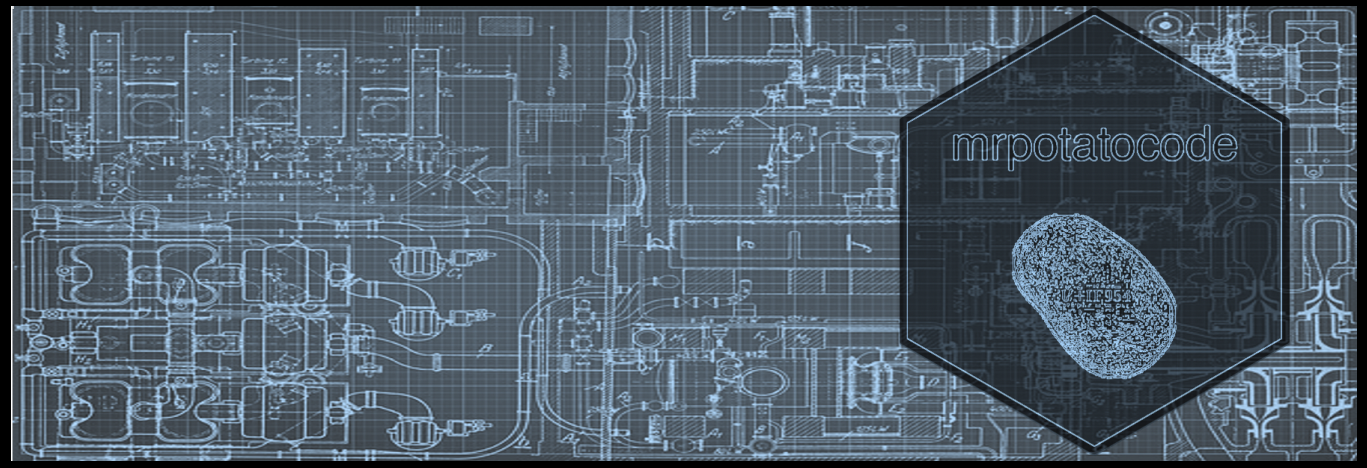



# Introduction to Data Access and Storage

Thomas Rosenthal – DSI @ UofT

Module 01

# Getting Started:

Welcome

Introduction

First Three Commands

# Getting Started:

Welcome

Introduction

First Three Commands

# Welcome

About Me

Course Content

Quick Technical Check

# Welcome

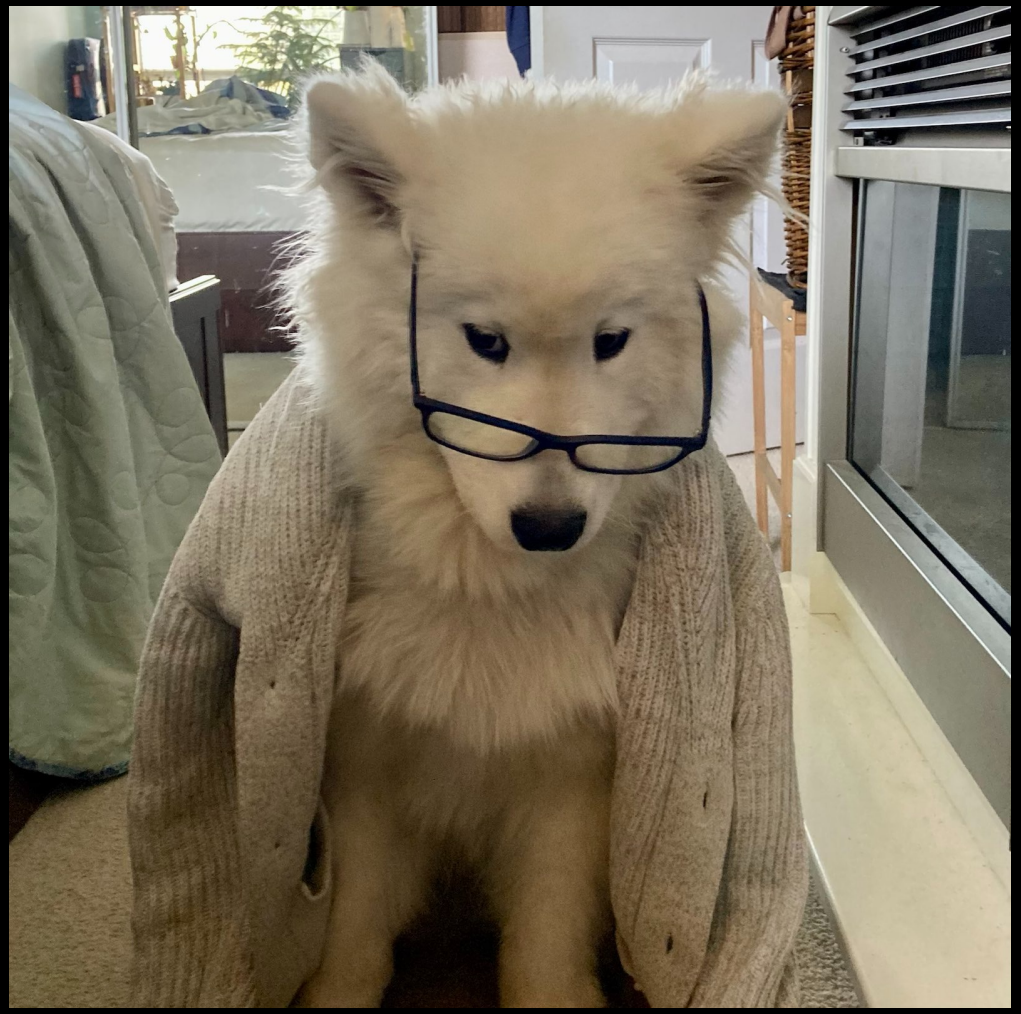
About Me

Course Content

Quick Technical Check

# About Me

- Fell into SQL in my first job out of undergraduate in 2013
- Worked as a Data Analyst in US healthcare until moving to Canada in 2018
  - Challenging data
  - Easy to see impact of good queries
  - I was pretty naive about data, probably lucky I didn't break anything 🤷‍♀️
- Worked as a Data Engineer at Plan Canada managing a CRM Data Warehouse backend
- Went back to UofT to do my Master of Information
  - Wanted to do less SQL, more R and python
  - SQL seemed like it was on its way out...NOPE, I was just wrong
- Currently doing Data Ethics implementation at Thomson Reuters
- My dog's name is Alto, and she's pretty wonderful



# Welcome

About Me

Course Content

Quick Technical Check

# Course Content

- Data Modelling
- Basic SQL Syntax
- Advanced SQL Syntax
- Data Structures, Forms, and Basic Data Management
- Importing and Exporting Data to and from SQL

- Data Stewardship, Ethics, and SQL in the Wild

- This course is an *Introduction* to SQL

- At the end of the course, I hope you will:
    - Feel comfortable with SQL
    - Know how to search for the right thing on Stack Overflow
    - Read documentation

- We won't cover advanced topics like:
    - Stored Procedures, Triggers, Jobs
    - DBA work (monitoring, server setup, etc)
    - Complex ETL or tooling

# Welcome

About Me

Course Content

Quick Technical Check

# Quick Technical Check

(a slide where we make sure everyone has DB Browser for SQLite installed)

If not, please download it here: sqlitebrowser.org/dl/

For live coding:

- Please download/fork the FarmersMarket.db from my GH:
  - github.com/mrpotatocode/DSI_SQL/tree/main/SQL
- Open it in SQLite with the "Open Database" button and navigate to wherever you have saved it

Good to go?

# Getting Started:

Welcome

Introduction

First Three Commands

# Introduction

SQL

Flavours

Data Modelling

# Introduction

SQL

Flavours

Data Modelling

# SQL

## What is SQL?

- SQL: Structured Query Language
  - Pronounced as either S.Q.L. (ess-cue-ell) or “sequel”
- SQL is a *query* language rather than a programming language
  - Code is often less reproducible than programming languages because it's domain specific
  - Querying
- Designed to manage data within Relational Database Management Systems (RDBMs), e.g.
  - MSSQL
  - Oracle DB
  - MySQL/MariaDB
  - PostgreSQL

# SQL

## SQL Formatting

- Like other programming/query languages, SQL has reserved keywords/commands to perform instructional operations
    - Generally, these keywords are written in all caps: `SELECT`
    - Most modern interpreters no longer require this, but it is the expected standard
- All statements/queries should be ended with a semicolon
    - A few SQL constructs (like common table expressions, we'll get to these later) require them, otherwise they are optional
        - I'll almost certainly forget to use them
        - There's some debate on whether or not it's best practice
- In SQL, white space does not matter

Introduction

SQL

Flavours

Data Modelling

# Flavours

- RDBMs differ from one to the next. This is manifested in many ways:
  - different keywords
    - e.g. return only 10 rows: `SELECT TOP 10...` vs `SELECT ... LIMIT 10`
  - different syntax
    - e.g. not equal: `!=` or `<>` (or both)
  - other, more nuanced/complex differences
    - e.g. architecture, data types, etc

- We are using SQLite:
  - Super easy to get setup
  - Requires almost no overhead
  - Open source, *free*
  - Similar enough in syntax to learn on

Introduction

SQL

Flavours

Data Modelling

# Data Modelling

## What are Relational Database Management Systems?

- Relational Database Management Systems (RDBMs) are software designed to:
    - Store large amounts of data
    - Utilize a query language to allow easy retrieval of the data
    - Allow multiple users to access the data simultaneously
    - Manage permissions for data access
    - Mitigate data corruption and unauthorized access
- Generally, data is stored in a *database*
    - a database is a collection of information
    - within a database, a collection of objects (e.g. tabular data "tables") is stored
- RDBMs allow users to define interactions between these objects, such as:
    - Establish the relationship between objects
    - Define procedural scripts to query specific data or trigger an action
    - Schedule routine work (e.g. procedures to run, maintenance, etc)

# Data Modelling

## What is a Data Model?

- A data model is a notation for describing data or information
- Data models consist of:
  - Structure of the data
  - Operations
  - Constraints on the data
  - Relationships

# Data Modelling

## Structure of Data

- SQL is comprised of tables

| Breed | Affectionate w/ Family | Good w/ Other Dogs | Shedding | Coat Type | Coat Length | Playfulness | Energy |
|---|---|---|---|---|---|---|---|
| Pugs | 5 | 4 | 4 | Smooth | Short | 5 | 3 |
| Akitas | 3 | 1 | 3 | Double | Medium | 3 | 4 |
| Samoyeds | 5 | 3 | 3 | Double | Long | 5 | 4 |

- Tables have Attributes and Observations
  - In SQL we call Attributes "Columns"
    - e.g. Breed, Coat Type, Coat Length
  - and Observations "Rows"
    - e.g. Samoyed, Double, Long

- SQL databases require tables to be named
  - e.g. We can call this table "breed_traits"

# Data Modelling

## Structure of Data

- Columns are defined (and restricted, i.e. constrainted) by data types
- Common ones include:
  - `INT` (integers)
  - `FLOAT`, `DECIMAL`, `REAL` (decimal)
  - `VARCHAR`, `NVARCHAR`, `TEXT` (text strings, with a maximum length associated)
  - `DATE`, `DATETIME`, `TIME` (dates and times)
- These may vary slightly by flavour (in SQLite they are simpler and less restricted)
- Data types are important:
  - they affect operation speed, storage size, data validity

# Data Modelling

## Constraints

- Data Models also specify constraints
- Constraints are rules that must be followed:
  - Referential-Integrity constraints
    - Ensure that values in one table have corresponding values in another table
  - Attribute Constraints
    - Ensure that certain types of values are always consistent within columns
    - May also ensure whether values are unique, not missing, etc

# Data Modelling

## Relationships: Entity Relationship Diagrams (ERDs)

- ERDs are diagrams depicting the structure of tables within database
    - This both *identifies the tables* and *describes their relationships*
- ERDs are useful for:
    - database design
    - debugging
    - writing logical, consistent, and efficient queries
- There three levels of details for ERD depictions:
    - Conceptual model
    - Logical model
    - Physical model

# Data Modelling

## Attributes of an ERD Entity

- For a given table:
  - Name
  - Relationship to another table
  - Column Names
  - Column Types
  - Primary Keys (if present)
  - Foreign Keys (if present)

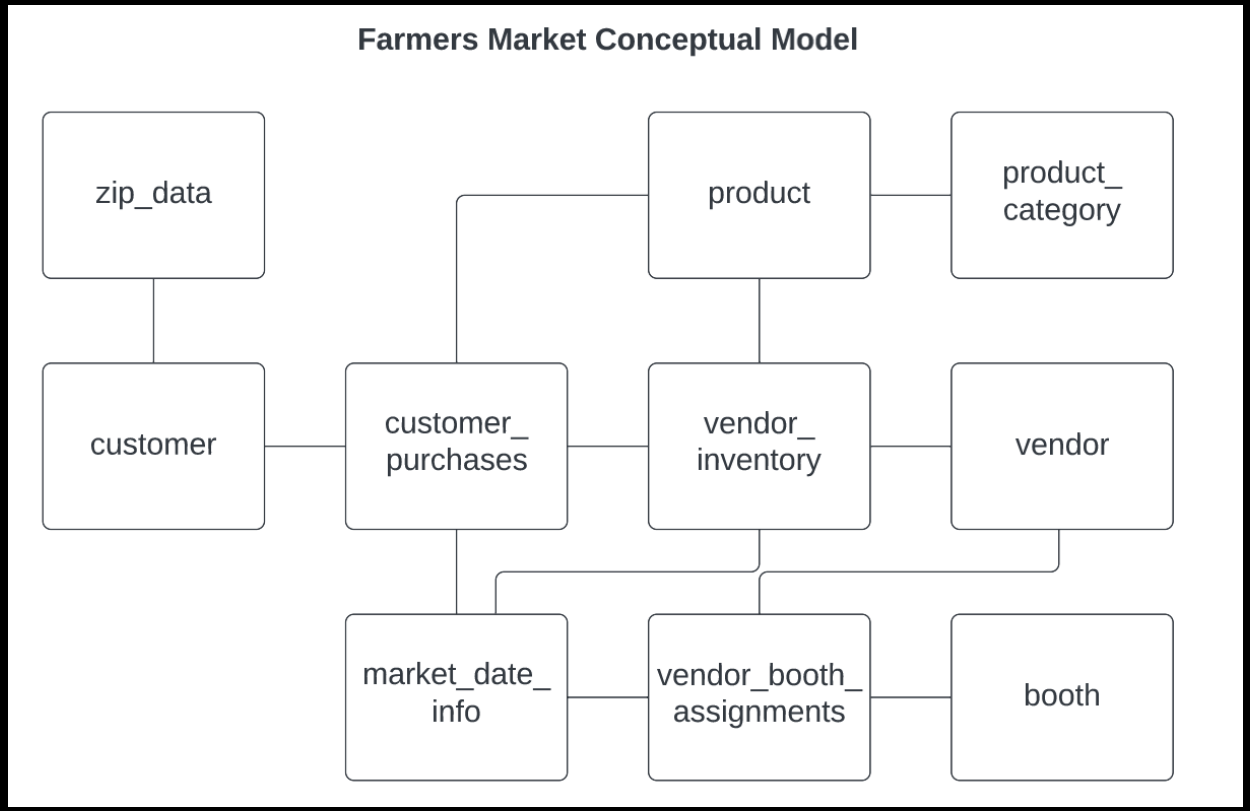
## Attributes of a Relationship

- Defines which columns are related
- Defines what type of relationship exists:
  - One-to-One
  - One-to-Many
  - Many-to-Many

# Data Modelling

## Conceptual Models

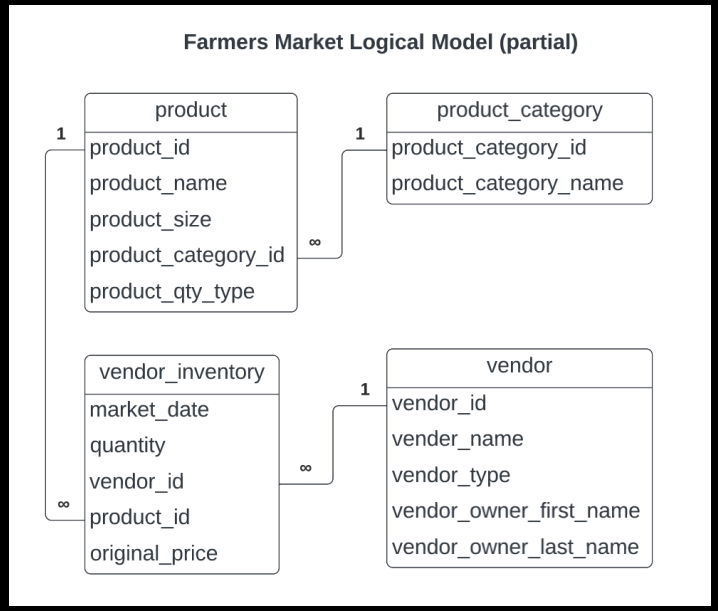
- Defines the tables (objects/entities) and their relationships



# Data Modelling

## Logical Models

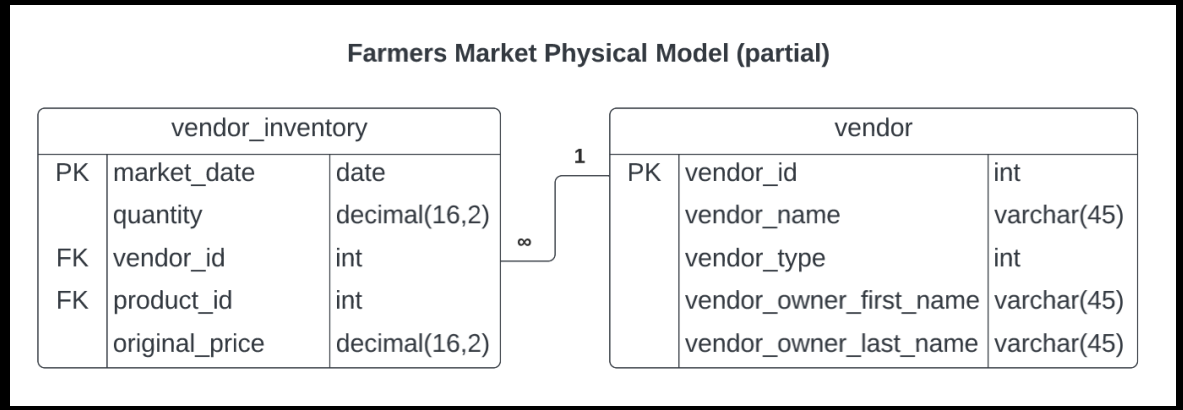
- Adds additional detail to the conceptual model by adding column names for each table
- Often indicates the type of relationship
  - One-to-One: where a given row within a table is associated with only a single row in another table
    - Table 1: Country — Table 2: Capital City
  - One-to-Many: where a given row within a table can be referenced by multiple rows in another table
    - Table 1: Customer — Table 2: Customer Orders
  - Many-to-Many: where multiple rows within a table can be referenced by multiple rows in another table
    - Table 1: Employee — Table 2: Employee Type
    - A is a professor at DSI *and* a TA at DSI. If a table contained Employee ID and Employee Type ID, A would be listed twice.
    - Even more of a many-to-many because A is part of two courses!



# Data Modelling

## Physical Models

- Adds additional detail to the logical model by adding key type and column data type
  - Primary Key (PK)
    - Ensures each value in a column is unique within the table (e.g. an ID field)
    - One PK per table
    - Cannot be NULL
    - Ensures database integrity by restricting record deletion
  - Foreign Key (FK)
    - Creates a linkage between a column in one table and a column in another table
      - Generally, foreign keys are linked to primary keys
    - As many FKs as needed per table
    - May be NULL
    - Record can be deleted



# Getting Started:

Welcome

Introduction

First Three Commands

# First Three Commands

SELECT

FROM

WHERE

# First Three Commands

- Our first three commands (`SELECT`, `FROM`, `WHERE`) are essential to nearly every SQL query
- The template for our initial SQL statement is as such:

  `SELECT` : *the columns we want to retrieve*

  `FROM` : *the table we are querying*

  `WHERE` : *filters/conditions (optional)*

  `ORDER BY` : *column sorting: ascending* `ASC` *or descending* `DESC` *(optional)*

  `LIMIT` : *how many rows we want to return (optional)*

# First Three Commands

- Always specified in this order:
  - `SELECT` will come first
  - `FROM` will come after `SELECT`
    - when we are querying more than one table at a time, each will come after `FROM` but before `WHERE` (more on this later)
  - `WHERE` will come after `FROM`
  - `ORDER BY` will come after `WHERE` clauses
- We'll sometimes use the `LIMIT` clause to look at data
  - This comes at the very end of a query
  - `LIMIT` shouldn't be used for analytics unless you have a specific reason
  - `ORDER BY` often impacts the usefulness of `LIMIT`
- One more thing:
  - In SQL, we use two dashes `--` to comment out lines, rather than `#`

# First Three Commands

SELECT

FROM

WHERE

# SELECT

- At its simplest `SELECT` specifies column names we are retrieving
  - commas come between each column name
    - `SELECT student, course, grade ...`
  - column names with a space need to be enclosed in square brackets
    - `SELECT [poorly named column], better_column_name, AnotherColumnName`
- Within `SELECT` statements we can perform manipulations on columns
  - e.g. rename a column
    - `SELECT [poorly named column] AS better_col`
  - combine two text columns
  - perform math on a numeric column
  - ...and many more things
- We can use select to perform math without a FROM statement
  - `SELECT 1 + 1`
  - `SELECT 10*5, cos(2), pi()`
- When selecting columns, they need to exist in the table!

# First Three Commands

SELECT

FROM

WHERE

# FROM

- `FROM` statements indicate which table the data and where the table is located
  - in more complicated RDBMs, you will often have multiple databases on the same server and multiple schema within those databases
  - a fully qualified location of a table would thus be `database.schema.table`
- `SELECT * FROM table_name` indicates *everything* in the table
- Best practice suggests that we should explicitly call each column, even if we want all of them
  - **Why do we think this is the case?**

# FROM

(`SELECT` & `FROM` live coding)

# First Three Commands

SELECT

FROM

WHERE

# WHERE

- `WHERE` clauses are conditions that the query will follow
- When we want to have multiple conditions, we use a single `WHERE` and then additional logical operations

```
SELECT *
FROM students
WHERE first_name = 'Thomas'
AND last_name = 'Rosenthal'
```

- Notice we put string values in single quotes
    - SQLite also allows double quotes, with a few minor caveats
- `WHERE` clauses always return rows evaluating to `TRUE`
    - Follows Boolean rules if more than one condition is present
- My favourite WHERE statement is `WHERE 1=1`
    - Any guesses as to why?

# WHERE

## Logical Operators

- `AND`
- `OR`
- `NOT`
- `NOT IN`
- equals: `=`
- does not equal: `<>` `!=`
    - (flavour dependent)
- greater than (equal to): `>` `>=`
- less than (equal to)`<` `<=`
- `BETWEEN`
- `EXISTS`
    - table specific
- `IS`
    - `NULL` specific

# WHERE

## NULL

- `NULL` is not a value (it's the absence of a value)
  - to check null values, we use `IS NULL` or `IS NOT NULL`
  - `= NULL` will not work

## LIKE

- `LIKE` allows for string wildcards
- `%` specifies the wildcard placement
  - `country_name LIKE 'and%'`
    - Andorra
  - `country_name LIKE '%and'`
    - Finland, Iceland ...more
  - `country_name LIKE '%and%'`
    - all of the above, *plus* Antigua and Barbuda, Netherlands, Rwanda ...more!
  - `country_name LIKE '%an%d%'`
    - Canada ...surely more!

# WHERE

(`WHERE` live coding)

Lingering Questions?