



Creating Buckets in Analytical Datasets Using PROC SQL

Katie Ronk



SYSTEMS SEMINAR CONSULTANTS, INC.

2997 Yarmouth Greenway Drive
Madison, WI 53711
(608) 278-9964



Systems Seminar Consultants, Inc. is a SAS Alliance Quality Partner™ of SAS. Our team of SAS software experts has a broad base of knowledge and experience working with a variety of complex systems in a number of diverse industry settings. This knowledge and experience is leveraged to help you effectively achieve your business goals.



Free SAS Newsletter

Our popular publication, *The Missing Semicolon*™, shares SAS software solutions developed by our staff and provides additional technical assistance to our customers.

SAS Training Services

For over 1,000 students each year, we make SAS software easier to understand, use, and support.

- Public training schedules are posted on our web site.
- Private on-site training options are also available.



SAS Consulting Services

Our staff of SAS consultants is well-versed in a variety of business areas.

Our specialty areas include:

- Data systems development
- Decision support and business consultation
- Market research and analysis

SAS Help Desk Services

Our SSC team of experts is available to solve your company's daily SAS problems. Call us to develop a customized support plan that meets your company's needs.

For More Information

To receive additional information about our services or discuss a specific cost-effective solution for your company:

- Call us at: (608) 278-9964
- Check our Website at: www.sys-seminar.com
- Contact us at: 2997 Yarmouth Greenway Drive, Madison, WI 53711



Katie M. Ronk, Principal Consultant

- 15 years experience working on analytical projects such as survey research, data analysis, statistical modeling and report development
- Designs and manages the development of analytic systems for her clients
- Works with many clients in the fields of direct marketing, health care and financial services
- Invited SQL Presenter at SAS Global Forum
- Devout SAS user for over 10 years
- Holds a Bachelor's of Science from the University of Wisconsin in Sociology, Concentration in Analysis and Research.

Creating Buckets for Analytical Datasets using SQL



- Why do we need analytical datasets?
- What are buckets?
- How do we use buckets?
- How do I create buckets in SAS?
 - PROC SQL method
 - Data step method

Tools for Creating Buckets for Analytical Datasets

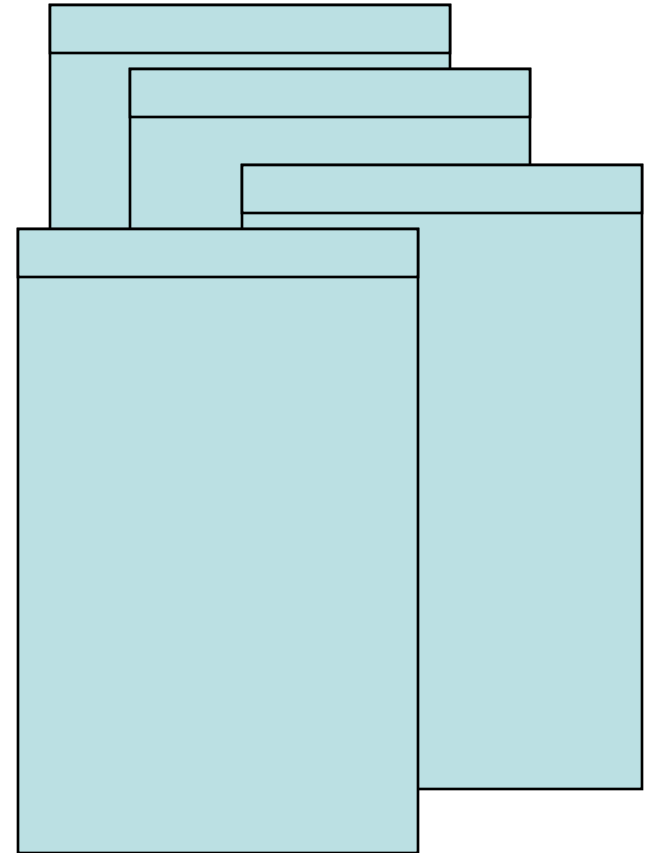
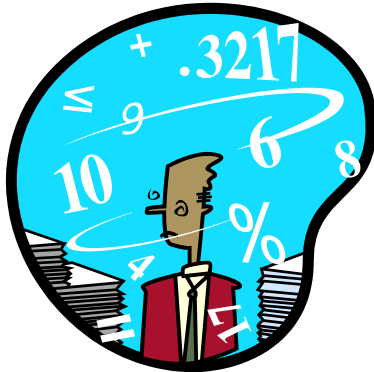


	Traditional SAS	SQL
Sorting	PROC SORT	ORDER BY
Summarized	PROC SUMMARY/ FIRST. LAST.	GROUP BY, summary functions
Joining	MERGE w/ BY	INNER AND OUTER JOINS
Conditional Processing	IF-THEN Statements	CASE-WHEN

Problem: Too Much Information



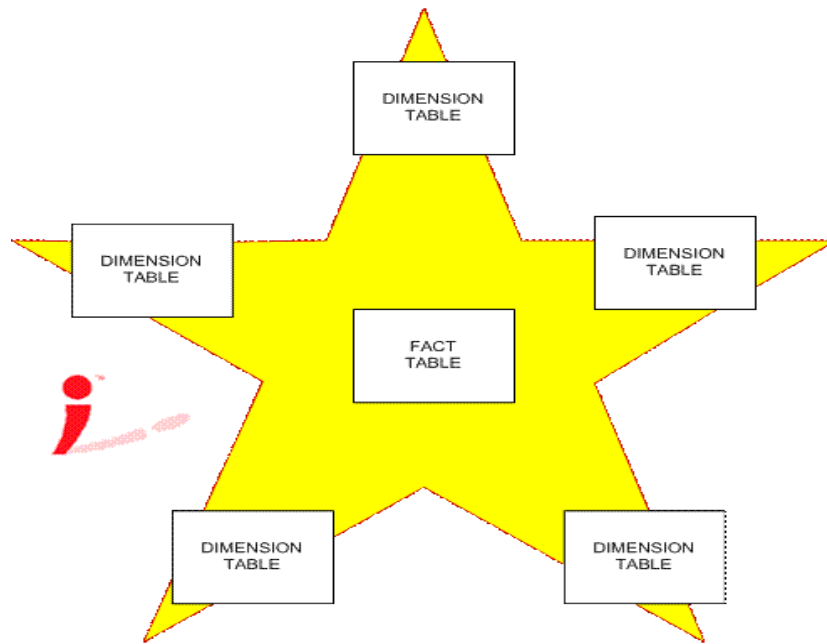
- Disparate data in a variety of sources.
- Need one data set to handle majority of analytical and reporting needs.



Solution: Build an Analytical Dataset

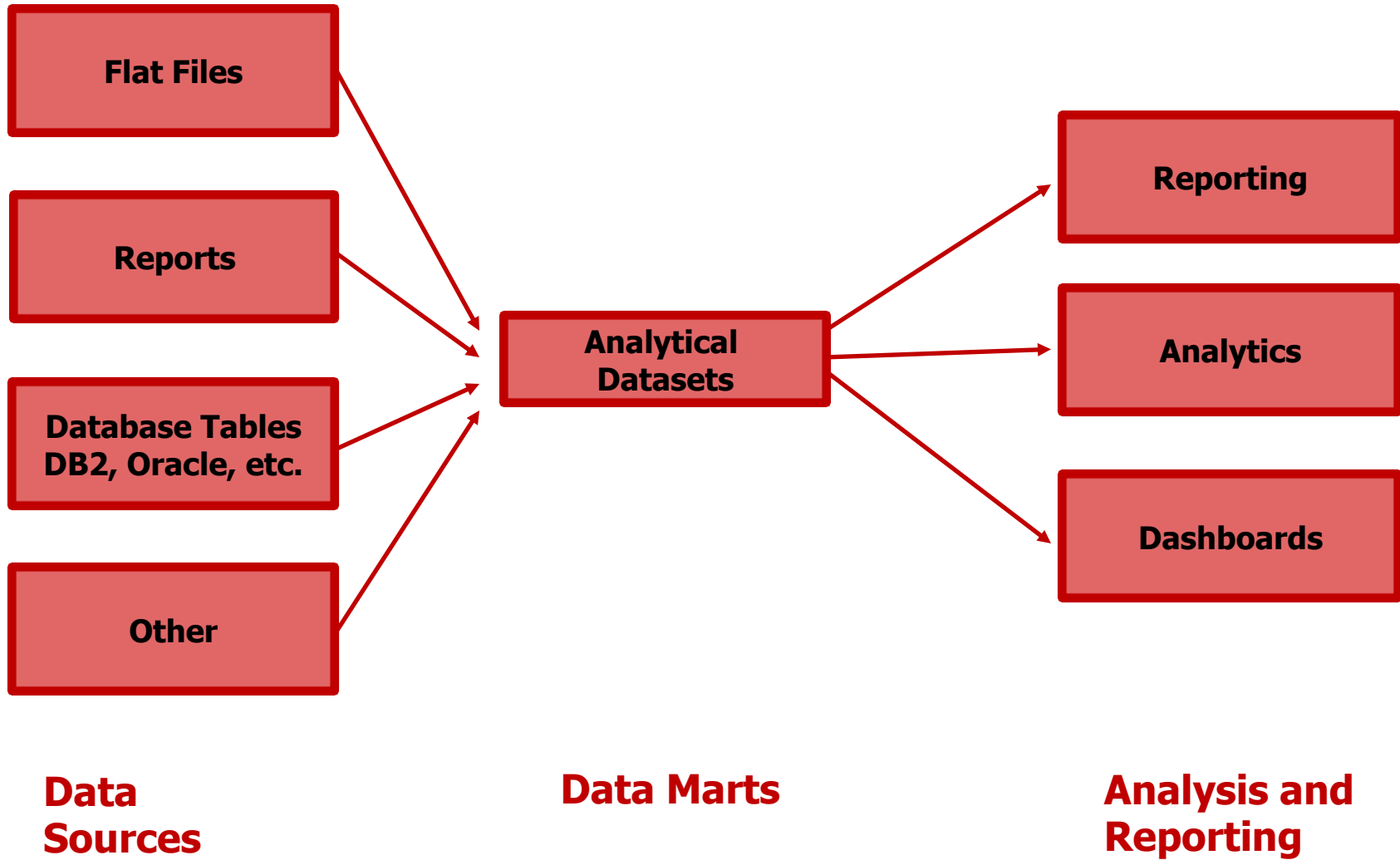


- Build a Summarized Fact Table for Reporting & Analysis
- Need to choose unit of analysis to summarize information
 - Customer
 - Product
 - Store / Sales Person



- Hierarchical data will not be discussed here
- Assume working towards one fact table

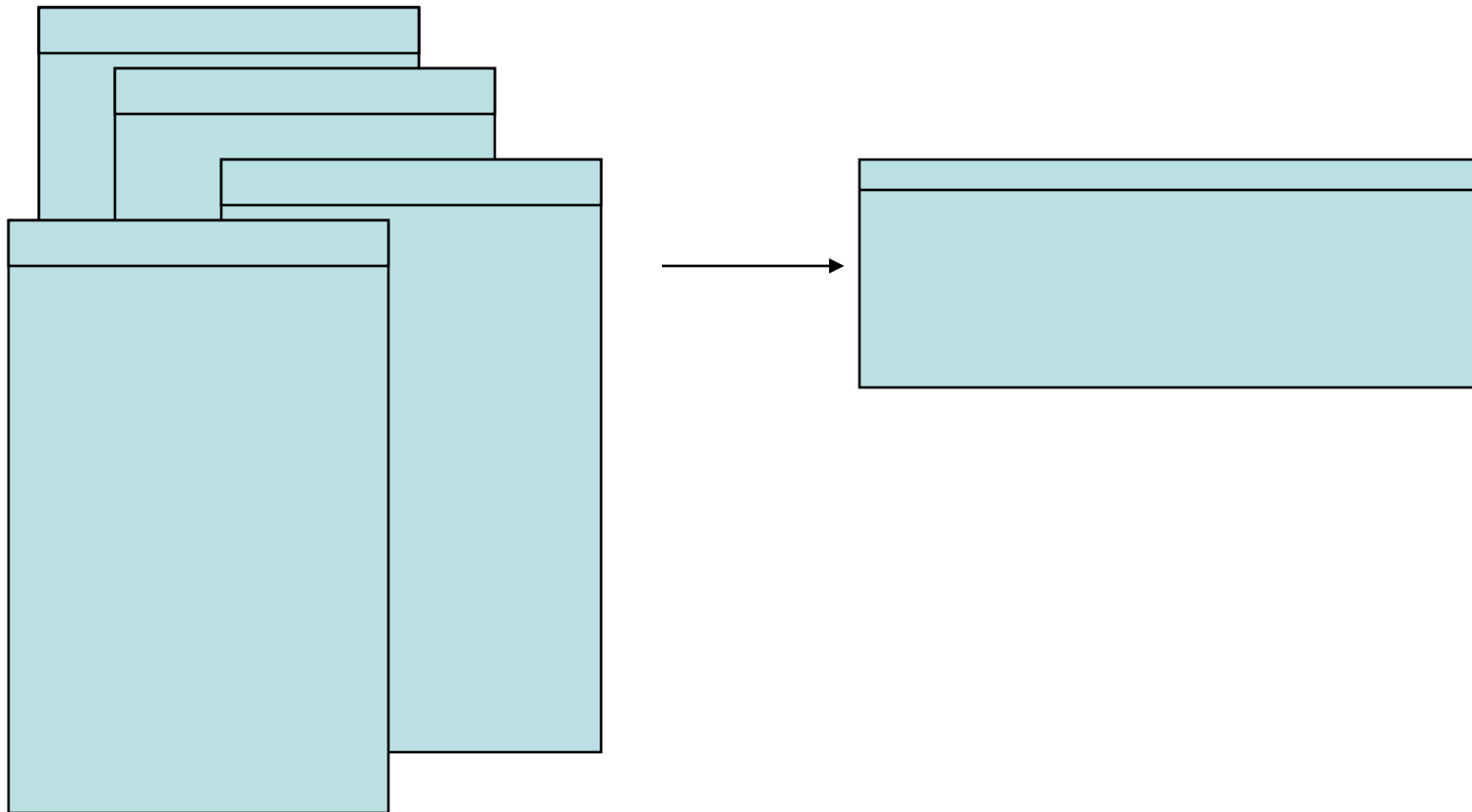
Analytic Dataset as a Type of Data Mart



Methods for Creating an Analytical Dataset



Summarize transactional data to one row per unit of analysis with bucketed information.



Case Study – Airline Data



Subset of Airline Tables

AIRPORT

AIRPORTCODE
AIRPORTCOUNTRYCODE
AIRPORTNAME
.....

CUSTOMERS

CUSTID
FF_NUMBER
ADDRESS1
ADDRESS2
CITY
DOB
STATE
ZIPCODE
.....

FLIGHTHISTORY

CUSTID
FLIGHTID
AIRCRAFTID
AIRPORTCODEARRIVAL
AIRPORTCODEDEPARTED
ARRIVALACTUALDATETIME
ARRIVALSCHEDULEDDATETIME
DEPARTUREACTUALDATETIME
DEPARTURESCHEDULEDDATETIME
DESTINATIONAIRPORTCODE
SEAT
.....

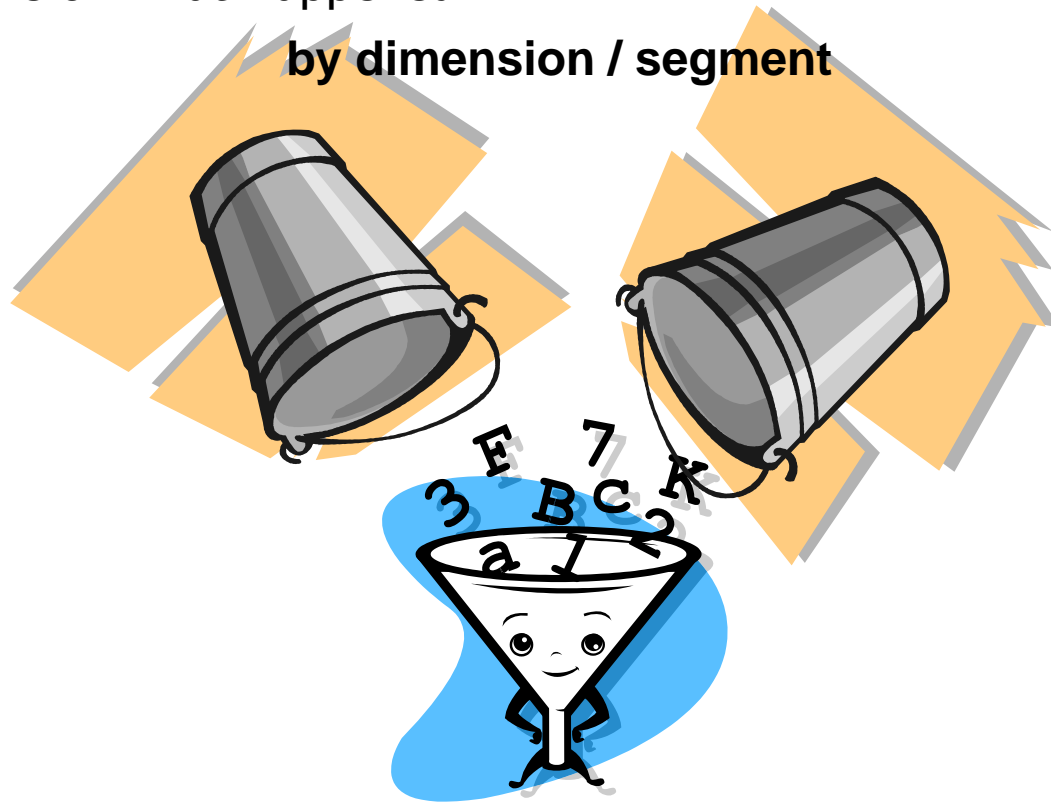
ORDERHISTORY

CUSTID
ORDERID
ARRIVALDATETIME
COST
DEPARTUREDATETIME
ORDERDATE
PROMOTIONCODE
TAXES
TOTALCOST
FINALDISTINATIONAIRPORTCODE
.....



- Querying Data
- Build Reports
- Get a Picture of What Happened

by dimension / segment



- *Sales Report for Customers with an International Flight in Last Year.*
- *Average Sales Revenue by Customer Type.*

Buckets can be used for Segmentation (airline example)



- Segmentation for Reports
- Often mutual exclusive (categorical), but not always
- Sometimes stored in dimension tables



Type of Traveler

Values:

International Only

Domestic Only

International/Domestic

Not Active Customer



**International
Traveler**

Values: Y/N



**Last International
Trip**

Values:

0-12 months

13-24 months

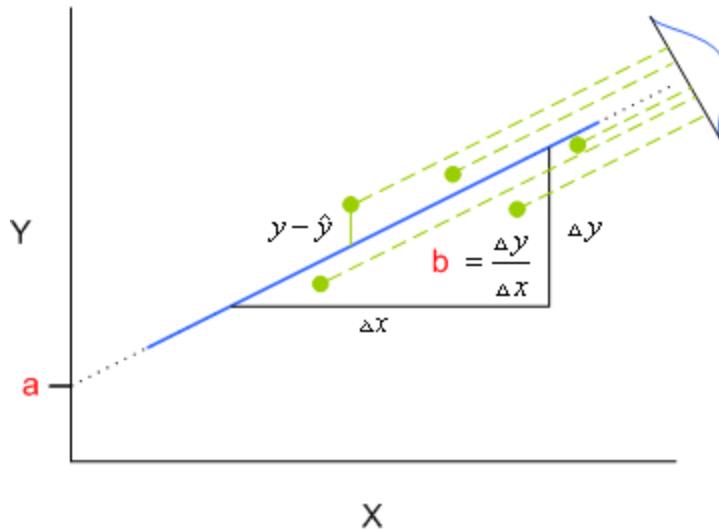
2+ years

N/A

Model Your Data!



- Everything summarized to the unit of analysis
- Find correlations between variables.
- What are you trying to explain or predict?
- What are possible explanations



- *Model to Predict Last Quarter Sales Revenue per Customer.*
- *Create correlation matrix of all variables (including demographics) to find relationships to International Flight propensity.*

Buckets can be used for Analysis (airline example)



- Continuous Values
- Report on Averages, Totals, Mins / Maxs
- Used in Statistical Models to Explain and Predict Behavior



**Number
of International
Trips**

**Number of
Domestic Trips**



**Date of Last
International
Trip**

**Date of Last
Domestic Trip**



**Revenue from
International
Trips**

**Revenue from
Domestic Trips**

Outline New Fields - Airline Analytic Table Structure



Field Name	Field Description	Source Table	Transformation Logic?
custID	customer ID	customers	select all U.S. based customers
	various customer demographics	customers	N/A
IntFlyer	International Flight Indicator	orderHistory / airport	if customer had international flight as final destination, set as 'Y', else set as 'N'
DomFlyer	Domestic Flight Indicator	orderHistory / airport	if customer had domestic flight as final destination, set as 'Y', else set as 'N'
IntFlightLastTrip	Date of Last International Flight	flightHistory / airport	Actual departure date of last trip with leg destination international.
DomFlightLast Trip	Date of Last Domestic Flight	flightHistory / airport	Actual departure date of last trip with leg destination domestic.
IntFlightRevTotal	Revenue from international final destinations.	orderHistory / airport	Revenue where final destination was international.
DomFlightRevTotal	Revenue from domestic final destinations.	orderHistory / airport	Revenue where final destination was domestic.
IntFlightTripsTotal	Total international destination purchases.	orderHistory / airport	Number purchases where final destination was international.
DomFlightTrips Total	Total domestic destination purchases.	orderHistory / airport	Number purchases where final destination was domestic.

Case Study – Airline Data



Subset of Airline Tables

AIRPORT

AIRPORTCODE
AIRPORTCOUNTRYCODE
AIRPORTNAME
.....

CUSTOMERS

CUSTID
FF_NUMBER
ADDRESS1
ADDRESS2
CITY
DOB
STATE
ZIPCODE
.....

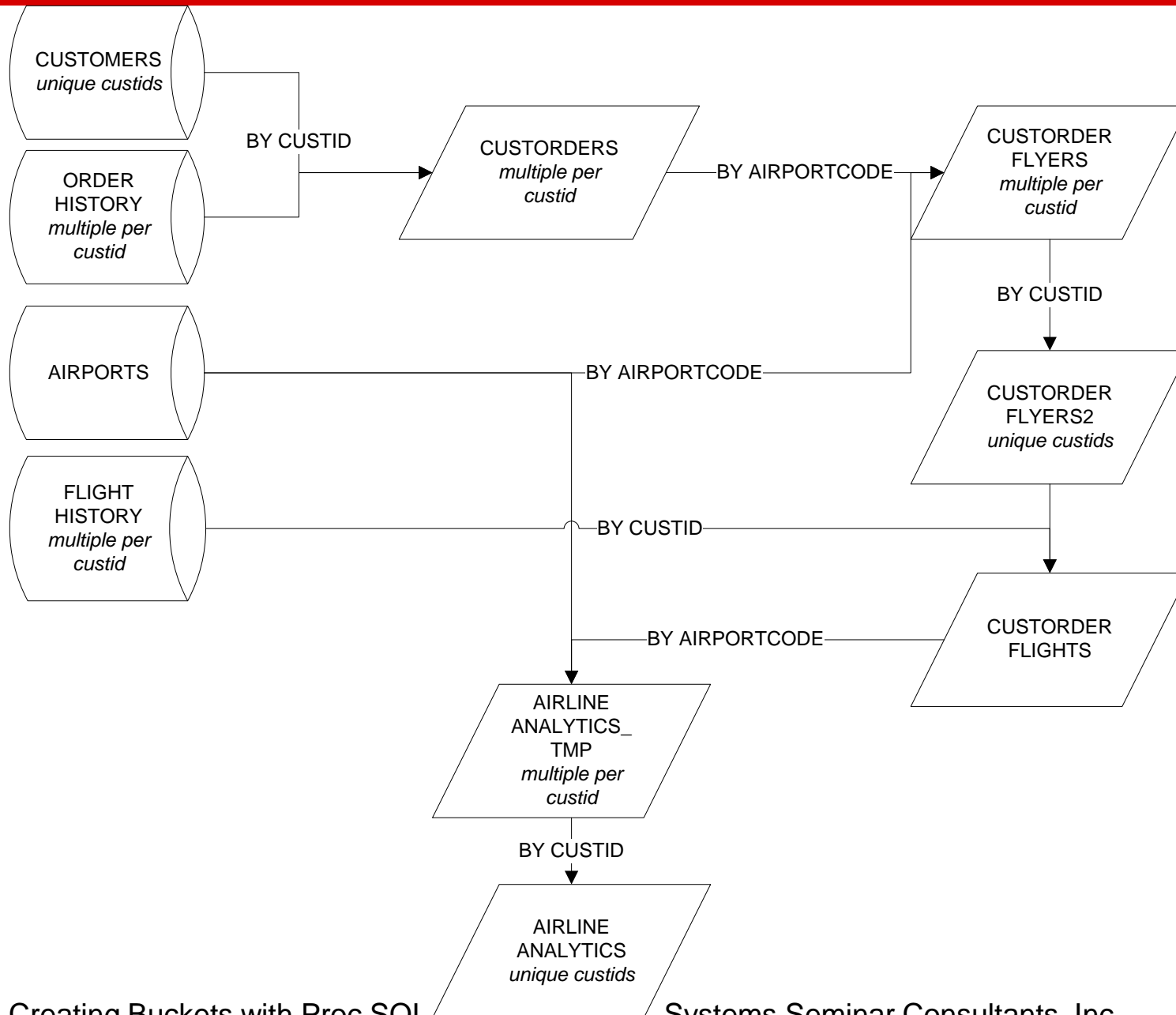
FLIGHTHISTORY

CUSTID
FLIGHTID
AIRCRAFTID
AIRPORTCODEARRIVAL
AIRPORTCODEDEPARTED
ARRIVALACTUALDATETIME
ARRIVALSCHEDULEDDATETIME
DEPARTUREACTUALDATETIME
DEPARTURESCHEDULEDDATETIME
DESTINATIONAIRPORTCODE
SEAT
.....

ORDERHISTORY

CUSTID
ORDERID
ARRIVALDATETIME
COST
DEPARTUREDATETIME
ORDERDATE
PROMOTIONCODE
TAXES
TOTALCOST
FINALDISTINATIONAIRPORTCODE
.....

Airline Analytic Dataset Non-SQL Flow Chart





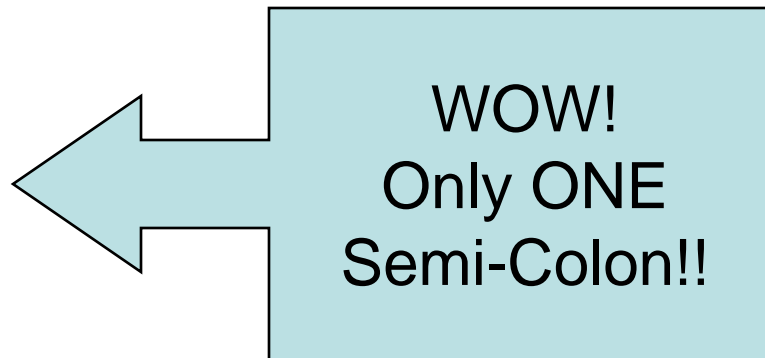
The terminology in SQL is slightly different than in standard SAS, but the meaning is the same.

<u>SAS</u>		<u>SQL</u>
dataset	=	table
variable	=	column
observation	=	row



```
PROC SQL options;
```

```
SELECT column(s)
FROM table-name | view-name
WHERE expression
GROUP BY column(s)
HAVING expression
ORDER BY column(s)
;
```



```
QUIT;
```

Notes:

- The SELECT statement describes the appearance of the query
- It contains several clauses
- The sequence of the clauses **is important**



```
PROC SQL;  
  SELECT STATE, SUM(SALES) AS TOTSALES  
  FROM USSALES  
  GROUP BY STATE;  
QUIT;
```


STATE	TOTSALES
IL	84976.57
MI	53341.66
WI	34238.57

Notes:

- GROUP BY summarizes
- Use summary functions on the numeric columns for statistics
- Other summary functions: AVG/MEAN, MAX, MIN, COUNT/FREQ/N, NMISS, STD, SUM, and VAR



The FROM clause is used to:

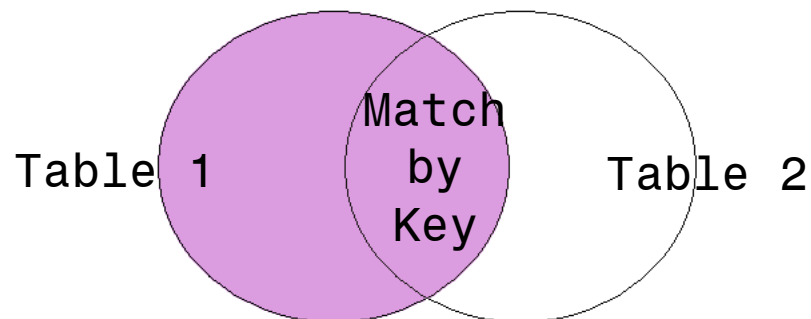
- Select source data
 - Join tables (INNER, FULL, LEFT and RIGHT)
 - Concatenate tables
- OUTER JOINS
- 

Example:

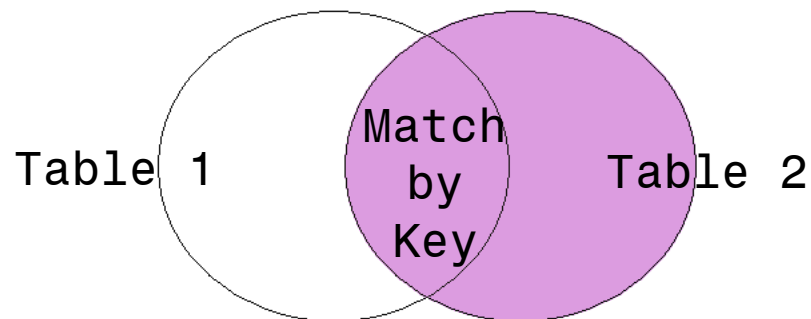
```
FROM datamart.customers c INNER JOIN  
      datamart.orders      o on c.custid=o.custid
```



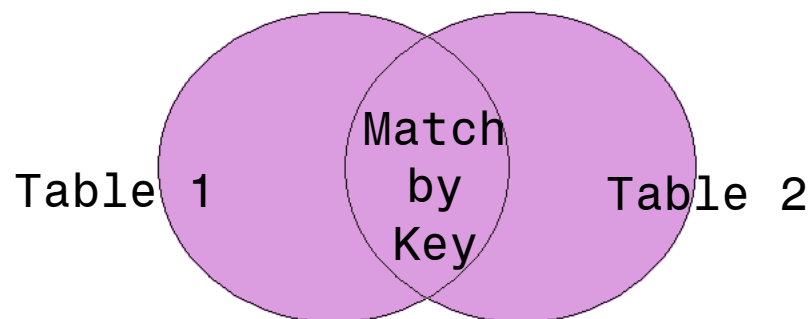
Left Join



Right Join



Full Join



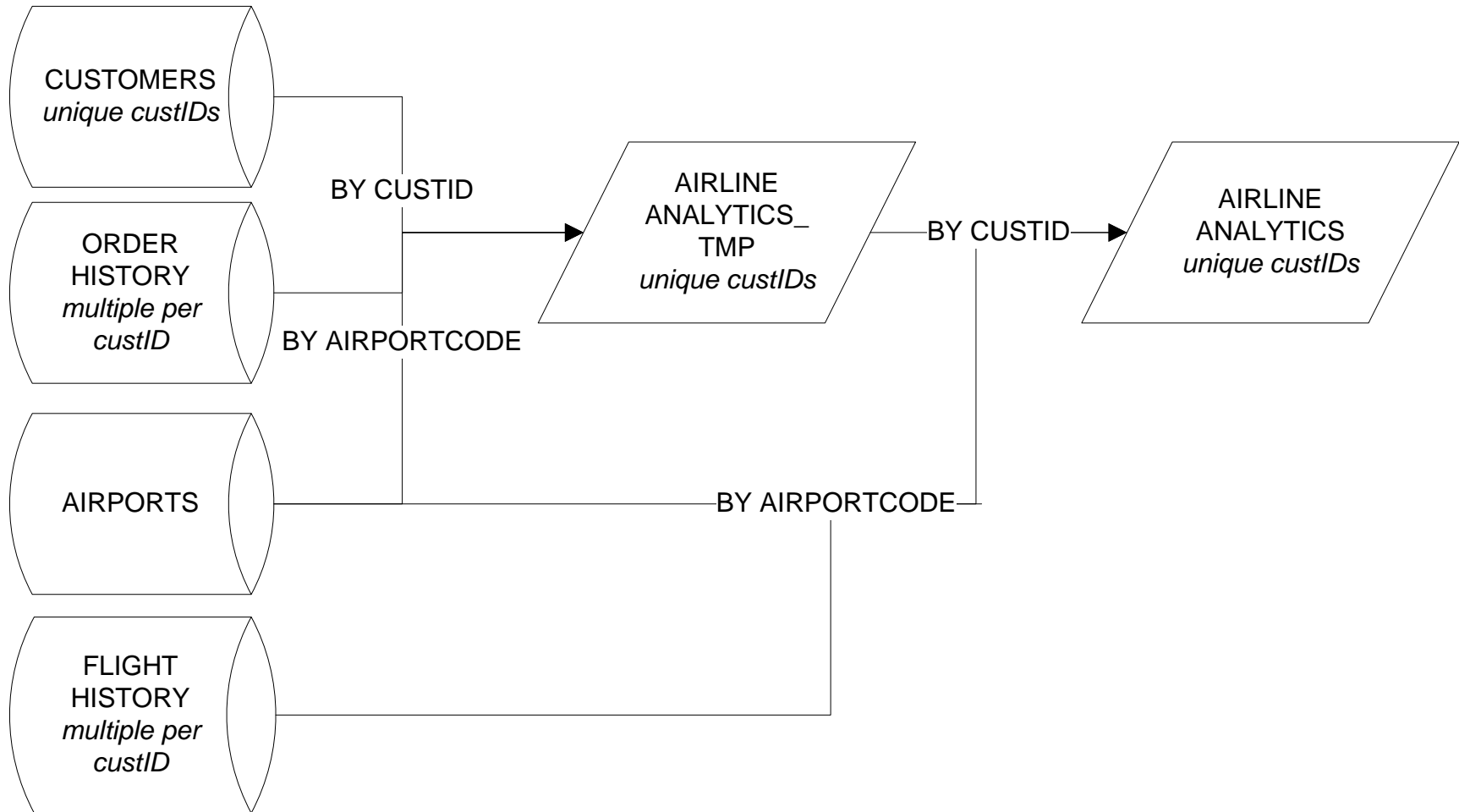


```
PROC SQL;  
  SELECT STATE,  
         CASE  
           WHEN SALES<10000 THEN 'LOW'  
           WHEN SALES<15000 THEN 'AVG'  
           WHEN SALES<20000 THEN 'HIGH'  
           ELSE 'VERY HIGH'  
         END AS SALESCAT  
  FROM USSALES;  
QUIT;
```

Notes:

- END is required when using the CASE
- WHENs in descending probability improve efficiency
- With no ELSE condition, missing values result

Airline Analytic Dataset SQL Flow Chart



SQL Method – Two Steps, First Join



```
proc sql;
  create table airlineAnalytics_tmp as
  select c.custID, c.dob, c.state, c.zipcode, c.ff_number,
         max(case when a.airportcountryCode not in ('USA','')
                  then 'Y'
                  else 'N'
                end) as IntFlyer,
         sum(case when a.airportcountryCode not in ('USA','')
                  then 1
                  else 0
                end) as IntFlightTripsTotal,
         sum(case when a.airportcountryCode not in ('USA','')
                  then totalCost
                  else 0
                end) as IntFlightRevTotal,
         max(case when a.airportcountryCode in ('USA')
                  then 'Y'
                  else 'N'
                end) as DomesticFlyer
  continued...
```



continued...

```
        sum(case when a.airportcountryCode in ('USA')
                then 1
                else 0
            end) as DomFlightTripsTotal,
        sum(case when a.airportcountryCode in ('USA')
                then totalCost
                else 0
            end) as DomFlightRevTotal
from airlines.customers      c
left join
    airlines.orderHistory    o
    on c.custid=o.custId
left join
    airlines.airports        a on
    on a.airportCode=o.finalDestinationAirportCode
group by c.custID, c.dob, c.state,
        c.zipcode,c.ff_number;

quit;
```



```
proc sql;
  create table airlineAnalytics_SQL as
  select tmp.custID, tmp.dob, tmp.state, tmp.zipcode,
         tmp.ff_number, tmp.IntFlyer,
         tmp.IntFlightTripsTotal, tmp.IntFlightRevTotal,
         tmp.DomesticFlyer, tmp.DomFlightTripsTotal,
         tmp.DomFlightRevTotal,
         max(case when a2.airportCountryCode in ('USA')
                  then datepart(f.DEPARTUREACTUALDATETIME)
                  else .
                end) as DomFlightLastTrip format=date9.,
         max(case when a2.airportCountryCode not in ('USA','')
                  then datepart(f.DEPARTUREACTUALDATETIME)
                  else .
                end) as IntFlightLastTrip format=date9.
```

Continued...



Continued...

```
from airlineAnalytics_tmp tmp
  left join  airlines.flightHistory f
    on tmp.custId=f.custID
  left join  airlines.airports      a2
    on a2.airportCode=f.destinationAirportCode
group by tmp.custID, tmp.dob, tmp.state, tmp.zipcode,
         tmp.ff_number,          tmp.IntFlyer,
         tmp.IntFlightTripsTotal, tmp.IntFlightRevTotal,
         tmp.DomesticFlyer,      tmp.DomFlightTripsTotal,
         tmp.DomFlightRevTotal;

quit;
```



About our SQL Based Program that Joins and Summarizes **Four** Tables:

- Only **TWO** Steps Needed
- Easy to understand
- Compared to traditional data step processing – saves resources
 - Reduced run time¹ (at least four times as fast with sample dataset)
 - Reduced disk space requirements
- Easy to code (once we understand syntax)

¹ SQL is not always a faster approach and should be carefully benchmarked before productionalizing code.



Spotlight Code – Joins

```
        sum(case when a.airportcountryCode in ('USA')
                then 1
                else 0
            end) as DomFlightTripsTotal,
        sum(case when a.airportcountryCode in ('USA')
                then totalCost
                else 0
            end) as DomFlightRevTotal
from airlines.customers      c
left join
    airlines.orderHistory    o
    on c.custid=o.custId
left join
    airlines.airports        a on
    on a.airportCode=o.finalDestinationAirportCode
group by c.custID, c.dob, c.state,
        c.zipcode,c.ff_number;

quit;
```



Spotlight Code – CASE-WHEN

```
max(case when a.airportcountryCode not in ('USA', '')
      then 'Y'
      else 'N'
     end) as IntFlyer,
```

Value Either 'Y' or 'N', new Column called IntFlyer



Spotlight Code – GROUP BY

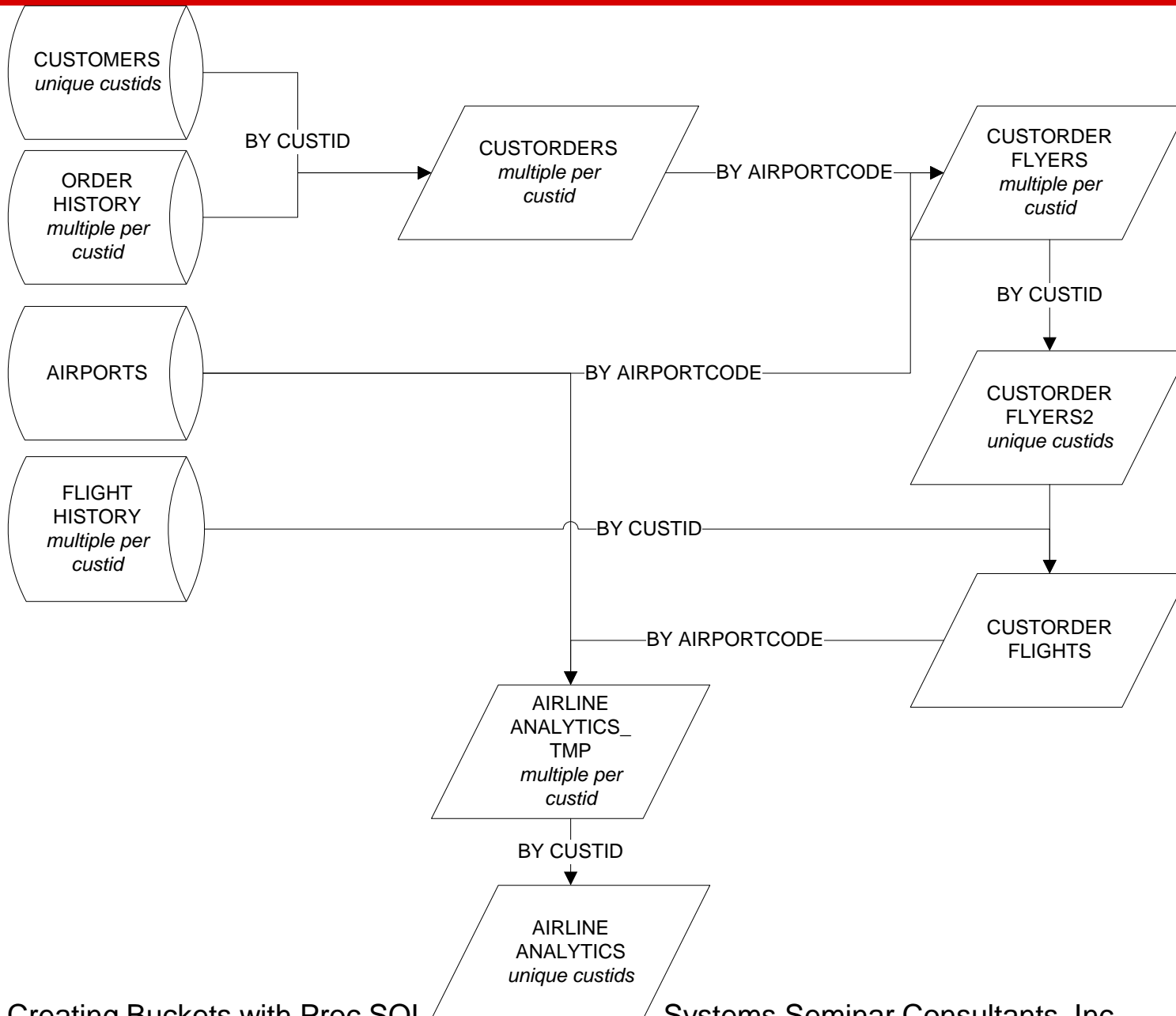
```
select c.custID, c.dob, c.state, c.zipcode, c.ff_number,
       max(case when a.airportcountryCode not in ('USA','')
                then 'Y'
                else 'N'
            end) as IntFlyer,
       sum(case when a.airportcountryCode not in ('USA','')
                then 1
                else 0
            end) as IntFlightTripsTotal,
....
from airlines.customers      c
  left join ....
group by c.custID, c.dob, c.state,
         c.zipcode,c.ff_number;
```



Spotlight Code – GROUP BY, results can be character or numeric.

```
select c.custID, c.dob, c.state, c.zipcode, c.ff_number,
       max(case when a.airportcountryCode not in ('USA','')
              then 'Y'
              else 'N'
            end) as IntFlyer,
       sum(case when a.airportcountryCode not in ('USA','')
              then 1
              else 0
            end) as IntFlightTripsTotal,
....
from airlines.customers      c
  left join ....
group by c.custID, c.dob, c.state,
         c.zipcode,c.ff_number;
```

Airline Analytic Dataset Non-SQL Flow Chart





Pull Customer & Order Table Together

```
proc sort data=airlines.customers
    (keep=custid dob state zipcode ff_number)
    out=work.customers;
  by custid;
run;

proc sort data=airlines.orderHistory
    (keep=custID finalDestinationAirportCode totalCost)
    out=work.orderHistory;
  by custid;
run;

data custOrder;
  merge work.customers(in=cust)
        work.orderHistory(in=oh);
  by custid;
  if cust;
  keep custid dob state zipcode ff_number
        finalDestinationAirportCode totalCost;
run;
```



Merge with Airport Dataset to Get Airport Country Code

```
proc sort data=work.custOrder;
  by finalDestinationAirportCode;
run;
```

```
proc sort data=airlines.airports
  (keep=airportCode airportCountryCode)
  out=work.airports;
  by airportCode;
run;
```

```
data custOrderFlyers;
  merge work.custOrder(in=cust
    rename=(finalDestinationAirportCode=airportCode))
    work.airports(in=oh );
  by airportCode;
  if cust;
  keep custid dob state zipcode ff_number totalCost
    airportCountryCode;
run;
```

Non-SQL Method – Third Sort and Data Step



Now Sort by Customer ID to Prep for FIRST. LAST. Processing.

```
proc sort data=custOrderFlyers;
  by custId;
run;
data custOrderFlyers2;
  set custOrderFlyers;
  by custId;
  retain IntFlyer          domesticFlyer 'N'
         DomFlightTripsTotal domFlightRevTotal
         IntFlightRevTotal  IntFlightTripsTotal 0;
  keep custid  dob  state  zipcode  ff_number
       intFlyer          domesticFlyer
       domFlightTripsTotal domFlightRevTotal
       intFlightRevTotal  intFlightTripsTotal;
  if airportCountryCode in ("USA") then
  do;
    domesticFlyer='Y';
    domFlightRevTotal+totalCost;
    domFlightTripsTotal+1;
  end;  cont...
```



Continued DATA Step using FIRST. LAST. Processing.

Cont..

```
else if airportCountryCode not in ("USA","") then
  do;
    intFlyer='Y';
    intFlightRevTotal+totalCost;
    intFlightTripsTotal+1;
  end;
if last.custid then
  do;
    output;
    intFlyer='N';
    domesticFlyer='N';
    domFlightTripsTotal=0;
    domFlightRevTotal=0;
    intFlightRevTotal=0;
    intFlightTripsTotal=0;
  end;
run;
```

Non SQL Method – Fourth Sort and Merge



Now Bring in Flight History Dataset to Get Destinations from Any Flight Legs.

```
proc sort data=airlines.flightHistory(keep=custId
    destinationAirportCode departureActualDatetime)
    out=work.flightHistory;
  by custid;
run;
```

```
data custOrderFlights;
  merge work.custOrderFlyers2(in=cust)
        work.flightHistory(in=fH);
  by custid;
  if cust;
  keep custid dob      state      zipcode ff_number
      IntFlyer      DomesticFlyer
      DomFlightTripsTotal      domFlightRevTotal
      IntFlightRevTotal      IntFlightTripsTotal
      destinationAirportCode departureActualDatetime ;
run;
```

Non SQL Method – Fifth Sort and Merge



Resort the Data by Airport Code so the Airport Data Can be Brought Back.

```
proc sort data=work.custOrderFlights;
  by destinationAirportCode;
run;

data work.airlineAnalyticsTmp;
  merge work.custOrderFlights(in=cust
    rename=(DestinationAirportCode=airportCode))
    work.airports(in=oh );
  by airportCode;
  if cust;
  keep custid dob      state      zipcode ff_number
      IntFlyer      DomesticFlyer
      DomFlightTripsTotal      domFlightRevTotal
      IntFlightRevTotal      IntFlightTripsTotal
      departureActualDatetime      airportCountryCode;
run;
```

Non SQL Method – Sixth Sort and Data Step



Resort the Data to Use FIRST. And LAST. to Create Final Dataset.

```
proc sort data=work.airlineAnalyticsTmp;
  by custid departureActualDatetime;
run;

data work.airlineAnalytics;
  set work.airlineAnalyticsTmp;
  by custid;
  retain IntFlightLastTrip DomFlightLastTrip;
  keep custid dob      state      zipcode ff_number
      IntFlyer          DomesticFlyer
      DomFlightTripsTotal  domFlightRevTotal
      IntFlightRevTotal   IntFlightTripsTotal
      IntFlightLastTrip   DomFlightLastTrip;
  format intFlightLastTrip domFlightLastTrip date9.;
```

Continued..

Non SQL Method – Sixth Sort and Data Step



Resort the Data to Use FIRST. And LAST. to Create Final Dataset.

Continued...

```
if airportCountryCode in ("USA") then
  do;
    DomFlightLastTrip=max(
      datepart(departureActualDateTime),
      domFlightLastTrip);
  end;
else if airportCountryCode not in ("USA","") then
  do;
    IntFlightLastTrip=max(
      datepart(departureActualDateTime),
      intFlightLastTrip);
  end;
if last.custid then
  do;
    output;
    intFlightLastTrip=.;
    domFlightLastTrip=.;
  end;
```

```
run;
```

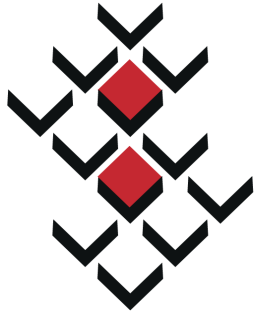


About the Program that Joins and Summarizes **Four** Tables:

- 14 Number of Steps Needed to Create Dataset
- 9 Work Datasets Created, Not Including Final Analytic Dataset
- 8 Sort Steps Required
- 6 Total Data Steps
- 4 Data Merge Steps
- 2 Data Set Steps
- More CPU Time Than Necessary?
- More Work Space Than Necessary?
- More Complicated Than Necessary?



- Many great tools in SAS to create analytic datasets
- Creating buckets is possible and often efficient using SQL
- Be sure to benchmark if looking for the method with the lowest CPU, etc.



SYSTEMS SEMINAR CONSULTANTS, INC.

SAS® Training, Consulting, & Support Services
2997 Yarmouth Greenway Drive • Madison, WI 53711
(608) 278-9964 x312 • Fax (608) 278-0065

www.sys-seminar.com



Katie Ronk

Principal Consultant

kronk@sys-seminar.com

