PROC SQL Fundamentals

1. Generating Simple Reports

2. Summarizing and Grouping Data

3. Creating and Managing Tables

Copyright © SAS Institute Inc. All rights reserved.

Creating Simple Reports

Customer

- Top 10 customers by income
- No BankID
- CreditScore greater than 700

- Born prior to December 31, 1940
- Employed

Copyright © SAS Institute Inc. All rights reserved.

Recall: Filter the Columns Using the SELECT Statement

PROC SQL *<options>*;
    SELECT *col-name, col-name*
    FROM *input-table*;
QUIT;

Customer ID	Customer Country	Customer Type Name
544	TR	Orion Club members high activity
908	TR	Orion Club Gold members high activity
928	TR	Orion Club members low activity
1033	TR	Orion Club members low activity
1100	TR	Orion Club members low activity
1684	TR	Orion Club members low activity
2788	TR	Orion Club members high activity

```
proc sql;
select Customer_ID, Customer_Country, Customer_Type
    from orion.customers;
quit;
```


Copyright © SAS Institute Inc. All rights reserved.

Filtering Rows Using the WHERE Clause

PROC SQL *<options>*;
    SELECT *col-name, col-name*
    FROM *input-table*
    WHERE *expression*;
QUIT;

```
proc sql;
select Customer_ID, Customer_Country, Customer_Type
    from orion.customers
    where Customer_Country='TR';
quit;
```

Customer ID	Customer Country	Customer Type Name
544	TR	Orion Club members high activity
908	TR	Orion Club Gold members high activity
928	TR	Orion Club members low activity
1033	TR	Orion Club members low activity
1100	TR	Orion Club members low activity
1684	TR	Orion Club members low activity
2788	TR	Orion Club members high activity

Copyright © SAS Institute Inc. All rights reserved.

Character and Numeric Values

WHERE *expression*;

ABC

Character

case sensitive

```
where Customer_Country = 'TR'
```

```
where Customer_Country = "TR"
```

123

Numeric

Standard

digits 0 – 9	YES
minus sign	YES
decimal point	YES
dollar sign	NO
comma	NO

```
where income > 30000
```

```
where month(Customer_BirthDate) = 9
```


Copyright © SAS Institute Inc. All rights reserved.

WHERE Comparison Operators

Mnemonic	Symbol
LT	<
GT	>
EQ	=
LE	<=
GE	>=
NE	< >
	¬= (EBCDIC) ^= (ASCII)

Copyright © SAS Institute Inc. All rights reserved.

Date, Time, and Datetime Values

SAS date

```
where DOB < "01JAN2000"d
```

14610

number of days since ***January 1, 1960***

SAS time

```
where Time >= "14:45:32"t
```

53132

number of seconds since ***midnight***

SAS datetime

```
where DateTime <= "01JAN2000 12:00:00"dt
```

1262347200

number of seconds since ***midnight January 1, 1960***

Copyright © SAS Institute Inc. All rights reserved.

Combining Expressions

WHERE *expression-1* **OR | AND** *expression-n*;

OR

```
proc sql;
select CustomerID, DOB
    from sq.customer
    where State = 'NY' or
          State = 'NC' or
          State = 'CA';
quit;
```

If ***any*** expression is true, select the row.

AND

```
proc sql;
select CustomerID, DOB
    from sq.customer
    where Income > 30000 and
          State = 'NC';
quit;
```

If ***both*** expressions are true, select the row.

Copyright © SAS Institute Inc. All rights reserved.

IN Operator

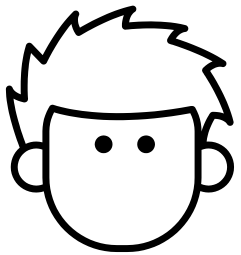
WHERE *col-name* **IN (***value-1,…,value-n***);**
WHERE *col-name* **NOT IN (***value-1,…,value-n***);**

```
where State in ("NC","GA","NY")
```

State="*NC*" *or* **State**="*GA*" *or* **State**="*NY*";

```
where State not in ('NC' 'GA' 'NY')
```

The **NOT** operator forms a negative condition.



Copyright © SAS Institute Inc. All rights reserved.

2.01 Activity

Open SAS EG, create a new project, locate the **Customers** dataset (Either assign the ORION library or copy the dataset into the WORK folder using the Copy Files Task.) Next, perform the following tasks:

1. Write a query to find all customers resides in Turkey.
   - SELECT statement with a WHERE clause, country code is TR
2. Add another expression using the OR operator to select only customers from Turkey (TR) ***or*** Germany (DE). How many customers are from either TR or DE?
3. Switch your current expression to use the IN operator. Which customers are from either US, CA or AU?

Copyright © SAS Institute Inc. All rights reserved.

2.01 Activity – Correct Answer

1. Write a query to find all customers resides in Turkey.

```
where Customer_Country = "TR"
```

2. Add another expression using the OR operator to select only customers from TR or DE? How many customers are from either country?

```
where Customer_Country = "TR" or Customer_Country = "DE"
```

3. Which customers are from either US, CA or AU?

```
where Customer_Country in ("US", "CA", "AU")
```

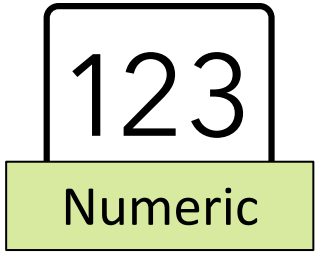

Copyright © SAS Institute Inc. All rights reserved.

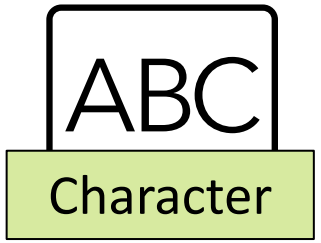
Special WHERE Operators: Missing Values

ANSI

WHERE *col-name* **IS NULL;**
WHERE *col-name* **IS NOT NULL;**

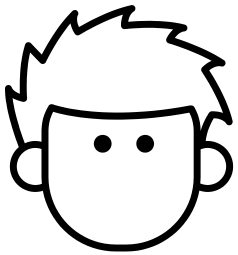
The **IS NULL** operator works for both ***character*** and ***numeric*** missing values.






```
where Income = .
```

```
where Married = " "
```



Copyright © SAS Institute Inc. All rights reserved.

2.02 Activity

Download **pva97nk.sas7bdat** from Blackboard. Perform the following tasks to find all donors who are younger than 30.

```
proc sql number;
select ID, DemAge
      from ORION.PVA97NK
      where /*complete the where statement*/;
 quit;
```

1. Write a WHERE clause to find all donors with a **DemAge** value that is less than 30 and run the query. What do you notice about the values in the **DemAge** column? How many rows are in your report?
2. Include the AND operator in the WHERE clause to find all rows that are less than 30 and not null. Use a method of your choice. How many rows are in your final report?

Copyright © SAS Institute Inc. All rights reserved.

continued...

2.02 Activity – Correct Answer

1. What do you notice about the values in the **DemAge** column? How many rows are in your report? **Missing values are included in the results, for a total of 2,691 rows.**

```
where DemAge < 30
```

Row	Control Number	Age
1	00000142	.
2	00000186	.
3	00000383	.
4	00000387	.
5	00000427	.
6	00000542	.
7	00000557	.
8	00000578	.
9	00000701	.
10	00000804	7
11	00000826	17
12	00000903	.
13	00001034	.

SAS treats missing values as ***smaller than*** nonmissing values.

2.02 Activity – Correct Answer

2. How many rows are in your report? **Missing values are *not* included in the results. The new report contains 284 rows.**


```
where DemAge < 30 and DemAge is not null
```

```
where DemAge < 30 and DemAge is not missing
```

```
where DemAge < 30 and DemAge ne .
```

equivalent statements

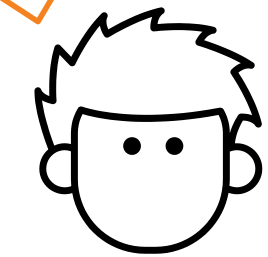
Copyright © SAS Institute Inc. All rights reserved.

Special WHERE Operators: Range of Values

WHERE *col-name* **BETWEEN** *value-1* **AND** *value-2*;

WHERE *col-name* **NOT BETWEEN** *value-1* **AND** *value-2*;

The BETWEEN-AND operator can be used on ***character*** values.



```
where CreditScore between 700 and 799
```

```
where CreditScore >= 700 and CreditScore <= 799
```

Selects rows with values ***between and including*** the endpoints that you specify

Copyright © SAS Institute Inc. All rights reserved.

Special WHERE Operators: Pattern Matching

WHERE *col-name* **LIKE** "*value*";

```
where FirstName like "Z%";
```

Zachary
Zelda
Zulma
Zula
Zoe
Zandra

wildcard for ***any number*** of characters

```
where FirstName like "Z_l%";
```

Zelda
Zulma
Zelma
Zola

wildcard for a ***single*** character

Copyright © SAS Institute Inc. All rights reserved.

2.03 Activity

Try wildcards such as % or _ in any query.

Copyright © SAS Institute Inc. All rights reserved.

2.03 Activity – Some Answers

Try wildcards such as % or _ in any query.

```
proc sql;
select Customer_ID, Customer_FirstName, Customer_LastName
       from orion.customers
       where Customer_FirstName like "%m_s";
quit;

/*Alternatively, you could try the following:*/

proc sql;
select Customer_ID, Customer_FirstName, Customer_LastName
       from orion.customers
       where Customer_FirstName like "J_m%";
quit;
```

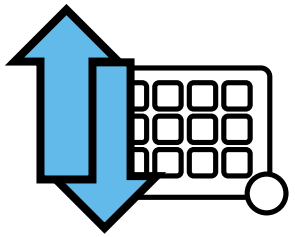

Copyright © SAS Institute Inc. All rights reserved.

Sorting the Output with the ORDER BY Clause

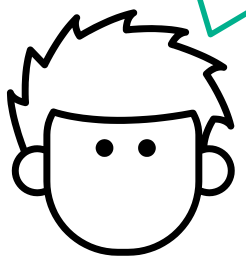
```
PROC SQL <options>;
SELECT col-name, col-name
      FROM input-table
      WHERE expression
      ORDER BY col-name <DESC> <, col-name>;
QUIT;
```

Sort Methods

- name or alias
- multiple columns
- calculated column
- integer position



The **ORDER BY** clause defines the order in which rows are displayed in the results.



Copyright © SAS Institute Inc. All rights reserved.

2.04 Activity

In Employee_Addresses table, filter the San Diego employees in the 920 postal code area. SELECT Employee_ID, Employee_Name, Street_Number, Street_Name, Postal_Code columns. Use the LIKE operator.

1. Add an ORDER BY clause and sort by **Postal_Code**. What is the default sort order?
2. Add the keyword DESC after the **Postal_Code** column in the ORDER BY clause. What does the DESC option do?
3. Add a secondary and a tertiary sort columns to sort by **Street_Number and Street_Name**. Who is the first employee on the report?
4. Remove **Street_Number** from the SELECT clause and rerun the query. Are the results still sorted by **Street_Number** within **Postal_Code and Street_Name**?

Copyright © SAS Institute Inc. All rights reserved.

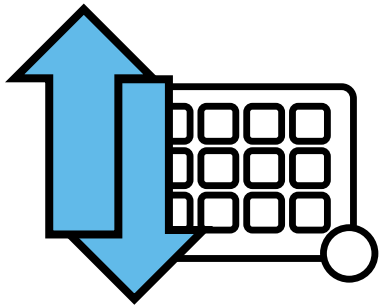
2.04 Activity – Correct Answer

1. What is the default sort order? **Ascending**
2. What does the DESC option do? **Changes the sort order to descending**
3. Who is the first employee on the report? **Tywanna Mcdade**
4. Are the results still sorted by **Street_Number** within **Postal_Code and Street_Name**? **Yes, you can sort by any columns in the input table even if they are not in the SELECT clause.**

```
proc sql;
select Employee_ID, Employee_Name, Street_Name, Postal_Code
       from orion.employee_addresses
       where City ="San Diego" AND Postal_Code LIKE "920%"
       order by Postal_Code desc, Street_Number, Street_Name;
quit;
```


Copyright © SAS Institute Inc. All rights reserved.

Order Columns by Position

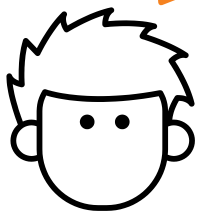


1 2 3

```
proc sql;
select FirstName, LastName, CreditScore
    from sq.customer
    where CreditScore > 830
    order by 3 desc, 2;
quit;
```

First Name	Last Name	CreditScore
Donald	Leyva	848
Elsie	Mathe	848
Christopher	Miras	848
Christopher	Murello	848
Gladys	Taylor	848
Joan	Beekman	847
Helene	Fearen	847

You can also sort by ***column position***.



Copyright © SAS Institute Inc. All rights reserved.

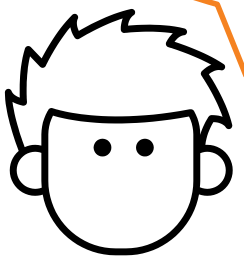
Creating a New Column

expression **AS** *alias*

```
proc sql;
select FirstName, LastName, UserID,
       yrdif(dob,'01jan2019'd) as Age
    from sq.customer(obs=100);
quit;
```

First Name	Last Name	User ID	Age
Rodney	Joyner	rodmajoyner6611@n/a.com	52.9726
Jeanne	Ballenger	jeacaballenger638@fakeemail.com	55.56712
Brian	Harper	bridaharper4714@invalid.com	71.55068
Thomas	Henderson	thoerhenderson6322@ismissing.com	55.10959
[illegible]	[illegible]	[illegible]	[illegible]

Create a report that contains customers who are age 70 and over.



Copyright © SAS Institute Inc. All rights reserved.

2.05 Activity

Use Customers dataset and perform the following tasks to find all customers 70 years old and older:

1. SELECT Customer_ID, Customer_FirstName and Customer_LastName
2. Use the expression **yrdif(Customer_BirthDate, today())** in the SELECT clause after the Customer_LastName to create a new column. Run the query and examine the results. What is the name of the new column?
3. Add **as AgeFracional** after your function. Run the query and examine the results. What changes?
4. Add a WHERE clause to return rows where **AgeFracional** is greater than or equal to 70. Run the query. Did the query run successfully?

Copyright © SAS Institute Inc. All rights reserved.

2.05 Activity – Correct Answer

2. What is the name of the new column? **Without the AS keyword, the new column does not have a name.**
3. What changes? **The AS keyword specifies a column name for the new column.**
4. Did the query run successfully? **No. The WHERE clause is evaluated *before* the SELECT clause. Therefore, columns used in the WHERE clause *must* exist in the table listed in the FROM clause.**

ERROR: The following columns were not found in the contributing tables: AgeFractional.

Copyright © SAS Institute Inc. All rights reserved.

Subsetting Calculated Values

```
proc sql;
select Customer_ID, Customer_FirstName, Customer_LastName,
       yrdif(dob, today()) as AgeFractional
    from orion.customers
    where yrdif(dob, today())>=70;
quit;
```

ANSI

```
proc sql;
select Customer_ID, Customer_FirstName, Customer_LastName,
       yrdif(dob,'01jan2019'd) as AgeFractional
    from sq.customer
    where calculated AgeFractional>=70;
quit;
```


Copyright © SAS Institute Inc. All rights reserved.

Another way of calculating Age

```
proc sql;
select Customer_ID, Customer_FirstName, Customer_LastName,
       intck('year', Customer_BirthDate, today()) as Age,
       yrdif(Customer_BirthDate, today()) as AgeFracional
    from ORION.Customers
quit;
```

What is the difference between Age and AgeFractional columns?

Copyright © SAS Institute Inc. All rights reserved.

Assigning Values to a New Column Conditionally

CODE	VALUE
M	**Married**
S	**Single**
D	**Divorced**
W	**Widowed**

CATEGORY	RANGE
Excellent	**750+**
Good	**700 – 749**
Fair	**650 – 699**
Poor	**550 – 649**
Bad	**550 & below**

Customer Partial

First Name	Last Name	State	Married	MarriedCategory	CreditScore	Category
Rodney	Joyner	WI	M	Married	711	Good
Jeanne	Ballenger	WA		Unknown	750	Excellent
Brian	Harper	WI	M	Married	790	Excellent
Thomas	Henderson	WA	S	Single	635	Poor
Becky	Cheers	WI	M	Married	716	Good
Alberto	Texter	WI	S	Single	684	Fair
Peter	Schmand	WA	M	Married	617	Poor
Danielle	Bell	WI	M	Married	639	Poor
Robert	Brousseau	WI	M	Married	687	Fair
Sharon	Howell	WA	S	Single	.	Unknown

Copyright © SAS Institute Inc. All rights reserved.

Simple CASE Expression

```
proc sql;
select FirstName, LastName, State, CreditScore,
       case
          when CreditScore >= 750 then "Excellent"
          when CreditScore >= 700 then "Good"
          when CreditScore >= 650 then "Fair"
          when CreditScore >= 550 then "Poor"
          when CreditScore >= 0 then "Bad"
          else "Unknown"
       end as Category
    from sq.customer(obs=1000);
quit;
```

GT LT <= >= NE = ...

Copyright © SAS Institute Inc. All rights reserved.

Simple CASE Expression

```
proc sql;
select FirstName, LastName, State, CreditScore,
       case
          when CreditScore >= 750 then "Excellent"
          when CreditScore >= 700 then "Good"
          when CreditScore >= 650 then "Fair"
          when CreditScore >= 550 then "Poor"
          when CreditScore >= 0 then "Bad"
          else "Unknown"
       end as Category
    from sq.customer(obs=1000);
quit;
```

ELSE provides alternate action if no WHEN expressions are true.

The first **WHEN** clause evaluated as ***true*** determines which value the CASE expression returns.

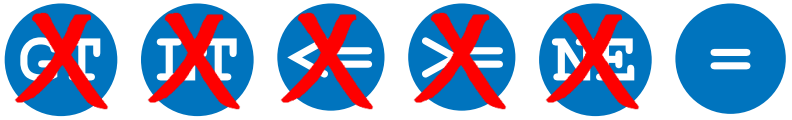
Copyright © SAS Institute Inc. All rights reserved.

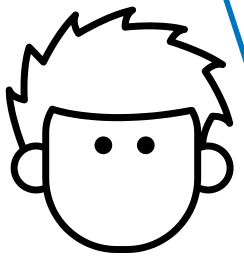
CASE-Operand Form

```
proc sql;
select FirstName, LastName, State, CreditScore,
       case Married
          when "M" then "Married"
          when "D" then "Divorced"
          when "S" then "Single"
          when "W" then "Widowed"
          else "Unknown"
       end as Category
    from sq.customer(obs=1000);
quit;
```

equivalent of
Married=“*D*“
(In this example Married is used as the name of the column)

A test of ***equality*** is implied.





Copyright © SAS Institute Inc. All rights reserved.

2.06 Activity

Let’s decode the Order_Type column in the Orders table as a new column called Order_Category.

Order_Category should be ‘Retail’ if the order type is 1, ‘Catalog’ if the order type is 2 and ‘Online’ if the order type is 3.

Use a CASE expression to create a new column.

Copyright © SAS Institute Inc. All rights reserved.

2.06 Activity – Correct Answer

Simple CASE

```
proc sql;
select Order_ID, Order_Type,
       case
          when Order_Type = 1 then "Retail"
          when Order_Type = 2 then "Catalog"
          when Order_Type = 3 then "Online"
          else "Unknown"
       end as Order_Category
    from ORION.Orders;
quit;
```

CASE-OPERAND

```
proc sql;
select Order_ID, Order_Type,
       case Order_Type
          when 1 then "Retail"
          when 2 then "Catalog"
          when 3 then "Online"
          else "Unknown"
       end as Order_Category
    from ORION.Orders;
quit;
```

Syntax Summary

SELECT *col-name, col-name*
FROM *input-table*
WHERE *expression*
ORDER BY *col-name* <**DESC**>**;**

Query

"01JAN2000 12:00:00"dt
"01JAN2019"d
"14:45:32"t

SAS Dates and Times

WHERE *col-name* **IS NULL**
WHERE *col-name* **LIKE**
WHERE *col-name* **BETWEEN-AND**
WHERE CALCULATED *column*

Filter Data

column **AS** *alias*
CASE EXPRESSION

Create Columns

Copyright © SAS Institute Inc. All rights reserved.

Practice Exercise 3

This exercise reinforces the concepts discussed previously.

Copyright © SAS Institute Inc. All rights reserved.