To load the database, connect to your mysql server and open the script sf_library. Once the script is open, you might also need to add a path to the csv file in the line

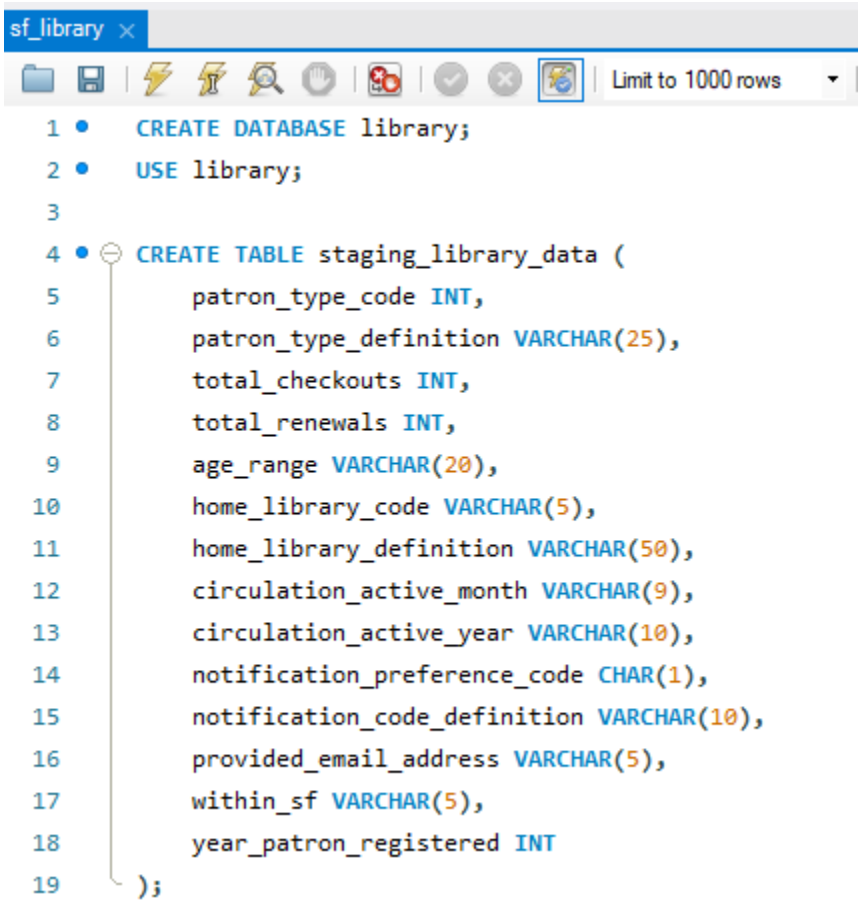
LOAD DATA LOCAL INFILE "SFPL_DataSF_library-usage_Jan_2023.csv"

In order to connect the python program to it, you will also need to replace these lines here with your own relevant information. I should mention as well I ran this code with IDLE version 3.12

```
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="password",
    database="library"
)
```

I'm not exactly sure what the proper way to document this project is, but I'll break down the mysql syntax here.

To start, I created a database and a temporary table to stage the data from the csv into while we load it into organized separate tables



I should note that in the staging table, within_sf and provided_email_address are both VARCHAR instead of BOOLEAN. I had issues trying to load them directly from the csv as a bool.

Next I loaded the data from the csv into my staging table.

```
20
21 •  SET GLOBAL local_infile=ON;
22
23 •  LOAD DATA LOCAL INFILE "SFPL_DataSF_library-usage_Jan_2023.csv"
24    INTO TABLE staging_library_data
25    FIELDS TERMINATED BY ','
26    LINES TERMINATED BY '\n'
27    IGNORE 12 ROWS
28    (@patron_type_code,
29    @patron_type_definition,
30    @total_checkouts,
31    @total_renewals,
32    @age_range,
33    @home_library_code,
34    @home_library_definition,
35    @circulation_active_month,
36    @circulation_active_year,
37    @notification_preference_code,
38    @notification_code_definition,
39    @provided_email_address,
40    @within_sf,
41    @year_patron_registered)
42    SET
43     patron_type_code = NULLIF(TRIM(@patron_type_code), ''),
44     patron_type_definition = NULLIF(TRIM(@patron_type_definition), ''),
45     total_checkouts = NULLIF(TRIM(@total_checkouts), ''),
46     total_renewals = NULLIF(TRIM(@total_renewals), ''),
47     age_range = NULLIF(TRIM(@age_range), ''),
48     home_library_code = NULLIF(TRIM(@home_library_code), ''),
49     home_library_definition = NULLIF(TRIM(@home_library_definition), ''),
50     circulation_active_month = NULLIF(TRIM(@circulation_active_month), ''),
51     circulation_active_year = NULLIF(TRIM(@circulation_active_year), ''),
52     notification_preference_code = NULLIF(TRIM(@notification_preference_code), ''),
53     notification_code_definition = NULLIF(TRIM(@notification_code_definition), ''),
54     provided_email_address = NULLIF(TRIM(@provided_email_address), ''),
55     within_sf = NULLIF(TRIM(@within_sf), ''),
56     year_patron_registered = NULLIF(TRIM(@year_patron_registered), '');
57
58
```

I should mention that I did have chatgpt show me how to deal with unexpected values and empty strings and the solution it showed me was using the @ symbol to load it into a placeholder and then use NULLIF(TRIM()) to set empty values to null.

After the csv is loaded into the staging table, we can start making separate organized tables. First we have library_info. This creates an auto incrementing primary key for each library's id and loads the home_library_code, home_library_definition, and within_sf. It also converts the VARCHAR into a BOOLEAN. I'm sure there was a way to load it in directly as a bool but I kept running into issues trying.

```
59  CREATE TABLE library_info (
60      library_id INT AUTO_INCREMENT PRIMARY KEY,
61      home_library_code VARCHAR(5) UNIQUE,
62      home_library_definition VARCHAR(50),
63      within_sf BOOLEAN
64  );
65
66  INSERT INTO library_info(
67  home_library_code,
68  home_library_definition,
69  within_sf)
70  SELECT
71  home_library_code,
72  MIN(TRIM(home_library_definition)) AS home_library_definition,
73  IF(MIN(TRIM(within_sf))='True', TRUE, FALSE) AS within_sf
74  FROM staging_library_data
75  GROUP BY home_library_code;
76
```

Our next table is for the notification codes and definitions.

```sql
CREATE TABLE notification_preferences (
notification_code CHAR(1) PRIMARY KEY,
notification_definition VARCHAR(50)
);

INSERT INTO notification_preferences(
notification_code,
notification_definition
)
SELECT DISTINCT
notification_preference_code,
notification_code_definition
FROM staging_library_data
WHERE notification_preference_code IS NOT NULL;
```

Now we have the patron_types table, this one is pretty small, it just links each patron code to it's definition

```sql

CREATE TABLE patron_types (
patron_type_code INT PRIMARY KEY,
patron_type_definition VARCHAR(30)
);

INSERT INTO patron_types (
patron_type_code,
patron_type_definition
)
SELECT DISTINCT
patron_type_code,
patron_type_definition
FROM staging_library_data
WHERE patron_type_code IS NOT NULL;


```

The next table we have is patrons, this keeps track of each individual patron and any relevant information associated with them specifically

```
107
108 CREATE TABLE patrons (
109 patron_id INT AUTO_INCREMENT PRIMARY KEY,
110 patron_type_code INT,
111 total_checkouts INT,
112 total_renewals INT,
113 age_range VARCHAR(20),
114 home_library_code VARCHAR(5),
115 notification_preference_code CHAR(1),
116 provided_email_address BOOLEAN,
117 circulation_active_month VARCHAR(9),
118 circulation_active_year INT,
119 year_patron_registered INT,
120 FOREIGN KEY (patron_type_code) REFERENCES patron_types(patron_type_code),
121 FOREIGN KEY (home_library_code) REFERENCES library_info(home_library_code),
122 FOREIGN KEY (notification_preference_code) REFERENCES notification_preferences(notification_code)
123 );
124
```

```sql
INSERT INTO patrons(
patron_type_code,
total_checkouts,
total_renewals,
age_range,
home_library_code,
notification_preference_code,
provided_email_address,
circulation_active_month,
circulation_active_year,
year_patron_registered
)
SELECT
patron_type_code,
total_checkouts,
total_renewals,
age_range,
CASE
    WHEN home_library_code IN (SELECT home_library_code FROM library_info)
    THEN home_library_code
    ELSE NULL
END AS home_library_code,
CASE
    WHEN notification_preference_code IN (SELECT notification_code FROM notification_preferences)
    THEN notification_preference_code
    ELSE NULL
END AS notification_preference_code,
CASE
    WHEN LOWER(TRIM(provided_email_address)) IN ('true', 'yes', '1') THEN TRUE
    WHEN LOWER(TRIM(provided_email_address)) IN ('false', 'no', '0') THEN FALSE
    ELSE NULL
END AS provided_email_address,
circulation_active_month,
circulation_active_year,
year_patron_registered
FROM staging_library_data;
```

I should note here that I did have chatgpt show me how to use the CASE WHEN THEN ELSE NULL statements here to prevent errors when loading the data from the other tables.

Next up we have the demographic_summary table. This just keeps track of the total individual library usage by age

```sql
CREATE TABLE demographic_summary (
demographic_id INT AUTO_INCREMENT PRIMARY KEY,
library_id INT,
age_range VARCHAR(20),
total_checkouts INT,
total_renewals INT,
FOREIGN KEY (library_id) REFERENCES library_info(library_id)
);

INSERT INTO demographic_summary(
library_id, age_range,
total_checkouts,
total_renewals
)
SELECT
l.library_id,
p.age_range,
SUM(p.total_checkouts),
SUM(p.total_renewals)
FROM patrons p
JOIN library_info l ON p.home_library_code = l.home_library_code
GROUP BY l.library_id, p.age_range;

```

The library_mail_digital table just pulls any digital or by mail library card types to keep track of separately.

```
184
185 CREATE TABLE library_mail_digital (
186 patron_id INT PRIMARY KEY,
187 patron_type_code INT,
188 age_range VARCHAR(20),
189 total_checkouts INT,
190 total_renewals INT,
191 FOREIGN KEY (patron_id) REFERENCES patrons(patron_id),
192 FOREIGN KEY (patron_type_code) REFERENCES patron_types(patron_type_code)
193 );
194
195 INSERT INTO library_mail_digital(
196 patron_id,
197 patron_type_code,
198 age_range,
199 total_checkouts,
200 total_renewals
201 )
202 SELECT
203 patron_id,
204 patron_type_code,
205 age_range,
206 total_checkouts,
207 total_renewals
208 FROM patrons
209 WHERE patron_type_code IN (12, 16);
```

The staff_teachers table does pretty much the same thing, it just keeps track of the retired/active staff and any teacher cards

```
210
211 • CREATE TABLE staff_teachers (
212   patron_id INT PRIMARY KEY,
213   patron_type_code INT,
214   age_range VARCHAR(20),
215   home_library_code VARCHAR(5),
216   total_checkouts INT,
217   total_renewals INT,
218   FOREIGN KEY (patron_id) REFERENCES patrons(patron_id),
219   FOREIGN KEY (patron_type_code) REFERENCES patron_types(patron_type_code),
220   FOREIGN KEY (home_library_code) REFERENCES library_info(home_library_code)
221   );
222
223 • INSERT INTO staff_teachers(
224   patron_id,
225   patron_type_code,
226   age_range,
227   home_library_code,
228   total_checkouts,
229   total_renewals
230   )
231   SELECT
232   patron_id,
233   patron_type_code,
234   age_range,
235   home_library_code,
236   total_checkouts,
237   total_renewals
238   FROM patrons
239   WHERE patron_type_code IN (5, 15, 55);
```