UNIT : 2

MANAGING TABLES AND DATA

1. EXPLAIN CREATE COMMAND WITH EXAMPLE

- This command is used to create table.
- The CREATE table command is useful a new table in your present user.
- By using, we have to specify fieldname, data type and size.
- Each of the fields should be separated by comma.
- the SQL statement will be terminated with a semicolon.
- Create tables to store data by executing the SQL CREATE TABLE statement.
- This statement is one of the data definition language (DDL) statements.
- These statements have an immediate effect on the database, and they also record information in the data dictionary.

RULES FOR CREATING Table names and column names:

- Must begin with a letter
- Must be 1 to 30 characters long
- Must contain only A–Z, a–z, 0–9, _, $, and #
- Must not duplicate the name of another object owned by the same user
- Must not be an Oracle Server reserved word

SYNTAX

CREATE TABLE <<table name>>

(<<column name>><Data type><Size> [CONSTRAINT], <<column name>><Data Type><Size> [CONSTRAINT],……);

In CREATE TABLE Statement:

<Table name>	Specify name of the table
<Column name>	Specify fieldname
<data type>	Specify the type of data that field hold
<size>	Specify length for field
[CONSTRAINT]	Specify name of constraint if any

EXAMPLE

CREATE TABLE student **(**rollno number(3),first_name varchar(20),last_name varchar(20), address varchar2(30),city varchar(10),pincode number(10)**)**;

2. EXPLAIN DDL STATEMENTS WITH SUITABLE EXAMPLE.

- DDL stands for Data Definition Language.
- The DDL commands are the set of SQL commands that can be used to create, modify or delete the database structure but not data.
  - **Following are the list of DDL commands:**
    a. **CREATE TABLE**
    b. **ALTER TABLE**
    c. **DROP TABLE**
    d. **RENAME**
    e. **TRUNCATE TABLE**

CREATE:

- This command is used to create table.

ALTER TABLE:

- By using alter table command we can modify the structure of the database.
- Using ALTER TABLE statement to:
- Add a new column
- Modify an existing column
- Define a default value for the new column
- Drop a column

SYNTAX

ADDING NEW COLUMNS

ALTER TABLE <table name> ADD (new<column name> data type (size));

MODIFYING EXISTING COLUMNS

ALTER TABLE <table name> MODIFY (<column name> new data type(new size));

DROPING EXISTING COLUMNS

ALTER TABLE <table name> DROP(<column name>);

In the syntax:

<Table name>	Specify name of the table
<Column name>	Specify fieldname
<data type>	Specify the type of data that field hold
<size>	Specify length for field

EXAMPLE

ADDING NEW COLUMNS

```
ALTER TABLE student ADD (country VARCHAR (10));
```

MODIFYING EXISTING COLUMNS

```
ALTER TABLE student MODIFY (last_name VARCHAR2 (30));
```

DROPING EXISTING COLUMNS

```
ALTER TABLE dept DROP(country);
```

DROP TABLE:

- This command is removes the definition of a table from oracle database.
- When you drop a table, the database loses all the data in the table and all the indexes associated with it

SYNTAX

DROP TABLE <table name>;

In syntax

<Table name>	Specify name of the table

EXAMPLE

DROP TABLE student;

RENAME:

- This command is used to rename a table, view, sequence, or synonym.

SYNTAX

RENAME <old name _table/view/synonym> TO <new name _table/view/synonym>;

In the syntax:

<old name>	Specify the old name of the table, view, sequence, or synonym
<new name>	Specify the new name of the table, view, sequence, or synonym

EXAMPLE

RENAME student TO stud;

TRUNCATE TABLE:

- This command is removes all the row of table to release the storage space used by that table.
- When you drop a table, the database loses all the data in the table and all the indexes associated with it

SYNTAX

TRUNCATE TABLE <table_name>;

In syntax

<Table name>	Specify name of the table

EXAMPLE

TRUNCATE TABLE student ;

3. EXPLAIN DML STATEMENTS WITH SUITABLE EXAMPLE.

- DML stands for Data Manipulation Language.
- The DML commands are the set of SQL commands that can be used to insert, modify or delete the records of the table.
- Following are the list of DDL commands:
  - **INSERT**
  - **UPDATE**
  - **DELETE**
  - **SELECT**

INSERT:

- This command is used to insert a row in table.
- With the help of Insert into command we can enter the data into the table one record at a time.

SYNTAX

INSERT INTO <table name> (<column name>, <column name>) VALUES (expression, expression);

- In syntax

<Table name>	Specify name of the table in which you want to insert records.
<Column name>	Specify fieldname in which you want to enter data
expression	Specify the values that you want to insert in your table

EXAMPLE

INSERT INTO student VALUES (1, 'avni','bhatt','1 narmada park','rajkot');

UPDATE:

- With the help of this command we can update the records of a table it means we can set the records of the tables.
- Update all the rows in a table at a time.
- In this type of condition we must be used where clause.

EXAMPLE

UPDATE TABLE SET <column_name> = value [, <column_name> = value….] [WHERE <condition>];

In syntax:

<Table name>	Specify name of the table in which you want to update records.
<Column name>	Specify fieldname for update value
WHERE<condition>	Specify Condition for updation

SYNTAX

UPDATE dept SET deptno = 20 WHERE empid=110;

4. EXPLAIN SELECT STATEMENTS WITH SUITABLE EXAMPLE.

- The SELECT statement is used to select data stored in your database.

SYNTAX

SELECT */<fieldname> |[DISTINCT <fieldname>] FROM <*table_name*>

[GROUP BY <fieldname> [HAVING <condition>]]

[ORDER BY <fieldname> [ORDER]];

In syntax

*	Use to display all fields & records
<field name>	Specify fieldname to display some specific fields

DISTINCT	Used to remove repeated values from fields
<table name>	Specify name of the table to display
WHERE <condition>	Used to retrieve records that match some specific condition
GROUP BY	Used to perform function on group value
HAVING	Used to match conditional records with group function
ORDER BY	Arrange records in some specific order

EXAMPLE

SELECT deptno,dname FROM dept;

SELECT * FROM emp;

SELECT WITH CLAUSES:

- **SELECT ALL ROWS/COLUMNS FORM TABLE**

SELECT * FROM DEPT;

- **SELECT SPECIFIC COLUMNS /ROWS FORM TABLE**

SELECT ENAME,SAL,JOB FROM EMP;

- **SELECT ALL ROWS/COLUMNS FORM TABLE**

SELECT * FROM DEPT;

- **REMOVE DUPLICATE ROWS**

SELECT DISTINCT * FROM DEPT;

- **USE ARITHMETIC OPERATOR(+,_,*,/)**

SELECT ENAME,SAL,SAL + 1000 FROM EMP;

- **USE RELATIONAL OPERATPR(>,<,>=,<=,=,<>)**

SELECT * FROM EMP WHERE DEPTNO>10;

- **USE LOGICAL OPERATOR(AND,OR,NOT)**

SELECT ENAME,SAL,JOB FORM EMP WHERE DEPTNO>10 OR DEPTNO <30;

- **SORTING DATA WITH SELECT**

SELECT * FROM EMP ORDER BY ENAME DESC;

- **RENAMING COLUMN WITH SELECT**
  SELECT ENAME AS "EMPLYEENAME" FROM EMP;

5. EXPLAIN CONSTRAINTS.

- The Constraints are rules that define which data values are valid during insert, update & delete operations.
- In some situation, we need the data that will be check before it will enter into the table.
- Constraints enforce rules at the table level.
- Constraints prevent the deletion of a table if there are dependencies.
- Oracle server uses constraints to prevent invalid data entry into table.
- Constraints can be used to do following:
- Defines rules on data in a table when a row is inserted, updated or deleted from table.
- Prevent the deletion of table if there are dependencies from other table.
- Provide rules for Oracle tools such as oracle developers.
- Oracle gives permission to attach the Data Constraints in the table.
- Once the constraints attach in a table then any type of operation is automatically checked before data storage.
- Constraint may be defined either At the same time as the table is created Or After the table has been created.
- 4 types of constraints are mainly used:
  - **PRIMORY KEY CONSTRAINTS**
  - **FOREIGN KEY CONSTRAINTS**
  - **UNIQUE CONSTRAINTS**
  - **CHECK CONSTRAINTS**
  - **NOT NULL**
- The general structure of constraints when you create a table:

SYNTAX

CREATE TABLE <table_name> (<column_name> data type (<size>), Column_CONSTRAINTS],……[table_Constraint])

In syntax:

<Table name>	Specify name of the table
<Column name>	Specify fieldname
Data type	Specify Types of the data that field hold
Constraint	Specify constraint name

PRIMARY KEYCONSTRAINTS

- A primary key constraint creates a primary key for table.
- A primary key constraint is column or set of column that uniquely identify each row in a table.
- A table can have only one primary key constraint but can have several unique constraints.
- Primary key is used to identify each row is unique or not in the table.
- The table in which primary key is defined that is called primary table.
- Primary key can not contains NULL values as well as can not contain duplicate entry.
- In Primary Key, LONG and LONG ROW data type can not be allowed.
- Unique Index is created automatically on Primary key.
- It can apply at column level as well as table level.

SYNTAX:

(COLUMN LEVEL)

CREATE TABLE <<table name>> (<<column name>> <Data Type><Size> PRIMARY KEY)

(TABLE LEVEL)

CREATE TABLE <<table name>> (…<Column Definition>… PRIMARY KEY (<<column name>>, <<column name>>, <<column name>>…));

EXAMPLE:

(COLUMN LEVEL)

CREATE TABLE stud (studno number (10) PRIMARY KEY, address varchar(20), city varchar(10), mobile varchar(10));

(TABLE LEVEL)

CREATE TABLE stud (studno number(10), address varchar(20), city varchar(10), mobile varchar(10), PRIMARY KEY(studno, address, city mobile));

FOREIGN KEYCONSTRAINTS

- A FOREIGN key is a column or set of column that refers to a primary key in the same table or another table.
- Foreign key are based on the data values and are purely logical.
- A foreign key value must match an existing primary key value or unique key value or else is Null.
- Foreign key must reference either a primary key or unique key column.

- Foreign key is a column that defines how tables related to each other.
- It is a single or group of column whose values are derived from the Primary Key of some other table.
- The table in which foreign key is defined that is called a Foreign Table or Detail Table.
- When we use foreign key constraint, two tables are used like Parent Table and Child Table.
- Parent table is a main table and this type of table contains primary key and foreign key.
- Child table is suitable of parent table whose data values are checked for the current database only.
- This type of foreign key constraint is called Referential Integrity Constraint.
- Parent table can be reference in the foreign key by using the word REFERENCE.
- Child table contains duplicate and null values.

SYNTAX:

(COLUMN LEVEL)

CREATE TABLE <<table name>>(<<column name>> <Data Type><Size> REFERENCES <<table name>> (<<column name>>));

(TABLE LEVEL)

CREATE TABLE <<table name>> (<<column name>> <Data Type><Size>, FOREIGN KEY (<<column name>>) REFERENCES <<table name>>);

EXAMPLE:

(COLUMN LEVEL)

CREATE TABLE stud (studno number(10) REFERENCES student (student_no), address varchar(20), city varchar(10), mobile varchar(10));

(TABLE LEVEL)

CREATE TABLE stud (studno number(10), address varchar(20), city varchar(10), mobile varchar(10), FOREIGN KEY(student_no,student_name) REFERENCES student);

UNIQUE KEYCONSTRAINTS

- Unique key constraints require that every value in a column or set of columns be unique.
- That means no two rows of a table can have duplicate values in specified column or set of columns.
- The column included in the definition of unique key constraints is called unique key.
- If more than one column contains unique constraints than it is known as composite unique key.

- Unique constraints allow the input of NULL values but it doesn't allow duplicate value.
- A unique constraint can be defined at the column level or table level.
- Unique index is created automatically
- This constraint can not support long & long row data type.

SYNTAX:

(COLUMN LEVEL)

CREATE TABLE <<table name>> (<<column name>> <Data Type><Size> UNIQUE);

(TABLE LEVEL)

CREATE TABLE <<table name>> (…<Column Definition>… UNIQUE (<<column name>>, <<column name>>, <<column name>>…));

EXAMPLE:

(COLUMN LEVEL)

CREATE TABLE stud (studno number(10) UNIQUE, address varchar(20), city varchar(10), mobile varchar(10));

(TABLE LEVEL)

CREATE TABLE stud (studno number(10), address varchar(20), city varchar(10), mobile varchar(10), UNIQUE (studno, address, city mobile));

CHECK CONSTRAINT

- Check constraints defines a condition that each row must satisfied.
- The condition can use the same as we specified in query condition.
- It checks the column in a table.
- It returns an output if the column is true otherwise it returns an error.
- It allows duplicate values.
- It does not allow NULL values.
- It can apply at column level as well as table level.

SYNTAX:

(COLUMN LEVEL)

CREATE TABLE <<table name>> (<<column name>> <Data Type><Size> CHECK (condition));

(TABLE LEVEL)

CREATE TABLE <<table name>> (…<Column Definition>… CHECK (<<column name>> CONDITION));

EXAMPLE: (TABLE LEVEL)

CREATE TABLE stud (empno number(10), empanel varchar(20),mobile varchar(10), CHECK(empno BETWEEN 101 AND 300));

EXAMPLE: (COLUMN LEVEL)

CREATE TABLE stud (studno number(10) CHECK (studno BETWEEN 0 AND 100), mobile varchar(10));

NOT NULL CONSTRAINT

- A NOT NULL constraint protects one or more columns in a table.
- It allows duplicate values.
- It does not allow NULL values.
- It can be apply only at a column level.

SYNTAX

CREATE TABLE <<table name>> (<<column name>><Data Type><Size> NOT NULL, <<column name>><Data Type><Size>);

EXAMPLE

CREATE TABLE address (name varchar(20) NOT NULL, address varchar(30),city varchar(15) NOT NULL);

6. EXPLAIN GROUP BY AND HAVING CALUSE WITH EXAMPLE.

Group By:

- A GROUP BY clauses is useful to divide group of the data whenever we use a group function in SELECT statement to perform calculation with the group of fields.

SYNTAX

SELECT <<aggregate function>>/expre FROM <table name> WHERE <condition> GROUP BY <column1>,<column2>… [HAVING <condition>];

EXAMPLE

SELECT deptno,sum(sal) FROM emp GROUP BY deptno;

SELECT deptno,job,max(sal) FROM emp GROUP BY deptno,job;

Having:

- Each column in HAVING must be occur within the function having add condition on the top of GROUP BY and provides additional filter

EXAMPLE

```
SELECT deptno,avg(sal) FROM emp GROUP BY deptno HAVING avg(sal)>1000 ;
```

7. WHAT IS JOIN? EXPLAIN THE TYPES OF JOIN.

- Join can be used when joining the data from the multiple tables.
- Tables are joined on columns having same Data type and size in the table.
- There are 4 types of join like Inner Join, Outer join, Self Join and Cross Join.
- With the help of SELECT statement we can join up to 15 tables.

INNER JOIN

- Inner join is also called equi join because WHERE CLAUSE compares two columns from the two tables with the help of =operator.
- Inner join operator is used equal (=) sign in the query.
- WHERE clause is used in this type of join.
- Condition is used in this type of join.

EXAMPLE

EMP		
EMPNO	**ENAME**	**DEPTNO**
110	**JAY**	**10**
111	**RAJ**	**20**
112	**RAM**	**10**
113	**MEET**	**30**
114	**NIL**	**10**

DEPT	
DEPTNO	**DNAME**
10	**ADMIN**
20	**SALES**
30	**MARKETING**

SELECT E.ENAME, E.DEPTNO, D.DNAME FROM EMP E, DEPT D WHERE E.DEPTNO=D.DEPTNO;

OUTPUT:

OUTPUT		
ENAME	**DEPTNO**	**DNAME**
JAY	10	ADMIN
RAJ	20	SALES
RAM	10	ADMIN
MEET	30	MARKETING
NIL	10	ADMIN

OUTER JOIN

- A join that returns only matching rows from the two tables so this type of join is called outer join.
- Outer join operator is used equal (+) sign in the query.
- WHERE clause is used in this type of join.
- Condition is used in this type of join.
- There are two types of outer join:
    - Left outer join
    - Right outer join

- **LEFT OUTER JOIN:**
    - It will return all rows from right table with matching rows of left table
    - If left side not found than put null
    - It is possible by using (+)= operator

EXAMPLE:

STUD	
RLNO	**NAME**
1	JAY
2	RAJ
3	RAM
4	MEET
5	NIL

MARK	
RLNO	**PER**
1	55
3	65
4	85
6	45
8	65

SELECT S.RLNO,S.NAME,M.PER FROM STUD S, MARK M WHERE S.RLNO(+)=M.RLNO;

OUTPUT		
RLNO	**NAME**	**PER**
1	JAY	55
		65
3	RAM	85
4	MEET	45
		65

- **RIGHT OUTER JOIN:**
    - It will return all rows from left table with matching rows of right table
    - If right side not found than put null
    - It is possible by using (+)= operator

EXAMPLE:

STUD	
RLNO	**NAME**
1	JAY
2	RAJ
3	RAM
4	MEET
5	NIL
9	AVNI

MARK	
RLNO	**PER**
1	55
3	65
4	85
6	45
8	65

SELECT S.RLNO,S.NAME,M.PER FROM STUD S, MARK M WHERE S.RLNO=M.RLNO(+);

OUTPUT		
RLNO	**NAME**	**PER**
1	JAY	55
2	RAJ	
3	RAM	85
4	MEET	45
5	NIL	
9	AVNI	

SELF JOIN

- A join that compares values in a single column of one table so this type of join is called self join.
- A self join can joins the table by itself.
- WHERE clause is used in this type of join.
- Condition is used in this type of join.

EXAMPLE:

STUD		
RLNO	**NAME**	**ID**
1	JAY	3
2	RAJ	5
3	RAM	1
4	MEET	4
5	NIL	2

SELECT S1.RLNO,S1.NAME,S2.NAME FROM STUD S1, STUD S2 WHERE S1.RLNO=S2.ID;

OUTPUT		
RLNO	**NAME**	**NAME**
1	JAY	RAM
2	RAJ	NIL
3	RAM	JAY
4	MEET	MEET
5	NIL	RAJ

CROSS JOIN

- A cross join is a join that is no join between the table.
- WHERE clause is not used in this type of join.
- Condition is not used in this type of join.

EXAMPLE

UNI	
ID	**COUSRENAME**
1	BBA
2	BCA
3	BSCIT

COLLEGE	
NO	**NAME**
1	GEETANJALI
2	MVM

SELECT C.NAME,U.COURSERNAME FROM COLLEGE C , UNI U;

OUTPUT	
NAME	**COUSE NAME**
GEETANJALI	BBA
MVM	BBA
GEETANJALI	BCA
MVM	BCA
GEETANJALI	BSCIT
MVM	BSCIT

8. WHAT IS SUB QUERY? EXPLAIN ITS TYPE.

- Query within a Query is called sub query.
- It is also known as nested query.
- The statement containing a sub query is called parent query.
- Typically sub query appear in the WHERE condition of SELECT statement
- There are 5 types of sub queries:
- Single row sub query
- Multiple row sub query
- Multiple value sub query
- Multiple column sub query
- Correlated sub query

SINGLE ROW SUB QUERY:

- It returns a single value from table.
- The sub query result is than compare to the value of columns in the parent query
- A single row sub query is use a equality operator

EXAMPLE:

SELECT ENAME, SAL FROM EMP WHERE SAL (SELECT MAX (SAL) FROM EMP);

MULTIPLE VALUE SUB QUERY:

- This returns more than one value to the parent query statements.
- IN operator this the most commonly use multiple row sub query.
- We can use exists, all or any operators.

EXAMPLE

SELECT ENAME, SAL, JOB FROM EMP WHERE SAL IN (SELECT MAX (SAL) FROM EMP GROUP BY JOB);

MULTIPLE COLUMN SUB QUERY:

- This returns more value from more than one column.
- A sub query is multiple columns where there is more than one column in select statement of sub query
- Multiple column sub query are generally used to compare column condition.

EXAMPLE:

SELECT ENAME, SAL, JOB FROM EMP WHERE (SAL, JOB) IN (SELECT MAX (SAL), JOB FROM EMP GROUP BY JOB);

MULTIPLE ROW SUB QUERY:

- This type of sub query returns more than one row of result from the sub query.

EXAMPLE:

SELECT ENAME, DEPTNO, FROM EMP WHERE DEPTNO NOT IN (SELECT DEPTNO FROM EMP WHERE ENAME='SCOTT');

CORELATED SUB QUERY:

- This type of sub query are perform when the sub query reference a column from a table refer to in the parent statement.
- The parent statement can be a select update or delete

EXAMPLE:

SELECT DEPTNO, ENAME, SAL FROM EMP E1 WHERE SAL(SELECT MAX(SAL) FROM EMP WHERE DEPTNO=E1.DEPTNO) OFRDER BY DEPTNO;

9. EXPLAIN BUILT IN FUNCTIONS OF ORACLE

ABS()	Returns positive value of negative numbers	SELECT ABS(-55) FROM DUAL;	55
COS()	Returns cos angles value.	SELECT COS(60) FROM DUAL;	-9.52413
SIN()	Returns sin angles value.	SELECT SIN(60) FROM DUAL;	-.3048106
TAN()	Returns tan angles value.	SELECT TAN(60) FROM DUAL;	-.32004039
SUM()	Returns sum of expression	SELECT SUM(SAL) FROM EMP;	29025
AVG()	Returns average of expression	SELECT AVG(SAL) FROM EMP;	2073.21429
MAX()	Returns maximum value of expression	SELECT MAX(SAL) FROM EMP;	5000
MIN()	Returns minimum value of expression	SELECT MIN(SAL) FROM EMP;	800
CEIL()	Returns the nearest highest integer values of an expression	SELECT CEIL(3.544) FROM DUAL;	4
FLOOR()	Returns the nearest highest integer values of an expression	SELECT FLOOR(3.545) FROM DUAL;	3
MOD()	Returns the reminder after dividing M with N	SELECT MOD(5,2) FROM DUAL;	1
POWER()	Returns the power of M with N	SELECT POWER(5,2) FROM DUAL;	25
LOG()	Returns the natural log of base M of N	SELECT LOG(10,1000) FORM DUAL;	2
ROUND()	Returns the round of values of given	SELECT ROUND(6.4885,2) FROM DUAL;	6.49

	expression		
TRUNC()	Similar to ROUND().Returns the round of values of given expression	SELECT TRUNC(6.4885,2) FROM DUAL;	6.48
SQRT()	Returns the square root of specified number	SELECT SQRT(25) FROM DUAL;	5
LOWER()	Convert string in lowercase	SELECT LOWER('GEETANJALI') FROM DUAL;	geetanjali
UPPER()	Convert string in uppercase	SELECT UPPER('geetanjali') FROM DUAL;	GEETANJALI
INITCAP()	Convert each word of string in capital letter.	SELECT INITCAP('geetanjali college') FROM DUAL;	Geetanajli College
LENGTH()	Returns the length of a given string	SELECT LENGTH('GEETANJALI') FROM DUAL;	11
SUBSTR()	Returns substring from given string starting position P to N characters	SELECT SUBSTR('GEETANJALI',5,6) FROM DUAL;	ANJALI
INSTR()	Used to find a particular character in a string	SELECT INSTR('GEETANJALI','E') FROM DUAL;	2
REPLACE()	Replace a specified string with given string.	SELECT REPLACE('GEETANJALI SCHOOL','SCHOOL','COLLEGE') FROM DUAL;	GEETANJALI COLLEGE
RPAD()	Rightpad given string with a given character to N number of character	SELECT RPAD('ABC',5,'*') FROM DUAL;	ABC**
LPAD()	Leftpad given string with a given character to N number of character	SELECT LPAD('ABC',5,'*') FROM DUAL;	**ABC

LTRIM()	Remove the leading space before the string from left side.	SELECT LTRIM(‘ ABC ‘) FROM DUAL;	ABC
RTRIM()	Remove the space after the string from left side.	SELECT LTRIM(‘ ABC ‘) FROM DUAL;	ABC
CONCAT()	Used to merge to strings.	SELECT CONCATE(‘ABC’,’IS NAME’) FROM DUAL;	ABC IS NAME
COUNT()	Returns the number of Rows in the query.	SELECT COUNT(ENAME) FORM EMP;	14
USER()	Returns the current logon users name	SELECT USER FROM EMP;	SCOTT
GREATEST()	Returns the highest value from given expression	SELECT GREATEST(10,20,30,45,60) FROM DUAL;	60
LEAST()	Returns the highest value from given expression	SELECT LEAST(10,20,30,45,60) FROM DUAL;	10
DECODE()	Compares expression with search value one by one if the expression does not match then any of search values returns the default value, if the default value is omitted than returns NULL Syntax: SELECT DECODE(expr,search value1,result1, search	SELECT DECODE(DEPTNO,10,’ADMIN’,20,’SALES’, 30,’PRODUCTION’) AS DEPTNAME FORM EMP;	

	value2,result2, ….) AS <column name> FROM <tablename>;		

10. EXPLAIN GRANT COMMAND WITH SUITABLE EXAMPLE.

- Grant means permission.
- The GRANT statement to give privileges to a specific user or role or to all users
- You can also use this command to grant a role to a user, to PUBLIC, or to another role.
- The following types of privileges can be granted:
    - Delete data from a specific table.
    - Insert data into a specific table.
    - Create a foreign key reference to the named table
    - Select data from a table, view, or a subset of columns in a table.
    - Create a trigger on a table.
    - Update data in a table or in a subset of columns in a table.
    - Run a specified function or procedure.
    - Use a sequence generator or a user-defined type.
- The GRANT command is used in a situation when a table is shared by the other users and we want to gives some specific privileges to the other user.
- By using GRANT command, we can allow the other users to access our database.

SYNTAX:
GRANT <PRIVILEGE> | ROLE ON [<OBJECT NAME>] TO <USER NAME>;
EXAMPLE:
GRANT CONNECT TO USER1;
GRANT INSERT, UPDATE TO CUSTOMER TO USER1;

11. EXPLAIN REVOKE COMMAND WITH SUITABLE EXAMPLE.

- Use the REVOKE statement to remove privileges from a specific user or role from all users.
- You can also use this statement to revoke a role from a user, from PUBLIC, or from another role.
- The following types of privileges can be revoked:
    - Delete data from a specific table.
    - Insert data into a specific table.
    - Create a foreign key reference to the named table
    - Select data from a table, view, or a subset of columns in a table.
    - Create a trigger on a table.

- Update data in a table or in a subset of columns in a table.
- Run a specified routine (function or procedure).
- Use a sequence generator or a user-defined type.

- By using this command, we can withdraw the privileges which have been granted to the user.

SYNTAX:
REVOKE <privilege>| ROLE ON [<object name>] FROM <user name>;

EXAMPLE:
REVOKE CONNECT FROM USER1;
REVOKE INSERT, UPDATE, ON CUSTOMER FROM USER1;

12. HOW CAN WE CREATE USER IN ORACLE? EXPLAIN WITH EXAMPLE.

- Oracle allows the user to create any new user and allows working with their area.
- To create user you need to login as an administrator.
- To create a new user oracle provides **CREATE USER** command.

SYNTAX:
CREATE USER <username> IDENTIFIED BY [password | externally]
[DEFAULT TABLESPACE tablespace]
[TEMPORARY TABLESPACE tablespace]
[QUOTA {integer [K|M]|UNLIMITED} ON tablespace]
[DEFAULT ROLE {role,[role],|ALL}]
[EXCEPT role[,role]...]| NONE

- In above syntax
- **< username> -** specify the name of the user that you want to create
- **IDENTIFIED BY-** specify the password for the new user which you can use for log in
- **DEFAULT TABLESPACE-** specify the table space where you to store users data.
- **TEMPORARY TABLESPACE-** specify the table space where you to store users data.
- **QUOTA-** this clause specify the limit of the data that user can store in tablespace.
- **DEFAULT ROLE-** allows you to specify the roles for the user
- **EXCEPT ROLE-** this role allows you to specify the name of the role that you don't want to provide.

SYNTAX:

CREATE USER **u123** IDENTIFIED BY **u12345;**

13. TRANSACTION CONTROL STATEMENT (tcs)?

- TCL- stands form Transaction control
- It is one type of control that is used for manage the RDBMS.
- Transaction control performs database transaction.
- Database transaction is one type of work that is used to store information in the database.
- If the database transaction is fail or wrong then main database is fail on the server.
- If the database transaction is good then main database is successfully running on the server.
- We can easy to develop the transaction by using ORACLE RDBMS SQL syntax or SYBASE SQL SERVER syntax.
- We can use the COMMIT, ROLLBACK and SAVEPOINT statement in our transaction process.

SYNTAX:

```
SET TRANSACTION { READ ONLY / USE ROLLBACK SEGMENT }
```

- If we can set "READ ONLY" in the transaction then we can only read the transaction.

EXAMPLE

```
Set Transaction READ ONLY;
Select * from stud where student_name = 'john';
Commit;
```

14. EXLAIN COMMIT, ROLLBACK AND SAVEPOINT COMMNAD IN BRIEF.

COMMIT:

- COMMIT command save all changes that made by during a transaction.
- If we can set COMMIT command before beginning a transaction then no errors are generated.

EXAMPLE 1:

EXAMPLE 1:

```
Set Transaction READ ONLY;
Select * from stud where city = 'rajkot';
Commit;
```

- In Example 1, we cannot set "COMMIT STATEMENT" before the transaction process so errors may be generated before the transaction process.

ROLLBACK:

- When the transaction is running some errors are generated so that transaction is terminated.
- If we can set ROLLBACK command in our transaction process then transaction is rollback means transaction is back to its beginning.
- If we can set ROLLBACK command in our transaction process and when some errors are generated during transaction process then transaction is not terminated.

EXAMPLE:

```
Set Transaction READ ONLY;

Insert into emp1 Values (1, 'Mira', 50000);
Insert into emp1 Values (2, 'Manali', 70000);
ROLLBACK;
Select * from emp1;
```

SAVEPOINT:

- We can save work by using SAVEPOINT command.
- SAVEPOINT is one type of point that divide the long transaction into smaller.
- It is used to identify the point in a transaction process.

SYNTAX:

```
SAVEPOINT SAVE POINT NAME;
```

EXAMPLE:

```
Set Transaction READ ONLY;
SAVEPOINT abc;

Insert into emp1 Values (1, 'Mira', 50000);
SAVEPOINT def;
 Insert into emp1 Values (2, 'Manali', 70000);
```