PLSQL Interview Question:

**DDL - Data definition Language**

**CREATE**: Create table with Primary Key

CREATE TABLE table\_name (

emp\_id INT,

emp\_name VARCHAR(10),

emp\_address VARCHAR(45),

mobile\_no INT,

PRIMARY KEY (emp\_id)

);

#Create table with Primary Key, Not Null, Unique and Auto Increment

create table table\_name (

sr\_no int **auto\_increment**,

me\_code varchar(15),

me\_name varchar(25),

mobile\_no int,

**primary key** (sr\_no),

**UNIQUE INDEX** **me\_code\_UNIQUE** (me\_code)

);

#Create table with Foreign Key

create table table\_team(

team\_name VARCHAR2(10),

team\_profile varchar2(15),

emp\_id int references table\_name(emp\_id)

);

How to Create backup of existing table.

create table table\_name\_29Dec2019 as select \* from table\_name;

View and materialized view In Oracle: is a virtual table that does not physically exist.

The first difference between View and materialized view is that In Views query result is not stored in the disk or database but Materialized view allow to store the query result in disk or table.

One more difference between View and materialized view in the database is that In case of View we always get latest data but in case of Materialized view we need to refresh the view for getting latest data.

ALTER:

a) Add Column

alter table table\_name **add** **empid** int;

alter table table\_name **add** **emp\_address** varchar(45);

b)Remove Column

ALTER TABLE table\_name **DROP** me\_address;

ALTER TABLE table\_name **DROP** column me\_adhar\_no;

c)Modify Column

alter table table\_name **MODIFY** **emp\_address** varchar(45);

alter table aksh4 modify mobile\_no **bigint**;

d)Rename Table name

alter table table\_name **rename** to table\_name2 ;

e)Constraints Modify

alter table table\_name add **UNIQUE INDEX** me\_code\_UNIQUE (me\_code);

alter table table\_name add constraint foreign\_key\_name foreign key (team\_id) references table\_name(emp\_id);

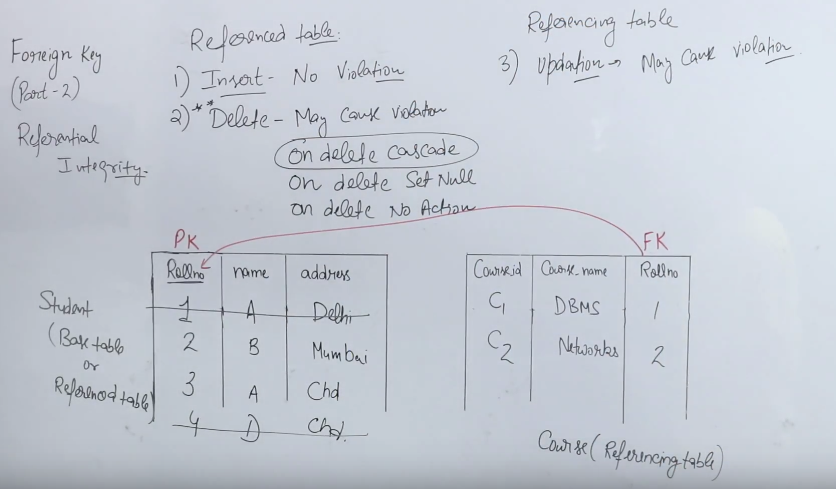
e.g

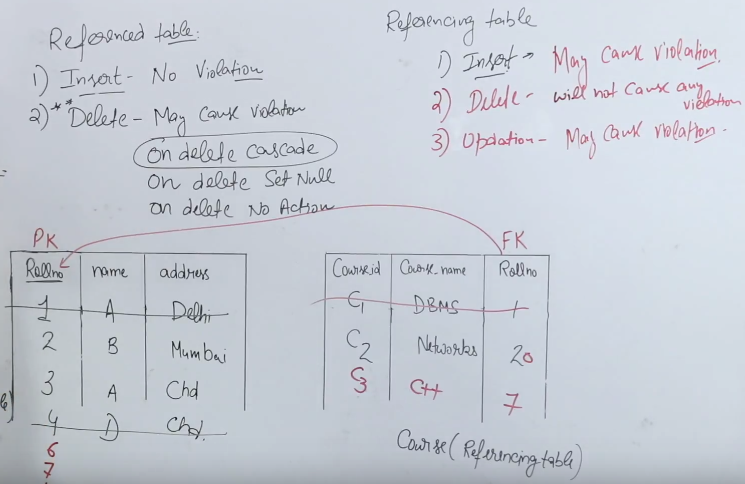
alter table table\_team add constraint foreign\_key\_name foreign key (emp\_id) references table\_name(emp\_id);

CONSTRAINTS:

Primary Key: In Single table only on primary key is allowed

Foreign Key: One table can have multiple foreign key column





DROP:

drop table table\_name;

TRUNCATE:

truncate table table\_name;

**DML - Data Manipulation Language (DISU)**

**SET SQL\_SAFE\_UPDATES = 0;**

**SELECT :**

**LIKE CONDITION**

**select \* from table\_name where me\_name like '%Sushil%';**

**select me\_code,me\_name from aksh4;.**

**SELECT \* FROM employees order by salary desc fetch first 5 rows only**

**select \* from table\_name:**

**SELECT COUNT(\*) FROM CITY;**

**SELECT \* FROM CITY order by Population asc;**

**SELECT \* FROM CITY order by id desc;**

**SELECT \* FROM CITY WHERE NAME='Kabul';**

**select \* from city where countRycode='NLD' and district='Noord-Holland' and id=25;**

**select \* from city where id between 100 and 105;**

**INSERT:**

**insert into table\_name (empid,empname,empaddress) values (95,'sushil','mumbai');**

**insert into table\_name (emp\_name) values ('sushil');**

**How to insert data from one table to another in one query**

**: Both table structure should be same i.e column count | data type exempt for number to varchar2 or vice versa**

**insert into EMPLOYEE\_DATA\_MAIN (SELECT \* FROM EMPLOYEE\_DATA\_TEMP) ;**

**To insert limited columns value into table**

**insert into EMPLOYEE\_DATA\_TEMP(emp\_id,full\_name,age)**

**(SELECT emp\_id,full\_name,age FROM EMPLOYEE\_DATA\_MAIN) ;**

**UPDATE:**

**update** table\_name **set mobile\_no=8007306992461121458;**

**UPDATE** table\_name **SET emp\_add = 'navi mumbai' where sr\_no=16;**

**UPDATE** table\_name **SET EMP\_NAME ='BMS';**

**DELETE:**

**delete from** table\_name **where sr\_no between 4 and 7;**

**Sequence:**

**CREATE SEQUENCE customers\_seq**

**START WITH 1000**

**INCREMENT BY 1**

**NOCACHE**

**NOCYCLE;**

**E.g We have specify SEQUENCE**

**insert into aksh4 (SR\_NO,me\_code,me\_name,mobile\_no) values (SEQ\_SR\_NO.NEXTVAL,'A113','Sushil33',1234567892);**

**Auto Increment:**

**ALTER TABLE** table\_name **AUTO\_INCREMENT = 2000;**

**e.g No need to specify sr\_no in insert query**

**insert into aksh4 (me\_code,me\_name,mobile\_no) values ('A113','Sushil33',1234567892);**

**SELECT \* FROM COLLEGESTD;**

**SELECT \* FROM DEPT;**

**INSERT INTO COLLEGESTD (STD\_ID,STD\_NAME,STD\_ADDRESS,MOBILE\_NO,DEPT\_ID) VALUES (6,'bhumi','delhi',78794563,102);**

**update COLLEGESTD set dept\_id=100 where std\_id=1;**

**select COLLEGESTD.std\_id,COLLEGESTD.std\_name,dept.std\_name,COLLEGESTD. mobile\_no from COLLEGESTD**

**join DEPT**

**on COLLEGESTD.dept\_id = dept.dept\_id**

**where COLLEGESTD.std\_id in (select max(COLLEGESTD.std\_id) from COLLEGESTD group by COLLEGESTD.dept\_id) ;**

**MySQL Database**

**PL/SQL:**

**Example 1:**

**CREATE DEFINER=`root`@`localhost` PROCEDURE `addition\_proc`(IN FIR int,IN SEC int,OUT THI int)**

**BEGIN**

**set THI=FIR+SEC;**

**END**

**To run:**

**set @FIR=10, @SEC=20 ;**

**CALL addition\_proc(@FIR,@SEC,@THI);**

**select @THI;**

**Example 2:**

**CREATE DEFINER=`root`@`localhost` PROCEDURE `city\_list`(IN CITY\_ID INT, OUT CITY\_NAME VARCHAR(255) )**

**BEGIN**

**SELECT NAME INTO CITY\_NAME FROM CITY WHERE ID = CITY\_ID;**

**END**

**To Run:**

**set @CITY\_ID=2;**

**CALL city\_list(@CITY\_ID,@CITY\_NAME);**

**SELECT @CITY\_NAME;**

**SEQUENCE:**

**SELECT SEQ\_MERCHANTCODE.NEXTVAL FROM DUAL;**

**DESC TABLE\_NAME;**

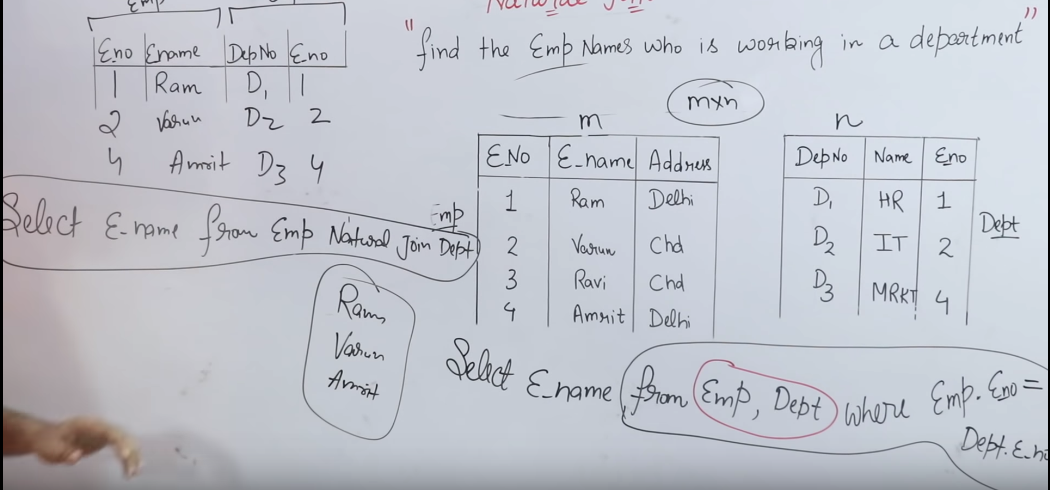
**VIEWS:**

**FUNCTIONS:**

**TRIGGERS:**

**Joins IN SQL**

**NATURAL JOIN: if you are using natural join keyword in between then e\_no should be same in both table , otherwise use (crossproduct + where conditions)**

****

**Example 1:**

**CREATE OR REPLACE PROCEDURE PROC\_EMPLOYEE(EMPNAME IN VARCHAR2,ADDRESS IN VARCHAR2,MOBILENO IN INT)**

**AS**

**SEQID INT;**

**BEGIN**

**SEQID :=SEQ\_MERCHANTCODE.NEXTVAL;**

**INSERT**

**INTO TABLE\_NAME**

**(**

**EMP\_ID,**

**EMP\_NAME,**

**EMP\_ADDRESS,**

**MOBILE\_NO**

**)**

**VALUES**

**(**

**SEQID,**

**EMPNAME,**

**ADDRESS,**

**MOBILENO**

**);**

**COMMIT;**

**DELETE FROM TABLE\_NAME WHERE EMP\_ID=(SEQID-5);**

**COMMIT;**

**DBMS\_OUTPUT.PUT\_LINE('HELLO WORLD');**

**END PROC\_EMPLOYEE;**

**Example:2**

**create or replace PROCEDURE PROC\_DEMO1 ( NUM1 IN NUMBER , NUM2 IN NUMBER , NUM3 OUT NUMBER ) AS**

**BEGIN**

**NUM3 := (NUM1+NUM2);**

**DBMS\_OUTPUT.PUT\_LINE(NUM3);**

**END PROC\_DEMO1;**

***TO procedure execute using command:***

***declare***

***x number;***

***BEGIN***

***PROC\_DEMO1(10,20,x);***

***end ;***

**To Create Package and inside package multiple procedure**

Package:

create or replace PACKAGE YOUR\_PACKAGE\_NAME AS

PROCEDURE GET\_DATA ( MID VARCHAR2,FROMDATE DATE, MY\_CURSOR OUT SYS\_REFCURSOR ); END YOUR\_PACKAGE\_NAME;

create or replace PACKAGE BODY YOUR\_PACKAGE\_NAME AS

PROCEDURE GET\_DATA ( MID VARCHAR2,FROMDATE DATE, MY\_CURSOR OUT SYS\_REFCURSOR )

IS BEGIN

IF MID IS NULL THEN OPEN MY\_CURSOR FOR

SELECT \* FROM TABLE\_NAME1; ELSE

OPEN MY\_CURSOR FOR

SELECT \* FROM TABLE\_NAME2;

END IF; EXCEPTION

WHEN OTHERS THEN RAISE;

END GET\_DATA;

END YOUR\_PACKAGE\_NAME;

**Procedure:**

1. cannot return any values through the RETURN statement.

2. CREATE PROCEDURE instructs the compiler to create new procedure. Keyword 'OR REPLACE'

instructs the compile to replace the existing procedure (if any) with the current one.

3. Procedure name should be unique.

4. Keyword 'IS' will be used, when the procedure is nested into some other blocks. If the procedure is standalone then 'AS' will be used. Other than this coding standard, both have the same meaning.

EXAMPLE 1:

create or replace PACKAGE YOUR\_PACKAGE\_NAME AS

PROCEDURE GET\_DATA (MY\_CURSOR OUT SYS\_REFCURSOR ); PROCEDURE GET\_SINGLE\_RESPONSE(P\_RESPONSE OUT VARCHAR2);

END YOUR\_PACKAGE\_NAME;

--BODY

create or replace PACKAGE BODY YOUR\_PACKAGE\_NAME AS

PROCEDURE GET\_DATA(MY\_CURSOR OUT SYS\_REFCURSOR ) IS

BEGIN

OPEN MY\_CURSOR FOR

SELECT \* FROM EPORTAL\_TERMINAL; EXCEPTION

WHEN OTHERS THEN RAISE;

END GET\_DATA;

PROCEDURE GET\_SINGLE\_RESPONSE(P\_RESPONSE OUT VARCHAR2) IS

ERR\_NUM NUMBER; ERR\_MSG VARCHAR2(100); BEGIN

--ANY LOGIC CAN BE WRIITEN HERE

--SELECT \* FROM EPORTAL\_TERMINAL; P\_RESPONSE :='SUCCESS BHACCHI'; EXCEPTION

WHEN OTHERS THEN P\_RESPONSE :='ERROR'; RAISE;

END GET\_SINGLE\_RESPONSE; END YOUR\_PACKAGE\_NAME;

**What is Function?**

Functions is a standalone PL/SQL subprogram. Like PL/SQL procedure, functions have a unique name by which it can be referred. These are stored as PL/SQL database objects. Below are some of the characteristics of functions.

• Functions are a standalone block that is mainly used for calculation purpose.

• Function use RETURN keyword to return the value, and the datatype of this is defined at the time of creation.

• A Function should either return a value or raise the exception, i.e. return is mandatory in functions.

• Function with no DML statements can be directly called in SELECT query whereas the function with

DML operation can only be called from other PL/SQL blocks.

• It can have nested blocks, or it can be defined and nested inside the other blocks or packages.

• It contains declaration part (optional), execution part, exception handling part (optional).

• The values can be passed into the function or fetched from the procedure through the parameters.

• These parameters should be included in the calling statement.

|  |  |  |  |
| --- | --- | --- | --- |
| • Function can also return the value through | OUT parameters | other than using | RETURN. |

• Since it will always return the value, in calling statement it always accompanies with assignment operator to populate the variables.

**Procedure Vs. Function: Key Differences**

**Procedure Function**

• Used mainly to a execute certain process • Used mainly to perform some calculation

• Cannot call in SELECT statement • A Function that contains no DML statements can be called in SELECT statement

• Use OUT parameter to return the value • Use RETURN to return the value

• It is not mandatory to return the value • It is mandatory to return the value

• RETURN will simply exit the control from subprogram.

• RETURN will exit the control from subprogram and also returns the value

• Return datatype will not be specified at

the time of creation • Return datatype is mandatory at the time of creation

**Built-in Functions in PL/SQL**

PL/SQL contains various built-in functions to work with strings and date datatype. Here we are going to see the commonly used functions and their usage.

Conversion Functions

These built-in functions are used to convert one datatype to another datatype.

**Function Name Usage EXAMPLE**

TO\_CHAR Converts the other datatype to character datatype TO\_CHAR(123);

TO\_DATE ( string, format )

Converts the given string to date. The string should match with the format.

TO\_DATE('2015-JAN-15', 'YYYY- MON-DD');

Output: 1/15/2015

Select TO\_NUMBER('1234','9999') fro m dual;

TO\_NUMBER (text, format)

Converts the text to number type of the given format. Informat Output: 1234

'9' denotes the number of digits

Select TO\_NUMBER('1,234.45','9,999.

99') from dual;

String Functions

These are the functions that are used on the character datatype.

Output: 1234

**Function Name Usage EXAMPLE**

INSTR(text, string, start

, occurance)

Gives the position of particular text in the given string.

• text – Main string

• string – text that need to be searched

• start – starting position of the search (optional)

Select INSTR('AEROPLANE','E',2,1) from dual

**Output**: 2 Select

INSTR('AEROPLANE','E',2,2) from dual

• accordance – occurrence of the searched string Output: 9 (2nd occurance of E) (optional)

SUBSTR ( text, start, le ngth)

Gives the substring value of the main string.

• text – main string

• start – starting position

• length – length to be sub stringed

select substr('aeroplane',1,7) from dual

**Output**: aeropla

UPPER ( text ) Returns the uppercase of the provided text

Select upper('guru99') from dual;

**Output**: GURU99

LOWER ( text ) Returns the lowercase of the provided text

Select lower ('AerOpLane') from dual;

**Output**: aeroplane

**Function Name Usage EXAMPLE**

INITCAP ( text)

Returns the given text with the starting letter in upper case.

Select ('guru99') from dual

**Output**: Guru99

Select ('my story') from dual

**Output**: My Story

LENGTH ( text ) Returns the length of the given string

Select LENGTH ('guru99') from dual;

**Output**: 6

LPAD ( text, length, pad\_char)

Pads the string in the left side for the given length (total string) with the given character

Select LPAD('guru99', 10, '$') from dual;

**Output**: $$$$guru99

RPAD (text, length, pad\_char)

Pads the string in the right side for the given length

(total string) with the given character

Select RPAD('guru99',10,'-') from dual

**Output**: guru99----

LTRIM ( text ) Trims the leading white space from the text

Select LTRIM(' Guru99') from dual;

**Output**: Guru99

RTRIM ( text ) Trims the trailing white space from the text

Select RTRIM('Guru99 ') from dual;

**Output**; Guru99

Date Functions

These are functions that are used for manipulating with dates.

**Function Name Usage EXAMPLE**

ADD\_MONTHS (date, no.of

months) Adds the given months to the date

ADD\_MONTH('2015-01-01',5);

**Output**: 05/01/2015

SYSDATE

Returns the current date and time of the server

Select SYSDATE from dual;

**Output**: 10/4/2015 2:11:43 PM

TRUNC

Round of the date variable to the lower possible value

select sysdate, TRUNC(sysdate) from dual; **Output**:

10/4/2015 2:12:39 PM 10/4/2015

ROUND

MONTHS\_BETWEEN

Rounds the date to the nearest limit either higher or lower

Returns the number of months between two dates

Select sysdate, ROUND(sysdate) from dual **Output**:

10/4/2015 2:14:34 PM 10/5/2015

Select MONTHS\_BETWEEN (sysdate+60, sysdate) from dual

**Output**:

**Q #3) How will you differentiate between VARCHAR & VARCHAR2?**

VARCHAR can store characters up to 2000 bytes while VARCHAR2 can store up to 4000 bytes.

VARCHAR will hold the space for characters defined during declaration even if all of them are not used whereas VARCHAR2 will release the unused space.

**Q #4) What is the difference between TRUNCATE & DELETE command? Ans:** Both the commands are used to remove data from a database.

**The finer differences between the two include:**

• TRUNCATE is a DDL operation while DELETE is a DML operation.

• The TRUNCATE command will free the object storage space while the DELETE command does not.

**Q #7) What is the difference between SUBSTR & INSTR functions?**

**Ans:** SUBSTR function returns the sub-part identified by numeric values from the provided string.

**Example:** [Select SUBSTR (‘India is my country’, 1, 4) from dual] will return “Indi”.

INSTR will return the position number of the sub-string within the string.

**Example:** [SELECT INSTR (‘India is my country’, ‘a’) from dual] will return 5.

**Q #8) How can we find out the duplicate values in an Oracle table?**

SELECT EMP\_NAME, COUNT (EMP\_NAME) FROM EMP

GROUP BY EMP\_NAME

HAVING COUNT (EMP\_NAME) > 1;

**Q #10) What is a NVL function? How can it be used?**

**Ans:** NVL is a function, which helps the user to substitute a value if null is encountered for an expression.

**It can be used as the below syntax.**

[NVL (Value\_In, Replace\_With)]

**Q #11) What is the difference between a Primary Key & a Unique Key?**

**Ans:** Primary key is used to identify each table row uniquely, while a Unique Key prevents duplicate values in a table column.

**Given below are few differences:**

• The primary key cannot hold null value at all while Unique key allows multiple null values.

• The primary key is a clustered index while a unique key is a non-clustered index.

**Q #32) What is meant by an index?**

**Ans:** An index is a schema object, which is created to search the data efficiently within the table. Indexes are usually created on certain columns of the table, which are accessed the most.

Indexes can be clustered or non-clustered.

**Q #13) How can we find out the current date and time in Oracle?**

SELECT SYSDATE into CURRENT\_DATE from dual;

**Q #17) What is the quickest way to fetch the data from a table?**

**Ans:** The quickest way to fetch the data would be to use primary key column with index in the SQL Query.

**Q #21) What is the use of Aggregate functions in Oracle?**

**Ans:** Aggregate functions perform summary operations on a set of values to provide a single value. There are several aggregate functions that we use in our code to perform calculations.

**Few of them are listed below:**

• AVG

• MIN

• MAX

• COUNT

• SUM

• STDEV

**Q #15) How will you write a query to get a 5th RANK student from a table STUDENT\_REPORT? Ans: The Query will be as follows:**

SELECT TOP 1 RANK

FROM (SELECT TOP 5 RANK FROM STUDENT\_REPORT

ORDER BY RANK DESC) AS STUDENT

ORDER BY RANK ASC;

**Q #22) What are the set operators UNION, UNION ALL, MINUS & INTERSECT meant to do? Ans:** Set operator facilitates the user to fetch the data from two or more than two tables at once if the columns and relative data types are same in the source tables.

• UNION operator returns all the rows from both the tables except the duplicate rows.

• UNION ALL returns all the rows from both the tables along with the duplicate rows.

• MINUS returns rows from the first table, which does not exist in the second table.

• INTERSECT returns only the common rows in both the tables.

**Q #23) Can we convert a date to char in Oracle and if so, what would be the syntax? Ans:** We can use the TO\_CHAR function to do the above conversion.

**The syntax will be as follows:**

[SELECT to\_char (to\_date ('30-01-2018′, ‘DD-MM-YYYY'), ‘YYYY-MM-DD') FROM dual;]

**Q #24) What do you mean by a database transaction & what all TCL statements are available in**

**Oracle?**

**Ans:** Transaction occurs when a set of SQL statements are executed in one go. To control the execution of these statements, Oracle has introduced TCL i.e. Transaction Control Statements that use a set of statements.

**The set of statements include:**

• **COMMIT:** Used to make a transaction permanent.

• **ROLLBACK:** Used to roll back the state of DB to last the commit point.

• **SAVEPOINT:** Helps to specify a transaction point to which rollback can be done later.

**Q #25) What do you understand by a database object? Can you list a few of them?**

**Ans:** An object used to store the data or references of the data in a database is known as a Database object.

The database consists of various types of DB objects such as tables, views, indexes, constraints, stored procedures, triggers etc.

**Q #27) Can we save images in a database and if yes, how?**

**Ans:** BLOB stands for Binary Large Object, which is a datatype that is generally used to hold images, audio & video files or some binary executables.

This datatype has the capacity of holding data up to 4 GB.

**Q #28) What do you understand by database schema and what does it hold?**

**Ans:** Schema is a collection of database objects owned by a database user who can create or manipulate new objects within this schema.

The schema can contain any DB objects like table, view, indexes, clusters, stored procs, functions etc.

**Q #30) What is a View and how is it different from a table?**

**Ans:** A view is a user-defined database object that is used to store the results of a SQL query, which can be referenced later. Views do not store this data physically but as a virtual table, hence it can be referred

as a logical table.

A table can hold data but not SQL Query results whereas View can save the query results, which can be used in another SQL Query as a whole.

The table can be updated or deleted while Views cannot be done so.

**Q #31) What is meant by a deadlock situation?**

**Ans:** Deadlock is a situation when two or more users are simultaneously waiting for the data, which is locked by each other and hence, results in all blocked user sessions.

**Q #37) What are the parameters that we can pass through a stored procedure?**

**Ans:** We can pass IN, OUT & INOUT parameters through a stored procedure and they should be defined while declaring the procedure itself.

**Q #38) What is a trigger and what are its types?**

**Ans:** A trigger is a stored program which is written in such a way that it gets executed automatically when some event occurs. This event can be any DML or a DDL operation.

**PL/SQL supports two types of triggers:**

• Row Level

• Statement Level

**Q #39) How will you distinguish a global variable with a local variable in PL/SQL?**

**Ans:** Global variable is the one, which is defined at the beginning of the program and survives until the end.

It can be accessed by any methods or procedures within the program, while the access to the local variable is limited to the procedure or method where it is declared.

**Q #40) What are the packages in PL SQL?**

**Ans:** A Package is a group of related database objects like stored procs, functions, types, triggers, cursors etc. that are stored in Oracle database. It is a kind of library of related objects which can be accessed by multiple applications if permitted.

**My Oracle Installation Guide:**

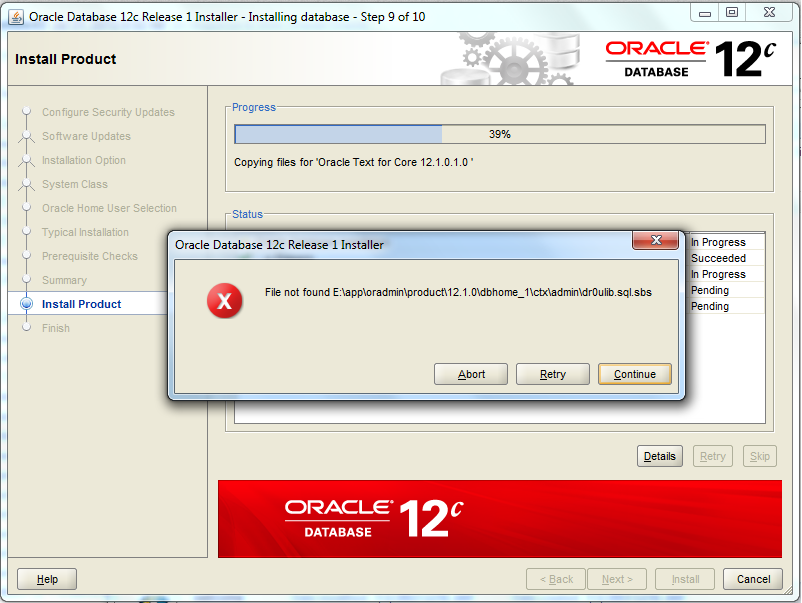
**winx64\_12102\_database\_1of2**

**winx64\_12102\_database\_2of2**

**Fixing missing files in Oracle 12c installation**

Oracle 12c has a pretty dumb way of packaging the installer into two zip files along with the instructions of extracting the two zip files into one directory. This is commonly misunderstood as putting ***winx64\_12c\_database\_1of2*** directory and  ***winx64\_12c\_database\_2of2*** directory in the one directory say oracle. This is also due to the default setting of the common extraction tools, such as Winzip etc.

However on using this instruction when the installation is started using the ***setup.exe*** the following error is returned:

[](http://2.bp.blogspot.com/-6nHEPlC6pWM/UzPwqboE2vI/AAAAAAAAEA4/_79j2MjQhHI/s1600/pic1.png)

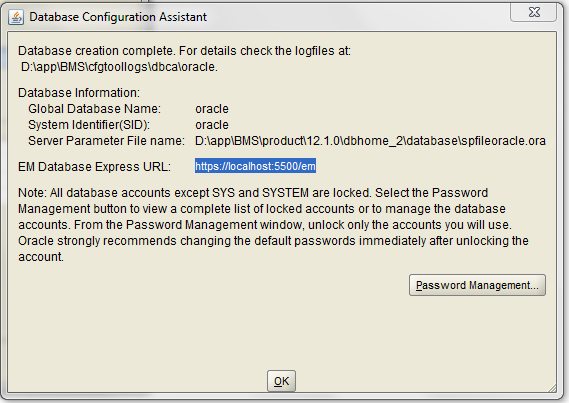
Files, such as ***dr0ulib.sql.sbs*** are reported by the installer as not found. To solve these you need to do the following:

1) Abort the current installer

2) Open the ***winx64\_12c\_database\_2of2*** directory and navigate to ..\***winx64\_12c\_database\_2of2\database\stage\Components*** directory copy all the files

3) Paste all files to the following location ***..\winx64\_12c\_database\_1of2\database\stage\Components***

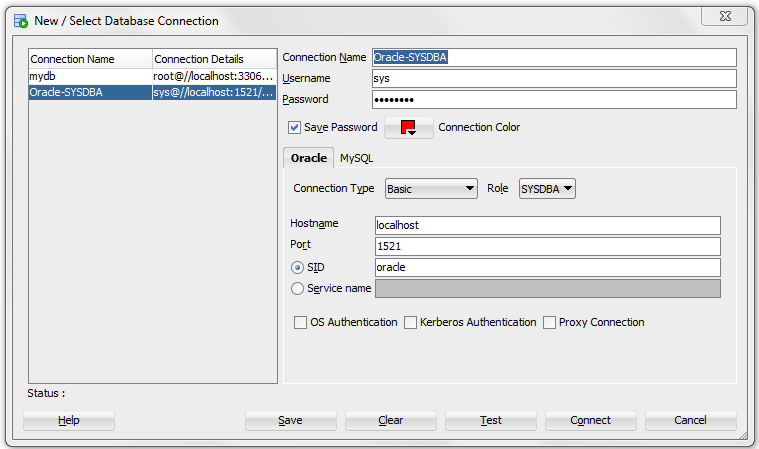
Now run the ***setup.exe*** as admin and follow the instructions to complete the installation without error.

****

**NOte: Role should be SYSDBA**

**Username:sys**

**Password: Asdf1234**

****

TO connect to HR schema

Step 1: Open tnsnames.ora file , you can get in the below path

Path: D:\app\BMS\product\12.1.0\dbhome\_2\NETWORK\ADMIN\tnsnames.ora

Step 2: Add below line and save it , close it

PDBORCL =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))

(CONNECT\_DATA =

(SERVER = DEDICATED)

(SERVICE\_NAME = pdborcl)

)

)

Step 3: Open CMD as administrator , type this ' lsnrctl reload'

Step 4: sqlplus / as sysdba

Step 5: show con\_name

Step 6: alter session set container = pdborcl;

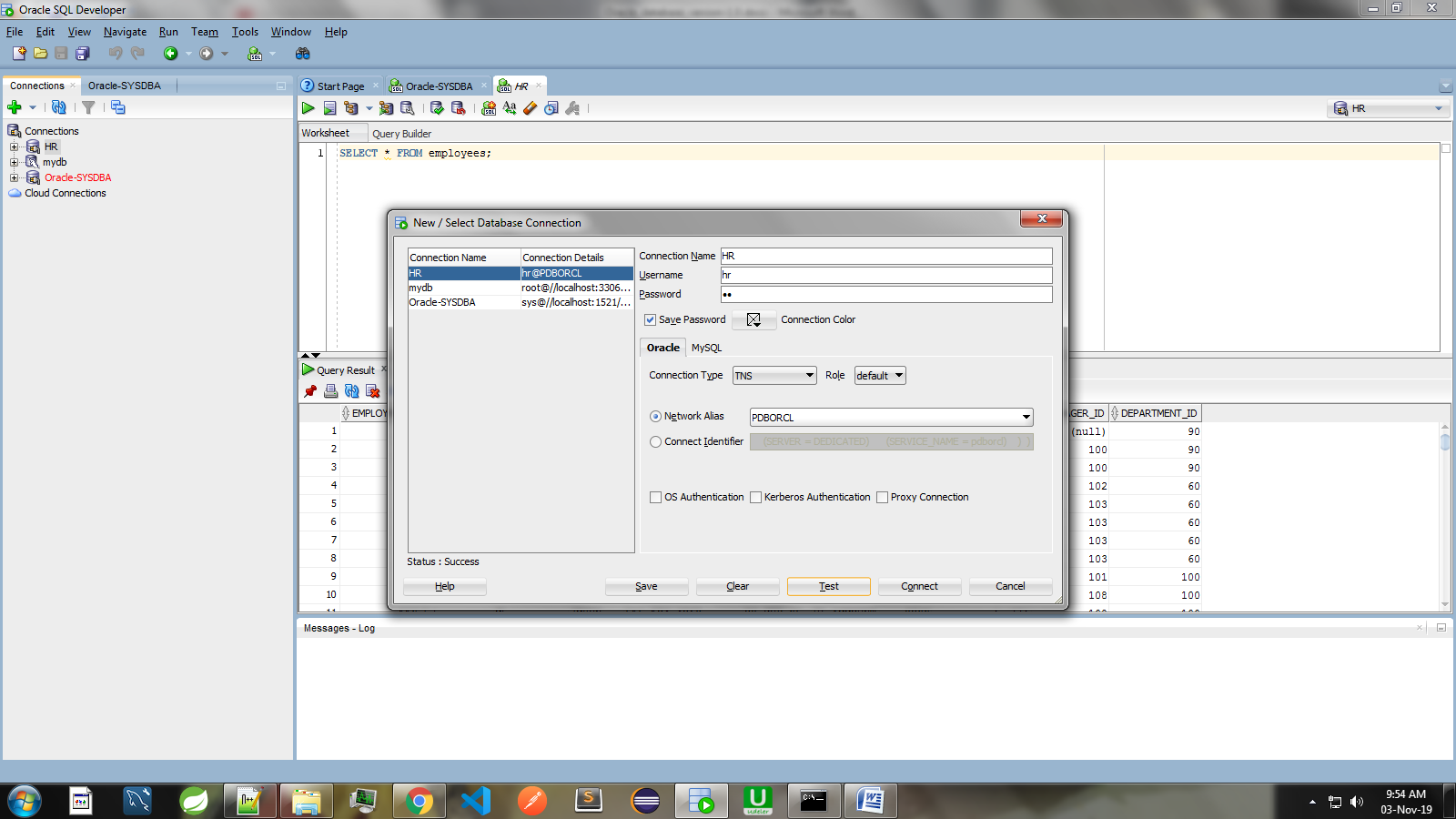
Step 7: select name, open\_mode from v$pdbs;

Step 8: alter pluggable database open;

Step 9: alter user hr identified by hr account unlock;

Step 10: conn hr/hr@PDBORCL

**Note: Username : hr , Password : hr**

****

**After Computer restart you wont be able to connect to HR schema , so do the following steps.**

Step 1: Open CMD

Step 2: sqlplus / as sysdba

Step 3: alter session set container = pdborcl;

Step 4: alter pluggable database open;

Now, connect HR schema.

TO create custom user:

<https://blogs.oracle.com/sql/how-to-create-users-grant-them-privileges-and-remove-them-in-oracle-database>

1. Open CMD,

2. sqlplus

3.type username:system

4.password: DBA password i.e Asdf1234 in your case Password will be different

5. alter session set "\_ORACLE\_SCRIPT"=true;

6. create user user\_name identified by password;

i.e create user **stacktech** identified by Asdf1234;

// TO give grant by DBA to the user which is created

7. **GRANT**  **CONNECT**, RESOURCE, DBA **TO** **stacktech**;

Next you’ll want to ensure the user has privileges to actually connect to the database and create a session using GRANT CREATE SESSION. We’ll also combine that with all privileges using GRANT ANY PRIVILEGES.

8. grant create session to **stacktech**;

We also need to ensure our new user has disk space allocated in the system to actually create or modify tables and data, so we’ll GRANT TABLESPACE like so:

9. alter user **stacktech** quota unlimited on users;

10. grant create view, create procedure, create sequence to **stacktech**;

Username:stacktech

Password:Asdf1234

