DUBLIN INSTITUTE OF TECHNOLOGY

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

Year 2

SAMPLE PAPER

DATABASES I [CMPU2007]

Ms. D. Lawless
Dr. D. Lillis
Mr. P. Clarke

TIME ALLOWED: 2 HOURS

INSTRUCTIONS TO CANDIDATES

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

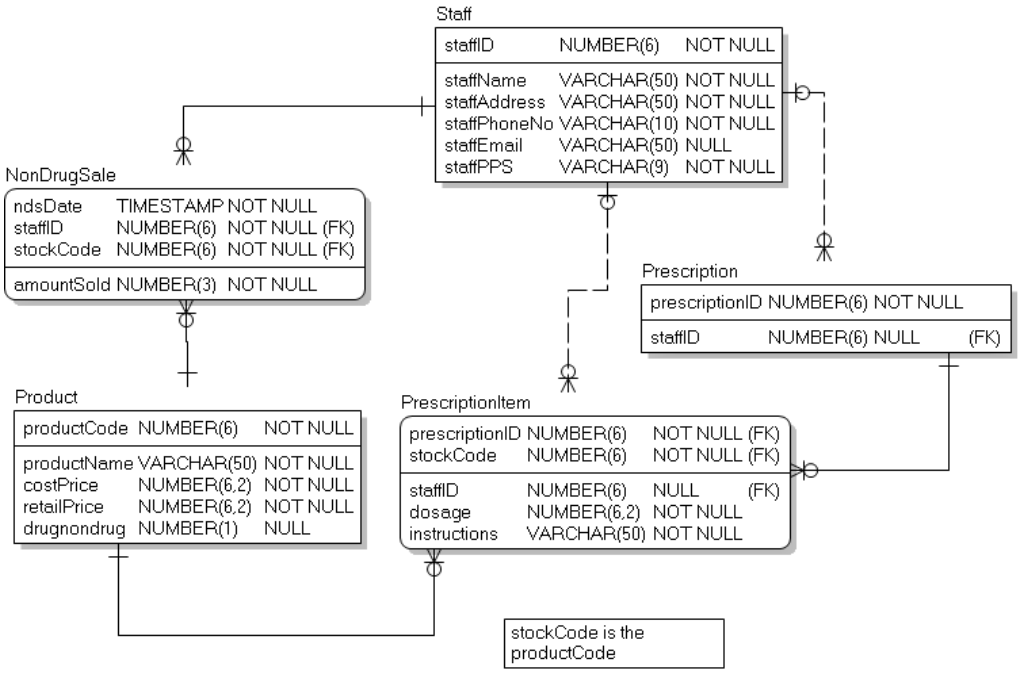
There is a SYNTAX TABLE at the end of the paper to assist you.

SECTION A

SECTION A

1. (a) George's Pharmacy wants to use a database to store the data needed to manage sales within the shop. Details of products available, the sale of non drug products and prescriptions and the staff involved in selling and prescribing 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

(4 x 2 marks)

```
This is a straightforward question. For each item 1 mark will be given for a
correct explanation, one for a correct example.
```

SECTION A

1. (b) Suppose the 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 drugnotdrug attribute can only take values 0 or 1;
- All staff email addresses must end with '@geopharm.ie';
- The last character of a staff PPPS must be between A and Z;
- The cost price of a product must be less than 2000.00.
- Staff email addresses must be unique.

(5 x 3 marks)

```
alter table product add constraint chk_drugnondrug check (drugnondrug in (0,1));
alter table staff add constraint chk_staffemail check (staffEmail like
'%@geopharm.ie');
alter table staff add constraint chk_pps check (substr(staffPPS,-1, 1) between
'A' and 'Z');
alter table product add constraint chk_cost check (costprice < 2000.00);
alter table staff add constraint uniq_stfemail unique(staffEmail);
There are some different ways of doning the checks, marks will be given for
appropriate answers. Constraints should be named and marks will be deducted if
not.
```

SECTION A

1. (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.* You can assume that your correction succeeds before the next insert statement is attempted.

```
INSERT INTO Staff ( staffID,staffName,staffAddress, staffPhoneNo,
staffEmail,staffPPS)  VALUES( 1, 'George Smith', '1, Kimmage
Road','0883333111', 'george@mail.com', '3333331A' );
INSERT INTO Staff ( staffID,staffName,staffAddress, staffPhoneNo,
staffEmail,staffPPS) VALUES( 2, 'Jane Smith', 'House 1, Kimmage
Road','0883333111', 'jane@geopharm.ie', '9234567' );
INSERT INTO Product ( productCode, productName,costPrice,
retailPrice, drugnondrug) VALUES( 1,'Drugx', 3000.99,.99,1);
INSERT INTO Product ( productCode, productName,costPrice,
retailPrice, drugnondrug) VALUES( 2,'Drugy', 0.99,.99,2);
INSERT INTO PrescriptionItem (prescriptionid, stockcode, dosage,
instructions, staffID) VALUES (1,1, 20, 'take every 2 hrs', 1);
INSERT INTO Staff ( staffID,staffName,staffAddress, staffPhoneNo,
staffEmail,staffPPS)  VALUES( 2, 'Jane Jones', '2, Crumlin
Road','08844422221', 'jane@geopharm.ie', '9234569D' );
INSERT INTO Prescriotion(prescriptionid, staffid) values (1,12);
```

(7 x 3 marks)

1. Email violates the constraint;
2. PPS number violates the constraint;
3. Costprice violates the constraint;
4. Drug non drug violates the constraint;
5. Prescription 1 does not exist;
6. Violates unique email constraint;
7. Staff id 12 does not exist.

You will lose marks if you do not state how to correct the error. Easiest way is to state a corrected insert statement in your answer.

Question One continues on the next page

SECTION A

1. (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 `product` table called `markup` which can take numeric values up to `999.99`.

(3 marks)

(ii) *Using a sub-query* set the value of this `markup` column to be `the difference between the product retailprice and the product costprice of a product called Aspirin.`

(5 marks)

(iii) *Remove the column* `retailprice` from the product table.

(3 marks)

(iv) For each `product prescribed` r*etrieve* the `prescription id, the name of each product prescribed` and `the retail price calculated as costprice*markup.`

(5 marks)

```
 (i)  Alter table product add  markup number(5,2);
(ii)  Update product set markup=(select retailprice-costprice from product where
       productname='Aspirin'); 2 marks for update; 3 for sub-query
iii)  Alter table product drop column retailprice;
(iv)    Select prescriptionid, productname, costprice*markup
      from prescriptionitem
          join product on stockcode=productcode;
       1 mark for correct  columns;  1 mark for correct driving table; 2 marks
       for correct join
```

SECTION A

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

Product

PRODUCTCODE	PRODUCTNAME	COSTPRICE	DRUGNONDRUG	MARKUP
1	Aspirin	0.99	1	20
2	Panadol Actifast	1	1	20
7	Umbrella	10	0	20
8	Insulin	5	1	20
9	Plavix	3	1	20
10	Lynx	1	0	20

Staff

STAFFID	STAFFNAME	STAFFADDRESS	STAFFPHONENO	STAFFEMAIL	STAFFPPS
1	George Jones	1 Kevin St, D2	402 2831	george@geopharm.ie	3333331AA
2	Steven Jones	1 Kevin St, D2	402 2831	steven@geopharm.ie	3333332BB
3	Ann Jones	1 Kevin St, D2	402 2831	ann@geopharm.ie	3333333CC
4	Beth Smith	4 Aungier St, D2	402 2840	beth@geopharm.ie	3333334DD
5	Jim Smith	5	402 2840	jim@geopharm.ie	3333335EE
6	Sue Smith	6	402 2840	(null)	3333336FF

PrescriptionItem

PRESCRIPTION	STOCKCODE	STAFFID	DOSAGE	INSTRUCTIONS
1	8	5	20	5ml every 2 hours
1	2	5	20	Take with food
1	1	5	20	2 every 4 hours for 24 hrs
2	2	(null)	20	2 every 4 hours. Do not exceed 12 in any 24 hour period
3	1	(null)	20	Take as required
4	1	3	20	2 every 3 hrs
5	8	(null)	50	10ml every 4 hours
4	9	3	20	When blood sugar is low
5	9	4	50	One per day

2. (a) *Write the SQL* to retrieve for each `prescription item`, the `code of the product` prescribed, the `name of the staff member who dispensed it`, the `dosage` and the `instructions`. If no staff member has been recorded you should output '`Unknown`' rather than null.

(6 marks)

```
Select stockcode, nvl(staffname,'Unknown'), dosage, instructions
From prescriptionitem
Join staff using (staffid);
2 marks driving query from correct table; 2 marks correct join;  2 marks correct use of nvl.
```

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

(4 marks)

```
This is straightforward. 4 marks for a clear, correct explanation
```

SECTION A

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

- *Output the* `stoc code` in a column named `Product;`
- *Output* the `staff name` in `uppercase` in a column named `Dispensed By`;
- *Output* the `doage` *formatted* `as 9999.00` in a column called `Prescribed Dose;`
- *Output* the `instructions` in `uppercase` in a column named `Customer Advice.`

(10 marks)

```
  Select stockcode "Product", upper( nvl(staffname,'Unknown')) "Dispensed By",
  to_char(dosage, 'fm9999.00') "Prescribed Dose",
  upper( instructions) "Customer Advice"
From prescriptionitem
Join staff using (staffid); 2 x 2 mark correct use of upper; 4 x 1 mark  for
changing column names in output; 2 marks  correct use of to_char;
```

SECTION B

SECTION B

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

3. (a) *Write SQL* to calculate for each product that has been prescribed, the `total number` of prescriptions it has appeared on. You need to:

- *Format your output* to follow this template:

  `Product <product name> has been included on <no. of prescriptions> prescriptions`

  NOTE: You do not include the < > in your output. Retrieve product name and calculate the no. of prescriptions;

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

Hint: This SQL requires a GROUP clause.

(7 marks)

```
1 mark  correct use of either concat or ||; 1 mark use of count; 2 marks correct
group by; 1 mark  correct order by ;  2 marks  explanation
select 'Product ' || productname || ' has been included on ' || count
(prescriptionid) || ' prescriptions'
from product
 join prescriptionitem on productcode=stockcode
group by productname
order by count(prescriptionid) asc;
```

'PRODUCT'||PRODUCTNAME||'HASBEENINCLUDEDON'||COUNT(PRESCRIPTIONID)||'PRESCR
Product Plavix has been included on 2 prescriptions
Product Panadol Actifast has been included on 2 prescriptions
Product Insulin has been included on 2 prescriptions
Product Aspirin has been included on 3 prescriptions

SECTION B

3. (b) *Amend the SQL* you wrote for part (a) so that it will ONLY include products that have NOT been prescribed 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)

```
select 'Product ' || productname || ' has been included on ' || count
(prescriptionid) || ' prescriptions'
from product
left outer join prescriptionitem on productcode=stockcode
group by productname
having count(prescriptionid)=0
order by count(prescriptionid) asc;
Could also use right outer join or use where as restriction
2 marks  correct use of right/left outer join ; 2 marks  correct use of
having/where; 2 marks for explanation; 1 mark  for output
```

'PRODUCT'||PRODUCTNAME||'HASBEENINCLUDEDON'||COUNT(PRESCRIPTIONID)||'PRESCR
Product Lynx has been included on 0 prescriptions
Product Umbrella has been included on 0 prescriptions

SECTION B

3. (c) *Using* `UNION` *write the SQL* to combine your answers to part (a) and part (b) to *create* a `view` called `PrescribedProducts` which has the following columns `ProductName`, `NumPrescriptions`. The SQL should be based on the SQL you wrote for parts (a) and (b). You need to:

- Include columns `ProductName` which is the product's name;
- Include a column `NumPrescriptions` which holds `the number of prescription that product appears on;`
- *Include a row* for each product whether it has been prescribed or not.

(6 marks)

```
Create View PrescribedProducts as
Select productname "ProductName", count(prescriptionid) "NumPrescriptions'"
from product
 join prescriptionitem on productcode=stockcode
group by productname
UNION
select  productname "ProductName", count(prescriptionid) "NumPrescriptions"
from product
left outer join prescriptionitem on productcode=stockcode
group by productname
having count(prescriptionid)=0;
2 marks  create view ; 2 marks for renaming columns; 2 marks  for correct use of
union
```

SECTION B

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

STAFFNAME	TRANSACTION TYPE	PRODUCTNAME
Ann Jones	prescribed	Aspirin
Ann Jones	prescribed	Plavix
Beth Smith	prescribed	Plavix
Beth Smith	sold	Lynx

The staff ids involved are 3 and 4.

Hint: You need to retrieve information from prescription item and nondrugsale using separate select statements. For prescription items, transaction type needs to be set to prescribed, for non drug sales transcation type needs to be set to sold.

(10 marks)

```
Select staffname,  'prescribed' "TRANSACTION TYPE", productname
from prescriptionitem
 join product on productcode=stockcode
 join staff using (staffid)
where staffid=3 or staffid=4
UNION
select  staffname, 'sold' "TRANSACTION TYPE", productname
from nondrugsale
 join product on productcode=stockcode
 join staff using (staffid)
where staffid=3 or staffid=4;
1 mark  correct selection of columns; 2 marks correct column moduleresult; 2 x 2
marks correct joins;    2 marks for where clauses;  1 mark  correct use of union
```

4. **(b)** Using `INTERSECT`, *write the SQL* to find the *staff* who have <u>*sold* at least one non drug product</u> **and** <u>*prescribed* at least one product on a prescription</u>. *Identify* clearly the output the SQL will produce when executed based on the data provided.

Hint: You need only output the staffname.

(4 marks)

```
1 mark correct columns; 1 mark correct use of INTERSECT; 2 marks  correct output
Select staffname
from prescriptionitem
 join product on productcode=stockcode
 join staff using (staffid)
where staffid=3 or staffid=4
intersect
select  staffname
from nondrugsale
 join product on productcode=stockcode
 join staff using (staffid)
where staffid=3 or staffid=4;
```

STAFFNAME
Beth Smith

SECTION B

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

- *Output* a `count of the number of non drug products sold and the number of products prescribed` in the output rather than outputting the product name;
- *Sort* the output `in order` of `staff name`.

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

Hint: use a GROUP BY clause

(6 marks)

```
Select staffname,  'prescribed' "Transaction Type", count(productname)
from prescriptionitem
 join product on productcode=stockcode
 join staff using (staffid)
where staffid=3 or staffid=4
 group by staffname
UNION
select  staffname, 'sold'  "Transaction Type", count(productname)
from nondrugsale
 join product on productcode=stockcode
 join staff using (staffid)
where staffid=3 or staffid=4
 group by staffname
 order by 1;-- can also order by staff name
1 mark correct use of count; 2 marks  correct group by ;  1 mark  correct order
by; 2 marks  correct output
```

STAFFNAME	Transaction Type	COUNT(PRODUCTNAM
Ann Jones	prescribed	2
Beth Smith	prescribed	1
Beth Smith	sold	1

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