CIT Semester 1 Examinations 2018/19

Note to Candidates:	Check the Programme Title and the Module Description to ensure that you have received the correct examination. If in doubt please contact an Invigilator.
Module Title:	**Data Management Systems**
Module Code:	**DATA8002**
Programme Title(s):	HDip Sc Data Science Analytics MSc Data Science & Anlytcs P1 HDipSc Data Science Anlytcs PT
Block Code(s):	**SDAAN_8_Y5** **SDAAN_91Y5** **SDAANP8_Y5**
External Examiner(s):	**Dr. James Egan**
Internal Examiner(s):	Ms. Triona Mc Sweeney, Mr. Byron Treacy
Instructions:	Answer question 1 (34 marks) and any 2 others Question 1 requires that you consult the relational database schema and sample data in Appendix A at the end of this paper. However, do not hardwire your answers to that specific data set MongoDB syntax available in Appendix B
Duration:	2 Hours
Required Items:	

Question 1: Mandatory question. **34 Marks**

Refer to the relational database schema and sample Orders data described in **Appendix A**

NB: **Your code must work for any data set**. Therefore, **do not hardwire your SQL code** to the sample data given in the appendix. The target output (answer) is given in each query, based on the sample data set. However, this is only a guide to understanding the required programming. Your SQL code must implement the queries in the general case.

Using the Orders database write SQL select statements to:

1.1 Find products made by the Hoover that cost more than 400? Ans: P3, Hoover Fridge
5 Marks

1.2 Find the list of Product numbers where the Order sale price (S_price) is less than the actual Product price (Price). Hint: find the Products that have been ordered and then filter rows based on the price difference condition. Ans P4, P5 5 Marks

1.3. List the date(s) of orders for the product named Acer 323. '2016-06-01', '2017-02-10'
6 Mark

1.4 Find the order numbers that have multiple products i.e. more than one part in an order. NB, not based on qty. The Order must have different products on it. E.g. O1, O2 have > 1 Products
6 marks

Q1.5 12 marks

Using the Product table in the Orders database devise MongoDB commands to

1.5.1 Insert a new row/collection for data P6, Apple IPad2, 999.00, Computer

1.5.2 Find all products in the computer department that cost less than 500.00

1.5.3 Find the total value (in terms of price) of the products in each department

See for reference Appendix B for sample MongoDb command syntax.

Question 2. Data Models: Relational – Object **33 marks**

2.1 Transformation from UML(object based model) to relational table designs. 13 marks

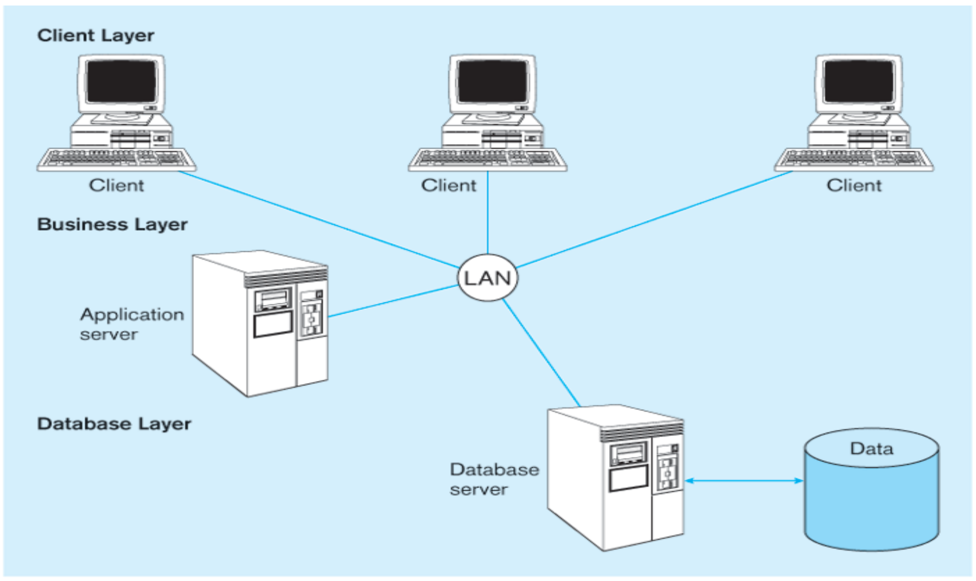
Explain, using examples, the guideline rules for how UML concepts (objects, associations etc) are transformed into relational tables.

- Objects
- 1:* binary association
- *:* binary association

2.2 Type/subtype hierarchies (generalisation/specialisation hierarchy) 20 marks

- Describe a type/subtype hierarchy using an example.
- Explain inheritance in the context of type/subtype hierarchy.
- Explain different ways to transform the UML object model into the relational model table design.

Question 3 : Data Management Systems **33 Marks**



.

Explain the figure above using the following guidelines:

3.1 Tiered architecture (Client, Application, Data)	15 marks
3.2 Advantages of a tiered (modular) architecture	8 marks
3.3 Horizontal V's Vertical Scaling	10 marks

Question 4 : Critical use of Keys in relational & non relational data systems **33 marks**

4.1 Explain the choice of key and use of surrogate keys in relational database design	20 marks
4.2 Explain the critical use of keys in Distributed Systems e.g. sharding	13 marks

Appendix A Orders database sample data

Order Design Solutions Tables & data

The following database is designed for a company that sells products to customers. Each product is attached to a department within the company e.g. there are departments for Computer, Home, Furniture etc. Each order is unique and links a customer to a list of products and quantities. An employee is responsible for each order (PrsiNo recorded in each order). The Paid field records a Yes/No value if the order has been paid for or not. Employees work in a given department.

Primary keys are in bold and underlined; foreign keys are bold and in italics.

CUSTOMERS (**CNo**, CName, Address, Sex)

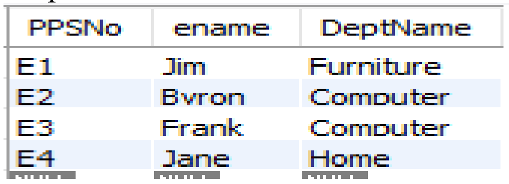
EMPS (**PPSNo**, EName, DeptName)

PRODUCTS (**PNo**, PName, Price, DeptName)

ORDERS (**OrdNo**, CNo, ODate, Paid, ***PPSNo***)

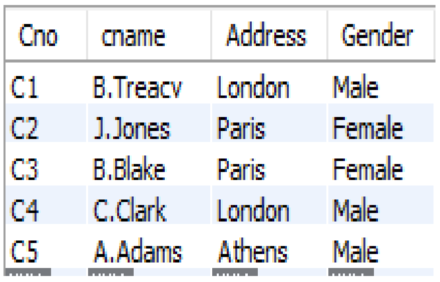
ORDER_DETAILS (**OrdNo, PNo**, qty)

Emps



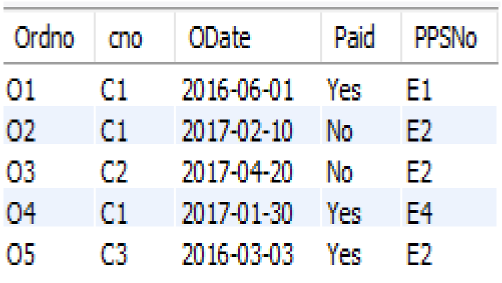
PPSNo	ename	DeptName
E1	Jim	Furniture
E2	Byron	Computer
E3	Frank	Computer
E4	Jane	Home

Customers



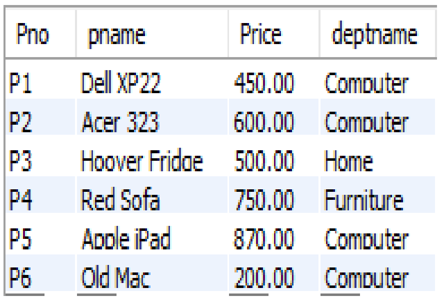
Cno	cname	Address	Gender
C1	B.Treacy	London	Male
C2	J.Jones	Paris	Female
C3	B.Blake	Paris	Female
C4	C.Clark	London	Male
C5	A.Adams	Athens	Male

Orders



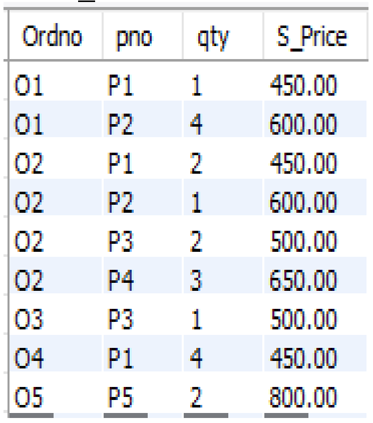
Ordno	cno	ODate	Paid	PPSNo
O1	C1	2016-06-01	Yes	E1
O2	C1	2017-02-10	No	E2
O3	C2	2017-04-20	No	E2
O4	C1	2017-01-30	Yes	E4
O5	C3	2016-03-03	Yes	E2

Products



Pno	pname	Price	deptname
P1	Dell XP22	450.00	Computer
P2	Acer 323	600.00	Computer
P3	Hoover Fridge	500.00	Home
P4	Red Sofa	750.00	Furniture
P5	Apple iPad	870.00	Computer
P6	Old Mac	200.00	Computer

Order_details



Ordno	pno	qty	S_Price
O1	P1	1	450.00
O1	P2	4	600.00
O2	P1	2	450.00
O2	P2	1	600.00
O2	P3	2	500.00
O2	P4	3	650.00
O3	P3	1	500.00
O4	P1	4	450.00
O5	P5	2	800.00

Appendix B MongoDB command syntax

Insert document(s) in a collection:

> db.*collection_name*.insert(JSON_object) → For inserting 1 document.
> db.*collection_name*.insert([JSON_object_1, …, JSON_object_n]) → For inserting n documents

Query a collection looking for documents:

> db.*collection_name*.findOne() → Looking for just one element >
db.*collection_name*.find() → Looking all elements

> db.*collection_name*.findOne({ *condition(s)* }) → Looking for element holding
> db.*collection_name*.find({ *condition(s)* }) the conditions.

> db.*collection_name*.find().limit(*number*) → Returns at most *number* documents.
> db.*collection_name*.find().skip(*number*) → Skips the first *number* documents and returns the rest of them.

Condition:

i. **Equality →** key : value
ii. **Less than →** key : {$lt : value}
iii. **Less-equal than →** key : {$lte : value}
iv. **Greater than →** key : {$gt : value}
v. **Greater-equal than →** key : {$gte : value}
vi. **Non Equality →** key : {$ne : value}

Logic AND of conditions:

{ condition1 **,** condition2 **,** … **,** conditionN }

Logic OR of conditions:

$or : [sub-condition1 , sub-condition2 , …, sub-conditionN]

Logic range: for a range within one document field:
{ field_name: { $gt: value1, $lt: value2 } }

Aggregations:

> db.*collection_name*.aggregate([{
$group : { *_id* : $*field_name*,
display_col_name : {aggregate_name : $*field_name*} #use $sum:1to count rows
}
}])

Note: use "$field_name"

Aggregate_name

i. **Sum →** $sum
ii. **Average →** $avg