# Greg Matthews

CS 500

February 11th, 2018

**HW 3: SQL**

**Part 1 (60 points):**

**(a).** For each job, list all candidates who have at least two of the necessary skills for that job. Result should have the schema (job, grade, candidate, cnt), where cnt is the number of skills a candidate has for a job.

**select job, grade, candidate, count(\*) as cnt**

**from Candidates\_Skills C\_S, Jobs\_Skills J\_S**

**where J\_S.skill = C\_S.skill**

**group by job, grade, candidate**

**having count(\*) > 1;**

**(b).** For each position, list all candidates whose salary requirements meet the salary budget. List candidates in order of required salary, from lower to higher. Result should have the schema(job, grade, company, candidate, candidate\_salary).

**select job, grade, company, candidate,**

**P.salary as salary, C.salary as candidate\_salary**

**from Candidates C, Positions P**

**where C.salary < P.salary**

**order by C.salary;**

**(c).** Are there job skills with no qualified candidates in our database? A candidate who has the skill but not the required number of years of experience is not qualified for the skill. The query should list all such skills, with the schema (skill, years).

**select distinct skill, years**

**from Jobs\_Skills J\_S**

**EXCEPT**

**select distinct J\_S.skill, J\_S.years**

**from Candidates\_Skills C\_S, Jobs\_Skills J\_S**

**where C\_S.years >= J\_S.years**

**and C\_S.skill = J\_S.skill;**

**(d).** For each skill (e.g., Java or DB), compute the maximum and the average number of years of experience among the candidates who have that skill, and the total number of candidates who have that skill in our database. Round the average number of years to 1 digit after the decimal point. Result should have the schema (skill, max\_years, avg\_years, cnt).

**select skill, MAX(years) as max\_years,**

**round(AVG(years),1) as avg\_years, count(\*) as cnt**

**from Candidates\_Skills C\_S**

**group by skill;**

**(e).** List pairs of candidates such that candidate 1 has more years of combined Java and DB experience but a lower salary requirement than candidate 2. The result should have the schema (candidate1, candidate2).

**select C\_1.candidate as candidate1, C\_2.candidate as candidate2**

**from Candidates as C\_1, Candidates as C\_2,**

**Candidates\_Skills as C\_11, Candidates\_Skills as C\_12,**

**Candidates\_Skills as C\_21, Candidates\_Skills as C\_22**

**where C\_1.candidate = C\_11.candidate**

**and C\_1.candidate = C\_12.candidate**

**and C\_2.candidate = C\_21.candidate**

**and C\_2.candidate = C\_22.candidate**

**and C\_11.skill = 'Java' and C\_12.skill = 'DB'**

**and C\_21.skill = 'Java' and C\_22.skill = 'DB'**

**and C\_11.year + C\_12.year > C\_21.year + C\_22.year**

**and C\_1.salary < C\_2.salary;**

**(f).** List pairs of candidates who both have at least 3 years of DB experience. Return each pair of candidates exactly once, i.e., do not return both ('Ann', 'Cathy') and ('Cathy', 'Ann'). Do not return a pair that contains the same candidate twice, e.g., do not return ('Ann', 'Ann'). Result should have the schema (candidate1, candidate2).

**select C\_1.candidate as candidate1, C\_2.candidate as candidate2**

**from Candidates\_Skills as C\_1, Candidates\_Skills as C\_2**

**where C\_1.candidate > C\_2.candidate**

**and C\_1.skill = 'DB' and C\_2.skill = 'DB'**

**and C\_1.years >= 3 and C\_2.years >= 3;**

**Part 2 (60 points):**

**(a).** For each country, compute the number of years in which one of its tennis players was ranked first. Result should have the schema (country, num\_years).

**select C.name as country, count(Y\_R\_F.year) as num\_years**

**from Tennis\_Players T\_P, Years\_Ranked\_First Y\_R\_F, Countries C**

**where T\_P.name = Y\_R\_F.name**

**and T\_P.country = C.name**

**group by C.name;**

**(b).** List pairs of tennis players (player1, player2) in which player1 both has a lower (better) ATP rank than player 2 and comes from a less populous country.

**select TP1.name as player1, TP2.name as player2**

**from Tennis\_Players TP1, Tennis\_Players TP2, Countries C1,Countries C2**

**where TP1.country = C1.name**

**and TP2.country = C2.name**

**and TP1.ATP\_rank < TP2.ATP\_rank**

**and C1.population > C2.population;**

**(c).** List pairs of players from the same country. List each pair exactly once. That is, you should list either (Djokovic, Raonic, Serbia) or (Raonic, Djokovic, Serbia), but not both. Result should have the schema (player1, player2, country).

**select TP1.name as player1, TP2.name as player2, C.name as country**

**from Tennis\_Players TP1, Tennis\_Players TP2, Countries C**

**where TP1.name > TP2.name**

**and TP1.country = C.name**

**and TP2.country = C.name;**

**(d).** For countries with at least 2 tennis players, list country name, GDP and average age of its tennis players. Result should have the schema (country, GDP, avg\_age).

**select C.name as country, C.gdp, AVG(TP.age) as avg\_age**

**from Countries C, Tennis\_Players TP**

**where TP.country = C.name**

**group by C.name, gdp**

**having count(\*) > 1;**

**(e).** List country name, GDP and population of each country. For countries that have tennis players in our database, also list the minimum age of its tennis players. Result should have the schema (country, GDP, population, min\_age).

**select C.name as country, C.gdp, population, MIN(TP.age) as min\_age**

**from Countries C, Tennis\_Players TP**

**where TP.country = C.name**

**group by C.name, C.gdp;**

**(f).** List names of countries who had a top-ranked tennis player both in 2010 or earlier (i.e., between 2004 and 2010, inclusive) and after 2010 (i.e., between 2011 and 2015, inclusive).

**select C.name**

**from Countries C, Years\_Ranked\_First YRF, Tennis\_Players TP**

**where C.name = TP.country**

**and TP.name = YRF.name**

**and YRF.year <= 2010 and YRF.year >= 2004**

**INTERSECT**

**select C.name**

**from Countries C, Years\_Ranked\_First YRF, Tennis\_Players TP**

**where C.name = TP.country**

**and TP.name = YRF.name**

**and YRF.year >= 2010 and YRF.year <=2015;**