**Assignment Description**

**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:**

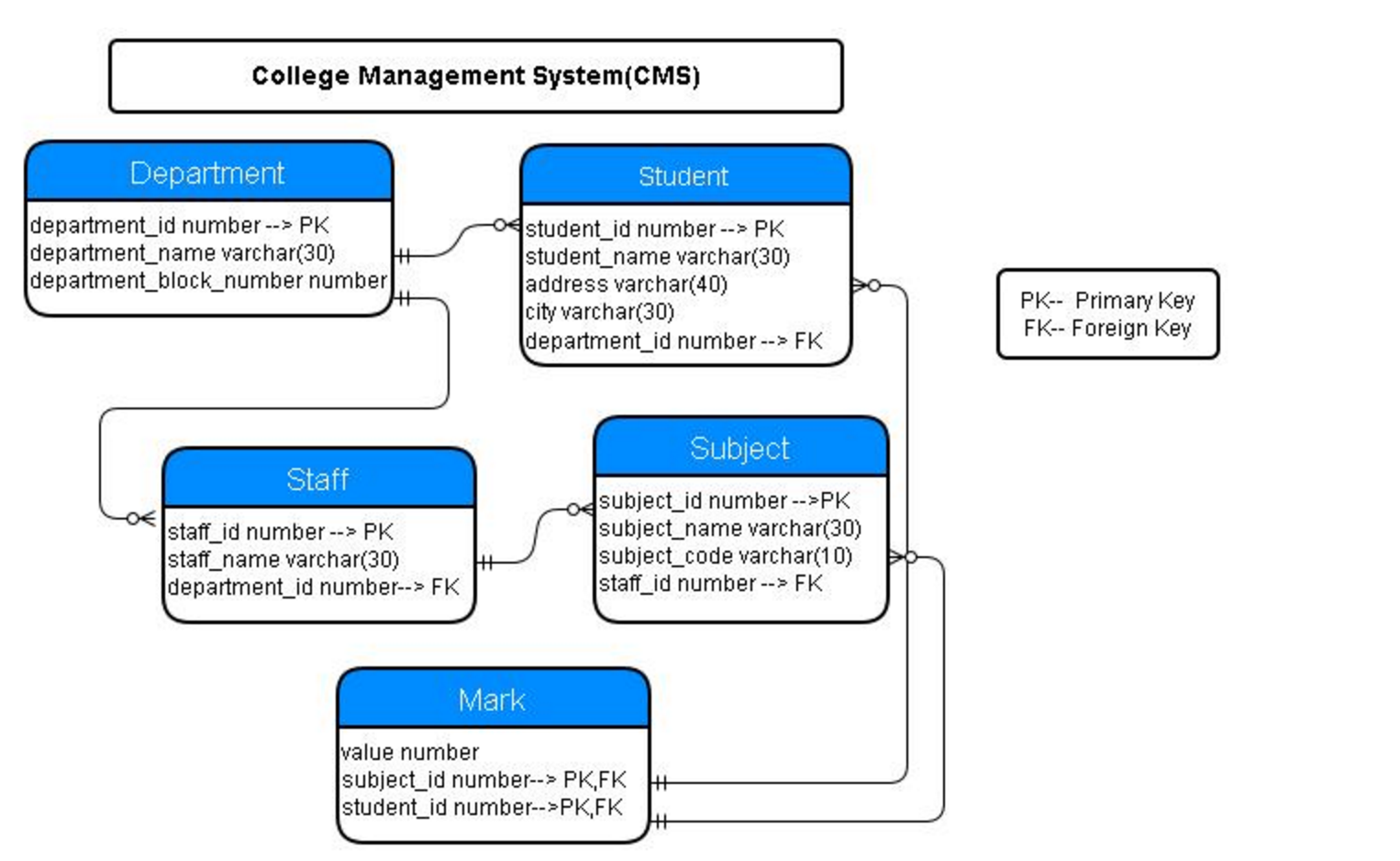
* **Make sure that you list the section number for each question and title/headings as well as question numbers when submitting your answer.**
* **Please enter all of your answers in a Microsoft Word Doc.**

**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,**

**department\_name VARCHAR(30),**

**department\_block\_number NUMBER,**

**CONSTRAINT department\_id\_pk PRIMARY KEY (department\_id)**

**);**

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

**-- added primary key to student\_id referenced in ERD**

**CREATE TABLE student(**

**student\_id NUMBER,**

**student\_name VARCHAR(30),**

**address VARCHAR (40),**

**city VARCHAR(30),**

**department\_id NUMBER,**

**CONSTRAINT student\_id\_pk PRIMARY KEY (student\_id),**

**CONSTRAINT department\_id\_fk FOREIGN KEY(department\_id) REFERENCES department(department\_id)**

**);**

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

**-- added primary key to staff\_id referenced in ERD**

**CREATE TABLE staff(**

**staff\_id NUMBER,**

**staff\_name VARCHAR(30),**

**department\_id NUMBER,**

**CONSTRAINT staff\_id\_pk PRIMARY KEY (staff\_id),**

**CONSTRAINT department\_id\_staff\_fk 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,**

**student\_name VARCHAR (30), --change this to subject\_name to match erd**

**student\_code VARCHAR(10),**

**staff\_id NUMBER,**

**CONSTRAINT subject\_id\_pk PRIMARY KEY (subject\_id),**

**CONSTRAINT staff\_id\_fk FOREIGN KEY(staff\_id) REFERENCES staff(staff\_id)**

**);**

**ALTER TABLE subject RENAME COLUMN student\_name to subject\_name;**

**-- mistake in instructions, following the ERD**

**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 value\_mark\_pk PRIMARY KEY(subject\_id, student\_id),**

**CONSTRAINT value\_mark\_sub\_fk FOREIGN KEY(subject\_id) REFERENCES subject(subject\_id),**

**CONSTRAINT value\_mark\_stud\_fk 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.**

**-- ADD COLUMN does not work for our version of SQL, returns error**

**ALTER TABLE student ADD emailid VARCHAR2(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 VARCHAR2(50);**

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

**ALTER TABLE student DROP COLUMN emailid;**

**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\_code= 1919,subject\_name ='Computer Science' WHERE subject\_name='Sales';**

**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= 'Chicago' OR city='Taylor' OR city='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 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\_block\_number) 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(student\_id) AS stud\_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(student\_id) AS student\_count FROM department INNER JOIN student ON department.department\_id = student.department\_id GROUP BY department.department\_name ORDER BY student\_count;**

**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 stud INNER JOIN mark mk ON stud.student\_id = mk.student\_id INNER JOIN subject sub ON mk.subject\_id = sub.subject\_id WHERE subject\_code > 1600;**

**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 FROM student stud INNER JOIN mark mk ON stud.student\_id = mk.student\_id INNER JOIN subject sub ON mk.subject\_id = sub.subject\_id WHERE 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) AS total\_department**

**FROM department**

**GROUP BY (department\_block\_number)**

**ORDER BY total\_department DESC**

**) WHERE ROWNUM=1;**

**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 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 (dept\_id NUMBER)**

**RETURN VARCHAR2**

**AS**

**d\_name department.department\_name%TYPE;**

**BEGIN**

**SELECT department\_name INTO d\_name**

**FROM department**

**WHERE department\_id = dept\_id;**

**RETURN d\_name;**

**EXCEPTION WHEN NO\_DATA\_FOUND THEN**

**DBMS\_OUTPUT.PUT\_LINE('No department name with this id');**

**RAISE;**

**END find\_dept\_name;**

**-- To Execute: EXECUTE DBMS\_OUTPUT.PUT\_LINE(find\_dept\_name( DEPARTMENT\_ID ));**

**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 NUMBER)**

**RETURN NUMBER**

**AS**

**d\_block department.department\_block\_number%TYPE;**

**BEGIN**

**SELECT department\_block\_number INTO d\_block**

**FROM department**

**WHERE department\_id = dept\_id;**

**RETURN d\_block;**

**EXCEPTION WHEN NO\_DATA\_FOUND THEN**

**DBMS\_OUTPUT.PUT\_LINE('No department block with this id');**

**RAISE;**

**END find\_dept\_block;**

**-- To Execute: EXECUTE DBMS\_OUTPUT.PUT\_LINE(find\_dept\_block( DEPARTMENT\_ID ));**

**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 (staf\_id NUMBER)**

**RETURN VARCHAR2**

**AS**

**s\_name staff.staff\_name%TYPE;**

**BEGIN**

**SELECT staff\_name INTO s\_name**

**FROM staff**

**WHERE staff\_id = staf\_id;**

**RETURN s\_name;**

**EXCEPTION WHEN NO\_DATA\_FOUND THEN**

**DBMS\_OUTPUT.PUT\_LINE('No staff member with this id');**

**RAISE;**

**END find\_staff\_name;**

**-- To Execute: EXECUTE DBMS\_OUTPUT.PUT\_LINE(find\_staff\_name( STAFF\_ID ));**

**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**

**IF UPDATING THEN**

**DBMS\_OUTPUT.PUT\_LINE('DEPARTMENTS table has been updated');**

**END IF;**

**END trigger\_department\_af\_update;**

**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**

**IF DELETING THEN**

**DBMS\_OUTPUT.PUT\_LINE('A row has been deleted from DEPARTMENT');**

**END IF;**

**END trigger\_department\_bf\_delete;**

**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.**

**-- Cannot store a query in an index without a function**

**CREATE INDEX stud\_names ON student ('student\_name');**

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

**CREATE VIEW staff\_names**

**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**

**stud\_info student%ROWTYPE;**

**CURSOR stud\_c IS**

**SELECT \* FROM student;**

**BEGIN**

**OPEN stud\_c;**

**LOOP**

**FETCH stud\_c INTO stud\_info;**

**EXIT WHEN stud\_c%NOTFOUND;**

**DBMS\_OUTPUT.PUT\_LINE(**

**stud\_info.student\_id || ' ' || stud\_info.student\_name || ' ' || stud\_info.address ||**

**' ' || stud\_info.city || ' ' || stud\_info.department\_id || stud\_info.emailid);**

**END LOOP;**

**CLOSE stud\_c;**

**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(stud\_id NUMBER)**

**RETURN VARCHAR2;**

**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**

**dept\_info department%ROWTYPE;**

**CURSOR dept\_c IS**

**SELECT \* FROM department;**

**BEGIN**

**OPEN dept\_c;**

**LOOP**

**FETCH dept\_c INTO dept\_info;**

**EXIT WHEN dept\_c%NOTFOUND;**

**DBMS\_OUTPUT.PUT\_LINE(**

**dept\_info.department\_id || ' ' || dept\_info.department\_name || ' ' || dept\_info.department\_block\_number);**

**END LOOP;**

**END select\_departments;**

**FUNCTION select\_student (stud\_id NUMBER)**

**RETURN VARCHAR2**

**AS**

**stud\_name student.student\_name%TYPE;**

**BEGIN**

**SELECT student\_name INTO stud\_name**

**FROM student**

**WHERE student\_id = stud\_id;**

**RETURN stud\_name;**

**EXCEPTION WHEN NO\_DATA\_FOUND THEN**

**DBMS\_OUTPUT.PUT\_LINE('No staff member with this id');**

**RAISE;**

**END select\_student;**

**END College;**

**-- To Execute: EXECUTE College.select\_departments;**

**-- To Execute: EXECUTE DBMS\_OUTPUT.PUT\_LINE(College.select\_student( student\_id ));**