**FREDERIC MOKOLO**

**Language: Oracle SQL**

**Number of Questions: 38**

**Duration: 3 hours**

**Programming Assignment**

**Assignment 1 – Database Creation and Querying Exercises**

**Objectives:**

**In this assignment you will create a database along with tables for which you will Write SQL queries for the instructions specified. These will include addition of constraints, modifications, deletions, insertions and updates. You will also be asked to perform Oracle SQL Joins, sub-queries, functions, triggers, views and indexes as well as packages and procedures. This will be used to assess your overall knowledge of Oracle SQL.**

**STRUCTURE REQUIREMENTS:**

* **Ensure proper syntax and naming convention for example: ensuring the table and column names are all lower case.**
* **Set up the primary keys for each table with constraints listed.**
* **Because you will not be asked to create an actual database in your environment, it is important that your answers clearly illustrate that you understand the questions being asked as well as how to properly structure your answers.**

**REPORTING REQUIREMENTS:**

* **Make sure that you list the section number for each question and title/headings as well as question numbers when submitting your answer.**
* **Please enter all of your answers in a Microsoft Word Doc.**

**SUBMISSION REQUIREMENTS:**

**Submissions will be in the Oracle material GitHub repository.**

**Activities:**

**Section 1-**

**Creating Tables:**

**Referring to the schema above for reference and using Oracle SQL, create the tables identified below using the following order:**

**TABLES TO CREATE:**

**1-department (department\_id number PK, department\_name varchar(30), department\_block\_number number)**

CREATE TABLE department

(

department\_id NUMBER PRIMARY KEY,

department\_name VARCHAR (30),

department\_block\_number VARCHAR(30),

department\_block\_name NUMBER);

);

**2-student (student\_id number, student\_name varchar(30), address varchar (40), city varchar(30), department\_id number FK)**

CREATE TABLE student

(

student\_id NUMBER PRIMARY KEY,

student\_name VARCHAR (30),

address VARCHAR (40),

city VARCHAR(30),

department\_ID NUMBER,

CONSTRAINT fk\_department\_id FOREIGN KEY(department\_id) REFERENCES department(department\_id)

);

**3-staff (staff\_id number, staff\_name varchar(30), department\_id number FK)**

CREATE TABLE staff

(

staff\_id NUMBER PRIMARY KEY,

staff\_name VARCHAR (30),

department\_id NUMBER,

CONSTRAINT fk\_new\_department\_id FOREIGN KEY(department\_id) REFERENCES department(department\_id)

);

**4- subject (subject\_id number PK, student\_name varchar (30), student\_code varchar(10), staff\_id number FK)**

CREATE TABLE subject

(

subject\_id NUMBER PRIMARY KEY,

subject\_name VARCHAR(30),

subject\_code VARCHAR(10),

staff\_id NUMBER,

CONSTRAINT fk\_staff\_id FOREIGN KEY(staff\_id) REFERENCES staff(staff\_id)

);

**5- mark (value number, subject\_id number PK FK, student\_id number PK FK)**

CREATE TABLE mark

(

value\_mark NUMBER,

subject\_id NUMBER,

student\_id NUMBER,

CONSTRAINT pk\_STUD PRIMARY KEY (subject\_id,student\_id),

CONSTRAINT fk\_stud\_li FOREIGN KEY(subject\_id) REFERENCES subject(subject\_id),

CONSTRAINT fk\_stud\_la 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.**

CREATE TABLE staff

(

staff\_id NUMBER PRIMARY KEY,

staff\_name VARCHAR (30),

department\_id NUMBER,

ALTER TABLE staff ADD (staff\_name Char(50) NOT NULL),

CONSTRAINT fk\_new\_department\_id FOREIGN KEY(department\_id) REFERENCES department(department\_id)

);

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

CREATE TABLE student

(

student\_id NUMBER,

student\_name VARCHAR (30),

address VARCHAR (40),

city VARCHAR(30),

department\_ID NUMBER,

CONSTRAINT fk\_department\_id FOREIGN KEY(department\_id) REFERENCES department(department\_id)

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

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

**Section 2-**

**Inserting Into Tables**

**10 - Download the following excel sheet in LMS under Documents -> Week 3 -> Oracle hands on:**

* **DEPARTMENT**
* **STAFF**
* **MARK**
* **SUBJECT**
* **STUDENT**

**11 - Insert those excel sheets to their corresponding tables**

**Section 3-**

**Updating Records**

**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\_code=1842;

**Section 4-**

**Deleting Records**

**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';

**Section 5-**

**Basic Selection of Records**

**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-**

**Selecting Single Rows**

**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 = 'San Jose' or city= 'Chicago' or city= 'Taylor'

Order by student\_name asc;

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

SELECT 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-**

**Selecting Groups**

**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

Order 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 std\_count

From student;

**Section 8-**

**SQL Joins**

**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 DEPARTMENT\_NAME, count(\*) AS student\_count

FROM department

JOIN student on student.DEPARTMENT\_ID = department.DEPARTMENT\_ID

GROUP BY department.department\_name

ORDER BY 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\_name, subject\_name

From student

Join mark on mark.STUDENT\_ID = student.STUDENT\_ID

join subject on subject.SUBJECT\_ID = mark.SUBJECT\_ID and subject.subject\_code > 1600

Order by student\_name asc;

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

SELECT student\_name, subject\_name, mark.value\_mark

From student

Join mark on student.STUDENT\_ID = mark.STUDENT\_ID

Join subject on mark.SUBJECT\_ID = subject.subject\_id and mark.value\_mark <3;

**Section 9 –**

**Selecting Sub-Queries**

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

SELECT department\_block\_number

from (select department\_block\_number, count(department\_block\_number)

From department

Group by department\_block\_number

Order by count(department\_block\_number) DESC);

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

SELECT staff\_name

FROM staff

WHERE staff.staff\_id NOT IN (select distinct subject.STAFF\_ID from subject)

ORDER BY STAFF\_NAME ASC;

**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(dpt\_id int)

Return varchar2

Is depName varchar2 (30);

Begin

Select department\_name

Into depName

From department

Where department.DEPARTMENT\_ID = dpt\_id;

Return (depName);

End find\_dept\_name;

**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 (dept\_id in int)

Return number

Is deptBlock number;

Begin

Select department\_block\_number

Into deptBlock

From department

Where dept\_id = department\_id;

Return (deptBlock);

End find\_dept\_block;

**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 (m\_staffid number)

Return varchar2

Is staffName varchar2 (30);

Begin

Select staff\_name

Into staffName

From staff

Where m\_staffid = staff\_id;

Return staffName;

End find\_staff\_name;

**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\_deparment\_af\_delete’

After update on DEPARTMENT

FOR EACH ROW

Begin

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

End;

**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

FOR EACH ROW

Begin

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

End;

**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 index Stud

on student(student\_name);

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

Create view v\_staffNames 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.**

**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.**

**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.**