Sample Solution - Assignment 3

(COMP 353 – Summer 2018)

Question 1

Give a list of clients(s) with the highest number of contracts in each Line of Business (LOB).

This query is completed in several steps. First the number of contracts that each company possesses is computed in a subquery requiring aggregation by company id and line of business. The maximum per line of business is then computed. Finally, an inner join with the company information itself is done.

Listing 1 – SQL query for question 1

```
SELECT maxContractsPerLob.lineOfBusiness, Company.id, Company.name, contractsCount
FROM Company
  INNER JOIN (
          -- Find how many contracts each company has per lob and inner join on max contract count to select only those
          -- companies that have the maximum contract count per lob
          SELECT companyId, lineOfBusiness, COUNT(*) AS contractsCount
          FROM Contract
          GROUP BY LineOfBusiness, CompanyId
          ) AS contractsPerLobAndCompany
    ON contractsPerLobAndCompany.companyId = Company.id
  INNER JOIN
  -- Select max contracts count per lob
  (SELECT lineOfBusiness, MAX(contractsCount) maxContractsCount
   FROM (
        -- Find how many contracts each company has per lob
        SELECT companyId, lineOfBusiness, COUNT(*) AS contractsCount
        FROM Contract
        GROUP BY LineOfBusiness, CompanyId
       ) AS contractsPerLobAndCompany
  GROUP BY lineOfBusiness) AS maxContractsPerLob
   ON maxContractsPerLob.maxContractsCount = contractsPerLobAndCompany.contractsCount AND
     maxContractsPerLob.lineOfBusiness = contractsPerLobAndCompany.lineOfBusiness;
```

Question 2

Give the details of the contracts started within the last 10 days with ACV between $10,000 to $20,000. This should be a generic query providing details for last 10 days from current date.

This query can be completed by simply querying contracts with the specified condition. An inner join is added to retrieve the company information for each contract as well.

Listing 2 – SQL query for question 2

```
SELECT * FROM Contract
INNER JOIN Company ON Contract.companyId = Company.id
WHERE startDate > NOW() - INTERVAL 10 DAY
    AND annualContractValue >= 10000
    AND annualContractValue <= 20000;
```

Question 3

Find the information of the clients who signed contract from "Quebec".

The query searches for company who are based in the Quebec province. An inner join with the Contracts table is done to ensure that at least one contract has been signed.

Listing 3 – SQL query for question 3

```
-- We use the inner join to ensure that at least one contract has been signed
SELECT Company.id, Company.name, Company.province, Company.name FROM Company
  INNER JOIN Contract on Company.id = Contract.companyId
WHERE province = 'Quebec';
```

Question 4

Find all the clients whose contracts, in each category, has the highest productivity of employees working on the project and based in Montreal.

For this question, it is assumed that productivity of employees working on a contract are measured in the number of deliverables completed per time unit. Thus, to complete this query, the productivity per contract is first computed in a nested subquery requiring aggregation by the contract id. Next the contract with the highest productivity per category is computed. Finally, a join with the Company table is done to determine the company responsible for these contracts, filtered by city = 'Montreal'.

Listing 4 – SQL query for question 4

```
SELECT MaxProductivityPerCategory.category, Company.*, maxProductivity FROM Company
  INNER JOIN Contract ON Company.id = Contract.companyId
  INNER JOIN (
    SELECT contractId, COUNT(*)/DATEDIFF(MAX(endDate), MIN(startDate)) AS deliverablesPerDay
FROM Deliverable
    WHERE endDate IS NOT NULL
    GROUP BY contractId
    ) AS ContractProductivity ON ContractProductivity.contractId = Contract.id
  -- Inner join to select those companies that have the maximum amount
  INNER JOIN (
    -- Find max productivity per category
    SELECT category, MAX(Productivity.deliverablesPerDay) AS maxProductivity FROM Contract
    INNER JOIN (
      -- Find productivity per contract
      SELECT contractId, COUNT(*)/DATEDIFF(MAX(endDate), MIN(startDate)) AS deliverablesPerDay
FROM Deliverable
      WHERE endDate IS NOT NULL
      GROUP BY contractId
            ) AS Productivity ON Productivity.contractId = Contract.id
    GROUP BY category
  ) AS MaxProductivityPerCategory ON Contract.category = MaxProductivityPerCategory.category AND
MaxProductivityPerCategory.maxProductivity = deliverablesPerDay
  WHERE city = 'Montreal';
```

Question 5

Considering an attrition rate of 8.7% annually, find the number of employees available to work after June-2018 on all the contracts.

To find the number of employees available, the time difference between the current date and the target date in years is computed and multiplied by the attrition rate of 8.7%. This is the estimated amount of employees who will leave the company during that interval and is subtracted from the current number of employees. Since June 2018 is in the past I instead interpret this question to be treated as if we are considering the first day the system is launched to June 2018. The first day is estimated by finding the first contract start date.

Listing 5 – SQL query for question 5

```
-- Since June 2018 is in the past, I instead interpret this question to be treated as if we are considering the first
-- day the system is launched to June 2018. The first day is obtained from the first contract starting date.
SELECT ROUND(COUNT(DISTINCT Employee.id) * POWER((1-0.087),(DATEDIFF('2018-07-01',
(SELECT MIN(Contract.startDate) FROM Contract))/365)), 0)
  AS AvailableEmployees
FROM Employee;
```

1 Relational Algebra Questions

1.1 Question 1

Find the number of the clients who signed the contracts providing "Cloud services".

Note that the G_{count} operator is used to apply the aggregate count operator the number of distinct companies in the result of the selection. This operator was not covered in the lecture notes however it is standard relational algebra [1]. We also rely on the fact that intermediate results are treated as sets by default in order to eliminate duplicate company ids and thus the count is over distinct elements.

$$G_{count(companyId)}(\sigma_{service\ =\text{"}Cloud\ services\text{"}}(contract))$$

1.2 Question 2

Get the list of clients who have signed multiple contracts.

A theta-join on the contracts relation is done with itself in order to find companies having more than 1 contract. Records are searched where the company ids are the same but the contract id differs. The final projection operator will eliminate duplicate company ids.

$$\pi_{companyId}(\rho_{c1}(contract) \bowtie_{c1.companyId=c2.companyId\ and\ c1.contractId<>c2.contractId} \rho_{c2}(contract))$$

1.3 Question 3

Find all the clients who have signed up for the premium contract delivery and the employees working on that contract have a "Normal employee plan".

For this question it is assumed that the statement "the employees working on the contract have a Normal employee plan" means that all employees working on the contract have the normal employee plan. To find this quantity we subtract those contracts that have at least one employee with an employee plan other than the normal plan.

$$\pi_{companyId}\left(\sigma_{category="Premium"}\left(contract \bowtie_{Contract.companyId=Company.id}\ company\right)\right) -$$

$$\pi_{companyId}(\sigma_{Employee.insurancePlan<>"Normal"}\left(contract \bowtie_{Contract.id=Assignment.contractId}\right.$$
$$\left.\bowtie\ (assignment \bowtie_{Assignment.employeeId=Employee.id}\ employee)\right))$$

1.4 Question 4

For the last 30-days find the employees who have resigned.

To complete this query the employees where the resigned date attribute has a value set in the last 30 days are searched. Note that the resigned date attribute does not exist during the SQL exercises. It has been added for this question only.

$$\pi_{id}(\sigma_{resignedDate \geq\ now\ -\ 30days}(employee))$$

1.5 Question 5

Find the employees who have worked on all contracts

This can be done conveniently with the division operator:

$$Employee \bowtie (\pi_{employeeId,\ contractId}(assignment) \div \pi_{contractId}(contract))$$

The $\pi_{employeeId,\ contractId}(assignment) \div \pi_{contractId}(contract)$ subquery will return all assignments belonging to employees that participate in all possible contracts. A natural join on the employee table is then done to receive complete information of each employee.