

# Introduction to Data Access and Storage

Thomas Rosenthal - DSI @ UofT

Module 01

# Introduction:

Welcome

What is SQL?

Data Modelling

# Getting Started:

Welcome

What is SQL?

Data Modelling

# 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
- 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
- Reluctant Data Scientist (*do we even need this model?*)
  - Currently doing Data Governance and Ethics implementation at Thomson Reuters
- Have a 2 year old Samoyed named Alto who takes up a lot of my free time 👏



# About Me

- Recently graduated from UofT with majors in Computer Science and Statistics, Minor in Mathematics
  - Held co-op positions at different banks
    - Citi (Software Development)
    - Bank of Montreal(Business Analytics)
- Now working full-time at RBC Capital Markets in their Quant and Tech Services division
  - Currently a Business Systems Analyst (position is a cross between SDE and BA)
  - Working with salesforce CRM backend data in SQL server
  - Moving to a more technical role in a new team (Supervisory and Regulatory controls) next week (for my next rotation)
- Outside of work, strongly believe in advocating for the representation of women in tech
  - Was the president of Women in Computer Science at UofT for 2 years
  - Now getting involved with Toronto Wxmen in Data Science and RWomen (RBC's women in tech ERG)



# Welcome

## About Me

## Course Content

## Quick Technical Check

# Course Content

This is the shortest version of this course I've taught. We will try and cover a lot but also focus on the key components of SQL querying.

- Data Modelling, Schema Design, Data Structures
- Basic SQL Syntax
- Essential SQL Syntax
- Advanced Techniques, including importing and Exporting Data to and from SQL

## Github Repo

[https://github.com/UofT-DSI/02-intro\\_sql](https://github.com/UofT-DSI/02-intro_sql)

- Schedule
- These slides (HTML & PDF)
- Our database for live coding
- All in-class code
- Homework (and answers, posted afterwards)
- Assignment details and rubrics
- Policies, due dates, etc

**It is crucial you visit the repo throughout the course, as I may make minor changes and push new content.**

# Course Content

- 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



The screenshot shows the SQLite Database Browser interface. The main window displays a list of employees from a database. The columns shown are Employee ID, First Name, Last Name, and Hire Date. The data includes rows for employees like '48571 Alana Mccormick' and '48572 Jennifer Wicks'. The browser's toolbar at the top includes icons for file operations, zoom, and search. The status bar at the bottom indicates the current database is 'Northwind'.

```
1 select *  
2 from employee  
3 where city = 'Redmond'  
4 and state = 'WA'  
5 and type = 'Customer'  
6 -485100128 customer Security storage shows that there were 8 winsores.  
7 -The first winsore lives at the last house in 'Northwestern Dr'.  
8 -The second winsore, named Accidental, lives somewhere in 'Picado Ave' - Redmond  
9  
10 select * from person  
11 where address_number = 1  
12 select address_number from person  
13 where address_street_name like '%Northwestern Dr%'  
14 order by address_number desc  
limit 1  
15 -48571 Accidental Miller 4857179 100 Picado Ave 328791248  
16 -48577 Murphy Schapira 1180009 4858 Northwestern Dr 328948940  
  
17 select * from customer  
18 where person_id is 48571  
19 -I want a person who has a customer with '48571'. Only good customers have been kept.  
20 -The customer number is 10 which matches with '48571'. Only good customers have been kept.  
21 -The customer number is 10 which matches with '48571'. Only good customers have been kept.  
22  
23  
24  
25  
26  
27 where id like '48571'  
28 and constituency_address = 'grid'  
29 -48571A 080009 Jive Democats 000000000 grid  
30 -48571B 070009 Jiving Demos 000000000 grid  
  
31 select * from person  
32 where location_id IS NULL  
33 select id from address_location  
34 where place_number like '485100128'  
  
35 -485798 Tidwell Obscuris 084790100 Phi Dc 12779886975  
36 -485828 Jiving Demos 00000007 0000 Washington Pl, Apt 3A 07105560779  
37 -485838 Maurice Whistley 1887779 100 Park Rd 0779886975  
  
38  
39  
40 select * from customer  
41 where person_id = 48571  
42 -48571 I was hired by a winsore with a lot of money
```

**Software:**  
**DB Browser for SQLite**

# Course Content

## Homework (required)

- Four homeworks total
- At the end of each SQL module, I will provide you with 3-10 queries to write on your own
  - Covers the topics of the modules
- Review answers in Tutorials with Ananya
  - I may go over one or two queries at the start of class if desired
- Designed to be relatively easy
  - Reaffirms what we wrote together
  - Doing work on your own helps reinforce the learning
- ChatGPT probably won't help you much

# Course Content

## Grading

- Pass/Fail. Do the work, pass the course :)
- One Assignment: Data Model Design, 50%, due one week from today (Jan 17)
- Homework: 40% (10% each), due the day after each module
- Class Attendance: 10%
  - Let Ananya or myself know if you are unable to attend a lesson
  - Code along!! Best way to learn.



# 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: <https://sqlitebrowser.org/dl/>

For live coding:

- Please download/fork the FarmersMarket.db from our GH repo:
  - [https://github.com/UofT-DSI/02-intro\\_sql/tree/main/SQL](https://github.com/UofT-DSI/02-intro_sql/tree/main/SQL)
- Open it in SQLite with the "Open Database" button and navigate to wherever you have saved it

Good to go?

If not, please message Ananya

# Getting Started:

## Welcome

## What is SQL?

## Data Modelling

# What is SQL?

## SQL

## Flavours

## Environments for SQL

# What is SQL?

## SQL

## Flavours

## Environments for SQL

# SQL

## SQL Fundamentals

- 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
  - Querying is closer to telling a computer *what you want*, rather than *what is has to do*
  - SQL code is often less reproducible than other programming languages because it's domain specific
    - Some SQL code, especially more advanced procedural code, is reproducible within the same flavour
  - SQL's domain is databases and is based on set theory
- 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 end 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 over whether or not it's best practice

# SQL

## SQL Formatting

- In SQL, white space and/or line breaks do not matter
  - Readability is important
  - Try to keep SQL statements to a reasonable screen width
  - Use sensible line breaks
  - Offset subqueries with indents
- Code is commented in/out with `--` rather than `#`
- Code blocks can be commented out with `/* */`

```
/*
somecode spanning
multiple lines
*/
```

# What is SQL?

SQL

Flavours

## Environments for SQL

# Flavours

- RDBMs differ from one to the next:
  - 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
  - Used all over the world and in many applications
    - e.g. Firefox uses a SQLite backend to write a user's history locally

# Flavours

- Broad observations about Open Source systems:
  - Excellent at what they are designed for
  - Varying data types (SQLite has some unique ones!)
  - Not every command exists, but workarounds are usually possible
  - Some utilize RDBMs that feel extremely outdated
- Broad observations about enterprise systems:
  - Powerful and designed to handle edge cases
  - Feel a bit more refined
    - Can be version dependent
  - Tend to "lock in" businesses/organizations
    - Migration is costly, sometimes outrageously so
  - Newer players (Snowflake, Databricks, etc) and cloud providers (Azure, AWS, GCP, etc) offer a lot more functionality than just database querying
    - Sometimes use different terminology to describe SQL tasks

# What is SQL?

## SQL

## Flavours

## Environments for SQL

# Environments for SQL

## Databases

- Relational databases are a collection of tables, views, procedural code, and other SQL-assisting artefacts
  - Generally the data stored in a database will be related to a real-world concept
  - Backends to data-collecting systems are often databases
    - e.g. CRMs, EMR software, ERPs, web-based applications
  - Usually not connected to other databases unless deemed necessary
  - Often transactional, meaning data is actively being written to by frontend systems
  - Tables are normalized
- There are also non-relational databases, often referred to as NoSQL
  - We won't cover these
  - Common tools include: Amazon DynamoDB, Azure CosmosDB, MongoDB, Google Cloud Datastore

# Environments for SQL

## Data Warehouses and Data Marts

- Data Warehouses are highly structured collections of (usually tabular) data
  - Data has been processed for a specific purpose, e.g. analytics
  - Data has been centralized
    - Often with the assistance of ETL (Extract, Transform, Load) tools
  - Have a lot of overhead, require governance, and strict rigidity
  - Tables are denormalized
  - Very common for enterprises, but losing traction in many industries
- Data Marts are created from Data Warehouses to focus on a single subject
  - Designed to make Data Warehouses easier to use
  - Data is structured, but flexibility is driven by the purpose of the Data Mart
  - Some denormalized tables may be normalized or undergo even greater normalization
    - *The subject/purpose of the Data Mart might drive these types of decisions*
  - Common for enterprises that have Data Warehouses

# Environments for SQL

## Data Lakes and Data Swamps

- Data Lakes allow on-demand access to raw, semi-structured, structured, and unstructured data
  - Not defined by a specific purpose
  - Highly scalable
  - Can be transactional, if systems are designed to produce outputs into Data Lakes
  - Often data sources for machine learning pipelines live in Data Lakes
  - Inexpensive compared to Data Warehouses
  - Increasingly common for enterprises to shift towards Data Lakes, especially with support from newer tools like Snowflake, Databricks, etc, which can maximize the analytical value of a Data Lake
- Data Swamps...are poorly governed Data Lakes
  - Lack of documentation, lack of governance, poorly designed Data Lakes become Swamps
  - Avoid building these



# Getting Started:

Welcome

What is SQL?

Data Modelling

# Data Modelling

Relational Database Management Systems

Data Models

Structure of Data

Constraints

(...)

# Data Modelling (continued)

Entity Relationship Diagrams

Attributes of an ERD: Entities & Relationships

Relationship Examples

Conceptual, Logical, Physical Models

Assignment 1: Design a Logical Model

# Data Modelling

## Relational Database Management Systems

Data Models

Structure of Data

Constraints

(...)

# Data Modelling

## 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

Relational Database Management Systems

Data Models

Structure of Data

Constraints

(...)

# Data Modelling

## Data Models

- 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

Relational Database Management Systems

Data Models

Structure of Data

Constraints

(...)

# 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. constrained) by data types
- Common ones include:
  - **INT** (integers: 1,2,3,-1,-2,-3)
    - most systems conserve storage space specifying their range
  - **FLOAT, DECIMAL, REAL** (decimal: 5.5, 3.333333)
  - **VARCHAR, NVARCHAR, TEXT** (text strings, with a maximum length associated: 'abc')
  - **DATE, DATETIME, TIME** (dates and times: '2023-01-09', '11:11:11.000')
- 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
    - Speed: it's computationally less expensive to compute smaller values
    - Storage: small is usually better, but the wrong size will affect systems; e.g. Psy's Gangnam Style exceeded 2,147,483,647 (32-bit signed,  $2^{32}/2-1$ ) views, causing YouTube to expand the view counter to 64-bit (~9.2b if signed)
    - Validity: ensures columns contain the right type of data for operations, e.g. avoiding  $5 + \text{'ten'} = ??$



# Data Modelling

Relational Database Management Systems

Data Models

Structure of Data

Constraints

(...)

# 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

## Constraints

- **NULL** and **NOT NULL**
  - If a value can be missing or not
- **UNIQUE**
  - All values are different
- **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
    - Sometimes share the same name as the linked column, but this isn't required
    - Linkage requires data types to be the same
  - As many FKS as needed per table
  - May be NULL
  - Record can be deleted

# Data Modelling (continued)

## Entity Relationship Diagrams

Attributes of an ERD: Entities & Relationships

Relationship Examples

Conceptual, Logical, Physical Models

Assignment 1: Design a Logical Model

# Data Modelling

## Entity Relationship Diagrams

- Entity Relationship Diagrams (ERDs) are diagrams depicting the structure of tables within a database
  - This both *identifies the tables* and *describes their relationships*
- ERDs are useful for:
  - Database design
  - Debugging
  - Writing logical, consistent, and efficient queries
- There are three levels of detail for ERD depictions:
  - Conceptual model
  - Logical model
  - Physical model

# Data Modelling (continued)

## Entity Relationship Diagrams

**Attributes of an ERD: Entities & Relationships**

Relationship Examples

Conceptual, Logical, Physical Models

Assignment 1: Design a Logical 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 an ERD Relationship

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

# Data Modelling (continued)

## Entity Relationship Diagrams (ERDs)

Attributes of an ERD: Entities & Relationships

### Relationship Examples

Conceptual, Logical, Physical Models

Assignment 1: Design a Logical Model

# Data Modelling

## Relationship Examples

**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

Table 1:Country	Table 2:Capital
Canada	1:1 Ottawa
USA	1:1 Washington DC
Mexico	1:1 Mexico City

# Data Modelling

## Relationship Examples

**One-to-Many:** where a given row within a table can be referenced by multiple rows in another table

Table 1: Country – Table 2: States

Table 1:Country	Table 2:States
Canada	1: $\infty$ Alberta
Canada	1: $\infty$ British Columbia
Canada	1: $\infty$ ... (11 more rows)
USA	1: $\infty$ Alabama
USA	1: $\infty$ Alaska
USA	1: $\infty$ ... (48 more rows)
Mexico	1: $\infty$ Aguascalientes
Mexico	1: $\infty$ Baja California
Mexico	1: $\infty$ ... (30 more rows)

# Data Modelling

## Relationship Examples

**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

For this example, imagine the following:

Previously, A was an instructor at DSI (python) *and* a TA at DSI (SQL). Thomas remains the instructor for SQL. Ananya is now the TA. If a table contained Employee ID and Employee Type ID, A would be listed twice.

Employee	Employee ID	Type	Type ID
Thomas	001	Instructor	10
A	002	TA	11
Ananya	003		

Employee ID	Type ID
001	∞:∞ 10
002	∞:∞ 10
002	∞:∞ 11
003	∞:∞ 11

We'd create additional Many-to-Many relationships if we created a course table, because A and Ananya are involved in multiple courses!



# Data Modelling (continued)

## Entity Relationship Diagrams

Attributes of an ERD: Entities & Relationships

Relationship Examples

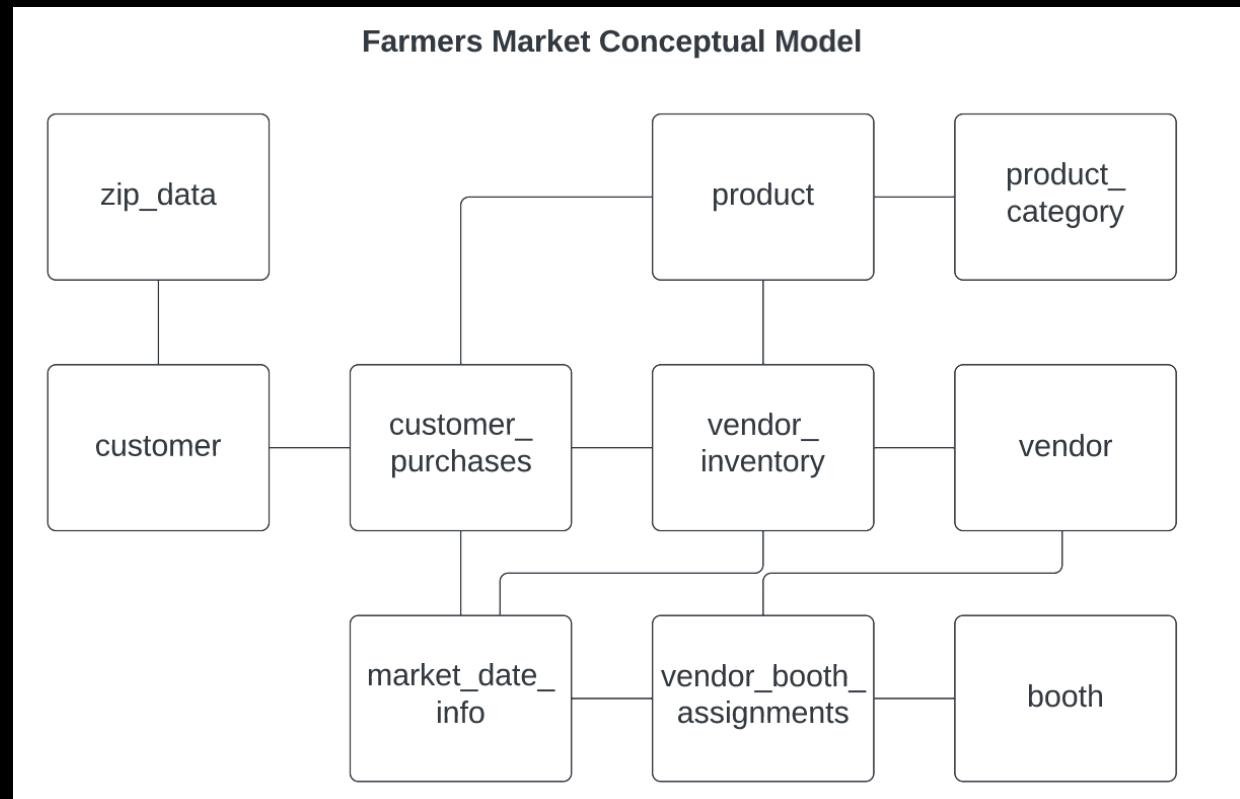
Conceptual, Logical, Physical Models

Assignment 1: Design a Logical Model

# Data Modelling

## Conceptual Models

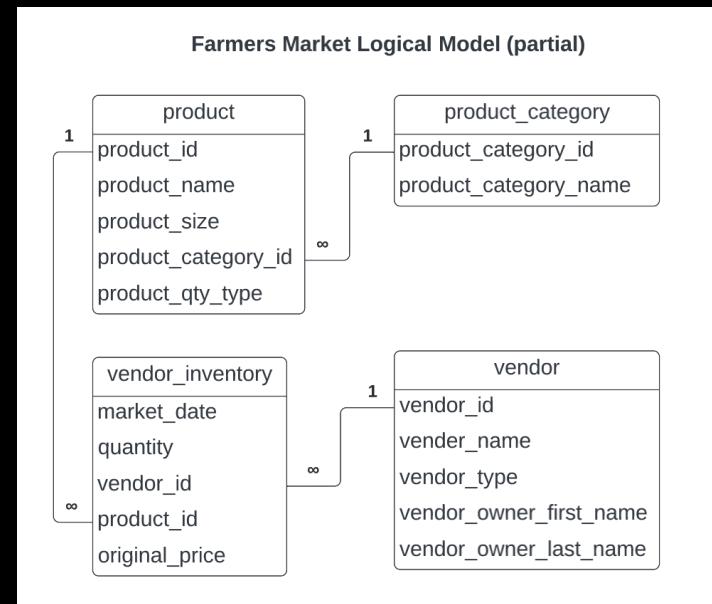
- Define the tables (objects/entities) and their relationships
- Our Farmers Market database:
  - 10 tables
  - Relationships between these tables
    - e.g. product and product\_category: *what type of thing a product is*
    - product and customer\_purchases: *what products a customer has bought*
    - product and vendor\_inventory: *what products each vendor has available*
  - Not all tables share a relationship to one another, but all tables have at least one relationship



# Data Modelling

## Logical Models

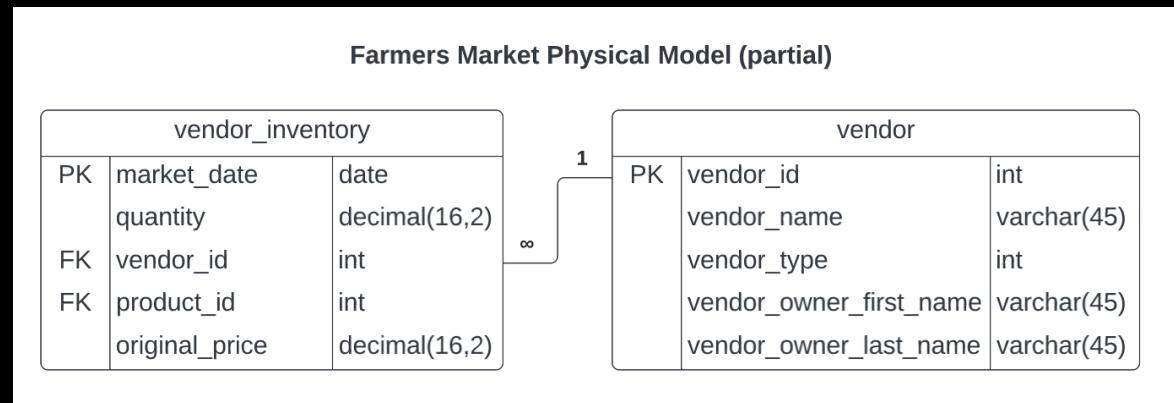
- Add additional detail to the conceptual model by adding column names for each table
- Often indicate the type of relationship
  - One-to-One
  - One-to-Many
  - Many-to-Many
- Our (partial) Farmers Market database:
  - product (5 columns) shares a One-to-Many relationship with vendor\_inventory (5 columns) on product\_id
  - product\_category (2 columns) shares a One-to-Many relationship with product (5 columns) on product\_category\_id
  - \_\_\_\_\_ shares a \_\_\_\_\_ relationship with \_\_\_\_\_ on vendor\_id



# Data Modelling

## Physical Models

- Add additional detail to the logical model by adding key type and column data type
- Our (partial) Farmers Market database:
  - vendor\_id (int) is the PK for vendor, which shares a One-to-Many relationship with vendor\_inventory on vendor\_id (FK)
  - product\_id (int) is a FK for vendor\_inventory (*so elsewhere in this diagram, we'd connect this to a PK of another table*)
  - market\_date (date) is the PK for vendor\_inventory
    - why?





# Data Modelling (continued)

## Entity Relationship Diagrams

Attributes of an ERD: Entities & Relationships

Relationship Examples

Conceptual, Logical, Physical Models

Assignment 1: Design a Logical Model

# Data Modelling

## Assignment 1: Design a Logical Model

**Q1)** Create a logical model for a small bookstore. 

At the minimum it should have employee, order, sales, customer, and book entities (tables). Determine sensible column and table design based on what you know about these concepts. Keep it simple, but work out sensible relationships to keep tables reasonably sized. Include a date table. There are several tools online you can use, I'd recommend [Lucidchart](#).

**Q2)** We want to create employee shifts, splitting up the day into morning and evening. Add this to the ERD.

**Q3)** The store wants to keep customer addresses. Propose two architectures for the CUSTOMER\_ADDRESS table, one that will retain changes, and another that will overwrite. Which is type 1, which is type 2? *Hint, search type 1 vs type 2 slowly changing dimensions.*

Bonus: Are there privacy implications to this, why or why not?

**Q4)** Review the AdventureWorks ERD here: [link](#)

Highlight at least two differences between it and your ERD. Would you change anything in yours?

