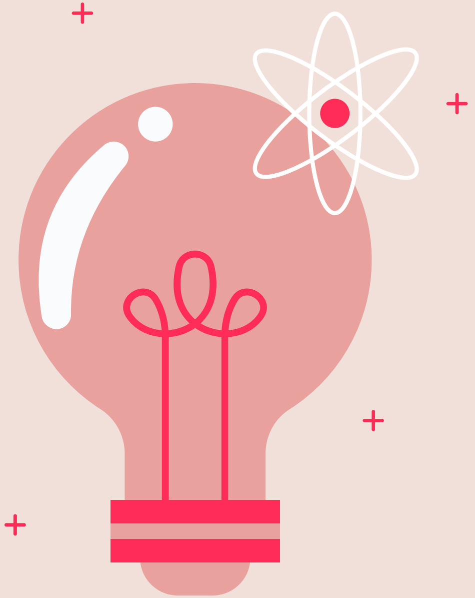
Intro to SQL

Take Notes
Focus
Debug your code

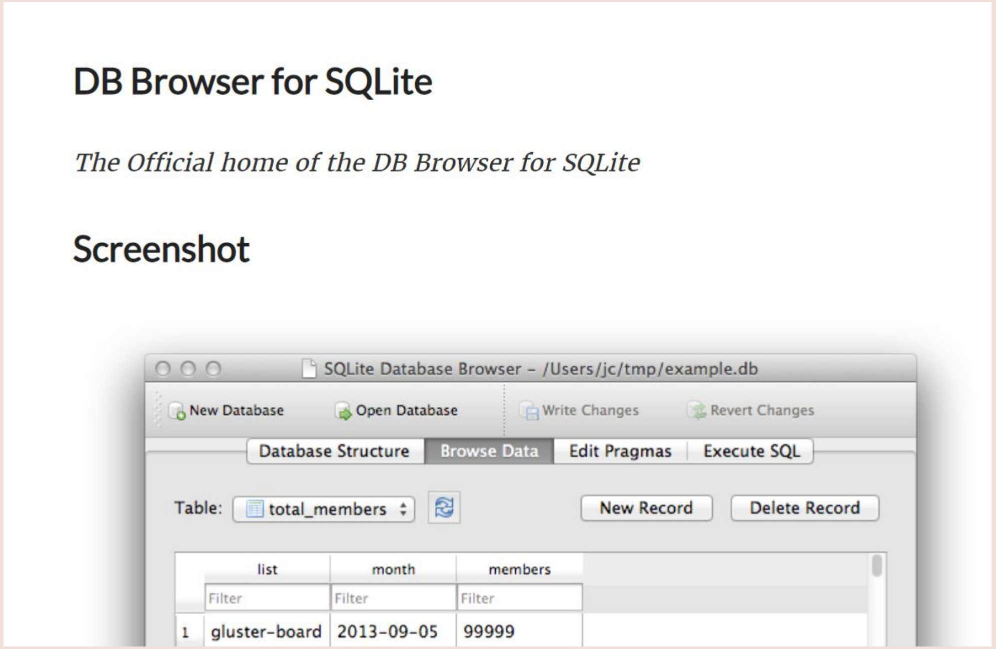


01

Database Queries

Download SQL Lite

https://sqlitebrowser.org/



DB Browser for SQLite

The Official home of the DB Browser for SQLite

Screenshot

	list	month	members
	Filter	Filter	Filter
1	gluster-board	2013-09-05	99999

Create Table Format

```
create table  TABLENAME
(
  COLUMNNAME integer   NOT NULL PRIMARY KEY,
   COLUMNNAME  DATATYPE
 );
```

Data Schema Sample

```
create table product (
 id integer   NOT NULL PRIMARY KEY,
 name  varchar(255),
 description varchar(255),
 cost     double,
 typeid  integer
 );

create table type (
 id integer   NOT NULL PRIMARY KEY,
   name  varchar(255));
```

```
create table user (
 id integer   NOT NULL PRIMARY KEY,
  username  varchar(255),
  lastname  varchar(255),
firstname  varchar(255),
email  varchar(255));
```

Data Schema Sample 2

```sql
CREATE DATABASE spca;


DROP TABLE IF EXISTS dogs;
CREATE TABLE dogs (
    dog_id integer PRIMARY KEY,
    breed varchar(50),
    type varchar(30),
    max_height integer,
    max_weight integer,
    max_life_span integer,
    general_health varchar(30),
    intelligence varchar(10),
    friendly varchar(10)
);
```

```sql
DROP TABLE IF EXISTS intelligence;
CREATE TABLE intelligence (
    breed varchar(50),
    classification varchar(50),
    obey numeric(5,2),
    reps_lower Integer,
    reps_upper Integer
);

DROP TABLE IF EXISTS popularity;
CREATE TABLE popularity (
    breed varchar(50),
    2016_rank Integer,
    2015_rank Integer,
    2014_rank Integer,
    2013_rank Integer
);
```

```sql
INSERT INTO dogs (dog_id, breed, type, max_height, max_weight, max_life_span, general_health,
intelligence, friendly)
VALUES (1, 'Chinese Shar-Pei', 'Working Dogs', 20, 55, 12, 'Poor', 'Low', 'Rarely');
```

Lab: Create Database

- Create Product Table
  - Create primary key, id
  - Create field, name
  - Create field, description
  - Create field , cost
  - Create field , typeid
- Create Table Type
  - Create primary key, id
  - Create field , name

- Reminder: Database field types:
  - integer
  - double
  - varchar

Main SQL Commands

DML: Data Manipulation Language

- **INSERT**
- **SELECT**
- **UPDATE**
- **DELETE**

Command	Description
SELECT	Retrieves certain records from one or more tables
INSERT	Creates records
UPDATE	Modifies records
DELETE	Deletes records

Lab: Insert Data

Structure:

INSERT INTO TABLENAME (col1, col2) values (value1, value2, ...)

Example:

INSERT INTO DOGS (id, name) values (1, 'spot');

Lab:

- Create 2 Insert statements to insert data into your product table
- Bonus: Create 2 Insert Statements to insert data into your type table

Sample Select Queries

Organize Alphabetically

```
SELECT dog_id, breed, type
FROM dogs ORDER BY
type;
```

Does Not Equal

```
SELECT breed, max_weight
FROM dogs
WHERE max_weight != 175;
```

Pattern Match

```
SELECT dog_id, breed, type
FROM dogs
WHERE breed LIKE '%German% ';
```

Compound Query - And

```
SELECT dog_id, breed, max_weight
FROM dogs
WHERE
max_weight > 175
AND
max_weight < 200
```

Compound Query - Or

```
SELECT dog_id, breed, max_weight
FROM dogs
WHERE
max_weight > 175
OR
breed != 'poodle';
```

Lab: Select Queries

- Create a query to:
  select all columns from product table where where cost is not null
- Create a query to:
  select all products where cost does not equal 10
- Create a query to:
  select the name and cost columns from the product table where cost is greater than 10
- Create a query to select the name column from the product where name has an a in it
- Bonus: Create a query to select the count of all names in the table

Join Types (Inner)

Old School Inner

SELECT product.name, type.name

FROM

product, type

WHERE

product.typeid=type.id

Standard Inner

SELECT product.name, type.name

FROM product

JOIN type

ON

product.typeid = type. id

Standard Inner

SELECT product.name, type.name

FROM product

INNER JOIN type

ON

product.typeid = type. id

Join Types (Left, RIght)

Right

(returns everything from 2nd table, & what matches in the first)

```
SELECT
product.name, type.name
FROM product
RIGHT JOIN type
ON
product.typeid = type. id
```

Left

(returns everything from first table and what matches in the first)

```
SELECT
product.name, type.name
FROM product
LEFT JOIN type
ON
product.typeid = type. id
```

Chained Joins

Left

(returns everything from first table and what matches in the first)

```
SELECT
product.name, type.name,
orders.amount
FROM product
LEFT JOIN type
ON
product.typeid = type. id
LEFT JOIN orders
ON
orders.productid = product.id
```

Chained Joins with Filter

Left

(returns everything from first table and what matches in the first)

```
SELECT
product.name, type.name,
orders.amount
FROM product
LEFT JOIN type
ON
product.typeid = type. id
LEFT JOIN orders
ON
orders.productid = product.id
WHERE
product.name like '%america%'
```

Lab: Create SQL Join

Create old school join on product & type table

```
SELECT table1.fieldname, table2.fieldname
FROM
table1, table2
WHERE
table1.foreignkey = table2.primarykey
```

Create standard join

```
SELECT table1.fieldname, table2.fieldname
FROM table1
JOIN table2
ON
table1.foreignkey = table2.primarykey
```