HOCHSCHULE
DER MEDIEN

Introduction to SQL

Grouping and Summarizing

Prof. Dr. Jan Kirenz

HdM Stuttgart

PostgreSQL Setup

pw = "your_password"

```
library(DBI)
library(RPostgres)

con <- dbConnect(RPostgres::Postgres(),
                 dbname = "postgres",
                 host = "localhost",
                 port = 5432,
                 user = "postgres",
                 password = pw)
```

The examples in this presentation are based on the excellent book "A Beginner's Guide to Storytelling with Data" from Anthony DeBarros (2018).

CREATE TABLES and INDEX

- Creating and filling the 2014 Public Libraries Survey table.
- We only take a look at the first few lines:

```
CREATE TABLE pls_fy2014_pupld14a (
    stabr varchar(2) NOT NULL,
    fscskey varchar(6) CONSTRAINT fscskey2014_key PRIMARY KEY,
    libid varchar(20) NOT NULL,
    libname varchar(100) NOT NULL,
    obereg varchar(2) NOT NULL,
    rstatus integer NOT NULL,
    statstru varchar(2) NOT NULL,
    statname varchar(2) NOT NULL,
    stataddr varchar(2) NOT NULL,
    longitud numeric(10,7) NOT NULL,
    latitude numeric(10,7) NOT NULL,
    fipsst varchar(2) NOT NULL,
    fipsco varchar(3) NOT NULL,
    address varchar(35) NOT NULL,
    city varchar(20) NOT NULL,
    zip varchar(5) NOT NULL,
    zip4 varchar(4) NOT NULL,
    cnty varchar(20) NOT NULL,
    phone varchar(10) NOT NULL,
```

CREATE INDEX

```
CREATE INDEX libname2014_idx ON pls_fy2014_pupld14a (libname);
```

```
CREATE INDEX stabr2014_idx ON pls_fy2014_pupld14a (stabr);
```

```
CREATE INDEX city2014_idx ON pls_fy2014_pupld14a (city);
```

```
CREATE INDEX visits2014_idx ON pls_fy2014_pupld14a (visits);
```

COPY FROM

- Data: pls_fy2014_pupld14a.csv

```
COPY pls_fy2014_pupld14a
FROM '/tmp/pls_fy2014_pupld14a.csv'
WITH (FORMAT CSV, HEADER);
```

CREATE TABLE

- Creating and filling the 2009 Public Libraries Survey table.
- We only take a look at the first few lines:

```
CREATE TABLE pls_fy2009_pupld09a (
    stabr varchar(2) NOT NULL,
    fscskey varchar(6) CONSTRAINT fscskey2009_key PRIMARY KEY,
    libid varchar(20) NOT NULL,
    libname varchar(100) NOT NULL,
    address varchar(35) NOT NULL,
    city varchar(20) NOT NULL,
    zip varchar(5) NOT NULL,
    zip4 varchar(4) NOT NULL,
    cnty varchar(20) NOT NULL,
    phone varchar(10) NOT NULL,
    c_relatn varchar(2) NOT NULL,
    c_legbas varchar(2) NOT NULL,
    c_admin varchar(2) NOT NULL,
    geocode varchar(3) NOT NULL,
    lsabound varchar(1) NOT NULL,
    startdat varchar(10),
    enddate varchar(10),
    popu_lsa integer NOT NULL,
    centlib integer NOT NULL,
```

CREATE INDEX

```
CREATE INDEX libname2009_idx ON pls_fy2009_pupld09a (libname);
```

```
CREATE INDEX stabr2009_idx ON pls_fy2009_pupld09a (stabr);
```

```
CREATE INDEX city2009_idx ON pls_fy2009_pupld09a (city);
```

```
CREATE INDEX visits2009_idx ON pls_fy2009_pupld09a (visits);
```

COPY

- Data: pls_fy2009_pupld09a.csv

```
COPY pls_fy2009_pupld09a
FROM '/tmp/pls_fy2009_pupld09a.csv'
WITH (FORMAT CSV, HEADER);
```

Exploring Data with Aggregate Functions

Aggregate functions combine values from multiple rows and return a single result based on an operation on those values.

Counting Rows and Values Using COUNT

- The COUNT aggregate function makes it easy to check the number of rows and peform other counting tasks.
- If we supply an **asterisk** as an input, such as COUNT(*), the asterisk acts as a wildcard, so the function returns the number of table rows regardless of whether they include NULL (missing) values.
- Check number of rows for pls_fy2014_pupld14a:

```
SELECT COUNT(*)
FROM pls_fy2014_pupld14a;
```

count
9305

Counting Rows and Values Using COUNT

- **Check number of rows for** pls_fy2009_pupld09a**:**

```
SELECT COUNT(*)
FROM pls_fy2009_pupld09a;
```

count
9299

Counting Rows and Values Using COUNT

- Counting the number of salaries column 2014:

```
SELECT COUNT(salaries)
FROM pls_fy2014_pupld14a;
```

count
5983

- This number is far lower than the number of rows that exist in the table

Counting Distinct Values in a Table

- When added after SELECT, DISTINCT returns a list of unique values
- When added to the COUNT function, DISTINCT causes the function to return a count of distinct values from a column
- Query counts all values in the 2014 table's libname column:

```
SELECT COUNT(libname)
FROM pls_fy2014_pupld14a;
```

count
9305

Counting Distinct Values in a Table

- Counts all values in the 2014 table's libname column but includes DISTINCT in front of the column name:

```
SELECT COUNT(DISTINCT libname)
FROM pls_fy2014_pupld14a;
```

count
8515

Finding Maximum and Minimum Values

MAX

- Use a SELECT statement followed by the function MAX with the name of a column visits supplied:

```
SELECT MAX(visits)
FROM pls_fy2014_pupld14a;
```

max
17729020

Finding Maximum and Minimum Values

MIN

- Use a SELECT statement followed by the function MIN with the name of a column visits supplied:

```
SELECT MIN(visits)
FROM pls_fy2014_pupld14a;
```

min
-3

- Note: -3 is used to indicate "not applicable" and is used when a library agency has closed either temporarily or permanently.

Aggregating Data Using GROUP BY

- When you use the GROUP BY clause with aggregate functions, you can group results according to the values in one or more columns.
- GROUP BY statement follows the FROM clause and includes the column name to group.
- The stabr (state abbreviation) grouped results:

```
SELECT stabr
FROM pls_fy2014_pupld14a
GROUP BY stabr;
```

stabr
NV
OH
NY
WV
AR
CT

ORDER BY

- The stabr grouped results are in alphabetical order:

```
SELECT stabr
FROM pls_fy2014_pupld14a
GROUP BY stabr
ORDER BY stabr;
```

stabr
AK
AL
AR
AS
AZ
CA

ORDER BY

- **Results get sorted by** city **and then by** stabr**:**

```
SELECT city, stabr
FROM pls_fy2014_pupld14a
GROUP BY city, stabr
ORDER BY city, stabr;
```

city	stabr
ABBEVILLE	AL
ABBEVILLE	LA
ABBEVILLE	SC
ABBOTSFORD	WI
ABERDEEN	ID
ABERDEEN	SD

Combining GROUP BY with COUNT

- We can get a count of agencies by state and sort them to see which states have the most.
- To sort the results and have the state with the largest number of agencies at the top, we can ORDER BY the COUNT function in descending order using DESC:

```
SELECT stabr, COUNT(*)
FROM pls_fy2014_pupld14a
GROUP BY stabr
ORDER BY COUNT(*) DESC;
```

stabr	count
NY	756
IL	625
TX	556
IA	543
PA	455
MI	389

GROUP BY on Multiple Columns with COUNT

- The stataddr column contains a code indicating whether the agency's address changed in the last year.
- The values in stataddr are 00 (= no change from last year), 07 (=moved to a new location), 15 (=minor address change).

GROUP BY on Multiple Columns with COUNT

- This code is counting the number of agencies in each state that moved, had a minor address change, or had no change using the GROUP BY with stabr and stataddr and adding COUNT:

```
SELECT stabr, stataddr, COUNT(*)
FROM pls_fy2014_pupld14a
GROUP BY stabr, stataddr
ORDER BY stabr, COUNT(*) DESC;
```

stabr	stataddr	count
AK	00	70
AK	15	10
AK	07	5
AL	00	221
AL	07	3
AR	00	58

Using GROUP BY on Multiple Columns with COUNT

- The effect of grouping by *two columns* is that COUNT will show the number of unique combinations of stabr and stataddr.
- The first few rows of the results show that the code "00" is the most common value for each state.
- This makes sense because it's likely there are more agencies that haven't changed address than those that have.

Use SUM to aggregate values

- Query total visits to libraries in 20014:
  - Use SUM to view total visits to libraries in 2014 (AS visits_2014).
  - Use a WHERE clause to specify that the result should include only those rows where visits are greater than or equal to "0".

```
SELECT SUM(visits) AS visits_2014
FROM pls_fy2014_pupld14a
WHERE visits >= 0;
```

visits_2014
1425930900

Use SUM to aggregate values

- Query total visits to libraries in 2009:
  - Use SUM to view total visits to libraries in 2009 (AS visits_2009).
  - Use a WHERE clause to specify that the result should include only those rows where visits are greater than or equal to "0".

```
SELECT SUM(visits) AS visits_2009
FROM pls_fy2009_pupld09a
WHERE visits >= 0;
```

visits_2009
1591799201

Use SUM and JOIN to aggregate values

- Using SUM to query total visits on **joined** 2014 and 2009 library tables:
  - Use ALIAS: "pls_fy2014_pupld14a AS pls14" and "pls_fy2009_pupld09a AS pls09"
  - Use a WHERE clause to specify that the result should include only those rows where visits are greater than or equal to 0.
  - JOIN the two tables ON the primary key fscskey

--

```
SELECT SUM(pls14.visits) AS visits_2014,
      SUM(pls09.visits) AS visits_2009
FROM pls_fy2014_pupld14a AS pls14 JOIN pls_fy2009_pupld09a AS pls09
ON pls14.fscskey = pls09.fscskey
WHERE pls14.visits >= 0 AND pls09.visits >= 0;
```

visits_2014	visits_2009
1417299241	1585455205

Use SUM and JOIN to aggregate values

- In our next SELECT statement, we use SUM to total visits columns from 2014 and 2009 tables.
- We declare pls14 as the **alias** for the 2014 table and pls09 as the **alias** for the 2009 table to avoid having to write the full table names throughout the query.
- We use a standard JOIN or INNER JOIN, so the query results will only include rows where the primary key values of both tables match (fscskey).
- In the WHERE clause, we specify that the result should include only those rows where visits are greater than or equal to 0.

Use SUM to aggregate values

- Query:

```
SELECT (SUM(pls14.visits) +
      SUM(pls09.visits)) AS total_visits
FROM pls_fy2014_pupld14a AS pls14 JOIN pls_fy2009_pupld09a AS pls09
ON pls14.fscskey = pls09.fscskey
WHERE pls14.visits >= 0 AND pls09.visits >= 0;
```

total_visits
3002754446

Using GROUP BY to track percent changes

- Now that we know library visits for the United States dropped as a whole between 2009 and 2014, we might want to know, whether every part of the country saw a decrease?
- Using GROUP BY to track percent change in library visits by state.

```
SELECT pls14.stabr,
      SUM(pls14.visits) AS visits_2014,
      SUM(pls09.visits) AS visits_2009,
      ROUND( (CAST(SUM(pls14.visits) AS decimal(10,1)) - SUM(pls09.visits)) /
            SUM(pls09.visits) * 100, 2 ) AS pct_change
FROM pls_fy2014_pupld14a AS pls14 JOIN pls_fy2009_pupld09a AS pls09
ON pls14.fscskey = pls09.fscskey
WHERE pls14.visits >= 0 AND pls09.visits >= 0
GROUP BY pls14.stabr
ORDER BY pct_change DESC;
```

```
<div id="htmlwidget-cc0ae150bc759e0338a9" style="width:100%;height:auto;" class="datatables
<script type="application/json" data-for="htmlwidget-cc0ae150bc759e0338a9">{"x":{"filter":"none"
```

Using HAVING to filter the results of an aggregate query

- We can refine our analysis by examining a subset of states and territories that share similar characteristics.
- In a small state, one library closure could have a significant effect, whereas a single closure in a large state might be scarcely noticed in a statewide count.
- To look at states with similar volume in visits, we could sort the results by either of the visits column, but it is cleaner to get a smaller result set in our query.
- We are familiar with using WHERE for filtering, but aggregate functions, such as SUM, can't be used within a WHERE clause because WHERE operates at a row level, and aggregate functions work across rows.
- HAVING places conditions on groups created by aggregating.

Using HAVING to filter the results of an aggregate query

- - We set our query to show only the very larges states (visits greater than 50 million)

```
SELECT pls14.stabr,
      SUM(pls14.visits) AS visits_2014,
      SUM(pls09.visits) AS visits_2009,
      ROUND( (CAST(SUM(pls14.visits) AS decimal(10,1)) - SUM(pls09.visits)) /
            SUM(pls09.visits) * 100, 2 ) AS pct_change
FROM pls_fy2014_pupld14a AS pls14 JOIN pls_fy2009_pupld09a AS pls09
ON pls14.fscskey = pls09.fscskey
WHERE pls14.visits >= 0 AND pls09.visits >= 0
GROUP BY pls14.stabr
HAVING SUM(pls14.visits) > 50000000
ORDER BY pct_change DESC;
```

Using HAVING to filter the results of an aggregate query

stabr	visits_2014	visits_2009	pct_change
TX	72876601	78838400	-7.56
CA	162787836	182181408	-10.65
OH	82495138	92402369	-10.72
NY	106453546	119810969	-11.15
IL	72598213	82438755	-11.94
FL	73165352	87730886	-16.60

- HAVING reduces the number of rows in the output to just six
- Each of the six states has experienced a decline in vists, but the percentage variation isn't as wide as in the full set of states.

Thank you!

Prof. Dr. Jan Kirenz

HdM Stuttgart
Nobelstraße 10
70569 Stuttgart