**Section 1-**

**Creating Tables:**

**1.)**

CREATE TABLE department(

department\_id NUMBER GENERATED ALWAYS as IDENTITY(START with 10 INCREMENT by 10),

department\_name VARCHAR(30),

department\_block\_number INT,

CONSTRAINT department\_pk PRIMARY KEY(department\_id)

);

**2.)**

CREATE TABLE student(

student\_id NUMBER GENERATED ALWAYS as IDENTITY(START with 1 INCREMENT by 1),

student\_name VARCHAR(30),

address VARCHAR(40),

city VARCHAR(30),

department\_id NUMBER,

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

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

);

**3.)**

CREATE TABLE staff(

staff\_id NUMBER GENERATED ALWAYS as IDENTITY(START with 100 INCREMENT by 1),

staff\_name VARCHAR(30),

department\_id NUMBER,

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

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

);

**4.)**

CREATE TABLE subject(

subject\_id NUMBER GENERATED ALWAYS as IDENTITY(START with 1 INCREMENT by 1),

subject\_name VARCHAR(30),

subject\_code VARCHAR(10),

staff\_id INT,

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

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

);

**5.)**

CREATE TABLE mark(

value\_mark NUMBER,

subject\_id NUMBER,

student\_id NUMBER,

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

CONSTRAINT fk\_subject\_id FOREIGN KEY (subject\_id) REFERENCES subject(subject\_id),

CONSTRAINT fk\_student\_id FOREIGN KEY (student\_id) REFERENCES student(student\_id)

);

**6.)**

ALTER TABLE staff MODIFY staff\_name not null;

**7.)**

ALTER TABLE student ADD emailid VARCHAR(20);

**8.)**

ALTER TABLE student MODIFY emailid VARCHAR(50);

**9.)**

ALTER TABLE student DROP COLUMN emailid;

**Section 2-**

**Inserting Into Tables**

**10.)**

Note: Disabled constraints to make importing data faster and enabled it after importing data.

**Section 3-**

**Updating Records**

**12.)**

Update subject

SET subject\_name = ‘Computer Science’, subject\_code = 1919

WHERE subject\_id = 8

**Section 4-**

**13.)**

**Deleting Records**

DELETE FROM subject WHERE subject\_name = Accounting

**Section 5-**

**Basic Selection of Records**

**14.)**

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.**

**//Does not include beginning or end values**

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

//**Includes begin and end values**

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

**16.)**

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.**

**//If it was meant to find a student from all three then this one applies**

SELECT student\_name FROM student WHERE city = 'Chicago' AND city = 'Taylor' AND city = 'San Jose' ORDER BY student\_id ASC;

**//If it was meant to find students from either of these cities then it is the below answer**

SELECT student\_name FROM student WHERE city = 'Chicago' OR city = 'Taylor' OR city = 'San Jose' ORDER BY student\_id ASC;

**18.)**

SELECT address || ',' || city AS Address\_Student FROM student;

**19.)**

**//below are 6 underscores to give names are of 6 characters in length**

SELECT student\_name FROM student WHERE student\_name LIKE '\_\_\_\_\_\_';

**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.**

**//Either query would give the correct result as long as**

SELECT COUNT(student\_id) AS stud\_count FROM student;

SELECT COUNT(student\_name) 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.department\_name, COUNT(student.student\_id) AS "student\_count"

FROM department

INNER JOIN student ON (department.department\_id = student.department\_id)

GROUP BY department.department\_id, department.department\_name

ORDER BY department.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 students**

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

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

**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**

**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**

**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**

**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**

**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**

**Section 12-**

**Views and Index- research how to create one**

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

**Oracle Assignment Question 32 revised - Create an Index command that will reference all of the students names on the student table**

CREATE INDEX all\_names ON student (student\_name);

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

CREATE VIEW [All Current Staff] 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 c1 is**

**Select \* from student;**

**1\_rec**

**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.**