10.01.20

09.30 - 11.30am

CMPU 2007 Databases 1

Nat Stad, Irish Athl Boxing Centre

Programme Code: DT228, DT282
Module Code: CMPU 2007
CRN: 22391, 26460

TECHNOLOGICAL UNIVERSITY DUBLIN

KEVIN STREET CAMPUS

BSc. (Honours) Degree in Computer Science

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

Year 2

SEMESTER 1 EXAMINATIONS 2019/20

Databases 1

Dr. Emma Murphy
Dr. Deirdre Lillis
Mr. Patrick Clarke

Two Hours

Instructions to candidates

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

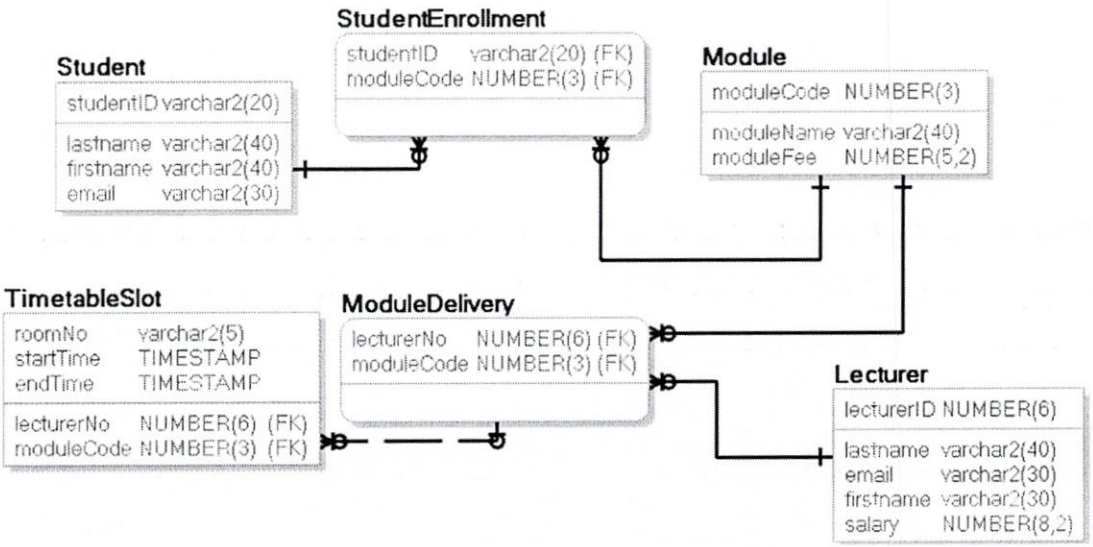
There is a SYNTAX TABLE at the end of the paper to assist you.

SECTION A

SECTION A

1. (a) A 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) *Explain* clearly what *a weak entity* is and *why they are needed* on a physical data model. Illustrate your answer by *identifying one example* of a weak entity on the physical model given in part (a).

(6 marks)

1. (c) 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 .

(5 x 3 marks)

Question One continues on the next page

SECTION A

1. (d) 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.* There is at least one error in every statement.

```
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');
```

(5 x 3 marks)

1. (e) How would you *persist* the data inserted by the statements in part (c)? Why do you need to do this?

(4 marks)

1. (f) Write the SQL needed to achieve the following:

(i) *Add a column* to the `lecturer` table called `bonus`which can take numeric values up to `99999.99`.

(3 marks)

(ii) *Using a sub-query* set the value of `bonus` to be `10% of the lowest salary of all lecturers.` Only set a bonus for lecturers with a `lowercase letter i` somewhere in their `lastname`.

(5 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

2. (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)

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

(4 marks)

2. (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 `Lecturer` and `ModuleDelivery` tables is as shown below:

Lecturer

LECTURERID	LASTNAME	EMAIL	FIRSTNAME	SALARY	BONUS
101	Byrne	pb@techu.com	Pat	75000	(null)
102	Smith	ss@techu.com	Sam	89000	7000
103	Dillon	ad@techu.com	Andrew	70000	7000

ModuleDelivery

LECTURERNO	MODULECODE
101	P101
101	D102
102	D102
102	P101
102	L103

(a) Write SQL to calculate for each `lecturer` the `total number` of modules they deliver and output this with their `salary` and `bonus`. You need to:

- *Format* your output to follow this template:

```
Lecturer <firstname> <lastname> is teaching <no. of modules>
modules and is earning a salary of <salary> and a bonus of
<bonus>.
```

NOTE: You do not include the < > in your output. Retrieve firstname, lastname, salary and bonus and calculate the no. of modules being taught.

- *Output <u>one</u> row* for each lecturer;
- *Output 0 if* a `bonus` *does not have a value* for a lecturer;
- *Explain* how the SQL works.

Hint: This SQL requires a GROUP clause; Use a function to output 0

(8 marks)

Question three continues on the next page

SECTION B

3. (b) *Amend the SQL* you wrote for part (a) so that it will only include lecturers who are NOT currently delivering any modules.

Identify what the output should be when the SQL is executed for the data provided.

Hint: Use an Outer join and think about how you restrict output from a select using GROUP BY.

(6 marks)

3. (c) Write the SQL needed to create a `view` called `LecturerPayDetails` which has the following columns `Firstname`, `LastName`, `ModulesDelivered`, `Salary` and `Bonus`. You need to:

- *Include* the `Firstname` and `lastname` of the lecturer;
- *Include* `ModulesDelivered` which is the number of modules delivered by the Lecturer;
- *Include* `Salary` which is the lecturer salary;
- *Include* `Bonus` **`which`** is the lecturer bonus or 0 if bonus has no value;
- *Include a row for each lecturer* where the salary is less than 71000 or greater than 85000.

Hint: Think UNION

(6 marks)

SECTION B

4. Suppose that the data in the `Lecturer` and `ModuleDelivery` tables is as shown below:

Lecturer

LECTURERID	LASTNAME	EMAIL	FIRSTNAME	SALARY	BONUS
101	Byrne	pb@techu.com	Pat	75000	(null)
102	Smith	ss@techu.com	Sam	89000	7000
103	Dillon	ad@techu.com	Andrew	70000	7000

ModuleDelivery

LECTURERNO	MODULECODE
101	P101
101	D102
102	D102
102	P101
102	L103

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

FIRSTNAME	LASTNAME	TEACHING
Pat	Byrne	is teaching databases
Sam	Smith	is teaching databases
Sam	Smith	is teaching legal issues

Hint: You need to join moduledelivery to one other table and to ensure that 'is teaching databases' is output for the Teaching column if the module code is 'D102' and 'is teaching legal issues' is output if module code is 'L103'. You are only interested in including lecturers teaching these modules in the output.

(10 marks)

Question four continues on the next page

SECTION B

4. (b) *Amend the SQL* you wrote for part (a) to use `INTERSECT` to find the lecturers *who teach databases* who also *teach legal issues.*

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

Hint: You need only output the firstname and lastname of the lecturer.

(4 marks)

4. (c) *Write the SQL* to output for each `lecturer` the `total number` of modules they deliver and their `salary` and `bonus`. You need to:

- *Output* the lecturer `firstname` and `lastname` separated by a space as a combined column called `LecturerName;`
- *Output one line* for each lecturer;
- *Ouptut* 0 if `bonus` *does not have a value* for a lecturer.

Hint: use a GROUP BY clause and a function.

(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.
```

SYNTAX TABLE

field1, field2 The fields to be joined.
compopr Any relational comparison operator: = < > <= >= or <>

```
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, 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