SQL For Data Science

mm/dd - Steve Mortimer, XXX XXXXXX, XXX XXXXXX

Why Learn SQL?

Fundamental data science skill

1. **Ubiquitous -** It's everywhere in the data community

2. SQL teaches you **Relational Algebra** and **Set Theory**

3. Expands Your **Computing Power**

4. Ease of **Access -** You can get started today!

What is a Relational Database?

Key terminology and concepts

- **Rows** represent single items (e.g. an MBA student)
- **Columns** are labeled attributes for each item (e.g. “Name: Steve”, “School: Darden”, “Year: 2”)
- **Tables** are sets of rows that share the same attributes (“Students”)
- **Views** are sets of any rows from different tables in response to a query (“Get me all Second-Year Students in Darden”)
- Proposed in 1970 by E. F. Codd while working at IBM

Source: Wikipedia

Why Relational Databases?

Key terminology and concepts

Relational data models:

- Are very easy to extend
- Improve and maintain data integrity
- Allow for re-assembling data many different ways without reorganizing the underlying structure

“Normalization” is the extent to which the data replicated into multiple places within the database.

Highly Normalized = Little Redundancy = More Joins

Denormalized = More Redundancy = Less Joins

AdventureWorks Cycles is a fictitious bicycle manufacturer

Scenarios include:

- Manufacturing
- Sales
- Purchasing
- Product Management
- Contact Management
- Human Resources

Data needs to be hosted on a running database!

PostgreSQL

AdventureWorks Source:
https://github.com/infinuendo/AdventureWorks/tree/master/OLTP/Original

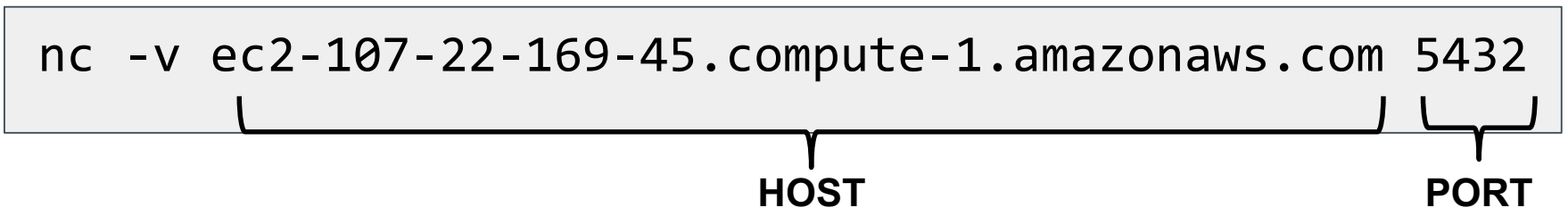
Connecting to the database server requires a set of credentials

You will need:

- Host - *ec2-107-22-169-45.compute-1.amazonaws.com*
- Port - *5432*
- Username - *rmfenpdczlrtqo*
- Password - *57a3df647d21f42d4dc029cd951ac7762e1ad440e411dfc8fb7dffb57d82d976*
- Database Name - *d54sjlbs6erhjs*

Check the database is running (open your terminal window):

```
nc -v ec2-107-22-169-45.compute-1.amazonaws.com 5432
```



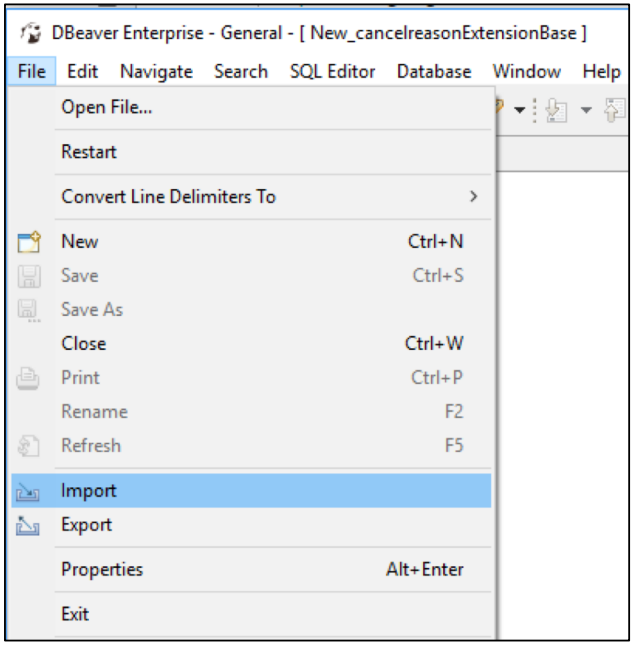
A SQL Client is a program to query databases

First, install DBeaver - http://dbeaver.jkiss.org/download/

Second, download profile (contains connection and SQL scripts)

Third, import the profile into DBeaver

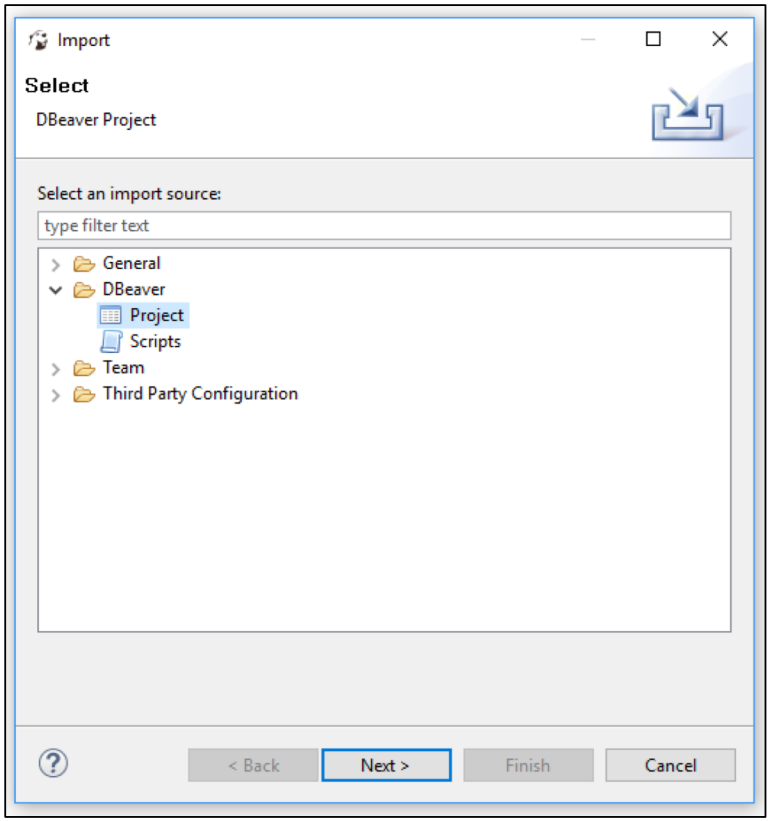
Open up DBeaver -> File -> Import



Next,

Select “Project” as the file type you’d like to import

Click “Next”



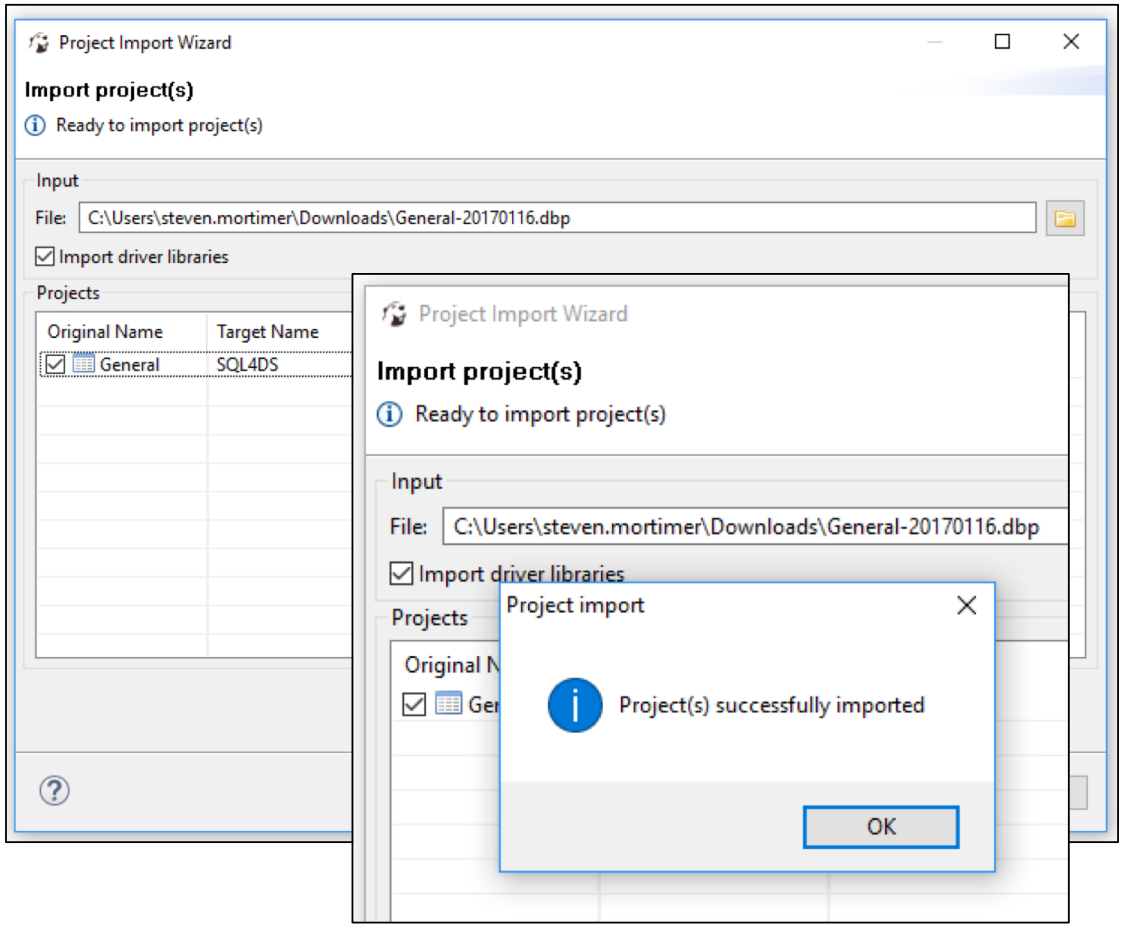
Next,

Select the .dbp file that you downloaded

Ensure that “Import driver libraries” is checked

Ensure the Target Name is “SQL4DS”

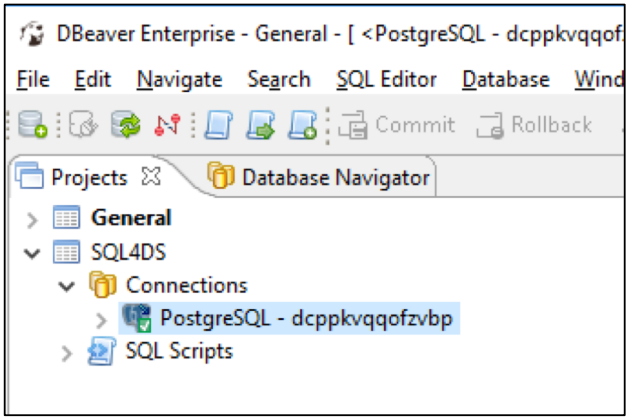
Click “Finish”



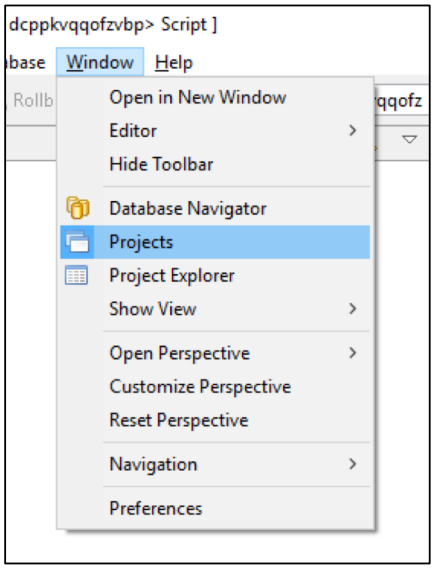
Getting Started

DBeaver - A SQL client (continued)

Click on SQL4DS Project. You should see a connection “PostgreSQL” with green check.



If you do not see a “Projects” tab, go to “Window” -> “Projects”

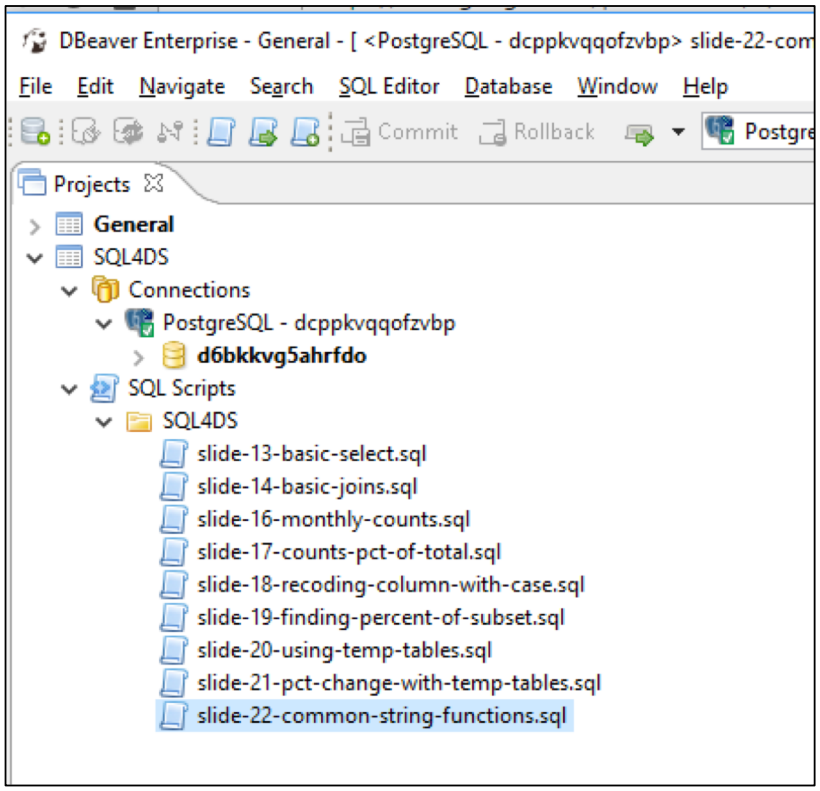


Getting Started

DBeaver - A SQL client (continued)

If you click on arrow to expand the “SQL Scripts” you will see scripts and folders of scripts that are part of your profile.

There is a folder called “SQL4DS” with all scripts for this presentation



PostgreSQL (Database)

- Do you have connection credentials?
- Can you confirm connection via “nc” or “telnet”?

DBeaver (SQL Client)

- Installed?
- Do you have project or connection profile?
- Can you connect to database?
- Can you see scripts? Do they execute?

The basic outline of a SQL query is SELECT, FROM, WHERE

```
SELECT
    productid
  , name
  , listprice
FROM product
WHERE productnumber = 'LO-C100'
;
```

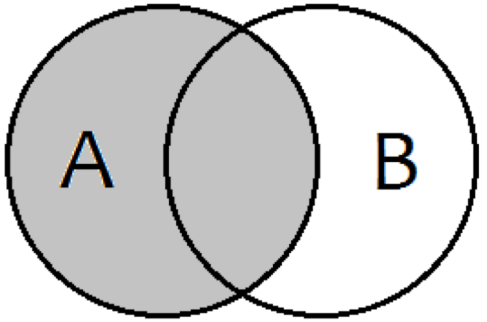
SQL is declarative, just tell the database what you want! But… it can get messy so format your queries to be easy to read.

See the SQL Style Guidelines Cheatsheet

A “JOIN” connects data from 2 or more tables (similar to VLOOKUP)

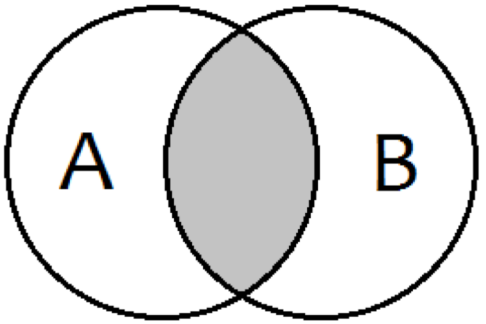
```
SELECT COUNT(*) -- 504 records
FROM product p
LEFT JOIN production.productcategory as pc
  ON p.productsubcategoryid = pc.productcategoryid
;
```

LEFT JOIN



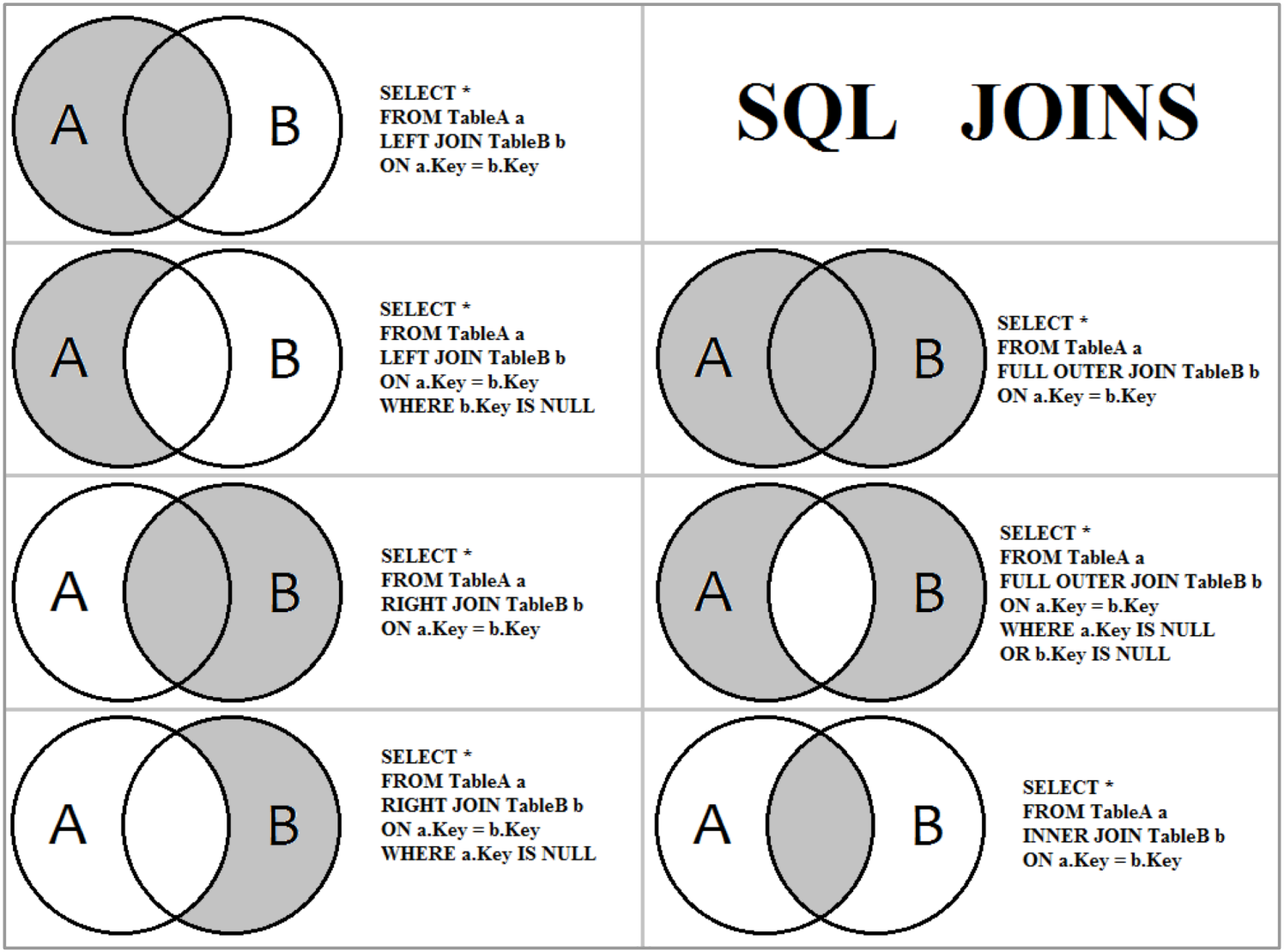

```
SELECT COUNT(*) -- 105 records
FROM product p
INNER JOIN productcategory as pc
  ON p.productsubcategoryid = pc.productcategoryid
;
```

INNER JOIN



The SQL Basics

The JOINs cheatsheet



Find Out Count of Transactions per Month

The GROUP BY statement

```
SELECT   p.name
       , COUNT(*) AS total
FROM transactionhistory th
INNER JOIN product p
  ON th.productid = p.productid
WHERE TO_CHAR(transactiondate, 'YYYY-MM') = ‘2017-01'
GROUP BY p.name
;
```

Count by month for all months in 2017.
Don’t copy/paste the query above! GROUP BY month!

```
SELECT   TO_CHAR(th.transactiondate, 'YYYY-MM') AS month
       , p.name
       , COUNT(*) AS total
FROM transactionhistory th
INNER JOIN product p
  ON th.productid = p.productid
WHERE DATE_PART('year', transactiondate) = ‘2017'
GROUP BY month, p.name
;
```

Find Out Distribution of Products by Category

Divide by a total count to get the proportion of total

```
SELECT    pc.name AS category
                , COUNT(p.productid)
                , CAST(COUNT(p.productid) AS decimal) / (
                SELECT COUNT(*) FROM production.product) AS pct
FROM product p
INNER JOIN productcategory pc
  ON p.productsubcategoryid = pc.productcategoryid
GROUP BY category
;
```

Be careful! Always inspect your results!!

```
SELECT    pc.name AS category
                , COUNT(p.productid)
                , CAST(COUNT(p.productid) AS decimal) / (
                SELECT COUNT(*) FROM production.product) AS pct
FROM product p
LEFT JOIN productcategory pc
  ON p.productsubcategoryid = pc.productcategoryid
GROUP BY category
;
```

Recoding a Column with CASE

The CASE statement

"CASE" follows *WHEN-THEN-ELSE* logic to change values, here we rename bikes and components to "Core".

```
SELECT
    p.name
  , CASE
          WHEN pc.name = 'Bikes' OR pc.name = 'Components' THEN
'Core'
          ELSE 'Non-Core'
    END AS finance_category
FROM product p
LEFT JOIN productcategory pc
  ON p.productsubcategoryid = pc.productcategoryid
;
```

Find Out What Percent of Products are “Core”

Convert a column to 0s and 1s and then the average is the proportion of 1s!

```
SELECT
    AVG(core_indicator) AS pct_core
FROM product p
LEFT JOIN (
 SELECT
      productcategoryid
    , CASE
           WHEN name = 'Bikes' OR name = 'Components' THEN 1
           ELSE 0
      END AS core_indicator
  FROM productcategory
) pc
  ON p.productsubcategoryid = pc.productcategoryid
;
```

Note: The product table is LEFT JOIN’ed to a “subquery”. A subquery is its own query that is part of a larger query.

Using Temp Tables

Smaller, separate tables to make large queries more manageable

```
WITH t AS (
  SELECT   productid
         , TO_CHAR(transactiondate, 'YYYY-MM') AS month
  FROM transactionhistory
  WHERE DATE_PART('year', transactiondate) = ‘2017'
)

, t2 AS (
  SELECT t.*, p.name
  FROM t
  INNER JOIN product p
    ON t.productid = p.productid
)

SELECT month, name, COUNT(*) AS total
FROM t2
GROUP BY month, name
ORDER BY month, name
;
```

Find Out the Percent Change Month over Month

The LAG() function

```
, monthly AS (
  SELECT month, COUNT(*) AS total
  FROM t2
  GROUP BY month
  ORDER BY month
)
, lagged AS (
  SELECT
      month
    , total
    , LAG(total) OVER (ORDER BY month) AS prior_month
  FROM monthly
)

SELECT month, total, prior_month,
       (total::float - prior_month) / prior_month AS pct_diff
FROM lagged
;
```

Manipulate Strings and Find via Fuzzy Match

Text columns can be modified and searched as needed

```
SELECT
    name AS original
  , LENGTH(name) AS str_length
  , UPPER(name) AS str_upper
  , SUBSTRING(name, 1, 3) AS str_sub
  , REPLACE(name, 'a', 'zzz') AS str_replace
  , CONCAT('PREFIX_', name) AS str_concat
FROM product
WHERE productnumber LIKE 'LO%'
;
```

LENGTH, UPPER, SUBSTRING, REPLACE, CONCAT will modify a column. LIKE will perform a fuzzy match

Resources

Follow these links for more information on learning SQL

W3Schools - http://www.w3schools.com/sql/

Hacker Rank - https://www.hackerrank.com/domains/sql/select

SQL Zoo - http://sqlzoo.net/wiki/AdventureWorks

Codecademy - https://www.codecademy.com/learn/learn-sql

Other Data Science Club Members

Experienced Professionals are all around you!

Questions?

In Person - Ask Now!

Via Email -

Club Inbox - DataScienceClub@darden.virginia.edu

Steve Mortimer - MortimerS19@darden.virginia.edu


DATA
SCIENCE
CLUB