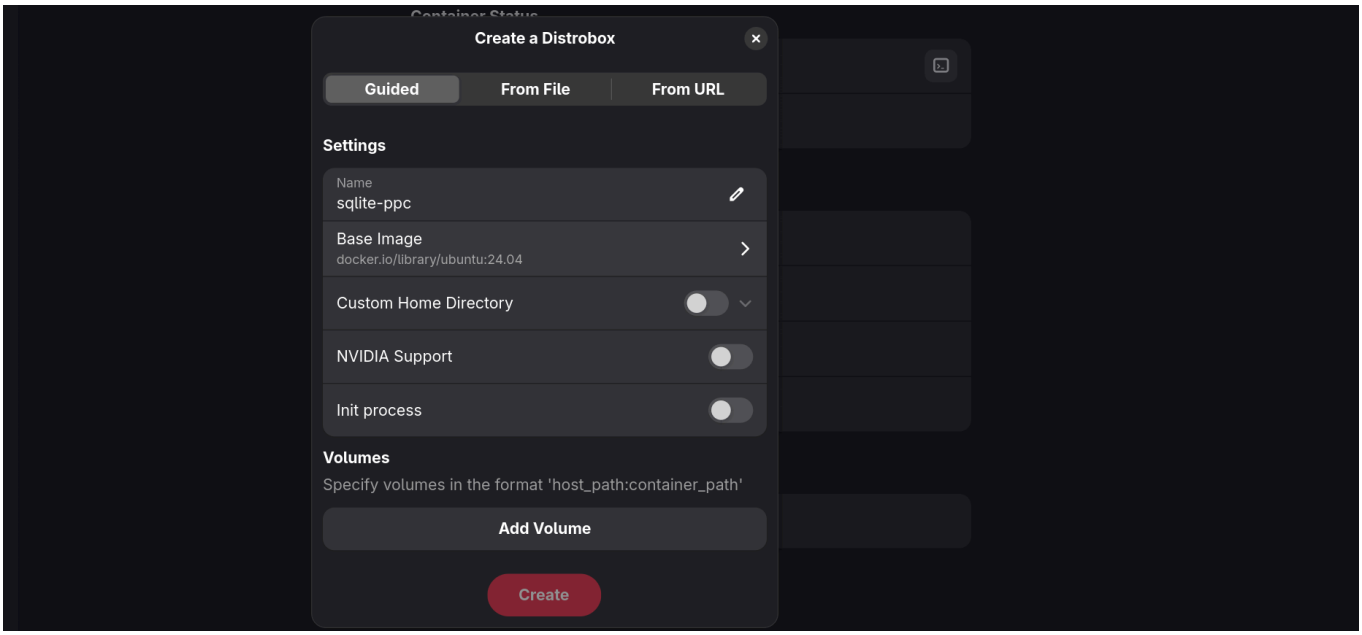
This assignment serves as an introductory exercise to reading csv data into a database.

Setup

I'm completing this assignment on the Project Bluefin O.S. I don't anticipate need SQLite installed on my host system, so i'm going to set up a basic Ubuntu Distrobox for this assignment.



I've gone with Ubuntu 24.04, because it's stable and I already have the image downloaded.

```
❯ distrobox enter sqlite-ppc
Starting container...                                  [ OK ]
Installing basic packages...                           [ OK ]
Setting up devpts mounts...                            [ OK ]
Setting up read-only mounts...                         [ OK ]
Setting up read-write mounts...                        [ OK ]
Setting up host's sockets integration...               [ OK ]
Integrating host's themes, icons, fonts...             [ OK ]
Setting up distrobox profile...                        [ OK ]
Setting up sudo...                                     [ OK ]
Setting up user groups...                              [ OK ]
Setting up user's group list...                        [ OK ]
Setting up existing user...                            [ OK ]
Ensuring user's access...                              [ OK ]

Container Setup Complete!
📦[b@sqlite-ppc ~]$
```

Here I've entered my container, and we can see that setup is successful. Let's enter it and get sqlite3 installed.

```
📦[b@sqlite-ppc ~]$ sudo apt install sqlite3
Reading package lists... 0%
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  sqlite3-doc
The following NEW packages will be installed:
  sqlite3
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 144 kB of archives.
After this operation, 584 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 sqlite3 amd64 3.45.1-1ubuntu2.5 [
144 kB]
Fetched 144 kB in 1s (171 kB/s)
Selecting previously unselected package sqlite3.
(Reading database ... 26618 files and directories currently installed.)
Preparing to unpack .../sqlite3_3.45.1-1ubuntu2.5_amd64.deb ...
Unpacking sqlite3 (3.45.1-1ubuntu2.5) ...
Setting up sqlite3 (3.45.1-1ubuntu2.5) ...
Processing triggers for man-db (2.12.0-4build2) ...
📦[b@sqlite-ppc ~]$
```

Alright, we've completed the installation and setup, and we can get started on the actual assignment.

Assignment

First, we need to copy our csv into the container.

On the host machine, i'm using podman's container manager tools to copy the csv file (nyt1.csv) into my container:

```
~/Downloads
❯ podman cp nyt1.csv sqlite-ppc:/var/home/b
```

Now, let's check that it successfully copied:

```
📦[b@sqlite-ppc ~]$ ls | grep nyt
nyt1.csv
```

Great! Our csv is on the container, and we can start reading the csv data into our sqlite database.

Let's examine our file. I'll use `head` to display the first 5 lines of the csv.

```
📦[b@sqlite-ppc ~]$ head -n 5 nyt1.csv
"Age","Gender","Impressions","Clicks","Signed_In"
36,0,3,0,1
73,1,3,0,1
30,0,3,0,1
49,1,3,0,1
📦[b@sqlite-ppc ~]$
```

We can see our csv contains 5 columns, all holding integer values: Age, Gender, impressions, clicks, and Signed_In.

Next, we need to start up the sqlite CLI.

```
📦[b@sqlite-ppc ~]$ sqlite3
SQLite version 3.45.1 2024-01-30 16:01:20
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite>
```

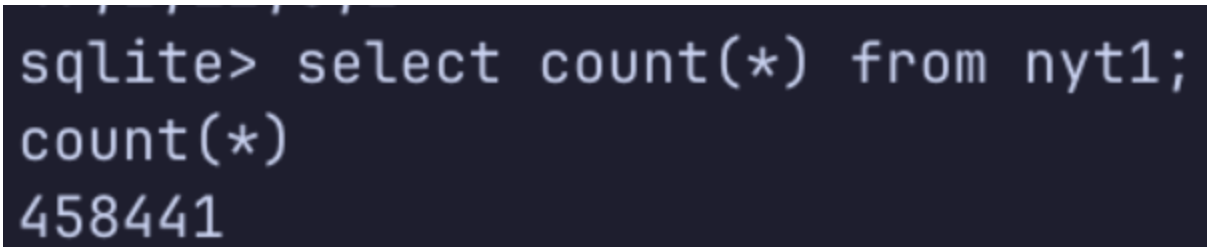
Let's start by creating a table following the structure of the csv. `.mode csv` tells SQLite to expect CSV data. `.header on` warns of a header that should be skipped. Then, we import the data from nyt1.csv into the nyt1 table. SQLite reads the CSV file, adding each row to the table. Finally, we'll display the first 5 rows of our table.

```
sqlite> CREATE TABLE nyt1(Age int, Gender int, impressions int, clicks int, Signed_In int);
sqlite> .mode csv
sqlite> .header on
sqlite> .import nyt1.csv nyt1
sqlite> select * from nyt1 LIMIT 5;
Age,Gender,impressions,clicks,Signed_In
Age,Gender,Impressions,Clicks,Signed_In
36,0,3,0,1
73,1,3,0,1
30,0,3,0,1
49,1,3,0,1
```

It looks like even though I warned SQLite of the header, it got imported anyway. Let's correct that.

```
sqlite> delete from nyt1 where Age='Age';
sqlite> select * from nyt1 LIMIT 5;
Age,Gender,impressions,clicks,Signed_In
36,0,3,0,1
73,1,3,0,1
30,0,3,0,1
49,1,3,0,1
47,1,11,0,1
```

Awesome! Now we've deleted the junk row.
Let's see how many rows we have in this table.



Great, we have 458,441 rows in our nyt1 table. Let's try out `AND`. How many rows have both signed in and had 3 impressions?

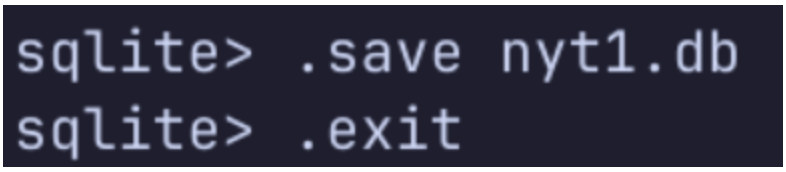
```
sqlite> SELECT COUNT(*) FROM nyt1 WHERE Signed_In in (SELECT 0) AND impressions in (SELECT 3);
COUNT(*)
19347
```

Let's try an `OR` statement. How about all the rows with non zero impressions or clicks?

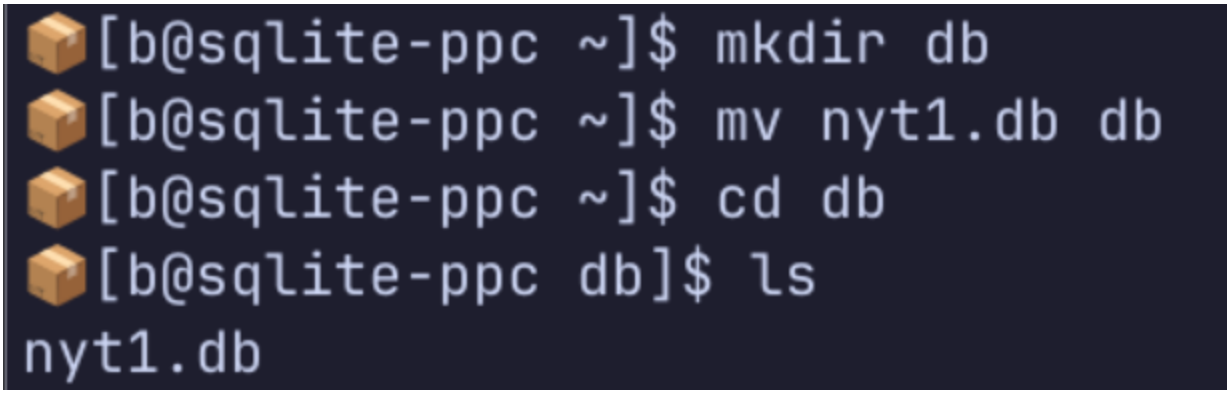
```
sqlite> SELECT COUNT(*) FROM nyt1 WHERE CLICKS NOT IN (SELECT 0) OR IMPRESSIONS NOT IN (SELECT 0);
COUNT(*)
455375
```

Nice! We've successfully setup a SQLite database, read in a CSV file, and tested out a few basic commands.

Let's save our work to a database file, and exit.



Finally, I'll make a new folder for database files in my container and move the new file there.



Conclusion

This was a nice exposure to SQLite, and I look forward to learning more with SQLite in this class.