PROC SQL Joins

Lesson 3: SQL Joins

Joining Tables

smallcustomer Partial

FirstName	LastName	…	AccountID
Gary	Sienkiewicz	…	1010159565
Sergio	Lefeld	…	1010367330
John	Oliver	…	2020012887
Iva	Bower	…	3030085224

smalltransaction Partial

AccountID	DateTime	BankID	…
.	07MAY18:15:35:02	.	…
1010159565	16SEP18:14:57:08	101010101	…
1010183063	24FEB18:17:27:42	101010101	…
1010367330	15MAY18:17:54:21	101010101	…
1010367330	17OCT18:11:02:38	101010101	…

Joining Tables

smallcustomer Partial

FirstName	LastName	...	AccountID
Gary	Sienkiewicz	...	1010159565
Sergio	Lefeld	...	1010367330
John	Oliver	...	2020012887
Iva	Bower	...	3030085224

Primary key

smalltransaction Partial

AccountID	DateTime	BankID	...
.	07MAY18:15:35:02	.	...
1010159565	16SEP18:14:57:08	101010101	...
1010183063	24FEB18:17:27:42	101010101	...
1010367330	15MAY18:17:54:21	101010101	...
1010367330	17OCT18:11:02:38	101010101	...

Foreign key

Joining Tables

smallcustomer Partial

FirstName	LastName	…	AccountID
Gary	Sienkiewicz	…	1010159565
Sergio	Lefeld	…	1010367330
John	Oliver	…	2020012887
Iva	Bower	…	3030085224

smalltransaction Partial

AccountID	DateTime	BankID	…
.	07MAY18:15:35:02	.	…
1010159565	16SEP18:14:57:08	101010101	…
1010183063	24FEB18:17:27:42	101010101	…
1010367330	15MAY18:17:54:21	101010101	…
1010367330	17OCT18:11:02:38	101010101	…

FirstName	LastName	…	AccountID	DateTime	BankID	…
Gary	Sienkiewicz	…	1010159565	16SEP18:14:57:08	101010101	…
Sergio	Lefeld	…	1010367330	15MAY18:17:54:21	101010101	…
Sergio	Lefeld	…	1010367330	17OCT18:11:02:38	101010101	…

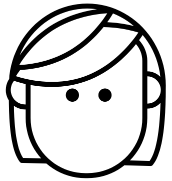
SQL Join Syntax

SELECT *col-name, col-name*
FROM *table1, table2*

```
proc sql;
select *
    from sq.smallcustomer, sq.smalltransaction;
quit;
```

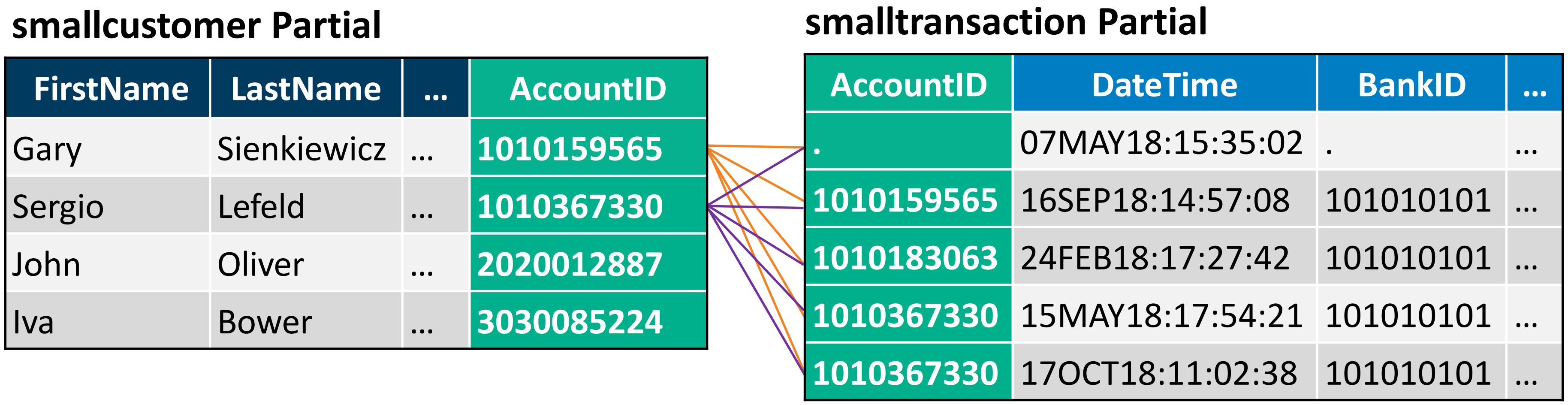
8 rows

12 rows



List the table names in the **FROM** clause.

Default Join



smallcustomer Partial

FirstName	LastName	…	AccountID
Gary	Sienkiewicz	…	1010159565
Sergio	Lefeld	…	1010367330
John	Oliver	…	2020012887
Iva	Bower	…	3030085224

smalltransaction Partial

AccountID	DateTime	BankID	…
.	07MAY18:15:35:02	.	…
1010159565	16SEP18:14:57:08	101010101	…
1010183063	24FEB18:17:27:42	101010101	…
1010367330	15MAY18:17:54:21	101010101	…
1010367330	17OCT18:11:02:38	101010101	…

By default, SQL joins every row in the **smallcustomer** table with every row in the **smalltransaction**.
8 rows x **12 rows** = **96 rows**

3.01 Activity

Using **a subset of customers** table and **country_region_lookup** tables, do the following tasks to perform a default join of two tables:

1. Write a CREATE TABLE query to select Gold members from the Customers table named **Gold_Members**. WORK.Gold_Members should include the columns: **Customer_ID, Customer_Country, Customer_Name, Customer_BirthDate, Customer_Type** and rows: where Customer_Group contains the word “Gold”

```
proc sql;
create table Gold_Members as
select /*Complete the column names*/
from orion.customers
where /*Complete the where clause*/;
quit;
```

3.01 Activity

Using **Gold_Members** and **country_region_lookup** tables, do the following tasks to perform a default join of two tables:

2. Confirm that the **WORK.Gold_Members** table contains 21 rows and the **ORION.country_region_lookup** table contains 7 rows.
3. Next, join the two tables by selecting all columns and listing the **WORK.Gold_Members** and **ORION.country_region_lookup** table in the FROM clause and separate the tables by a comma. Run the query and view the log. What note do you see?
4. View the results. Name two issues with the report.

continued...

3.01 Activity – Correct Answer

3. What note do you see?

```
NOTE: The execution of this query involves performing one or more Cartesian product
joins that can not be optimized.
```

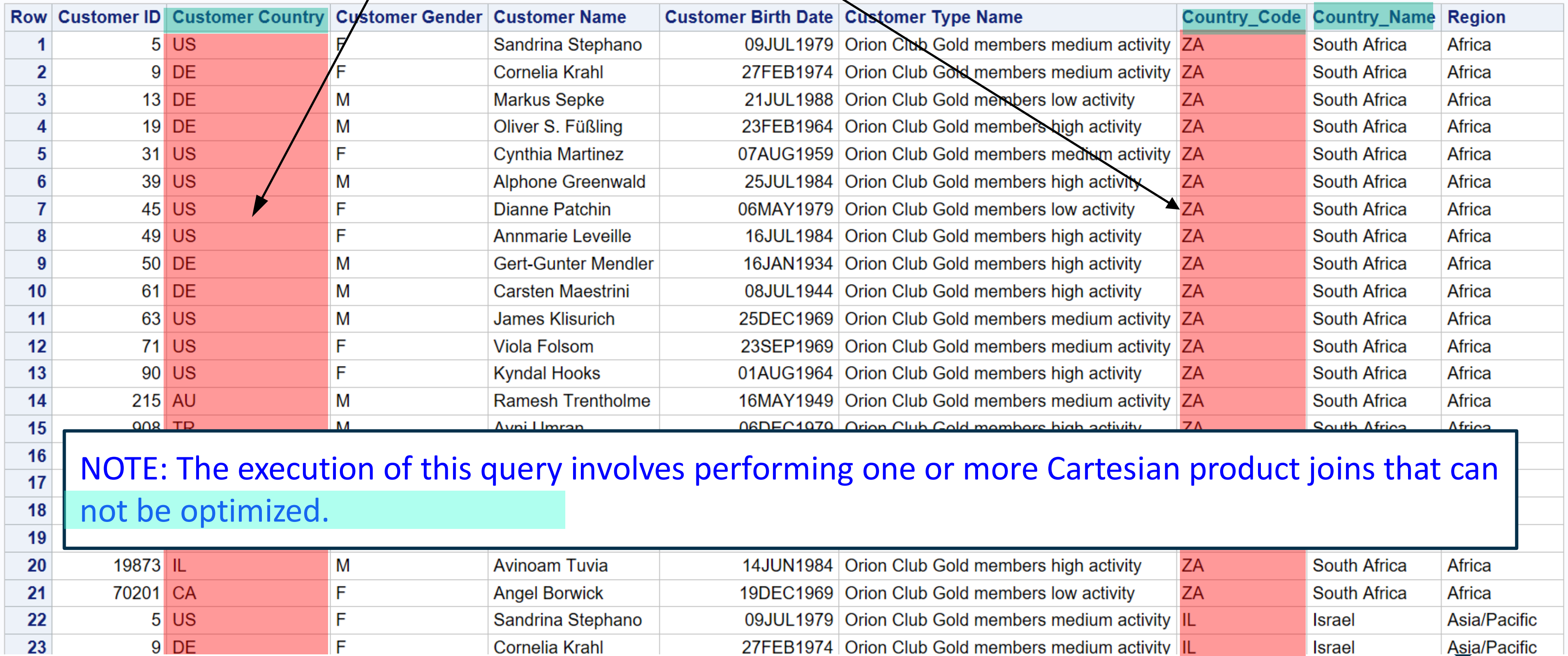
The default JOIN combines every row in each table. This is called the *Cartesian product*. Typically, the Cartesian product is not the desired result.

3.01 Activity – Correct Answer

Nonmatching IDs

Redundant columns

Partial



Row	Customer ID	Customer Country	Customer Gender	Customer Name	Customer Birth Date	Customer Type Name	Country_Code	Country_Name	Region
1	5	US	F	Sandrina Stephano	09JUL1979	Orion Club Gold members medium activity	ZA	South Africa	Africa
2	9	DE	F	Cornelia Krahl	27FEB1974	Orion Club Gold members medium activity	ZA	South Africa	Africa
3	13	DE	M	Markus Sepke	21JUL1988	Orion Club Gold members low activity	ZA	South Africa	Africa
4	19	DE	M	Oliver S. Füßling	23FEB1964	Orion Club Gold members high activity	ZA	South Africa	Africa
5	31	US	F	Cynthia Martinez	07AUG1959	Orion Club Gold members medium activity	ZA	South Africa	Africa
6	39	US	M	Alphone Greenwald	25JUL1984	Orion Club Gold members high activity	ZA	South Africa	Africa
7	45	US	F	Dianne Patchin	06MAY1979	Orion Club Gold members low activity	ZA	South Africa	Africa
8	49	US	F	Annmarie Leveille	16JUL1984	Orion Club Gold members high activity	ZA	South Africa	Africa
9	50	DE	M	Gert-Gunter Mendler	16JAN1934	Orion Club Gold members high activity	ZA	South Africa	Africa
10	61	DE	M	Carsten Maestrini	08JUL1944	Orion Club Gold members high activity	ZA	South Africa	Africa
11	63	US	M	James Klisurich	25DEC1969	Orion Club Gold members medium activity	ZA	South Africa	Africa
12	71	US	F	Viola Folsom	23SEP1969	Orion Club Gold members medium activity	ZA	South Africa	Africa
13	90	US	F	Kyndal Hooks	01AUG1964	Orion Club Gold members high activity	ZA	South Africa	Africa
14	215	AU	M	Ramesh Trentholme	16MAY1949	Orion Club Gold members medium activity	ZA	South Africa	Africa
15	908	TR	M	Avni Umran	06DEC1979	Orion Club Gold members high activity	ZA	South Africa	Africa
16									
17									
18									
19									
20	19873	IL	M	Avinoam Tuvia	14JUN1984	Orion Club Gold members high activity	ZA	South Africa	Africa
21	70201	CA	F	Angel Borwick	19DEC1969	Orion Club Gold members low activity	ZA	South Africa	Africa
22	5	US	F	Sandrina Stephano	09JUL1979	Orion Club Gold members medium activity	IL	Israel	Asia/Pacific
23	9	DE	F	Cornelia Krahl	27FEB1974	Orion Club Gold members medium activity	IL	Israel	Asia/Pacific

NOTE: The execution of this query involves performing one or more Cartesian product joins that can not be optimized.

Types of Joins

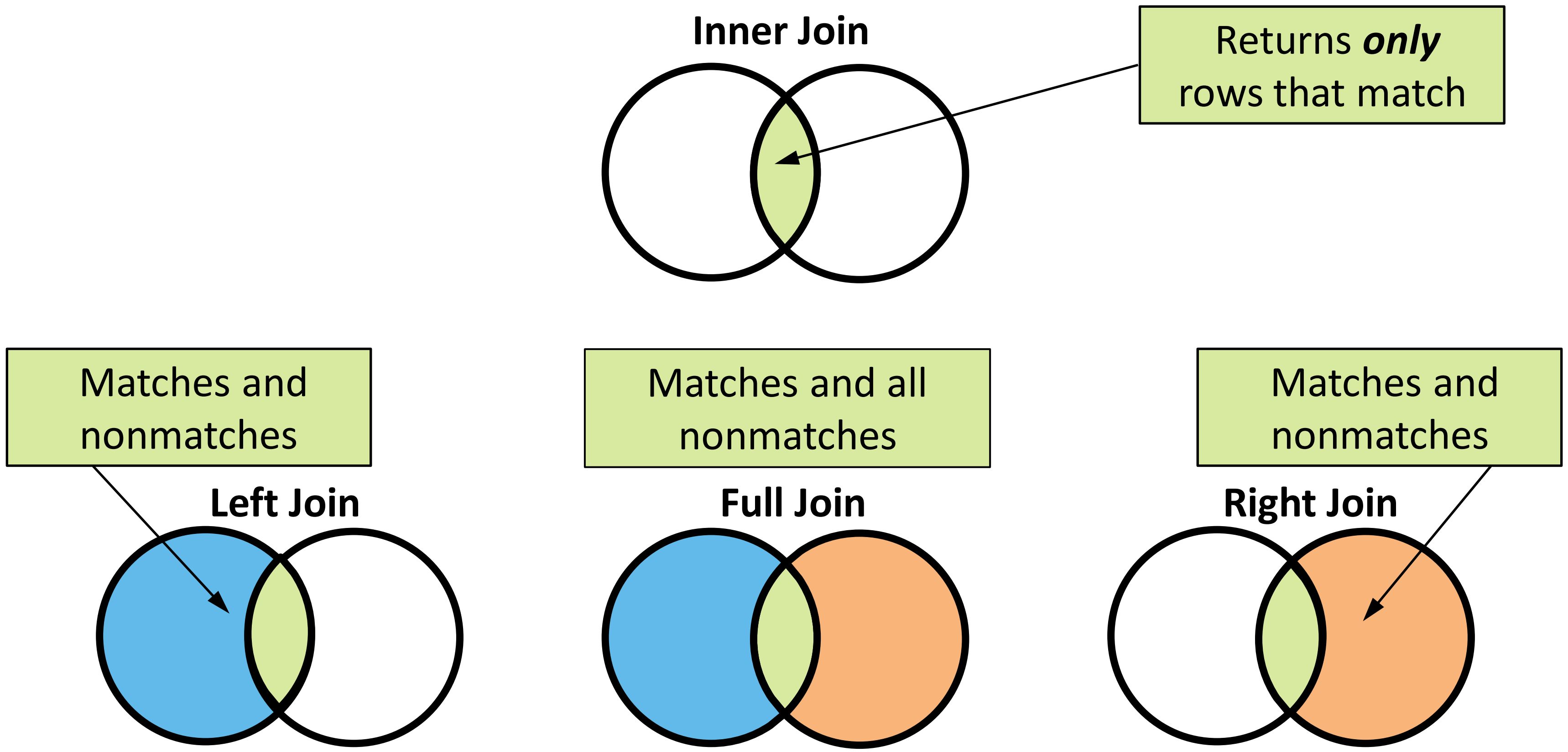
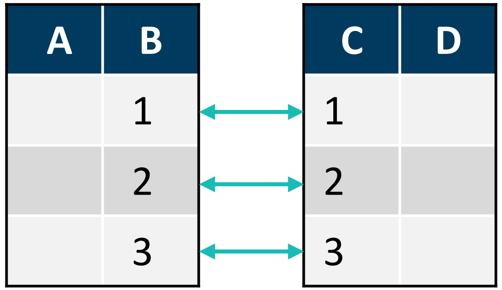
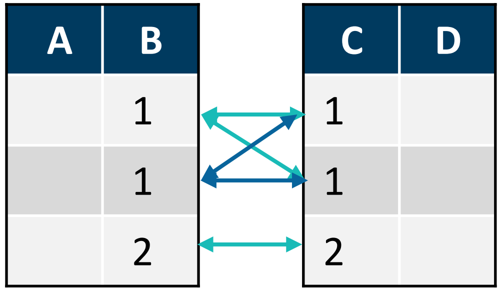


Table Relationships

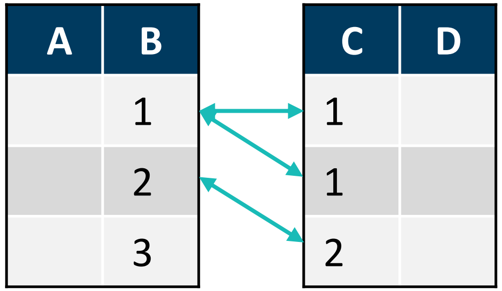
One-to-One



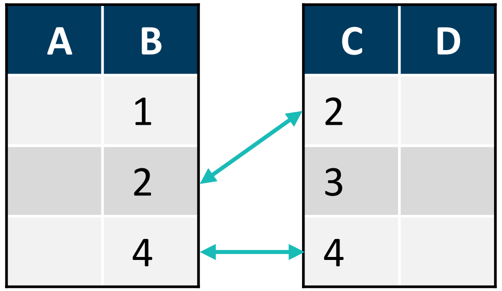
Many-to-Many



One-to-Many



Nonmatches



Lesson 3: SQL Joins

Scenario

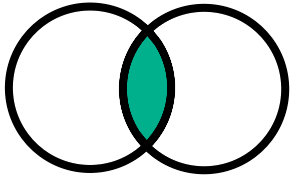
Report showing ***only*** customers with their ***matching*** transaction

SQL Inner Join Syntax

SELECT *col-name, col-name*
    FROM *table1* **INNER JOIN** *table2*
    ON *table1.column* **=** *table2.column***;**

```
proc sql;
select FirstName, LastName, State, Income, DateTime, Amount
    from sq.smallcustomer inner join sq.smalltransaction
    on smallcustomer.AccountID = smalltransaction.AccountID;
quit;
```

Specify the join type in the FROM clause.



s103d01

SQL Inner Join Syntax

SELECT *col-name, col-name*
FROM *table1* **INNER JOIN** *table2*
ON *table1.column* **=** *table2.column***;**

```
proc sql;
select FirstName, LastName, State, Income, DateTime, Amount
    from sq.smallcustomer inner join sq.smalltransaction
    on smallcustomer.AccountID = smalltransaction.AccountID;
quit;
```

Specify the join criteria in the ON clause. You can use other comparison operators, such as the **greater than**, **less than**, or **special where** operators.

s103d01

SQL Inner Join Syntax

SELECT *col-name, col-name*
    FROM *table1* **INNER JOIN** *table2*
    ON *table1.column* **=** *table2.column***;**

```
proc sql;
select FirstName, LastName, State, Income, DateTime, Amount
    from sq.smallcustomer inner join sq.smalltransaction
    on smallcustomer.AccountID = smalltransaction.AccountID;
quit;
```

Qualify the column names to specify the location of each column.

s103d01

Alternative SQL Inner Join Syntax

SELECT *col-name, col-name*
FROM *table1, table2*
WHERE *table1.column* **=** *table2.column***;**

```
proc sql;
select FirstName, LastName, State, Income, DateTime, Amount
    from sq.smallcustomer, sq.smalltransaction
    where smallcustomer.AccountID = smalltransaction.AccountID;
quit;
```

List the tables in the FROM clause.

List the join criteria in the WHERE clause.

Using Table Aliases

FROM *table1* <**AS**> *alias1*, *table2* <**AS**> *alias2*

```
proc sql;
select FirstName, LastName, State, Income, DateTime, c.AccountID
    from sq.smallcustomer as c inner join
         sq.smalltransaction as t
    on c.AccountID = t.AccountID;
quit;
```

Assign an alias (or nickname) to a table in the FROM clause by adding the keyword AS.

3.02 Activity

Join **WORK.Gold_Members** and **ORION.country_region_lookup** tables to find out where are the gold members from. Fix the issues resulted by the Cartesian product join (recall Activity 3.01). Perform *an inner join*:

1. Create a table called Gold_Memeber_Countries
2. Select all columns except *Customer_Country* from **Gold_Members** table and *Country_Name* from **country_region_lookup** table
3. Specify the tables in the FROM clause and perform an inner join. Add the alias **g** for the **WORK.Gold_Members** table, and the alias **c** for the **ORION.country_region_lookup** table.
4. Complete the ON expression to match rows where **g.Customer_Country** = **c.Country_Code**.
5. Highlight and run the query. How many rows are in the new report?

continued...

3.02 Activity – Correct Answer

1. Create a table called Gold_Memeber_Countries

```
proc sql;
create table Gold_Member_Countries as
select Customer_ID, Country_Name, Region, Customer_Gender,
       Customer_Name, Customer_BirthDate, Customer_Type
    from work.Gold_Members as g inner join
                                  orion.country_region_lookup as c
    on g.Customer_Country = c.Country_Code;
quit;
```

continued...

3.02 Activity – Correct Answer

2. Select all columns except *Customer_Country* from **Gold_Members** table and *Country_Name* from **country_region_lookup** table

```
proc sql;
create table Gold_Member_Countries as
select Customer_ID, Country_Name, Region, Customer_Gender,
       Customer_Name, Customer_BirthDate, Customer_Type
    from work.Gold_Members as g inner join
                                        orion.country_region_lookup as c
    on g.Customer_Country = c.Country_Code;
quit;
```

continued...

3.02 Activity – Correct Answer

3. Add the alias **g** for the **WORK.Gold_Members** table, and the alias **c** for the **ORION.country_region_lookup** table

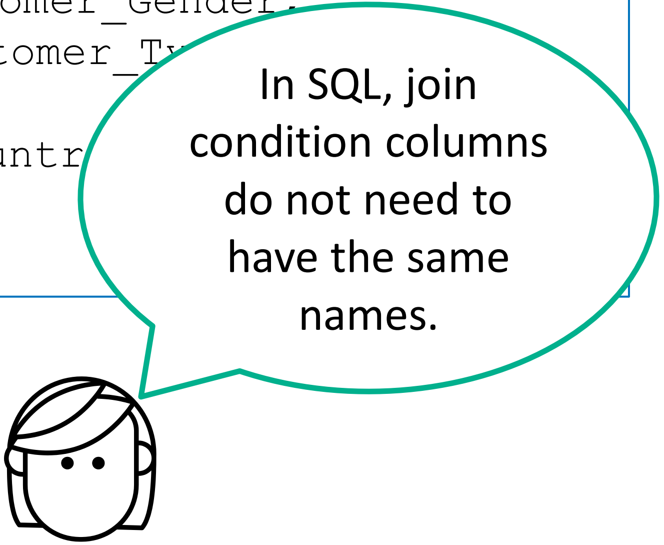
```
proc sql;
create table Gold_Member_Countries as
select Customer_ID, Country_Name, Region, Customer_Gender,
       Customer_Name, Customer_BirthDate, Customer_Type
    from work.Gold_Members as g inner join
                                        orion.country_region_lookup as c
    on g.Customer_Country = c.Country_Code;
quit;
```

continued...

3.02 Activity – Correct Answer

4. Complete the ON expression to match rows where **g.Customer_Country = c.Country_Code**.

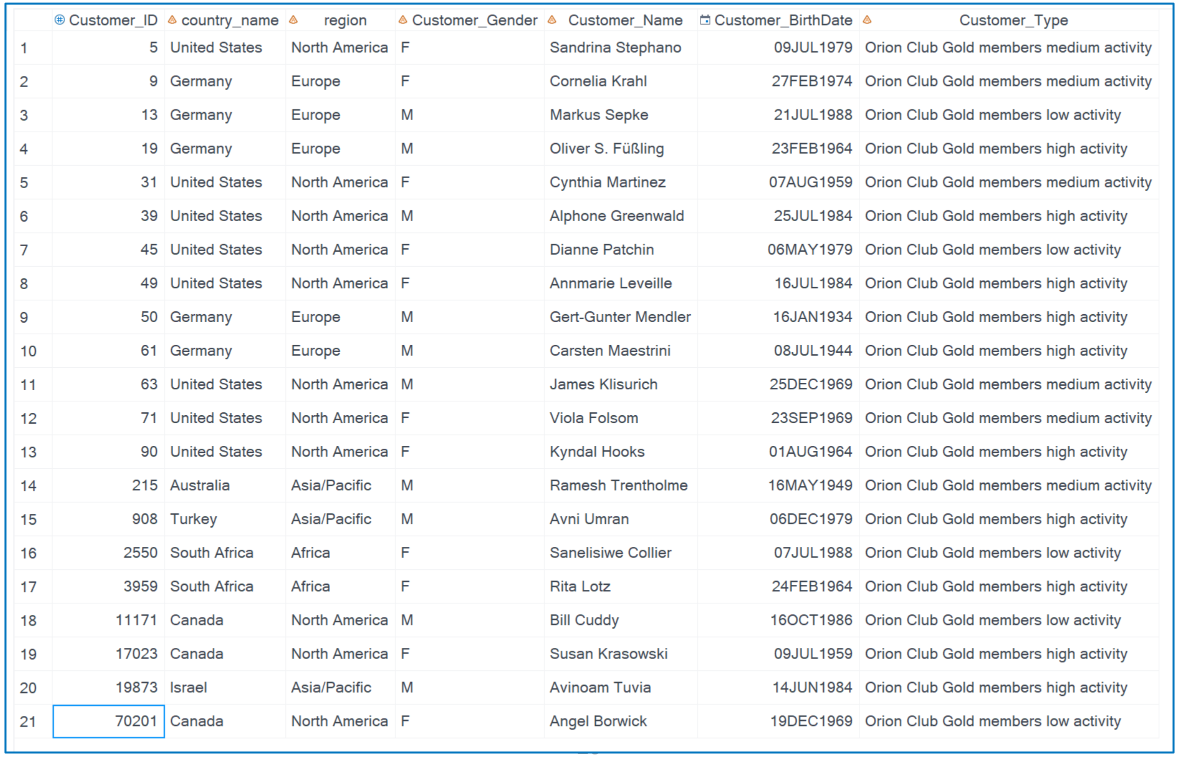
```
proc sql;
create table Gold_Member_Countries as
select Customer_ID, Country_Name, Region, Customer_Gender,
       Customer_Name, Customer_BirthDate, Customer_Ty
    from work.Gold_Members as g inner join
                                   orion.countr
    on g.Customer_Country = c.Country_Code;
quit;
```


In SQL, join condition columns do not need to have the same names.

continued...

3.02 Activity – Correct Answer

5. Highlight and run the query. How many rows are in the new report? **21**

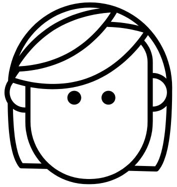


	Customer_ID	country_name	region	Customer_Gender	Customer_Name	Customer_BirthDate	Customer_Type
1	5	United States	North America	F	Sandrina Stephano	09JUL1979	Orion Club Gold members medium activity
2	9	Germany	Europe	F	Cornelia Krahl	27FEB1974	Orion Club Gold members medium activity
3	13	Germany	Europe	M	Markus Sepke	21JUL1988	Orion Club Gold members low activity
4	19	Germany	Europe	M	Oliver S. Füßling	23FEB1964	Orion Club Gold members high activity
5	31	United States	North America	F	Cynthia Martinez	07AUG1959	Orion Club Gold members medium activity
6	39	United States	North America	M	Alphone Greenwald	25JUL1984	Orion Club Gold members high activity
7	45	United States	North America	F	Dianne Patchin	06MAY1979	Orion Club Gold members low activity
8	49	United States	North America	F	Annmarie Leveille	16JUL1984	Orion Club Gold members high activity
9	50	Germany	Europe	M	Gert-Gunter Mendler	16JAN1934	Orion Club Gold members high activity
10	61	Germany	Europe	M	Carsten Maestrini	08JUL1944	Orion Club Gold members high activity
11	63	United States	North America	M	James Klisurich	25DEC1969	Orion Club Gold members medium activity
12	71	United States	North America	F	Viola Folsom	23SEP1969	Orion Club Gold members medium activity
13	90	United States	North America	F	Kyndal Hooks	01AUG1964	Orion Club Gold members high activity
14	215	Australia	Asia/Pacific	M	Ramesh Trentholme	16MAY1949	Orion Club Gold members medium activity
15	908	Turkey	Asia/Pacific	M	Avni Umran	06DEC1979	Orion Club Gold members high activity
16	2550	South Africa	Africa	F	Sanelisiwe Collier	07JUL1988	Orion Club Gold members low activity
17	3959	South Africa	Africa	F	Rita Lotz	24FEB1964	Orion Club Gold members high activity
18	11171	Canada	North America	M	Bill Cuddy	16OCT1986	Orion Club Gold members low activity
19	17023	Canada	North America	F	Susan Krasowski	09JUL1959	Orion Club Gold members high activity
20	19873	Israel	Asia/Pacific	M	Avinoam Tuvia	14JUN1984	Orion Club Gold members high activity
21	70201	Canada	North America	F	Angel Borwick	19DEC1969	Orion Club Gold members low activity

Matching Rows with a Natural Join

SELECT *col-name, col-name*
FROM *table1* **NATURAL JOIN** *table2*

```
proc sql;
select *
    from sq.smallcustomer as c natural join
         sq.smalltransaction as t;
quit;
```



A ***natural join*** assumes that you want to base the join on all pairs of ***common columns***.

FEEDBACK Option

PROC SQL FEEDBACK;

```
proc sql feedback;
select *
    from sq.smallcustomer as c natural join
         sq.smalltransaction as t;
quit;
```

The FEEDBACK option expands a SELECT statement to the SAS log.

```
NOTE: Statement transforms to:

     select COALESCE(T.AccountID, C.AccountID) as AccountID, COALESCE(T.BankID, C.BankID) as
BankID, T.DateTime...
```

Selecting Data from More Than Two Tables

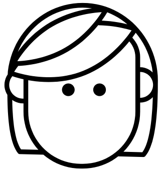
Results

FirstName	LastName	State	Income	DateTime	MerchantID	Amount	AccountID	BankID
Gary	Sienkiewicz	NY	67210.91	16SEP18:14:57:08	568268	107.16	1010159565	101010101
Sergio	Lefeld	CA	86859.07	15MAY18:17:54:21	542058	23.39	1010367330	101010101
Sergio	Lefeld	CA	86859.07	17OCT18:11:02:38	525576	21.02	1010367330	101010101
John	Oliver	CA	43623.75	23FEB18:09:25:37	525576	108.22	2020012887	202020202
Iva	Bower	NY	67949.96	27JUL18:12:05:48	525576	26.1	3030085224	303030303
Janet	Sienkiewicz	NY	50111.59	18SEP18:12:13:40	549940	37.38	3030101942	303030303
		NY	31896.96	11MAR18:10:07:14	580881	319.95	3030165207	303030303

Merchant Name

Bank Name

How can I retrieve the ***merchant name*** and the ***bank name*** in the results?



3.03 Activity

Performing an Inner Join with Three Tables: From which countries are the top products supplied? Use **topproducts**, **products** *(import this dataset first)* and **country_lookup** tables:

1. Import the product list as **products** SAS dataset from products.xls file.
2. Complete the query on the next page to create Top_Supplier_Countries by joining three tables. Include all rows from the **topproducts** table (alias t), and matching rows from the **products** (alias p) and **country_lookup** (alias c) tables.

3.03 Activity

Performing an Inner Join with Three Tables: From which countries are the top products supplied? Use **topproducts**, **products** *(import this dataset first)* and **country_lookup** tables:

```
proc sql;
create table Top_product_countries as
select t.Product_ID, t.Sum_of_Profit, c.Country_Name
    from /*Complete the query: topproducts joined with
products joined with country_lookup*/;
quit;
```

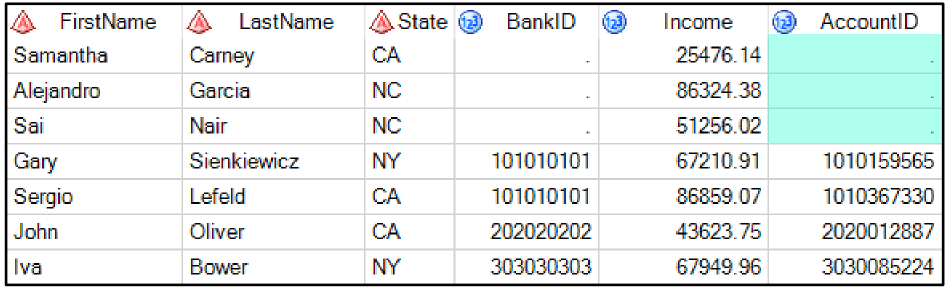
3.03 Activity – Correct Answer

Performing an Inner Join with Three Tables: From which countries are the top products supplied? Use **topproducts**, **products** *(import this dataset first)* and **country_lookup** tables:

```
proc sql;
create table Top_product_countries as
select t.Product_ID, t.Sum_of_Profit, c.Country_Name
    from orion.topproducts as t
       inner join work.products as p
                                  on t.Product_ID = p.Product_ID
       inner join orion.country_lookup as c
                          on p.Supplier_Country = c.Country_Key;
quit;
```

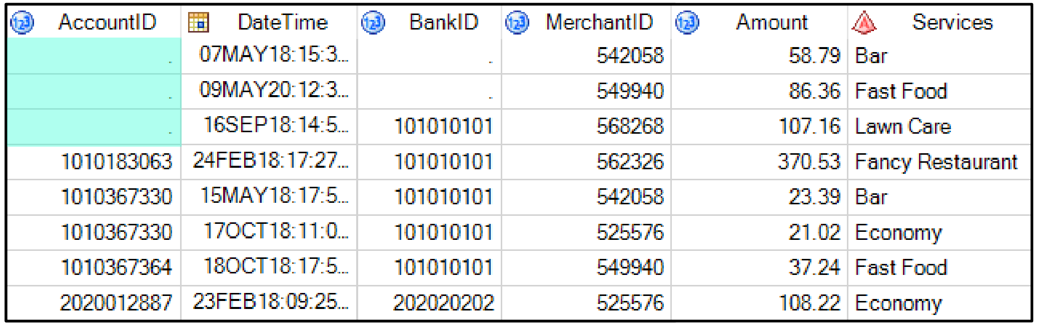
Effects of Missing Values on Joins

smallcustomer2 Partial



FirstName	LastName	State	BankID	Income	AccountID
Samantha	Carney	CA	.	25476.14	.
Alejandro	Garcia	NC	.	86324.38	.
Sai	Nair	NC	.	51256.02	.
Gary	Sienkiewicz	NY	101010101	67210.91	1010159565
Sergio	Lefeld	CA	101010101	86859.07	1010367330
John	Oliver	CA	202020202	43623.75	2020012887
Iva	Bower	NY	303030303	67949.96	3030085224

smalltransaction2 Partial



AccountID	DateTime	BankID	MerchantID	Amount	Services
.	07MAY18:15:3...	.	542058	58.79	Bar
.	09MAY20:12:3...	.	549940	86.36	Fast Food
.	16SEP18:14:5...	101010101	568268	107.16	Lawn Care
1010183063	24FEB18:17:27...	101010101	562326	370.53	Fancy Restaurant
1010367330	15MAY18:17:5...	101010101	542058	23.39	Bar
1010367330	17OCT18:11:0...	101010101	525576	21.02	Economy
1010367364	18OCT18:17:5...	101010101	549940	37.24	Fast Food
2020012887	23FEB18:09:25...	202020202	525576	108.22	Economy

PROC SQL treats *missing* values as *matches* for joins.

Effects of Missing Values on Joins

FirstName	LastName	State	BankID	Income	AccountID	AccountID	DateTime	BankID	MerchantID	Amount	Services
Samantha	Carney	CA	.	25476.14	.	.	07MAY18:15:35:02	.	542058	58.79	Bar
Sai	Nair	NC	.	51256.02	.	.	07MAY18:15:35:02	.	542058	58.79	Bar
Alejandro	Garcia	NC	.	86324.38	.	.	07MAY18:15:35:02	.	542058	58.79	Bar
Samantha	Carney	CA	.	25476.14	.	.	09MAY20:12:30:08	.	549940	86.36	Fast Food
Sai	Nair	NC	.	51256.02	.	.	09MAY20:12:30:08	.	549940	86.36	Fast Food
Alejandro	Garcia	NC	.	86324.38	.	.	09MAY20:12:30:08	.	549940	86.36	Fast Food
Samantha	Carney	CA	.	25476.14	.	.	16SEP18:14:57:08	101010101	568268	107.16	Lawn Care
Sai	Nair	NC	.	51256.02	.	.	16SEP18:14:57:08	101010101	568268	107.16	Lawn Care
Alejandro	Garcia	NC	.	86324.38	.	.	16SEP18:14:57:08	101010101	568268	107.16	Lawn Care
Sergio	Lefeld	CA	101010101	86859.07	1010367330	1010367330	15MAY18:17:54:21	101010101	542058	23.39	Bar
Sergio	Lefeld	CA	101010101	86859.07	1010367330	1010367330	17OCT18:11:02:38	101010101	525576	21.02	Economy

Missing values are joined, and this typically is not the desired result.

Effects of Missing Values on Joins

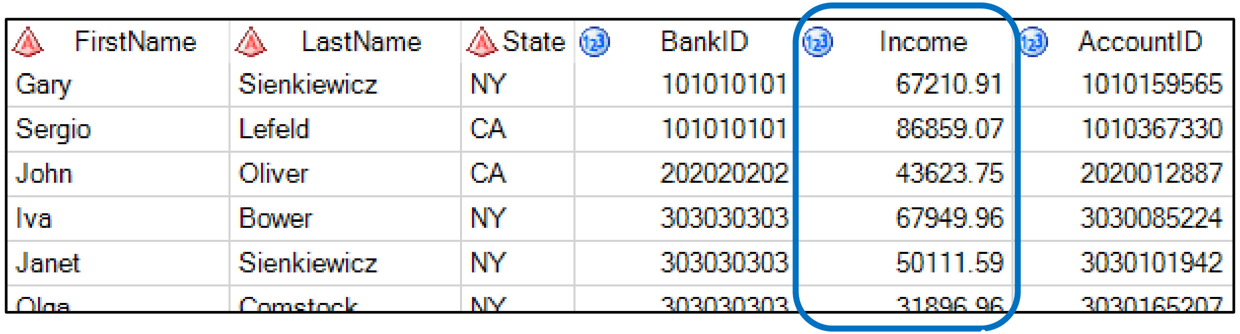
```
proc sql;
   select *
   from sq.smallcustomer2 as c inner join
        sq.smalltransaction2 as t
   on c.AccountID = t.AccountID and
      c.AccountID is not null;
quit;
```

Adding the IS NOT NULL operator to the ON clause prevents the missing values from joining.

FirstName	LastName	State	BankID	Income	AccountID	AccountID
Sergio	Lefeld	CA	101010101	86859.07	1010367330	1010367330
Sergio	Lefeld	CA	101010101	86859.07	1010367330	1010367330
John	Oliver	CA	202020202	43623.75	2020012887	2020012887
Iva	Bower	NY	303030303	67949.96	3030085224	3030085224
Janet	Sienkiewicz	NY	303030303	50111.59	3030101942	3030101942
Olga	Comstock	NY	303030303	31896.96	3030165207	3030165207

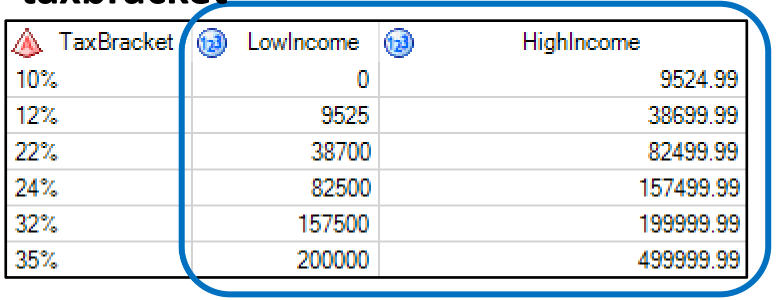
Non-Equijoin

smallcustomer

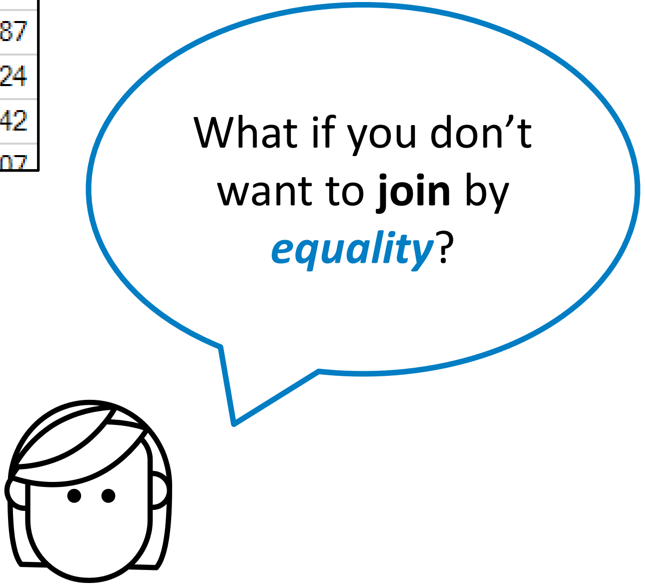


FirstName	LastName	State	BankID	Income	AccountID
Gary	Sienkiewicz	NY	101010101	67210.91	1010159565
Sergio	Lefeld	CA	101010101	86859.07	1010367330
John	Oliver	CA	202020202	43623.75	2020012887
Iva	Bower	NY	303030303	67949.96	3030085224
Janet	Sienkiewicz	NY	303030303	50111.59	3030101942
Olga	Comstock	NY	303030303	31896.96	3030165207

taxbracket



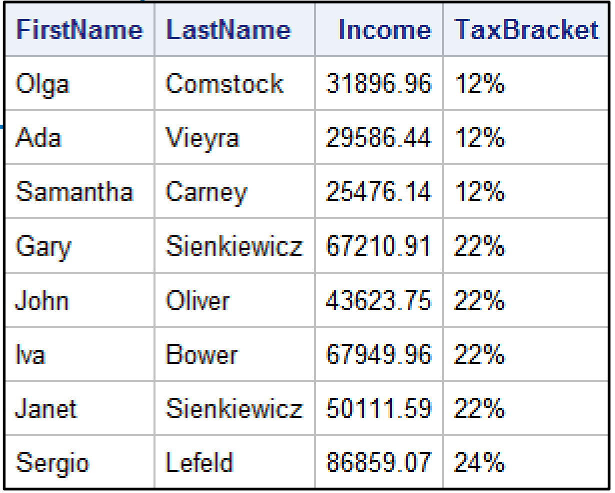
TaxBracket	LowIncome	HighIncome
10%	0	9524.99
12%	9525	38699.99
22%	38700	82499.99
24%	82500	157499.99
32%	157500	199999.99
35%	200000	499999.99



Non-Equijoin

```
select FirstName, LastName, Income,
       TaxBracket
    from sq.smallcustomer as c inner join
         sq.taxbracket as t
    on c.Income >= t.LowIncome and
       c.Income <= t.HighIncome;
```

Use comparison operators in the ON clause instead of equality.



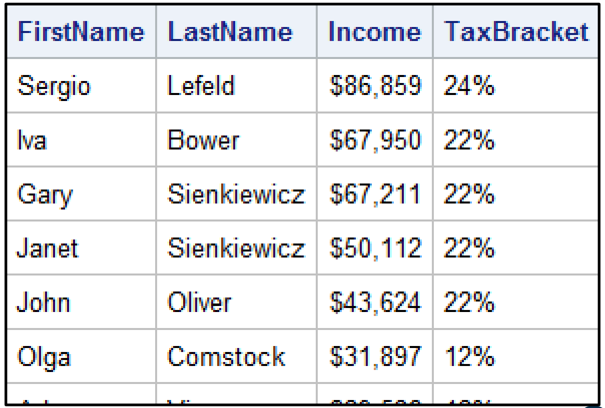
FirstName	LastName	Income	TaxBracket
Olga	Comstock	31896.96	12%
Ada	Vieyra	29586.44	12%
Samantha	Carney	25476.14	12%
Gary	Sienkiewicz	67210.91	22%
John	Oliver	43623.75	22%
Iva	Bower	67949.96	22%
Janet	Sienkiewicz	50111.59	22%
Sergio	Lefeld	86859.07	24%

continued...

Alternative to Non-Equijoin

```
select FirstName, LastName,
       Income format=dollar16., TaxBracket
    from sq.smallcustomer as c inner join
         sq.taxbracket as t
    on c.Income between t.LowIncome and t.HighIncome
    order by TaxBracket desc, Income desc;
```

Use the BETWEEN-AND special where operator.



FirstName	LastName	Income	TaxBracket
Sergio	Lefeld	$86,859	24%
Iva	Bower	$67,950	22%
Gary	Sienkiewicz	$67,211	22%
Janet	Sienkiewicz	$50,112	22%
John	Oliver	$43,624	22%
Olga	Comstock	$31,897	12%

Non-Equijoin

What note do you see?

```
select FirstName, LastName,
       Income format=dollar16., TaxBracket
    from sq.smallcustomer as c inner join
         sq.taxbracket as t
    on c.Income between t.LowIncome and t.HighIncome
    order by TaxBracket desc, Income desc;
```

```
NOTE: The execution of this query involves performing one or more Cartesian product
joins that can not be optimized.
```

Syntax Summary

SELECT *col-name, col-name*
    FROM *table1* **INNER JOIN** *table2*
    ON *table1.column* **=** *table2.column***;**

Inner Join

SELECT *col-name, col-name*
    FROM *table1* **INNER JOIN** *table2*
    ON *table1.column* **=** *table2.column*
    INNER JOIN *table3*
    ON *join criteria***;**

Joining More Than Two Tables

ON *table1.column* **<** *table2.column* **AND**
    table1.column **>** *table2.column***;**

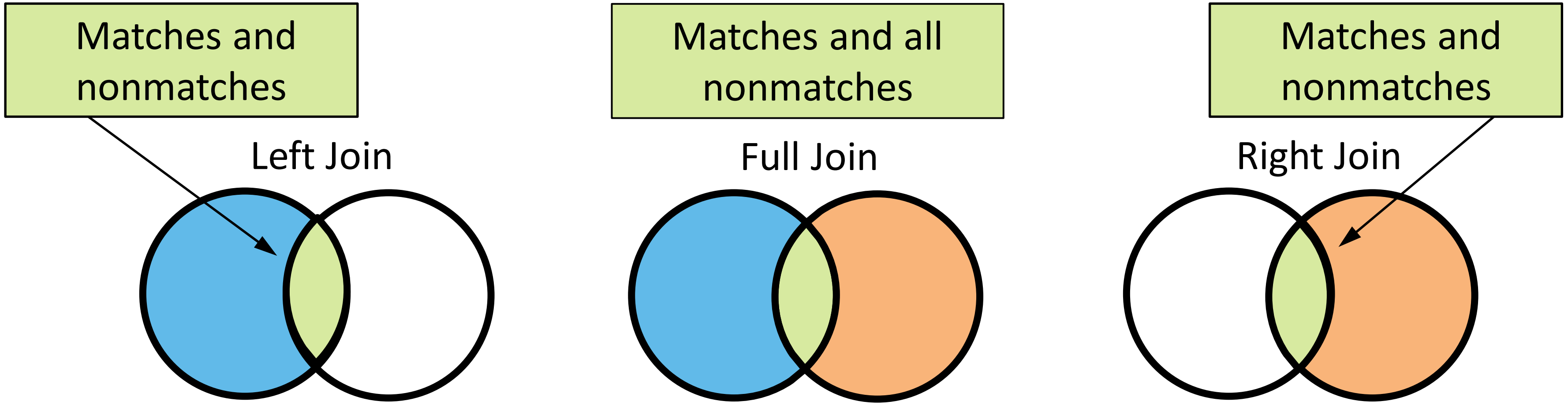
Non-equijoin

PROC SQL FEEDBACK;

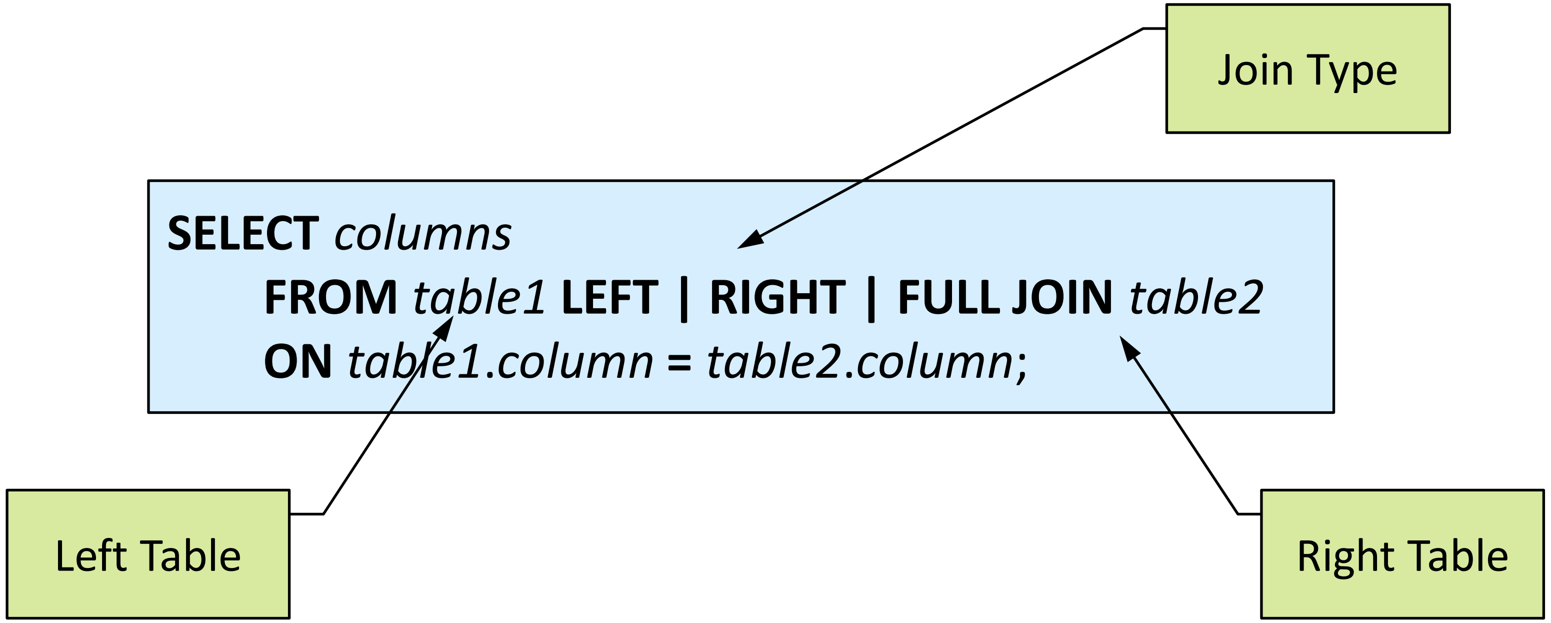
PROC SQL Options

Lesson 3: SQL Joins

SQL Outer Joins



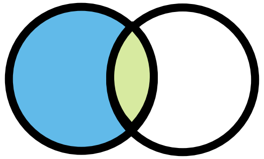
SQL Outer Join Syntax



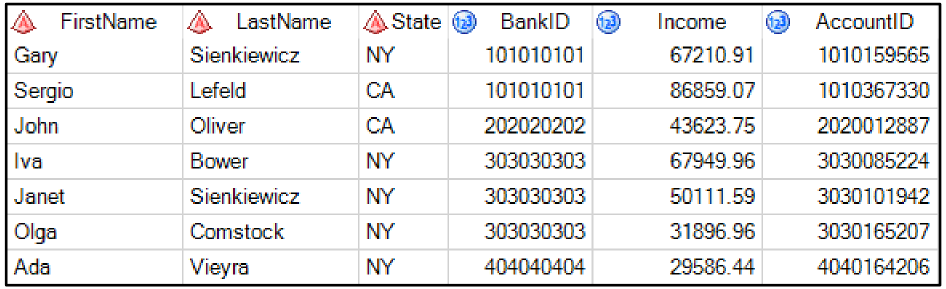
```
SELECT columns
       FROM table1 LEFT | RIGHT | FULL JOIN table2
       ON table1.column = table2.column;
```

Scenario

Left Join

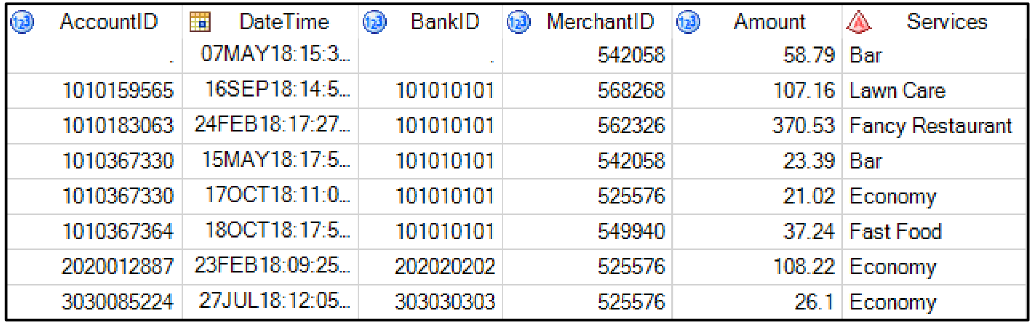


smallcustomer Partial



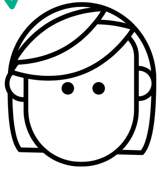
FirstName	LastName	State	BankID	Income	AccountID
Gary	Sienkiewicz	NY	101010101	67210.91	1010159565
Sergio	Lefeld	CA	101010101	86859.07	1010367330
John	Oliver	CA	202020202	43623.75	2020012887
Iva	Bower	NY	303030303	67949.96	3030085224
Janet	Sienkiewicz	NY	303030303	50111.59	3030101942
Olga	Comstock	NY	303030303	31896.96	3030165207
Ada	Vieyra	NY	404040404	29586.44	4040164206

smalltransaction Partial



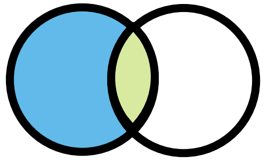
AccountID	DateTime	BankID	MerchantID	Amount	Services
.	07MAY18:15:3...	.	542058	58.79	Bar
1010159565	16SEP18:14:5...	101010101	568268	107.16	Lawn Care
1010183063	24FEB18:17:27...	101010101	562326	370.53	Fancy Restaurant
1010367330	15MAY18:17:5...	101010101	542058	23.39	Bar
1010367330	17OCT18:11:0...	101010101	525576	21.02	Economy
1010367364	18OCT18:17:5...	101010101	549940	37.24	Fast Food
2020012887	23FEB18:09:25...	202020202	525576	108.22	Economy
3030085224	27JUL18:12:05...	303030303	525576	26.1	Economy

Report of ***all*** customers with or without a transaction

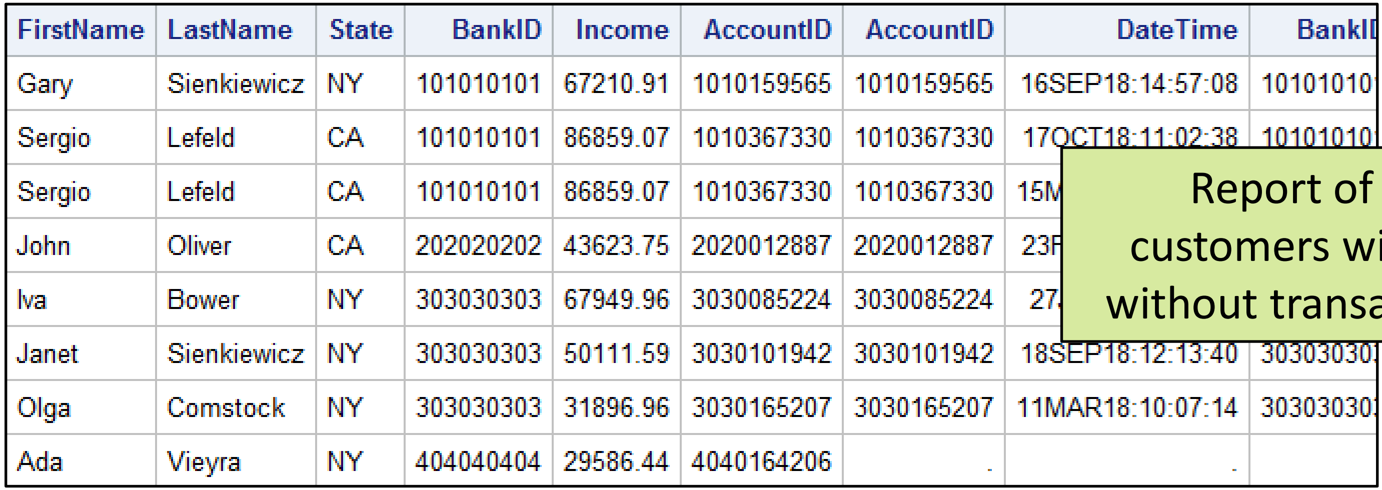


SQL Left Join Syntax

Left Join



```
select *
    from sq.smallcustomer as c left join
         sq.smalltransaction as t
    on c.AccountID = t.AccountID;
```

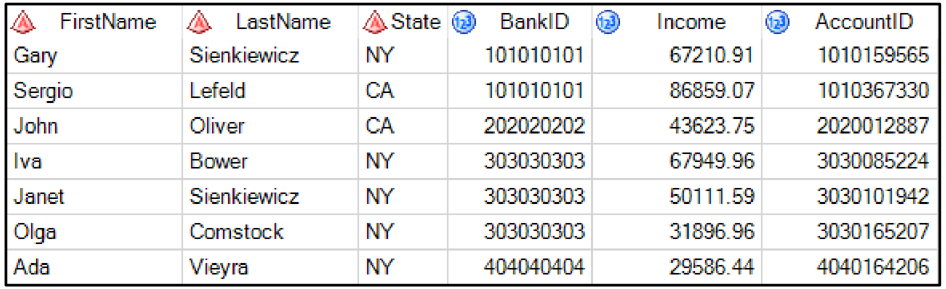


FirstName	LastName	State	BankID	Income	AccountID	AccountID	DateTime	BankI
Gary	Sienkiewicz	NY	101010101	67210.91	1010159565	1010159565	16SEP18:14:57:08	10101010
Sergio	Lefeld	CA	101010101	86859.07	1010367330	1010367330	17OCT18:11:02:38	10101010
Sergio	Lefeld	CA	101010101	86859.07	1010367330	1010367330	15M	
John	Oliver	CA	202020202	43623.75	2020012887	2020012887	23F	
Iva	Bower	NY	303030303	67949.96	3030085224	3030085224	27.	
Janet	Sienkiewicz	NY	303030303	50111.59	3030101942	3030101942	18SEP18:12:13:40	30303030
Olga	Comstock	NY	303030303	31896.96	3030165207	3030165207	11MAR18:10:07:14	30303030
Ada	Vieyra	NY	404040404	29586.44	4040164206	.	.	

Report of ***all*** customers with or without transactions

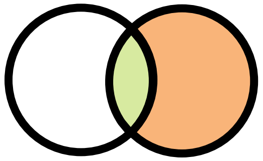
Scenario

smallcustomer Partial

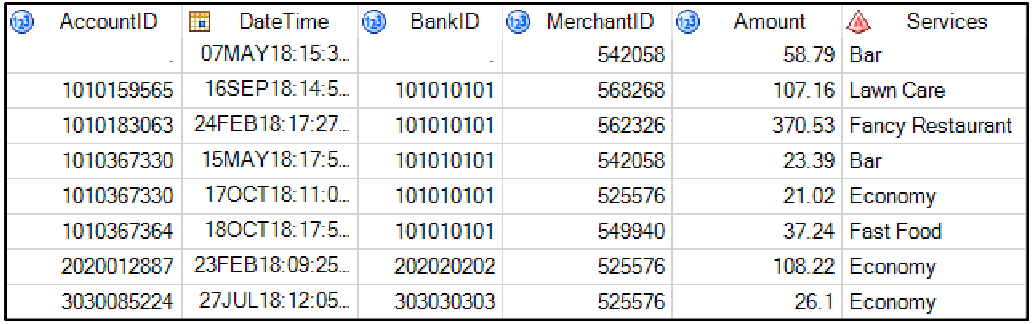


FirstName	LastName	State	BankID	Income	AccountID
Gary	Sienkiewicz	NY	101010101	67210.91	1010159565
Sergio	Lefeld	CA	101010101	86859.07	1010367330
John	Oliver	CA	202020202	43623.75	2020012887
Iva	Bower	NY	303030303	67949.96	3030085224
Janet	Sienkiewicz	NY	303030303	50111.59	3030101942
Olga	Comstock	NY	303030303	31896.96	3030165207
Ada	Vieyra	NY	404040404	29586.44	4040164206

Right Join



smalltransaction Partial

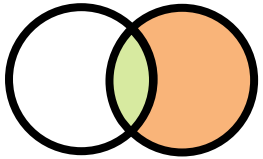


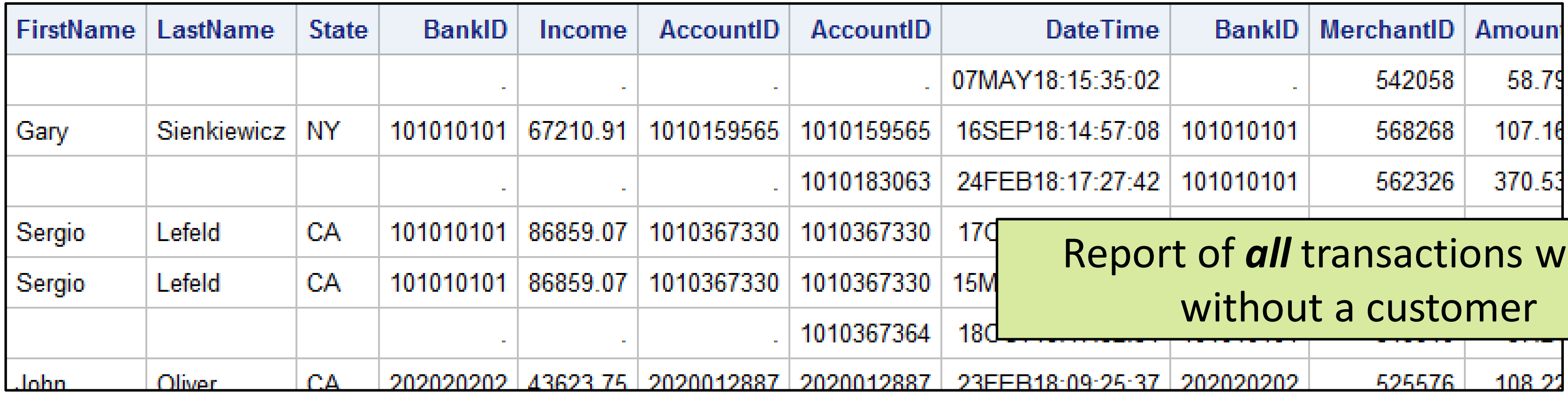
AccountID	DateTime	BankID	MerchantID	Amount	Services
.	07MAY18:15:3...	.	542058	58.79	Bar
1010159565	16SEP18:14:5...	101010101	568268	107.16	Lawn Care
1010183063	24FEB18:17:27...	101010101	562326	370.53	Fancy Restaurant
1010367330	15MAY18:17:5...	101010101	542058	23.39	Bar
1010367330	17OCT18:11:0...	101010101	525576	21.02	Economy
1010367364	18OCT18:17:5...	101010101	549940	37.24	Fast Food
2020012887	23FEB18:09:25...	202020202	525576	108.22	Economy
3030085224	27JUL18:12:05...	303030303	525576	26.1	Economy

SQL Right Join Syntax

```
select *
    from sq.smallcustomer as c right join
         sq.smalltransaction as t
    on c.AccountID = t.AccountID;
```

Right Join





FirstName	LastName	State	BankID	Income	AccountID	AccountID	DateTime	BankID	MerchantID	Amoun
			.	.	.	.	07MAY18:15:35:02	.	542058	58.79
Gary	Sienkiewicz	NY	101010101	67210.91	1010159565	1010159565	16SEP18:14:57:08	101010101	568268	107.16
			.	.	.	1010183063	24FEB18:17:27:42	101010101	562326	370.53
Sergio	Lefeld	CA	101010101	86859.07	1010367330	1010367330	17O[illegible]	[illegible]	[illegible]	[illegible]
Sergio	Lefeld	CA	101010101	86859.07	1010367330	1010367330	15M[illegible]	[illegible]	[illegible]	[illegible]
			.	.	.	1010367364	18O[illegible]	[illegible]	[illegible]	[illegible]
John	Oliver	CA	202020202	43623.75	2020012887	2020012887	23FEB18:09:25:37	202020202	525576	108.22

Report of ***all*** transactions with or without a customer

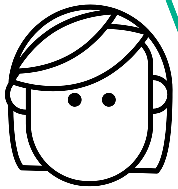
Selecting Columns in Inner Joins

c.AccountID

t.AccountID

```
proc sql;
select FirstName, LastName, Income, AccountID,
       DateTime, MerchantID, Amount
    from sq.smallcustomer as c inner join
         sq.smalltransaction as t
    on c.AccountID = t.AccountID;
quit;
```

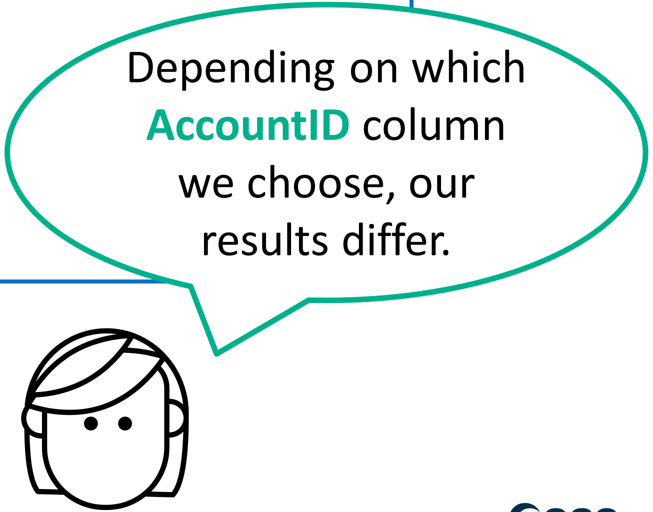
With an inner join, you can select either **AccountID** column.



Selecting Columns in Outer Joins

c.AccountID $\cancel{\neq}$ t.AccountID

```
proc sql;
select FirstName, LastName, Income, AccountID,
       DateTime, MerchantID, Amount
    from sq.smallcustomer as c left join
         sq.smalltransaction as t
    on c.AccountID = t.AccountID;
quit;
```


Depending on which AccountID column we choose, our results differ.

3.04 Activity

Do the following tasks to find information on the *employees who donated*:

1. Run the query and identify the problem.
2. Add **a.** in front of Employee_ID in the SELECT clause. Run the query and examine the results.
3. Replace **a.Employee_ID** with **d.Employee_ID** in the SELECT clause. Run the query and examine the results.

```
proc sql number;
select Employee_ID, Employee_Name, Postal_Code,
Recipients, Paid_By
    from orion.employee_addresses as a full join
orion.employee_donations as d
    on a.Employee_ID = d.Employee_ID;
quit;
```

3.04 Activity – Correct Answer

Do the following tasks to find information about the employees who donated:

4. Change the full join to **right join** to include all the rows in the donations table

```
proc sql number;
select d.Employee_ID, Employee_Name, Postal_Code,
Recipients, Paid_By
    from orion.employee_addresses as a right join
orion.employee_donations as d
    on a.Employee_ID = d.Employee_ID;
quit;
```

COALESCE Function

COALESCE(*argument-1, argument-2<, …argument-n>***)**

```
select coalesce(c.AccountID, t.AccountID) as AccountID
```

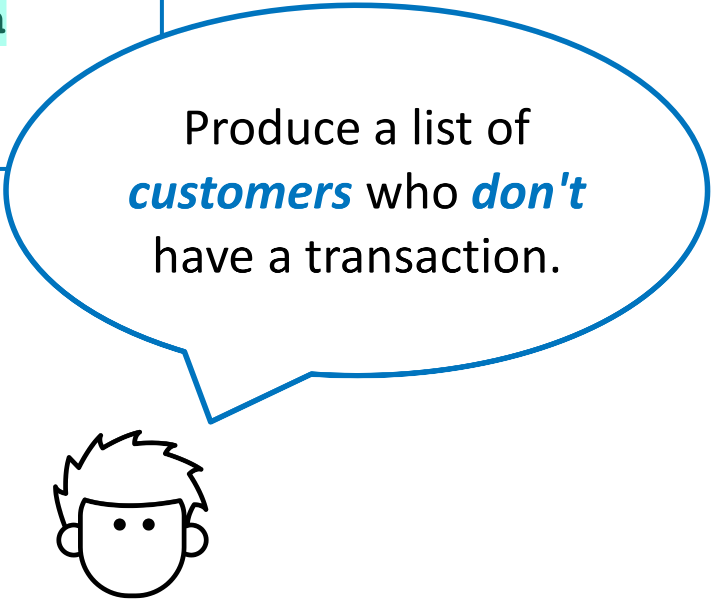
c.AccountID	t.AccountID
.	1010183063
4040164206	.
3030165207	3030165207

The COALESCE function returns the value of the ***first nonmissing argument***.

Identifying Nonmatching Rows

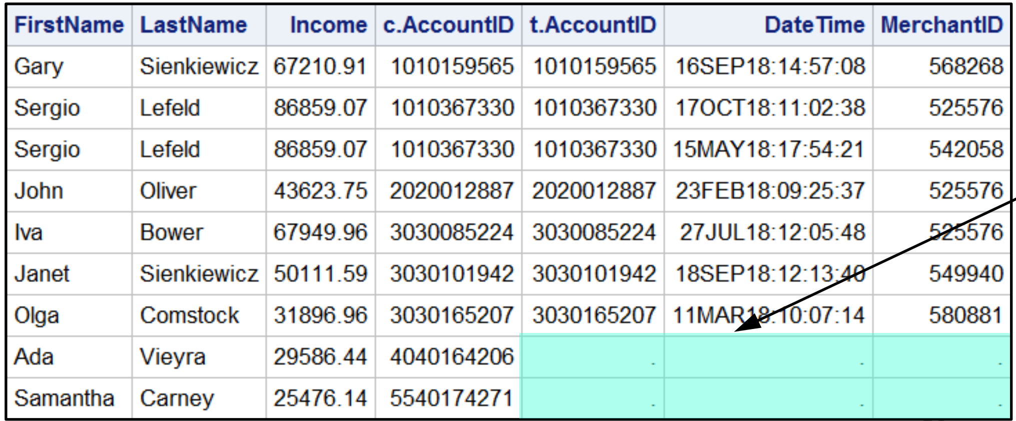
```
select FirstName, LastName, Income,
       c.AccountID "c.AccountID",
       t.AccountID "t.AccoundID",
       DateTime, MerchantID
    from sq.smallcustomer as c left join
         sq.smalltransaction as t
    on c.AccountID = t.AccountID;
```

FirstName	LastName	Income	c.AccountID	t.AccountID	DateTime	MerchantID
Gary	Sienkiewicz	67210.91	1010159565	1010159565	16SEP18:14:57:08	568268
Sergio	Lefeld	86859.07	1010367330	1010367330	17OCT18:11:02:38	525576
Sergio	Lefeld	86859.07	1010367330	1010367330	15MAY18:17:54:21	542058
John	Oliver	43623.75	2020012887	2020012887	23FEB18:09:25:37	525576
Iva	Bower	67949.96	3030085224	3030085224	27JUL18:12:05:48	525576
Janet	Sienkiewicz	50111.59	3030101942	3030101942	18SEP18:12:13:40	549940
Olga	Comstock	31896.96	3030165207	3030165207	11MAR18:10:07:14	580881
Ada	Vieyra	29586.44	4040164206	.	.	.
Samantha	Carney	25476.14	5540174271	.	.	.


Produce a list of ***customers*** who ***don't*** have a transaction.

Identifying Nonmatching Rows

```
select FirstName, LastName, Income,
       c.AccountID "c.AccountID",
       t.AccountID "t.AccoundID",
       DateTime, MerchantID
    from sq.smallcustomer as c left join
         sq.smalltransaction as t
    on c.AccountID = t.AccountID;
```



FirstName	LastName	Income	c.AccountID	t.AccountID	DateTime	MerchantID
Gary	Sienkiewicz	67210.91	1010159565	1010159565	16SEP18:14:57:08	568268
Sergio	Lefeld	86859.07	1010367330	1010367330	17OCT18:11:02:38	525576
Sergio	Lefeld	86859.07	1010367330	1010367330	15MAY18:17:54:21	542058
John	Oliver	43623.75	2020012887	2020012887	23FEB18:09:25:37	525576
Iva	Bower	67949.96	3030085224	3030085224	27JUL18:12:05:48	525576
Janet	Sienkiewicz	50111.59	3030101942	3030101942	18SEP18:12:13:40	549940
Olga	Comstock	31896.96	3030165207	3030165207	11MAR18:10:07:14	580881
Ada	Vieyra	29586.44	4040164206	.	.	.
Samantha	Carney	25476.14	5540174271	.	.	.

customers who do not have a transaction

Identifying Nonmatching Rows

```
select FirstName, LastName, Income,
       c.AccountID "c.AccountID",
       t.AccountID "t.AccoundID",
       DateTime, MerchantID
    from sq.smallcustomer as c left join
         sq.smalltransaction as t
    on c.AccountID = t.AccountID
    where t.AccountID is null;
```

The WHERE clause filters for all customers with a missing transaction **AccountID**.

FirstName	LastName	Income	c.AccountID	t.AccoundID	DateTime	MerchantID
Ada	Vieyra	29586.44	4040164206	.	.	.
Samantha	Carney	25476.14	5540174271	.	.	.

Syntax Summary

```
SELECT columns
       FROM table1 <LEFT |RIGHT |FULL > JOIN  table2
       ON table1.column = table2.column
```

Outer Join

```
COALESCE(argument-1, argument-2<, ...argument-n>)
```

COALESCE Function

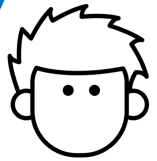
Lesson 3: SQL Joins

Reflexive Join

employee
EmployeeID
EmployeeName
ManagerID
…

The **employee** table includes a list of all employees.

Find **ManagerName** for each employee.



EmployeeID	EmployeeName	ManagerID	ManagerName
121044	Abbott, Ray	121144	
120145	Aisbitt, Sandy	120103	
120761	Akinfolarin, Tameaka	120746	
121144	Capachietti, Renee	121142	

Reflexive Join

EmployeeID	EmployeeName	ManagerID	ManagerName
121044	Abbott, Ray	121144	
120145	Aisbitt, Sandy	120103	
120761	Akinfolarin, Tameaka	120746	
121144	Capachietti, Renee	121142	

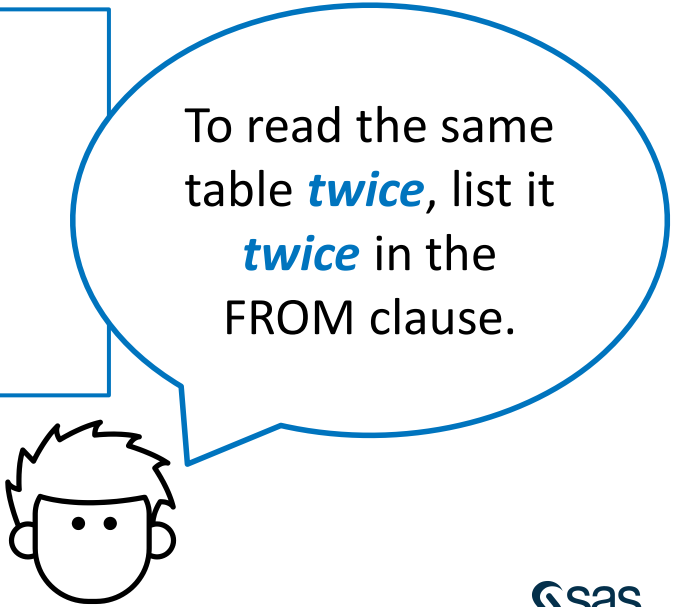
EmployeeID	EmployeeName	ManagerID	ManagerName
121044	Abbott, Ray	121144	Capachietti, Renee
120145	Aisbitt, Sandy	120103	
120761	Akinfolarin, Tameaka	120746	
121144	Capachietti, Renee	121142	

Self-join on the **employee** table to retrieve manager names.

Required Table Aliases

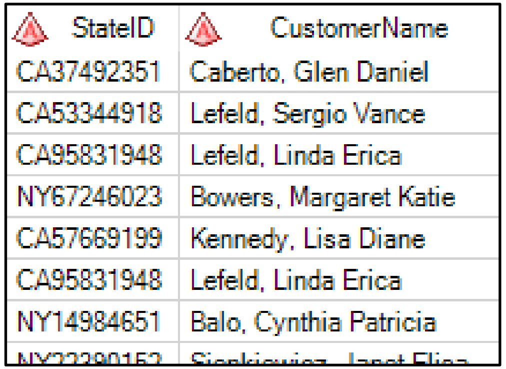
FROM *table1* **<AS>** *alias1* **JOIN-TYPE**
table1 **<AS>** *alias2*

```
select e.EmployeeID, e.EmployeeName,
       e.StartDate format=date9.,
       e.ManagerID,
       m.EmployeeName as ManagerName
    from sq.employee as e inner join
         sq.employee as m
    on e.ManagerID = m.EmployeeID;
```


To read the same table ***twice***, list it ***twice*** in the FROM clause.

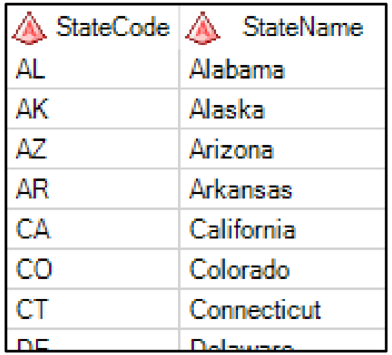
Scenario

transactionfull

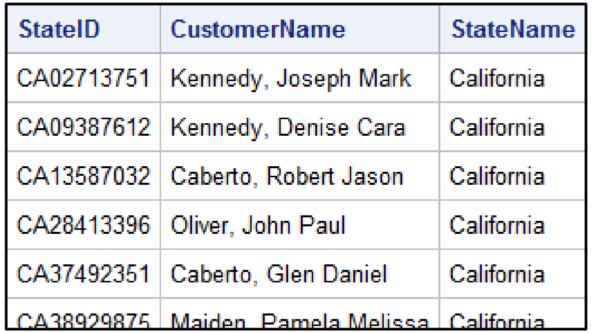


StateID	CustomerName
CA37492351	Caberto, Glen Daniel
CA53344918	Lefeld, Sergio Vance
CA95831948	Lefeld, Linda Erica
NY67246023	Bowers, Margaret Katie
CA57669199	Kennedy, Lisa Diane
CA95831948	Lefeld, Linda Erica
NY14984651	Balo, Cynthia Patricia
NY22290152	Sienkiewicz, Janet Elina

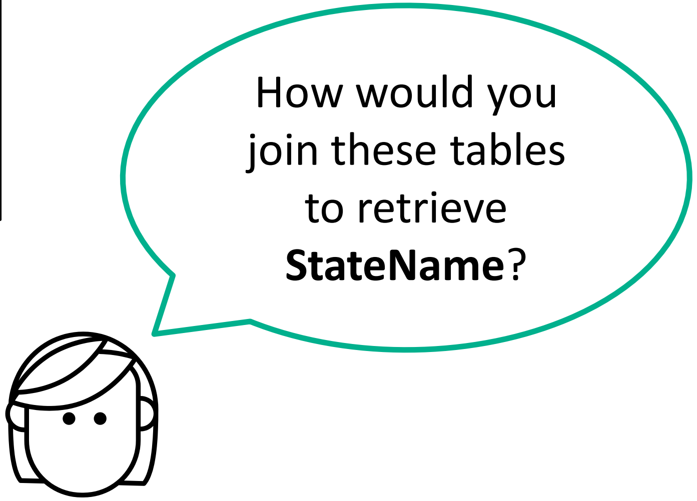
statecode



StateCode	StateName
AL	Alabama
AK	Alaska
AZ	Arizona
AR	Arkansas
CA	California
CO	Colorado
CT	Connecticut
DE	Delaware

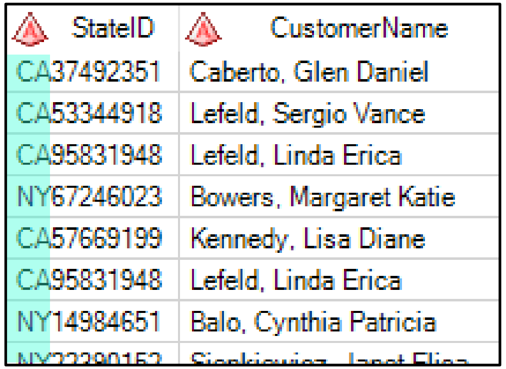


StateID	CustomerName	StateName
CA02713751	Kennedy, Joseph Mark	California
CA09387612	Kennedy, Denise Cara	California
CA13587032	Caberto, Robert Jason	California
CA28413396	Oliver, John Paul	California
CA37492351	Caberto, Glen Daniel	California
CA38929875	Maiden, Pamela Melissa	California



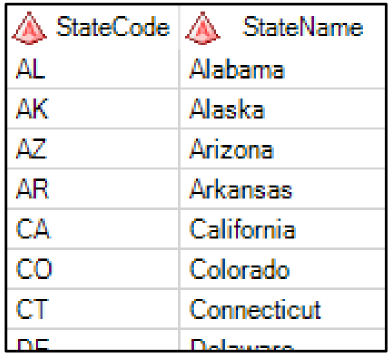
Using Functions to Join Tables

transactionfull



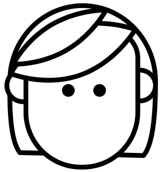
StateID	CustomerName
CA37492351	Caberto, Glen Daniel
CA53344918	Lefeld, Sergio Vance
CA95831948	Lefeld, Linda Erica
NY67246023	Bowers, Margaret Katie
CA57669199	Kennedy, Lisa Diane
CA95831948	Lefeld, Linda Erica
NY14984651	Balo, Cynthia Patricia
NY22290152	Sienkiewicz, Janet Elina

statecode



StateCode	StateName
AL	Alabama
AK	Alaska
AZ	Arizona
AR	Arkansas
CA	California
CO	Colorado
CT	Connecticut
DE	Delaware

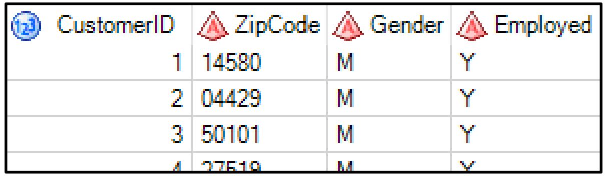
Use the SUBSTR function to extract the first two characters from **StateID**.



```
select StateID, CustomerName, StateName
    from sq.transactionfull as t inner join
         sq.statecode as s
      on substr(t.StateID,1,2) = s.StateCode;
```

Using Functions to Join Tables

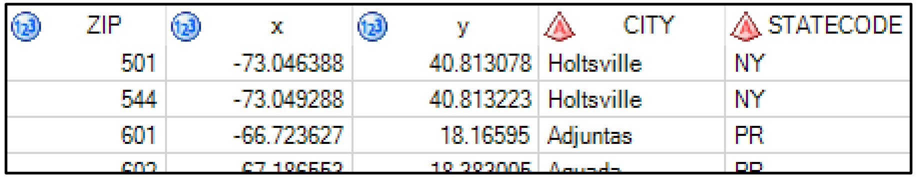
customerzip



CustomerID	ZipCode	Gender	Employed
1	14580	M	Y
2	04429	M	Y
3	50101	M	Y
4	27519	M	Y

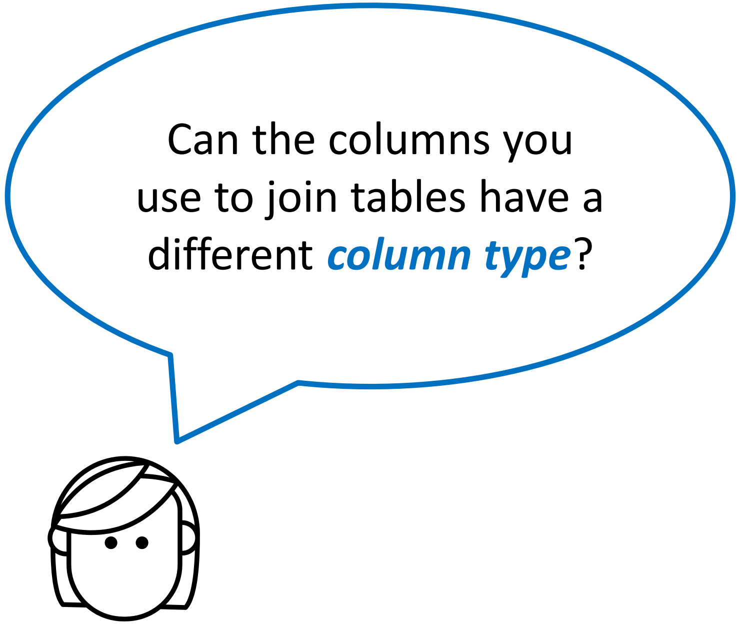
Character ZipCode

sashelp.zipcode



ZIP	x	y	CITY	STATECODE
501	-73.046388	40.813078	Holtsville	NY
544	-73.049288	40.813223	Holtsville	NY
601	-66.723627	18.16595	Adjuntas	PR
602	-67.186553	18.282005	Aguada	PR

Numeric ZIP


Can the columns you use to join tables have a different *column type*?

Using Functions to Join Tables

A syntax error was generated when you joined columns of different types.

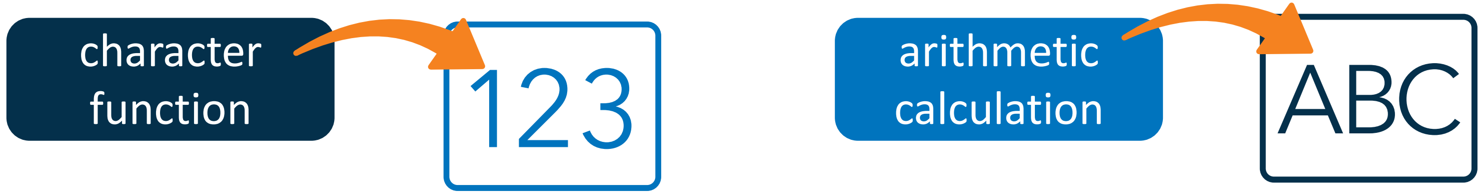
```
select c.CustomerID, c.ZipCode, c.Gender,
       z.Zip, z.City, z.StateCode
    from customerzip as c inner join
         sashelp.zipcode as z
      on c.ZipCode = z.Zip;
```

Character ZIP code

Numeric ZIP code

ERROR: Expression using equals (=) has components that are of different data types.

Converting Column Value Functions

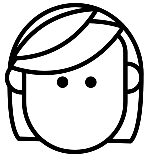


Function	What it does
INPUT(*source*, *informat*)	Converts character values to numeric values using a specified informat
PUT(*source*, *format*)	Converts numeric or character values to character values using a specified format

Converting Numeric to Character Values

```
put(z.Zip,z5.)
```

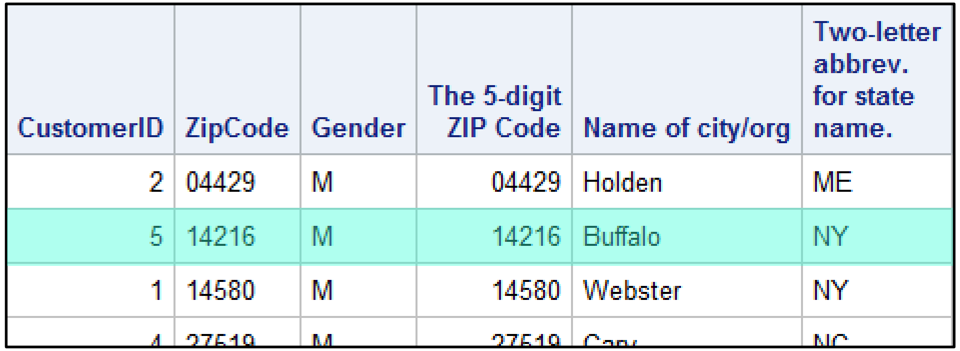
The **Z format** writes standard numeric data with leading 0s.



The PUT function with the **Z format** converts the numeric ZIP code *4429* to *04429*.

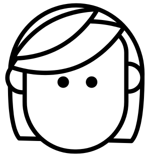
Converting Numeric to Character Values

```
select c.CustomerID, c.ZipCode, c.Gender,
       z.Zip, z.City, z.StateCode
    from customerzip as c inner join
         sashelp.zipcode as z
      on c.ZipCode = put(z.Zip,z5.);
quit;
```



CustomerID	ZipCode	Gender	The 5-digit ZIP Code	Name of city/org	Two-letter abbrev. for state name.
2	04429	M	04429	Holden	ME
5	14216	M	14216	Buffalo	NY
1	14580	M	14580	Webster	NY
4	27519	M	27519	Cary	NC

The **Zip** column converts to character in the ON clause.



Beyond SQL Essentials

What if you want to ...

. . . learn more about merging using the DATA step?

- Take the SAS Programming 2 course to learn more about DATA step match-merges.
- Visit the **SQL and the DATA Step** section on the ELP for additional resources about comparisons of the DATA step and PROC SQL.

. . . download the SQL Join Summary cheat sheet?

- Visit the **Course Handouts** section on the ELP and download the **SQL Join Summary** PDF.

. . . learn more about functions in PROC SQL?

- Read the SAS paper Top 10 Most Powerful Functions for PROC SQL.