# Assignment 2: Recursive Queries

Due date: March 7, 2021, 11:55pm IST

**General Instructions**

1. Please complete this assignment *individually.*
2. You will submit just 1 file: query.sql.
3. Use PostgreSQL 13 for your homework. See **this link** for instructions on how to download and install it on your OS. The .sql files are run automatically using the psql command using the \i option, so please ensure that there are no syntax errors in the file. *If we are unable to run your file, you get an automatic reduction to 0 marks.*

   To understand how to run many queries at once from text file, a dummy query file **example.sql** is available. To run **example.sql** in PostgreSQL, type the following command in the terminal:

   sudo -u postgres psql *dbname*

   `\i /address/to/example.sql`

   This command will run all the queries listed in example.sql at once.
4. The format of the file should be as follows. You can have a preamble where you may create views if you like (please note that no procedures are allowed (this has to be pure SQL), and correspondingly have a cleanup section where these views are removed. One line should identify the query number (note the two hyphens before and after the query number), followed by the actual, syntactically correct SQL query. Leave a blank line after each query.

   - -PREAMBLE- -

   OPTIONAL DEFINITIONS

   - -1- -

   SQL QUERY

   - -2- -

   SQL QUERY

   - -3- -

   SQL QUERY

   - -CLEANUP- -

   CLEANUP EVERYTHING YOU CREATED HERE
5. Some of the queries below require an 'ORDER BY' clause. If you made an error in this clause, your answer may be evaluated as incorrect, giving you zero marks for that query.
6. The submission will be done on Moodle. No changes are allowed in attribute names or table names.
7. If unspecified, order in ascending order by column 1, then column 2 .. etc. In case of any doubts please ask on Piazza. The instructors ordering will be final and no queries will be entertained on the same.
8. There are no NULL values in the dataset, so you need not worry about that.
9. There is no data provided for these queries. For testing, make your own dataset. The .sql file that you submit however, should contain only queries
10. If any query asks you to output top $x$ rows and you are getting $y$ rows, output $min(x, y)$ rows.
11. You are allowed to submit the assignment one day late with a penalty of 25% of assignment marks.

# 1 Dataset 1

1. In this first part of the assignment, you'll work with a real world Flights and Airports dataset. The schema of the same is described below, but we won't share the actual dataset that will be used for evaluation.

2. The database will include following two tables and you should use only these tables while writing solution of the queries. You can create temporary views while handling any SQL query but you should include SQL queries for creating and deleting these temporary views at the starting and end of your SQL file respectively. Note - you don't have to define these tables in the submission file, these will already be present while evaluation.

   (a) airports

   | airportid : integer | city : text | state : text | name : text |
   |---|---|---|---|

   (b) flights

   | flightid : integer | originairportid : integer | destairportid : integer | carrier : text |
   |---|---|---|---|
   | dayofmonth : integer | dayofweek : integer | departuredelay : integer | arrivaldelay : integer |

3. Keys:

   (a) `airportid` is primary key for `airports` table

   (b) `flightid` is primary key for `flights` table

4. Flight `f` is a direct flight between airports `A` and `B` if there exists a flight `f` in `flights` table such that `f.originairportid = A`, `f.destairportid = B`

5. Airports `a` and `b` are said to be connected via a series of connecting flights if there exist a sequence of flights `f0`, `f1`, ... , `fn`: each a direct flight between `(a, a1)`, `(a1, a2)`, `(a2, a3)`, ..., `(an, b)` respectively.

6. Note that in the `airports` table, a city can potentially have multiple airports. If we want you to consider a particular airport for a city, we will mention the `airportid`.

7. In `flights` table, the `dayofmonth` represents the calendar date of the month and lies in the range $[1, 31]$

8. In `flights` table, the `dayofweek` represents the enumerated index of the days of week (Sunday, Monday ... Saturday) and lies in the range $[1, 7]$

9. In `flights` table, the `arrivaldelay` and `departuredelay` represents the numeric value of the delay (in minutes) where a negative value of delay signifies the flight was early. For eg: `departuredelay = -10` will signify the flight departed 10 minutes early.

# 2 Queries

**Air Transport Network** is an example of transport networks and spatial networks. The nodes of the network are the airports and the links represent direct flight routes between two airports. Alternatively, cities can be considered as the nodes with links representing direct flight connection between them. Air transport networks can be defined worldwide as well as for one region or for one airline company; the scale of the network can be global or domestic

Consider the collaboration network $G$ formed by the `airports` and `flights` tables. The nodes for this graph will be the airports, and there will exist a directed edge from airport A to B iff there is a flight from A to B. There is an edge from airport A to B are if there exists a direct flight between A and B.

Mathematically, $G = (V, E)$ where:

- $V = \{$`airports.airportid`$\}$
- $E = \{(u, v) : \exists$ `f` in `flights` s.t. `f.originairportid = u` and `f.destairportid = v` $\}$

1. Find all cities reachable from **Albuquerque** (airportid: **10140**) through a series of one or more connecting flights by the same carrier.
   **Output column: name. Sort output by name in ascending order.**

2. Find all cities reachable from **Albuquerque** (airportid: **10140**) through a series of one or more connecting flights, with all connecting flights operating on the same day of the week.
   **Output column: name. Sort output by name in ascending order.**

3. Find all the cities reachable from **Albuquerque** (airportid: **10140**) by exactly one path, ie list all cities 'c' such that there is exactly one path from Albuquerque to 'c'.
   **Output column: name. Sort output by name in ascending order.**

4. Find the length of the longest possible circular trip containing **Albuquerque** (airportid: **10140**) through a series of one or more connecting flights. "Longest" here is defined by the number of stops (or the number of cities).
   **Output column: length.**

5. Find the length of the longest possible circular trip through a series of one or more connecting flights. "Longest" here is defined by the number of stops (or the number of cities).
   **Output column: length.**

6. Given a source city (**Albuquerque**) and a destination city (**Chicago**) return the number of paths between these two cities through interstate flights. Interstate means the state of the source city should be different than the state of the destination.
   **Output column: count.**

7. Given a source city (**Albuquerque**) and a destination city (**Chicago**) return the number of paths between these two cities which also pass through Washington.
   **Output column: count.**

8. Find all the pairs of cities (c1, c2) such that there is no path from c1 to c2.
   **Output columns: (name1, name2). Sort your output based on the first column in ascending order. Resolve ties using the second column.**

9. When taking a flight from Albuquerque, output days of the Month when you will face the least amount of delays. Delays here means the sum of arrival delay and departure delay.
   **Output column: day. Sort output of days of the month based on the increasing amount of delays. If 2 days have the same delay, then order by date.**

10. Find all the cities in the State of "**New York**" which act as a travel center for the state, i.e. all the cities in the state can be visited via that city through direct flights.
    **Output column: name. Sort output by name in ascending order.**

11. Find all the pairs of cities (c1, c2) such that there is a path from c1 to c2, such that successive delay times between the consecutive cities is in increasing order (delay time here is the sum of arrival and departure delays).
    **Output columns: (name1, name2). Sort your output based on the first column.**

# 3 Dataset 2

1. In this second part of the assignment, you'll work with a synthetically generated dataset. The schema of the same is described below, but we won't share the actual dataset that will be used for evaluation.

2. The database will include following four tables and you should use only these tables while writing solution of the queries. You can create temporary views while handling any SQL query but you should include SQL queries for creating and deleting these temporary views at the starting and end of your SQL file respectively. Note - you don't have to define these tables in the submission file, these will already be present while evaluation.

   (a) authordetails

| authorid : integer | authorname : text | city : text | gender : text | age : integer |
|---|---|---|---|---|

(b) paperdetails

| paperid : integer | papername : text | conferencename : text | score : integer |
|---|---|---|---|

(c) authorpaperlist

| authorid : integer | paperid : integer |
|---|---|

(d) citationlist

| paperid1 : integer | paperid2 : integer |
|---|---|

3. Keys:
   (a) `authorid` is primary key for `authordetails` table
   (b) `paperid` is primary key for `paperdetails` table
   (c) `(authorid, paperid)` is primary key for `authorpaperlist` table
   (d) `(paperid1, paperid2)` is primary key for `citationlist` table
4. Authors `a` and `b` are connected by an edge if there exists a paper with paperid `p` such that `(a, p)` $\in$ `authorpaperlist` and `(b, p)` $\in$ `authorpaperlist`.
5. Authors `a` and `b` are connected if there exist a sequence of authors `a1`, `a2`, ... , `an` such that there is an edge between `(a, a1)`, `(a1, a2)`, `(a2, a3)`, ..., `(an, b)`.
6. A paper with paperid `p` cites a paper with paperid `c` if `(p, c)` $\in$ `citationlist`
7. `(p, c)` is treated as an direct citation if `(p, c)` $\in$ `citationlist`.
8. `(p, c)` is treated as an indirect citation if there exist a sequence of papers with paperids `p1`, `p2`, ... , `pn` such that `(p, p1)`, `(p1, p2)`, `(p2, p3)`, ..., `(pn, c)` $\in$ `citationlist`.

# 4 Queries

**Scientific collaboration network** is a social network where nodes are scientists and links are co-authorships as the latter is one of the most well documented forms of scientific collaboration.It is an undirected, scale-free network where the degree distribution follows a power law with an exponential cutoff - most authors are sparsely connected while a few authors are intensively connected

Consider the collaboration network $G$ formed by the `authorpaperlist` and `authordetails` and `paperdetails` tables. The nodes for this graph will be the authors, and there will exist an edge between them iff they have co-authored a paper. Two authors are connected by an edge if there exist a paper `p` auch that `(author1, p)`, `(author2, p)` is present in `citationlist` table. In the queries that follow, "components" refer to the usual components in a graph.

Mathematically, $G = (V, E)$ where:

- $V = \{$`authordetails.authorid`$\}$
- $E = \{(u, v, p) : (u, p) \in$ `authorpaperlist` and $(v, p) \in$ `authorpaperlist`$; p \in$ `paperdetails.paperid`$; u, v \in$ `authordetails.authorid`$\}$

12. Find the length of shortest path from author **A** to every other author in G. If the shortest path goes like: A $\rightarrow a_1 \rightarrow a_2 \rightarrow \cdots \rightarrow a_n \rightarrow$ `B`, output $n + 1$. If there is no path from author **A** to some author, output -1. Give the author id, length of shortest path from author **A**. Sort in descending order of length of shortest path and resolve all ties by ascending order of destination authorid.
    **Output columns: authorid, length. Take A = 1235**
13. Find the number of paths in Graph G from author **A** to **B** such that all authors on the path have an age more than 35 and consecutive authors on the path have different genders. Return -1 if they don't belong to the same component, else return number of paths.
    **Output column: count. Take A = 1558, B = 2826.**

14. Find the number of paths from **A** to **B** in $G$ such that at least one person on the path has directly or indirectly cited a paper with paperid **p**. Return -1 if they don't belong to the same component, else return number of paths. **Output column: count. Take A = 704, B = 102, p=126**

15. Find the number of paths in graph $G$ from author **A** to **B** such that the total number of citations of authors on the path is in strictly increasing order or strictly decreasing order. The total number of citations of an author is defined as the sum of citations of papers that the author is associated with. Return -1 if they don't belong to the same component, else return number of paths.
**Output column: count. Take A = 1745, B = 456**

16. We define "number of future collaborations of a author A" as authors whose papers has been cited by author A and haven't co-authored a paper with author A. Give the author ids of top 10 people with most number of future collaborations. Sort in descending order of number of future collaborations and resolve all ties by ascending order of author ids. **Output column: authorid**

17. Author A's third degree connections are people connected to him with shortest path length of 3 in graph $G$. Give the author ids of top 10 authors with the most number of citations of papers written by their third degree connections. Sort in descending order of number of citations and resolve all ties by ascending order of user ids. **Output column: authorid**

18. Find the number of paths in Graph G from author **A** and **B** that also pass through at least one of the authors **C1**, **C2**, **C3**. Note that A $\rightarrow a_1 \rightarrow C \rightarrow a_2 \rightarrow$ B and A $\rightarrow a_1 \rightarrow C \rightarrow a_3 \rightarrow$ B constitute two distinct paths in graph $G$, that is, in path from A to B (via C), if even a single node is different (may be in A$\rightarrow$C part or in C$\rightarrow$B part), then it counts as a different path. All the vertices in the path $A \rightarrow C$ should be distinct, that is, a path like $A \rightarrow \cdots a_1 \rightarrow \cdots B \rightarrow \cdots a_1 \rightarrow \cdots C$ is not valid. Return -1 if they don't belong to the same component.
**Output column: count. Take A = 3552, B = 321, C1 = 1436, C2 = 562, C3=921**

19. Find the number of paths in Graph $G$ from author **A** to **B** such that no two authors on the path are from the same city and no two persons have directly cited each other's papers. Return -1 if they don't belong to the same component, else return number of paths.
**Output column: count. Take A = 3552, B = 321**

20. Find the number of paths in Graph $G$ from author **A** to **B** such that any author on the path hasn't directly or indirectly cited any other author on that path. Return -1 if they don't belong to the same component, else return number of paths. **Output column: count. Take A = 3552, B = 321**

21. We say "authors A and B are conference(X)-connected" if there exists a path from author A to B in G where all the edges on the path correspond to papers published in conference X. A conference connected component is a subgraph(H) of G where every pair of vertices in H are conference-connected. For every conference X, find total number of conference(X) connected components in $G$. Sort in descending order of conference connected components and resolve all ties by ascending order of conferencename.
**Output columns: conferencename, count**

22. For each conference, give the conferencename, number of authors in each conference connected component of $G$. If a conference has three connected components then the output should have three rows with the conference name and the number of authors in each component. Sort in ascending order of number of authors and resolve all ties by ascending order of conferencename.
**Output columns: conferencename, count**