SECTION -1

1. DDL FOR DEPARTMENT TABLE

CREATE TABLE department (

department\_ID number NOT NULL,

department\_name varchar(30) ,

department\_block\_number number,

    PRIMARY KEY (department\_ID)

);

2) DDL FOR STUDENT TABLE

CREATE TABLE student (

student\_ID number NOT NULL,

student\_name varchar(30) ,

Address varchar(40),

city varchar(30),

department\_ID number,

     PRIMARY KEY (student\_ID),

    foreign KEY (department\_ID) references

 department(department\_ID)

);

3) DDL FOR STAFF TABLE

CREATE TABLE STAFF (

staff\_ID number NOT NULL,

staff\_name varchar(30) ,

department\_ID number,

    PRIMARY KEY (staff\_ID),

    foreign KEY (department\_ID) references

 department(department\_ID)

);

4) DDL FOR SUBJECT TABLE

CREATE TABLE Subject (

subject\_ID number NOT NULL,

subject\_name varchar(30) ,

subject\_code varchar(10) ,

staff\_ID number,

    PRIMARY KEY (subject\_ID),

    foreign KEY (staff\_ID) references Staff (staff\_ID)

);

5) DDL FOR MARK TABLE

CREATE TABLE Mark (

value\_mark number,

Subject\_id number,

Student\_id number ,

PRIMARY KEY (subject\_ID ,student\_id),

     foreign KEY (Subject\_id) references Subject (Subject\_id),

     foreign KEY (Student\_id) references Student (Student\_id)

);

6) Add a constraint by writing a query to add a not null constraint to the column staff\_name in the staff table.

ALTER TABLE Staff MODIFY Staff\_name NOT NULL;

7) Add a column by writing a query named emailid of type varchar (20) to the student table

ALTER TABLE student ADD emailid varchar(20);

8) Modify the size of the type of field emailid on the student table by writing a query to change it to varchar(50

ALTER TABLE student MODIFY emailid varchar(50);

9) Remove the emailid column on the student table by writing a query.

   ALTER TABLE student DROP COLUMN emailid ;

**Error report -**

**SQL Error: ORA-12988: cannot drop column from table owned by SYS**

**12988. 00000 -  "cannot drop column from table owned by SYS"**

**\*Cause:    An attempt was made to drop a column from a system table.**

**\*Action:   This action is not allowed**

SECTION - 2

10) DONE

11) DONE - MARK table data needs to be inserted after all the tables are inserted.

SECTION - 3

12) Update a record by writing a query to update the subject\_name in the subject table from Sales to Computer Science and subject\_code from 1842 to 1919.

UPDATE SUBJECT

SET subject\_name = 'Computer Science', Subject\_code = 1919

WHERE subject\_name = 'Sales' AND Subject\_code = 1842;

SECTION - 4

13) Delete the row from the subject table where subject name is Accounting by writing the appropriate query.

DELETE FROM Subject WHERE subject\_name = 'Accounting' ⇒

Error report -

SQL Error: ORA-02292: integrity constraint (SYS.SYS\_C008462) violated - child record found

02292. 00000 - "integrity constraint (%s.%s) violated - child record found"

\*Cause:    attempted to delete a parent key value that had a foreign

          dependency.

\*Action:   delete dependencies first then parent or disable constraint.

**DELETE CASCADE WILL NOT GIVE ABOVE ERROR**

Section - 5

14) Display the names of the department in the college by writing the appropriate query. Please note that these must be displayed in ascending order.

SELECT department\_name FROM department ORDER BY department\_name ASC;

15) Display the names of the departments where departments block number is between 3 and 10 by writing the appropriate query.

SELECT department\_name FROM department WHERE department\_block\_number BETWEEN 3 AND 10;

16) Display the names of all the students in the college by writing the appropriate query. Please note these must be displayed in ascending order

SELECT student\_name FROM student ORDER BY student\_name ASC;

Section - 6

17) Display the names of the students who are from Chicago, Taylor and San Jose. Please note these must be displayed in ascending order of their respective id.

SELECT student\_name FROM student WHERE city IN (“Chicago”, “Taylor”, “San Jose”) ORDER BY student\_id ASC;

18) Writing the correct query, display the address and city of the students table give the alias as Address\_Student.

SELECT concat (ADDRESS, CITY) AS ADDRESS\_student FROM student;

19) Display all of the student’s names whose names are of 6 characters in length by writing the correct query.

SELECT student\_name FROM student WHERE LENGTH(student\_name) = 6;

Section 7

20) Display the blocknumber and number of departments in each block by writing the correct query that is ordered by block id. Make sure it is displayed as count (department\_name)

SELECT department\_block\_number, COUNT(department\_name) FROM department GROUP BY department\_block\_number;

21) Display the number of students in the college by writing the correct query and give an alias as stud\_count.

SELECT count(\*) AS stud\_count FROM student;

Section 8

22) Display the names of the department and the student count in each department by writing the correct query. The student count in each department must be in ascending order based on the department name and an alias of student\_count for the student count.

SELECT d.department\_name , count(s.student\_id) as student\_count

FROM department d INNER JOIN student s

ON d.department\_id = s.department\_id

GROUP BY d.department\_name

ORDER BY d.department\_name ASC;

23) Display the Student\_Name from STUDENT and the Subject\_name from SUBJECT where the Subject\_code from SUBJECT is greater than 1600.

SELECT student.student\_name, subject.subject\_name

FROM student CROSS JOIN subject

WHERE subject.subject\_code > 1600;

24) Display the Student\_Name from STUDENTS and the Subject\_name from SUBJECT where the value on MARK table is less 3.

Three tables join

SELECT student.Student\_Name, Subject.Subject\_Name

FROM

   student

INNER JOIN

   Mark ON student.student\_id = Mark.student\_id

INNER JOIN

   subject ON subject.subject\_id = MARK.subject\_id

WHERE MARK.value\_mark < 3;

Section 9 – SUBQUERIES

25) Display the block number in which the maximum number of departments is located by writing the correct sub-query.

SELECT department\_block\_number, COUNT(department\_block\_number)

FROM department  GROUP BY department\_block\_number

HAVING COUNT (department\_block\_number)=(

SELECT MAX(deptcount)

FROM (

SELECT department\_block\_number, COUNT(department\_block\_number) deptcount

FROM department

GROUP BY department\_block\_number));

26) Display the names of the staff who are not handling any subjects by ascending order using the correct sub-query.

SELECT staff\_id, staff\_name

FROM staff

WHERE staff\_id NOT IN (select staff\_id FROM subject);

Section 10 – FUNCTIONS

27) Write a function that takes department\_id as the input and returns the department\_name. Use the function name below: Function name: find\_dept\_name

create or replace function find\_dept\_name(i\_department\_id number)

return varchar2

as

l\_dname department.department\_name%type;

begin

 select department\_name into l\_dname

 from department

 where department\_id = i\_department\_id;

return l\_dname;

end;

28) Write a function that takes department id as the input and returns the block number. Use the function name below: Function name: find\_dept\_block

create or replace function find\_dept\_block(i\_department\_id number)

return varchar2

as

l\_dblockno department.department\_block\_number%type;

begin

 select department\_block\_number into l\_dblockno

 from department

 where department\_id = i\_department\_id;

return l\_dblockno;

end;

29) Write a function that takes the staff id as the input and returns the staff name. Use the function name below: Function name: find\_staff\_name

create or replace function find\_staff\_name(i\_staff\_id number)

return varchar2

as

l\_sname staff.staff\_name%type;

begin

 select staff\_name into l\_sname

 from staff

 where staff\_id = i\_staff\_id;

return l\_sname;

end;

Section 11 - Triggers

30) Create a trigger with the name 'trigger\_department\_af\_update' which will display “DEPARTMENTS table has been updated” after an attempt to update the DEPARTMENTS has been made. Trigger name: trigger\_department\_af\_update

create or replace trigger trigger\_department\_af\_update

after update on department

begin

dbms\_output.put\_line('DEPARTMENTS table has been updated');

end;

**Error report -**

**ORA-04089: cannot create triggers on objects owned by SYS**

**04089. 00000 -  "cannot create triggers on objects owned by SYS"**

**\*Cause:    An attempt was made to create a trigger on an object owned by SYS.**

**\*Action:   Do not create triggers on objects owned by SYS.**

31) Create a trigger with the name ‘trigger\_department\_bf\_delete’ which will display “A row has been deleted from DEPARTMENT” before an attempt to delete a row is execute on DEPARTMENT. Trigger name : trigger\_department\_bf\_delete

create or replace trigger trigger\_department\_bf\_delete

before delete on department

begin

dbms\_output.put\_line('A row has been deleted from department');

end;

**Error report -**

**ORA-04089: cannot create triggers on objects owned by SYS**

**04089. 00000 -  "cannot create triggers on objects owned by SYS"**

**\*Cause:    An attempt was made to create a trigger on an object owned by SYS.**

**\*Action:   Do not create triggers on objects owned by SYS.**

Section 12 - Views and Index

32) Create an Index command that will reference all of the students names containing the letter ‘b’ on the Student table.

CREATE INDEX idx\_sname ON student (student\_name);

33) Create a view from the staff table that will display staff names.

CREATE VIEW staff\_view AS SELECT staff\_name FROM staff;

Section 13 - Cursors

36) Declare an explicit cursor using the STUDENT table to select of column. Fetch the rows using a loop and display each data retrieved.

DECLARE

 CURSOR stu\_cur IS SELECT \* FROM student;

 stu\_rec stu\_cur%ROWTYPE;

BEGIN

 open stu\_cur;

 LOOP

   fetch stu\_cur into stu\_rec;

   exit when stu\_cur%notfound;

   if stu\_cur%isopen then

   dbms\_output.put\_line(stu\_rec.student\_id || ' ' || stu\_rec.student\_name || ' ' || stu\_rec.address || ' ' || stu\_rec.city || ' ' || stu\_rec.department\_id);

    -- dbms\_output.put\_line(stu\_rec.student\_id || stu\_rec.student\_name || stu\_rec.address || stu\_rec.city || stu\_rec.department\_id);

   end if;

 END LOOP;

 close stu\_cur;

END;

Section 14- PACKAGES AND PROCEDURES

37) Define a package give it the name of College. The definition should include a procedure give it the name of select\_departments. It should also include a function that takes a argument of type number and returns a variable of type VARCHAR, give it the name of select\_student.

create or replace package College as

procedure select\_departments;

function select\_student (stu\_num in number) return varchar;

end college;

38) Implement the body of College department. the select\_departments procedure should display all column from DEPARTMENT. The select\_student function should take the id of the student and return that student name.

create or replace package body College as

PROCEDURE select\_departments AS

cursor dep\_cur is select \* from department;

dep\_rec department%rowtype;

BEGIN

open dep\_cur;

loop

fetch dep\_cur into dep\_rec;

exit when dep\_cur%notfound;

dbms\_output.put\_line(dep\_rec.department\_id || ' ' || dep\_rec.department\_name || ' ' || dep\_rec.department\_block\_number);

end loop;

close dep\_cur;

END select\_departments;

FUNCTION select\_student(stu\_num NUMBER)

RETURN VARCHAR

AS

stu\_name student.student\_name%TYPE;

BEGIN

SELECT student\_name INTO stu\_name

FROM student

WHERE student\_id = stu\_num;

RETURN stu\_name;

END select\_student;

end college;