# STSCI 5060 Homework 5

Pete Rigas (pbr43 cornell.edu)

November 27, 2020

---

## 1 Beginning Steps

### 1.1 beginning comments

```
/* Fall 2019 STSCI 5060 HW 5 */
/* Pete Rigas */
/* pbr43 */
/* 1 */
```

### 1.2 defining the libname for HW 4 from the SAS data sets on Canvas

```
libname HW5 = \\rschfs1x\usrcl\pbr43_STSCI5010\Desktop\HW5
```

Also, in all of the statements below,

- each proc sql statement ends with a quit;
- all of the data sets are correctly referred to from the HW5 libname statement above

## 2 Problem: Report of employee identification

In a similar way that we have used proc sql in several of the following problems, we must also use a proc sql statement, and a select statement, to get the employee identification of each current Level 3 and 4 sales staff hired in a particular year.

```
/* Give diagram output for the first question */
    title "Question 1";
    title2 "Output 1";
    proc sql;
        select Employee_ID
        from orion.Sales
        where year(Hire_date) = 2004 and scan(Job_title,-1)
        in ("III", "IV") intersect all
        select distinct Employee_ID
            from orion.Order_fact
            where year(Order_date) le 2005;
quit;
```

## 3 Problem: Levels of Mechanics table with 3 columns

Because some of the columns of the table that we would like to produce have 3 columns, we know that we must use the outer union corr keyword, from proc SQL, in addition to the asterisk operator with a select statement to get all of the observations for each of the 3 specified levels of mechanics. We are taking all of the data for the levels of each mechanics from the data stored in sasuser library. The resulting table that the code below outputs has 3 columns.

```
/* give diagram outputs for the second question */
    proc sql;
    select *
        from sasuser.mechanicsLevel1
    outer union corr
    select *
        from sasuser.mechanicsLevel2
    outer union corr
    select *
        from sasuser.mechanicsLevel3;
        quit;
```

## 4 Problem: Vertically displaying summarized data

To vertically display summarized data for the frequent flyers of some program, we will use a union and select statements, which after the proc sql statement will display the totals of all desired quantities.

```
/* give diagram outputs for the third question */
    title "Question 3";
title2 "Output 1";
proc sql;
select  sum(PointsEarned) as summary
from datasets.Frequentflyers
union
select  sum(PointsUsed) as summary
from datasets.Frequentflyers
union
select  sum(MilesTraveled) as summary
from datasets.Frequentflyers;
quit;
title2 "Output 2";
proc sql;
select sum(PointsEarned) as total_points_earned,
  sum(PointsUsed) as total_points_used,
  sum(MilesTraveled) as total_miles_traveled
from datasets.Frequentflyers;
quit;
```

In order to display the first 10 rows of the output, we will use a proc sql statement, and after creating a new diagram, create separate indices for the city, state, etc, with a corresponding footnote.

## 5 Problem: Simple index from Employee ID column

```
/* give diagram outputs for the fourth question */
    title "Problem 4";
title2 "output 1";
proc sql;
create unique index Employee_ID
on datasets.employee_addresses(Employee_ID);
quit;
proc sql;
create index EmpLocation
on datasets.employee_addresses(City, State, Country);
quit;
```

```
OPTIONS MSGLEVEL=I;
proc sql outobs=10;
footnote "INFO: Index Locale selected for WHERE clause optimization.
select* from datasets.employee_addresses
where City='Miami-Dade' and state='FL';

quit;
```

# 6 Problem: Business manager scenario

For the given business scenario, we can make use of the given business rules, in addition to using the Proc SQL view as given in the hint, for the following. First, we know that the code below captures the specifications of the business scenario in capturing all of the data that Tom needs, specifically the addresses, payroll, and organization of each employee.

```
/* give diagram outputs for the fifth question */
    proc sql;
    create view HW5.tom as select Employee_Name as Name format = $25.0,
    Job_Title as Title format = $15.0,
    Salary 'Annual Salary' format = comma10.2, int((today( ) - Employee_Hire_Date)/365) as YOS
    'Years of Service'
    from employee_addresses as a, employee_payroll as p,
    employee_orgnaization as o where a.Employee_ID = p.Employee_ID and
    o.Employee_ID = p.Employee_ID and Manager_ID = 120102$
```

From the code above, we are able to create a view, and then we calculate the number of years that the employee has worked for Tom by using the time() function in SAS, which gives us a time value that is currently stored in the SAS system without passing any parameters to the function. Next, we are able to store the addresses of each employee, and moreover, the ID of each employee and the ID of the manager, which is all that is required.