Object Oriented Programming (Using Python)

UNIT- V

Using Databases with Python:

- Using Databases
- Single Table CRUD (Create, Read, Update, and Delete)
- Designing and representing a data model
- Reconstructing data with JOIN
- Many-to-many relationships

Prof. R. MADANA MOHANA
Professor, Artificial Intelligence & Data Science
http://rmadanamohana.com/

References

https://www.coursera.org/learn/python-databases/home/info

https://www.py4e.com/book

http://do1.dr-chuck.com/pythonlearn/EN_us/pythonlearn.pdf

https://www.py4e.com/code3/

https://www.py4e.com/lectures3/

https://www.youtube.com/playlist?list=PLlRFEj9H3Oj7Bp8-DfGpfAfDBiblRfl5p

http://sqlitebrowser.org/

Reconstructing data with JOIN

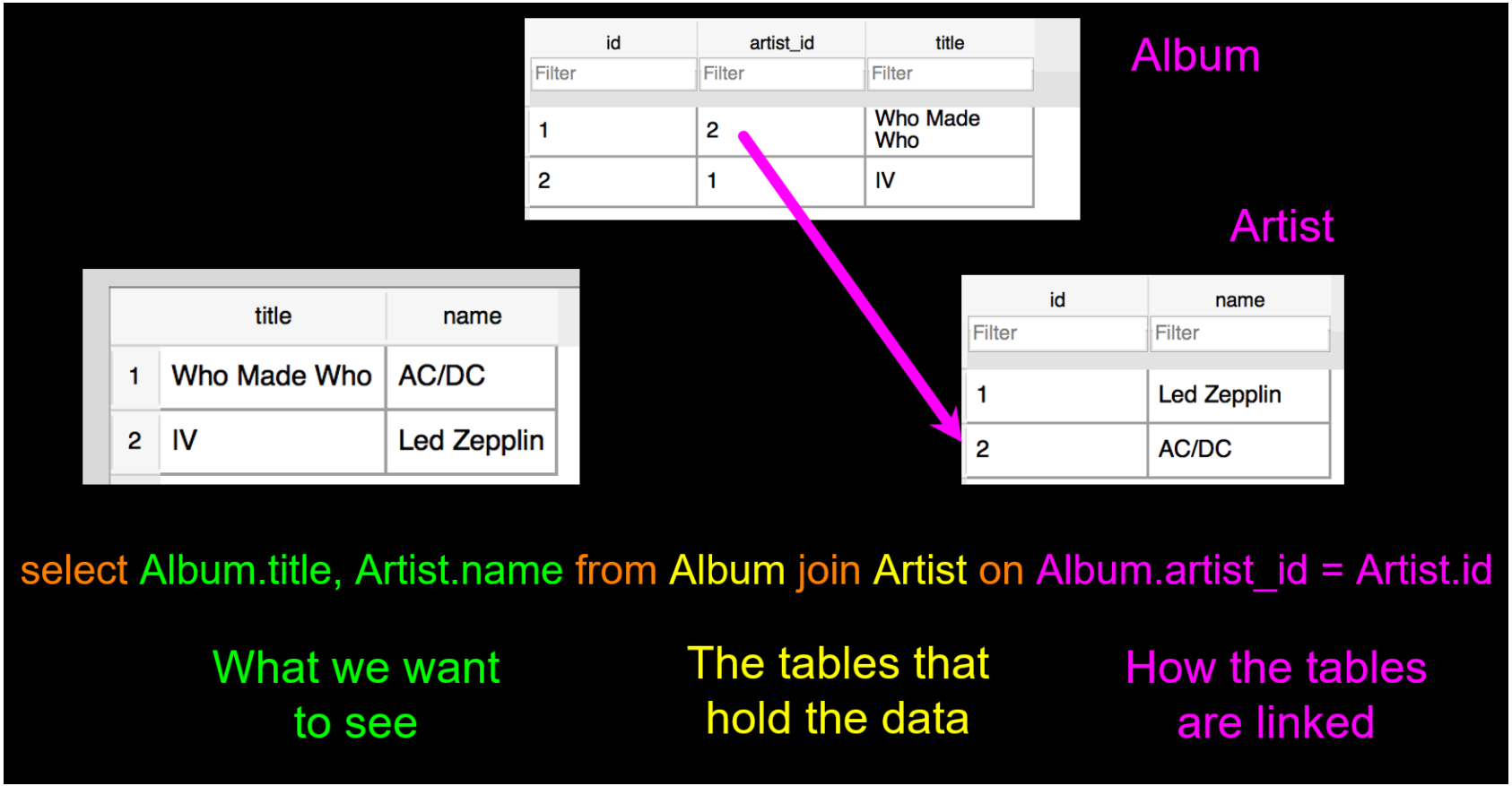
Relational Power

- By removing the replicated data and replacing it with references to a single copy of each bit of data we build a "web" of information that the relational database can read through very quickly - even for very large amounts of data
- Often when we want some data it comes from a number of tables linked by these foreign keys

The JOIN Operation

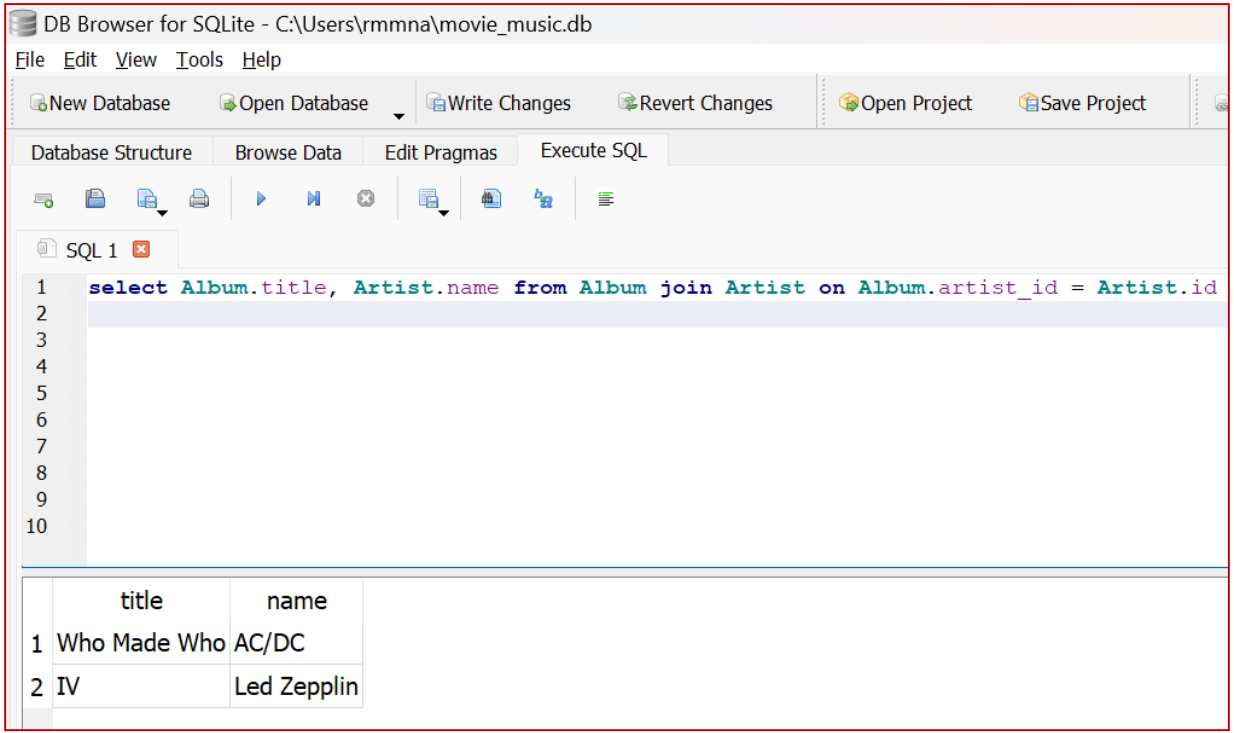
- The JOIN operation links across several tables as part of a SELECT operation
- We must tell the JOIN how to use the keys that make the connection between the tables using an ON clause

The JOIN Operation



The JOIN Operation

```
select Album.title, Artist.name from Album join Artist on
Album.artist_id = Artist.id
```



The JOIN Operation

id	artist_id	title
Filter	Filter	Filter
1	2	Who Made Who
2	1	IV

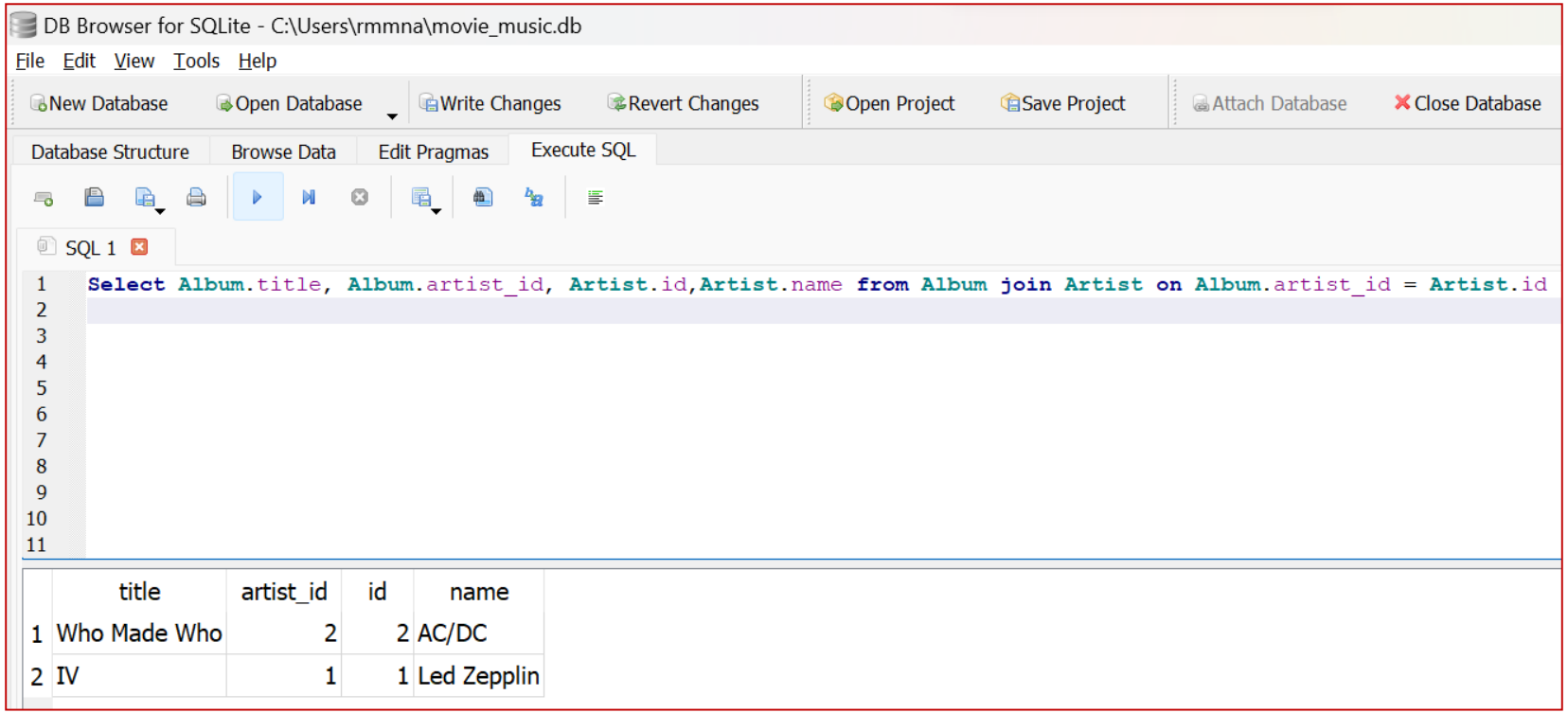
id	name
Filter	Filter
1	Led Zepplin
2	AC/DC

	title	artist_id	id	name
1	Who Made Who	2	2	AC/DC
2	IV	1	1	Led Zepplin

```
select Album.title, Album.artist_id, Artist.id,Artist.name
from Album join Artist on Album.artist_id = Artist.id
```

The JOIN Operation

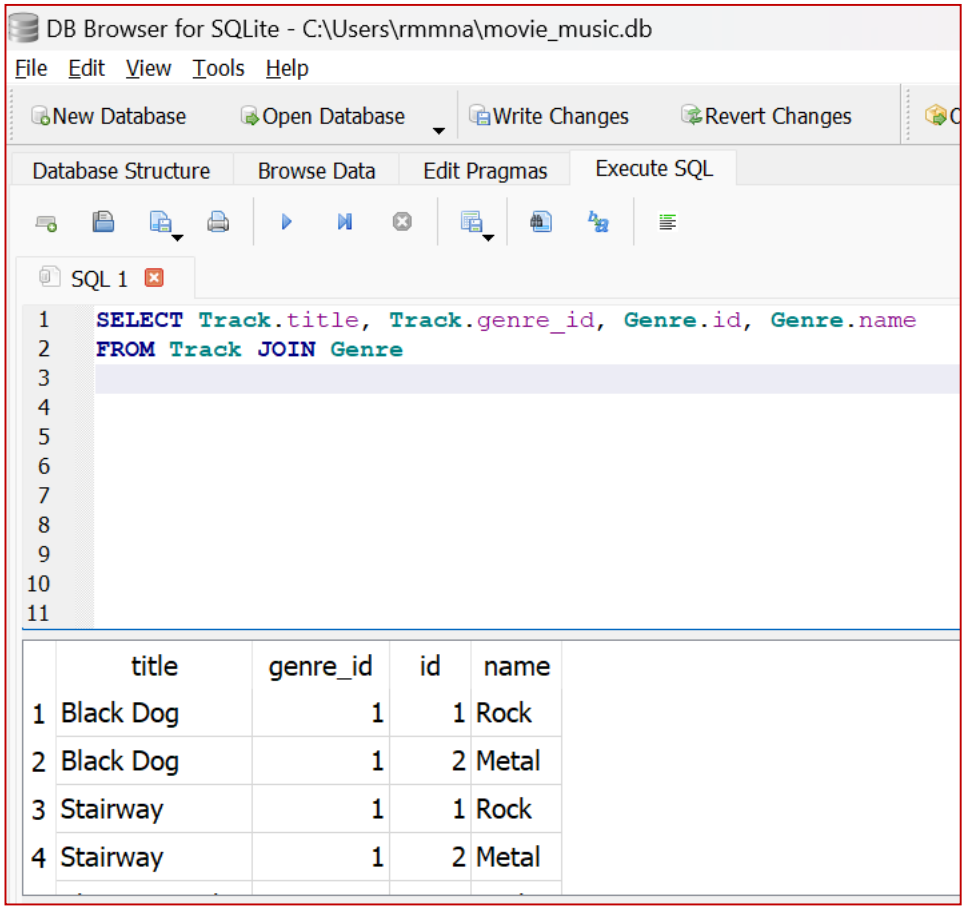
```
Select Album.title, Album.artist_id, Artist.id,Artist.name
from Album join Artist on Album.artist_id = Artist.id
```



The JOIN Operation

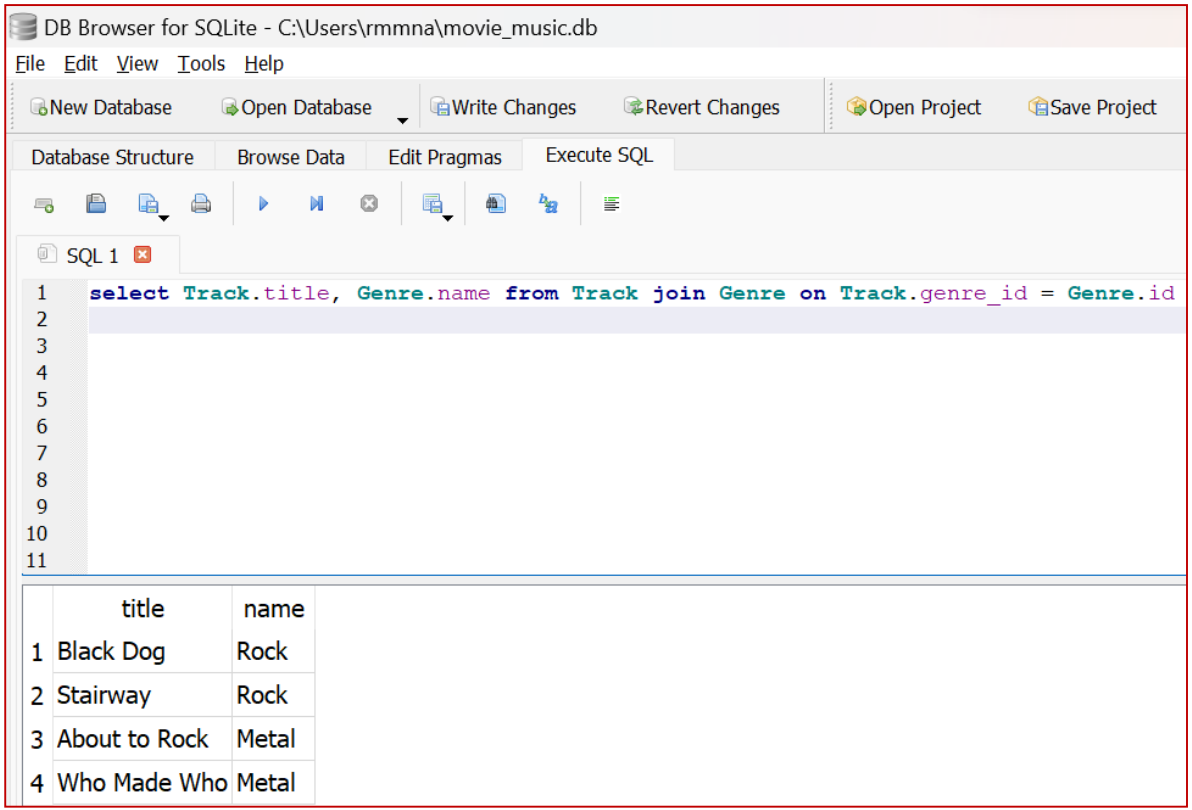
```
SELECT Track.title,
Track.genre_id, Genre.id,
Genre.name
FROM Track JOIN Genre
```

Joining two tables without an ON clause gives all possible combinations of rows.



The JOIN Operation

```
select Track.title, Genre.name from Track join Genre on
Track.genre_id = Genre.id
```



The JOIN Operation

	title	name
1	Black Dog	Rock
2	Stairway	Rock
3	About to Rock	Metal
4	Who Made Who	Metal

id	title	album_id	genre_id	len	rating	count
Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	Black Dog	2	1	297	5	0
2	Stairway	2	1	482	5	0
3	About to Rock	1	2	313	5	0
4	Who Made Who	1	2	207	5	0

id	name
Filter	Filter
1	Rock
2	Metal

```
select Track.title, Genre.name from Track join Genre on Track.genre_id = Genre.id
```

What we want to see

The tables that hold the data

How the tables are linked

The JOIN Operation

```
select Track.title, Artist.name, Album.title, Genre.name
from Track join Genre join Album join Artist on
Track.genre_id = Genre.id and Track.album_id = Album.id and
Album.artist_id = Artist.id
```

	title	name	title	name
1	Black Dog	Led Zepplin	IV	Rock
2	Stairway	Led Zepplin	IV	Rock
3	About to Rock	AC/DC	Who Made Who	Metal
4	Who Made Who	AC/DC	Who Made Who	Metal

The JOIN Operation

```
select Track.title, Artist.name, Album.title,
Genre.name from Track join Genre join Album join
Artist on Track.genre_id = Genre.id and
Track.album_id = Album.id and Album.artist_id =
Artist.id
```

	title	name	title	name
1	Black Dog	Led Zepplin	IV	Rock
2	Stairway	Led Zepplin	IV	Rock
3	About to Rock	AC/DC	Who Made Who	Metal
4	Who Made Who	AC/DC	Who Made Who	Metal

What we want to see

The tables which hold the data

How the tables are linked

The JOIN Operation

Hells Bells	5:13	AC/DC	Who Made Who	Rock	★★★★★	61
Shake Your Foundations	3:54	AC/DC	Who Made Who	Rock	★★★★★	70
Chase the Ace	3:01	AC/DC	Who Made Who	Rock		56
For Those About To Rock (We ...	5:54	AC/DC	Who Made Who	Rock	★★★★★	61
Dúlamán	3:43	Altan	Natural Wonders M...	New Age		31
Rode Across the Desert	4:10	America	Greatest Hits	Easy Listen...	★★★★★	23
Now You Are Gone	3:08	America	Greatest Hits	Easy Listen...	★★★★★	18
Tin Man	3:30	America	Greatest Hits	Easy Listen...	★★★★★	23
Sister Golden Hair	3:22	America	Greatest Hits	Easy Listen...	★★★★★	24
Track 01	4:22	Billy Price	Danger Zone	Blues/R&B	★★★★★	26
Track 02	2:45	Billy Price	Da			
Track 03	3:26	Billy Price	Da			
Track 04	4:17	Billy Price	Da			
Track 05	3:50	Billy Price	Da			
War Pigs/Luke's Wall	7:58	Black Sabbath	Pa			
Paranoid	2:53	Black Sabbath	Pa			
Planet Caravan	4:35	Black Sabbath	Pa			
Iron Man	5:59	Black Sabbath	Pa			
Electric Funeral	4:53	Black Sabbath	Pa			
Hand of Doom	7:10	Black Sabbath	Pa			
Rat Salad	2:30	Black Sabbath	Pa			
Jack the Stripper/Fairies Wear ...	6:14	Black Sabbath	Pa			
Bomb Squad (TECH)	3:28	Brent	Br			
clay techno	4:36	Brent	Br			
Heavy	3:08	Brent	Br			
Hi metal man	4:20	Brent	Brent's Album			1
Mistro	2:58	Brent	Brent's Album			1

	title	name	title	name
1	Black Dog	Led Zepplin	IV	Rock
2	Stairway	Led Zepplin	IV	Rock
3	About to Rock	AC/DC	Who Made Who	Metal
4	Who Made Who	AC/DC	Who Made Who	Metal

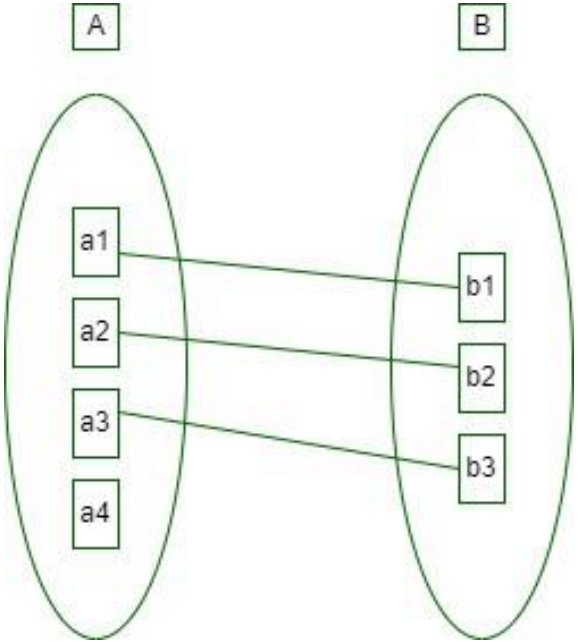
Many-to-Many Relationships

Cardinality in Relational Database Management System (RDBMS)

- Cardinality in RDBMS refers to the number of relationships between records in two related tables.
- It defines how many records from one table are associated with how many records from another table.
- There are four main types of cardinality in database relationships:
  - ✓ One-to-One (1:1),
  - ✓ One-to-Many (1:N),
  - ✓ Many-to-One (N:1), and
  - ✓ Many-to-Many (N:M)

One-to-One (1:1)

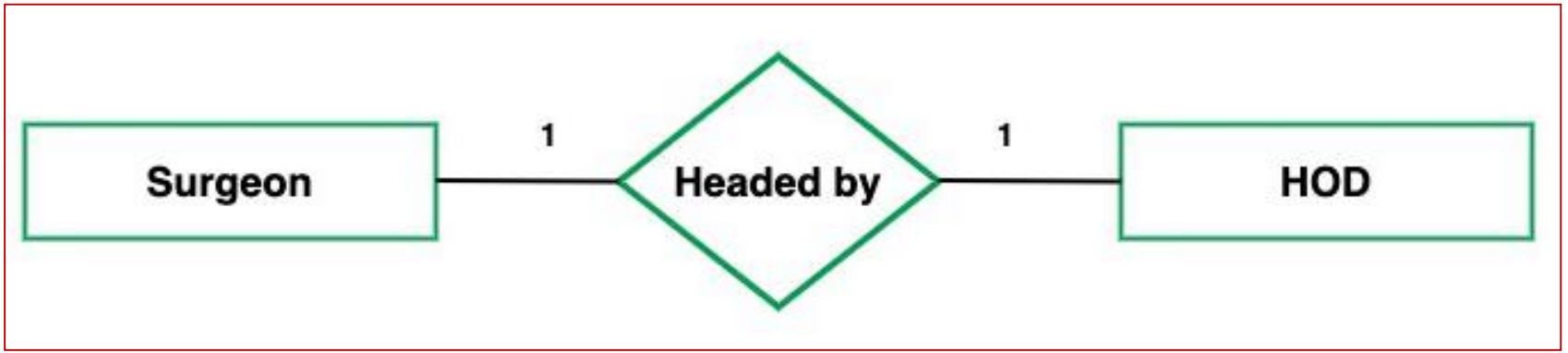
- In a one-to-one relationship, each record in one table is *associated* with only one record in another table, and vice versa. This type of relationship is relatively uncommon in databases.



One-to-One (1:1)

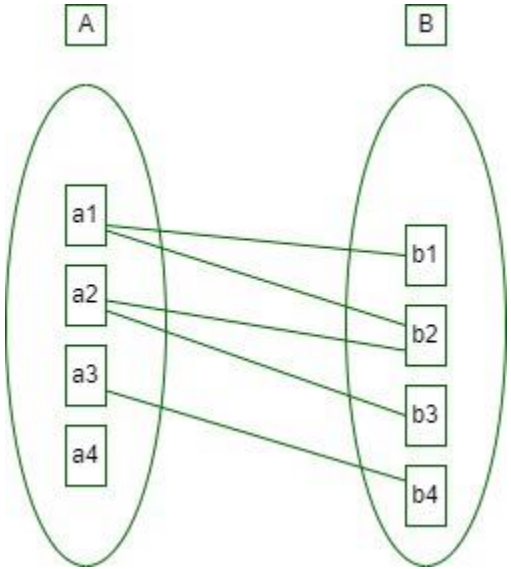
Example:

- In a particular hospital, the surgeon department has one head of department.
- They both serve one-to-one relationships.



One-to-Many (1:N)

- In a one-to-many relationship, each record in one table can be *associated* with multiple records in another table, but each record in the second table is *associated* with only one record in the first table. This is the most common type of relationship in databases



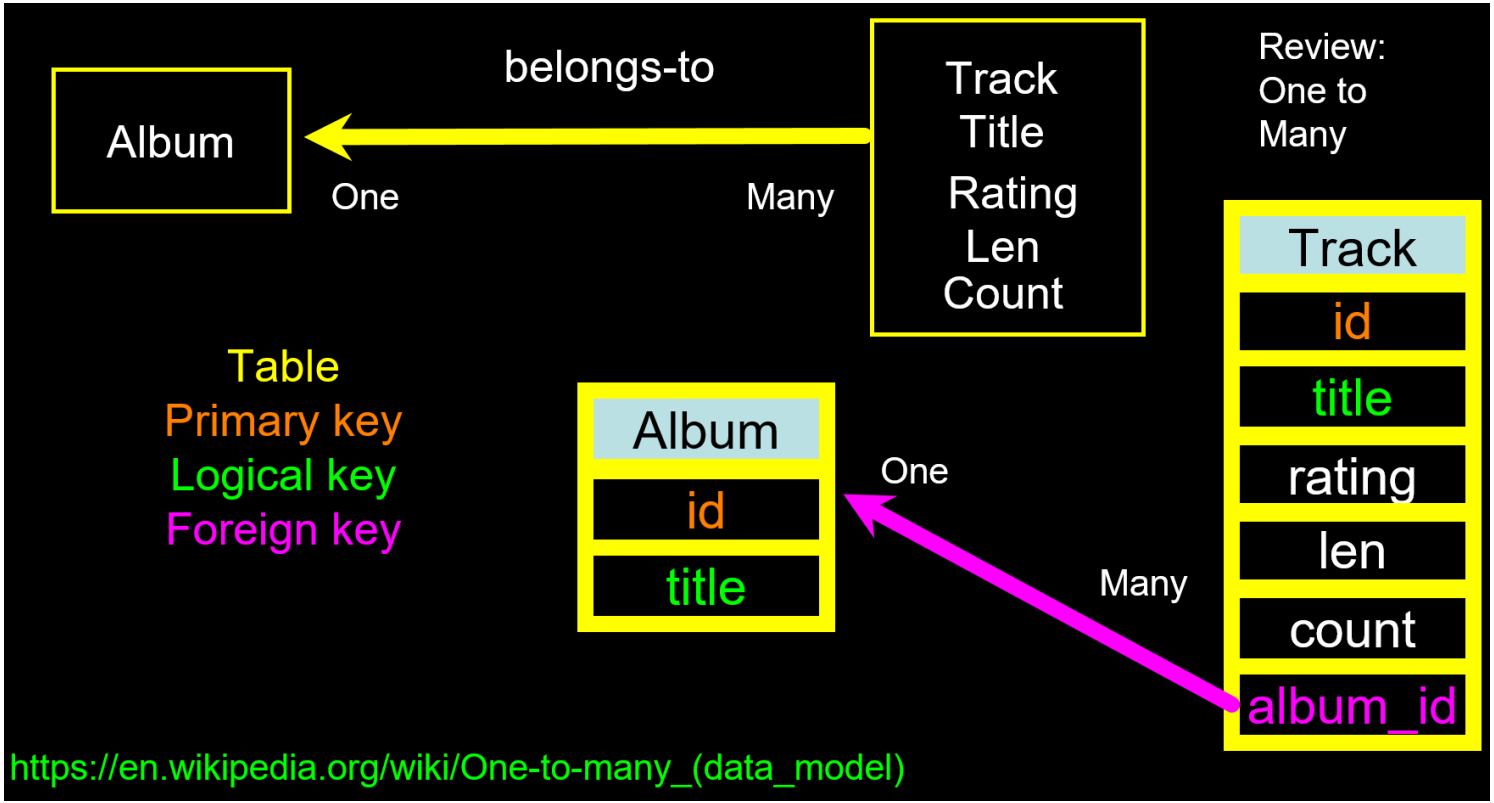
One-to-Many (1:N)

Example-1

- In a particular hospital, the surgeon department has multiple doctors. They serve one-to-many relationships.

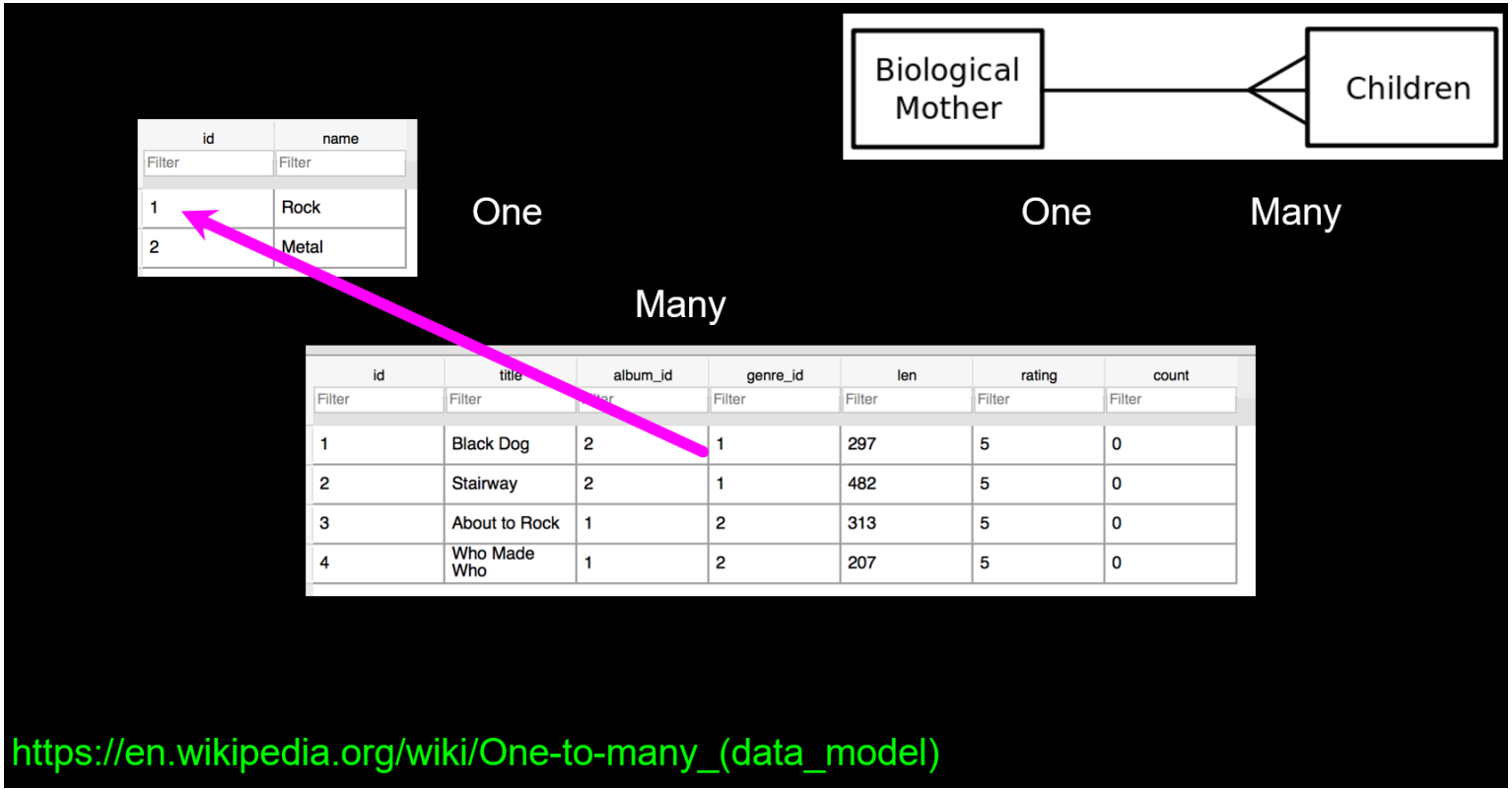
One-to-Many (1:N)

Example-2



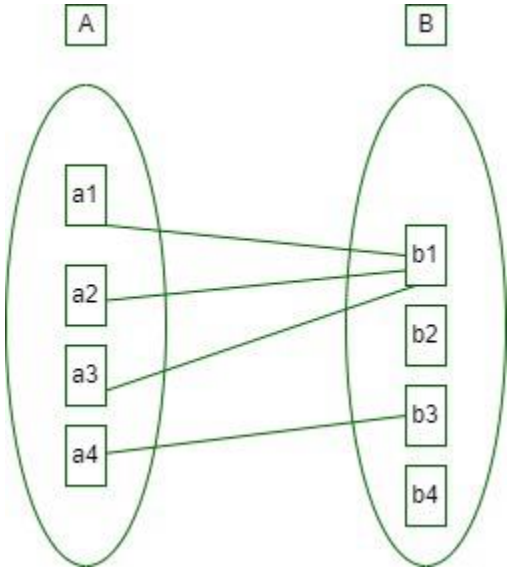
One-to-Many (1:N)

Example-2 cont'd



Many-to-One (N:1)

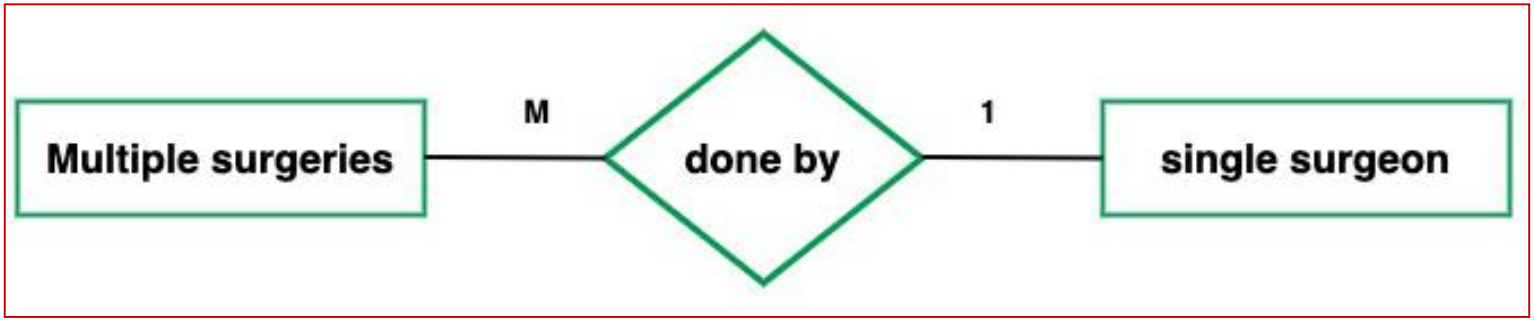
- In this type of cardinality mapping, an entity in A is *connected* to at most one entity in B.
- Or we can say a unit or item in B can be *associated* with any number (zero or more) of entities or items in A.



Many-to-One (N:1)

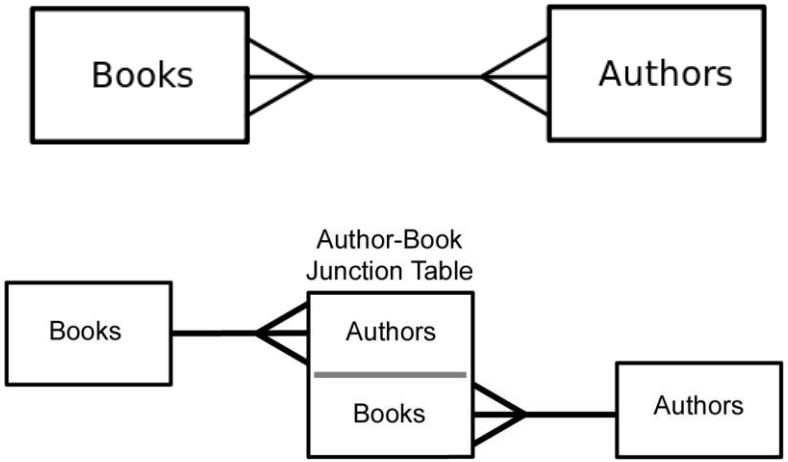
Example

- In a particular hospital, multiple surgeries are *done by* a single surgeon. Such a type of relationship is known as a many-to-one relationship.



Many-to-Many (N:M)

- Sometimes we need to model a relationship that is many-to-many
- We need to add a "connection" table with two foreign keys
- There is usually no separate primary key

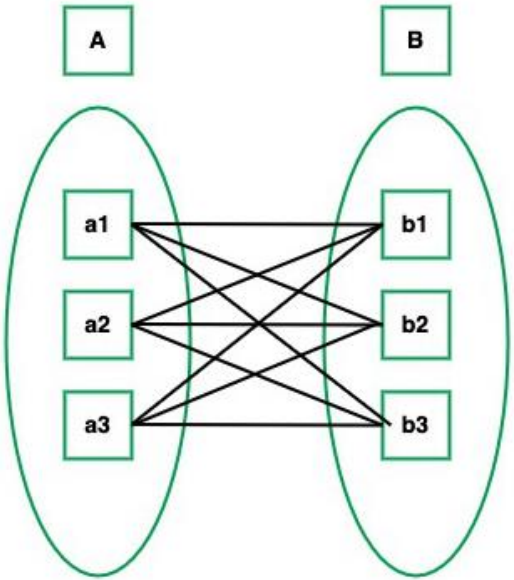


Many-to-Many (N:M)

- In a many-to-many relationship, each record in one table can be *associated* with multiple records in another table, and vice versa. This type of relationship requires a junction table to link the two main tables together.

Many-to-Many (N:M)

- In this type of cardinality mapping, an entity in A is *associated* with any number of entities in B, and an entity in B is *associated* with any number of entities in A.



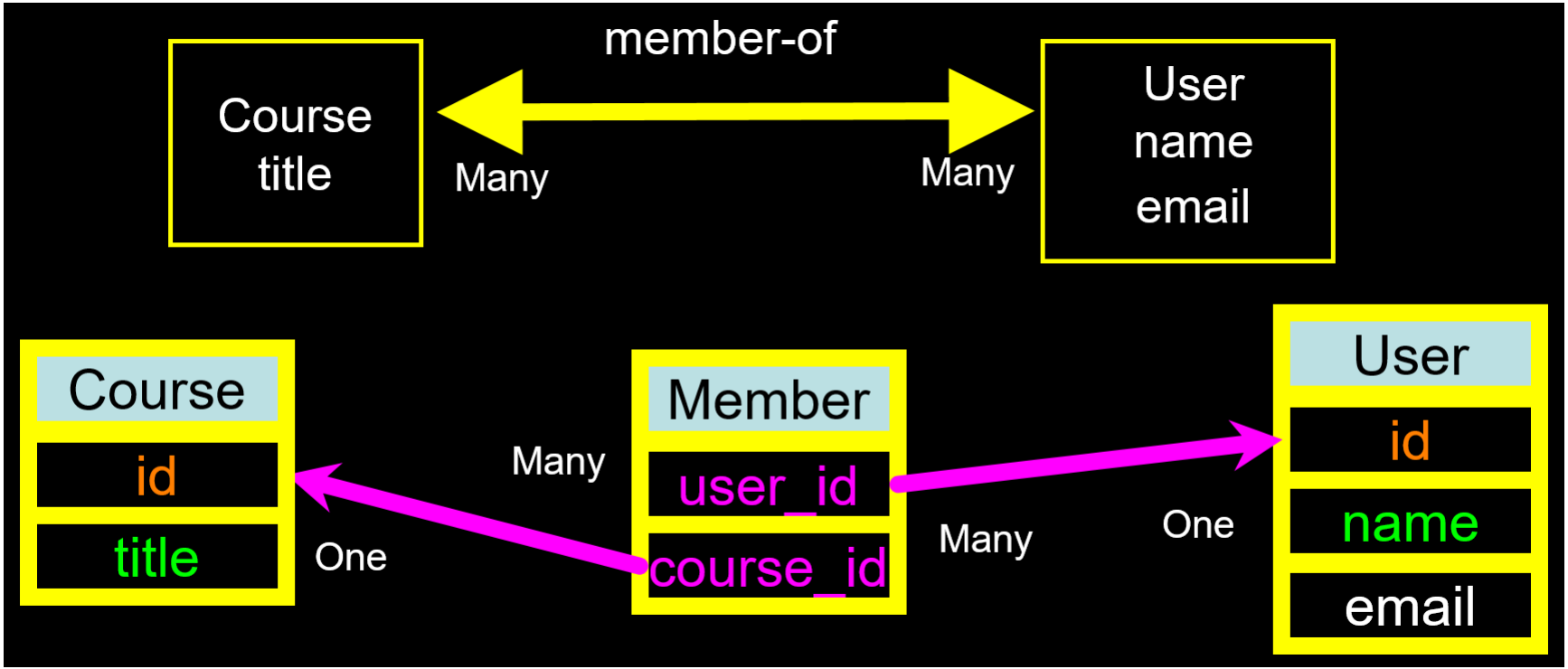
Many-to-Many (N:M)

Example-1

- In a particular company, multiple people work on multiple projects. They serve many-to-many relationships.

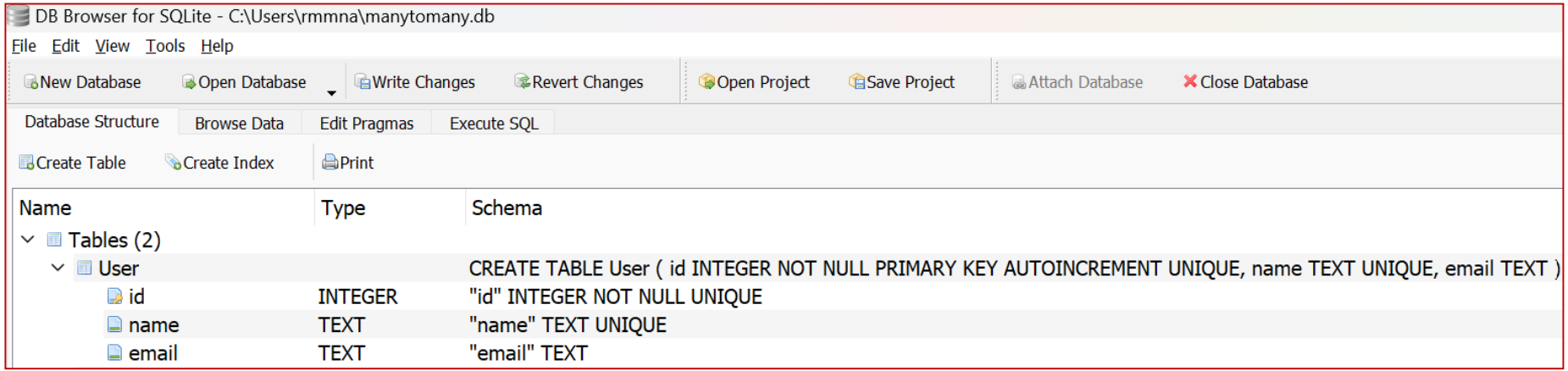
Many-to-Many (N:M)

Example-2



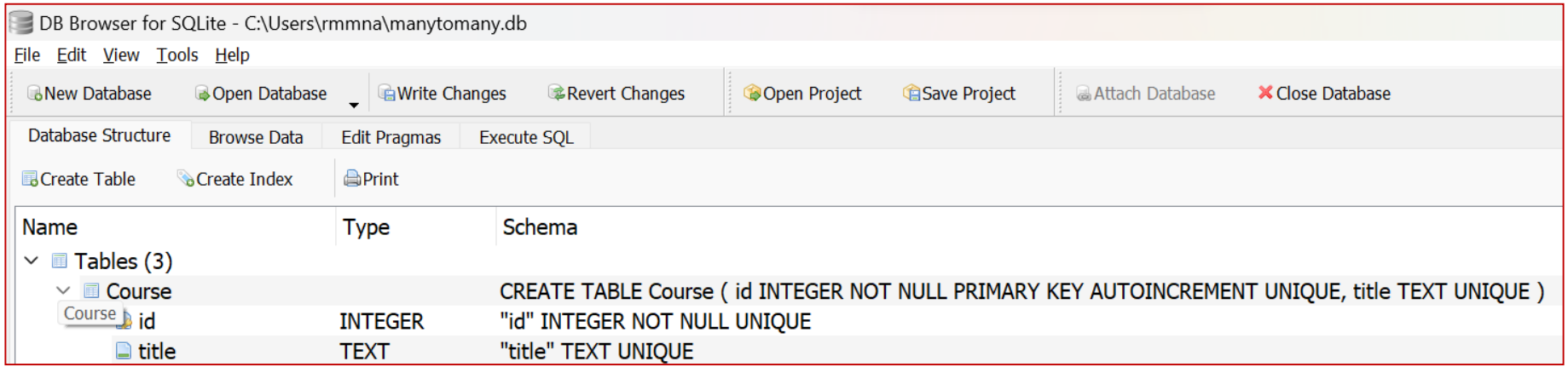
Many-to-Many (N:M): SQLite demo

```
CREATE TABLE User (
    id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    name    TEXT UNIQUE,
    email   TEXT
)
```



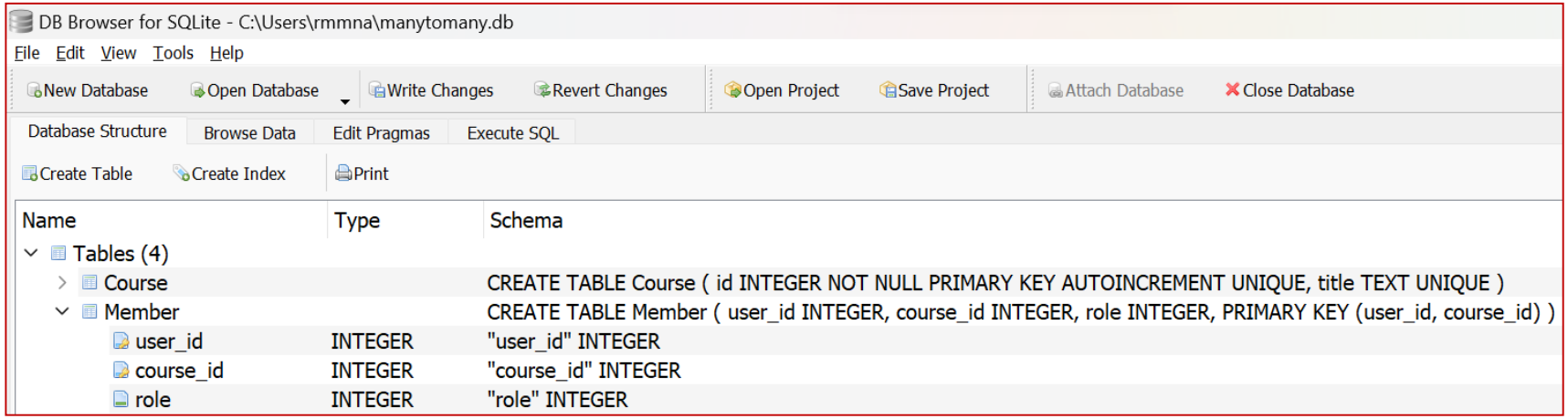
Many-to-Many (N:M): SQLite demo

```
CREATE TABLE Course (
    id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    title  TEXT UNIQUE
)
```

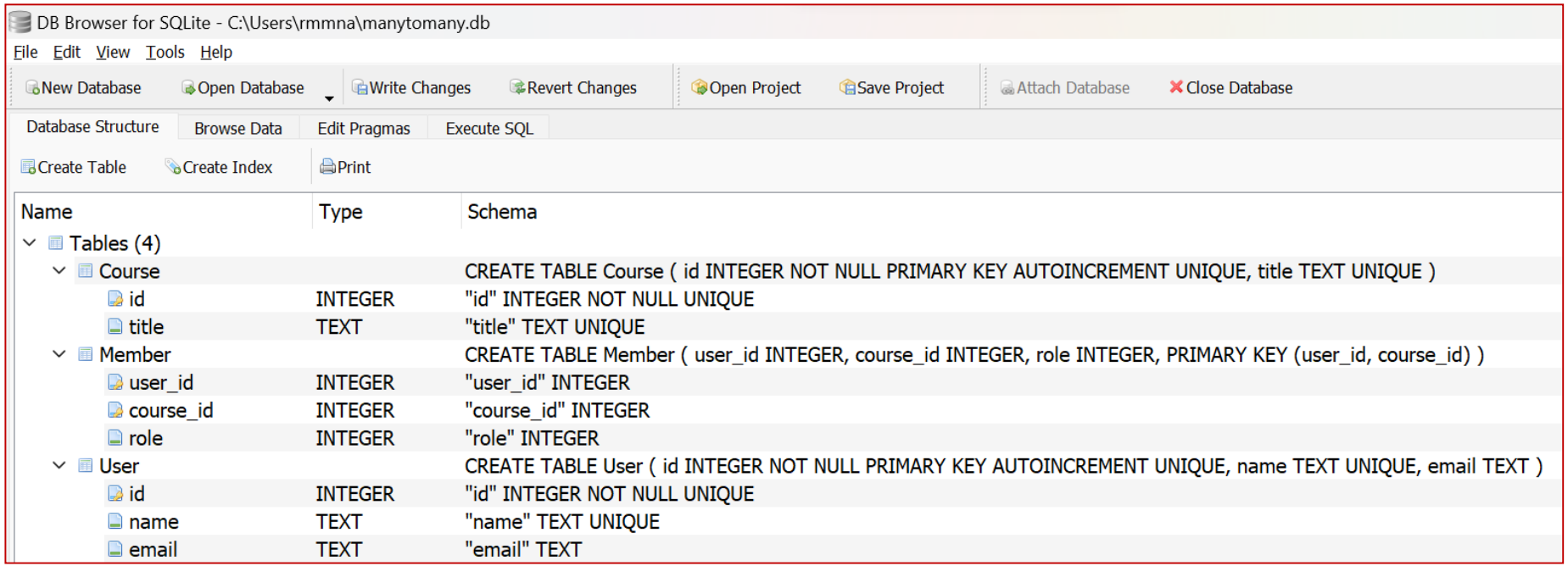


Many-to-Many (N:M): SQLite demo

```
CREATE TABLE Member (
    user_id      INTEGER,
    course_id    INTEGER,
    role         INTEGER,
    PRIMARY KEY (user_id, course_id)
)
```



Many-to-Many (N:M): SQLite demo



Many-to-Many (N:M): SQLite demo

```
INSERT INTO User (name, email) VALUES ('Prof R Madana
Mohana', 'rmmnaidu@gmail.com');

INSERT INTO User (name, email) VALUES ('Dr. R. Madana
Mohana', 'rmmnaidu@yahoo.co.in');

INSERT INTO User (name, email) VALUES ('R. Madana
Mohana', 'madanmohanar@gmail.com');
```

Many-to-Many (N:M): SQLite demo

```
Select * From User
```

	id	name	email
1	1	Prof R Madana Mohana	rmmnaidu@gmail.com
2	2	Dr. R. Madana Mohana	rmmnaidu@yahoo.co.in
3	3	R. Madana Mohana	madanmohanar@gmail.com

Many-to-Many (N:M): SQLite demo

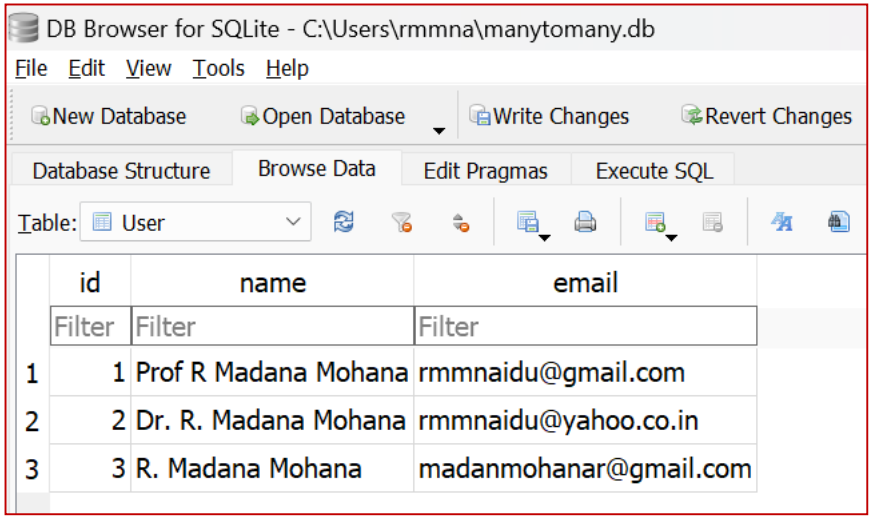
```
INSERT INTO Course (title) VALUES ('Python');

INSERT INTO Course (title) VALUES ('SQL');

INSERT INTO Course (title) VALUES ('PHP');
```

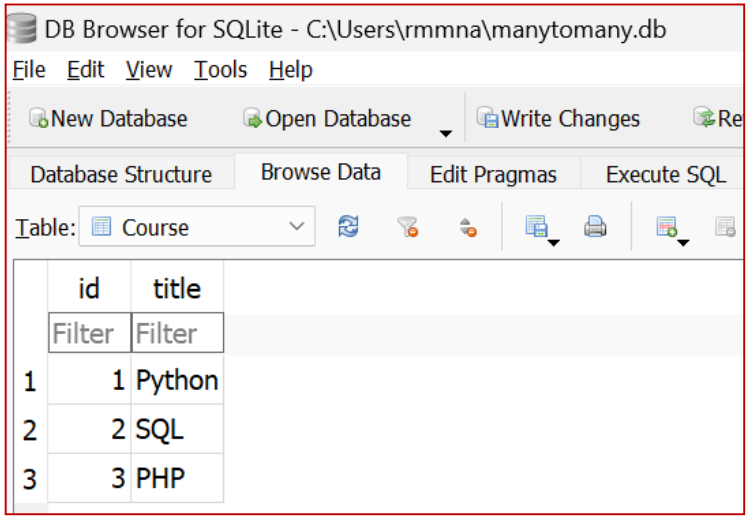
Many-to-Many (N:M): SQLite demo

```
Select * From Course
```

	id	title
1	1	Python
2	2	SQL
3	3	PHP

Many-to-Many (N:M): SQLite demo



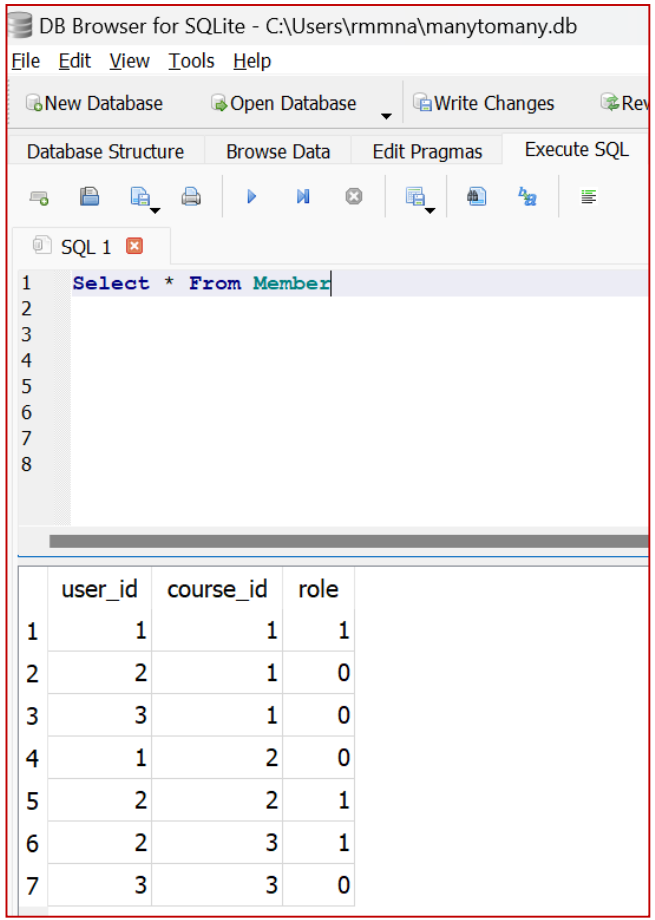


Many-to-Many (N:M): SQLite demo

```
INSERT INTO Member (user_id, course_id, role) VALUES (1, 1, 1);

INSERT INTO Member (user_id, course_id, role) VALUES (2, 1, 0);

INSERT INTO Member (user_id, course_id, role) VALUES (3, 1, 0);

INSERT INTO Member (user_id, course_id, role) VALUES (1, 2, 0);

INSERT INTO Member (user_id, course_id, role) VALUES (2, 2, 1);

INSERT INTO Member (user_id, course_id, role) VALUES (2, 3, 1);

INSERT INTO Member (user_id, course_id, role) VALUES (3, 3, 0);
```

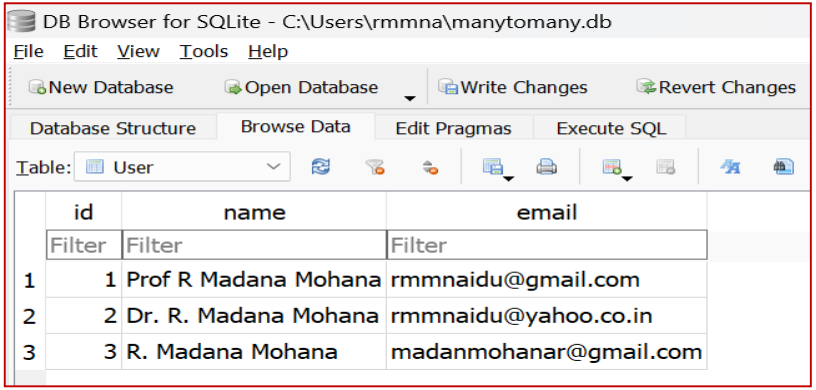
Many-to-Many (N:M): SQLite demo

```
Select * From Member;
```

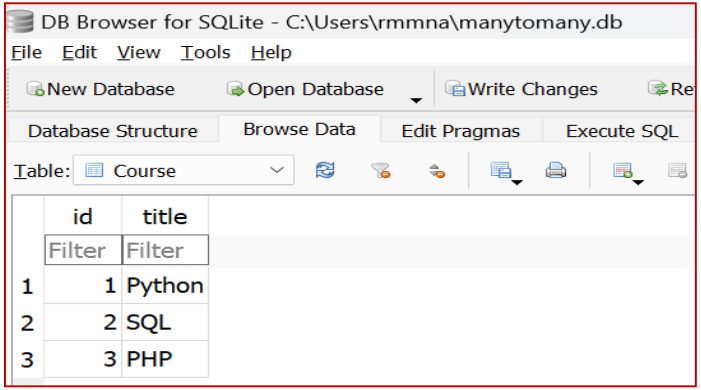


	user_id	course_id	role
1	1	1	1
2	2	1	0
3	3	1	0
4	1	2	0
5	2	2	1
6	2	3	1
7	3	3	0

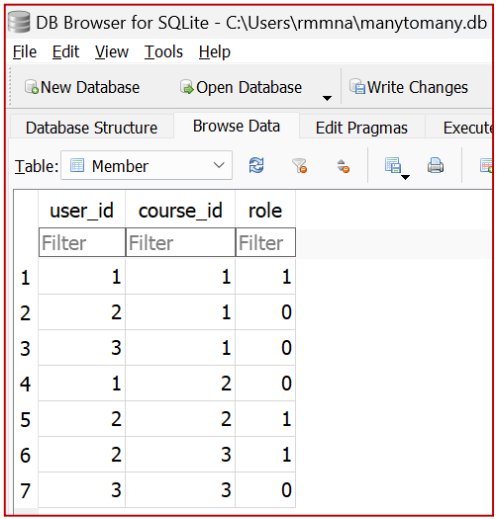
Many-to-Many (N:M): SQLite demo



	id	name	email
	Filter	Filter	Filter
1	1	Prof R Madana Mohana	rmmnaidu@gmail.com
2	2	Dr. R. Madana Mohana	rmmnaidu@yahoo.co.in
3	3	R. Madana Mohana	madanmohanar@gmail.com



	id	title
	Filter	Filter
1	1	Python
2	2	SQL
3	3	PHP



	user_id	course_id	role
	Filter	Filter	Filter
1	1	1	1
2	2	1	0
3	3	1	0
4	1	2	0
5	2	2	1
6	2	3	1
7	3	3	0

Many-to-Many (N:M): SQLite demo

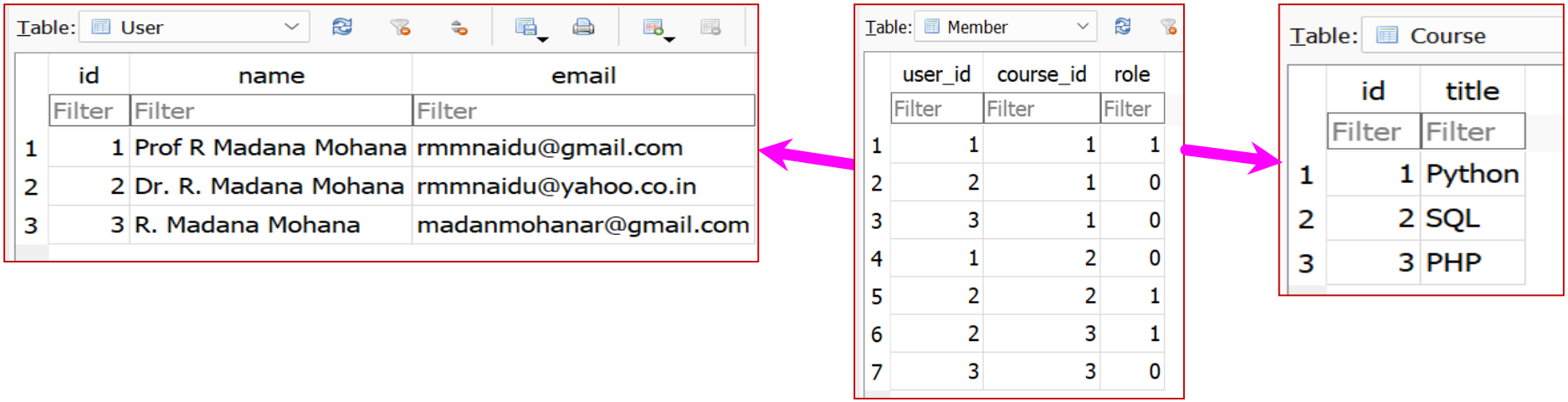


Table: User

	id	name	email
	Filter	Filter	Filter
1	1	Prof R Madana Mohana	rmmnaidu@gmail.com
2	2	Dr. R. Madana Mohana	rmmnaidu@yahoo.co.in
3	3	R. Madana Mohana	madanmohanar@gmail.com

Table: Member

	user_id	course_id	role
	Filter	Filter	Filter
1	1	1	1
2	2	1	0
3	3	1	0
4	1	2	0
5	2	2	1
6	2	3	1
7	3	3	0

Table: Course

	id	title
	Filter	Filter
1	1	Python
2	2	SQL
3	3	PHP

Many-to-Many (N:M): SQLite demo

```
SELECT User.name, Member.role, Course.title
FROM User JOIN Member JOIN Course
ON Member.user_id = User.id AND
Member.course_id = Course.id
ORDER BY Course.title, Member.role DESC, User.name
```

	name	role	title
1	Dr. R. Madana Mohana	1	PHP
2	R. Madana Mohana	0	PHP
3	Prof R Madana Mohana	1	Python
4	Dr. R. Madana Mohana	0	Python
5	R. Madana Mohana	0	Python
6	Dr. R. Madana Mohana	1	SQL
7	Prof R Madana Mohana	0	SQL

Many-to-Many (N:M) using SQLite JOIN

Python Demo

Creation of Relations

File Name: sqlitejoin_create.py

sqlitejoin_create.py - C:/Users/rmmna/AppData/Local/Programs/Python/Python310/sqlitejoin_create.py (3.10.0)

File Edit Format Run Options Window Help

```
import sqlite3
conn = sqlite3.connect('sqlitemanytomany.db')
print('Opened database successfully');
conn.execute('''CREATE TABLE User (
    id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    name   TEXT UNIQUE,
    email  TEXT
)''')
conn.execute('''CREATE TABLE Course (
    id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    title  TEXT UNIQUE
)''')
conn.execute('''CREATE TABLE Member (
    user_id     INTEGER,
    course_id   INTEGER,
    role        INTEGER,
    PRIMARY KEY (user_id, course_id)
)''')
print('Tables created successfully')
conn.close()
```

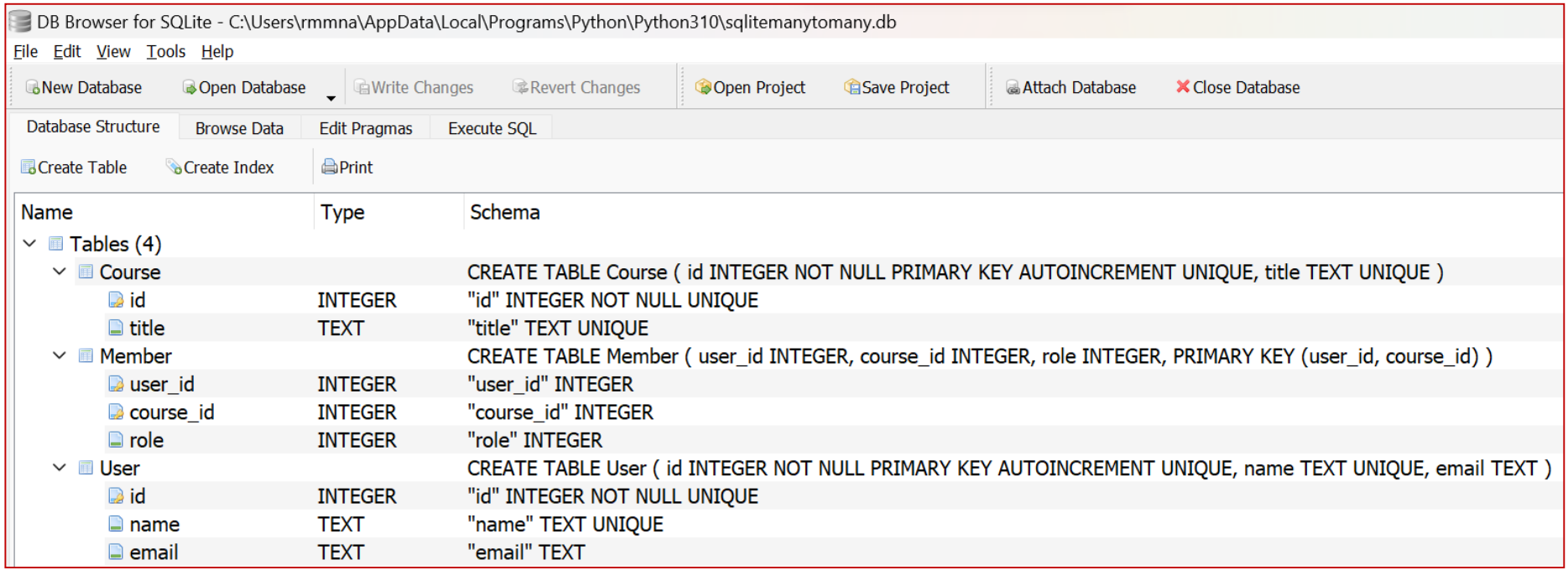
IDLE Shell 3.10.0

File Edit Shell Debug Options Window Help

```
Python 3.10.0 (tags/v3.10.0:b494f59, Oct  4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win3
2
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/rmmna/AppData/Local/Programs/Python/Python310/sqlitejoin_create.py
Opened database successfully
Tables created successfully
>>>
```

Creation of Relations

Database Name: sqlitemanytomany.db



Insertion of Data

File Name: sqlitejoin_insert.py

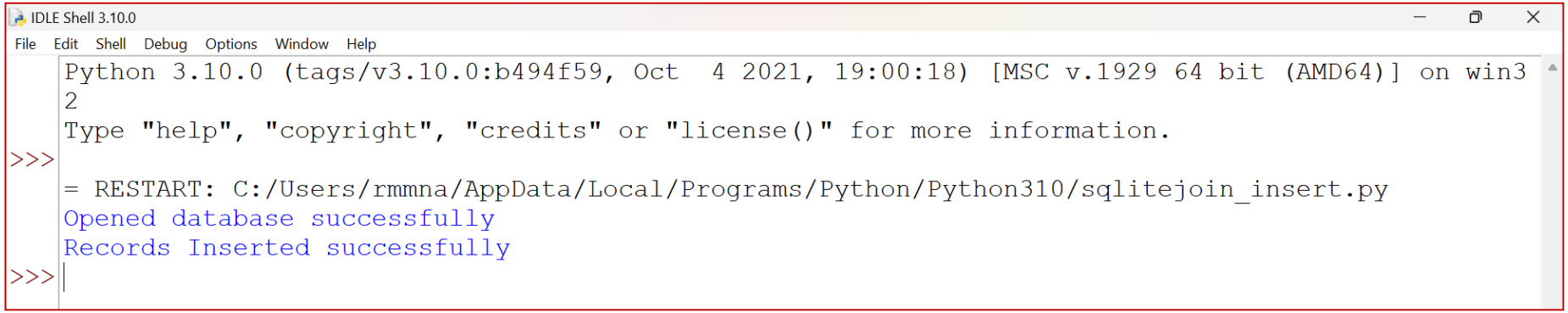
sqlitejoin_insert.py - C:/Users/rmmna/AppData/Local/Programs/Python/Python310/sqlitejoin_insert.py (3.10.0)

File Edit Format Run Options Window Help

```
import sqlite3
conn = sqlite3.connect('sqlitemanytomany.db')
print('Opened database successfully');
# Inserting data inot User relation
conn.execute("INSERT INTO User (name, email) VALUES ('Prof R Madana Mohana', 'rmmnaidu@gmail.com')"
conn.execute("INSERT INTO User (name, email) VALUES ('Dr. R. Madana Mohana', 'rmmnaidu@yahoo.co.in'
conn.execute("INSERT INTO User (name, email) VALUES ('R. Madana Mohana', 'madanmohanar@gmail.com')"
# Inserting data inot Course relation
conn.execute("INSERT INTO Course (title) VALUES ('Python')")
conn.execute("INSERT INTO Course (title) VALUES ('SQL')")
conn.execute("INSERT INTO Course (title) VALUES ('PHP')")
# Inserting data inot Member relation
conn.execute("INSERT INTO Member (user_id, course_id, role) VALUES (1, 1, 1)")
conn.execute("INSERT INTO Member (user_id, course_id, role) VALUES (2, 1, 0)")
conn.execute("INSERT INTO Member (user_id, course_id, role) VALUES (3, 1, 0)")
conn.execute("INSERT INTO Member (user_id, course_id, role) VALUES (1, 2, 0)")
conn.execute("INSERT INTO Member (user_id, course_id, role) VALUES (2, 2, 1)")
conn.execute("INSERT INTO Member (user_id, course_id, role) VALUES (2, 3, 1)")
conn.execute("INSERT INTO Member (user_id, course_id, role) VALUES (3, 3, 0)")
conn.commit()
print('Records Inserted successfully')
conn.close()
```

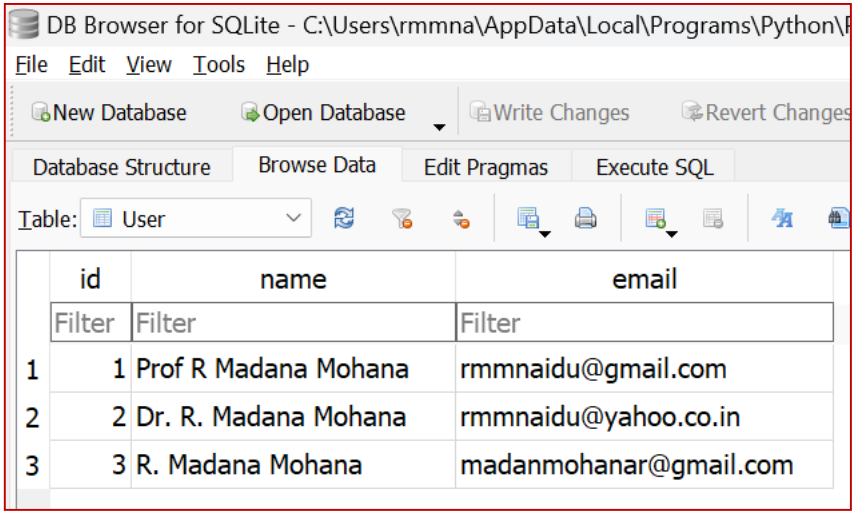
Insertion of Data

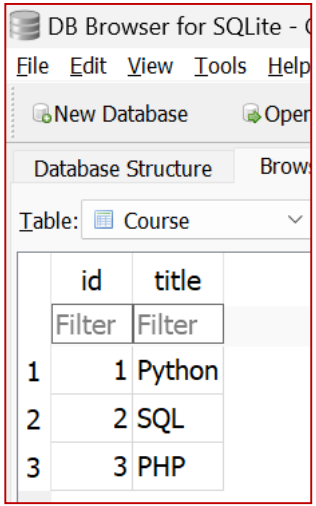
File Name: sqlitejoin_insert.py

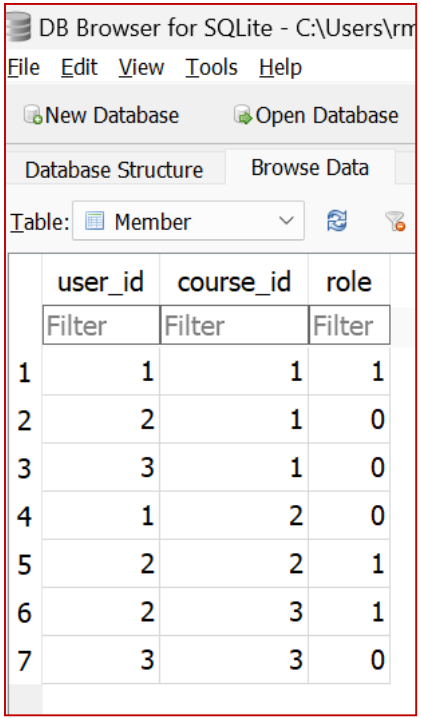


Insertion of Data

File Name: sqlitejoin_insert.py







Join Operation

File Name: sqlitejoin_select.py

```
import sqlite3
conn = sqlite3.connect('sqlitemanytomany.db')
print('Opened database successfully');
#Join Operation
cursor=conn.execute("SELECT User.name, Member.role, Course.title
FROM User JOIN Member JOIN Course ON Member.user_id = User.id AND
Member.course_id = Course.id ORDER BY Course.title, Member.role
DESC, User.name")
print('JOIN Operation Results Retrieval:')
for row in cursor:
    print("Name = ", row[0])
    print("Role = ", row[1])
    print("Title = ", row[2], "\n")
print('Records Retrieved successfully')
conn.close()
```

Join Operation

File Name: sqlitejoin_select.py