W02D1

SQL Intro

Instructor: Eric Elmoznino

Outline for today

- Context and landscape (40 mins)
- Break (10 mins)
- Demo of simple queries
    - Filtering, ordering, limiting, etc.
    - Joining tables
    - Grouping records
    - Aggregate functions

Why databases?

- Enforce strict structure and relationships (forces you to keep data clean)
    - Makes it easier to train Machine Learning algorithms
- Store massive amounts of data
- Efficient retrieval of data
- Enables data governance (e.g. availability, usability, integrity, and security)

A company cannot be managed with Excel spreadsheets

Why SQL?

- Most common format for data storage/retrieval in enterprise
- Still most proficient tool to investigate, filter, slice, and dice your data

NoSQL databases

- Stored in formats other than relational tables, or retrieved in other ways
- MongoDB (JSON-like): https://www.mongodb.com/nosql-explained

Why are ***we*** learning SQL?

- Public dataset format (less common use-case)
- Internal database querying
    - May contain information you don’t want
    - May want to train your model on combinations/aggregations of fields from various tables
- Data exploration through simple operations on different groups/subsets
- Better understand transformations on data
- Mentioned in almost every data science job posting
    - Data science jobs ask for a ton of skills, good to have exposure to all for interview purposes, even if they won’t all realistically be necessary

Challenges when writing SQL

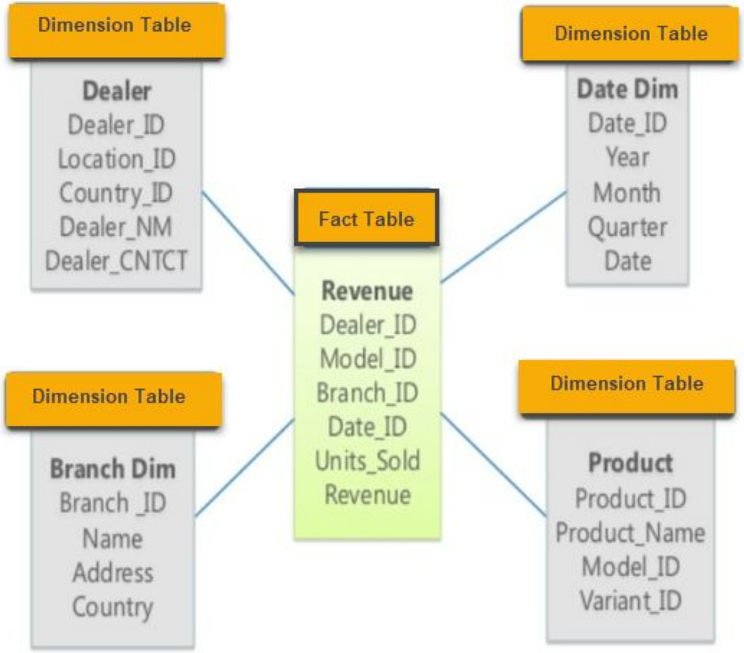
- Declarative (SQL) vs. imperative (Python). Basically, no control flow
- Long, nested queries with many variable names
  - Vs. imperative programming where most readable programs break logic up into multiple steps
- Many things happening concurrently in a single statement, order not explicit
  - Vs. imperative programming where code executes line-by-line
- Debugging is more difficult due to the above
  - Can help to break a complex query down into steps and test those out first. Incremental approach to writing the query
- To review the fundamentals: https://www.w3schools.com/sql/

Database schemas

- How do the different tables relate to each other?
- Arguably most difficult part of relational databases is designing the schema
    - Less of a concern for data scientists — not our job!
- For our purposes, useful to understand table structure of a database to know how to write our queries (e.g. what tables to join)
- Common design principles
    - Star schema
    - Snowflake shema

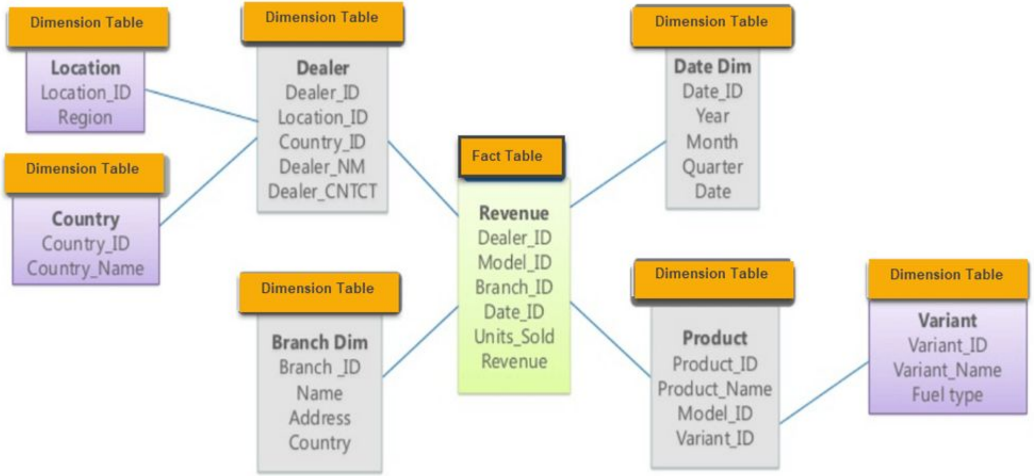
Star schema

- Every dimension represented by only one dimension table
- Dimension table contains set of attributes
- Dimension tables are joined to the fact table using a foreign key
- Dimension tables **are not** joined to each other
- Fact table contains key and measure



Snowflake schema

- Nested version of star
- Dimensions attributes can also be complex entities
- Make dedicated tables for the sub-entities and reference with foreign key
- Normalized (efficient, non-repeating) database



Dimension Table

Location
Location_ID
Region

Dimension Table

Country
Country_ID
Country_Name

Dimension Table

Dealer
Dealer_ID
Location_ID
Country_ID
Dealer_NM
Dealer_CNTCT

Dimension Table

Branch Dim
Branch _ID
Name
Address
Country

Fact Table

Revenue
Dealer_ID
Model_ID
Branch_ID
Date_ID
Units_Sold
Revenue

Dimension Table

Date Dim
Date_ID
Year
Month
Quarter
Date

Dimension Table

Product
Product_ID
Product_Name
Model_ID
Variant_ID

Dimension Table

Variant
Variant_ID
Variant_Name
Fuel type

RDBMS Landscape

- Software system that enables users to define, create, maintain and control access to the database
- Closed source (i.e. paid)
  - Vendors: Oracle, SQL Server (Microsoft), IBM DB2, Microsoft Access - local small databases
  - Could come with integrations and services that make things easier
- Open source
  - MySQL, PostgreSQL, SQLite, MariaDB
  - Good developer community makes these great options
  - This website offers a good comparison of open source systems options.

Why SQLite?

- Not directly comparable to client/server SQL database engines such as MySQL, Oracle, PostgreSQL, or SQL Server
- Used as on-disk file format for desktop applications
- No concurrency (multiple users accessing simultaneously)
- Great to learn on to get a hang of SQL

Why PostgreSQL (postgres)?

- Open source nature makes it easy to upgrade or extend
- High compliance to the SQL standard
- Easily runs on Windows, Mac OS X, and almost all Linux distributions
- MySQL would be a good choice too (less compliance to SQL standard)
- We will use psql to make/interact with databases (terminal application)

Demo

- https://github.com/EricElmoznino/lighthouse_sql_tutorial