Sample Solution

Here is a sample solution to the assignment. Chances are you did not get exactly the same answer – there is more than one way to achieve the correct solution. However, if you have missed major steps, concepts or – just missed the boat completely – you will probably find that you lost marks.

Hopefully – the solutions will be instructive and will help you to study for the final.

Good luck!

Question 1. Entity-Relationship Model [30 marks]

ReadBooks is a running shoes manufacturer. To support its operations, *ReadBooks* maintains a complete database of all the running shoe types that it offers for sale. The database stores sale figures for each country where the company sells shoes. Shoes are targeted for specific cultural markets so that a style of shoe sold in one market may remain approximately the same but have external stylistic additions or a different sole added to fit a different market. For example, the United States has a large basketball shoe market, but the turns and twists on the basketball court are the same as those found on a volleyball court. Taiwan has a huge volleyball shoe market. Thus, the same shoe is sold in both places but with a different set of markings and names.

Some shoes are also different gradations of the same class of shoes, i.e., the stitching may be a little less well done to save money or a cheaper form of foam cushioning or reinforcement plastic may be built into the cheaper shoe. The foreign exchange rate and inflation in some markets dictate that only lower priced shoes are sold in these countries. Markets also differ by climate. For example, hot, humid countries need mesh inserts to remove moisture from the shoes and polypropylene linings to absorb moisture from the foot.

Shoe models may create a new line of show products, or may substitute an older model that is taken off the market. For new models you need to keep information on the designers (name, address, phone number) and the date it was introduced. For models that substitute an older model you need to store the relationship between the new and the substituted model, also the date it was introduced. When a model is an adaption of another one, you need to capture the relationship between the two, the date when the adaption was introduced and the country for which the adaption is intended.

A database expert (you!) has been called in to design a new database for *ReadBook*. The database needs to accommodate for each shoe model, its name, the older model it has replaced (if any), manufactured cost, retail price and shipping costs in each country it is sold, design type, soles type, cushion support type and the major sports it is intended for. In cases where a model is adapted for a particular country, the database needs to store a model name, the name of the adapted model, the designated country and climate type for which the model is designed.

In addition, the database needs to store yearly sales by country, including number of pairs sold, price, and net profit, as well as climate type for each country.

Design and draw an ER schema that captures the information given above. Your schema should model explicitly entities and relationships in the domain of shoe sales around the world, also their attributes, generalization relationships, keys and cardinality constraints.

Here is a sample of the kinds of data your database is supposed to handle. Note that your ER schema may have very different structure from what is shown below.

Shoe Models

Model Name	Subst Model	Mfg Cost	Retail Price	Type Design	Type Soles	Cushion Support	Major Sport
Actif	Pasif	$12	$89	RX45	Neo	Neo	Bsktbl
HiJump	Hitop	$23	$95	K389	Poly	Poly	Runng
RB Xtra	Actif	$28	$113	RX45	Neo	Neo	Bsktbl
RB Tops	Pasif	$13	$93	RX45	Poly	Poly	Bsktbl
Air 1000	Actif	$24	$109	K389	Prion	Prion	Runng
Air 2000	Air 1000	$43	$189	K389	Prion	Prion	Runng

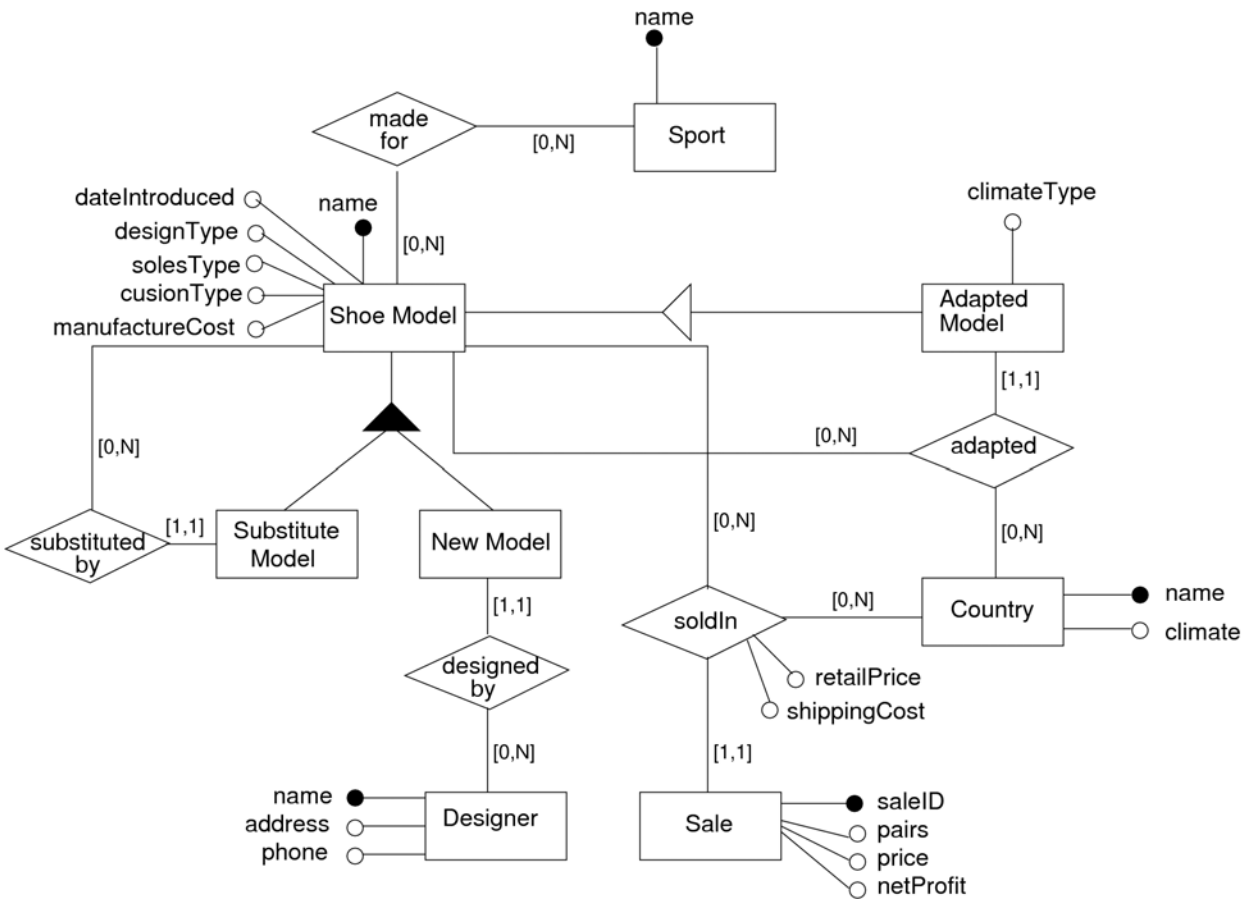
Yearly Sales by Country

Country Name	Model Name	No. Sold	Net Profit
Mali	Actif	23,000	$1.2M
Korea	Air 1000	45,000	$4.3M
Japan	Air 2000	109,000	$8.9M
Ireland	Hitop	12,000	$0.9M

Model Adaption

Model Name	Lower Priced Model Name	Designated Country	Climate Type
Air 2000	Air 500	Ethiopia	Dry
RB Tops	RB Bottoms	Ethiopia	Dry
RB Xtra	RB Less	Ethiopia	Dry
Air 1000	Air Less	Panama	Humid
Actif	R Sets	Panama	Humid
HiJump	FlatFeet	Bhutan	Mountainous
Pasif	Flyers	Bhutan	Mountainous

Sample Solution:


name
made for
[0,N]
Sport
dateIntroduced
designType
solesType
cusionType
manufactureCost
name
[0,N]
Shoe Model
climateType
Adapted Model
[1,1]
[0,N]
adapted
[0,N]
[0,N]
substituted by
[1,1]
Substitute Model
New Model
[0,N]
[0,N]
soldIn
[0,N]
Country
name
climate
retailPrice
shippingCost
[1,1]
designed by
[0,N]
name
address
phone
Designer
[1,1]
Sale
saleID
pairs
price
netProfit

Question 2. Relational Schema Design [30 marks]

Assume the following workload for the database:

- **Operation 1**: Insert a new shoe model [10 times/year]
- **Operation 2**: Insert an adapted version of a shoe model [50 times/year]
- **Operation 3**: Insert a country [2 times/year]
- **Operation 4**: Add a country sales report to the database [100 times/year]
- **Operation 5**: Report annual sales in all countries for a particular show model [20 times/year]
- **Operation 6**: Report all sales for all shoe models [10 times/year]
- **Operation 7**: Report all adaptions for a particular shoe model, and their sales in all countries [10 times/year]

You may assume that on average there are 500 shoe models, and 80 countries in the database. On average, there are 20 shoe models sold in each country. You may make additional assumptions about the workload as you need them. Please be sure to state clearly any such assumptions you make.

Design a relational database from the ER schema you created for question 1. Show explicitly the input and output of each step of your design process.

Sample Solution:

1. Analysis of redundancies.

There is a redundant attribute in the ER diagram: netProfit, which could be calculated as pairs * (price – manufactureCost). In the following, we analyze whether we need to keep this redundant attribute. The only entities and relationships involved in this analysis are: Shoe Model, Adapted Model, adapted, soldIn, Sale, and Country, and only operations 4--7 may be affected by our decision.

Assumptions: There are 100 base models and 400 adapted models, and for each base model, there is 4 adaptions on average. Each shoe model is sold in 3 countries on average.

Table of volumes:

Concept	*Type*	*Volume*
Shoe Model	E	500
Adapted Model	E	400
adapted	R	400
Country	E	80
soldIn	R	1600
Sale	E	1600

Table of operations:

Operation	*Type*	*Frequency*
Operation 1	I	10 per year
Operation 2	I	50 per year
Operation 3	I	2 per year
Operation 4	I	100 per year
Operation 5	B	20 per year
Operation 6	B	10 per year
Operation 7	B	10 per year

Tables of Accesses, *with* redundancy:

Operation 4
To calculate netProfit, reading Shoe Model's manufactureCost is necessary.

Concept	*Type*	*Accesses*	*Type*
soldIn	R	1	W
Shoe Model	E	1	R
Sale	E	1	W

cost = 100 * (2 W + 1 R) = 100 * (5 access) = 500 access per year.

Operation 5
No need to calculate netProfit for each report (already stored as an attribute).

Concept	*Type*	*Accesses*	*Type*
soldIn	R	3	R
Sale	E	3	R

cost = 20 * (6 access) = 120 access per year.

Operation 6

Concept	*Type*	*Accesses*	*Type*
soldIn	R	1600	R
Sale	E	1600	R

cost = 10 * (3200 access) = 32000 access per year.

Operation 7

Concept	*Type*	*Accesses*	*Type*
adapted	R	4	R
soldIn	R	12	R
Sale	E	12	R

cost = 10 * (28 access) = 280 access per year.

Tables of Accesses, *without* redundancy:

Operation 4
No need to calculate netProfit, so there is no need to read Shoe Model's manufactureCost.

Concept	*Type*	*Accesses*	*Type*
soldIn	R	1	W
Sale	E	1	W

cost = 100 * (2 W) = 100 * (4 access) = 400 access per year.

Operation 5
We need to calculate netProfit, so we should read Shoe Model's manufactureCost once.

Concept	*Type*	*Accesses*	*Type*
Shoe Model	E	1	R
soldIn	R	3	R
Sale	E	3	R

cost = 20 * (7 access) = 140 access per year.

Operation 6

Concept	*Type*	*Accesses*	*Type*
Shoe Model	E	500	R
soldIn	R	1600	R
Sale	E	1600	R

cost = 10 * (3700 access) = 37000 access per year.

Operation 7

Concept	*Type*	*Accesses*	*Type*
Shoe Model	E	1	R
adapted	R	4	R
soldIn	R	12	R
Sale	E	12	R

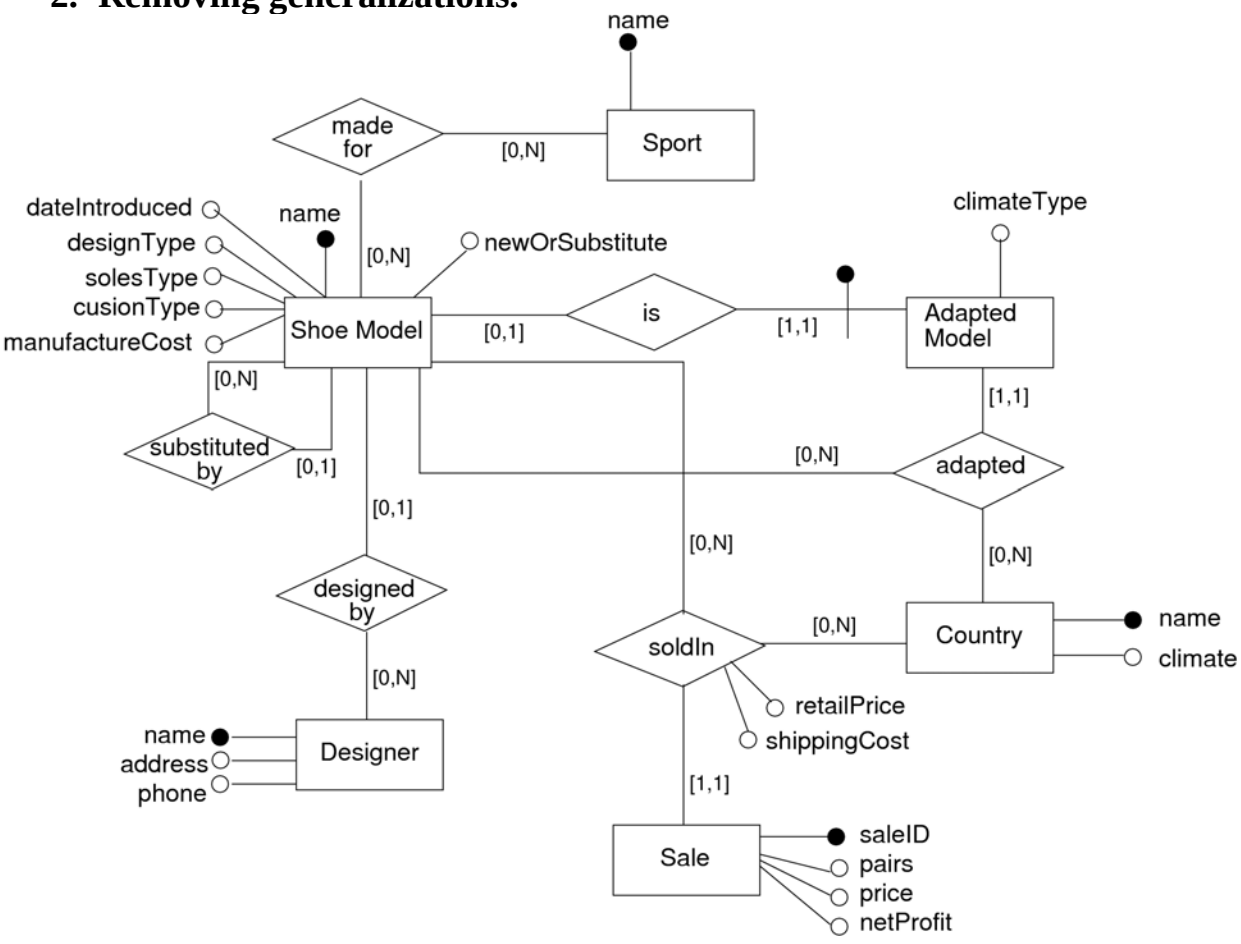
cost = 10 * (29 access) = 290 access per year.

Comparing cost of operations with and without redundancy:

Operation	*Cost with redundancy*	*Cost without redundancy*
Operation 4	500	400
Operation 5	120	140
Operation 6	32000	37000
Operation 7	280	290

We notice that operations 4,6,7, especially 6, are considerably less expensive with the presence of redundancy. Only operation 5 is more expensive with redundancy, which is negligible. Therefore, we decide to keep the redundant attribute netProfit.

2. Removing generalizations.



There is a single-attribute primary key for every entity, and we do not have any multi-valued attritute. So we continue to the next step: translating into a schema.

2. Translating into a logical schema.

The logical schema of our database consists of the following relations:

ShoeModel(modelName, dateIntroduced, designType, solesType, cusionType, manufactureCost, newOrSubstitute)
AdaptedModel(modelName, climateType, modelAdaptedFrom, countryName)
SubstitutedBy(modelName, substituteModelName)
Designer(designerName, address, phone)
DesignedBy(modelName,designerName)
Sport(sportName)
MadeFor(modelName,sportName)
Country(countryName, climate)
Sale(saleID, modelName, countryName, pairs, price, netProfit, retailPrice, shippingCost)

Question 3. Functional Dependencies [40 marks]

Consider the following functional dependencies over the attribute set `ABCDEFGH`:

A → E, BE → D, AD → BE, BDH → E, AC → E,

F→ A, E → B, D → H, BG → F, CD → A

(a) [10 marks] Find a minimal cover for this set of functional dependencies.

(b) [15 marks] Decompose the relation `ABCDEFGH` into a lossless 3NF schema.

(c) **[15 marks]** Check whether your answer to (b) is in BCNF. If not, decompose it into a lossless BCNF schema.

Solutions:

(a) First, we change every functional dependency (FD) into the form X→A (with only one attribute on the right-hand side):

A→E
BE→D
AD→B
AD→E
BDH→E
AC→ E
F→A
E→B
D→H
BG→F
CD→A

Then we check if any of the attributes on the left-hand side of the FDs are redundant:

- Attribute B is redundant in BE→D since we have E→B.
- Attribute D is redundant in AD→B since we have A→E and E→B.
- Attribute D is redundant in AD→E since we have A→E, and therefore the functional dependency AD→E is redundant.
- Attribute H is redundant in BDH→E since we have D→H.
- Attribute C is redundant in AC→ E since we have A→E, and therefore the functional dependency AC→ E is redundant.

Now we check the new set of FDs to see if any of them is redundant (i.e. they can be inferred from the others). An FD X→A is redundant if the closure of X contains A after removing the FD X→A. Let *K* denote the new set of FDs:

A→E
E→D
A→B
BD→E
F→A
E→B
D→H
BG→F
CD→A

- $(A)^+_{K-\{A\rightarrow E\}}$={A,B} so the FD A→E is NOT redundant.
- $(E)^+_{K-\{E\rightarrow D\}}$={E,B} so the FD E→D is NOT redundant.
- $(A)^+_{K-\{A\rightarrow B\}}$={A,E,B} so the FD A→B is redundant and will be removed from *K*.
- $(BD)^+_{K-\{BD\rightarrow E\}}$={B,D,H} so the FD BD→E is NOT redundant.
- $(F)^+_{K-\{F\rightarrow A\}}$={F} so the FD F→A is NOT redundant.
- $(E)^+_{K-\{E\rightarrow B\}}$={E,D,H} so the FD E→B is NOT redundant.
- $(D)^+_{K-\{D\rightarrow H\}}$={D} so the FD D→H is NOT redundant.
- $(BG)^+_{K-\{BG\rightarrow F\}}$={B,G} so the FD BG→F is NOT redundant.
- $(CD)^+_{K-\{CD\rightarrow A\}}$={C,D,H} so the FD CD→A is NOT redundant.

So the minimal cover is as follows:
A→E
E→D
BD→E
F→A
E→B
D→H
BG→F
CD→A

(b) A possible 3NF decomposition:

R_1(A,E) A→E

R_2(B,D,E) E→D, BD→E, E→B
R_3(A,F) F→A
R_4(D,H) D→H
R_5(B,F,G) BG→F
R_6(A,C,D) CD→A
R_7(B,C,G) no functional dependency.

(c) The solution to part (b) is also in BCNF.

Question 4. XML DTDs and Queries [50 marks]

Consider the following student database. We are interested in placing this information in document format.

Student Information:

STID	Name	Phone	St. No	Street name	City	Prov.	Courses taken	Student Status	Degree
995435245	Ali	416-4245979	406	Main	Toronto	ON	1 3 4 5	Degree Student	BSc
995267842	Bob	613-5345660	12	Charles	Ottawa	ON	3 4 5	Special Student	
997458623	Carlos	905-2348638	5	King	Oshawa	ON		Special Student	
998112455	Fernando		101	Avenue	Montreal	QC	2 4 8 10	Degree Student	BEng
993457622	Jason	204-4562983	32	Main	Winnipeg	MB	1 2 3	Degree Student	BA
996112321	Jun	204-7893242	160	Pembina	Winnipeg	MB	5 7 10	Degree Student	BEng
995987345	Lee	647-9982342	35	Charles	Toronto	ON	3 4 7 9	Degree Student	BSc
997821345	Lueng		11	Yong	Toronto	ON		Degree Student	BSc
996453222	Mark	613-4561190	30	University	Ottawa	ON	1 5 6 7	Special Student	
997424563	Maria		11	Queen	Ottawa	ON	5	Special Student	
997345632	Nicolas	613-8932456	13	Cumberland	Ottawa	ON	9 10	Special Student	

Courses information:

ID	Course Name	Department	Allowable Student Status
1	Introduction to Databases	CS	Degree Student, Special Student
2	Numerical Methods	CS	Degree Student, Special Student
3	Operating Systems	CS	Degree Student
4	Computer Graphics	CS	Degree Student
5	Calculus Sci I	MAT	Special Student
6	Complex Variables	MAT	Degree Student, Special Student
7	Groups and Symmetry	MAT	Degree Student
8	Introduction to Economics	Economics	Degree Student, Special Student
9	Microeconomic Theory	Economics	Degree Student
10	Energy & Resources	Economics	Special Student

(a) [10 marks] Define a DTD, *students.dtd*, for documents that list all courses offered, and then the courses taken by each student. The DTD should allow each registered student to take 0 or more courses. Phone numbers are optional for. Also, only degree students have a degree attribute. A course may be taken by students whose status is degree, special or both. A student may be taking a course for which she can't get credit (e.g., a special student taking a course for which only degree students get credit).

```
<?xml version="1.0" encoding="UTF-16LE"?>
<!ELEMENT Report (Students,Courses)>
<!ELEMENT Students (DegreeStudents,SpecialStudents)>
<!ELEMENT Courses (Course+)>
<!ELEMENT DegreeStudents (Student+)>
<!ELEMENT SpecialStudents (Student+)>
<!ELEMENT Student (Address,(Phone?),(CourseTaken*))>
<!ELEMENT Address EMPTY>
<!ELEMENT CourseTaken EMPTY>
<!ELEMENT Phone (#PCDATA)>
<!ELEMENT Course (Acceptance+)>
<!ELEMENT Acceptance (#PCDATA)>
<!ATTLIST CourseTaken CourseID IDREF #REQUIRED>
<!ATTLIST Address StNo CDATA #REQUIRED>
<!ATTLIST Address StName CDATA #REQUIRED>
<!ATTLIST Address City CDATA #REQUIRED>
<!ATTLIST Address Prov CDATA #REQUIRED>
<!ATTLIST Student STID CDATA #REQUIRED>
<!ATTLIST Student Name CDATA #REQUIRED>
<!ATTLIST Student Degree CDATA #IMPLIED>
<!ATTLIST Student Status CDATA #REQUIRED>
<!ATTLIST Course CourseID ID #REQUIRED>
<!ATTLIST Course Name CDATA #REQUIRED>
<!ATTLIST Course Department CDATA #REQUIRED>
```

(b) [10 marks] Provide an XML document, *students.xml*, that captures the information given in the tables above and is consistent with *students.dtd*.

```
<?xml version="1.0"?><!DOCTYPE Report SYSTEM "students.dtd">
<Report>
<Students>
  <DegreeStudents>
    <Student STID="995435245" Name="Ali" Degree="BSc" Status="Degree Student">
      <Address StNo="406" StName="Main" City="Toronto" Prov="ON"/>
      <Phone>416-4245979</Phone>
      <CourseTaken CourseID="c1" />
      <CourseTaken CourseID="c3" />
      <CourseTaken CourseID="c4" />
      <CourseTaken CourseID="c5" />
    </Student>
   <Student STID="998112455" Name="Fernando" Degree="BEng" Status="Degree Student">
      <Address StNo="101" StName="Avenue" City="Montreal" Prov="QC"/>
      <CourseTaken CourseID="c2" />
      <CourseTaken CourseID="c4" />
      <CourseTaken CourseID="c8" />
      <CourseTaken CourseID="c10" />
    </Student>
    <Student STID="993457622" Name="Jason" Degree="BA" Status="Degree Student">
      <Address StNo="32" StName="Main" City="Winnipeg" Prov="MB"/>
      <Phone>204-4562983</Phone>
      <CourseTaken CourseID="c1" />
```

```
      <CourseTaken CourseID="c2" />
      <CourseTaken CourseID="c3" />
    </Student>
    <Student STID="996112321" Name="Jun" Degree="BEng" Status="Degree Student">
      <Address StNo="160" StName="Pembina" City="Winnipeg" Prov="MB"/>
      <Phone>204-7893242</Phone>
      <CourseTaken CourseID="c5" />
      <CourseTaken CourseID="c7" />
      <CourseTaken CourseID="c10" />
    </Student>
    <Student STID="995987345" Name="Lee" Degree="BSc" Status="Degree Student">
      <Address StNo="35" StName="Charlse" City="Toronto" Prov="ON"/>
      <Phone>647-9982342</Phone>
      <CourseTaken CourseID="c3" />
      <CourseTaken CourseID="c4" />
      <CourseTaken CourseID="c7" />
    </Student>
    <Student STID="997821345" Name="Leung" Degree="BSc" Status="Degree Student">
      <Address StNo="11" StName="Yong" City="Toronto" Prov="ON"/>
    </Student>
  </DegreeStudents>

  <SpecialStudents>
    <Student STID="995267842" Name="Bob" Status="Special Student">
      <Address StNo="12" StName="Charles" City="Ottawa" Prov="ON"/>
      <Phone>613-5345660</Phone>
      <CourseTaken CourseID="c3" />
      <CourseTaken CourseID="c4" />
      <CourseTaken CourseID="c5" />
    </Student>
    <Student STID="997458623" Name="Carlos" Status="Special Student">
      <Address StNo="5" StName="King" City="Oshawa" Prov="ON"/>
      <Phone>905-2348638</Phone>
    </Student>
    <Student STID="996453222" Name="Mark" Status="Special Student">
      <Address StNo="30" StName="University" City="Ottawa" Prov="ON"/>
      <Phone>613-4561190</Phone>
      <CourseTaken CourseID="c1" />
      <CourseTaken CourseID="c5" />
      <CourseTaken CourseID="c6" />
      <CourseTaken CourseID="c7" />
    </Student>
    <Student STID="997424563" Name="Maria" Status="Special Student">
      <Address StNo="11" StName="Queen" City="Ottawa" Prov="ON"/>
      <CourseTaken CourseID="c5" />
    </Student>
    <Student STID="997345632" Name="Nicolas" Status="Special Student">
      <Address StNo="13" StName="Cumberland" City="Ottawa" Prov="ON"/>
      <Phone>613-8932456</Phone>
      <CourseTaken CourseID="c9" />
      <CourseTaken CourseID="c10" />
    </Student>
  </SpecialStudents>
</Students>
<Courses>
  <Course CourseID="c1" Name="Introduction to Databases" Department="CS">
    <Acceptance>Degree Student</Acceptance>
    <Acceptance>Special Student</Acceptance>
```

```
  </Course>
  <Course CourseID="c2" Name="Numerical Methods" Department="CS">
    <Acceptance>Degree Student</Acceptance>
    <Acceptance>Special Student</Acceptance>
  </Course>
  <Course CourseID="c3" Name="Operating Systems" Department="CS">
    <Acceptance>Degree Student</Acceptance>
  </Course>
  <Course CourseID="c4" Name="Computer Graphics" Department="CS">
    <Acceptance>Degree Student</Acceptance>
  </Course>
  <Course CourseID="c5" Name="Calculus Sci I" Department="MAT">
    <Acceptance>Special Student</Acceptance>
  </Course>
  <Course CourseID="c6" Name="Complex Variables" Department="MAT">
    <Acceptance>Degree Student</Acceptance>
    <Acceptance>Special Student</Acceptance>
  </Course>
  <Course CourseID="c7" Name="Groups and Symmetry" Department="MAT">
    <Acceptance>Degree Student</Acceptance>
  </Course>
  <Course CourseID="c8" Name="Introduction to Economics" Department="ECO">
    <Acceptance>Degree Student</Acceptance>
    <Acceptance>Special Student</Acceptance>
  </Course>
  <Course CourseID="c9" Name="Microeconomic Theory" Department="ECO">
    <Acceptance>Degree Student</Acceptance>
  </Course>
  <Course CourseID="c10" Name="Energy and Resource" Department="ECO">
    <Acceptance>Degree Student</Acceptance>
  </Course>
</Courses>
</Report>
```

(c) [30 marks] Represent the following queries in XQuery:

I. Display student name and status for all students. Sort the result by student name in ascending order.

```
<result>
{
for $c in doc("students.xml")//Student
order by $c/@Name ascending
return <Student> {$c/@Name, $c/@Status} </Student>
}
</result>
```

output:

```
<result>
<Student Name="Ali" Status="Degree Student"/>
<Student Name="Bob" Status="Special Student"/>
<Student Name="Carlos" Status="Special Student"/>
<Student Name="Fernando" Status="Degree Student"/>
<Student Name="Jason" Status="Degree Student"/>
<Student Name="Jun" Status="Degree Student"/>
<Student Name="Lee" Status="Degree Student"/>
<Student Name="Leung" Status="Degree Student"/>
<Student Name="Maria" Status="Special Student"/>
```

```
<Student Name="Mark" Status="Special Student"/>
<Student Name="Nicolas" Status="Special Student"/>
</result>
```

II. Display the number of students for each city mentioned in the database; sort the result by number of students in a descending order. You need to output city names and the number of students for each city.

```
<result>
{
for $c in distinct-values(
        doc("students.xml")//Address/@City
      )
let $num :=count( for $s in doc("students.xml")//Student
             where $s/Address/@City=$c
             return $s
           )
order by $num descending
return <city> {$c}, {$num}</city>
 }
 </result>
```

Output:

```
<result>
<city>Ottawa, 4</city>
<city>Toronto, 3</city>
<city>Winnipeg, 2</city>
<city>Montreal, 1</city>
<city>Oshawa, 1</city>
</result>
```

III. Display the number of students who can get a credit for each course. This includes all students whose status is allowed by any given course. Display course ID, course name and the number of students who are allowed for each course.

```
<result>{
for $c in doc("students.xml")//Course
let $students :=
     count (
       for $s in doc("students.xml")//Student
       where
         some $ids in $s/CourseTaken/@CourseID ,
             $status in $c/Acceptance
         satisfies $c/@CourseID = $ids and $s/@Status = $status
       return $s
     )
order by $students descending
return
  <course>
   {$c/@ID, $c/@Name}
    <students> { $students }</students>
  </course>
}
</result>
```

Output:

```
<result>
<course ID="1" Name="Introduction to Databases"><students>3</students></course>
<course ID="3" Name="Operating Systems"><students>3</students></course>
<course ID="4" Name="Computer Graphics"><students>3</students></course>
<course ID="5" Name="Calculus Sci I"><students>3</students></course>
<course ID="2" Name="Numerical Methods"><students>2</students></course>
<course ID="7" Name="Groups and Symmetry"><students>2</students></course>
<course ID="10" Name="Energy and Resource"><students>2</students></course>
<course ID="6" Name="Complex Variables"><students>1</students></course>
<course ID="8" Name="Introduction to Economics"><students>1</students></course>
<course ID="9" Name="Microeconomic Theory"><students>0</students></course>
</result>
```

IV. Display the name of students who have no credits for CS department courses. Include in your answer students who sit in a CS course without having allowable student status. Sort the result by student name in an ascending order.

```
<return>
{
for $s in doc("students.xml")//Student
where
  not( for $ids in $s/CourseTaken/@CourseID,
            $c in doc("students.xml")//Course[@Department="CS"]
       where $ids=$c/@CourseID
          and (
               some $status in $c[@CourseID=$ids]/Acceptance
               satisfies $s/@Status = $status
              )
       return $ids
     )
order by $s/@Name ascending
return <Student> {$s/@Name} </Student>
}
</return>
```

Output:

```
<return>
<Student Name="Bob"/>
<Student Name="Carlos"/>
<Student Name="Jun"/>
<Student Name="Leung"/>
<Student Name="Maria"/>
<Student Name="Nicolas"/>
</return>
```

V. Display the names of students who take at least two courses they are allowed to get credit for from the same department. Sort the result by student name in descending order.

```
<result>
{
let $departments := distinct-values( doc("students.xml")//Course/@Department)

for $s in doc("students.xml")//Student
where
 not(
    empty(
    for $d in $departments
    where count(
              for $ids in $s/CourseID,
                  $c in doc("courses.xml")//Course[@Department=$d]
              where $ids=$c/@ID
                and (
                    some $status in $c[@ID=$ids]/Acceptance
                    satisfies $s/@Status = $status
                    )
               return $ids
             )>1
    return $d
    )
   )
order by $s/@Name descending
return <student> {$s/@Name} </student>
}
</result>
```

Output:

```
<result>
<student Name="Mark"/>
<student Name="Lee"/>
<student Name="Jason"/>
<student Name="Fernando"/>
<student Name="Ali"/>
</result>
```