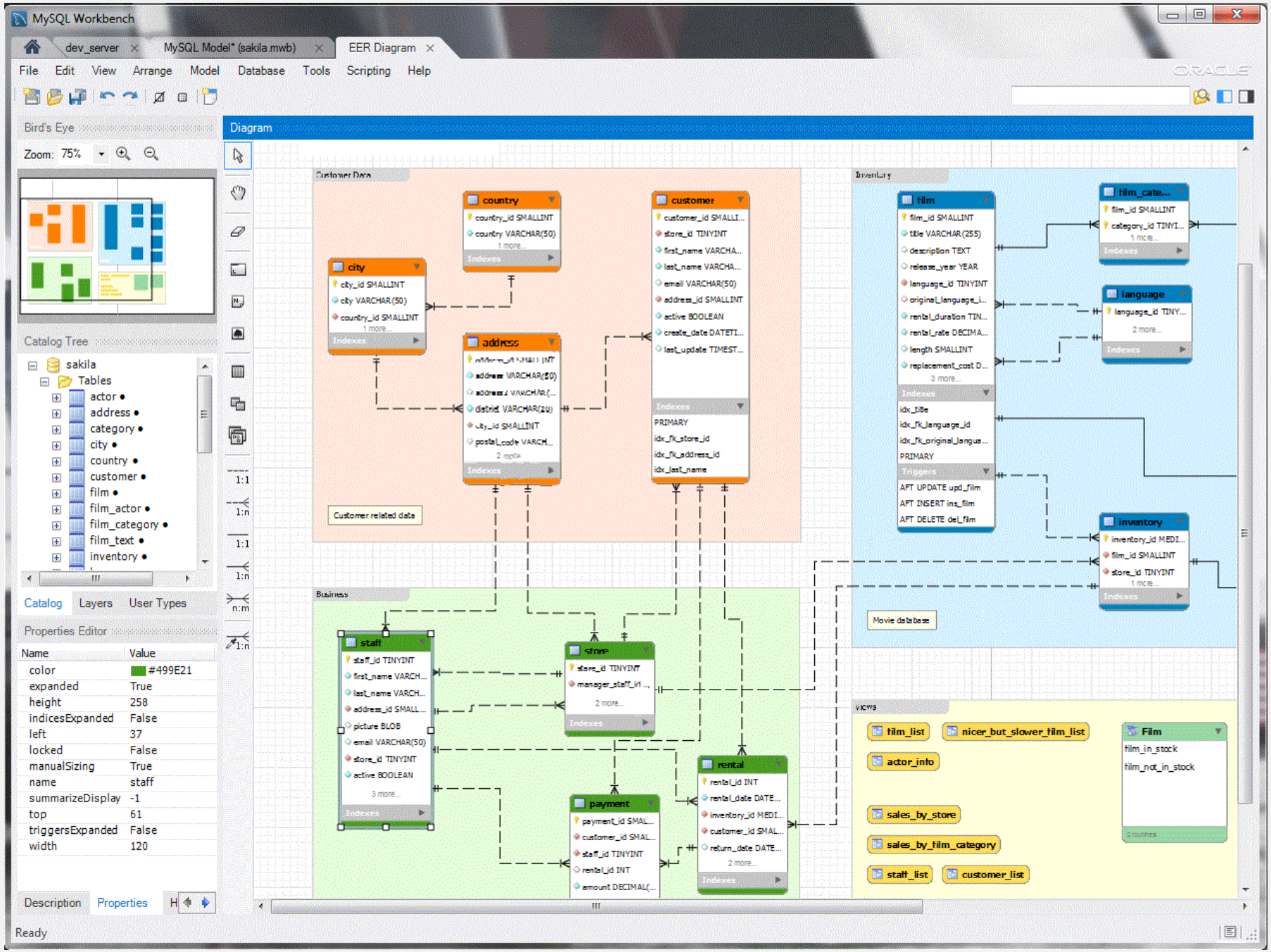
INTRODUCTION TO PROC SQL

Sarah Conner

April 26th, 2018

STRUCTURED QUERY LANGUAGE (SQL)

- Programming language for querying, manipulating, and designing data structures
- Geared towards relational databases with 'normalized' structure (long data)
- Used in programs such as Microsoft Access, SQL Server, and MySQL Workbench
- Variations of syntax: TSQL, MySQL, NoSQL


MySQL Workbench
dev_server
MySQL Model* (sakila.mwb)
EER Diagram
File Edit View Arrange Model Database Tools Scripting Help
Bird's Eye
Zoom: 75%
Catalog Tree
sakila
Tables
actor
address
category
city
country
customer
film
film_actor
film_category
film_text
inventory
Catalog Layers User Types
Properties Editor
Name Value
color #499E21
expanded True
height 258
indicesExpanded False
left 37
locked False
manualSizing True
name staff
summarizeDisplay -1
top 61
triggersExpanded False
width 120
Description Properties
Diagram
Customer related data
Movie database
Ready

COMPARISON TO SAS AND R

SAS	R Packages
• DATA step (including MERGE) • PROC MEANS • PROC FREQ • PROC PRINT • PROC SORT	• DPLYR • RESHAPE2 • DATA.TABLE

BASIC SYNTAX

```
SELECT * FROM sashelp.baseball
```

SYNTAX: SUMMARY VARIABLES

Optional. Without this, it will print out your results.

```
CREATE TABLE teams as
SELECT team, count(*) as PlayerCount
FROM sashelp.baseball
WHERE div='NW'
GROUP BY team
HAVING PlayerCount>12
ORDER BY PlayerCount DESC
```

Counts the number of observations. Can also count a certain variable, i.e. count(name), or perform other summaries

WHERE restricts the observations in the table specified in FROM

GROUP BY tells SQL how to aggregate the summary variables. Any variables in **SELECT** must be in **GROUP BY.**

HAVING restricts the values of a summary variable. Can also write count(*) instead of PlayerCount.

IN SAS

Allows SAS to accept SQL language

```
PROC SQL;
CREATE TABLE teams as
SELECT team, count(*) as PlayerCount
FROM sashelp.baseball
WHERE div='NW'
GROUP BY team
HAVING PlayerCount>12
ORDER BY PlayerCount DESC
;
QUIT;
```

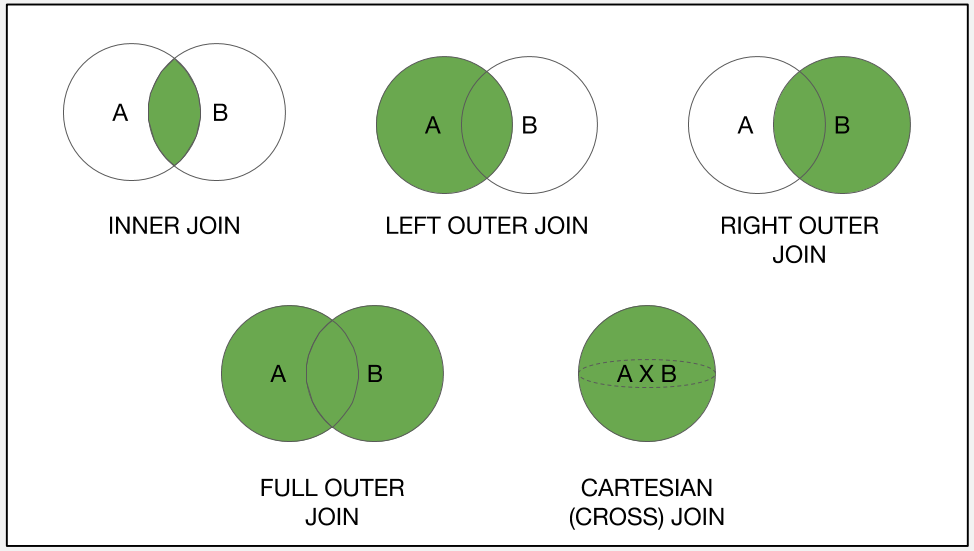
One semicolon at the end and QUIT;
Do not need RUN;

SUMMARY OPTIONS

- **AVG/MEAN**: average
- **COUNT**
- **MAX:** maximum
- **MIN:** minimum
- **RANGE**
- **STD:** standard deviation
- **STDERR:** standard error of the mean
- **SUM**
- **VAR:** variance
- **and others..**

JOINS

SQL JOIN	Description	SAS Equivalent
INNER JOIN	observations in both tables	merge data1(in=a) data2(in=b); if a and b;
LEFT JOIN	observations in the first table	merge data1(in=a) data2; if a;
RIGHT JOIN	observations in the second table	merge data1 data2(in=b); if b;
OUTER JOIN	observations in either table	merge data1 data2;
CROSS JOIN	cross product of all elements in all tables	proc freq; table var1*var2 / sparse;



IN SAS

```
PROC SQL;
CREATE TABLE zipcode2 as
SELECT u.division, u.region, z.*
FROM zipcode z
LEFT JOIN us_data u on z.statename=u.statename
WHERE z.statename='New York';
;
QUIT;
```

'Z' and 'U' are table **ALIASES,** or nicknames for tables in the rest of the query

Can merge/join by variables with different names. No need to sort beforehand. PROC SQL requires you to write both variables, but outside of SAS, you would write '**USING** statename' to simplify.

ALTERNATIVES TO 'SELECT'

- **ALTER TABLE**: add, drop, and modify columns in a table
- **CREATE**: build new tables, views, or indexes
- **DELETE**: eliminate unwanted rows from a table or view
- **DESCRIBE**: display table and view attributes
- **DROP**: eliminate entire tables, views, or indexes
- **INSERT**: add rows of data to tables or views
- **RESET**: add to or change PROC SQL options without re-invoking the procedure
- **UPDATE**: modify data values in existing rows of a table or view

HELPFUL TRICKS IN SAS

NESTED QUERIES, OR SUBQUERIES

- Used to manipulate/subset data without creating extra, intermediate datasets
- Insert a query within **SELECT**, **FROM**, **JOIN**, or **WHERE** lines

IN SAS

```
PROC SQL;
SELECT zip, city
FROM zipcode
WHERE city in
      (SELECT team
      FROM sashelp.baseball)
;
QUIT;
```

Can also give the subquery an alias

DISTINCT AND COUNT DISTINCT

- DISTINCT retrieves unique values
- Useful in combination with COUNT to identify duplicate records

IN SAS

```
PROC SQL;
SELECT distinct team
FROM sashelp.baseball;

SELECT count(distinct team)
FROM sashelp.baseball;
QUIT;
```

Can be helpful to identify multiple observations

CREATE A MACRO VARIABLE

- Use PROC SQL to select values **INTO** a macro variable
- Useful when…
  - There isn't a clear shortcut in SAS
  - Need to access DICTIONARY.COLUMNS (metadata about all libraries, datasets, variables, etc.)
- Example: US_DATA contains variables with the same *suffix*, not prefix. How can we subset the data to include variables that end with '2010'?