Simon Parr

[Company name]  [Company address]

158.337 Database development – Assignment 2

Contents

# Section A

(SQL Queries) (20 marks)

Note: For each of the questions in Section A {(a)-(j)}, write only a single query (i.e. you cannot write two select statements to arrive at an answer and use only the given criteria). We will either give full or zero marks for questions in this section.

* Write a query that will list the students who have completed 26 years. Display their ages in completed years and in the order of decreasing age.

**ANSWER:**

SELECT student.s\_id, student.s\_last, student.s\_first,

ROUND((sysdate-student.s\_dob)/365) AS "Age",

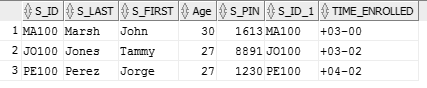
student.s\_pin, student.s\_id, student.time\_enrolled

FROM student

WHERE (sysdate-student.s\_dob)/365>=26

ORDER BY student.s\_dob asc;

**OUTPUT:**



* Write a query that will list the total building capacity for every building. This list (in the increasing order of the total capacity) should only contain the buildings with a total building capacity of 100 or over.

**ANSWER:**

SELECT location.bldg\_code, SUM(location.capacity)

AS "Building Capacity"

FROM location

GROUP BY location.bldg\_code

HAVING SUM(location.capacity)>=100

ORDER BY location.bldg\_code ASC;

**OUTPUT:**



* Write a query that will list all the students and their respective (faculty) supervisors. Arrange your list in the order of faculty supervisor's names.

**ANSWER:**

SELECT student.s\_first ||','|| student.s\_last AS "Student Name",

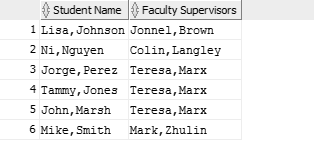
faculty.f\_first ||','|| faculty.f\_last AS "Faculty Supervisors"

FROM student, faculty

WHERE student.f\_id = faculty.f\_id

ORDER BY faculty.f\_last , faculty.f\_first ASC;

**OUTPUT:**



* Write a query that will list the faculty members who are located in the (LIB)rary building. e. Write a query that will list the students (along with their grade and course details) who got at least B or better grade (i.e. B or A) in any of their courses.

**ANSWER:**

SELECT faculty.f\_first ||','|| faculty.f\_last AS "Faculty Members",

location.bldg\_code as "Location"

FROM faculty, location

WHERE faculty.LOC\_ID=location.LOC\_ID AND location.bldg\_code like '%LIB%'

ORDER BY faculty.f\_last ASC;

**OUTPUT:**



* Write a query that will list students who enrolled in the courses offered in the terms Spring 2013 or Spring 2014. Do not display the duplicate student names.

**ANSWER:**

SELECT student.s\_first ||', '|| student.s\_last AS "Student Name",

course.course\_no AS "Course Number" ,

course.course\_name AS "Course Details",

course.credits,enrollment.grade

FROM student, enrollment, course\_section, course

WHERE student.s\_id=enrollment.s\_id

AND enrollment.c\_sec\_id=course\_section.c\_sec\_id

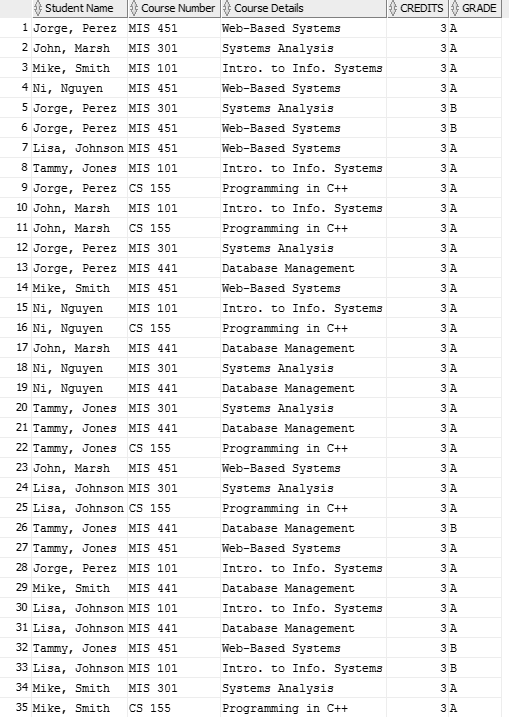
AND course\_section.course\_no=course.course\_no

AND enrollment.grade LIKE '%B%'or enrollment.grade like '%A%'

GROUP BY student.s\_first ||', '|| student.s\_last, enrollment.grade,

course.course\_no ,course.course\_name, course.credits;

**OUTPUT:**



* Write a query listing the details of the faculty member(s) who supervises the highest number of students. Also display the highest number of supervised students.

**ANSWER:**

SELECT student.s\_first ||', '|| student.s\_last AS "Student Name"

FROM student, enrollment, course\_section, term

WHERE student.s\_id = enrollment.s\_id

AND enrollment.c\_sec\_id = course\_section.c\_sec\_id

AND course\_section.term\_id = term.term\_id

AND term.term\_desc LIKE '%Spring 2013%'

OR term.term\_desc LIKE '%Spring 2014%'

GROUP BY student.s\_first ||', '|| student.s\_last;

SELECT DISTINCT student.s\_first ||', '|| student.s\_last

AS "Student Name"

FROM student, enrollment, course\_section, term

WHERE student.s\_id=enrollment.s\_id

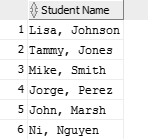
AND enrollment.c\_sec\_id = course\_section.c\_sec\_id

AND course\_section.term\_id = term.term\_id

AND term.term\_desc LIKE '%Spring 2013%'

OR term.term\_desc LIKE '%Spring 2014%';

**OUTPUT:**



* Write a query that will list students enrolled with a total of 12 or more credit points (in the decreasing order). Do not assume or hard code the value of the course credits. i. Write a query that will list student(s) enrolled with the maximum total credit points.

**ANSWER:**

SELECT faculty.f\_first ||', '|| faculty.f\_last AS "Faculty Members",

COUNT(\*)AS "Number of Students"

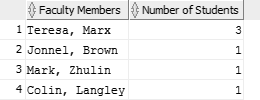
FROM faculty, student

WHERE faculty.f\_id=student.f\_id

GROUP BY faculty.f\_first ||', '|| faculty.f\_last

ORDER BY COUNT(\*) DESC;

**OUTPUT:**



* j. Write a query that lists all the courses (with their course names) and the course sections that are offered either on a (M)onday or at least four times a week. Also display the number of the days that the courses are offered (e.g. 5 days).

**Note:** In table Course\_Section, the attribute c\_sec\_day lists weekdays, where the first letter represents a weekday (e.g. M-Monday,....F-Friday; for Thursday R is used). The course section weekdays are listed in the order M(onday) to (F)riday i.e. Monday being the first.

**ANSWER:**

SELECT DISTINCT course.course\_no,course.course\_name as " Course Names",

course\_section.c\_sec\_day as "Days Taught",

LENGTH(course\_section.c\_sec\_day) as "Number of Days Offered"

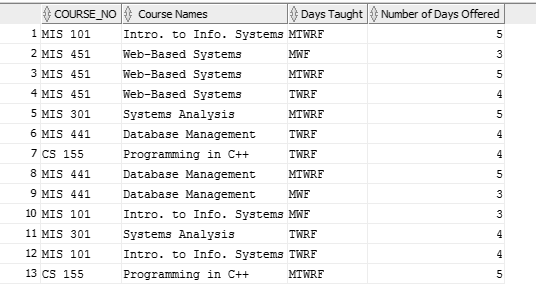
FROM course\_section, course

WHERE course.course\_no= course\_section.course\_no and course\_section.c\_sec\_day

like '%M%' or length(course\_section.c\_sec\_day)>=4

group by course.course\_no,course.course\_name, course\_section.c\_sec\_day;

**OUTPUT:**



# Section B

(PL SQL) (26 marks)

For (k) below, provide the equivalent plain English statements (that clearly explain the purpose of your triggers), PL SQL code and the results. Provide sensible and useful examples and do not use the already given or similar triggers as your examples for this question. Note: For Statement and row level Generic Trigger question

(k) – triggers shouldn’t do what could be easily done within the create table commands (e.g. if something is null or not and then display this has a null/not null value – no credit for such triggers will be given as the null/not null constraint could achieve the same). Triggers should be mainly based on the tables already provided and not unnecessarily create too many and /or similar tables. Altering table (adding a field) is fine. Adding one or two tables may be okay – but justification needed.

Comprehensive testing needed for all triggers and procedures.

* k. Write two triggers one statement level and another row level. Display the successful creation and running of the triggers. Please ensure that you also display the relevant tables before and after (results of the trigger) the triggers are fired. Remember to provide the equivalent English statements about your triggers’ purpose.

**TRIGGER ONE - ENGLISH STATEMENT:**

This trigger is fired when a new student is created. It checks, before anything is inserted, whether or not the student is of the correct age to enrol in university. It does this by obtaining the system date, working out the student's age, and acting on the result

**TRIGGER ONE:**

CREATE OR REPLACE TRIGGER row\_level

BEFORE

INSERT OR UPDATE OF s\_dob

ON student

FOR EACH ROW

DECLARE

no\_dob\_error EXCEPTION;

BEGIN

IF floor(months\_between( sysdate, :new.s\_dob ) / 12) < 16 THEN

DBMS\_OUTPUT.PUT\_LINE('Students must be older than 16 to enroll');

RAISE no\_dob\_error;

END IF;

**TRIGGER ONE - ENGLISH STATEMENT:**

This trigger is fired when a new student is being

**TRIGGER ONE:**

CREATE OR REPLACE TRIGGER row\_level

BEFORE

INSERT OR UPDATE OF s\_dob

ON student

FOR EACH ROW

DECLARE

no\_dob\_error EXCEPTION;

BEGIN

IF floor(months\_between( sysdate, :new.s\_dob ) / 12) < 16 THEN

DBMS\_OUTPUT.PUT\_LINE('Students must be older than 16 to enroll');

RAISE no\_dob\_error;

END IF;

* l. Write a trigger that does not allow more than two 'Full' ranked professors as part of the faculty (For example, trigger should fire if a new (third) Full professor is added or rank is updated to Full for one of the existing Associate professors.). Provide comprehensive test data and results to confirm that the trigger works.
* m. Write a procedure to insert a new faculty record. The procedure should also automatically calculate the faculty salary value. This calculated salary is 20% less than the average salary of the existing faculty members. Provide rest of the attribute values as input parameters. Execute your procedure to insert at least one faculty record.
* n. Write a function to format a number (9, 2) to $9,999,999.99. Use this function in a SQL statement for displaying a faculty member’s salary

Section C

(LINQ Queries)

For (o) – (r) below, write queries using LINQ. o. List all faculty who earn 90,000 or over. p. List all courses that have MIS in their course number. q. Display the total number of rooms in each building. r. Display total number of students supervised by each faculty in the order of faculty name.

* Section D (NoSQL) (8 marks)
* Use the code provided in NoSQL.txt to create a collection of nine rows called hobbits.
* Now write MapReduce code to generate a report based on the gender and total cash. (e.g. – for the female gender the total cash will be 99999 (some value))
* Include both code and the report generated in your printed work. Place the code in script file as well.