

HW2: SQL

TOTAL POINTS: 1+1+1+1+2=6 [AND 1+1 BONUS POINTS!]

IN THIS ASSIGNMENT, YOU WILL CODE SOLUTIONS TO THE **FIVE** SQL PROBLEMS GIVEN BELOW.

ALL THE SQL KNOWLEDGE/COMMANDS YOU NEED TO ANSWER THE QUESTIONS HAVE BEEN COVERED IN CLASS! YOU **do NOT** NEED TO LEARN MORE COMMANDS OR TECHNIQUES (EG. USE OF 'TRIGGERS') ETC. ON YOUR OWN IN ORDER TO DO THIS HW SET.

TO RUN SQL CODE, YOU CAN USE ONE OF THE THREE WAYS MENTIONED IN THE LECTURE NOTES [A LOCALLY INSTALLED DB, OR A REMOTE SERVER-BASED DB VIA AN ONLINE SHELL RUNNING IN A BROWSER PAGE, OR A CLOUD DB] TO DO THE PROBLEMS.

ONE MORE CLOUD-Y WAY, NOT MENTIONED IN CLASS, TO DO YOUR HOMEWORK (AND CONTINUE LEARNING AND PRACTICING SQL) IS TO USE [HTTPS://LIVESQL.ORACLE.COM/](https://livesql.oracle.com/) ([HTTPS://LIVESQL.ORACLE.COM/](https://livesql.oracle.com/)); AFTER YOU SIGN UP FOR A FREE ACCOUNT, YOU CAN CREATE TABLES, INSERT ROWS AND DO QUERIES - YOU CAN SAVE ALL YOUR WORK IN SEPARATE SESSIONS, RELOAD [ANY/ALL OF] THEM AFTER LOGGING IN AGAIN IN ORDER TO RECREATE YOUR TABLES+DATA AND RERUN QUERIES, AND ALSO ADD MORE TABLES/DATA/CODE TO THE MIX - IT'S VERY CONVENIENT AND COOL, TRY IT!

WHAT YOU NEED TO SUBMIT ARE TEXT FILES WITH THE SQL COMMANDS THAT YOU COME UP WITH, ONE FILE FOR EACH QUESTION (Q1.SQL, Q2.SQL.. Q5.SQL). PLEASE MENTION AT THE TOP OF EACH FILE, WHICH DATABASE (EG. ORACLE, SQLITE..) YOU USED FOR THAT QUESTION! YOUR GRADER(S) WILL EXECUTE THE SQL COMMANDS FROM THE TEXT FILES YOU SUBMIT, USING THE SAME SOFTWARE YOU USED, TO SEE IF THEY PRODUCE THE EXPECTED RESULTS.

YOU CAN TALK TO YOUR FRIENDS/CLASSMATES TO INFORMALLY DISCUSS APPROACHES TO THE PROBLEMS, BUT DO NOT {'SHARE'/LOOK UP/ASK OTHERS FOR} ACTUAL CODE - THAT WOULD BE PLAGIARISM, AND WILL GET YOU AN 'F' IN THE COURSE.

PLEASE SEE YOUR TAs/GRADERS IF YOU NEED ONE-ON-ONE HELP (OR SEE ME). YOU CAN ALSO POST (AND REPLY!) IN THE PIAZZA 'HW2' FORUM. DON'T WAIT, START EARLY. GOOD LUCK, HAVE FUN!

Q1 (1 POINT). HERE IS A TABLE FOR RECORDING GUESTS' STAYS AT A HOTEL (ARRDATE DENOTES ARRIVAL DATE, DEPDATE IS DEPARTURE DATE):

```
CREATE TABLE HotelStays
(roomNum INTEGER NOT NULL,
arrDate DATE NOT NULL,
depDate DATE NOT NULL,
guestName CHAR(30) NOT NULL,
PRIMARY KEY (roomNum, arrDate));
```

THERE ARE TWO PROBLEMS (ISSUES) WITH THE ABOVE. FIRST, THE ARRIVAL DATE COULD BE INCORRECTLY ENTERED TO BE LATER THAN THE DEPARTURE DATE. SECOND, A NEW ENTRY (FOR A NEW GUEST) COULD BE ACCIDENTALLY PUT IN FOR A ROOM NUMBER, EVEN BEFORE THE EXISTING GUEST IN THAT ROOM HAS CHECKED OUT:

```
INSERT INTO HotelStays(roomNum, arrDate, depDate, guestName)
VALUES
(123, to_date('20160202', 'YYYYMMDD'), to_date('20160206', 'YYYYMMDD'),
(123, to_date('20160204', 'YYYYMMDD'), to_date('20160208', 'YYYYMMDD'),
(201, to_date('20160210', 'YYYYMMDD'), to_date('20160206', 'YYYYMMDD'),
;
```

NOTE THAT THE TO_DATE() FUNCTION IN THE ABOVE INSERT STATEMENT IS SPECIFIC TO POSTGRESQL (I USED V.9.3). SO IF YOU WANT TO USE THE ABOVE LINE AS-IS, YOU NEED TO RUN IT UNDER POSTGRES; ALTERNATELY, YOU CAN USE ANY OTHER RDB (ORACLE, MYSQL..) - JUST BE SURE TO USE THE APPROPRIATE DATE REPRESENTATION. IN OTHER WORDS, THE ABOVE STATEMENT IS THERE JUST AS AN EXAMPLE, YOU CAN MODIFY IT HOWEVER YOU WANT.

HOW WOULD YOU **REDESIGN THE TABLE** TO FIX BOTH THESE ISSUES? FOR YOUR ANSWER, YOU CAN EITHER PROVIDE A TEXTUAL EXPLANATION, AND/OR PROVIDE SQL STATEMENTS. HINT - "DO NOT BE CONCERNED WITH EFFICIENCY" - ANY WORKING SOLUTION IS ACCEPTABLE :)

Q2 (1 POINT). GIVEN THE FOLLOWING ENROLLMENT TABLE, **WRITE A QUERY** TO CREATE A LISTING THAT INCLUDES COURSE NAME AND THE NUMBER OF STUDENTS ENROLLED IN THE COURSE, SORTED IN REVERSE ORDER OF ENROLLMENT.

SID	ClassName	Grade
123	ARCH201	A
123	QUAN432	B
662	AMER352	B
662	ECON966	A
662	QUAN432	B
345	QUAN432	A
345	ECON966	D

GIVEN SOMETHING LIKE THE ABOVE, THE OUTPUT COULD BE:

ClassName	Total
QUAN432	3
ECON966	2
AMER352	1
ARCH201	1

Q3 (1 POINT). BELOW IS A SMALL TABLE THAT TRACKS WORK ORDERS FOR PROJECTS. WE HAVE A PROJECT ID COLUMN ON THE LEFT, A 'STEP' COLUMN IN THE MIDDLE (0,1,2.. DENOTE STEPS OF THE PROJECT), AND A STATUS COLUMN ON THE RIGHT (WHERE 'W' DENOTES 'WAITING', 'C' DENOTES 'COMPLETED'). SUCH A TABLE LETS PROJECT MANAGERS GET A QUICK STATUS ON THE VARIOUS ASPECTS (STEPS) OF THEIR PROJECTS ("WHERE THEY'RE AT", IN COLLOQUIAL, UNGRAMMATICAL LANGUAGE).

ProjectID	Step	Status
-----------	------	--------

P100	0	C
P100	1	W
P100	2	W
P201	0	C
P201	1	C
P333	0	W
P333	1	W
P333	2	W
P333	3	W

WRITE A QUERY TO OUTPUT THE PROJECT(S) WHERE ONLY STEP 0 HAS BEEN COMPLETED, IE. THE PROJECT GOTTEN STARTED BUT THE REST OF THE STEPS ARE IN WAITING MODE. IN THE ABOVE TABLE, SUCH A QUERY WOULD OUTPUT JUST 'P100'. YOU CAN ASSUME THAT STEPS GET COMPLETED IN ORDER, IE. P333 WILL NEVER HAVE C,W,C,W FOR EXAMPLE [ALL Cs WILL OCCUR BEFORE ALL Ws].

Q4 (1 POINT). BELOW IS A SMALL SAMPLE OF A JUNKMAIL DATABASE WE OWN, IE. PEOPLE WE WANT TO 'SPAM' VIA POSTAL MAIL [LOL].

Name	Address	ID	SameFam
'Alice'	'A'	10	NULL
'Bob'	'B'	15	NULL
'Carmen'	'C'	22	NULL
'Diego'	'A'	9	10
'Ella'	'B'	3	15
'Farkhad'	'D'	11	NULL

EACH ENTRY CONSISTS OF A NAME, ADDRESS, ID, AND WHETHER THERE IS A PRIOR FAMILY MEMBER ALREADY IN THE DB; IN THE LAST COLUMN, NULL MEANS THAT ENTRY IS THE FIRST FAMILY MEMBER IN OUR DB, AND A NON-NULL VALUE IS THE ID OF THE FIRST FAMILY MEMBER (EG. DIEGO POINTS TO ALICE, AND ELLA POINTS TO BOB).

WRITE A QUERY TO DELETE FROM THE TABLE, NAMES THAT HAVE 'NULL' FOR SAMEFAM **AND** ANOTHER FAMILY MEMBER IN THE DB. SO IN OUR EXAMPLE ABOVE, THE QUERY WOULD DELETE ALICE AND BOB, BUT NOT CARMEN AND FARKHAD. IN OTHER WORDS, WE WANT TO SAVE POSTAGE BY ONLY SENDING ONE MAIL PER HOUSEHOLD IF WE HAVE TWO PEOPLE FROM A HOUSE IN OUR DB (WE ASSUME THAT OUR DB CONTAINS EITHER ONE PERSON PER HOUSEHOLD, OR TWO).

Q5 (2 POINTS). WE ARE A DESSERT CATERING COMPANY. BELOW IS A TABLE OF CHEFS WHOSE SERVICES WE ACCESS ON AN AS-NEEDED 'GIG' BASIS, AND THE YUMMY DESSERTS THEY KNOW TO MAKE [FOR FUN, GOOGLE (IMAGE SEARCH) ONES YOU AREN'T FAMILIAR WITH!]:

Chef	Dish
'A'	'Apple pie'
'A'	'Beignet'
'A'	'Andes Chocolate Cake'
'B'	'Tiramisu'
'B'	'Creme brulee'
'B'	'Beignet'
'C'	'Tiramisu'
'C'	'Creme brulee'
'D'	'Apple pie'
'D'	'Tiramisu'
'D'	'Creme brulee'
'E'	'Apple pie'
'E'	'Bananas Foster'
'E'	'Creme brulee'
'E'	'Tiramisu'

WRITE A QUERY THAT WILL PICK OUT CHEFS WHO CAN MAKE EVERY ITEM IN THE TABLE BELOW (AN ORDER FROM SOMEONE WHO WANTS US TO CATER TO A WEEKEND PARTY; WE'D LIKE TO WORK WITH JUST THOSE CHEFS WHO CAN EACH MAKE AND DELIVER ALL ITEMS):

```
'Creme brulee'  
'Apple pie'  
'Tiramisu'
```

WITH THE ABOVE DATA, THE QUERY WOULD OUTPUT

Chef

'D'

'E'

YOU CAN HARDCODE THE DISHES JUST FOR SUBMISSION PURPOSES, BUT YOUR QUERY SHOULD WORK FOR ANY SUCH TABLE! USING COMMENTS IN THE CODE, **YOU NEED TO EXPLAIN IN YOUR OWN WORDS, WHAT THE QUERY DOES** (HOW IT WORKS); DON'T JUST SAY THINGS LIKE "NOW I'M USING A WHERE CONDITION", INSTEAD EXPLAIN WHAT IT'S FOR (WHY YOU'RE USING IT).

BONUS(1(+1) POINT(S)). YOU WILL GET 1 EXTRA POINT IF YOU CAN REFORMULATE THE QUERY IN A **VERY DIFFERENT WAY!** IF YOU DO THIS, SUBMIT A SEPARATE TEXT FILE WITH THE CODE, EG. Q5_V2.SQL. IF YOU COME UP WITH **YET ANOTHER VERY DIFFERENT WAY**, YOU CAN GET 1 MORE BONUS POINT (SUBMIT YET ANOTHER FILE, EG. Q5_V3.SQL). 'VERY DIFFERENT' MEANS JUST THAT - THE APPROACHES DO HAVE TO BE TOTALLY DISTINCT, EG. YOU CAN'T USE NOT TO INVERT AN EXISTING SOLUTION, OR USE IN() INSTEAD OF OR, ETC. SOOO... IS THIS ACTUALLY POSSIBLE [TO DO IT IN TWO, OR THREE, DIFFERENT WAYS]? YES!

TO REITERATE, EVERYTHING YOU NEED IS IN THE SLIDES (THE MATERIAL WE WENT THROUGH IN CLASS) - YOU DO NOT NEED TO READ AHEAD OR LOOK UP MORE COMMANDS ONLINE! LOOK AT EACH RELATIONAL OPERATOR, EACH COMMAND, EACH FUNCTION, EACH KEYWORD THAT WE COVERED, AND ASK YOURSELF HOW IT COULD BE OF USE IN CONSTRUCTING YOUR QUERY.

YOU'LL **LOSE POINTS** IF YOU:

- DON'T NAME YOUR FILES PROPERLY
- SUBMIT A .ZIP (YOU NEED TO SUBMIT INDIVIDUAL FILES INSTEAD)
- NEGLECT TO MENTION WHAT DB SOFTWARE YOU USED
- DON'T EXPLAIN PROPERLY, IN Q5, WHAT THE QUERY DOES
- SUBMIT/RESUBMIT AFTER THE DEADLINE (BE SURE TO DO IT AHEAD OF TIME, AND MAKE SURE YOU DID)

GOOD LUCK, **HAVE FUN** FIGURING IT ALL OUT!!
