

**CSC 370 - SUMMER 2020
DATABASE SYSTEMS
ASSIGNMENT 3
UNIVERSITY OF VICTORIA**

Due: Thursday, June 18th, 2020 at **23:55 Victoria Time (PDT)**. **Late assignments will not be accepted.**

This assignment will be accepted electronically. See the ‘Submission and Evaluation’ section below for details on the submission process and expected formatting of your answers. For all of the questions below, your answer must be **one** SQL query (including a terminating semicolon) which runs without errors on the `studdb1.csc.uvic.ca` or `studdb2.csc.uvic.ca` PostgreSQL database servers. Note that timeout errors (in which the server terminates your query for exceeding the maximum execution time) are considered errors. Queries which have errors or which produce incorrect output will receive a mark of zero. Queries which run without errors and produce the correct output will be marked out of two, with full marks given only to queries that contain no assumptions besides what is given in the question (see the advice sections below for more details).

The ordering of rows in your query result will not be considered during marking (so it is not necessary that your query produce the same row ordering as the model output, just the same set of rows). However, your query must have the same set of columns (with the same names, in the same order) as the model output to be considered correct.

Question 1: IMDB Queries [12 marks]

Create queries for each of the data retrieval problems below, using the `imdb` database. Place your answers in the appropriate positions in the `a3q1_queries.txt` file. In the questions below, any reference to ‘films’ refers to titles with `title_type = 'movie'`.

Hint: When you need the ‘primary name’ of a title, add the following to the `WITH` clause of your query and then join the resulting `primary_names` subquery to `titles` as needed.

```
primary_names as (select title_id, name as primary_name
                    from title_names where is_primary = true)
```

For example, the query below uses the ‘primary name’ to find the production year, title ID and duration of all films (`title_type = 'movie'`) with primary title ‘The Shining’.

with

```
primary_names as (select title_id, name as primary_name
                    from title_names where is_primary = true)
```

```
select *
```

```
from
```

```
    titles
```

```
  natural join
```

```
    primary_names
```

```
where primary_name = 'The Shining' and title_type = 'movie';
```

The result of the above query is shown below.

Query Result				
title_id	title_type	year	length_minutes	primary_name
6812278	movie	2017	136	The Shining
81505	movie	1980	146	The Shining

- (a) Find the primary name, year and title ID of all titles from the year 1989 with a length of 180 minutes and a title type of ‘tvSpecial’.

Expected Query Result		
primary_name	year	title_id
Survivor Series	1989	264059
Starrcade	1989	348114
The 16th Annual American Music Awards	1989	790592
The 1989 Miss Tennessee Pageant	1989	1837666

- (b) Find the primary name, year and length (in minutes) of all films whose total length in minutes is at least 4320 (72 hours).

Expected Query Result		
primary_name	year	length_minutes
Modern Times Forever	2011	14400
Beijing 2003	2004	9000
Nari	2017	6017
Hunger!	2015	6000
London EC1	2015	5460
The Cure for Insomnia	1987	5220
Ember Glow	2015	4980
Fail	2016	4680
Writing on Snow	2017	4320

- (c) Find the primary name, year and length (in minutes) of all films which have ‘Meryl Streep’ in the cast/crew **and** have a production year of 1985 or earlier. Note: Use the `cast_crew` table, not the `known_for` table.

Expected Query Result		
primary_name	year	length_minutes
Kramer vs. Kramer	1979	105
The Seduction of Joe Tynan	1979	108
The French Lieutenant’s Woman	1981	124
Sophie’s Choice	1982	150
Still of the Night	1982	93
Silkwood	1983	131
Falling in Love	1984	106
Out of Africa	1985	161
Plenty	1985	121

- (d) Find the primary name, year and length (in minutes) of all films which are associated with **both** of the genres ‘Film-Noir’ and ‘Action’. Use the `title_genres` table to map titles to

their genres (titles may have any number of genres).

Expected Query Result		
primary_name	year	length_minutes
Blackmail	1947	67
Dangerous Mission	1954	75
Dick Tracy	1945	61
Dick Tracy vs. Cueball	1946	62
His Kind of Woman	1951	120
Peking Express	1951	95
Road House	1948	95
Rogues' Regiment	1948	86
Scotland Yard Investigator	1945	68
Sirocco	1951	98
The Pay Off	1942	74
Unmasked	1950	60

- (e) Find the names of all people who were associated as writers or directors (using the appropriate relationship tables) with the movie 'Die Hard'. There is only one title in the database which is both a movie and has primary name 'Die Hard'.

Expected Query Result
name
Jeb Stuart
John McTiernan
Roderick Thorp
Steven E. de Souza

- (f) Find the primary name, year and length (in minutes) of all films which have **both** 'Tom Hanks' and 'Meryl Streep' as cast/crew. Use the `cast_crew` table, not the `known_for` table.

Expected Query Result		
primary_name	year	length_minutes
Everything Is Copy	2015	89
The Ant Bully	2006	88
The Post	2017	115

Question 2: BC Ferries Queries [16 marks]

The queries you write below should work correctly on all of the BC Ferries databases (`ferries_1month`, `ferries_3months`, `ferries_6months`, `ferries_9months`, `ferries_1year` or `ferries_3years`). For comparison, sample output is shown for both `ferries_1month` and `ferries_3years` (but for marking, your query will be compared to a model solution on all of the different databases).

- (a) Print the total number of sailings for each route number for all routes in the `routes` table, including routes that have zero sailings in the dataset.

Expected Query Result (ferries_1month)	
route_number	num_sailings
1	297
2	0
3	342
4	234
8	440
30	395

Expected Query Result (ferries_3years)	
route_number	num_sailings
1	24160
2	16895
3	17544
4	8419
8	15419
30	15504

- (b) Print the total number of sailings per vessel in the database, showing only those vessels with at least one sailing.

Expected Query Result (ferries_1month)	
vessel_name	count
Coastal Inspiration	232
Coastal Renaissance	39
Queen of Alberni	163
Queen of Capilano	440
Queen of Coquitlam	68
Queen of Surrey	274
Skeena Queen	234
Spirit of British Columbia	164
Spirit of Vancouver Island	94

Expected Query Result (ferries_3years)	
vessel_name	count
Bowen Queen	1268
Coastal Celebration	6506
Coastal Inspiration	6545
Coastal Renaissance	5637
Mayne Queen	1
Queen of Alberni	6903
Queen of Capilano	14277
Queen of Coquitlam	6650
Queen of Cowichan	7071
Queen of Cumberland	974
Queen of New Westminster	3966
Queen of Oak Bay	6609
Queen of Surrey	12759
Quinitsa	8
Skeena Queen	7312
Spirit of British Columbia	5539
Spirit of Vancouver Island	5916

- (c) Print the route number and number of vessels serving that route for every route which was served by at least two vessels (that is, all routes for which at least two distinct vessels each had at least one sailing)

Expected Query Result (ferries_1month)	
route_number	num_vessels
1	3
3	2
30	2

Expected Query Result (ferries_3years)	
route_number	num_vessels
1	6
2	5
3	6
4	5
8	2
30	6

- (d) For each route which appears at least once in the sailings table, print the route number and the name and production year of the oldest vessel(s) on the route. Note that there may be multiple vessels tied for oldest (each should be printed separately). Hint: You will likely need a join on a subquery.

Expected Query Result (ferries_1month)		
route_number	vessel_name	year_built
1	Spirit of British Columbia	1993
3	Queen of Coquitlam	1976
4	Skeena Queen	1997
8	Queen of Capilano	1991
30	Queen of Alberni	1976

Expected Query Result (ferries_3years)		
route_number	vessel_name	year_built
1	Queen of New Westminster	1964
2	Queen of Coquitlam	1976
2	Queen of Cowichan	1976
3	Bowen Queen	1965
4	Bowen Queen	1965
4	Mayne Queen	1965
8	Bowen Queen	1965
30	Queen of New Westminster	1964

- (e) List all vessels which used any port (source or destination) that was at any time used (as either source or destination) by the vessel 'Coastal Renaissance'. The result should contain the Coastal Renaissance itself. Remember not to make any assumptions about the data.

Expected Query Result (ferries_1month)	
vessel_name	
Coastal Inspiration	
Coastal Renaissance	
Queen of Alberni	
Skeena Queen	
Spirit of British Columbia	
Spirit of Vancouver Island	

Expected Query Result (ferries_3years)	
vessel_name	
Bowen Queen	
Coastal Celebration	
Coastal Inspiration	
Coastal Renaissance	
Mayne Queen	
Queen of Alberni	
Queen of Capilano	
Queen of Coquitlam	
Queen of Cowichan	
Queen of Cumberland	
Queen of New Westminster	
Queen of Oak Bay	
Queen of Surrey	
Quintisa	
Skeena Queen	
Spirit of British Columbia	
Spirit of Vancouver Island	

- (f) Print the route number and the number of vessels serving that route for every route served by the largest number of distinct vessels (compared to other routes). Notice that there may be multiple routes which are tied for having the most vessels (so the result may have multiple rows).

Expected Query Result (ferries_1month)	
route_number	num_vessels
1	3

Expected Query Result (ferries_3years)	
route_number	num_vessels
1	6
3	6
30	6

- (g) For each port and route number, list the total number of sailings for that route that served that port (as either a source or destination). Only list port/route pairs that have at least one sailing.

Expected Query Result (ferries_1month)		
port	route_number	sailings
Bowen Island	8	440
Duke Point	30	395
Fulford Harbour	4	234
Horseshoe Bay	3	342
Horseshoe Bay	8	440
Langdale	3	342
Swartz Bay	1	297
Swartz Bay	4	234
Tsawwassen	1	297
Tsawwassen	30	395

Expected Query Result (ferries_3years)		
port	route_number	sailings
Bowen Island	8	15419
Departure Bay	2	16895
Duke Point	30	15504
Fulford Harbour	4	8419
Horseshoe Bay	2	16895
Horseshoe Bay	3	17544
Horseshoe Bay	8	15419
Langdale	3	17544
Swartz Bay	1	24160
Swartz Bay	4	8419
Tsawwassen	30	15504
Tsawwassen	1	24160

- (h) For each port, list the route number and number of sailings of the route(s) with the largest number of sailings which served that port (as either source or destination).

Expected Query Result (ferries_1month)		
port	route_number	sailings
Bowen Island	8	440
Duke Point	30	395
Fulford Harbour	4	234
Horseshoe Bay	8	440
Langdale	3	342
Swartz Bay	1	297
Tsawwassen	30	395

Expected Query Result (ferries_3years)		
port	route_number	sailings
Bowen Island	8	15419
Departure Bay	2	16895
Duke Point	30	15504
Fulford Harbour	4	8419
Horseshoe Bay	3	17544
Langdale	3	17544
Swartz Bay	1	24160
Tsawwassen	1	24160

Advice: Assume Nothing

You will lose marks if your query contains any ‘hard-coded’ assumptions about the data in the database other than the assumptions given in the question. For example, suppose you were asked to create the following query.

Using the **fruit** database, print the first and last names of every customer who placed an order containing the product ‘Pear’, along with the order number of their order.

One query which correctly implements the requirements above (and would receive full marks) is

```
select distinct customer_firstname, customer_lastname, order_num
from
  products
natural join
  orders
natural join
  order_contents
where products.name = 'Pear'
order by order_num asc;
```

The result of the above query is shown below.

Query Result		
customer_firstname	customer_lastname	order_num
Franz	Kafka	1001
Fiona	Framboise	1002

It is possible to write plenty of other queries which produce the same result (and any such query which makes no assumptions about the data would receive full marks). In some cases, it is possible to write a query which takes advantage of properties of the data which cannot generally be assumed. For example, if you happen to know that the product ID number for ‘Pear’ is 2, the query below would produce the same result as above.

Bad Query 1:

```
select distinct customer_firstname, customer_lastname, order_num
from
  orders
```

```
natural join
  order_contents
where product_id = 2
order by order_num asc;
```

Taking the assumptions even further, if you happen to know that the only orders meeting the criteria of the question are orders 1001 and 1002, the query below would produce the correct output as well.

Bad Query 2:

```
select distinct customer_firstname, customer_lastname, order_num
  from
    orders
  where order_num = 1001 or order_num = 1002
  order by order_num asc;
```

Both of the queries above would lose marks (and the second bad query would likely receive no marks at all), since both include hard-coded assumptions about the data in the database. If the product ID of **Pear** were to change, or more orders were added to the database, the two bad queries would no longer work (but the original, correct query would continue to work properly).

Advice: Don't Plagiarize

You are encouraged to discuss solution methods with your peers, and even to look up possible solution ideas on the internet, but all of your submitted queries must be your own work. As a rule of thumb, to ensure you do not accidentally plagiarize, do not look at anyone else's queries (or allow them to see yours). Additionally, if you use a version control system (such as Github) to store your work, ensure that the repository is private. If your queries are posted in a public setting (even inadvertently), you may become entangled in any ensuing academic integrity investigation if your work is copied by someone else.

Submission and Evaluation

This assignment will be marked through a combination of automated testing (that is, running your submitted queries and examining the result) and human inspection. To expedite the marking process (and ensure consistency between different student submissions), you are required to submit your queries for each question inside of premade query template files. Two empty template files have been posted to **conneX**: **a3q1_queries.txt** and **a3q2_queries.txt**. Please place your query for each subquestion in the indicated spaces in the templates before submitting (we will be using an automated system to extract each query individually for marking, so failure to comply with this requirement will likely result in you receiving a mark of zero for any queries which do not meet the formatting requirements). Notice that the provided files have the **.txt** extension instead of **.sql** (which would normally be used for SQL queries); this is due to a technical limitation of **conneX** (which does not properly handle files with the **.sql** extension when they are submitted, probably because it sees them as a security risk).

You are required to submit your answers to each question in two files called `a3q1_queries.txt` and `a3q2_queries.txt`, using the provided templates as a starting point. Your answer for each query must consist of a single SQL statement (which may have a `WITH` clause containing multiple subqueries, and/or several `SELECT` statements joined by set operators). As a point of reference, your answer is a ‘single SQL statement’ if it contains only one semicolon (at the end of the query). Although your files will be associated with your account, please ensure that each submitted file contains a comment with your name and student number.

You are permitted to delete and resubmit your submission as many times as you want before the due date, but no submissions or resubmissions will be accepted after the due date has passed. You will receive a mark of zero if you have not officially submitted your assignment (and received a confirmation email) before the due date.

Only the files that you submit through `conneX` will be marked. The best way to make sure your submission is correct is to download it from `conneX` after submitting and test it. You are not permitted to revise your submission after the due date, and late submissions will not be accepted, so you should ensure that you have submitted the correct version of your code before the due date. `conneX` will allow you to change your submission before the due date if you notice a mistake. After submitting your assignment, `conneX` will automatically send you a confirmation email. **If you do not receive such an email, you did not submit the assignment.** If you have problems with the submission process, send an email to the instructor **before** the due date.