Database Management Systems

Practice Problem Set II: SQL

Q1. Consider a SQL database being used to store a directed graph. The database has one table: `Edge(n1,n2)`. A tuple `(X,Y)` in this table encodes the fact that there is a directed edge from the node with identifier `X` to the node with identifier `Y`. There are no duplicates in the table. You can assume that every node in the graph is involved in at least one edge. (Otherwise the database would need to have a separate `Node` table.)

(a) Write a SQL query to find the identifiers of the nodes in the graph with the *highest out-degree*. That is, find all nodes that have at least as many outgoing edges as any other node in the graph.

(b) You probably used `Group By` in your solution to part (a). If you used `Group By` in your solution to part (a), now write the same query without using `Group By`. If you didn't use `Group By` in part (a), now write the query using `Group By`. Which one do you like better?

(c) Modify either your solution for part (a) or for part (b) to find the identifiers of the nodes in the graph with the *highest in-degree*. That is, find all nodes that have at least as many incoming edges as any other node in the graph.

(d) Write a SQL query to find a *directed path* from the node with identifier `X` to the node with identifier `Y`. Specifically, your query should return a set of node identifiers such that `X` and `Y` are in the set, and the nodes in the set lay out a path starting at node `X` and ending at node `Y`. You can make several important assumptions:

- `X` <> `Y`, i.e., `X` and `Y` are different nodes.
- There is exactly one path from `X` to `Y`.
- The *diameter* of the graph (i.e., the longest path in the entire graph) is at most length 5.
- The graph has no cycles. [Technically, this assumption is implied by the previous one.]

You may return the set of node identifiers either as a one-attribute table with one tuple for each node, or as a single tuple whose attribute values contain the identifiers. We are not concerned about duplicates or NULLs -- either may be included as long as the set of non-NULL node identifiers is correct.

(e) In the query you wrote for part (d), what would happen if `X` = `Y`, i.e., `X` and `Y` are the same node? What would happen if there's no path from `X` to `Y`? What would happen if there are multiple paths from `X` to `Y`?

(f) Can you expand or modify your solution to part (d) to find the *shortest directed path* from node `X` to the node `Y`? Specifically, your query should return a set of node identifiers such that `X` and `Y` are in the set, the nodes in the set lay out a path starting at node `X` and ending at node `Y`, and there is no smaller set of nodes that also lays out a path from `X` to `Y`. Assumptions for this problem are as follows; only the second one has changed from part (e):

- `X` <> `Y`, i.e., `X` and `Y` are different nodes.
- There is exactly one shortest path from `X` to `Y`.
- The *diameter* of the graph (i.e., the longest path in the entire graph) is at most length 5.
- The graph has no cycles. [Technically, this assumption is implied by the previous one.]

Q2. The intinno portal you're using for this class is backed by a SQL database that maintains scores for all submissions of assignments. Consider the table: `Scores(sID,aID,timestamp,score)`, where `sID` identifies a student, `aID` identifies an assignment, and the three attributes `[sID,aID,timestamp]` together form the only key.

Write a SQL query to find the IDs of all students whose highest score on at least one assignment is lower than the average score for that same assignment across all submissions of all students. No duplicates please.

Q3. Consider the schema :

CLIENT (client_id, name, city);
PRODUCT(product_id, description, cost);
SALESMAN(salesman_id, name, target, city);
SALES_ORDER(order_id, date, client_id, salesman_id);
ORDER_DETAILS(order_id, product_id, quantity);

For each relation, the fields with _id suffix together form the primary key except in the case of ALES_ORDER where order_id is the primary key.

Write SQL queries to implement the following. Make sure to eliminate duplicates where they may be generated.

1. Print descriptions of products that have been ordered atleast once.
2. Print all cities in which atleast one client or salesman is located.
3. Print names of all salesmen in cities with more than one salesman
4. Print names of clients who have ordered products before a specified date (say 15 August 2013).
5. Print descriptions of all products ordered by chedi (client name equals chedi)
6. Print descriptions of all products ordered by chedi and sold by a salesman located in the city "Hijli"
7. Print the names of all salesmen who have sold tinku (product description equals tinku)

Q4. For the schema in question 3, implement the queries listed below in SQL.

1. List all salesmen along with the total cost of all items sold by them, in decreasing order of total cost.

2. As in part 1, but grouped by city, with salesmen within the city listed in decreasing order.

3. As in part 2, but grouped by product. That is, for each product, list all salesmen who sold the product, in decreasing order of quantity of product they sold. Sort products in alphabetical order.

4. List the top city by total sales to all clients based in each city, as well as all cities with total sales within 50 percent of the top city's sales. Again, sort in decreasing order of total sales.

5. Find the weighted average cost of all products sold (i.e., weighted by sales, so if you sell 2 units of cost 1 and 1 of cost 4, the average is 2)

6. Find the fraction of all sales due to products with cost less than the weighted average cost.

7. List all salesmen who sold more (by total cost) to clients outside their city than to clients in their city. Print out the total sales in each of these categories for each of these salesmen.