Chapter 2

Performing Advanced Queries Using PROC SQL

Displaying All Columns

- To select all columns included in a table use one of two options
  - List all variables from the table in the select clause
    - The order of the columns will be based on the order the columns appear in the select clause
  - Use select * in the select clause
    - The order of the columns will be based on the order in which they are stored in the table

Example – Displaying all Fields

```
Proc sql;
 select player, atbats,
 hits, bb
 from bbstats;
```

```
Proc sql;
 select *
from bbstats;
```

* Both sets of code return the same result from the bbstats dataset.

Example – Displaying all Fields Resulting Dataset

Player	At Bats	Hits	BB
Christian Walker	271	97	36
Scott Wingo	240	81	44
Brady Thomas	231	73	23
Evan Marzilli	220	64	25
Robert Beary	211	61	12
Adrian Morales	249	70	30
Peter Mooney	254	71	44
Jake Williams	209	56	21
Jackie Bradley Jr.	162	40	22

FEEDBACK OPTION

- Use when select * is included in the select clause to see the list of columns
  - The list of columns will be written to the SAS log

OUTOBS= Option

- Use to limit the number of rows displayed
- Similar to the obs= data set option
- OUTOBS does not limit the number of rows that are read. To restrict the number of rows read use the INOBS= option

Example – OUTOBS=Option

```
Proc sql outobs=5;
 select player, atbats
 from bbstats;
quit;
```

player	atbats
Christian Walker	271
Scott Wingo	240
Brady Thomas	231
Evan Marzilli	220
Robert Beary	211

Removing Rows That Contain Duplicate Values

- Use the keyword DISTINCT in the select statement to eliminate rows with the same values

Example – Eliminating Rows that Contain Duplicate Values

Player	At Bats	Hits	BB
Christian Walker	271	97	36
Scott Wingo	240	81	44
Brady Thomas	231	73	23
Evan Marzilli	220	64	25
Robert Beary	211	61	12
Adrian Morales	249	70	30
Peter Mooney	254	71	44
Jake Williams	209	56	21
Jackie Bradley Jr.	162	40	22
Scott Wingo	240	81	44

* In the table above, Scott Wingo appears twice.

Example – Eliminating Rows that Contain Duplicate Values

```
proc sql;
  select distinct player, atbats, hits, bb
  from bbstats;
quit;
```

Example – Eliminating Rows that Contain Duplicate Values

Player	At Bats	Hits	BB
Christian Walker	271	97	36
Scott Wingo	240	81	44
Brady Thomas	231	73	23
Evan Marzilli	220	64	25
Robert Beary	211	61	12
Adrian Morales	249	70	30
Peter Mooney	254	71	44
Jake Williams	209	56	21
Jackie Bradley Jr.	162	40	22

Conditional Operators

- Between-and
- Contains or ?
- In
- Is missing or is null
- Like
- Any
- All
- Exists

Between-and Operator

- Used to extract rows based on a range of numeric or character values
- Used in the where clause

Example – Between-and Operator

```
Proc sql;
 select player, atbats
 from bbstats
 where atbats
 between 162 and 215;
quit;
```

player	atbats
Robert Beary	211
Jake Williams	209
Jackie Bradley Jr.	162

Example – not Between-and Operator

```
Proc sql;
 select player, atbats
 from bbstats
 where atbats
 not between
 162 and 215;
quit;
```

player	atbats
Christian Walker	271
Scott Wingo	240
Brady Thomas	231
Evan Marzilli	220
Adrian Morales	249
Peter Mooney	254

Contains or Question Mark (?) Operator to Select a String

- Usually used to select rows based on a particular string in a character column.
- Matching is case sensitive when making comparisons
  - Use the UPCASE function if comparison is based on all capital letters and there is a mix of upper and lower case letters

Example – Contains or Questions Mark (?) Operator

```
Proc sql;
 select player, atbats
 from bbstats
 where upcase(name)
   contains 'IA';
quit;
```

player	atbats
Christian Walker	271
Adrian Morales	249
Jake Williams	209

IN Operator to Select Values from a List

- Use to select rows that match values in a list
- List can include numeric or character values

Example – IN Operator

```
Proc sql;
 select player, atbats
 from bbstats
 where name in
   ('Christian Walker',
   'Jake Williams');
quit;
```

player	atbats
Christian Walker	271
Jake Williams	209

Is Missing or Is NULL Operator

- Use to select rows that contain missing values, character and numeric
- The IS MISSING and IS NULL operators are interchangeable

Like Operator

- Use to select rows that contain a specific pattern of characters
- Special characters
  - Use underscore (_) to represent a single character
  - Use the percent sign (%) to represent any sequence of characters

Example – Like Operator

```
Proc sql;
 select player, atbats
 from bbstats
 where player like
 'Ja%';
quit;
```

Player	atbats
Jake Williams	209
Jackie Bradley Jr.	162

Sounds-Like (=*) Operator

- Use to select rows that contain a value that sounds like another value.

Subsetting Rows by Calculated Values

- Use the keyword CALCULATED in the where clause to subset the data based on a value that is calculated within the query

Example – Using a Calculated Value in the Where Clause

```
Proc sql;
 select player,
   hits/atbats as avg
 from bbstats
 where calculated
 avg > .300;
quit;
```

player	avg
Christian Walker	.358
Scott Wingo	.338
Brady Thomas	.316

Column Labels, Column Formats Titles and Footnotes

- Use the label= option to specify the label to display for the column
- Use the format= option to specify the format to display data in the column
- Title and footnote statements must be placed in one of the following locations
  - Before the PROC SQL statement
  - Between the PROC SQL statement and the select statement

Example – Label, Format, and Title

```
Proc sql;
title 'Averages for 2011
   USC Gamecocks';
 select player
   label='Player Name',
   hits/atbats as avg
   label='Average'
   format=4.3
 from bbstats;
quit;
```

Averages for 2011 USC Gamecocks

Player Name	Average
Christian Walker	.358
Scott Wingo	.338
Brady Thomas	.316
Evan Marzilli	.291
Robert Beary	.289
Adrian Morales	.281
Peter Mooney	.280
Jake Williams	.268
Jackie Bradley Jr.	.247

Adding a Character Constant to Output

- To define a new column that contains a character string, include a text string in quotation marks in the SELECT clause.

Example – Adding a Character Constant to Output

```
Proc sql;
 select player,
 'average is:',
 hits/atbats as avg
 from bbstats;
quit;
```

player		avg
Christian Walker	average is:	.358
Scott Wingo	average is:	.338
Brady Thomas	average is:	.316
Evan Marzilli	average is:	.291
Robert Beary	average is:	.289
Adrian Morales	average is:	.281
Peter Mooney	average is:	.280
Jake Williams	average is:	.268
Jackie Bradley Jr.	average is:	.247

Summarizing and Grouping Data

- A summary function can be used in PROC SQL to produce a statistical summary of data in a table.
  - Examples of summary functions
    - avg – average of values
    - count – number of nonmissing values
    - min – smallest value
    - std – standard deviation
    - sum – sum of values

Summarizing and Grouping Data

- If a GROUP BY clause is not present in the query, PROC SQL will apply the function to the entire table.
- If a GROUP BY clause is present in the query, PROC SQL will apply the function to each group specified in the GROUP BY clause.

Example – Summarizing and Grouping Data

Player	Position	At Bats	Hits	BB
Christian Walker	Infield	271	97	36
Scott Wingo	Infield	240	81	44
Brady Thomas	Infield	231	73	23
Evan Marzilli	Outfield	220	64	25
Robert Beary	Infield	211	61	12
Adrian Morales	Infield	249	70	30
Peter Mooney	Infield	254	71	44
Jake Williams	Outfield	209	56	21
Jackie Bradley Jr.	Outfield	162	40	22

Example – Summarizing Data

```
Proc sql;
 select sum(atbats) as
 totalatbats,
 sum(hits) as totalhits
 from bbstats;
quit;
```

totalatbats	totalhits
2,047	613

Example – Summarizing and Grouping Data

```
Proc sql;
 select position,
 sum(atbats) as
 totalatbats,
 sum(hits) as totalhits
 from bbstats
group by position;
quit;
```

position	totalatbats	totalhits
Infield	1,456	453
Outfield	591	160

Counting Values Using the Count Function

- count (*) – counts the total number of rows in a group or in a table
- count (column) – counts the total number of rows in a group or in a table for which there is a nonmissing value in the selected column
- count (distinct column) – counts the total number of unique values in a column

Example – Counting Values

```
Proc sql;
 select count (*) as count
 from bbstats;
```

count
9

Example – Counting Values

```
Proc sql;
 select count (position)
   as  count
 from bbstats;
quit;
```

count
9

* Because there is no missing data, you get the same output with this query as you would by using count (*).

Example – Counting Values

```
Proc sql;
 select count (distinct
   position) as  count
 from bbstats;
```

count
2

Having Clause

- The HAVING clause follows the GROUP BY clause
- Works with the GROUP BY clause to restrict groups that are displayed in the output, based on one or more conditions
- You do not have to include the keyword CALCULATED in a HAVING clause; you do have to include in it a WHERE clause.

Example – Having Clause

```
Proc sql;
 select position,
 sum(atbats) as
 totalatbats,
 sum(hits) as totalhits
 from bbstats
group by position
having totalhits > 160;
quit;
```

position	totalatbats	totalhits
Infield	1,456	453

Subqueries

- A subquery is a query that is nested in, and is part of, another query.
- Types of subqueries
  - Noncorrelated – a self-contained subquery that executes independently of the outer query
  - Correlated – a dependent subquery that requires one or more values to be passed to it by the outer query before the subquery can return a value to the outer query.

Example – Noncorrelated Subquery

```
Proc sql;
 select position,
 sum(atbats) as
 totalatbats,
 sum(hits) as totalhits
 from bbstats
group by position
having totalhits >
   (select sum(hits)
   from bbstats where
    position='Outfield');
 quit;
```

position	totalatbats	totalhits
Infield	1,456	453

Example – Correlated Subquery

AtBats

Player	Atbats
Walker	271
Wingo	240
Thomas	231
Marzilli	220
Beary	211
Morales	249
Mooney	254
Williams	209
Bradley Jr.	162

Playerposition

Player	Position
Walker	Infield
Wingo	Infield
Thomas	Infield
Marzilli	Outfield
Beary	Infield
Morales	Infield
Mooney	Infield
Williams	Outfield
Bradley Jr.	Outfield

Example – Correlated Subquery

```
Proc sql;
 select player, atbats
 from atbats
 where 'Infield'=
    (select position
      from playerposition
      where atbats.player=
                playerposition.player);
```

Example Correlated Subquery

- Step 1 – The outer query takes the first row in atbats table and finds the columns player and atbats.
- Step 2 – Match atbats.player (passed from table in outer query) with playerposition.player to find the qualifying row in the playerposition table.
- Step 3 – The inner query now passes the position of the selected row in playerposition back to the outer query via the = operator, where the position is matched for the selection in the outer query.

Example – Correlated Subquery

Player	Atbats
Walker	271
Wingo	240
Thomas	231
Beary	211
Morales	249
Mooney	254

Validating Query Syntax

- To verify the syntax and existence of columns and tables referenced in your query without executing the query use the NOEXEC option or the VALIDATE keyword
- Use the NOEXEC option in the PROC SQL statement
- Use the VALIDATE keyword before a SELECT statement

Example – NOEXEC Option

```
proc sql noexec;
 select position, atbats, hits
from bbstats;
quit;
```

If the query is valid and all columns and tables exist, the SAS log will have the following message.

NOTE: Statement not executed due to NOEXEC option.

Example – VALIDATE Keyword

```
proc sql;
 validate
 select position, atbats, hits
from bbstats;
quit;
```

If the query is valid, the SAS log will have the following message.

NOTE: PROC SQL statement has valid syntax.