Introduction to SQL

Outline

- SQL
- SQL syntax
- Subsetting
- Summary function
- Grouping data
- Subquaries
- Combining tables

SQL

Structured Query Language (SQL)

- is a standardized language that is widely used to retrieve and update data in tables and in views based on those tables
- was originally designed as a query tool for relational databases, but it is now used by many software products.

SQL

- Database software: MySQL, SQL server, Oracle, SQLite…
- PROC SQL procedure in SAS
- R: DBI, dplyr, {sql} chunk in R Notebook

SELECT Statement Syntax

General form of the SELECT statement:

```
SELECT column-1<, column-2>
      FROM table-1|view-1<, table-2|view-2>
      <WHERE expression>
      <GROUP BY column-1<, column-2>…>
      <HAVING expression>
      <ORDER BY column-1<, column-2>…
<DESC>>;
```

SELECT Statement Syntax

where

SELECT specifies the column(s) that will appear in the output

FROM specifies the table(s) or view(s) to be queried

WHERE uses an expression to subset or restrict the data based on one or more condition(s)

GROUP BY classifies the data into groups based on the specified column(s)

HAVING uses an expression to subset or restrict groups of data based on a group condition

ORDER BY sorts the rows that the query returns by the value(s) of the specified column(s).

Retrieving Data from a Table

```
select EmpID, JobCode, Salary
       from airline.payrollmaster;
```

```
select *
       from airline.payrollmaster;
```

Expressions

```
select EmpID, JobCode, Salary,
       Salary * .10 as Bonus
   from payrollmaster;
```

```
select EmpID, JobCode,
       int((today()-DateOfBirth)/365.25)
       as Age
   from payrollmaster;
```

Eliminating Duplicate Rows

```
select distinct FlightNumber,
       Destination
  from internationalflights;
```

Subsetting with the WHERE Clause

- Usual logical operators: < > <= >= = <>
- BETWEEN-AND: with an inclusive range
- IN: match one of a list of values
- Keyword NOT used for negation
- =*:sound like
- contains or ?: contain a specified string
- LIKE operator allows wildcards

  _ means single character, % means anything

  SELECT salary WHERE name LIKE 'Fred %'
- AND(&&) and OR(||) to combine conditions

Subsetting with the WHERE Clause

```
where JobCategory in ('PT','NA','FA')
```

```
where DayOfWeek in (2,4,6)
```

```
where word ? 'LAM'
```

```
where Date between '01mar2000'd
      and '07mar2000'd
```

```
where Salary between 70000 and 80000
```

Subsetting with the WHERE Clause

```
where JobCode like '__1'
```

```
where boarded is missing
```

```
where LastName like 'H%'
```

```
where LastName =* 'SMITH'
```

selects values SMITT, SMYTHE, and SMOTHE, in addition to SMITH.

Subsetting with the WHERE Clause

```
select EmpID,JobCode,Salary
   from payrollmaster
   where Salary > 112000;
```

Subsetting with the WHERE Clause

Because a WHERE clause is evaluated first, columns used in the WHERE clause must exist in the table or be derived from existing columns.

```
select FlightNumber,Date,Destination,
          Boarded + Transferred + Nonrevenue
          as Total
      from marchflights
      where Total < 100;
```

ERROR

Subsetting with the WHERE Clause

Because a WHERE clause is evaluated first, columns used in the WHERE clause must exist in the table or be derived from existing columns.

```
   select FlightNumber, Date, Destination,
          Boarded+Transferred+Nonrevenue
          as Total
      from marchflights
      where Boarded+Transferred+Nonrevenue < 100;
```

```
   select FlightNumber, Date, Destination,
          Boarded + Transferred + Nonrevenue
          as Total
      from marchflights
      where calculated Total < 100;
```

Ordering Data

```
select EmpID,JobCode,Salary
   from airline.payrollmaster
   where JobCode contains 'NA'
   order by Salary desc;
```

Summary Functions

Example: Find the total number of passengers for each flight in March.

```
select Date, FlightNumber, Boarded,
       Transferred, Nonrevenue,
 sum(Boarded,Transferred,Nonrevenue) as Total
   from marchflights;
```

Example: Determine the average salary for the company.

```
select avg(Salary) as MeanSalary
   from payrollmaster;
```

Summary Functions

The following are selected functions:

AVG, MEAN

COUNT

MAX

MIN

NMISS

STD

SUM

VAR

Summary Functions

Counting Values by Using the COUNT Summary Function:

```
select count(*) as Count
```

```
select count(jobcode) as Count
```

```
select count(distinct jobcode) as Count
```

Grouping Data

By combining with the GROUP BY command, useful summaries can be obtained.

```
select JobCode, avg(Salary) as
        average format=dollar11.2
   from payrollmaster
   group by JobCode;
```

Grouping Data

The WHERE clause selects data based on values for individual rows. To select entire groups of data, use the HAVING clause

```
select JobCode, avg(Salary) as average
      format=dollar11.2
      from payrollmaster
   group by JobCode
   having avg(Salary) > 50000 ;
```

Subquaries

Subqueries are inner queries that return values to be used by an outer query to complete a subsetting expression in a WHERE or HAVING clause.

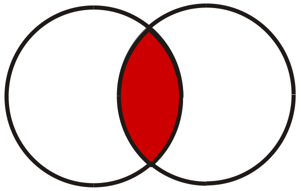
```
select JobCode,avg(Salary) as MeanSalary
      from payrollmaster
      group by JobCode
      having avg(Salary) >
         (select avg(Salary)
            from payrollmaster);
```

Subquaries

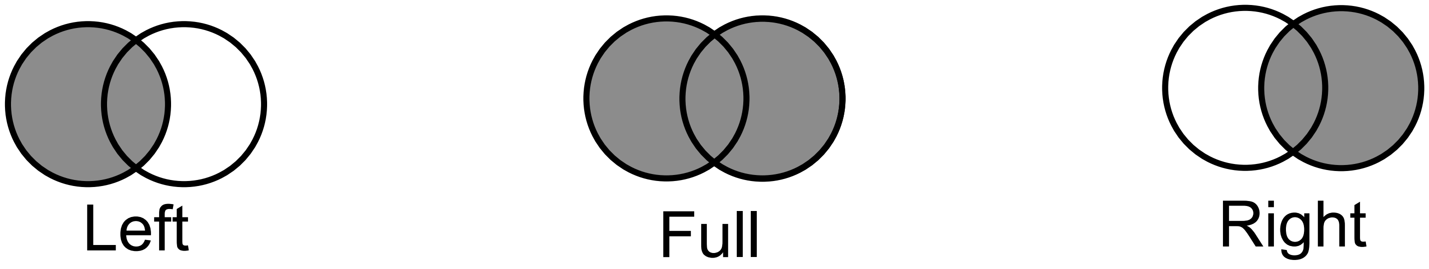
```
select EmpID, LastName,    FirstName, City, State
from staffmaster
where EmpID in
  (select EmpID
   from payrollmaster
   where month(DateOfBirth)=2);
```

Combining Tables

- **inner joins**: return only matching rows



- **outer joins**: return all matching rows, plus nonmatching rows from one or both tables



Cartesian Product (Cross Join)

A join of every row of one table to every row of another table.

```
select *
      from Table1, Table2;
```

Cartesian Product(Cross Join)

X	A
1	a
2	b
3	c

X	Y
2	x
4	y
5	z

X	A	X	Y
1	a	2	x
1	a	4	y
1	a	5	z
2	b	2	x
2	b	4	y
2	b	5	z
3	c	2	x
3	c	4	y
3	c	5	z

Inner Joins

An inner join combines and displays only the rows from the first table that match rows from the second table.

```
SELECT column-1<,...column-n>
      FROM table-1, table-2 <,...table-n >
      WHERE join-condition(s)
            <AND other subsetting condition(s)>
      <other clauses>;
```

Inner Joins

X	A
1	a
2	b
3	c

X	Y
2	x
4	y
5	z

X	A	X	Y
1	a	2	x
1	a	4	y
1	a	5	z
2	b	2	x
2	b	4	y
2	b	5	z
3	c	2	x
3	c	4	y
3	c	5	z

```
select *
   from table1,table2
   where table1.X= table2.X;
```

Inner Joins

Eliminating Duplicate Columns

```
select *
   from table1, table2
   where table1.X= table2.X;
```

X	A	X	Y
2	b	2	x

```
select table1.X, A, Y
   from table1, table2
   where table1.X= table2.X;
```

X	A	Y
2	b	x

```
select table1.*, Y
   from table1, table2
   where table1.X= table2.X;
```

Inner Joins

Renaming a Column by Using a Column Alias

```
select table1.X as ID, table2.X, A, Y
   from table1, table2
   where table1.X= table2.X;
```

ID	X	A	Y
2	2	b	x

Inner Joins

Specifying a Table Alias

```
select staffmaster.empid, lastname,
       firstname, jobcode
from staffmaster, payrollmaster

where staffmaster.empid=payrollmaster.empid;
```

Inner Joins

Specifying a Table Alias

```
select s.empid, lastname,
    firstname, jobcode
from staffmaster as s,
   payrollmaster as p
where s.empid=p.empid;
```

Outer Joins

An outer join combines and displays all rows that **match** across tables, **plus** some or all of the rows that do **not match**.

General form, SELECT statement for inner join:

```
SELECT column-1<,...column-n>
    FROM table-1
           LEFT JOIN | RIGHT JOIN | FULL JOIN
           table-2
           ON join-condition(s)

    <other clauses>;
```

Left Join

Return rows from both tables, plus nonmatching rows from the left table.

Table1

X	A
1	a
2	b
3	c

Table2

X	Y
2	x
4	y
5	z

$\Rightarrow$

X	A	X	B
1	a	.	
2	b	2	x
3	c	.	

```
select *
   from Table1 left join Table2
   on Table1.X= Table2.X;
```

Right Join

Return rows from both tables, plus nonmatching rows from the right table.

Table1

X	A
1	a
2	b
3	c

Table2

X	Y
2	x
4	y
5	z

→

X	A	X	Y
2	b	2	x
.		4	y
.		5	z

```
select *
 from Table1 right join Table2
   on Table1.X= Table2.X;
```

Full Join

Return matching rows and nonmatching rows from both tables.

Table1

X	A
1	a
2	b
3	c

Table2

X	Y
2	x
4	y
5	z

X	A	X	Y
1	a	.	
2	b	2	x
.		4	y
3	c	.	
.		5	z

```
select *
 from Table1 full join Table2
   on Table1.X= Table2.X;
```

PROC SQL procedure in SAS

- Each statement is processed individually.
- No PROC PRINT step is needed to view query results.
- No PROC SORT step is needed to order query results.
- No RUN statement is needed.
- Use a QUIT statement to terminate PROC SQL.

SQL in R

- https://db.rstudio.com/getting-started/database-queries/
- DBI
- dplyr
- R Notebook SQL engine

Resources

- https://jpsmonline.umd.edu/SASOnlineTutor/sot12/en/60477/index.htm
- https://db.rstudio.com/getting-started/database-queries/
- https://www.coursera.org/learn/intro-sql#reviews