School of Computer Science, McGill University

COMP-421B Database Systems, Winter 2016

Written Assignment 2: SQL / XML

Due date: March 7; 10am (no late submission!!)

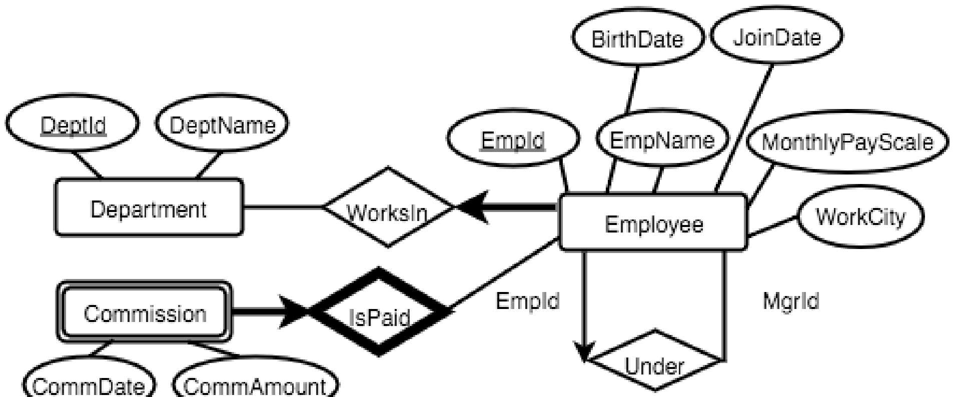
Exercise 1: SQL (70 Pts.)

The following schema stores information about a Company's employees. There can be two types of employees, *Mobile* employees who travel from location to location to client sites, and therefore has no `WorkCity` associated with them and those who are *Stationary* employees, who has a value for the `WorkCity` attribute (which implies that the company has an office in that city and the employee works there.) This is very important, because for mobile employees this attribute would be `NULL` !. You will need to use this information in many of your queries to distinguish between Mobile and Stationary employees.

All employees belong to some `Department`. `DeptId` is the primary key for `Department`, but for all practical purposes you can assume that `DeptName` is unique and no two departments can have the same name. Most employees also have a manager associated with them. But it is possible that some employees are at senior positions (like CEO) and will not have an associated manager. The employees also join the workforce on the first day of a month (yes we will not bother if it is a Sunday or any holiday).

Some employees also get paid commission (in addition to their monthly pay), the amount of which is recorded in Commission table along with the date. For a given employee, a commission payout is made once in a day, but can receive multiple commission payouts through the month, where each payout falls into a different date.

As mentioned, all employees are also entitled to a fixed monthly pay, as captured by MonthlyPayScale, and every month's paycheck is issued out on some date during the last week of that calendar month. In this simplified model, we will assume that the company does not issue any pay raises and pays the same amount to an employee all throughout their employment. Also since employees join on the first day of the month, they are entitled to full paycheck that month. There is no concept of *leave without pay*. To keep things further simple, we will assume that employees do not leave the company (kind of like the Eagle's "Hotel California" situation).



Following is the relational definition (The SQL create table statements are provided separately in the compressed file along with constraints).

```
Department(DeptId, DeptName)
Employee(EmpId, EmpName, MgrId, DeptId, MonthlyPayScale, WorkCity, BirthDate, JoinDate)
```

```
      DeptId references Department(DeptId), MgrId references Employee(EmpId)
Commission(EmpId, CommDate, Amount)
      EmpID references Employee(EmpId)
```

Important !!

All the sql solutions will be evaluated by an automated system, which compares the output data produced by executing your query on our dataset with the expected output result for the correct query. So it is important that you include the correct column names, in the correct order, perform any ordering on output tuples as asked etc. **For more details read the attached sql formatting guide**. If you have questions about this post it in the discussion forum. **Remember you will either get 0 or all points for a given SQL question !!**.

Unless specified, your output query should not produce duplicate results in your output resultset. Unless explicitly mentioned we require real world attributes in the output and not artificial keys. I.e "List the employees who were paid in August" is asking for `EmpName` and not `EmpId`. Unless otherwise specified order by implies an ascending order.

For this assignment you will not create views or intermediate tables in your solution. All your answers should be comprised of only a select query. **Output ONLY the attributes in the question, following the exact order mentioned in the question.** Adding attributes not mentioned can result in a 0 score !

1. (3 Points) List the id of all employees who has ever received a commission pay more than $2000 in a single payment. Output only the id of the employee in the increasing order of the employee ids. Ensure you do not have duplicates.
2. (4 Points) List the information of the employees working for the manager 'Thomas McCarthy'. Output only `EmpId` followed by `EmpName`. Sort the output in the alphabetical order of their names and then by increasing order of employee ids.
3. (4 Points) List the names of all employees who is not under a manager working in 'Montreal'. Output only `EmpId` followed by `EmpName`. Sort the output in the alphabetical order of their names and then by increasing order of employee ids.
4. (4 Points) List the names of departments which has no Stationary Employees. Output only the name of the departement order in the alphabetical order of their names.
5. (5 Points) List the name(s) of the stationary employees making the highest monthly salary (commission not counted). Output only the name of the employee followed by their monthly salary, order the output in the alphabetical order of the employee names.
6. (5 Points) Generate the average monthly pay for each department (not including commission). Output the name of department followed by the average monthly pay (name this average column as `AvgPay`, do not round the output to integer). Sort the output in the descending order of the average pay and then on alphabetical order of the department name.
7. (5 Points) List the names of all Managers in the 'Finance' department, who has employees under them that works in 'Vancouver' and has ever received a commission. Output the employee id of the manager(s) followed by the managers name. Do not include a manager's information more than once. sort the output in the alphabetical order of the Manager's name.
8. (5 Points) For each department, display the number of stationary employees they have in each city. Output only department name, work city and the number of employees (name the column as `EmpCnt`). Order the output in the decreasing order of the number of employees, alphabetical order of the department name and then city name.

9. (5 Points) Which department(s) has more than 10 employees ? Output only Department name. order the output in the alphabetical order of the department names.

10. (6 Points) Generate the average monthly pay (not including commission) for each department's mobile and stationary employees separately. Output the name of the department, employee type (name the column as `EmpType`), use the value 'Mobile' to indicate a mobile employee and 'Stationary' to indicate a stationary employee, followed by the average monthly pay (name this column `AvgMnthPay`) do not round this output to integer value. Order the output by the name of the department, the type of the employee. Below is an example of the output of such a query. You do not need to worry if your `AvgMnthPay` generates more decimals than is depicted in the table. Do not perform any explicit numeric conversion in your query.

DeptName	EmpType	AvgMnthPay
Finance	Mobile	3450.12
Finance	Stationary	4683.35
HR	Mobile	3156.40
HR	Stationary	4072.50

11. (6 Points) Find the manager and employee information of all employees who joined in the last one year and is more than 40 years old and works in a department with less than 5 employees. Output name of the manager (name the attribute as `MgrName`), followed by the employee's id, and name. order the output in the alphabetical order of the name of the manager and then on the id of the employee. (Hint:- explore `CURRENT_DATE` system attribute, and `INTERVAL` time data type for `YEAR`. See if you can do arithmetic involing them.).

12. (6 Points) List the names of managers who has at least 3 employees in Montreal, at least 5 employees in Toronto and at least 2 Mobile employees. Output the name of the manager (`EmpName`) and order the output in alphabetical order. (Hint:- explore a solution using set operators studied in class)

13. (6 Points) List the information of all employees who got a commission on the month of august 2015 along with the total commission they earned on that month. Output employee's id, name, and total commission amount (name this column `TotComm`). Order the output by the decreasing order of total commission received by the employee and then by increasing order of employee's id. (Hint:- use SQL functions that can extract year / month etc from a date attribute).

14. (6 Points) List employees who were in the company in august 2015 but did not receive a commission. You ARE REQUIRED to use some form of an outer join to solve this. Output the employee's id, followed by employee's name and sort the output in the increasing order of employee's id. You are not allowed to use SQL functions that extracts year / month from date in this query. Instead explore using BETWEEN operator on the date column.

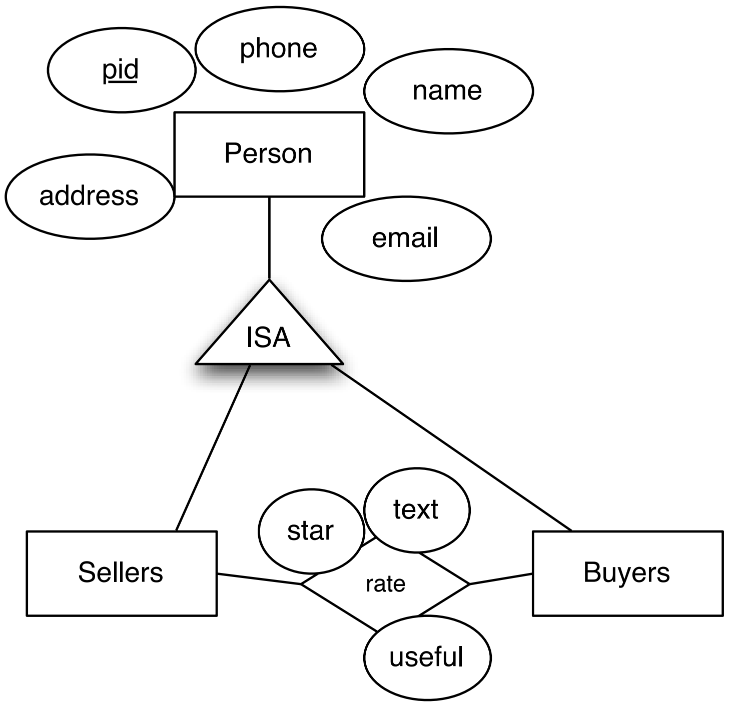
Submit all your SQL queries in separate files, following the instructions in the accompanying formating guide.

Exercise 1: XML (30 Points)

1. (14 Points) Given below an E/R diagram with reflects a small simplified portion of the E/R diagram of an online buyer/seller system. Now assume that the online trading site has four persons registered: (p1, Chang, ch@cmail.ca, 514-123-4567), (p2, Barrett, ba@bmail.ca, 905-987-6543), (p3, Mareda, ma@mmail.ca, 111-111-1111), (p4, Gauthier, ga@gamail.ca, 999-999-999).

   p1 and p2 are sellers, p3 is a buyer, and p4 is seller and buyer. p3 has rated p1 with 4 stars ("no complaints so for"), and the rating has been perceived useful by one person so far (useful = 1). p3 has also rated p2 with 2 stars and no comment, and that rating hasn't been perceived useful by anyone so far. p4 has only rated p1 with 5 stars ("best seller ever"). Nobody has perceived that rating as useful so far.

How would you model this E/R diagram and the associated data using XML? Give an example XML document that contains exactly all entities and relationships indicated above.



2. Given an XML document *report.xml* whose structure obeys to the following DTD

```
<!DOCTYPE report [
<!ELEMENT report (title,section+)>
<!ELEMENT section (title,body?,section*)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT body (para+)>
<!ELEMENT para(#PCDATA)>
<!ATTLIST book version CDATA #REQUIRED>
<!ATTLIST section number ID CDATA #REQUIRED>
]>
```

Write XQuery or XPath statements for the following queries.

(a) (3 Points) Return the title of section with section number 4

(b) (3 Points) Return the titles of all sections

(c) (5 Points) Return the numbers and titles of all sections that have at least 10 paragraphs in their body or 5 nested sections. For each such section the output should be:
$< large >$ number title $< /large >$

(d) (5 Points) Return all titles that appear at least twice (i.e., two sections have the same title). Make sure that you really catch two different sections. You can solve this with a nested loop and join in XQuery