BETWEEN LOSAL AND HISAL;

(iii).**OUTER JOIN:** If we want to extract all the rows in the result from both the Tables even though there is less information in one Table compared to another; then we use OUTER JOIN with the help of symbol (+).

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

(iv).**SELF JOIN:** When we want to join a Table back to the same Table again, then it is called as SELF JOIN.

In case of SELF JOIN, one Table name will be used Twice in the SELECT statement by using different aliases & they will be joined by giving a condition.

SQL>SELECT E.EMPNO EMPLOYEE_NO, E.ENAME EMPLOYEE_NAME M.EMPNO BOSS_NO M.ENAME BOSS_NAME FROM EMP E, EMP M WHERE E.MGR=M.EMP.NO;

UNION: When we want to club the results of two queries which look similar, then we use operator UNION. Simple union will bring Non repetitive rows; whereas UNION ALL will bring every row present in both the queries.

```
SQL>SELECT JOB FROM EMP WHERE DEPTNO=10 UNION
        (UNION ALL) is for displaying repetitive results.

SQL>SELECT JOB FROM EMP WHERE DEPT NO=20;
```

INTERSECT: If we use this operator, it will bring only those rows, which are present in both the queries.

```
SQL>    SELECT JOB FROM EMP WHERE DEPTNO=10
        INTERSECT
        SELECT JOB FROM EMP WHERE DEPTNO=20;
```

MINUS: This operator will give only those rows which are present in first query but not present in second query.

```
SQL>    SELECT JOB FROM EMP WHERE DEPTNO=10
        MINUS
        SELECT JOB FROM EMP WHERE DEPTNO-20;
```

SYSTEM DATE & DUAL: If few want to show the system, we use SYSDATE command. It is called as PESUDO column. It is not present in any Table, but you feel that it is present in every table. Hence, even the following command is valid but we don't use it because it gives too may rows in the result.

```
SQL> SELECT SYSDATE FROM EMP;
SQL> SELECT SYSDATE FROM DEPT;
```

As already told again, it leads to too more being shown in the result. To get the correct result we use the following command

```
SQL> SELECT SYSDATE FROM   DUAL;
```

DUAL: It is a special system Table given by the ORACLE which acts like a White Board or Notepad through which any Dummy results can be displayed and scratched OFF.

FUNCTIONS OF SQL:

In Oracle, there are 5 different types of Functions. They are:-

(i). CHARACTER FUNCTIONS

(ii). NUMERICAL FUNCTIONS
(iii). DATE FUNCTIONS
(iv). CONVERSION FUNCTIONS
(v). GROUP FUNCTIONS

Following 10 are different CHARACTER FUNCTIONS:

(i).LOWER: This command is used to convert Capital Letters to Small cases.
SQL> SELECT LOWER (ENAME) FROM EMP;
SQL> SELECT LOWER (‘RAILWAY’) FROM DUAL;

(ii).UPPER: This command is used to convert Small Letters to Capital cases.
SQL> SELECT UPPER (‘railway’) FROM DUAL;

(iii).INICAP: This function converts the first Letter to Capital, rest to Small Letters.
SQL> SELECT INICAP (ENAME) FROM EMP;

(iv).RPAD: Whenever we want to fill up the right side blank spaces with special characters like * to avoid manipulation, we use RPAD.

SQL> SELECT RPAD (ENAME, 10 ‘*’) FROM EMP;

(v).LPAD: This command is used to fill up the Left side blank spaces especially for Numerical fields.

SQL> SELECT LPAD (SAL, 7, “#”) FROM EMP;

(vi).LENGTH: This command is used to get the length of the character length.

SQL> SELECT ENAME, LENGTH (ENAME)

(vii).INSTR: This command is called as In string Function. It is used to check whether a particular character is present in a string or not and if present, what is its position in the sting.

SQL> SELECT ENAME, INSTR (ENAME ‘A’) FROM EMP;

(viii).SUBSTR: It is popularly called as Substring function is used to extract one or few characters from a given string. Function requires two numbers as parameters. The first number represents from where to start, the second number represents how many characters to be removed.

SQL> SELECT SUBSTR (‘INDIAN RAILWAYS’, 3,4) FROM DUAL;

(ix).REPLACE: When we want to replace one string with another temporarily only in the result, then this function is used.

SQL> SELECT JOB REPLACE (JOB, ‘SALESMAN’,’MARKETING’) FROM EMP;

(x).DECODE: As if conditions are not possible with the SQL statements, we use DECODE instead of IF. In the function, we can give one General parameter, which acts like if condition.

SQL> SELECT ENAME, JOB, SAL, DECODE (JOB, ‘CLERK’, SAL * .1, ‘SALESMAN’, SAL * .2, SAL *.3) HRA FROM EMP;

SQL> SELECT ENAME, DECODE (COMM, NULL, ‘COMMISSION IS NULL, 0, ‘COMMISSION IS ZERO’) COMM FROM EMP;

NUMERICAL FUNCTIONS: There are 8 types of Numerical functions. They are:-

(i).ROUND: It is used for rounding the Decimal part of the specific position.
SQL> SELECT ROUND (200.2555) FROM DUAL; Result=200.25

(ii).CEIL: Irrespective of the Decimal Number if we want a Decimal number always rounded to the next nearest Integer, then we use Ceil.

SQL> SELECT CEIL (200.15)FROM DUAL; Result=200

(iii).FLOOR: It is opposite to the Ceil Function means; irrespective of the Decimal Numbers, it will always be rounded to the Current Integer.

SQL> SELECT FLOOR (200.95) FROM DUAL; Result=200

(iv).POWER: It is used to get the result of a square of a given number.

SQL> SELECT POWER (10, 3) FROM DUAL;

(v).SQRT: It is used to get the square root of the Number.

SQL> SELECT SQRT (25) FROM DUAL; Result =5

(vi).SIGN: This will give the result in terms of 1 or -1 representing whether the result achieved is Positive/Negative after arithmetical expression is evaluated.

SQL> SELECT COMM, SIGN (-COMM) FROM EMP;

(vii).ABS: This function called as Absolute function is used to convert Negative Numbers to Positive.

SQL> SELECT ABS (-500 *2) FROM DUAL; Result=1000

(viii).MOD: This function will give us the Reminder after one number is divided by another.
SQL> SELECT MOD (100,40) FROM DUAL; Result =20

CONVERSION FUNCTIONS: Following two options are called as Conversions function. They are:-

(i).TO_CHAR: Using this option with the help of system date, we can get different styles of system date like year only, Month only, Short date, Long date, Even time.

SQL> SELECT TO_CHAR (SYSDATE, ‘YYYY’) FROM DUAL; Result = 2010
SQL> SELECT TO_CHAR (SYSDATE, ‘MM’) FROM DUAL; Result = 02
SQL> SELECT TO_CHAR (SYSDATE, ‘MONTH’) FROM DUAL; Result = July
SQL> SELECT TO_CHAR (SYSDATE, ’MON’) FROM DUAL; Result = Jul
SQL> SELECT TO_CHAR (SYSDATE, ’DY’) FROM DUAL; Result = Fri
SQL> SELECT TO_CHAR (SYSDATE, ’DAY’) FROM DUAL; Result = Friday
SQL> SELECT TO_CHAR (SYSDATE, ‘HH.MI.SS’) FROM DUAL; Result = 12.03.31

(ii).TO_DATE: This function will convert the CHARACTER to date; it is very helpful when we want to insert the data to the date field in different styles than the normal format of Oracle.

SQL> INSERT INTO EMP (EMPNO, ENAME, HIREDATE) VALUES (1001, ‘RICHARD’, TO_DATE (‘13/02/1998’, ‘DD/MM/YYYY’));

DATE FUNCTIONS: Following four options are called as Date functions. They are:-

(i).MONTHS_BETWEEN: Using this function, we can get the number of months between the two given dates.

SQL> SELECT ENAME, ROUND (MONTHS_BETWEEN (SYSDATE, HIREDATE) NO_OF_MONTHS) FROM EMP;

SQL> SELECT ENAME, ROUND (MONTHS_BETWEEN (SYSDATE, HIREDATE)/12 NO_OF_YEARS) FROM EMP;

(ii).ADD_MONTHS: Using this function, we can get a future date (or) previous by adding specific number of months positively/negatively.

SQL> SELECT ADD_MONTHS (SYSDATE, 6) FROM DUAL;
SQL> SELECT ADD_MONTHS (SYSDATE, -6) FROM DUAL;

(iii).NEXT_DAY: This function will give the date of the immediately coming day mentioned within the brackets.

SQL> SELECT NEXT_DAY (SYSDATE, 'FRIDAY') FROM DUAL;

(iv).LAST_DAY: This function will give the last date of any month like 31, 30 even 28 in case of February.

SQL> SELECT LAST_DAY (SYSDATE) FROM DUAL;

GROUP FUNCTIONS: Following five options are called as group Functions because they look meaningful when implemented on an entire Table or atl3east a set of rows. They are:-

(i). **AVG:** This function gives the Average of a Numerical field.
SQL> SELECT AVG (SAL) FROM EMP;

(ii). **MIN:** This function gives the lowest value in a Numerical field.
SQL> SELECT MIN (SAL) FROM EMP WHERE DEPTNO=10 OR DEPTNO=20;

(iii). **MAX:** This function gives the highest value in a Numerical field.
SQL> SELECT MAX (SAL) FROM EMP WHERE JOB=('MANAGER');

(iv). **COUNT:** This function is used to count the Number of rows. It will not include Null values when counting * is used as parameter with the count function. However, if you fell that Null valves are present in the field instead of u sing *, you can use the field name also

SQL> SELECT COUNT (*) FROM EMP;
SQL> SELECT COUNT (COMM) FROM EMP;

(v). **GROUP BY:** When we want to implement a Group function or more than One group at the same time, then GROUP BY clause should be used.

SQL> SELECT DEPTNO, MIN(SAL) FROM EMP GROUP BY DEPTNO;
SQL> SELECT JOB, MAX(SAL) FROM EMP GROUP BY JOB;

SQL> SELECT JOB, MAX(SAL) FROM EMP WHERE JOB='PRESIDENT' GROUP BY JOB; (Particular Column elimination)

(vi). **HAVING:** If where clause is used to filter the rows present in Table, HAVING clause is used to filter the rows arrived out of a Group Result.

SQL> SELECT JOB, MAX(SAL) FROM EMP GROUP BY JOB HAVING COUNT (JOB)>2;

SQL> SELECT DEPTNO, COUNT(DEPTNO) FROM EMP GROUP BY DEPTNO;

SQL> SELECT COUNT (EMPNO) FROM EMP GROUP BY EMPNO.HAVING COUNT (EMPNO)>1;

SUB QUERIES:

Whenever we want to extract the values of a Group function with single rows or when we want to extract the known values after extracting the unknown values then we use sub queries.

In case of sub queries, there will be minimum two queries, one called as Main or Outer query and other called as Inner Query.

For one main query, there can be a maximum of 255 inner queries. But usually, we use two or at the maximum three inner queries. In case of sub queries, always the inner most query gets executed first. Its result will be secretly passed to the outer query for further execution.

SQL> SELECT ENAME, JOB, SAL FROM EMP WHERE SAL=(SELECT MIN (SAL) FROM EMP);
SQL> SELECT ENAME, JOB, SAL FROM EMP WHERE (ENAME!='SCOTT');

SQL> SELECT ENAME, JOB FROM EMP WHERE (ENAME!='SCOTT' AND JOB=(SELECT JOB FROM EMP WHERE ENAME ='SCOTT') ;

SQL> SELECT ENAME, JOB, MGR FROM EMP WHERE MGR=(SELECT EMPNO FROM EMP WHERE JOB== 'PRESIDENT');

SQL> SELECT MAX (SSAL) FROM EMP WHERE SAL<(SELECT MAX(MAX(SAL) FROM EMP);

<u>DIFFERENT TYPES OF QUERIES:</u>

Display ENAME, SAL, DNAME, & JOB of all those who are getting salary more than the highest salary of any clerk.

SQL> SELECT MAX (SAL) FROM EMP WHERE JOB='CLERK';

SQL> SLELECT ENAME, JOB, SAL, DNAME, LOC FROM EMP, DEPT WHRE EMP.DEPTNO=DEPT.DEPTNO AND SAL>(SELECT MAX (SL) FROM EMP WHERE JOB='CLERK');

Display ENAME, SAL, DEPTNO & their LOC, their Grade for all those who are getting salary more than any persons salary working in DEPTNO=20.

```
SQL> SLELECT ENAME, JOB DEPTNO, LOC, GRADE, SAL FROM EMP, DEPT,
     SALGRADE WHERE EMP.DEPTNO=DEPT.DEPTNO AND SAL BETWEEN
     LOSAL AND HISAL AND SAL> (SELECT MAX (SAL) FROM EMP WHERE
     ENAME IN ('ALLEN', ADAMS, 'JAMES');
```

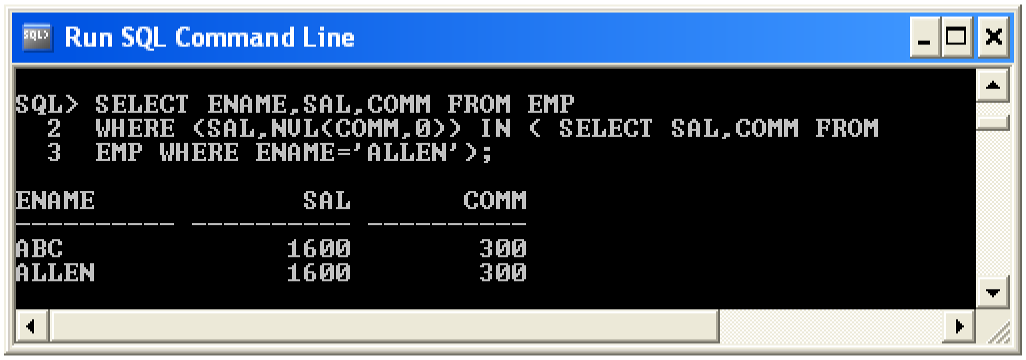
Create a query which will display ENAME, SAL DEPTNO & DNAME of those who are getting salary more than the highest salary of any person working in Research Department.

```
SQL> SLELECT ENAME, SAL, EMP.DEPTNO, DNAME FROM EMP, DEPT
     WHERE EMP.DEPTNO=DEPT.DEPTNO AND SAL > (SELECT MAX (SAL)
     FROM EMP WHERE DEPTNO = (SELECT DEPTNO FROM DEPT WHERE
     DNAME='RESEARCH'));
```

Display the DEPTNO AND AVG SALARY, which is getting the highest average salary.

```
SQL> SELECT DEPTNO, AVG (SAL) FROM EMP GROUP BY DEPTNO HAVING
AVG (SAL) = (SELECT MAX (AVG (SAL)) FROM EMP GROUP BY DEPTNO);
```

Display the ENAME, SALARY, COMM of those whose both SALARY and COMMISSION matches that of ALLEN.



```
SQL> SELECT ENAME,SAL,COMM FROM EMP
  2  WHERE (SAL,NVL(COMM,0)) IN ( SELECT SAL,COMM FROM
  3  EMP WHERE ENAME='ALLEN');

ENAME             SAL       COMM
---------- ---------- ----------
ABC              1600        300
ALLEN            1600        300
```

CORRELATED QUERIES:

A correlative query works exactly opposite of a normal sub query means; In this case outer query will be executed first and its result will be passed to the inner query for further execution.

One more difference between a sub query and a correlated query is in case of sub query, the inner query will be executed only once; whereas in case of correlated query, the inner query will be executed so many times depending upon the candidate rows selected by the outer query.

Following are the steps for the way a correlated query gets executed: -

i. Select all the candidate rows by using the outer query;
ii. Send One by one candidate row to the inner query;
iii. Execute the inner query by using one of the key columns present in the outer query;
iv. Compare the result brought by the inner query with the candidate rows values to select or discard the candidate row.
v. All the above steps will be executed till all the candidate rows are over.

Note: It is very difficult to distinguish between a sub query and a correlated query, but one method to identify could be by looking at the **Table alias** in the inner queries where clause;

```
SQL> SELECT E.ENAME, E.SAL, E.DEPTNO FROM EMP E
        WHERE SAL>(SELECT AVG (SAL) FROM EMP
        WHERE DEPTNO=E.DEPTNO);
```

Following query will display the DEPTNO, DNAME, & LOC of those DEPTNO, which are present in DEPT TABLE but not present in EMP TABLE.

```
SQL> SELECT DEPTNO, DNAME, LOC FROM DEPT D WHERE NOT EXISTS
(SELECT * FROM EMP WHERE DEPTNO=D.DEPTNO);
```

SUBSTITUTIONAL VARAIBLES:

Whenever we want to pass some temporary information when the query is getting executed, then it can be done with the help of substitution variables. These variables are temporary for that session. These variables can be declared by using ‘&” sign or DEFINE Command.

```
SQL> SELECT * FROM EMP WHERE JOB=’&JOB’,
SQL> DEFINE AL= (SAL +NVL (COMM, 0)) *12;
```

TRANSACTION CONTROL LANGUAGE: COMMIT and ROLLBACK are the two important sections of TCL.

COMMIT=SAVE; ROLLBACK UNDO

ALL THE INSERT, UPDATE & DELETE OPERATIONS (DML), WHICH WE DO DURING A SESSION ARE TEMPORARY ONLY IN THE MEMORY TILL WE GIVE COMMIT. EVEN WHEN THE SESSION IS SHUT DOWN PROPERLY OR THE DATABASE IS SHUTDOWN PROPERLY BY THE ADMINISTRATOR ALL THE DML OPERATIONS DONE BY US AUTOMATICALY GET COMMITTED.

But still **COMMIT** should be done as early as possible for the following reasons:-

a) To avoid any technical/power failure this could be lead to Loss of Data.
b) To make sure that every one present on the network will see the latest & correct data.
c) To avoid unwanted rollbacks.
d) To stop unknowingly troubling on the network.

ROLL BACK: It should be implemented before COMMIT. Once COMMIT is issued and transactions are made permanent on the hard disk, you cannot ROLLBACK back.

ADDITIONAL COMMANDS IN SQL FOR REPORTING AND FORMATING

SQL> TTITLE/BTITLE: using these two options, we can create Headings before the query results & after the query results.

SQL> SET LINESIZE 150. Maximum 180

SQL> TTITLE “INDIAN RAILWAYS”; (Date, Heading, Page No will be displayed automatically.)

SQL> TTITLE ‘INDIAN RAILWAYS’|’SOUTH WESTERN RAILWAY’|’HUBLI’;

SQL> TTITLE LEFT ‘|INDIAN RAILWAYS’|’SOUTH WESTERN RAILWAY’|’HUBLI’ SKIP2;

For not displaying/displaying system commands

SQL> SET FEEDBACK OFF/ON: By giving this command, we can suppress the system messages given after the SQL command is executed.

```
SQL> SET HEADING OFF;/ON;
SQL> BTITLE ‘CONFIDENTIAL’|THANK U’;

SQL> SELECT DEPTNO, ENAME, JOB, SAL FROM EMP ORDER BY DEPTNO SKIP2;

SQL> BREAK ON DEPTNO;

SQL> COMPUTE SUM OF SAL ON DEPTNO;

SQL>SET TIME ON/OFF;

SQL>SET TIMING ON/OFF: It will display the time taken to execute a query.

SQL>SET COLSEP ‘*’: Column can be separated by any symbol.

SQL>SET UNDERLINE ‘*’;
```

Any query results if you want to send a text file Keep SPOOLING OPTION ON.

Print out cannot be taken directly from SQL prompt.

DATA DEFINITION LANGUAGE COMMANDS:

CREATE, ALTER, DROP AND TRUNCATE, are the four major sections of DDL.

Using CREATE command; we can create any database object including Tables, Views, Synonyms, Sequence, Index.

Oracle allows Constraints to be defined at Table Level & at Column Level enforce the correct date entry from the end user.

CLASSIFICATION OF CONSTRAINTS:

In Oracle, Constraints can be classified either at the column level or at the Table level. Column level constraints will be defined immediately after the column names and their data types.

Table level constraints will be defined at the end after the column names are over.

Usually composite primary key or foreign keys are declared as Table constraints and rest all will be column constraints.

Naming of Constraints: In Oracle whenever we create a constraints either at the column level or at the Table level it compulsorily requires a Name, because the same name will be stored in the oracle's dictionary and it is better if the proper name is given by the user himself for any better future reference. If we don't give the name, then Oracle will automatically generate a number with the Prefix SYS and stores it as a Name for total constraint. This name could be very difficult to identify whenever we see it by giving the following COMMAND.

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

TABLE NAME SHOULD BE IN CAPITAL ONLY.

TYPES OF CONSTRAINTS: In Oracle, there are 5 different types of constraints. They are:-

- **(i).** NOT NULL
- **(ii).** UNIQUE
- **(iii).** PRIMARY KEY
- **(iv).** FOREIGN KEY
- **(v).** CHECK

(i). **NOT NULL:** If implemented on one or more columns, this constraint will not allow Null or Blank values to that column. However, it does allow repetitive values.

(ii). **UNIQUE:** When implemented on one or more column, this constraint does not allow repetitive values. However, it does allow NULL values.

(iii). **PRIMRY KEY:** It is a combination of NOT NULL and UNIQUE. It means when implemented on a particular Table, this column will neither allow NULL VALUE NOR ALLOW REPETITIVE VALUES. However, per table, ONLY ONE PRIMARY KEY IS ALLOWED.

However, if the situation demands, then we can create a composite primary key at the Table level by combining the columns. Maximum up to 7 columns we can combine.

(iv). **FOREIGN KEY:** When we want to link one Table with another by using a common column; then it has to be implemented by using rule FOREIGN KEY like COMPOSITE PRIMARY KEY. Usually, even FOREIGN KEY will be at the Table Level.

(v). **CHECK:** Whenever we want to Force certain special types of rules like ENAMES, entered must be CAPITAL ONLY job should be CLERK, MANAGER, SALESMAN ONLY or SLAARY should be less than 5000 only, and then it will be implemented by CHECK constraint.

REFERENCE: When we want to link one column with another within the same Table, like BOSSNO & EMPNO. Columns, then it has to be implemented by using the REFERENCE, but not FOREIGN KEY.

ON DELETE CASCADE: By default whenever a child record exists, we will not be able to remove the Parent Key from the Parent Table. However, while creating the Foreign key, if we had given option ON DELETE CASCADE, then it allows us to remove the Parent Key even when Child records exists. But the moment, the primary Key is Deleted, All the corresponding Child records automatically get deleted.

DEFAULT: Even though officially not called, many people consider this as a constraint only. While creating the Table to a particular column, if we set certain DEFAULT VALUE If the END user omits its value in the INSERT statement, it will automatically use that DEFAULT value.

TABLE NAMING RULES: Whenever a Table is created even to give a Name that rule. There are certain rules:

1) A Table name should start with Alphabet only and can be followed by Characters.
2) Even though officially, it is allowed, Oracle requests you not to use Special Characters like ‘*’, ‘?’Etc., in the Table name.
3) A Table name can be maximum 35 Characters in Length
4) A Table name cannot use SQL Reserve word like SELECT, DELETE etc.,

TITLE TAB: Even though the word TAB looks like a SQL Reserve word, we can create a Table under the Tab. After creating the Table, we can insert the rows also as usual. Hence, if we give command SQL> SELECT * FROM TAB, it will give the Table contents, but not the original Table listing.

In such situation, we can use/give the following command:-

SQL> SELECT * FROM CAT; If a Table exists under the name CAT also then the only way to see the list of the Tables is following DBA command.

```
SQL> SELECT TABLE_NAME FROM USER_TABLES;
```

CREATING MASTER TABLE:

```
SQL> CREATE TABLE DPT (
        DEPTNO NUMBER (2) CONSTRAINT DP_PKEY PRIMARY KEY,
        DNAME VARCHAR2 (15) CONSTRAINT DP_UNQ UNIQUE,
        LOC VARCHAR2 (15)  NOT NULL);

SQL> CREATE TABLE AMP (
        EMPNO NUMBER (4) CONSTRAINT EMP_PKEY PRIMARY KEY,
        ENAME VARCHAR2 (15) NOT NULL,
        JOB VARCHAR2 (15),
        BOSS_NO NUMBER (4)
        CONSTRAINT AMP_REF REFERENCES AMP (EMPNO),
        DOJ DATE,
        SAL NUMBER (7),
        DEPTNO NUMBER (2),
       CONSTRAINT EMP_FKEY FOREIGN KEY (DEPTNO) REFERENCES DPT
       (DEPTNO));

SQL> SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE FROM
        USER_ CONSTRAINT WHERE TABLE_NAME='DPT';
```

This command i.e. USER_CONSTRAINTS will stores all the constraints with their name and type of all the tables create in all the tables that is of way it requires the tables name to specify which table constraints w have to display.

CREATION OF DUPLICATE TABLES:

```
SQL> CREATE TABLE EMP1 AS SELECT * FROM EMP;
```
Only NOT NULL constraint of the Original Table gets passed to the Duplicate Table also.

DIFFERENCE BETWEEN TRUNCATE & DELETE:

Truncate Command is equal to deleting all the records from the Table plus committing it. However, there are two major differences between DELTE AND TRUNCATE. In case if delete we can use ROLLBACK but in case of TRUNCATE, it is not possible. Rows will be deleted permanently.

Also in case of DELETE, rows will be removed only logically; means the space occupied by those rows will remain with the Table only; whereas in case of TRUNCATE, the rows will be removed permanently and the space occupied by those rows will be released to the Hard disk.

```
SQL> TRUNCATE TABLE EMP1;
```

DATA CONTROL LANGUAGE:

Whenever we create a new user account by going to SYSTEM/MANAGER, then to give the permissions for doing the work in the new area, we use command GRANT and to take back the permissions, we use command REVOKE.

```
SQL> CREATE USER RAILWAY IDENTIFED BY HUBLI;

SQL> GRANT CREATE  SESSION TO RAILWAY;

SQL> GRANT CREATE  TABLE TO RAILWAY;


SQL> GRANT RESOURCE TO RAILWAY;
```
(Allowing the user to INSERT, UPDATE or DELETE)

```
SQL> CONNECT TO RAILWAY/HUBLI;


SQL> REVOKE CREATE TABLE FROM RAILWAY;

SQL> REVOKE CREATE SESSION FROM RAILWAY;

SQL>DROP USER RAILWAY CASCADE;
```

GENERAL SYNTAX
```
SQL> CREATE USER<USERNAME> IDENTIFIED BY <PASSWORD>;
```

OTHER DATABASE OBJECTS: VIEWS, SYNONYMS, & SEQUENCE

VIEWS:

1) A view is a virtual Table which does not exist, but appears to the end user as if it exists like a mirror image.

2) A view is an object through which partial data can be seen (or) changed.

3) A view is referred to a moon because it derives the data from the base table and presents us with the data.

4) A view can be created only by using SELECT statement.

5) On a view, we can do any DML operation, i.e., INSERT, UPDATE, DELETE exactly in the same way ado it on a Table; But whenever the operations are done on a view, it goes to the Table only because view is not having anything.

CREATING DUPLICATE TABLE WITHOUT RECORDS:

SQL> CRATE TABLE EMP1 AS SELECT * FROM EMP WHERE 1=2
(GIVE COMMAND WHICH IS NOT EXECUTEABLE FOR THE ABOVE SUBJECT)

COPYING THE RECORDS TO A NEW TABLE:

SQL> INSERT INTO EMP1 (SELECT * FROM EMP);

RENAMING A TABLE:

SQL> RENAME EMP1 TO EMPLOYEE

ALTER: By using the ALTER command of DLL which as got 3 sub options like ADD, MODIFY AND DROP, we can change the structure of the existing table.

ADD: By using ADD option of ALTER TABLE, we can add new columns as well as NEW CONSTRAINT to the existing Table.

SQL> ALTER TABLE AMP ADD (COMM NUMBER (5));
SQL> ALTER TABLE AMP ADD (CHECK (JOB IN (‘CLERK’, ‘SALESMAN’, ‘MANAGER’)));
SQL> ALTER TABLE AMP (CHECK (SAL <=7000));
SQL> ALTER TABLE AMP ADD (CHECK (ENAME=UPPER (ENAME)));

MODIFY: By using the MODIFY option of the ALTER TABLE; we can increase/decrease the length of an existing column. Decrement is possible if column is not having any data

DROP: By using the DROP action of ALTER COLUMN we can drop a column (or) we can drop a CONSTRINT.

```
SQL> ALTER TABLE AMP COLUMN COMM;
```

DROP: By using the DROP command OF DDL, we can drop any database object including the TABLES.

```
SQL> DROP TABLE EMPLOYEE;
```

CASCADE CONSTRAINTS: When the CHILD TABLE exists and if you want to forcibly DROP the PARENT TABLE, then use CASCADE CONSTRAINS option.

```
SQL> DROP TA\BLE DPT CASCADE CONSTRAINTS;
SQL> ALTER TABLE AMP DROP CONSTRAINT AMP-REF;
```

APPLICATION OF USE:
Views will be permanent. It is not temporary for a session.

SECURITY:
Views are basically used as a part of Security, means; in many organizations, the end user (data entry operators) will never be give3n Original Tables & all the data entry will be done with the help of views only. But Data Base Administrator will be able to see everything because all the operations done by the different users will come to the same Table.

PROVIDING EXTRA INFORMATION:
Views can be used to provide extra information which does not exists even in the Original Table.

SUPPERESSING THE COMPLEX QUERIES:
Views are also used to suppress the compress/complicated queries like Sub queries/Correlated queries etc.,
Operations done on Views will be permanent not for temporary.

```
SQL> CREAT6E VIEW D10 AS SELECT * FROM EMP WHERE DEPT NO=10
WITH CHECK OPTION;
```

```
SQL> DROP VIEW D10;
```

```
SQL> CREATE VIEW ABC (EMPNO, ENAME, ANNUAL_INCOME AS SELECT
EMPNO, ENAME (SAL+NVL (COMM, 0)) *12 FROM EMP;
```

```
SQL> CREATE VIEW ABC AS SELECT EMPNO, ENME, DNAME, LOC FROM
EMP, DEPT WHERE EMP.DEPTNO=DEPT.DEPTNO;
```

SYNONYMS:

When we want to access the object belonging to another user, then we have to give the user name & object name every time we access the object. Instead of that, we can create a synonym also.

```
SQL> SELECT * FROM SCOTT.EMP;
SQL> CREATE SYNONYM STAFF FOR SCOTT.EMP;
```

SEQUENCE:

When we generate a series of numbers into a database object which they can be utilized to enter or insert values into a Table specially having PRIMARY KEY, it is useful. We can find out the value present in the sequence by using NEXTVAL option.

```
SQL> CREATE SEQUENCE STISH INCREMENT BY 1 START WITH
        MAXVALUE100;
SQL> INSERT INTO STAFF VALUES (SATISH, ‘NEXTVAL’, ‘KIRAN’,
        ‘MANAGER’);
```

ROW ID:

Every Oracle Table created by you will have an internal primary key known as ROWID, which will represent the uniqueness of that row. Hence, if we want to do a operation on a repetitive value, we can use ROWID.

```
SQL> SELECT * FROM STUDENT;
SQL> SELECT ROWID, STNAME FROM STUDENT;
SQL> DELETE FROM STUDENT WHERE ROWID=’-------------‘;
```

INDEX:

Oracle Corporation strongly recommends the usage of INDEXES because it will help us in speeding up the query results. INDEXES are very useful when we are searching for a value in a repetitive date column like ENAME it is assumed that a index will speed up query results nearly by 30% especially on those Tables which have 1000 or more Number of rows.

Whenever we create an index, a separate database supportive object will be created which will sort the information on which index was created, it will be stored in terms of ROWID. Our work is to only create an index. Oracle knows it very well when to use it. Whenever we give a query, if an index is present on that query, it first goes to the index, collects that ROWID and searches for the same row in the EMP table.

```
SQL> CREATE INDEX SATISH ON EMP 9ENAME);
SQL> CREATE INDEX SATISH ON (EMP (DEPTNO, ENAME));
```

Note: There is no need to create indexes on Primary key or Unique column because they are automatically indexed.

NEW FEATURES OF 8i AND 9i ORACLE

Oracle 8i has got some new features compared to its previous versions. They are:-

(i). New Datatypes (LOB)
(ii). User Defined datatypes
(iii). Partitioned Tables
(iv). Instead of Triggers
(v). DDL Triggers
(vi). Global Temporary table

Oracle 9i has got some new features compared to its previous versions. They are:-

(i). New type of SELECT STATEMENTS (JOINS)
(ii). New data type called TIME STAND
(iii). Flash back make of the data base.

(i). NEW DATATYPES (LOB): Oracle 8i introduced a new set of data types namely LOBS (Large OBJECTS). They are classified into 3 categories. They are:- CLOB, BLOB & BFILE.

The one common feature of all these 3 datatypes is they all have the capacity to hold value up to 4GB per row/per column. CLOB is called as INLINE datatype means; Data can be inserted by using normal INSERT command; whereas for the other two which are basically used to store the graphics into database Table, we use special PL/SQL program with the help of DBMS LOB package.

```
SQL> CREATE TABLE SATISH (ENAME VARCHAR (10), REMARKS CLOB);
```

(ii). USER DEFINED DATATYPES:

When similar kind of field names are getting repeated across different Tables instead of repeating them again & again, we can create one object initially consisting of those field names & then we can call it wherever we require that.

Step-1: Create an Object:

```
SQL> CREATE OR REPLACE TUPE ADD AS OBJECT (
        HNO NUMBER (6),
        STREET VARCHAR2(15),
        CITY VARCHAR2(15),
        PIN NUMBER (6));
```

Step-2: Create a Table using own datatype:

```
SQL> CREATE TABLE STUDENT (
        STNO NUMBER (2),
        STNAME VARCHAR2(10),
        STADDRESS ADD);
```

Step-3: Insert the data to the Table:

```
SQL> INSERT INTO STUDENT VALUES (1, ‘KRISHNA’, ADD (17, ‘VIDYNAGAR’,
‘HUBLI’, 580020));
```

(iii).INSTEAD OF TRIGGER: Whenever a view is created based on more than one Table then we cannot do any insert (or) update (or) delete operations because to which Table the data reach. Oracle will not know, hence an operation needs to be.. we have to create one program called INSTEAD OF TRIGGER, then only it is possible to do operation on a view which is having more than one Tables information.

```
SQL> CREATE OR REPLACE TRIGGER ABC INSTEAD OF INSERT ON EMP;

DECLARE
SQL> DNO NUMBER (5);

BEGIN
SQL> SELECT DEPTNO INTO DNO FROM DEPT
        WHERE DEPTNO= NEW .DEPTNO;

EXCEPTION
WHEN NO_DATA_FOUND THEN INSERT
VALUES (NEW DEPTNO, NEW DNAME);
INSET NTO EMP (EMPNO, ENAME)
VALUES (NEW.EMPNO, NEW.ENAME);

END;

SQL> @ KR1
```

(iv).DDL TRIGGERS: By default it you want to stop some from dropping the table, then we have to control by using DBA commands, but Oracle 8i onwards, it is possible that even…We can create a DEL Trigger through which we can stop from creating or dropping a Table etc.,

```
SQL> CREATE OR REPLACE TRIGGER NDP BEFORE DROP ON
SCOTT.SCHEMA

BEGIN
OF DICTIONARY_OBJ NAME ‘EMP’ THEN
RAISE_APPLICATION ERROR (20010 ‘CANT DROP EMP TABLE);

END IF;
END
```

SCHEMA is nothing but collection of all the objects like Tables, View, and Synonyms belonging to a user.

(v).TEMPORARY TABLES: Whenever we want to create the Table which will hold the data temporary only for the session only even though you may commit(or) make a proper exit from the session, then we should create a temporary table. The temporary table rows will automatically get deleted as soon as the session is closed in other words, it will act like a fresh Table without any records every time you open a new session.

```
SQL> CREATE GLOBAL TEMPORARY TABLE STAFF (
        ENO NUMBER (2), ENAME VARCHAR2(10)
        SAL NUMBER (6) ON COMMITPRESSRUE ROWS);
```