WORKING WITH SELECT DISTINCT

Nicolas Thomas

1. Introduction

Working with many different datasets in SAS with multiple variables and various observations can lead to a jumbled mess in the output.

In order to combat this, proc sql, and more specifically the “distinct” feature is a common technique utilized to make results appear in more user-friendly ways, along with having more clarity.

This paper goes into the basics of select distinct and proc sql, and how they can complete different objectives in SAS.

There are five different implementations of distinct with other different functions to return these simplified results.

In doing so, readers may have a better understanding of the power of distinct within proc sql, along with how they can implement it and other functions in their own data explorations.

2. What is Select Distinct?

Select distinct is an effective and efficient procedure commonly utilized in PROC SQL.

Proc sql is a procedure that completes both the DATA and PROC steps into one, using simpler statements and providing more functionality to users.

Combining select distinct with other SQL and general functions in SAS allows users to conduct unique tests to search and discover certain information within a dataset, relaying it back in a user-friendly way.

Within SQL Procedures, distinct is under the category of a “Select Statement” or Select clause, similar to the Into Clause, From Clause, Where Clause, Group By Clause, Having Clause, and Order By Clause.

Through all select statements, distinct is a very simple and effective way to receive information from a dataset quickly.

3. How can you utilize select distinct in different ways?

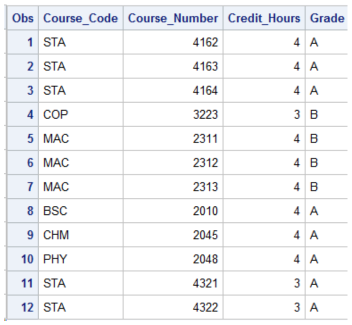
To first use distinct, you should implement the following steps.

1. Read in raw data from file or into SAS directly.
2. Start function with “proc sql” and read in with function “from” to connect to the dataset.
3. Above the from statement, write your “select distinct” line.

In sections 3.1-3.5, two datasets will be shown, where different utilizations of select distinct will be implemented to complete a variety of functions.

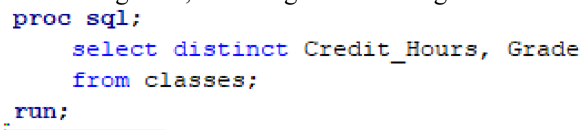
3.1 Finding Distinct Values

We will begin with creating a table of a student’s transcript, indicating the course code, course number, credit hours, and grade for each class.

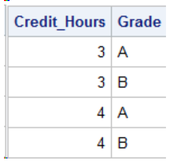


Obs	Course_Code	Course_Number	Credit_Hours	Grade
1	STA	4162	4	A
2	STA	4163	4	A
3	STA	4164	4	A
4	COP	3223	3	B
5	MAC	2311	4	B
6	MAC	2312	4	B
7	MAC	2313	4	B
8	BSC	2010	4	A
9	CHM	2045	4	A
10	PHY	2048	4	A
11	STA	4321	3	A
12	STA	4322	3	A

In this chart of classes taken, it can be seen that there are only 12 entries each with a course code, course number, hours in each, and a final grade. To find the different values of hours and grades, we can use the “distinct” function to sort by the distinct values of hours and grades, receiving the following dataset.



```
proc sql;
     select distinct Credit_Hours, Grade
     from classes;
run;
```



Credit_Hours	Grade
3	A
3	B
4	A
4	B

As we can now see, there are now four distinct inputs in the dataset for hours and grades, meaning that all courses on the student's transcript fall under one of the four criteria.

3.2 Alphabetically Sorting

Another operation that can be completed within select distinct is to sort any number of categorical variables by alphabetical, without having to add any more functions within the code. This will work to improve the clarity of the code, along with removing necessary lines.

Using the main dataset, we can look at the course codes to order in the user transcript alphabetically and sort out all other inputs.

```
proc sql;
    select distinct Course_Code
    from classes;
run;
```

Course_Code
BSC
CHM
COP
MAC
PHY
STA

Now in the output, we can see that the course codes in the transcript are organized alphabetically, with BSC (Biology) at the top, and STA (Statistics) at the bottom. Any repeats in course codes are also removed, so the results show that there are six different course codes that the student has taken.

We can also use the "order by" function to relay back data ordered by a certain variable specification alphabetically. To do this, we insert it after the table specification.

```
proc sql;
    select distinct Grade
    from classes
    order by Grade;
run;
```

Grade
A
B

By selecting the grades, we are able to see the different grades in the dataset ordered alphabetically.

We could also relay back the entire table with the "select distinct" and "order by" function, receiving every datapoint ordered by grade.

```
proc sql;
    select distinct *
    from classes
    order by Grade;
run;
```

Course_Code	Course_Number	Credit_Hours	Grade
BSC	2010	4	A
CHM	2045	4	A
PHY	2048	4	A
STA	4162	4	A
STA	4163	4	A
STA	4164	4	A
STA	4321	3	A
STA	4322	3	A
MAC	2311	4	B
MAC	2312	4	B
MAC	2313	4	B
COP	3223	3	B

Furthermore, if we wanted to see the course codes and their potential corresponding grades, we can utilize select distinct.

```
proc sql;
    select distinct Course_Code, Grade
    from classes
    order by Grade;
run;
```

The output is shown below for the table with course codes and course grades.

Course_Code	Grade
BSC	A
CHM	A
PHY	A
STA	A
COP	B
MAC	B

3.3 Finding variable specifications

Further using select distinct with other functions, we can combine it with the "where" function to determine which course codes correlate to a class with over four credit hours on a student's transcript.

```
data test;
     set classes;
     where Credit_Hours > 3;
run;
```

The resulting dataset will now show all courses that have credit hours greater than 3.

Course_Code	number	hours	grade
BSC	2010	4	A
CHM	2045	4	A
MAC	2311	4	B
MAC	2312	4	B
MAC	2313	4	B
PHY	2048	4	A
STA	4162	4	A
STA	4163	4	A
STA	4164	4	A

Using DISTINCT, we can choose to only see the distinct course codes, finding the codes that have over three credit hours and returning once for each, even if duplicates exist.

```
proc sql;
     select distinct Course_Code
     from test;
run;
```

Course_Code
BSC
CHM
MAC
PHY
STA

As we can see from the resulting table, there are only five course codes that have a credit hours greater than 3.

Combining points from 3.2 and 3.3, we can find the distinct values of course codes that have resulting credit hours above 3, ordered by grade alphabetically.

To do this, we can include the same step of creating data called "test" and finding where the hours is greater than 3.

After that, we can use select distinct to find the distinct course codes and grades and order the resulting table by grades.

```
proc sql;
     select distinct Course_Code, Grade
     from test
     order by Grade;
run;
```

In the resulting table, we can see that there are only five course codes with credit hours over three, sorted alphabetically by grades.

Course_Code	Grade
BSC	A
CHM	A
PHY	A
STA	A
MAC	B

3.4 Sorting values

In another scenario, looking at a student's record of calculus classes, their transcript shows two attempts on a course, but we only want to see the completed courses that count towards the student's graduation.

Obs	Course_Code	Course_Number	Credit_Hours	Grade
1	MAC	2311	4	B
2	MAC	2312	4	F
3	MAC	2312	4	B
4	MAC	2313	4	B

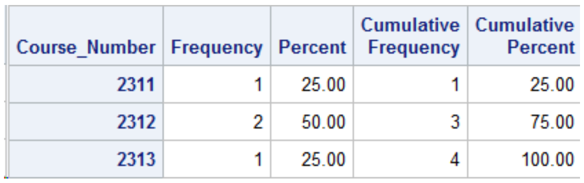
The sample dataset is small, but in a larger dataset, we can use distinct to sort by grade to find if there are any failed classes on a student's record.

```
proc sql;
     select distinct Grade
     from calcclasses;
run;
```

Grade
B
F

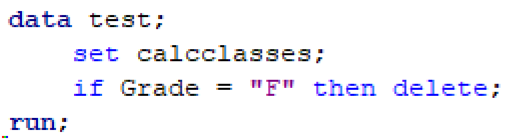
As we can see, there are values for grades indicating that a class was failed. We can utilize "proc freq" to find the number of times a class shows up. If it appears more than one time, we know that it is a repeated class.

```
proc freq data=calcclasses;
     TABLE Course_Number / MISSING;
RUN;
```

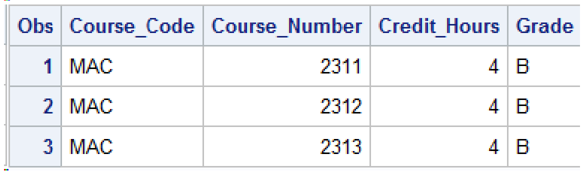


Course_Number	Frequency	Percent	Cumulative Frequency	Cumulative Percent
2311	1	25.00	1	25.00
2312	2	50.00	3	75.00
2313	1	25.00	4	100.00

Now that we can see the course “2312” was failed, we can remove rows that have the failed class, simplifying our dataset and removing unnecessary information.

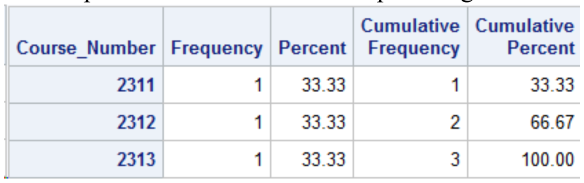


```
data test;
     set calcclasses;
     if Grade = "F" then delete;
run;
```



Obs	Course_Code	Course_Number	Credit_Hours	Grade
1	MAC	2311	4	B
2	MAC	2312	4	B
3	MAC	2313	4	B

After removing the unnessesary class, we can clearly see the passes courses and their respective grades.



Course_Number	Frequency	Percent	Cumulative Frequency	Cumulative Percent
2311	1	33.33	1	33.33
2312	1	33.33	2	66.67
2313	1	33.33	3	100.00

We can also use the proc freq command again to verify our data and make sure that the extra course number of “2312” was removed.

3.5 Working in the log

To visualize different values in the dataset without directly printing output each time, you can also work in the log of SAS to find different values and variables.

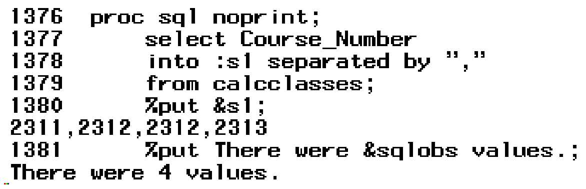
By using the “into” clause with distinct, we can have distinct values show up in the log rather than clog up the output viewer tab. We once again are going to find any failed classes by looking at the repeat course numbers and grades.

1. Using the condensed calc classes table, we can choose to select distinct inputs in the Course Number column to begin.

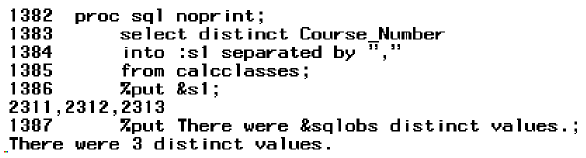
2. Then, we can choose to put the distinct values in a line of output, with a comma delimiter.

3. Finally, we can use a %put function to call back the results into the log.

In the code, we can use the “select number” line to find the amount of course numbers, and then change this in our main function to “select distinct number” to find the number of distinct values.



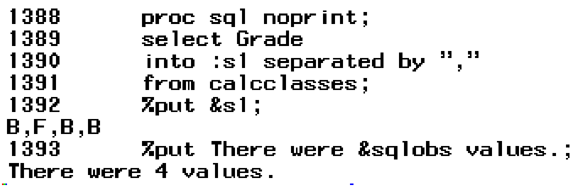
```
1376  proc sql noprint;
1377        select Course_Number
1378        into :s1 separated by ","
1379        from calcclasses;
1380        %put &s1;
2311,2312,2312,2313
1381        %put There were &sqlobs values.;
There were 4 values.
```



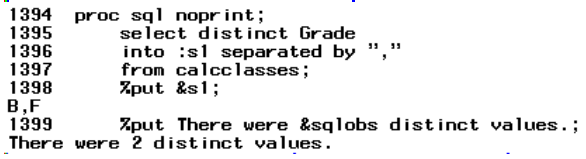
```
1382  proc sql noprint;
1383        select distinct Course_Number
1384        into :s1 separated by ","
1385        from calcclasses;
1386        %put &s1;
2311,2312,2313
1387        %put There were &sqlobs distinct values.;
There were 3 distinct values.
```

Moreover, we can see that there are four values for course numbers, but only three distinct values, indicating that a course was failed by the student at some point.

We can further this by looking at the grades by doing a similar function of looking at the different grades, and the distinct values.



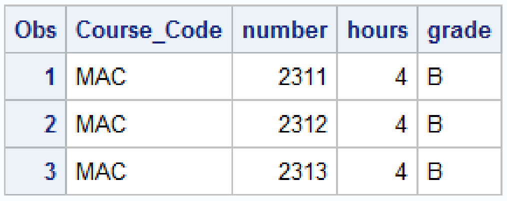
```
1388        proc sql noprint;
1389        select Grade
1390        into :s1 separated by ","
1391        from calcclasses;
1392        %put &s1;
B,F,B,B
1393        %put There were &sqlobs values.;
There were 4 values.
```



```
1394  proc sql noprint;
1395        select distinct Grade
1396        into :s1 separated by ","
1397        from calcclasses;
1398        %put &s1;
B,F
1399        %put There were &sqlobs distinct values.;
There were 2 distinct values.
```

As we can see, there does seem to be a value for “F”, indicating a failing grade in a course.

Now that it has been verified that a failing grade occurs, we can utilize the “delete” function once more to delete an results with an “F” indicating a failure.



Obs	Course_Code	number	hours	grade
1	MAC	2311	4	B
2	MAC	2312	4	B
3	MAC	2313	4	B

As we can see, the three distinct values for course numbers are all present, and all grades are passing in the table.

Using this method, we can avoid crowing up the output and instead find points in our data quicker by working in the log.

CONCLUSION

This paper focused on explaining what select distinct and proc sql are, and then delved into different examples using this and other functions to create data representations within SAS. Some of these involved finding values and sorting alphabetically or looking at the log to find results faster.

Hopefully, this paper makes it easier to understand and implement select distinct and other proc sql functions on data within SAS.

REFERENCES

Bhalla, Deepanshu. "SAS : Count Distinct Values of Variables." ListenData, www.listendata.com/2016/04/sas-count-distinct-values-of-variables.html. Accessed 1 Dec. 2023.

Lafler, K. P. (2007). "A Hands-On Tour Inside the World of PROC SQL," SAS Support https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjUyuOP0MKCAxU5pIkEHTiJBl0QFnoECBsQAQ&url=https%3A%2F%2Fsupport.sas.com%2Fresources%2Fpapers%2Fproceedings%2Fproceedings%2Fforum2007%2F109-2007.pdf&usg=AOvVaw2g3gQE6iw-UjSvLkhiPHa5&opi=89978449

SAS® Viya™ SQL Procedure User's Guide. Available at https://documentation.sas.com/doc/en/sqlproc/1.0/p12ohgh32ffm6un13s7l2d5p9c8y.htm

SAS® Viya™ SQL Procedure User's Guide. Available at https://documentation.sas.com/doc/en/sqlproc/1.0/p0hwg3z33gllron184mzdoqwpe3j.htm

SAS® Viya™ SQL Procedure User's Guide. Available at https://documentation.sas.com/doc/en/sqlproc/1.0/n082a03omu3i21n1k889zfklh4ps.htm

Sen, B. (2020). "Creating Data Listings Utilizing PROC Procedures," SEGUI https://sesug.org/proceedings/sesug_2020_final_papers/Reporting_and_Visualization/SESUG2020_Paper_163_Final_PDF.pdf

Tran, HoaiNam and Ngo, Mai Anh (2019), "The Power of PROC SQL's SELECT DISTINCT INTO," https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjV9tK7v8KCAxUrkIkEHeMRA1sQFnoECBkQAQ&url=https%3A%2F%2Fwww.lexjansen.com%2Fsesug%2F2019%2FSESUG2019_Paper-177_Final_PDF.pdf&usg=AOvVaw0XdouzH1GL-srcT_0eGEle&opi=89978449

TRADEMARKS