**Answers to Exercises in Oracle Database 12c for beginners  
- Srikanth Pragada**

**Chapter 1: INTRODUCTION TO DBMS**

1. **\_\_\_\_\_\_\_\_ designed relational model**

Dr. E.F. CODD

1. **Data models are \_\_\_\_\_\_\_, \_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_ and \_\_\_\_\_\_\_\_.**

Hierarchical, Network, Relational, Object-oriented

1. **Composite primary key is \_\_\_\_\_\_\_\_\_\_\_.**

A primary key that is consisting of multiple columns

1. **A row is otherwise known as \_\_\_\_\_\_\_\_.**

Tuple

1. **How many tables does SELECT operator take?**

One

1. **\_\_\_\_\_\_ is an example for an RDBMS.**

Oracle Database

1. **SQL command used to create table belongs to \_\_\_\_\_\_ category of SQL Commands.**

DDL

1. **\_\_\_\_\_\_\_key is used to join a child table with parent table.**

Foreign

1. **\_\_\_\_\_\_ is the standard language for RDBMS.**

SQL

1. **What is a domain?**

A set of potential values

**Chapter 2: INTRODUCTION TO ORACLE**

1. **Oracle instance is a collection of \_\_\_\_\_\_\_\_\_ and \_\_\_\_\_\_\_\_\_\_\_\_.**

Memory structures and Processes

1. **What is the use of Host String?**

Host string is used to connect to server from clients like Sql\*plus

1. **SQL\*PLUS commands must be terminated with semicolon (;) [TRUE/FALSE]**

FALSE

1. **What is the password of user SYSTEM?**

Whatever is given at the time of installation

1. **\_\_\_\_\_\_\_\_\_\_\_\_\_ is where we enter SQL commands in SQL Developer.**

Worksheet

**Chapter 3: GETTING STARTED WITH ORACLE DATABASE 12C**

1. **\_\_\_\_\_\_ is the operator used to compare a column with null value.**

IS NULL

1. **\_\_\_\_\_\_ operator is used to compare one value with a set of values.**

IN

1. **The maximum number of characters that can be stored in CHAR type is \_\_\_\_\_\_.**

2000

1. **In LIKE operator, % stands for \_\_\_\_\_\_\_\_\_\_\_\_\_.**

Zero or more characters of any kind

1. **\_\_\_\_\_\_ is used to change the heading of a column.**

Alias

1. **\_\_\_\_\_\_ command is used to display definition of a table.**

DESCRIBE

1. **Display list of courses where course code starts with letter ‘c’.**

select \* from courses where code like 'c%'

1. **Display rows of COURSES table in the ascending order of course fee and descending order of course code.**

select \* from courses order by fee, code desc

1. **Select rows from COURSES where course fee is in the range 3000 to 5000.**

select \* from courses where fee between 3000 and 5000

1. **Add a new row to COURSES table with the following data - course code - cpp, name – C++ Programming, duration – 30, fee – 3500, prereq – C programming.**

insert into courses values('cpp','C++ Programming',30,3500,'C Programming')

1. **Display all the rows where course fee is not known but duration is known.**

select \* from courses where fee is null and duration is not null

**Chapter 4: CREATING SAMPLE TABLES**

1. **\_\_\_\_\_\_\_\_\_\_ constraint can be used to implement business rules.**

CHECK

1. **\_\_\_\_\_\_\_\_\_\_ option of REFERENCES constraint is used to delete all child rows when parent row is being deleted.**

DELETE CASCADE

1. **Data dictionary view used to get information about constraints is \_\_\_\_\_\_\_\_\_\_\_\_\_.**

USER\_CONSTRAINTS

1. **When a table has a composite primary key, where is the PRIMARY KEY constraint defined?**

At table level using table constraint

1. **What is the relationship between COURSES and COURSE\_FACULTY table?**

to Many

1. **How do you get details of all CHECK constraints of all tables?**

select \* from user\_constraints where constraint\_type = 'C'

1. **Is it possible to create a constraint to prevent a date that is less than the system date?**

No. SYSDATE cannot be used in CHECK constraint

**Chapter 5: CHANGING STRUCTURE AND DATA**

1. **How do you add a check constraint to an existing column?**

Using ALTER TABLE command with ADD option

1. **How do you drop a constraint?**

Using constraint name in ALTER TABLE command with DROP option

1. **Is it possible to know the name of the constraint? If yes, how?**

Yes. We can use USER\_CONSTRAINTS table to get details of all constraints.

1. **How do you give primary key constraint if two or more columns are part of the primary key?**

We have to use PRIMARY KEY constraint at table level and mention columns that make up primary key.

1. **\_\_\_\_\_\_\_\_\_ command is used to mark a location in a transaction.**

SAVEPOINT

1. **What is ROLLBACK/UNDO segment and how is it used?**

ROLLBACK segment is used to store before image for changes. It is used to get back old data when changes are rolled back. It is also used to provide read consistency and enable flashback queries.

1. **When does a transaction begin and when does it end?**

A transaction begins when a session starts or when another transaction ends. It ends either with COMMMIT or ROLLBACK.

1. **If a row that is to be updated is already locked then what happens?**

The program which is trying to change a row that is locked will wait for row to be unlocked indefinitely.

1. **What happens if a row is updated and not committed or rolled back?**

Row might be committed if any event that causes auto commit takes place. It might also be rolled back, if any event causes Oracle to roll back changes.

1. **What is read consistency?**

Read consistency is where data provided by Oracle regarding a query is consistent to the point of starting query. That means, no changes made after query started will be provided.

**Chapter 6: ARITHMETIC AND DATE FUNCTIONS**

1. **\_\_\_\_\_\_ function can be used to subtract months from a date.**

ADD\_MONTHS() with negative number as second parameter will subtract that many months from the given date

1. **The return value of ROUND (2323.343, 2) is \_\_\_\_\_\_\_\_\_.**

2323.34

1. **To get the remainder of a division\_\_\_\_\_ function is used.**

MOD

1. **In date arithmetic \_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ and \_\_\_\_\_\_\_\_\_\_\_ operations are allowed.**

Adding a number, Subtracting a number, Subtracting a date

1. **\_\_\_\_\_\_\_\_\_\_\_\_ is the result of LAST\_DAY (SYSDATE) assuming SYSDATE is 24th August.**

31-AUG

1. **Which function can be used to set time portion of the DATE data type to 00:00:00, without effecting date portion?**

TRUNC(date)

1. **Display details of students who have joined in the last 4 months.**

select \* from students where months\_between(sysdate, dj) < 4

1. **Display ADMNO, NAME, DJ and number of days between current date and DJ for each student.**

select admno, name, dj, sysdate - dj nodays from students

1. **Display the first Sunday since batch with code b2 started.**

select next\_day(stdate,'Sun') from batches where code = 'b2'

1. **Display details of batches that started three or more months back.**

select \* from batches where months\_between(sysdate,stdate) >= 3

1. **Display the details of payments of last Monday.**

select \* from payments where trunc(dp) = trunc( next\_day(sysdate - 8, 'Mon'))

1. **\_\_\_\_\_\_\_\_\_\_ is the function to get number of years between two dates.**

MONTHS\_BETWEEN() / 12 will give number of years

**Chapter 7: STRING, CONVERSION AND MISCELLANEOUS FUNCTIONS**

1. **\_\_\_\_\_\_ function performs one to one character substitution.**

TRANSLATE

1. **\_\_\_\_\_\_ format option is used to get complete year spelled out in TO\_CHAR function.**

select to\_char(sysdate,'year') from dual

1. **\_\_\_\_\_ symbol is used to concatenate strings.**

||

1. **What happens if ‘replace string’ is not given for REPLACE function?**

It replaces source with nothing, effectively removing occurrences of source from string

1. **Can a NUMBER be converted to DATE? [Yes/No] \_\_\_\_\_.**

No

1. **How do you change the name of each student to uppercase in STUDENTS table?**

update students set name = upper(name)

1. **Display the names of the students who have more than 15 characters in the name.**

select name from students where length(name) > 15

1. **Display students’ firstname second and lastname first. For example, Luis Figo should be displayed as Figo Luis.**
2. select name,substr(name, instr(name,' ')), substr(name,1, instr(name,' '))

from students

1. **Display the details of the students who have more than 10 characters in the firstname.**

select \* from students where instr(name, ' ') > 10

1. **What is the result of AMOUNT – DISCOUNT if column DISCOUNT is null?**

NULL. Any expression that contains a null returns null.

1. **How do you get the position of 5th occurrence of letter ‘o’ in student’s name?**

Use INSTR(name,'o',1,5)

1. **What will be the result of select ‘10’ \* ‘20’ from dual?**

200 as Oracle automatically converts both strings to numbers

**Chapter 8: GROUPING DATA**

1. **\_\_\_\_\_\_\_ clause is used to select groups based on condition.**

HAVING

1. **Select count(\*) from students; Is it a valid query?**

Yes. It returns number of rows in STUDENTS table

1. **What is the correct order of GROUP BY, ORDER BY and WHERE clauses in SELECT?**

WHERE, GROUP BY and ORDER BY

1. **Display ADMNO of students who have paid more than twice.**

select admno from payments group by admno having count(\*) > 2

1. **Display average time (in days) taken to complete Oracle course.**

select avg(enddate - stdate) from batches where enddate is not null and course\_code = 'ora'

1. **Display faculty who can take more than 2 courses.**

select faculty\_code from course\_faculty group by faculty\_code having count(\*) > 2

1. **Display least course fee.**

select min(fee) from courses

1. **Display the number of months between first and last batches of course Java SE.**

select months\_between( max(stdate), min(stdate)) from batches where course\_code = 'jse'

1. **Display year, faculty and number of batches taken by faculty.**

select to\_char(stdate,'yyyy'), faculty\_code, count(\*) from batches group by to\_char(stdate,'yyyy'), faculty\_code order by 1,2

1. **Display the number of students joined in each month.**
2. select to\_char(dj,'mm-yyyy'), count(\*)
3. from students
4. group by to\_char(dj,'mm-yyyy')
5. order by 1
6. **Display the number of students joined in each month of the current year.**
7. select to\_char(dj,'mm-yyyy'), count(\*)
8. from students
9. where to\_char(dj,'yyyy') = to\_char(sysdate,'yyyy')
10. group by to\_char(dj,'mm-yyyy')
11. order by 1

**Chapter 9: JOINING TABLES**

1. **What is required to join two tables?**

A common column is required to join two tables

1. **What is meant by self-join?**

Self-join is where a table is joined to itself

1. **How do you qualify a column that is existing in two or more tables that are being joined?**

We qualify a column by using tablename.columnname. Ex: students.admno

1. **What is table alias? Is it stored anywhere?**

A table alias is what we use to shorten reference to table in query. It is not stored anywhere.

1. **What happens when you join two tables without any condition?**

Each row in first table is joined with each row of second table thus resulting in product of tables also called as Cartesian product

1. **Display admno, student name, pay date and amount paid.**
2. select admno, name, dp, amount
3. from payments natural join students
4. **Display admno, student name, batch code, stdate of batch and faculty name.**
5. select admno, s.name student, batch\_code, stdate, f.name faculty
6. from students s join batches b on ( s.batch\_code = b.code)
7. join faculty f on (f.code = b.faculty\_code)
8. **Display admno, student name, course name, stdate of batch and faculty code.**
9. select admno, s.name student, c.name Course, stdate, faculty\_code
10. from students s join batches b on ( s.batch\_code = b.code)
11. join courses c on (c.code = b.course\_code)

order by 1

1. **Display student name, course name, faculty code and enddate of all batches that were completed.**
2. select s.name student, c.name course, faculty\_code, enddate
3. from students s join batches b on (s.batch\_code = b.code)
4. join courses c on ( c.code = b.course\_code)

where enddate is not null

1. **Display students who have more number of characters in name than the student with admno 10.**
2. select s1.\*
3. from students s1 join students s2
4. on ( length(s1.name) > length(s2.name) )

where s2.admno = 10

1. **Display admno, student name, email, pay date and amount paid.**
2. select admno, name, email, dp, amount

from students join payments using(admno)

1. **In previous query include the details of students who haven’t paid anything so far.**
2. select admno, name, email, dp, amount

from students left outer join payments using(admno)

1. **Display the details of students who haven’t paid any amount so far.**
2. select admno, name, batch\_code, dj, phone, email
3. from students s left outer join payments using(admno)

where payments.amount is null

**Chapter 10: SUBQUERIES**

1. **A correlated subquery is executed for \_\_\_\_\_\_ number of times.**

N number of times where N is number of rows we have in main query

1. **Subquery nesting can be up to \_\_\_\_\_\_ levels.**

16

1. **What is the result of x > ANY (10,20), if x is 15? \_\_\_\_\_\_\_\_.**

TRUE

1. **Subquery always passes the result to the main-query [T/F] \_\_\_\_\_.**

TRUE

1. **Subquery can be used in VALUES clause of INSERT command [T/F] \_\_\_\_.**

TRUE

1. **Display details of courses taken by students who joined in the month of June 2015.**
2. select \* from courses
3. where code in
4. (select course\_code from batches where code in
5. (select batch\_code from students where to\_char(dj,'mm-yyyy') = '06-2015')

)

1. **Delete the details of students who haven’t paid anything so far.**
2. delete from students
3. where not exists
4. (select 1 from payments

where admno = students.admno)

1. **Display the details of courses for which there are more than 3 batches.**
2. select \* from courses
3. where code in
4. (select course\_code
5. from batches
6. group by course\_code

having count(\*) > 3)

1. **Display the details of course that has highest number of batches.**
2. select \* from courses where code in
3. (select course\_code
4. from batches group by course\_code
5. having count(\*) =
6. (select max(count(\*)) from batches
7. group by course\_code)
8. )
9. **Change the ENDDATE of batch B8 to the ENDDATE of most recently ended batch.**
10. update batches set enddate = (select max(enddate) from batches)

where code = 'b8'

1. **Display the details of students who haven’t paid total amount so far.**
2. select \* from students
3. where admno in
4. (select admno
5. from payments
6. group by admno
7. having sum(amount) <
8. (select fee from courses c join batches b on (b.course\_code = c.code)
9. and b.code = students.batch\_code)
10. )
11. **Display the details of payments made by students of Oracle batch started on 5-May-2015.**
12. select \* from payments
13. where admno in
14. (select admno
15. from students where batch\_code in
16. (select code from batches
17. where stdate = '15-may-2015' and course\_code ='ora')
18. )

**Chapter 11: VIEWS**

1. **What are the major applications of a view?**

Controlling Access, Query Simplification, Data Independence, and Presenting data in different forms

1. **A view can be used with ALTER TABLE command [T/F] ?\_\_\_\_\_\_\_ .**

FALSE

1. **The table on which a view is based is called as \_\_\_\_\_.**

Base Table

1. **When a table is dropped then all the views based on it will be dropped automatically [T/F]? \_\_\_\_\_\_.**

Views based on table being dropped are marked invalid, but not dropped.

1. **A view can be used to manipulate base table when it follows certain rules [T/F]? \_\_\_\_\_.**

TRUE

1. **Create a view, which contains the course name and number of students who have taken that course so far.**
2. create view course\_students
3. as
4. select c.name, count(\*) nostudents
5. from students s join batches b on (b.code = s.batch\_code) join courses c on (c.code = b.course\_code)

group by c.name;

1. **Create a view to provide the following: batch code, course name, faculty name, stdate, enddate and no. of days between enddate and stdate for all completed batches.**
2. create view completed\_batches
3. as
4. select b.code batchcode, c.name CourseName, f.name FacultyName, stdate, enddate, enddate - stdate nodays
5. from batches b join courses c on (b.course\_code = c.code) join faculty f on (f.code = b.faculty\_code)

where enddate is not null

1. **Create a view to get batch code, course code, faculty code, timings, start date and end date for all completed batches. Also ensure the changes made to base table through view are retrievable through view.**
2. create or replace view completed\_batches
3. as
4. select code, course\_code, faculty\_code, timings, stdate, enddate
5. from batches
6. where enddate is not null

with check option

**Chapter 12: INDEXING, SEQUENCES AND PSEUDO COLUMNS**

1. **Which constraints automatically create index?**

Primary Key and Unique

1. **What does ONLINE option in CREATE INDEX command do?**

ONLINE option allows index to be created without having to lock table

1. **How do you create an index on FACULTY\_CODE and COURSE\_CODE of BATCHES table?**

create index idx\_batches\_fcode\_ccode on batches (faculty\_code, course\_code)

1. **\_\_\_\_\_\_\_ option in CREATE SEQUENCE is used to generate numbers in reverse order.**

Negative value for INCREMENT BY option

1. **\_\_\_\_\_\_\_\_\_ is the pseudo column used to get the next available number from a sequence.**

NEXTVAL

1. **Create a sequence called REVERSE to generate numbers in the descending order from 10000 to 1000 with a decrement of 5.**
2. Create sequence reverse\_seq start with 10000 maxvalue 10000

minvalue 1000 increment by -5;

1. **Change the decrement value of sequence REVERSE (created earlier) to 2.**

Alter sequence reverse\_seq increment by -2

1. **What is the purpose of ROWID?**

ROWID is used internally by Oracle to identify each row uniquely

**Chapter 13: SECURITY**

1. **\_\_\_\_\_\_\_\_\_ command is used to change user password.**

ALTER USER

1. **Which object privilege allows user to create an index on the table?**

INDEX

1. **\_\_\_\_\_\_\_ option is used to grant a privilege along with permission to grant the privilege to other users.**

WITH GRANT OPTION

1. **A Role is \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.**

A collection of privileges

1. **\_\_\_\_\_\_ command is used to revoke a system privilege.**

REVOKE

1. **\_\_\_\_\_\_\_\_\_\_ data dictionary view may be used to list tables that a user can access.**

ALL\_TABLES

1. **\_\_\_\_\_\_ data dictionary view is used to know the list of tables owned by the current user.**

USER\_TABLES

1. **\_\_\_\_ is the synonym for USER\_CATALOG data dictionary.**

CAT

1. **Grant UPDATE privilege on STDATE column of BATCHES table to user PRANEETH with permission to grant the privilege to others.**

grant update(stdate) on batches to praneeth with grant option

1. **Create a role and assign a few privileges to that role. Assign the role to user PRANAV.**
2. create role programmer;
3. grant all on students to programmer;
4. grant select on courses to programmer;

grant programmer to pranav;

1. **Enable the role LEADER.**

set role leader

1. **Display the table name, column name of all columns that you can access.**
2. select table\_name, column\_name
3. from all\_tab\_columns

order by 1,2

**Chapter 14: REPORT GENERATION USING SQL\*PLUS COMMANDS**

1. **\_\_\_\_\_ system variable is used to automatically commit changes made to database.**

AUTOCOMMIT

1. **Which command is used to change the heading of a column?**

COLUMN

1. **\_\_\_\_\_ is the numeric format to display number 12345 as 12,345.00.**

99,999.00

1. **\_\_\_\_\_ command is used to display the values of system variables.**

SHOW

1. **Display the title ‘Sales Report’ at the top of each printed page by aligning it to center.**

TTITLE center 'Sales Report'

1. **Define the following attributes for column NAME. Heading should be ‘Course Name’. Allow only first 20 characters to be displayed. Display ‘NONE’ if the value is null.**

COLUMN TITLE heading 'Book Title' format a20 null 'None'

1. **\_\_\_\_\_\_\_\_ variable is used to display the name of the user in the title in TTITLE command.**

SQL.USER

1. **\_\_\_\_\_\_\_\_ and \_\_\_\_\_\_\_\_\_ variables are used to set the dimensions of the report page.**

PAGESIZE and LINESIZE

1. **What is the purpose of LABEL option of COMPUTE command?**

to display text for computed value

1. **How do you turn off top title?**

TITILE OFF

**Chapter 15: ADVANCED FEATURES**

1. **\_\_\_\_\_\_\_\_\_\_\_\_\_ function returns number of matches for the given regular expression.**

REGEXP\_COUNT

1. **How many times MERGE scans source table?**

Only once

1. **What is the difference between INSERT ALL and INSERT FIRST?**

INSERT FIRST stops after first condition is satisfied, INSERT ALL evaluates all conditions

1. **What is meant by regular expression “[^0-9]$” ?**

for non-digits at the end of string

1. **Which clause of SELECT is used for flashback query?**

AS OF TIMESTAMP

1. **What does \2 represent in REGEXP\_REPLACE function?**

\2 represents second group in regular expression in source

**Chapter 16: INTRODUCTION TO PL/SQL**

1. **In \_\_\_\_\_\_\_ part of PL/SQL block errors are handled.**

Exception Handling

1. **\_\_\_, \_\_\_\_ and \_\_\_\_ are the valid values for BOOLEAN data type.**

True, False, Null

1. **The part of Oracle Server that executes SQL commands is called as \_\_\_\_\_\_\_.**

SQL Statement Executor

1. **\_\_\_\_\_\_ is an example of Oracle tool that contains PL/SQL engine.**

Oracle Forms

1. **\_\_\_\_\_ is the operator for exponentiation.**

\*\*

1. **\_\_\_\_\_ is used for commenting a single line.**

--

1. **Write PL/SQL block to change the DURATION of ORA course to the duration of JSE course.**
2. declare
3. v\_duration courses.duration%type;
4. begin
5. select duration into v\_duration
6. from courses
7. where code = 'jse';
8. update courses set duration = v\_duration
9. where code = 'ora';

end;

1. **Insert a new row into COURSE\_FACULTY table with the following details: Course Name is Java SE, Faculty name is Craig Walls, and SINCEWHEN column is 2012.**
2. declare
3. v\_faculty faculty.code%type;
4. v\_course courses.code%type;
5. begin
6. select code into v\_course
7. from courses
8. where name = 'Java SE';
9. select code into v\_faculty
10. from faculty
11. where name = 'Craig Walls';
12. insert into course\_faculty values( v\_faculty,v\_course,2012);
13. end;

**Chapter 17: CONTROL STRUCTURES**

1. **Write a PL/SQL block to decrease the duration to 35 for course with code ora, if more than 2 batches have started in the last two months.**
2. declare
3. v\_count number(2);
4. begin
5. select count(\*) into v\_count
6. from batches
7. where course\_code = 'ora' and months\_between(sysdate,stdate) <= 2;
8. if v\_count > 2 then
9. update courses set duration = 35
10. where code = 'ora';
11. end if;
12. end;
13. **Write a PL/SQL block to insert a new row into PAYMENTS table with the following data:**
    * **INVNO is to be taken from INVNO\_SEQ sequence.**
    * **ADMNO of student with the name George Michael.**
    * **Date of payment is previous Monday.**
    * **Amount is the balance amount to be paid by the student.**
14. declare
15. v\_admno students.admno%type;
16. v\_course\_fee courses.fee%type;
17. v\_fee\_paid courses.fee%type;
18. begin
19. select admno into v\_admno
20. from students
21. where name = 'George Michael';
22. select sum(amount) into v\_fee\_paid
23. from payments
24. where admno = v\_admno;
25. select fee into v\_course\_fee
26. from courses
27. where code = (select course\_code from batches where code = (select batch\_code from students where admno = v\_admno) );
28. insert into payments values (invno\_seq.nextval, v\_admno, next\_day(sysdate-8,'Mon'), v\_course\_fee - v\_fee\_paid);
29. end;
30. **Display how many students have joined in each month in the current year.**
31. set serveroutput on
32. declare
33. v\_count number(3);
34. begin
35. for i in 1..12
36. loop
37. select count(\*) into v\_count
38. from students
39. where to\_char(dj,'yyyy') = to\_char(sysdate,'yyyy') and to\_char(dj,'mm') = i;
40. dbms\_output.put\_line( i || ' - ' || v\_count);
41. end loop;
42. end;

**Chapter 18: EXCEPTION HANDLING**

1. **Look for student number 1008. If it is not found then display a suitable error message on the screen otherwise display student name and total amount paid by student so far.**
2. declare
3. v\_total\_paid payments.amount%type;
4. v\_name students.name%type;
5. begin
6. select name into v\_name
7. from students
8. where admno = 1008;
9. select nvl(sum(amount),0) into v\_total\_paid
10. from payments
11. where admno = 1008;
12. dbms\_output.put\_line('Total amount paid by ' || v\_name || ' is ' || to\_char(v\_total\_paid));
13. exception
14. when no\_data\_found then
15. dbms\_output.put\_line('Sorry! Student 1008 not found!');

end;

1. **\_\_\_\_\_\_\_\_\_ statement is used to re-raise an exception.**

RAISE

1. **\_\_\_\_\_\_\_\_\_ function is used to get error message of the most recent error.**

SQLERRM

1. **How do you associate an Oracle error with a user-defined error?**

Using PRAGMA EXCEPTION\_INIT procedure

1. **When UPDATE command does not update any row then which of the following will happen?**

**a. NO\_DATA\_FOUND exception occurs**

**b. INVALID\_UPDATE exception occurs**

**c. No exception is raised**

C

1. **When an exception is not handled in the current block, which of the following happens?**

**a. It results in error and terminates the block**

**b. It is propagated to outer block**

**c. It is ignored**

B

**Chapter 19: CURSOR HANDLING**

1. **Which attribute is used to find out how many rows were fetched from cursor so far?**

ROWCOUNT

1. **Can we use ISOPEN attribute with implicit cursor?**

YES. But it always returns false

1. **How can we know whether the most recent DML operation has affected any row?**

Using SQL%FOUND

1. **How do you declare an input argument for the cursor and how do you pass value to it?**
2. declare
3. cursor course\_batches (p\_course varchar2) is
4. select \* from batches
5. where course\_code = p\_course;
6. begin
7. -- pass ora as parameter to cursor
8. open course\_batches('ora');
9. -- remaining process
10. end;
11. **What is the use of CURRENT OF clause in DELETE and UPDATE commands?**

CURRENT OF clause in DELETE and UPDATE commands is used to affect row in the table that corresponds to current row in cursor

1. **Display 10th to 15th students in the order of joining date.**
2. declare
3. cursor students\_cursor is
4. select \* from students
5. order by dj;
6. begin
7. for student\_rec in students\_cursor
8. loop
9. if students\_cursor%rowcount >= 10 then
10. dbms\_output.put\_line( student\_rec.name);
11. exit when students\_cursor%rowcount = 15;
12. end if;
13. end loop;
14. end;

**Chapter 20: PROCEDURES, FUNCTIONS AND PACKAGES**

1. **\_\_\_\_\_ command is used to display errors that occurred during compilation of a stored procedure.**

SHOW ERRORS

1. **\_\_\_\_\_\_\_ view provides information about stored procedures.**

USER\_SOURCE

1. **\_\_\_\_\_\_\_ option is used to specify that a parameter is both input and output parameter.**

INOUT

1. **What is the command used to compile a procedure explicitly?**

ALTER PROCEDURE with COMPILE option

1. **Create a function to take batch code and return the number of students in the batch.**
2. create or replace function students\_count(p\_batch varchar2)
3. return number
4. is
5. v\_count number(3);
6. begin
7. select count(\*) into v\_count
8. from students
9. where batch\_code = p\_batch;
10. return v\_count;
11. end;
12. **Create a function to return the first missing admission number. If no admission number is missing then return the highest admission number + 1.**
13. create or replace function next\_admno
14. return number
15. is
16. v\_count number(1);
17. v\_admno students.admno%type;
18. begin
19. -- start with minimum admno + 1
20. select min(admno) + 1 into v\_admno
21. from students;
22. loop
23. select count(\*) into v\_count
24. from students
25. where admno = v\_admno;
26. if v\_count = 0 then
27. return v\_admno;
28. end if;
29. v\_admno := v\_admno + 1;
30. end loop;
31. end;
32. **Create a function to take faculty code and return the number of batches the faculty can handle.**
33. create or replace function faculty\_course\_count(p\_faculty varchar2)
34. return number
35. is
36. v\_count number(3);
37. begin
38. select count(\*) into v\_count
39. from course\_faculty
40. where faculty\_code = p\_faculty;
41. return v\_count;
42. end;
43. **Create a procedure to take course code and return minimum and maximum duration of batches of that course.**
44. create or replace procedure course\_batches\_duration(p\_course varchar2, p\_min out number, p\_max out number)
45. is
46. begin
47. select min( trunc(enddate-stdate)), max( trunc(enddate-stdate)) into p\_min, p\_max
48. from batches
49. where course\_code = p\_course and enddate is not null;
50. end;
51. The following code shows how to call the above procedure.
52. declare
53. v\_min number(2);
54. v\_max number(2);
55. begin
56. course\_batches\_duration ('ora', v\_min,v\_max);
57. dbms\_output.put\_line( v\_min || '-' || v\_max);
58. end;
59. **Create a package to contain the following functions.**

**a. Function GET\_STATUS – takes batch code and returns S - if batch is yet to start, C – if batch is completed or R – if batch is currently running.**

**b. Function GET\_TOTAL\_AMOUNT – returns the total amount collected from the given batch code.**

create or replace package batch\_actions

is

function get\_status(p\_batch varchar2) return varchar2;

function get\_total\_amount (p\_batch varchar2) return number;

end;

create or replace package body batch\_actions

is

function get\_status(p\_batch varchar2) return varchar2

is

v\_stdate date;

v\_enddate date;

begin

select stdate, enddate into v\_stdate, v\_enddate

from batches

where code = p\_batch;

if v\_stdate > sysdate then

return 'S';

elsif v\_enddate is null then

return 'R';

else

return 'C';

end if;

end;

function get\_total\_amount (p\_batch varchar2) return number

is

v\_total number(6);

begin

select nvl(sum(amount),0) into v\_total

from payments

where admno in ( select admno from students where batch\_code = p\_batch);

return v\_total;

end;

end;

**Chapter 21: DATABASE TRIGGERS**

1. **Which data dictionary view contains information about triggers?**

USER\_TRIGGERS

1. **How many before triggers can we create?**

6 different before triggers can be created on a single table. One before insert, One before delete, and One before update for row-level and the same three for statement-level.

1. **Is it possible to create two or more triggers for the same event (BEFORE INSERT)?**

YES

1. **What is the default type of trigger? [Statement/Row]**

Statement. Unless FOR EACH ROW is given, a trigger is considered to be statement-level trigger

1. **Create a trigger to prevent any increase to FEE column of COURSES table if the increase is more than 50% of the existing course fee.**
2. create or replace trigger trg\_fee\_hike\_check
3. before update of fee
4. on courses
5. for each row
6. begin
7. if :new.fee - :old.fee > :old.fee \* 0.50 then
8. raise\_application\_error(-20555,'Fee cannot be increased by more than 50% of existing fee');
9. end if;
10. end;
11. **Create a trigger to prevent all deletions from COURSES table between 9p.m to 9 a.m.**
12. create or replace trigger trg\_prevent\_courses\_deletion
13. before delete
14. on courses
15. begin
16. if to\_char(sysdate,'hh24') < 9 or to\_char(sysdate,'hh24') >= 21 then
17. raise\_application\_error(-20222,'No deletions can be made before 9AM and after 9PM');
18. end if;
19. end;

**Chapter 22: RECORDS, COLLECTIONS AND LOBS**

1. **Which keyword is used to refer to a row in nested table?**

VALUE

1. **Create a record type – DEPENDENT\_TYPE, which contains two fields – dependent name and age. Create EMPLOYEES table with id, name, designation and dependents, which is a nested table of DEPENDENT\_TYPE.**
2. create or replace type dependent\_type as object
3. (
4. name varchar2(50),
5. age number(3)
6. );
7. create or replace type dependents\_table as table of dependent\_type;
8. create table employees
9. ( id number(5),
10. name varchar2(50),
11. desg varchar2(50),
12. dependents dependents\_table
13. )
14. nested table dependents store as dependents\_tab;

insert into employees values (1, 'Stagner','Programmer', dependents\_table ( dependent\_type('Lucy',30)))

1. **Create APPLICANTS table with the following columns.**

**Column    Type**

**Name      VARCHAR2(20)**

**Resume    CLOB**

**Photo     BFILE**

create table applicants

( name varchar2(20),

resume clob,

photo bfile

)

1. **Insert a row into APPLICANTS table with the following details.**

**NAME - ‘Nike’, RESUME - ‘Subjects: Oracle, Java, jQuery’, and PHOTO - ‘nike.jpg’ which is directory referred by PHOTOS directory alias.**

insert into applicants values('Nike','Subjects : Oracle, Java, jQuery',bfilename('PHOTOS','nike.jpg') )

1. **Write a PL/SQL block to find out whether the pattern jQuery exists in the RESUME column of applicant ‘Nike’. If found display the starting position otherwise display error message using DBMS\_OUTPUT package.**
2. set serveroutput on
3. declare
4. v\_resume clob;
5. pos number(3);
6. begin
7. select resume into v\_resume
8. from applicants
9. where name = 'Nike';
10. pos := dbms\_lob.instr(v\_resume,'jQuery',1,1);
11. if pos > 0 then
12. dbms\_output.put\_line('Found at : ' || pos);
13. else
14. dbms\_output.put\_line('jQuery Not Found');
15. end if;

end;

1. **Check whether the physical file for applicant Nike is existing on the disk.**
2. declare
3. v\_photo bfile;
4. begin
5. select photo into v\_photo
6. from applicants
7. where name = 'Nike';
8. if dbms\_lob.fileexists(v\_photo) = 1 then
9. dbms\_output.put\_line('File is existing');
10. else
11. dbms\_output.put\_line('File is NOT existing');
12. end if;

end;

1. **Insert a new row into APPLICANTS with the following details. NAME - ‘Bob’, RESUME - Empty, PHOTO - Empty.**

insert into applicants values('Bob',empty\_clob(), null);

1. **Change the value of PHOTO column of Bob to ‘bob2016.jpg’ in PHOTOS directory.**
2. create directory photos as 'd:\photos';
3. update applicants set photo = bfilename('PHOTOS','bob2016.jpg')

where name = 'Bob';

1. **Display the length of RESUME column of applicant ‘Nike’.**
2. select dbms\_lob.getlength(resume)
3. from applicants

where name = 'Nike'

**Chapter 23: DYNAMIC SQL**

1. **\_\_\_\_\_\_\_\_ option is used to pass bind arguments to placeholders.**

USING

1. **When there are 2 placeholders used three times in the command then how many bind arguments are to be passed?**

3 values must be passed

1. **Create a function that takes table name and a condition and returns the number of rows in the table that satisfy the given condition.**
2. -- on error returns null
3. create or replace function get\_row\_count(tn varchar2, cond varchar2)
4. return number
5. is
6. v\_count number(5);
7. begin
8. execute immediate 'select count(\*) from ' || tn || ' where ' || cond
9. into v\_count;
10. return v\_count;
11. exception
12. when others then
13. return null;
14. end;
15. select get\_row\_count('students', 'admno > 5') from dual;

**Chapter 24 UTILITIES**

1. **\_\_\_\_\_\_\_\_\_\_ parameter is used in EXPDP to copy only table data without definition.**

CONTENT=DATA\_ONLY

1. **\_\_\_\_\_\_\_\_\_\_ parameter of IMPDP can be used to append data from source file to tables in database.**

TABLE\_EXISTS\_ACTION=APPEND

1. **\_\_\_\_\_\_\_\_\_\_ file contains records that could not be loaded by SQL\*Loader.**

BAD File

1. **How do you specify that you want to export from BATCHES table where ENDDATE is not null to EXPDP command?**
2. TABLES=batches
3. QUERY=batches:"WHERE enddate is not null"
4. **What is the difference between loading data using SQL\*Loader and creating external table?**

In case of external table, data is not loaded into Database. SQL\*Loader loads data into Database physically