

# Workshop

## Introducing SQL: A Foundation of Data Analytics

Robb Sombach  
University of Alberta  
Alberta School of Business



# Agenda

- Introduction
  - Why SQL?
  - What about Python? R?
  - Data Analytics
- Relational Database
  - What is a database?
  - Terminology
  - SQLite
  - **Exercise 1**
- SQL
  - Data Definition Language (DDL)
  - **Exercise 2**
  - Data Manipulation Language (DML)
  - **Exercise 3**
- Open Data Portal
  - How I prepared for today

# Robb Sombach

- Work Experience
  - 15+ years working in the IT industry
  - 10+ years Self-Employed IT Consultant
- IT Positions
  - Systems Analyst / Business Analyst
  - Database Administrator (Oracle / SQL Server)
  - Network Administrator
  - Developer

# Robb Sombach

- Teaching Experience
  - 5 years teaching at NAIT
    - Computer Systems Technology (CST)
    - Digital Media and Information Technology (DMIT)
  - 6+ years teaching at University of Alberta
    - Technology Training Centre
    - Alberta School of Business

# Resources

All Workshop files can be downloaded here

**[http://bit.ly/odd\\_2019](http://bit.ly/odd_2019)**

# Introduction

Workshop

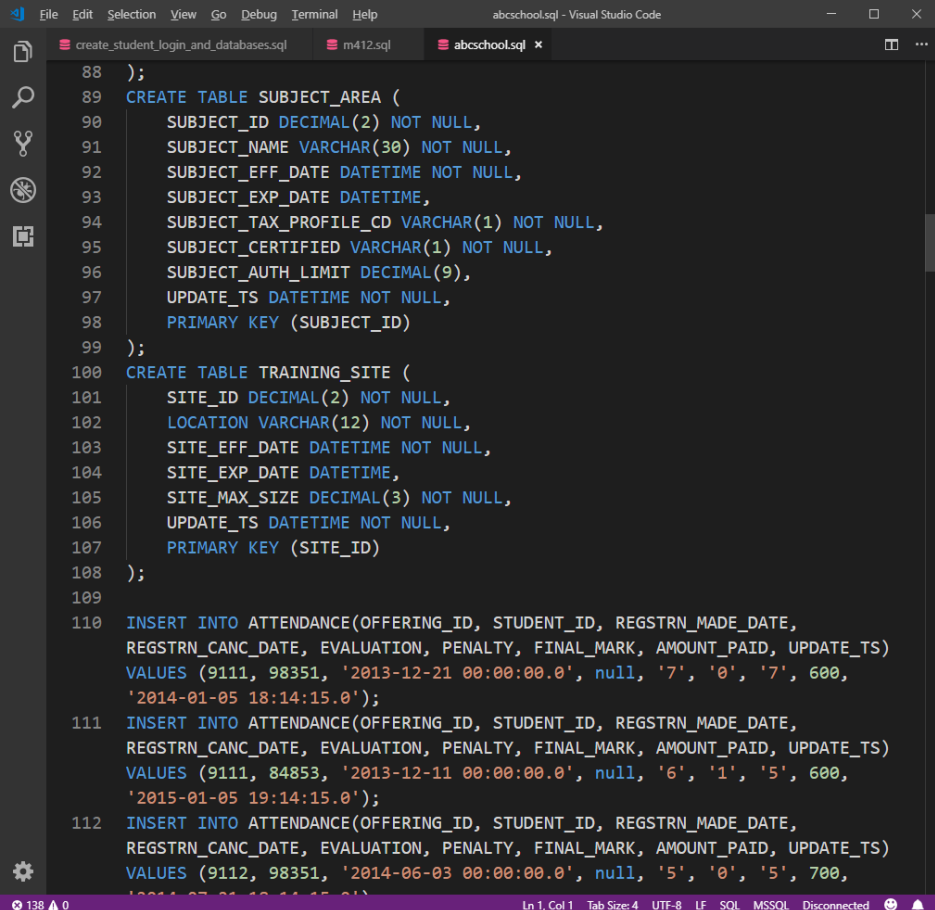
Introducing SQL: Foundation of Data Analytics

# Goals

- Introduce relational database concepts
- Provides hands-on, real world database experience using data from the City of Edmonton Open Data Portal
- Foster a collaborative workshop
  - Please interrupt and ask questions

# Why SQL?

- Simple
- Accessible
- Applicable
- Powerful
- Pervasive
- Valuable
- Universal



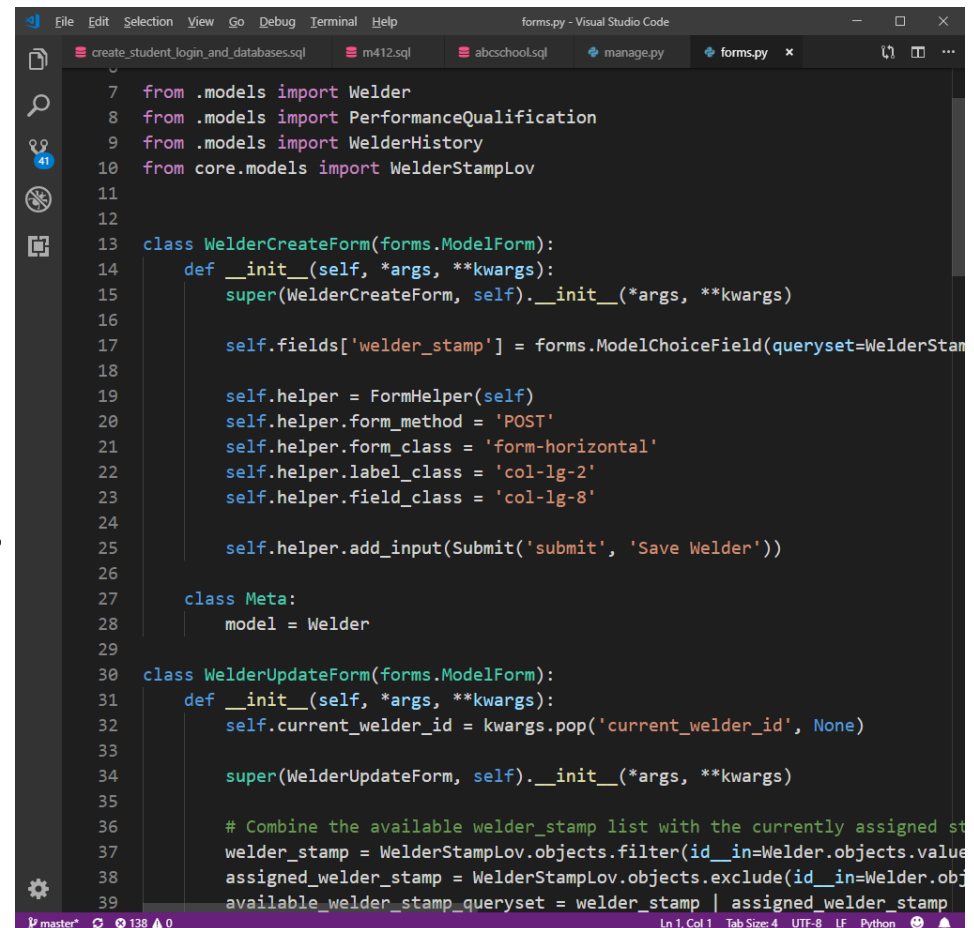
```

88 );
89 CREATE TABLE SUBJECT_AREA (
90     SUBJECT_ID DECIMAL(2) NOT NULL,
91     SUBJECT_NAME VARCHAR(30) NOT NULL,
92     SUBJECT_EFF_DATE DATETIME NOT NULL,
93     SUBJECT_EXP_DATE DATETIME,
94     SUBJECT_TAX_PROFILE_CD VARCHAR(1) NOT NULL,
95     SUBJECT_CERTIFIED VARCHAR(1) NOT NULL,
96     SUBJECT_AUTH_LIMIT DECIMAL(9),
97     UPDATE_TS DATETIME NOT NULL,
98     PRIMARY KEY (SUBJECT_ID)
99 );
100 CREATE TABLE TRAINING_SITE (
101     SITE_ID DECIMAL(2) NOT NULL,
102     LOCATION VARCHAR(12) NOT NULL,
103     SITE_EFF_DATE DATETIME NOT NULL,
104     SITE_EXP_DATE DATETIME,
105     SITE_MAX_SIZE DECIMAL(3) NOT NULL,
106     UPDATE_TS DATETIME NOT NULL,
107     PRIMARY KEY (SITE_ID)
108 );
109
110 INSERT INTO ATTENDANCE(OFFERING_ID, STUDENT_ID, REGSTRN_MADE_DATE,
111 REGSTRN_CANC_DATE, EVALUATION, PENALTY, FINAL_MARK, AMOUNT_PAID, UPDATE_TS)
112 VALUES (9111, 98351, '2013-12-21 00:00:00.0', null, '7', '0', '7', 600,
113 '2014-01-05 18:14:15.0');
114
115 INSERT INTO ATTENDANCE(OFFERING_ID, STUDENT_ID, REGSTRN_MADE_DATE,
116 REGSTRN_CANC_DATE, EVALUATION, PENALTY, FINAL_MARK, AMOUNT_PAID, UPDATE_TS)
117 VALUES (9111, 84853, '2013-12-11 00:00:00.0', null, '6', '1', '5', 600,
118 '2015-01-05 19:14:15.0');
119
120 INSERT INTO ATTENDANCE(OFFERING_ID, STUDENT_ID, REGSTRN_MADE_DATE,
121 REGSTRN_CANC_DATE, EVALUATION, PENALTY, FINAL_MARK, AMOUNT_PAID, UPDATE_TS)
122 VALUES (9112, 98351, '2014-06-03 00:00:00.0', null, '5', '0', '5', 700,
123 '2014-07-01 10:14:15.0');
  
```



# Why not Python? R?

- Difficult for beginners
- Complicated syntax
- Requires programming knowledge (logic, algorithms)
- Is SQL better than Python or R?
  - SQL is good for some things
  - Python/R is good for other things
  - Compliment each other
- SQL is a great starting point



```
File Edit Selection View Go Debug Terminal Help forms.py - Visual Studio Code
create_student_login_and_databases.sql m412.sql abcschool.sql manage.py forms.py x
7 from .models import Welder
8 from .models import PerformanceQualification
9 from .models import WelderHistory
10 from core.models import WelderStampLov
11
12
13 class WelderCreateForm(forms.ModelForm):
14     def __init__(self, *args, **kwargs):
15         super(WelderCreateForm, self).__init__(*args, **kwargs)
16
17         self.fields['welder_stamp'] = forms.ModelChoiceField(queryset=WelderStampLov.objects.all())
18
19         self.helper = FormHelper(self)
20         self.helper.form_method = 'POST'
21         self.helper.form_class = 'form-horizontal'
22         self.helper.label_class = 'col-lg-2'
23         self.helper.field_class = 'col-lg-8'
24
25         self.helper.add_input(Submit('submit', 'Save Welder'))
26
27     class Meta:
28         model = Welder
29
30 class WelderUpdateForm(forms.ModelForm):
31     def __init__(self, *args, **kwargs):
32         self.current_welder_id = kwargs.pop('current_welder_id', None)
33
34         super(WelderUpdateForm, self).__init__(*args, **kwargs)
35
36         # Combine the available welder_stamp list with the currently assigned stamp
37         welder_stamp = WelderStampLov.objects.filter(id__in=Welder.objects.values('welder_stamp'))
38         assigned_welder_stamp = WelderStampLov.objects.exclude(id__in=Welder.objects.values('welder_stamp'))
39         available_welder_stamp_queryset = welder_stamp | assigned_welder_stamp
```

# Data Analytics

- Analytics is the discovery, interpretation, and communication of meaningful patterns in **data**; and the process of applying those patterns towards effective decision making
- Organizations may apply analytics to business **data** to describe, predict, and improve business performance
  - <https://en.wikipedia.org/wiki/Analytics>

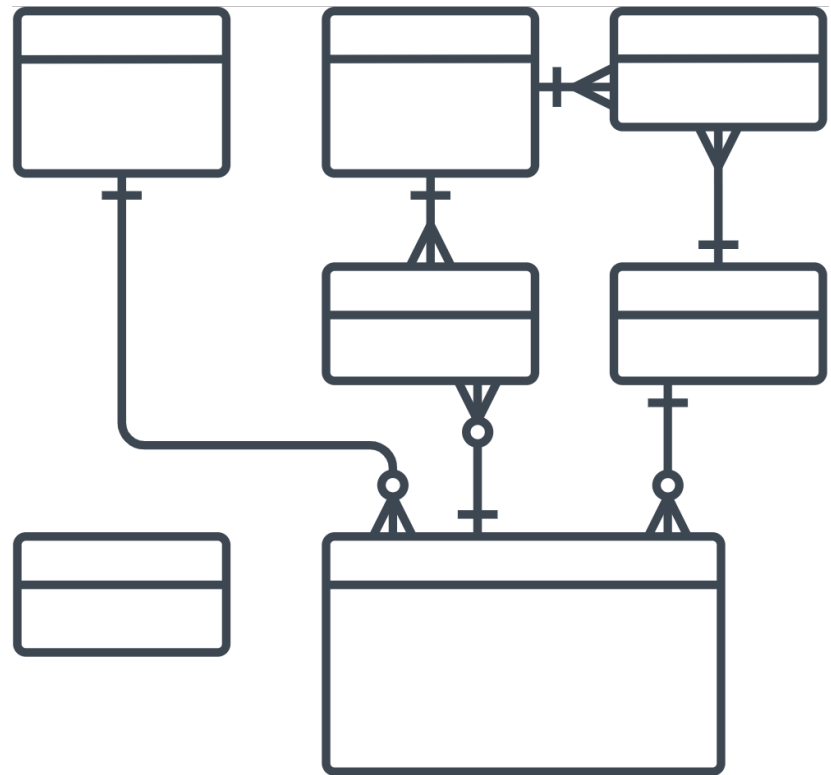
# Relational Database

Workshop

Introducing SQL: Foundation of Data Analytics

# What is a database?

- A relational “database” management system (RDBMS) organizes **data**
- The logical structure of the database is based upon the information needs of an organization
  - Entities (“things” of interest to the organization),  
AND
  - Relationships (how the Entities are associated with each other)



# Advantages of a RDBMS

- Establish a centralized, logical view of data
- Minimizes data duplication (i.e. “redundancy”)
- Promote data accuracy and integrity
- Capacity of database
- Superior multi-user or concurrent access
- Security
- Retrieve information quickly
- Inter-operability



<https://www.bespokesoftwaredevelopment.com/blog/advantages-database-development-business/>

# Database Terminology

- **Table**, Entity, Relation, (similar to an Excel Worksheet)
- **Row**, Record, Instance
- **Column**, Field, Attribute
- **Primary Key** – unique and mandatory
- **Foreign Key** – a cross-reference between tables because it references the primary key of another table
- **Relationship** – created though foreign keys



# How to introduce SQL?

- Microsoft Access
  - <https://products.office.com/en-ca/access>
- Microsoft SQL Server
  - <https://www.microsoft.com/en-us/sql-server/sql-server-2017>
- MariaDB, MySQL
  - <https://mariadb.org/>
  - <https://www.mysql.com/>
- Postgresql
  - <https://www.postgresql.org/>
- Oracle
  - <https://www.oracle.com/database/>
- Hadoop, Spark, Hive, Pig
  - <https://hadoop.apache.org/>



# A database that ...

- Has full-featured SQL
- Has billions and billions of deployments
- Is a single-file database
- Has public domain source code
- Small footprint
- Has a max DB size of 140 terabytes
- Has a max row size of 1 gigabyte
- Is faster than direct file access
- Aviation-grade quality and testing
- Zero-configuration
- Has ACID (Atomic, Consistent, Isolated, and Durable) transactions, even after power loss
- Has a stable, enduring file format
- Is has extensive, detailed documentation
- Has long-term support (to the year 2050)

<https://www.sqlite.org/about.html>



# SQLite

- “SQLite is the most widely deployed database in the world with more applications than we can count, including several high-profile projects”
  - <https://www.sqlite.org/famous.html>
- “SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine”
  - <https://www.sqlite.org/about.html>
- Perfect for learning SQL (the foundation of data analytics)

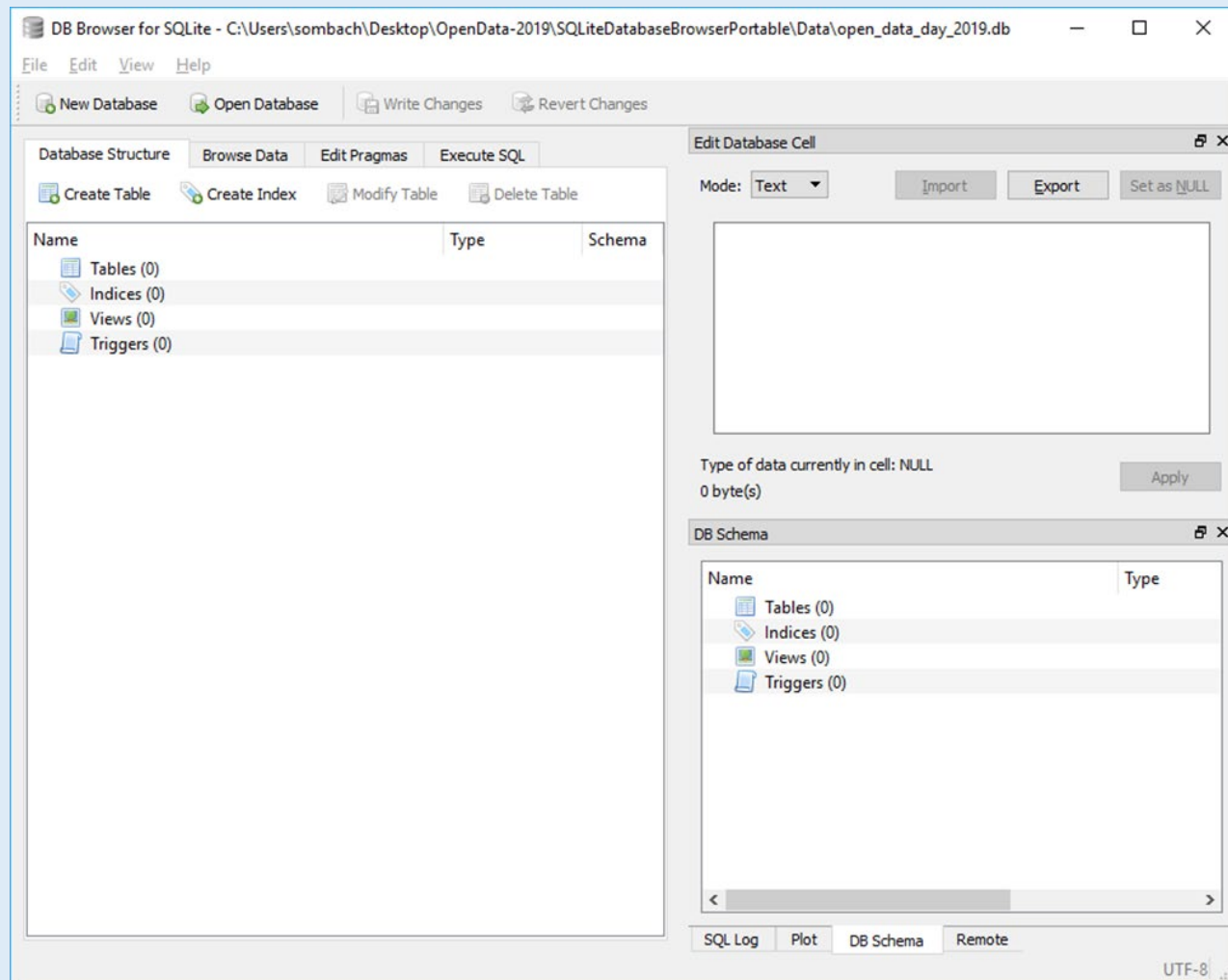
# Exercise 1: Download and Run SQLite BD Browser

- Download SQLite
- Download SQLite DB Browser Portable
  - <https://sqlitebrowser.org/dl/>

# Exercise 1: Download and Run SQLite

- Extract the ZIP archive to the Desktop
- Start SQLite
  - SQLiteDatabaseBrowserPortable.exe
- Create a New database
  - open\_data\_day\_2019.db
- Save the database in the Data folder
- Click Cancel when prompted to create a table
- Done!

# Exercise 1: Completed



# SQL

Workshop

Introducing SQL: Foundation of Data Analytics

# What is SQL?

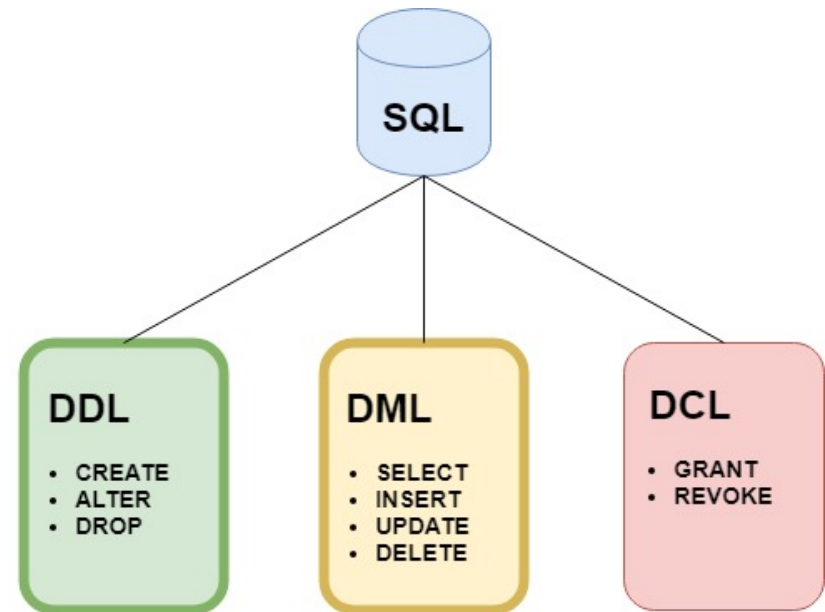
- SQL stands for Structured Query Language
  - SQL is pronounced S-Q-L or seque<sup>l</sup>
  - SQL is a standard language for managing, manipulating and querying databases
  - Developed at IBM in the early 1970's
  - In 1986, ANSI and ISO standard groups officially adopted the standard "Database Language SQL" definition
  - Most SQL databases have their own proprietary extensions in addition to the SQL standard
- SQL is the language used to ask questions (query) of a database which will return answers (results)

# Why is SQL the foundation of Data Analytics?

- Data engineers and database administrators will use SQL to ensure that everybody in their organization has access to the data they need
- Data scientists will use SQL to load data into their models
- Data analysts will use SQL to query tables of data and derive insights from it

# Components of SQL

- SQL consists of three components which offer everything required to manage, maintain and use a database
  1. Data Definition Language
  2. Data Manipulation Language
  3. Data Control Language





# Data Definition Language (DDL)

- This component is used to define the structure (or schema) of the database
- For tables there are three main commands:
  - **CREATE TABLE table\_name**
    - To create a table in the database
  - **ALTER TABLE table\_name**
    - To add or remove columns from a table in the database
  - **DROP TABLE table\_name**
    - To remove a table from the database

# Exercise 2: Data Definition Language

- Select the Execute SQL tab in SQLite
- Type or copy/paste the CREATE TABLE statement into the empty SQLite Execute SQL window
- Click the **Execute SQL** ► button on the toolbar
- If the table is created successfully, you should receive the following message:
  - Query executed successfully: CREATE TABLE "MOSQUITO\_TRAP\_DATA"
- Click Write Changes to make commit the changes permanent
- View the changes in the Database Structure tab

```
CREATE TABLE "MOSQUITO_TRAP_DATA" (  
  `SAMPLEID` INTEGER PRIMARY KEY AUTOINCREMENT,  
  `TRAP_DATE` NUMERIC,  
  `GENUS` TEXT,  
  `SPECIES` TEXT,  
  `TYPE` TEXT,  
  `GENDER` TEXT  
);
```

# Exercise 2: Data Definition Language

- Select the Execute SQL tab in SQLite
- Type or copy/paste the ALTER TABLE statements into the empty SQLite Execute SQL window
- Click the **Execute SQL** ► button on the toolbar
- If the table is created successfully, you should receive the following message:
  - Query executed successfully: ALTER TABLE "MOSQUITO\_TRAP\_DATA"
- Click **Write Changes** to make commit the changes permanent
- View the changes in the **Database Structure** tab

```
ALTER TABLE "MOSQUITO_TRAP_DATA" ADD COLUMN `RURALSOUTHWEST` INTEGER;  
ALTER TABLE "MOSQUITO_TRAP_DATA" ADD COLUMN `RURALSOUTHWEST` INTEGER;  
ALTER TABLE "MOSQUITO_TRAP_DATA" ADD COLUMN `RURALSOUTHWEST` INTEGER;  
ALTER TABLE "MOSQUITO_TRAP_DATA" ADD COLUMN `RURALSOUTHWEST` INTEGER;  
ALTER TABLE "MOSQUITO_TRAP_DATA" ADD COLUMN `RURALSOUTHWEST` INTEGER;  
ALTER TABLE "MOSQUITO_TRAP_DATA" ADD COLUMN `RURALSOUTHWEST` INTEGER;  
ALTER TABLE "MOSQUITO_TRAP_DATA" ADD COLUMN `RURALSOUTHWEST` INTEGER;  
ALTER TABLE "MOSQUITO_TRAP_DATA" ADD COLUMN `RURALSOUTHWEST` INTEGER;  
ALTER TABLE "MOSQUITO_TRAP_DATA" ADD COLUMN `RURALSOUTHWEST` INTEGER;  
ALTER TABLE "MOSQUITO_TRAP_DATA" ADD COLUMN `RURALSOUTHWEST` INTEGER;  
ALTER TABLE "MOSQUITO_TRAP_DATA" ADD COLUMN `RURALSOUTHWEST` INTEGER;  
ALTER TABLE "MOSQUITO_TRAP_DATA" ADD COLUMN `RURALSOUTHWEST` INTEGER;
```

# Exercise 2: Data Definition Language

- Select the Execute SQL tab in SQLite
- Type or copy/paste the DROP TABLE statement into the empty SQLite Execute SQL window
- Click the **Execute SQL** ► button on the toolbar
- If the table is created successfully, you should receive the following message:
  - Query executed successfully: DROP TABLE "MOSQUITO\_TRAP\_DATA"
- Click Write Changes to make commit the changes permanent
- View the changes in the Database Structure tab

```
DROP TABLE "MOSQUITO_TRAP_DATA";
```

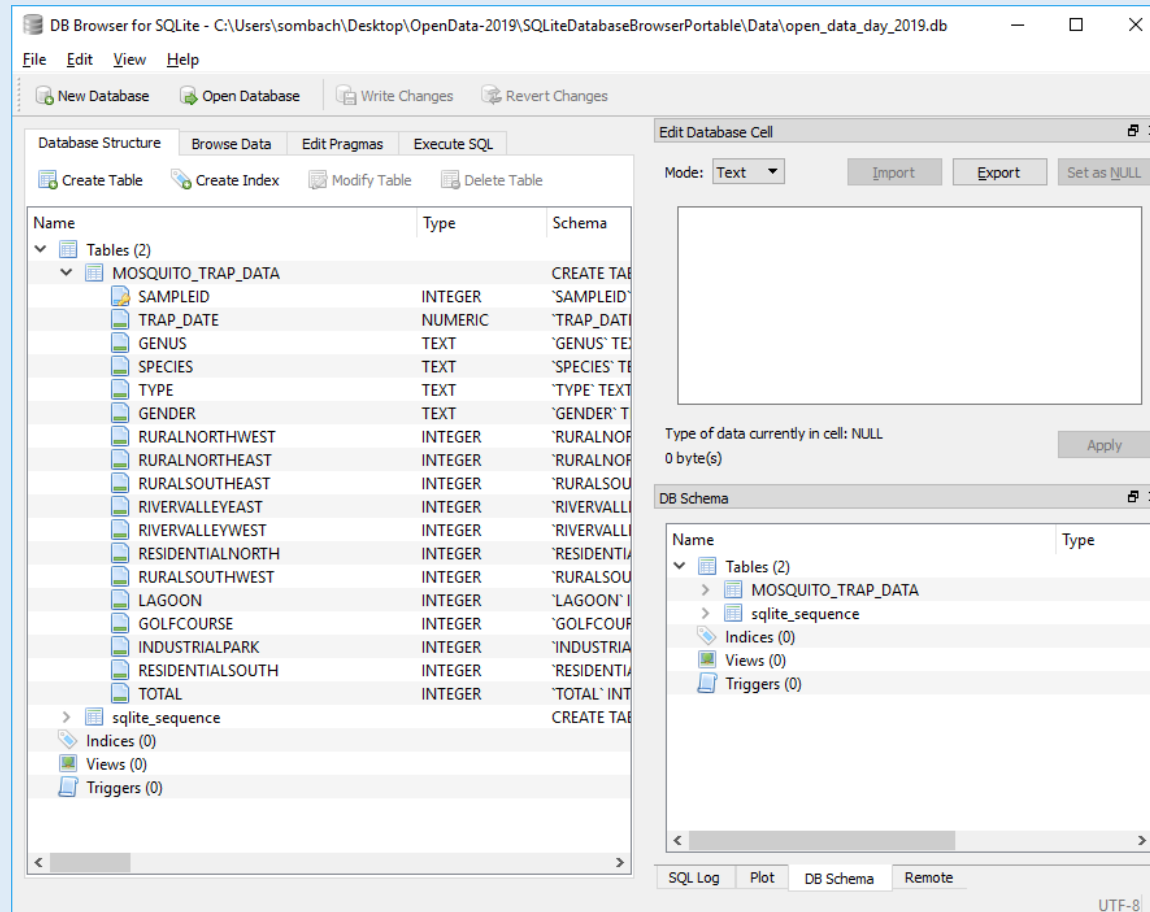
# Exercise 2: Data Definition Language

- Create the MOSQUITO\_TRAP\_DATA table again using the DDL on the next slide
- Click **Write Changes** to make commit the changes permanent
- View the changes in the Database Structure tab
- Done!



```
CREATE TABLE "MOSQUITO_TRAP_DATA" (  
  `SAMPLEID` INTEGER PRIMARY KEY AUTOINCREMENT,  
  `TRAP_DATE` NUMERIC,  
  `GENUS` TEXT,  
  `SPECIES` TEXT,  
  `TYPE` TEXT,  
  `GENDER` TEXT,  
  `RURALSOUTHWEST` INTEGER,  
  `RURALSOUTHEAST` INTEGER,  
  `RURALSOUTHWEST` INTEGER,  
  `RIVERVALLEYEAST` INTEGER,  
  `RIVERVALLEYWEST` INTEGER,  
  `RESIDENTIALNORTH` INTEGER,  
  `RURALSOUTHWEST` INTEGER,  
  `LAGOON` INTEGER,  
  `GOLFCOURSE` INTEGER,  
  `INDUSTRIALPARK` INTEGER,  
  `RESIDENTIALSOUTH` INTEGER,  
  `TOTAL` INTEGER  
)
```

# Exercise 1: Completed



# Data Manipulation Language

- This component is used to manipulate data within a table
- There are four main commands:
  - SELECT
    - To select rows of data from a table
  - INSERT
    - To insert rows of data into a table
  - UPDATE
    - To change rows of data in a table
  - DELETE
    - To remove rows of data from a table

# Exercise 3: SELECT

## Data Manipulation Language

- Select the Execute SQL tab in SQLite
- Type or copy/paste the SELECT statement into the empty SQLite Execute SQL window
  - `SELECT COUNT(*) FROM MOSQUITO_TRAP_DATA;`
- Click the **Execute SQL** ► button on the toolbar
- Do you get an answer? Why not?

# Exercise 3: INSERT

## Data Manipulation Language

- Add some data to the MOSQUITO\_TRAP\_DATA table created in Exercise 2
- Type or copy/paste the INSERT statement into the empty SQLite Execute SQL window
- Click the **Execute SQL** ► button on the toolbar
- Click **Write Changes** to make commit the changes permanent
- View the changes in the **Browse Data** tab
- The MOSQUITO\_TRAP\_DATA table now has seven rows of data

```

INSERT INTO "MOSQUITO_TRAP_DATA" (TRAP_DATE, GENUS, SPECIES, TYPE, GENDER, RURALNORTHWEST,
RURALNORTHEAST, RURALSOUTHEAST, RIVERVALLEYEAST, RIVERVALLEYWEST, RESIDENTIALNORTH,
RURALSOUTHWEST, LAGOON, GOLFCOURSE, INDUSTRIALPARK, RESIDENTIALSOUTH, TOTAL) VALUES ('2014-
07-01','Aedes','spencerii','Black legs','Female',0,0,0,0,0,1,0,0,0,1,1,3);
INSERT INTO "MOSQUITO_TRAP_DATA" (TRAP_DATE, GENUS, SPECIES, TYPE, GENDER, RURALNORTHWEST,
RURALNORTHEAST, RURALSOUTHEAST, RIVERVALLEYEAST, RIVERVALLEYWEST, RESIDENTIALNORTH,
RURALSOUTHWEST, LAGOON, GOLFCOURSE, INDUSTRIALPARK, RESIDENTIALSOUTH, TOTAL) VALUES ('2014-
07-01','Aedes','dorsalis','Banded legs','Female',0,1,0,0,0,0,2,0,0,0,0,3);
INSERT INTO "MOSQUITO_TRAP_DATA" (TRAP_DATE, GENUS, SPECIES, TYPE, GENDER, RURALNORTHWEST,
RURALNORTHEAST, RURALSOUTHEAST, RIVERVALLEYEAST, RIVERVALLEYWEST, RESIDENTIALNORTH,
RURALSOUTHWEST, LAGOON, GOLFCOURSE, INDUSTRIALPARK, RESIDENTIALSOUTH, TOTAL) VALUES ('2014-
07-01','Aedes','euedes','Banded legs','Female',1,1,0,0,2,0,0,0,0,0,0,4);
INSERT INTO "MOSQUITO_TRAP_DATA" (TRAP_DATE, GENUS, SPECIES, TYPE, GENDER, RURALNORTHWEST,
RURALNORTHEAST, RURALSOUTHEAST, RIVERVALLEYEAST, RIVERVALLEYWEST, RESIDENTIALNORTH,
RURALSOUTHWEST, LAGOON, GOLFCOURSE, INDUSTRIALPARK, RESIDENTIALSOUTH, TOTAL) VALUES ('2014-
07-01','Aedes','excrucians','Banded legs','Female',1,2,0,0,2,1,0,0,0,1,0,7);
INSERT INTO "MOSQUITO_TRAP_DATA" (TRAP_DATE, GENUS, SPECIES, TYPE, GENDER, RURALNORTHWEST,
RURALNORTHEAST, RURALSOUTHEAST, RIVERVALLEYEAST, RIVERVALLEYWEST, RESIDENTIALNORTH,
RURALSOUTHWEST, LAGOON, GOLFCOURSE, INDUSTRIALPARK, RESIDENTIALSOUTH, TOTAL) VALUES ('2014-
07-01','Aedes','fitchii','Banded legs','Female',0,2,0,0,1,0,0,0,0,0,0,4,7);
INSERT INTO "MOSQUITO_TRAP_DATA" (TRAP_DATE, GENUS, SPECIES, TYPE, GENDER, RURALNORTHWEST,
RURALNORTHEAST, RURALSOUTHEAST, RIVERVALLEYEAST, RIVERVALLEYWEST, RESIDENTIALNORTH,
RURALSOUTHWEST, LAGOON, GOLFCOURSE, INDUSTRIALPARK, RESIDENTIALSOUTH, TOTAL) VALUES ('2014-
07-01','Aedes','flavescens','Banded legs','Female',6,5,8,0,0,0,5,0,0,3,1,28);
INSERT INTO "MOSQUITO_TRAP_DATA" (TRAP_DATE, GENUS, SPECIES, TYPE, GENDER, RURALNORTHWEST,
RURALNORTHEAST, RURALSOUTHEAST, RIVERVALLEYEAST, RIVERVALLEYWEST, RESIDENTIALNORTH,
RURALSOUTHWEST, LAGOON, GOLFCOURSE, INDUSTRIALPARK, RESIDENTIALSOUTH, TOTAL) VALUES ('2014-
07-01','Aedes','vexans','Banded legs','Female',3,168,1,21,38,8,16,0,0,3,32,290);

```

# Exercise 3: SELECT

## Data Manipulation Language

- Type or copy/paste the SELECT statement into the empty SQLite Execute SQL window
  - `SELECT COUNT(*) FROM MOSQUITO_TRAP_DATA;`
- Click the **Execute SQL** ► button on the toolbar
- When you execute the query, you are asking the database a question
  - Can you tell me the number of rows in the MOSQUITO\_TRAP\_DATA table?
- The database gives you an answer (the result) and you should have received the following message:
  - 7 rows returned in 1ms from: `SELECT * FROM MOSQUITO_TRAP_DATA;`

# Exercise 3: SELECT

## Data Manipulation Language

- What if you want to see all the rows in your database?
  - `SELECT * FROM MOSQUITO_TRAP_DATA;`
  - Returns all columns and rows in a table
- What if you only want to see the Genus, Species and Total of each row?
  - `SELECT GENUS, SPECIES, TOTAL FROM MOSQUITO_TRAP_DATA;`
  - Returns only the GENUS, SPECIES, TOTAL columns for each row in a table



# Data Manipulation Language

- The WHERE clause
  - Uses operators to extract only those records that fulfill a specified condition
- Used to ask more complicated questions
- SQL will do exactly what you ask, not always what you expect
- “I do not think it means what you think it means”
  - *Inigo Montoya*

Operator	Description
=	Equal
<>	Not equal. <b>Note:</b> In some versions of SQL this operator may be written as !=
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between a certain range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

# Exercise 3: SELECT

## Data Manipulation Language

- Show the rows that have a mosquito TYPE of “Black legs”
  - `SELECT * FROM MOSQUITO_TRAP_DATA WHERE TYPE = 'Black legs';`

### ***YOUR TURN***

- Write and execute a DML statement to answer the question below:
  - Which mosquito species' were caught in the traps placed in the west river valley?

# Exercise 3: UPDATE

## Data Manipulation Language

- Select the Execute SQL tab in SQLite
- Type or copy/paste the UPDATE statement into an empty SQLite Execute SQL window
- Click the **Execute SQL ►** button on the toolbar
- You should receive the following message:
  - Query executed successfully: ... (took 1ms, 4 rows affected)

```
UPDATE MOSQUITO_TRAP_DATA  
SET GENDER = 'Male'  
WHERE SAMPLEID IN (1,3,5,7);
```

# Data Manipulation Language

- The GROUP BY clause
  - Used in collaboration with the SELECT statement to arrange identical data into groups
- The GROUP BY statement is often used with aggregate functions

Function	Description
AVG	Calculates the average of a set of values
COUNT	Counts rows in a specified table or view
MAX	Gets the minimum value in a set of values
MIN	Gets the maximum value in a set of values
SUM	Calculates the sum of values

# Exercise 3: SELECT

## Data Manipulation Language

### ***YOUR TURN***

- Write and execute a DML statement to answer the question below:
  - How many mosquitos of each gender were caught in traps throughout the city?

# Exercise 3: DELETE

## Data Manipulation Language

- Select the Execute SQL tab in SQLite
- Type or copy/paste the DELETE statement into an empty SQLite Execute SQL window
- Click the **Execute SQL ►** button on the toolbar
- You should receive the following message:
  - Query executed successfully: ... (took 0ms, 4 rows affected)

```
DELETE FROM  
MOSQUITO_TRAP_DATA WHERE  
GENDER = "Male";
```



# Exercise 3: SELECT

## Data Manipulation Language

### ***YOUR TURN***

- Write and execute a DML statement to answer the question below:
  - At which traps were more mosquitos caught? Rural north east or rural north west?
- Done!

# Advanced SQL

- The MOSQUITO database only has one table
- Databases with more than one table require tables to be joined
- Foreign keys create relationships between tables and must be joined in a DML statement

- Download the LED Streetlight Conversion database called odd\_streetlight.db
- Execute the query below

```
SELECT LED_STREETLIGHT.STREETLIGHT_ID, LED_STREETLIGHT.TYPE,  
LOCATION.LOCATION  
FROM LED_STREETLIGHT, LOCATION  
WHERE LED_STREETLIGHT.STREETLIGHT_ID = LOCATION.STREETLIGHT_ID  
AND LED_STREETLIGHT.STREETLIGHT_ID = 12;
```

# City of Edmonton Open Data Portal

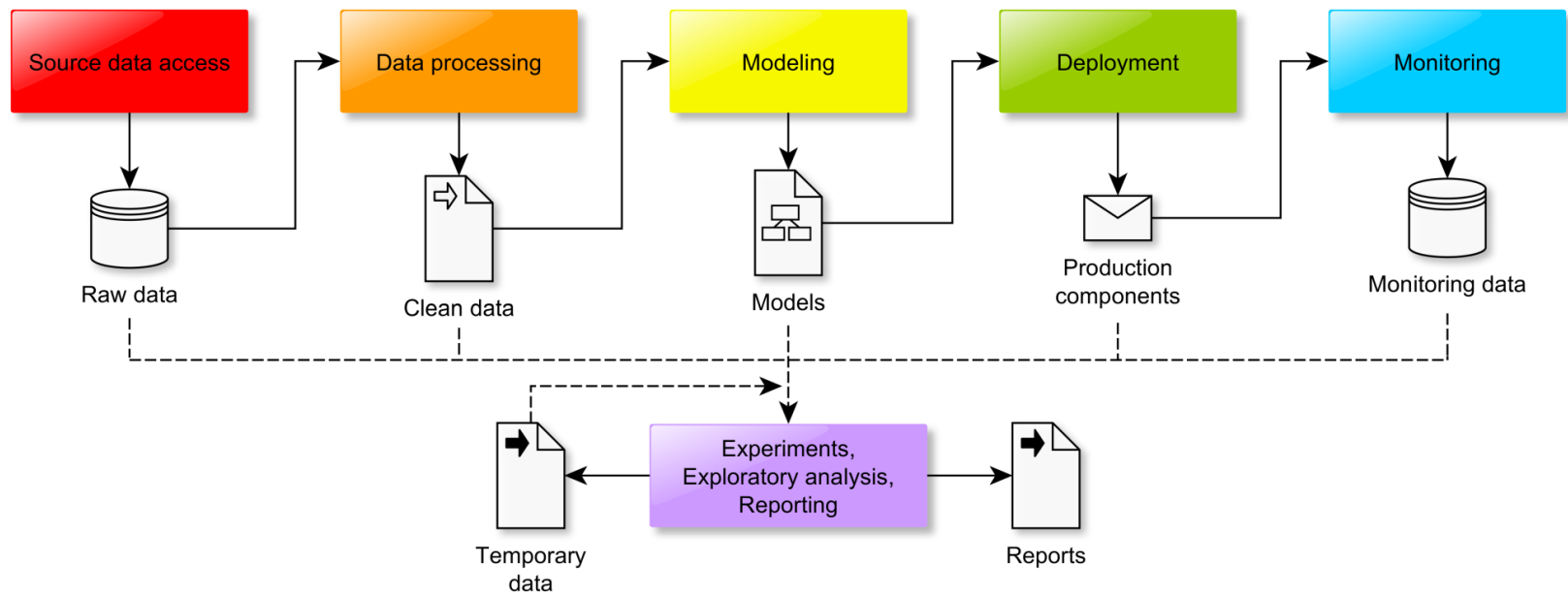
Workshop

Introducing SQL: Foundation of Data Analytics

# Using the Open Data Portal

- <https://data.edmonton.ca/>
- Data sets are usually available in comma separated value (CSV) format
- To use the dataset requires cleaning, importing, exploring and understand the data set
  - Workshop: Exploring & Cleaning Data with OpenRefine
- Requires work

# Data Work Flow



# How I prepared the data sets for today

- Selected data sets from the Open Data Portal
- Downloaded the CSV and surveyed in Google Sheets
- Cleaned the data set
  - E.g. reformatted dates from MMM DD YYYY to YYYY-MM-DD
- Imported into directly into SQLite tables
- Added primary keys
- Explored data set using DML

# Some “Mosquitoes Trap Data” questions

- How many mosquitos caught in 2014?

```
SELECT strftime('%Y', TRAP_DATE) as YEAR, SUM(TOTAL)  
FROM MOSQUITO_TRAP_DATA  
WHERE TOTAL <> ''  
AND TOTAL > 0  
GROUP BY YEAR;
```

- How many mosquitos of each species were caught?
- Which traps caught the most mosquitos?



# Some “LED Streetlight Conversion” questions

- How many total streetlights?
- How many streetlights are converted to LED?
- How many streetlights were converted by year?

```
SELECT strftime('%Y', STARTDATE) as YEAR, TYPE,  
COUNT(STREETLIGHT_ID)  
FROM LED_STREETLIGHT  
WHERE TYPE = "LED"  
GROUP BY YEAR;
```

# SQL and Climate Change

- Connecting and linking various data sets
- Builds an understanding of what that data means
- Data is a universal language,  
climate change is a global  
problem

# Next steps

- Playing with data and SQL forces you to think and understand the data (builds knowledge)
  - The relationships between data
  - The meaning of those relationships
  - The validity of the data
- SQL is iterative, often a “trial and error” process
  - Don’t be afraid to make mistakes
  - Team sport – discuss, share, question, collaborate
- Data is everywhere which raises questions of privacy, security and ethics

# Experiment



<https://www.manchester.ac.uk/discover/news/major-leap-towards-storing-data-at-the-molecular-level/>

# If there's time ... (I talked too fast)

- Let's (democratically):
  1. Choose a dataset not discussed during the workshops
  2. Formulate a question related to the dataset
  3. Load the data into SQLite
  4. Execute some DML to answer the question

# Thank you!

- Robb Sombach
  - [sombach@ualberta.ca](mailto:sombach@ualberta.ca)
  - [robb@sombach.com](mailto:robb@sombach.com)
  - [LinkedIn](#)



# References

- <https://opendataday.org/>
- <https://data36.com/sql-for-data-analysis-tutorial-beginners/>
- <https://www.datascience.com/blog/to-sql-or-not-to-sql-that-is-the-question>
- <https://codebeautify.org/sqlformatter>