DUBLIN INSTITUTE OF TECHNOLOGY

DT228 BSc. (Honours) Degree in Computer Science
DT282 BSc. (Honours) Degree in Computer Science (International)

Year 2

WINTER EXAMINATIONS 2017/2018

DATABASES I [CMPU2007]

MS. DEIRDRE LAWLESS
DR. DEIRDRE LILLIS
MR. PATRICK CLARKE

TUESDAY 9TH JANUARY 9.30 – 11.30 A.M.

INSTRUCTIONS TO CANDIDATES

Answer **ALL** Questions from **SECTION A**
AND
Answer **ONE** Question from **SECTION B**

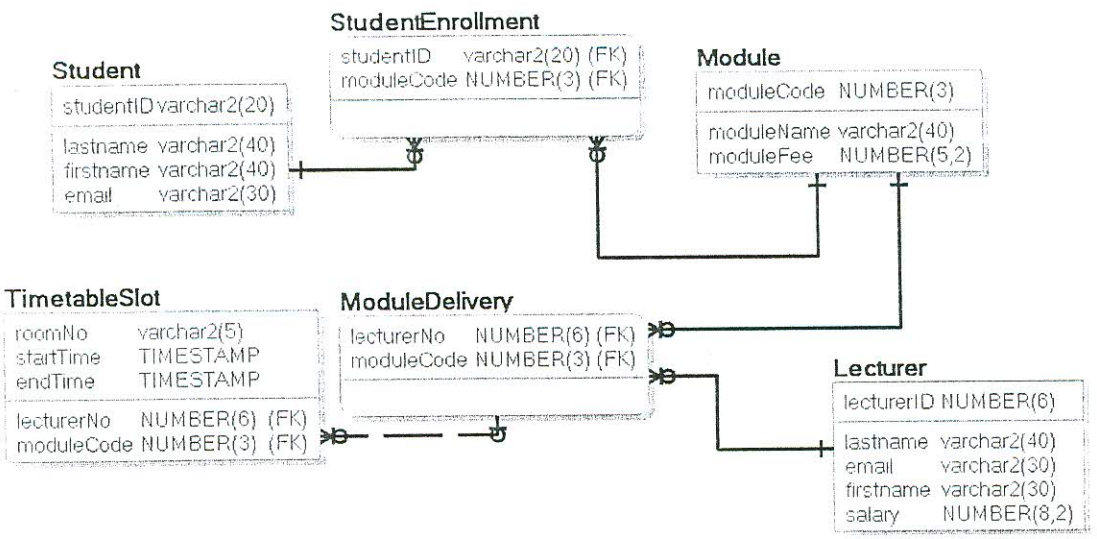
There is a SYNTAX TABLE at the end of the paper to assist you.

SECTION A

SECTION A

1. **(a)** Tech University wants to use a database to store the data needed to generate timetables for students. Details of modules available, the students enrolled in these modules, the lecturers who teach the modules and the rooms which are used for teaching these modules need to be stored.

From a brief initial analysis the following *Physical Entity Relationship Diagram* has been created:



For each of the following concepts provide a clear *explanation of the concept* and *identify an example* of it from the diagram above:

(i) Entity

(ii) Attribute

(iii) Primary Key

(iv) Foreign Key

(v) Identifying Relationship

(vi) Non-Identifying Relationship

(6 x 2 marks)

Question One continues on the next page

SECTION A

1. (b) Suppose the `Student`, `Module`, `Lecturer` and `TimetableSlot` tables have been created in an Oracle database with the attributes and datatypes identified in the model shown in part (a).

Write the SQL needed to add the following `value constraints` to the tables using ALTER statements:

- The first character of studentID must be one of the following C, R, D, A;
- Student email addresses must include both the @ symbol and the . symbol;
- Lecturer email addresses must end with @techu.com.
- The last character of roomNo must be between A and Z;
- The second character of the roomNo must be a .
- Lecturer email addresses must be unique.

(6 x 3 marks)

(c) Suppose you have applied the constraints you derived in part (b). You now attempt to execute the `insert statements` shown below in the order they are written.

Identify the errors that exist in this SQL and *explain how you would correct these errors.*

```
Insert into student (studentid, lastname, firstname, email) values
(D1, 'murphy', 'sean', 'sm@mail.com');

Insert into student (studentid, lastname, firstname, email) values
('G102345', 'murphy', 'shane', 'sm@mail.com');

Insert into module (modulename) values ('Operating Systems I');

Insert into lecturer (lecturerNo, lastname, firstname,
email,salary) values (12345, 'james', 'brady', 'jm@techu.com',
1564555.99);

Insert into lecturer (lecturerNo,lastname, firstname,
email,salary) values (23456, 'jane', 'dj@gmail.com', 45000.00);

Insert into moduledelivery(lectuerno, modulecode) values
(12345,'OS');
```

(6 x 3 marks)

(d) Suppose you have corrected the SQL statements in part (c) and successfully inserted data.

Write the SQL needed to achieve the following:

(i) *Add a column* to the `studentenrollment` table called `result` which can take numeric values up to `999`.

(3 marks)

(ii) *Using a sub-query* set the value of this `result` column to be `70` where a student's firstname is `sean`.

(5 marks)

(iii) *Retrieve* the `names of all students` and the `results they achieved in each module` including the `module code` in the output.

(4 marks)

SECTION A

2. Suppose we have cleared all data from the tables and inserted new data as follows:

Student

STUDENTID	LASTNAME	FIRSTNAME	EMAIL
D1001	Watt	James	JW@gmail.com
D1021	May	Teresa	TM@gmail.com
D1031	Herriot	James	JH@gmail.com
D1041	Cleeves	Anne	AC@gmail.com

Lecturer

LECTURERID	LASTNAME	EMAIL	FIRSTNAME	SALARY
101	Byrne	pb@techu.com	Pat	75000
102	Smith	ss@techu.com	Sam	89000
103	Dillon	ad@techu.com	Andrew	75000

Module

MODULECODE	MODULENAME	MODULEFEE
101	Programming	550
102	Databases	550
103	Legal Issues	550

ModuleDelivery

LECTURERNO	MODULECODE
101	101
101	102
102	101
102	103

(a) *Write the SQL* to retrieve for each module, the `module name`, the `fee` associated and the `name of the lecturer` delivering the module.

(6 marks)

(b) *Explain* how the SQL you wrote for part (a) retrieves the data needed from the tables involved.

(4 marks)

(c) *Amend the SQL* you wrote for part (a) to *use* `Oracle PL/SQL functions` and *format the output* as follows:

- *Output* the `module name` in `uppercase` in a column named `Module`;
- *Output* the `lecturer firstname` and `lastname combined into a single column` called `Delivered By` which is output in `uppercase`;
- *Output* the `module fee` *preceded* by the `Local Currency Symbol` and *formatted* `as 999.99` in a column named `Module Fee`.

(10 marks)

SECTION B

SECTION B

3. Suppose the data in the database is as given in Q2 (a).

(a) *Write SQL* to calculate for each lecturer the `total number` of modules delivered by that lecturer. You need to:

- *Format your output* to follow this template:

  `Lecturer <firstname, lastname> is delivering <no. of modules> modules`

  NOTE: You do not include the < > in your output. Retrieve firstname and lastname and calculate the no. of modules;

- *Output* <u>one</u> row for each lecturer;
- *Sort* the output in `ascending order` of `number of modules` delivered;
- *Explain* how the SQL works.

Hint: This SQL requires a GROUP clause.

(7 marks)

3. **(b)** *Amend the SQL* you wrote for part (a) so that it will <u>ONLY</u> include lecturers <u>who are NOT currently delivering</u> any modules in the output.

Identify what the output should be when the SQL is executed for the data provided and *explain* how the SQL works.

Hint: Use an Outer join

(7 marks)

3. **(c)** *Using* `UNION` *write the SQL* needed to *create* a `view` called `LecturerModules` which has the following columns `firstname, lastname, modulesdelivered`. The SQL should be based on the SQL you wrote for parts (a) and (b). You need to:

- Include columns `firstname` and `lastname` which are the lecturer's name;
- Include a column `modulesdelivered` which holds `the number of modules delivered` by the lecturer;
- *Include a row* for each lecturer whether they are delivering modules or not.

(6 marks)

SECTION B

4. Suppose that the data in the database is as given in Q2 and that the `studentenrollment` table has been been created and populated with a `Result` column added as follows:

STUDENTID	MODULECODE	RESULT
D1001	101	70
D1001	102	70
D1021	102	40
D1021	103	(null)
D1031	101	70
D1031	102	70
D1041	102	(null)
D1041	101	(null)

(a) Using `UNION` *write the SQL* to create the following output:

STUDENTID	FIRSTNAME	LASTNAME	MODULERESULT	MODULENAME
D1001	James	Watt	Passed	Databases
D1001	James	Watt	Passed	Programming
D1021	Teresa	May	Failed	Legal Issues
D1021	Teresa	May	Passed	Databases
D1031	James	Herriot	Passed	Databases
D1031	James	Herriot	Passed	Programming
D1041	Anne	Cleeves	Failed	Databases
D1041	Anne	Cleeves	Failed	Programming

Hint: You need to join studentenrollment to two other tables and to ensure that Passed is output for ModuleResult when the student result is >=40 and Failed output if the student result is <40 or null.

(10 marks)

4. **(b)** Using `INTERSECT`, *write the SQL* to find the *students* who have *passed* at least one module **and** *failed* at least one module. *Identify* clearly the output the SQL will produce when executed based on the data provided.

Hint: You need only output the studentid, firstname and lastname.

(4 marks)

4. **(c)** *Amend the SQL* you created for part (a) to:

- *Output* a `count of the number of modules passed and failed` in the output rather than outputting the module name;
- *Sort* the output `in order` of studentID.

Identify clearly the output the SQL will produce when executed based on the data provided.

Hint: use a GROUP BY clause

(6 marks)

SYNTAX TABLE

```
ALTER TABLE table column_clauses;
 column_clauses:
   ADD (column datatype [DEFAULT expr] [column_constraint(s)] [,...] )
   DROP COLUMN column   [CASCADE CONSTRAINTS]
   MODIFY column datatype [DEFAULT expr] [column_constraint(s)]
   RENAME COLUMN column TO new_name
ALTER TABLE table constraint_clause [,...];
constraint_clause:
   DROP PRIMARY KEY [CASCADE] [{KEEP|DROP} INDEX]
   DROP UNIQUE (column [,...]) [{KEEP|DROP} INDEX]
   DROP CONSTRAINT constraint [CASCADE]
   MODIFY CONSTRAINT constraint constrnt_state
   MODIFY PRIMARY KEY constrnt_state
   MODIFY UNIQUE (column [,...]) constrnt_state
   RENAME CONSTRAINT constraint TO new_name
COMMIT
CASE [ expression ]
   WHEN condition_1 THEN result_1
   WHEN condition_2 THEN result_2
   WHEN condition_n THEN result_n
   ELSE result
END
Conditions:=,>,<,>=,<=,<>, BETWEEN .. AND.., IN (list),IS NULL, IS NOT NULL,
LIKE
CREATE TABLE table ( column datatype [DEFAULT expr] [column_constraint(s) [,...]]
[,column datatype [,...]]
         [table_constraint [,...]])
Datatypes:    [CHAR [(n)] | VARCHAR2(n) | NUMBER [ n,p] | DATE | DATETIME]
Constraints: {[NOT NULL | UNIQUE | CHECK | PRIMARY KEY | FOREIGN KEY coltable1
        FOREIGN KEY REFERNECES table2(coltable2)]}
CREATE VIEW view_name AS
  SELECT columns
  FROM tables
  [WHERE conditions];
DELETE FROM tablename WHERE condition
DROP [TABLE tablename|DROP VIEW viewname]
INSERT INTO tablename (column-name-list) VALUES (data-value-list)
Logical operators:  AND, OR, NOT
ROLLBACK
SELECT [DISTINCT] select_list
      FROM table_list
         [WHERE conditions]
               [GROUP BY group_by_list]
                  [HAVING search_conditions]
                     [ORDER BY order_list [ASC | DESC]]
SELECT
      ... FROM table1 LEFT JOIN table2
         ON table1.field1 compopr table2.field2 | USING clause
      ... FROM table1 RIGHT JOIN table2
         ON table1.field1 compopr table2.field2 | USING clause
      ... FROM table1 INNER JOIN table2
         ON table1.field1 compopr table2.field2| USING clause
Key
   table1, table2   The tables from which records are combined.
   field1, field2   The fields to be joined.
   compopr          Any relational comparison operator: =  <  >  <=  >=  or  <>
```

SYNTAX TABLE

```
SELECT expression1, expression2, ... expression_n
FROM tables [WHERE conditions]
UNION
SELECT expression1, expression2, ... expression_n
FROM tables [WHERE conditions];

SELECT expression1, expression2, ... expression_n
FROM tables [WHERE conditions]
INTERSECT
SELECT expression1, expression2, ... expression_n
FROM tables [WHERE conditions];

SELECT expression1, expression2, ... expression_n
FROM tables [WHERE conditions]
MINUS
SELECT expression1, expression2, ... expression_n
FROM tables [WHERE conditions];

UPDATE tablename
[SET column-name= <data-value>] [WHERE condition]
```

ORACLE FUNCTIONS

Null Handling Functions: NVL, NVL2, NULLIF, COALESCE, CASE, DECODE.

Case Conversion functions - Accepts character input and returns a character value: UPPER, LOWER and INITCAP.

Character functions - Accepts character input and returns number or character value: CONCAT, LENGTH, SUBSTR, INSTR, LPAD, RPAD, TRIM and REPLACE.

Date functions - Date arithmetic operations return date or numeric values: MONTHS_BETWEEN, ADD_MONTHS, NEXT_DAY, LAST_DAY, ROUND and TRUNC.

Group Functions: SUM([ALL | DISTINCT] expression); AVG([ALL | DISTINCT] expression); COUNT([ALL | DISTINCT] expression); COUNT(*); MAX(expression);MIN(expression)

Number fucntions - accept numerical input and return number output - ROUND, TRUNC, MOD

Formatting: TO_CHAR(value [, format_mask])| TO_DATE(string1 [, format_mask]) | TO_NUMBER(string1 [, format_mask] [, nls_language])

Formats: Year, year spelled out; YYYY 4-digit year; YY 2-digit year;
MM Month (01-12; JAN = 01); MON Abbreviated name of month; MONTH Name of month, padded with blanks to length of 9 characters;
WW Week of year (1-53) where week 1 starts on the first day of the year and continues to the seventh day of the year; W Week of month (1-5) where week 1 starts on the first day of the month and ends on the seventh;
D Day of week (1-7); DAY Name of day; DD Day of month (1-31);
HH Hour of day (1-12); MI Minute (0-59); SS Second (0-59);
9 Represents a number; 0 Forces a zero to be displayed; $ Places a floating dollar sign; U Local currency sign;
. Prints a decimal point; , Prints a comma as thousands indicator