CSC 370 - SPRING 2018
DATABASE SYSTEMS
ASSIGNMENT 3
UNIVERSITY OF VICTORIA

Due: Sunday, March 18th, 2018 at 11:55pm. **Late assignments will not be accepted.**

This assignment will be accepted electronically, and will be marked using the same schema as the previous assignment. 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 will receive a mark of zero. All queries without errors will be marked out of two, with full marks given only to queries which produce the correct output and contain no assumptions besides the data given in the question (see the advice sections of assignment 2 for more details).

Some of the queries in this assignment involve floating point data. In the expected output shown below, there might be slight differences in floating point values due to rounding and formatting issues. You may assume that floating point values in your query results match the expected output if they are essentially identical to the values shown (e.g. '2.999999' instead of '3.0').

Question 1: IMDB Queries [4 marks]
Create queries for each of the data retrieval problems below, using the `imdb` database. In the questions below, any reference to 'films' refers to titles with `title_type = 'movie'`.

(a) For each year between 2000 and 2017 (inclusive), list the primary name, production year, rating and number of votes of the film or films which attained the highest rating among all movies produced in that year which received at least 10000 votes. Both the rating and number of votes are stored in the `ratings` table.

Expected Query Result			
primary_name	year	rating	votes
Gladiator	2000	8.5	1095891
Memento	2000	8.5	942432
The Lord of the Rings: The Fellowship of the Ring	2001	8.8	1370113
The Lord of the Rings: The Two Towers	2002	8.7	1221886
Anbe Sivam	2003	8.9	10116
The Lord of the Rings: The Return of the King	2003	8.9	1349934
Black Friday	2004	8.6	13513
Earthlings	2005	8.7	14649
The Lives of Others	2006	8.5	290278
The Departed	2006	8.5	975612
The Prestige	2006	8.5	959259
Like Stars on Earth	2007	8.5	113421
The Dark Knight	2008	9.0	1864795
Home	2009	8.6	19621
Inception	2010	8.8	1653611
The Intouchables	2011	8.6	591006
Django Unchained	2012	8.4	1087769
The Dark Knight Rises	2012	8.4	1269644
CM101MMXI Fundamentals	2013	9.3	38106
Interstellar	2014	8.6	1120400
RangiTaranga	2015	8.7	10286
The Mountain II	2016	9.6	93071
Ayla: The Daughter of War	2017	9.1	15807

(b) Select the primary name and episode count of each TV series (contained in the `tv_series` table) for which at least 6000 episodes have been produced.

Expected Query Result	
`series_name`	`episode_count`
Days of Our Lives	10240
Ohayou Tokushima	9502
Coronation Street	9316
Neighbours	8911
Six O'Clock News	8646
The Price Is Right	8461
One O'Clock News	8100
Jeopardy!	7599
EastEnders	7393
The Bold and the Beautiful	7236
Home and Away	7081
Wheel of Fortune	6564
Gute Zeiten, schlechte Zeiten	6413
The Tonight Show Starring Johnny Carson	6037
The Young and the Restless	6024

Question 2: BC Ferries Queries [14 marks]
The queries you write below should work correctly on any of the BC Ferries databases (`ferries_1month`, `ferries_3months`, `ferries_6months`, `ferries_9months` or `ferries_12months`). For comparison, sample output is shown for both `ferries_1month` and `ferries_12months`.

(a) For many routes, there are simultaneous sailings in both directions at the same time. For example, on a typical day at 9:00am, a ferry leaves Tsawwassen for Swartz Bay and a different ferry leaves Swartz Bay for Tsawwassen. Not all routes or sailings have this property. We will define two vessels as 'paired up' if they have both served the same route number at the same (scheduled) departure time/date. Construct a query to count the number of times each distinct pair of ferries have been paired up. Your result must not contain counts for pairs of vessels which have never been paired up. Each distinct pair of ferries should appear only once in your result, and in the result rows, the vessel names of each pair must be ordered alphabetically (so the alphabetically lowest vessel name will be listed first).

Expected Query Result (`ferries_1month`)		
`vessel1`	`vessel2`	`num_pairings`
Coastal Inspiration	Queen of Alberni	204
Coastal Celebration	Coastal Renaissance	154
Queen of Coquitlam	Queen of Cowichan	90
Coastal Celebration	Spirit of Vancouver Island	86
Queen of Cowichan	Queen of Oak Bay	74
Coastal Renaissance	Queen of New Westminster	28
Coastal Renaissance	Queen of Alberni	8

Expected Query Result (`ferries_12months`)

`vessel1`	`vessel2`	`num_pairings`
Queen of Cowichan	Queen of Oak Bay	1652
Coastal Inspiration	Queen of Alberni	1634
Spirit of British Columbia	Spirit of Vancouver Island	1236
Coastal Celebration	Spirit of Vancouver Island	898
Coastal Celebration	Queen of New Westminster	714
Coastal Renaissance	Queen of Oak Bay	557
Coastal Renaissance	Queen of Alberni	532
Coastal Celebration	Coastal Renaissance	463
Queen of Coquitlam	Queen of Surrey	423
Coastal Inspiration	Queen of New Westminster	205
Coastal Renaissance	Queen of New Westminster	189
Queen of Coquitlam	Queen of Oak Bay	157
Queen of Coquitlam	Queen of Cowichan	145
Queen of Coquitlam	Queen of New Westminster	110
Coastal Renaissance	Spirit of British Columbia	110
Coastal Celebration	Spirit of British Columbia	40
Coastal Renaissance	Spirit of Vancouver Island	36
Coastal Celebration	Coastal Inspiration	24
Island Sky	Queen of Coquitlam	15

(b) The routes table contains the ‘nominal duration’ of each route, which is the expected crossing time. The nominal duration is determined by BC Ferries based on the average marine and traffic conditions, along with information like loading times and the speed of the vessels. We can test the accuracy of this calculation by computing the average time of each crossing. Construct a query to find, for each route number, the nominal duration (in minutes) and the average duration (in minutes) of a crossing based on all available data for that route. For this question, assume that the ‘duration’ of a particular sailing is the time between its scheduled departure and its arrival.

Expected Query Result (`ferries_1month`)

`route_number`	`nominal_duration`	`avg_duration`
1	95	91.0869565217
2	100	103.406374502
3	40	46.6299376299
4	35	30.7208333333
8	20	18.7871396896
30	120	122.126126126

Expected Query Result (ferries_12months)		
route_number	nominal_duration	avg_duration
1	95	93.369592089
2	100	104.379765886
3	40	45.9705357143
4	35	31.7031914894
8	20	24.307451594
30	120	121.420770263

(c) Suppose we define a sailing to be 'late' if the duration is at least five minutes longer[1] than the nominal duration in the `routes` table. Construct a query to find, for each month, the number of days for which **no** late sailings occurred at all on route number 1.

Expected Query Result (ferries_1month)	
month	count
1	17
12	7

Expected Query Result (ferries_12months)	
month	count
1	21
2	16
3	15
4	18
5	9
6	7
7	8
8	3
9	13
10	12
11	15
12	26

(d) Construct a query to find, for each vessel with any sailings, the total number of sailings it has made, the number of late sailings it has made (which may be zero) and the fraction of its sailings that were late (that is, the number of late sailings divided by the total number of sailings). You should be careful with this query: a vessel may be involved in multiple routes, each with a different nominal duration.

[1] A duration which is exactly five minutes longer is still considered late

Expected Query Result (`ferries_1month`)

`vessel_name`	`total_sailings`	`late_sailings`	`late_fraction`
Bowen Queen	334	18	0.0538922155689
Coastal Celebration	242	10	0.0413223140496
Coastal Inspiration	209	56	0.267942583732
Coastal Renaissance	216	12	0.0555555555556
Queen of Alberni	223	18	0.0807174887892
Queen of Capilano	118	3	0.0254237288136
Queen of Coquitlam	177	83	0.468926553672
Queen of Cowichan	286	58	0.202797202797
Queen of New Westminster	64	0	0.0
Queen of Oak Bay	89	22	0.247191011236
Queen of Surrey	430	120	0.279069767442
Skeena Queen	240	1	0.00416666666667
Spirit of Vancouver Island	88	0	0.0

Expected Query Result (`ferries_12months`)

`vessel_name`	`total_sailings`	`late_sailings`	`late_fraction`
Bowen Queen	411	27	0.0656934306569
Coastal Celebration	2342	247	0.105465414176
Coastal Inspiration	1983	464	0.233988905698
Coastal Renaissance	2173	263	0.12103083295
Island Sky	205	145	0.707317073171
Mayne Queen	1	0	0.0
Queen of Alberni	2255	233	0.10332594235
Queen of Capilano	4780	1580	0.330543933054
Queen of Coquitlam	2277	619	0.271848924023
Queen of Cowichan	2339	699	0.298845660539
Queen of Cumberland	18	0	0.0
Queen of New Westminster	1382	183	0.132416787265
Queen of Oak Bay	2571	932	0.362504861921
Queen of Surrey	4853	1683	0.346795796415
Skeena Queen	2724	106	0.0389133627019
Spirit of British Columbia	1425	305	0.214035087719
Spirit of Vancouver Island	2203	238	0.108034498411

(e) For each route, find the maximum number of consecutive days without any late sailings. Remember to include the cases where consecutive days without late sailings occur at the beginning and end of the dataset.

Expected Query Result (`ferries_1month`)	
`route_number`	`days_without_a_late_sailing`
1	11
2	1
3	2
4	22
8	7
30	1

Expected Query Result (`ferries_12months`)	
`route_number`	`days_without_a_late_sailing`
1	15
2	2
3	7
4	24
8	7
30	6

(f) For each route, find the maximum number of consecutive days without any late sailings and output all date ranges where that number of consecutive days was achieved. For some routes, only one date range will meet this criteria, but for others (e.g. route 2) there may be multiple date ranges of the same size. The columns containing the start and end dates should contain values of type `DATE` (created, for example, by the `make_date` function). Remember to consider successive days at the beginning and end of the dataset.

Expected Query Result (`ferries_1month`)			
`route_number`	`start_day`	`end_day`	`days_without_a_late_sailing`
1	2017-12-25	2018-01-04	11
2	2017-12-25	2017-12-25	1
2	2018-01-01	2018-01-01	1
3	2018-01-05	2018-01-06	2
3	2018-01-09	2018-01-10	2
4	2018-01-04	2018-01-25	22
8	2017-12-30	2018-01-05	7
30	2018-01-05	2018-01-05	1
30	2018-01-10	2018-01-10	1
30	2018-01-12	2018-01-12	1
30	2017-12-28	2017-12-28	1

Expected Query Result (`ferries_12months`)			
route_number	start_day	end_day	days_without_a_late_sailing
1	2017-12-21	2018-01-04	15
2	2017-12-12	2017-12-13	2
2	2017-09-26	2017-09-27	2
2	2017-03-08	2017-03-09	2
2	2017-12-24	2017-12-25	2
2	2017-02-27	2017-02-28	2
2	2017-05-20	2017-05-21	2
3	2017-02-03	2017-02-09	7
4	2017-09-24	2017-10-17	24
8	2017-12-30	2018-01-05	7
30	2017-09-23	2017-09-28	6

(g) Usually, you would expect that when the beginning of a sailing is delayed (that is, when `actual_departure` is much later than `scheduled_departure`), the arrival is also delayed and the sailing is late. However, in some cases, a vessel that departs late may still arrive on time. Define a 'made up sailing' to be any sailing which leaves at least 15 minutes after its scheduled departure but arrives less than (or equal to) five minutes late. Write a query to list the number of made up sailings in the dataset for each vessel. Only vessels with at least one made up sailing should be listed.

Expected Query Result (`ferries_1month`)	
vessel_name	made_up_sailings
Coastal Celebration	4
Coastal Renaissance	2

Expected Query Result (`ferries_12months`)	
vessel_name	made_up_sailings
Coastal Celebration	58
Coastal Inspiration	14
Coastal Renaissance	17
Queen of Alberni	6
Queen of Coquitlam	1
Queen of New Westminster	7
Skeena Queen	2
Spirit of British Columbia	7
Spirit of Vancouver Island	26

Question 3: VWSN Queries [10 marks]

Create queries for each of the data retrieval problems below, using the `vwsn_1year` database.

(a) Find the highest observed temperature in the dataset, along with the station number, station name and observation time of all cases where that temperature was reported.

Expected Query Result			
station_id	name	temperature	observation_time
165	Alberni Elementary School	44.4	2018-01-07 04:17:00

(b) For each station with station ID between 1 and 10 (inclusive), list the station ID, station name, maximum temperature observed at that station and observation time of **all** observations in the dataset in which the maximum temperature was attained at that station.

Expected Query Result			
station_id	name	max_temperature	observation_time
1	Ian Stewart Complex/Mt. Douglas High School	32.0	2017-08-02 17:32:00
1	Ian Stewart Complex/Mt. Douglas High School	32.0	2017-08-02 17:42:00
3	Strawberry Vale Elementary School	31.5	2017-09-03 16:12:00
4	Oaklands Elementary School	30.9	2017-06-25 15:21:00
5	Cedar Hill Middle School	32.1	2017-08-02 18:12:00
5	Cedar Hill Middle School	32.1	2017-08-02 18:27:00
5	Cedar Hill Middle School	32.1	2017-08-02 18:32:00
6	Marigold Elementary School/Spectrum High School	31.5	2017-06-25 14:31:00
6	Marigold Elementary School/Spectrum High School	31.5	2017-06-25 14:41:00
7	Campus View Elementary	31.3	2017-08-02 16:27:00
7	Campus View Elementary	31.3	2017-08-02 16:32:00
7	Campus View Elementary	31.3	2017-08-02 17:47:00
7	Campus View Elementary	31.3	2017-08-02 17:52:00
8	Victoria High School	31.7	2017-06-25 15:06:00
9	Frank Hobbs Elementary School	29.5	2017-08-02 18:07:00
10	Macaulay Elementary School	28.8	2017-08-02 19:12:00

(c) Find the IDs and names of all stations which have reported at least one observation at some point, but which did not report **any** observations in June, 2017.

Expected Query Result	
station_id	name
61	Cordova Bay Elementary School
68	Bayside Middle School
72	Race Rocks Ecological Reserve
100	District of Highlands Office
166	Maquinna Elementary School
181	West-Mont Montessori School
101	West Highlands District Firehall
184	NEPTUNE Port Alberni
220	Kyuoquot Elementary Secondary School

(d) In this question (and the next question), define the ‘daily average temperature’ for a particular day to be the average of all observations from all stations on that day. Furthermore, for each month, define the ‘10 hottest days’ to be the top 10 days (by daily average temperature) and define the ‘10 coolest days’ to be the bottom 10 days (by daily average temperature). For each month/year pair in the dataset, compute the average daily average temperature across the ten hottest days and the average daily average temperature across the ten coolest days. Notice that you are computing an average of averages (first, find the average daily temperatures of each of the ten hottest days, then average those temperatures to produce the result). Hint: Use the rank() function with an appropriate over() clause (maybe in multiple places)

Expected Query Result			
year	month	hottest10_average	coolest10_average
2017	2	6.08079083461	1.0197114637
2017	3	8.51687267652	3.67343895948
2017	4	10.382219865	7.76894703987
2017	5	16.5795991405	9.65679053523
2017	6	18.4126217218	13.224076244
2017	7	19.1035603049	16.8268137752
2017	8	21.1982192845	16.9197621431
2017	9	19.4114370311	12.7721461982
2017	10	11.3806084221	8.37786432667
2017	11	9.06097128498	4.11479124774
2017	12	5.68924368456	1.44416497515
2018	1	7.29715489128	3.76990281632

(e) List the day, month and year (in separate columns) of all days whose daily average temperature (see previous question) was lower than the daily average temperature of **any** of the previous 28 days in the dataset. The result should not list any of the first 28 days in the dataset, since the metric cannot be computed for those days, but the data for those days should be used to compute the rest of the result. Hint: You may want to use both the min() and count() aggregation functions in combination with an over() clause that includes both partitioning and windowing.

Expected Query Result		
`year`	`month`	`day`
2017	9	9
2017	9	10
2017	9	17
2017	9	18
2017	9	19
2017	10	5
2017	10	6
2017	10	7
2017	10	8
2017	10	9
2017	10	11
2017	10	12
2017	10	13
2017	10	14
2017	11	2
2017	11	3
2017	11	4
2017	12	19
2017	12	20
2017	12	21
2017	12	23

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 inadvertantly), 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. Three empty template files have been posted to conneX: `a3q1_queries.txt`, `a3q2_queries.txt` and `a3q3_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 three files called `a3q1_queries.txt`, `a3q2_queries.txt` and `a3q3_queries.txt`, using the provided templates as a starting point. Your answer for each query must consist of a single SQL statement (which may having 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.