**Assignment 2 (Problem 1).**

**1.**

1. πPname{Presidents}
2. πJname{σ(LawSchool=Yale OR LawSchool=Harvard)(Judges)}
3. Yale\_Pres <= πPname{[πJname{σ(LawSchool=Yale)(Judges)} \* Appoints]
Harv\_Pres <= πPname{[πJname{σ(LawSchool=Harvard)(Judges)} \* Appoints]
Multi <= ρn1(jnameY)(Yale\_Pres) ***(cross)*** ρn2(jnameH)(Harv\_Pres)
Result <= πJnameY{σ(jnameY=jnameH)(Multi)}

 Note: another way to do this is to use intersection.

1. num\_judges <= {law\_schoolЭcount(jname)(judges)}
2. pairs <= ρn1(jname1, lawschool1)(πJname,LawSchool{Judges}) **(cross*)*** ρn2(jname2, lawschool2)(πJname,LawSchool{Judges})
result <= πJname1,JName2{σ(lawschool1=lawschool2)(pairs)}

Note: Make sure you use pairs with conditions that judges names are different but law school is the same.

1. j\_not\_yale <= πJname{σ(LawSchool!=Yale)(Judges)}
party\_not\_yale <= πParty{(j\_not\_yale \* Appoints) \* Presidents}
all\_party <= πParty{Presidents}
party\_only\_yale <= all\_party – party\_not\_yale

Note: Key words here is only Yale. Therefore, you will need to subtract “not yale” from “all”

1. numjudge <= ρn1(pname, count)(pnameЭcount(jname)(appoints))
πpname{σ(count=2)(numjudge)}

Note: When you use group functions, you may need to rename the columns in order to use it again.
2. all\_pres <= πpname{presidents}
app\_pres <= πpname{appoints}
no\_appoints = all\_pres – app\_pres

Note: You cannot use group function here and test the count is zero. Because if the count is zero, it won’t even appear in the table.
3. judge\_count <= ρn1(pname, count)(pnameЭcount(jname)(appoints))
πpname{σ(count>2)(judge\_count)}
4. judges\_republican <= πjname, jdateofbirth, lawschool{[σ(party='republican')(presidents) \* appoints] \* judges}
judges\_not\_oldest <= πjudge1.jname{ρjudge1(judges\_republican) ***(join)*(judge1.jdateofbirth > judge2.jdateofbirth AND judge1.lawschool = judge2.lawschool)** ρjudge2(judges\_republican)}
judges\_oldest\_per\_lawschool = πjname{judges\_republican} – judges\_not\_oldest

Note: Another way to handle this is to use aggregate function when finding the oldest judge, i.e., the one with the smallest birth date.