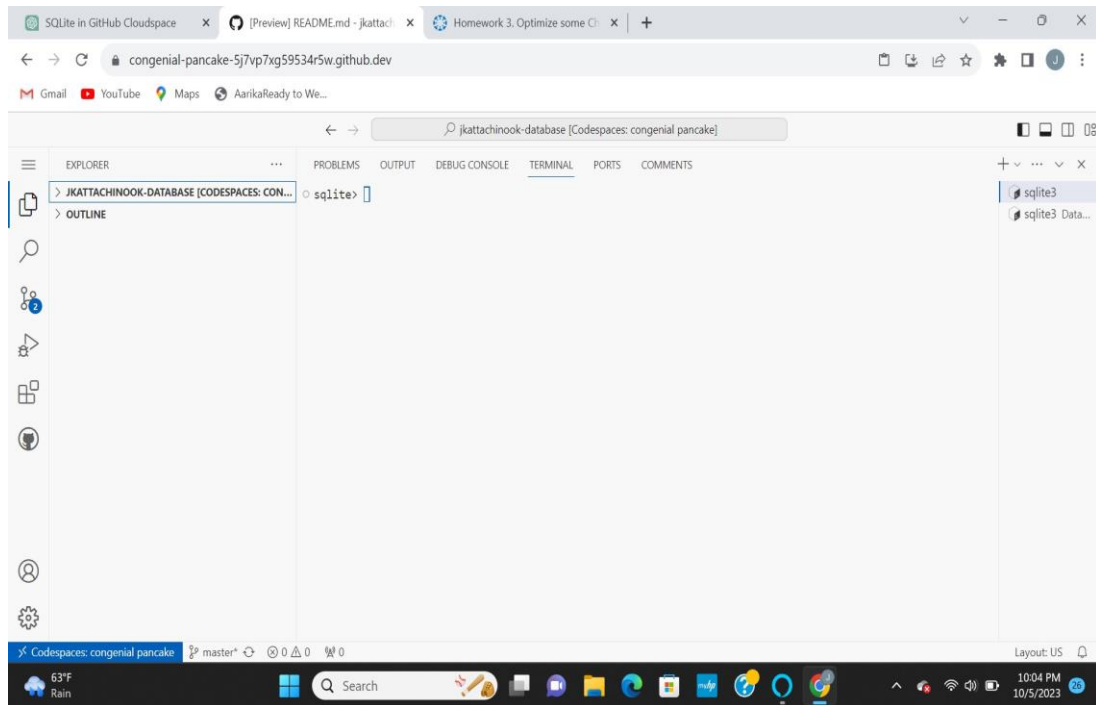


Homework-3

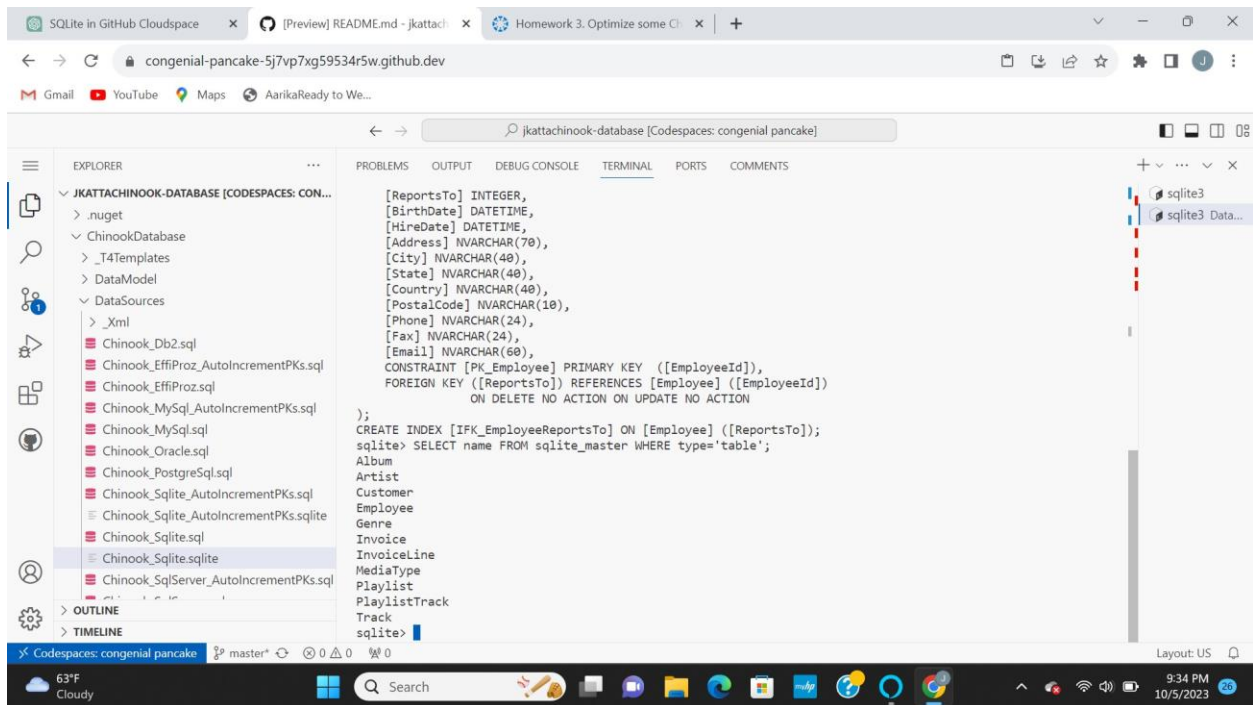
Optimize some Chinook queries

1. Get the Chinook database running on your workspace, whether in your codespace or on your computer. Use the SQLite version. You have several options there.

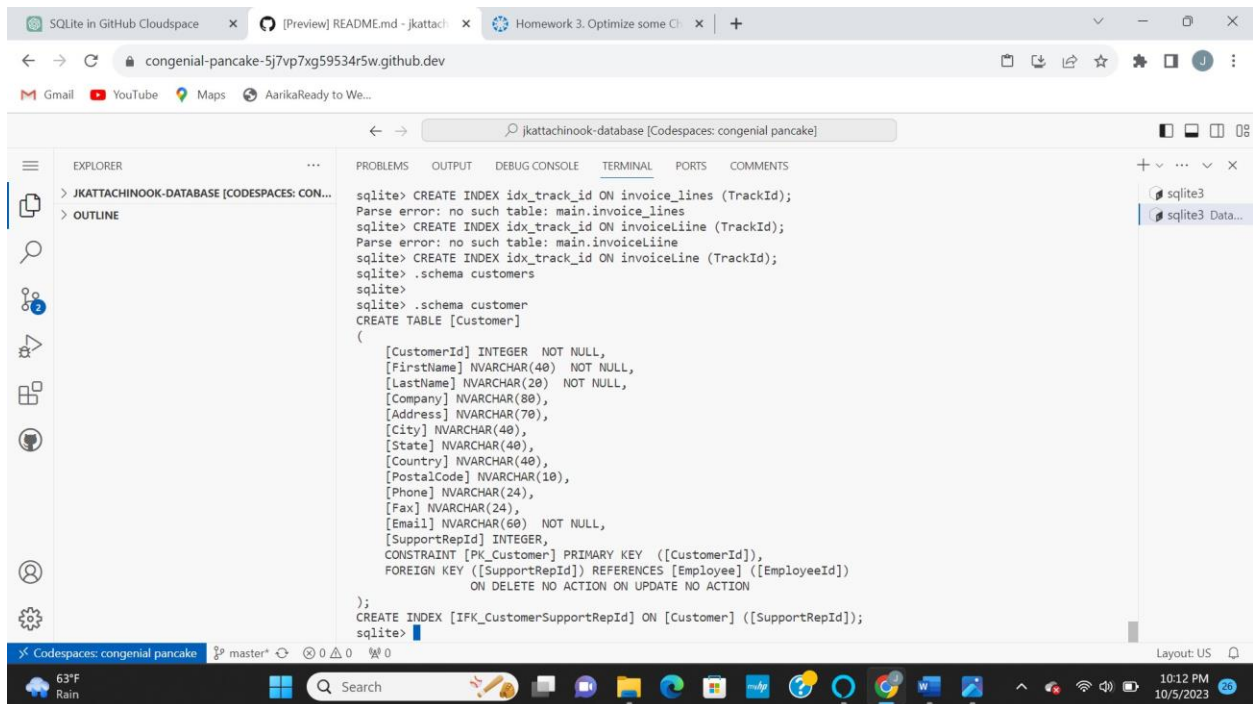


2. Try some queries to verify that the tables and relations are working.

2.1 To list all tables



2.2 To view Scheme



2.3 Join Tables

The screenshot shows a Visual Studio Code window with a SQLite database connection. The Explorer pane on the left shows the project structure, including a folder named 'JKATTACHINOOK-DATABASE [CODESPACES: CON...]'. The Terminal pane on the right shows the following SQL query and its result:

```
CREATE INDEX [IFK_InvoiceCustomerId] ON [Invoice] ([CustomerId]);
sqlite> SELECT customer.FirstName, customer.LastName, invoice.InvoiceId
FROM customer
INNER JOIN invoice ON customer.CustomerId = invoice.CustomerId;
```

The result of the query is displayed in the right-hand pane, showing a list of customers and their corresponding invoice IDs. The list includes customers like Luis Gonçalves, Leonie Köhler, and François Tremblay, each with multiple invoice IDs.

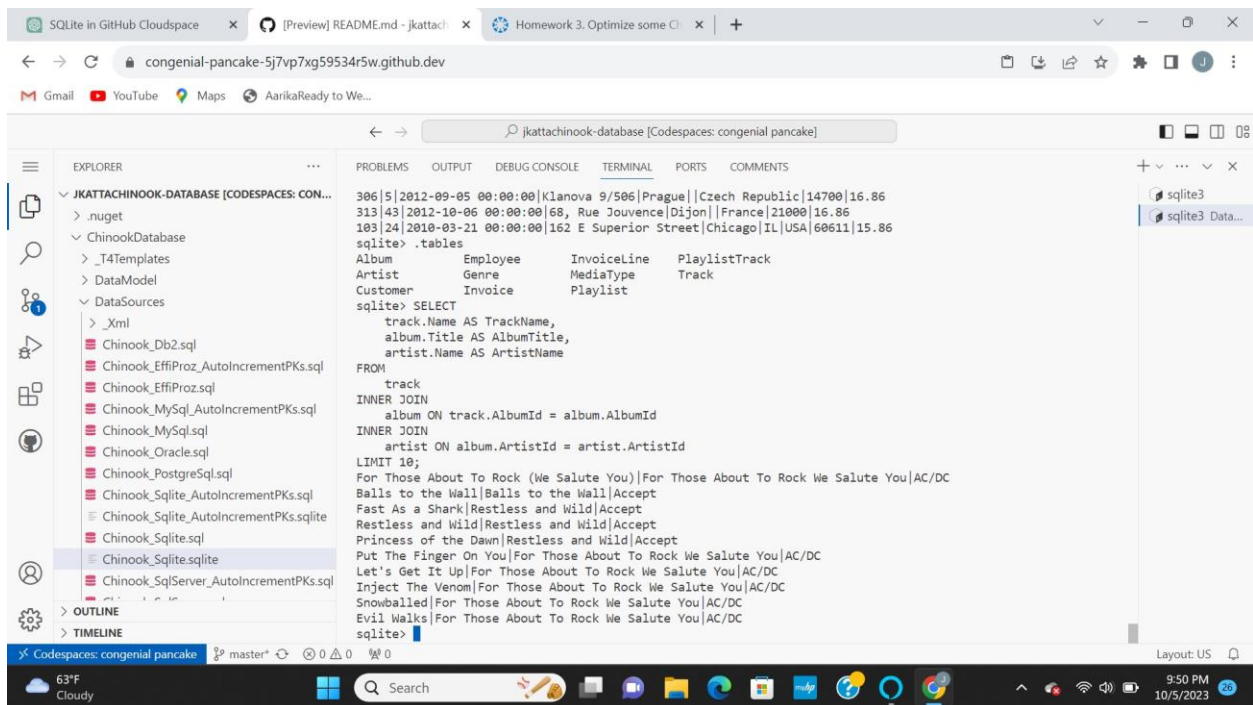
2.4 order and limit results

The screenshot shows a Visual Studio Code window with a SQLite database connection. The Explorer pane on the left shows the project structure, including a folder named 'JKATTACHINOOK-DATABASE [CODESPACES: CON...]'. The Terminal pane on the right shows the following SQL query and its result:

```
Parse error: no such table: invoices
sqlite> SELECT FirstName, LastName
FROM customer
WHERE CustomerId IN (SELECT CustomerId FROM invoice WHERE Total > 100.0);
sqlite> SELECT *
FROM invoices
ORDER BY Total DESC
LIMIT 10;
```

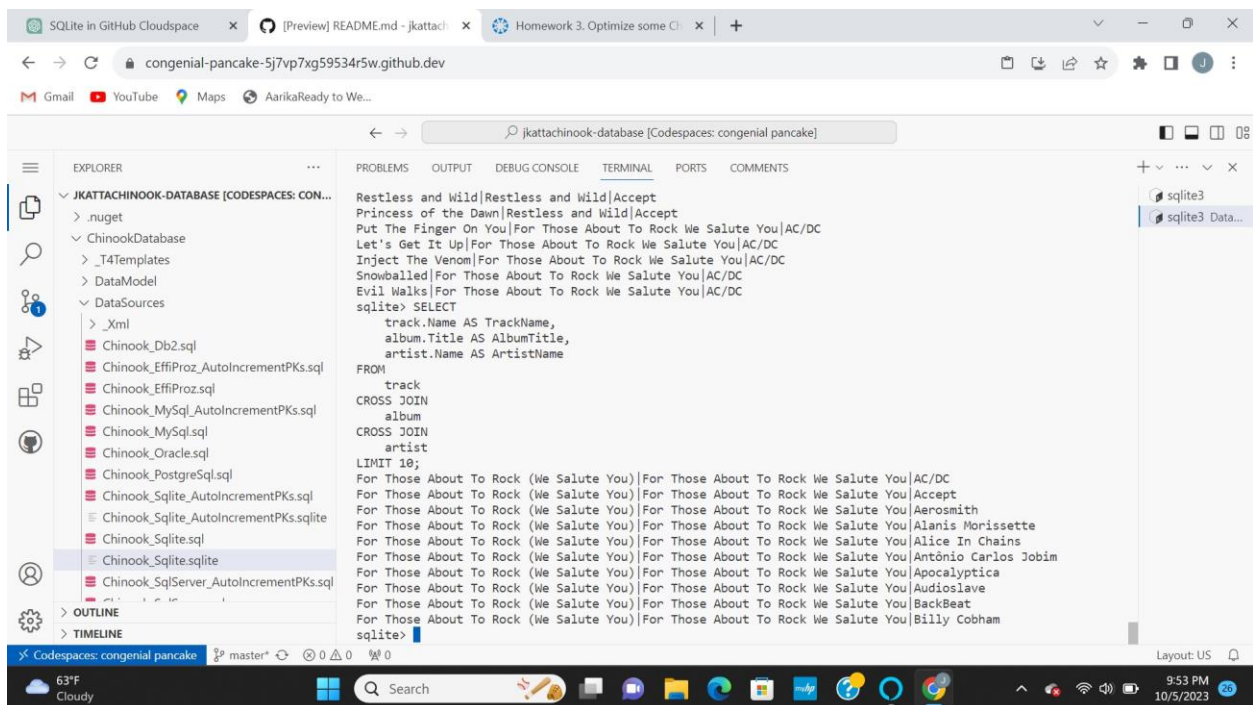
The result of the query is displayed in the right-hand pane, showing a list of customers and their corresponding invoice IDs. The list includes customers like Rilska, Erzsébet, Chatham, Rotenturmstraße, and Calle Lira, each with multiple invoice IDs.

3. Try a query statement that joins across at least three tables.



```
306|5|2012-09-05 00:00:00|Klanova 9/506|Prague||Czech Republic|14700|16.86
313|43|2012-10-06 00:00:00|68, Rue Jouvence|Dijon||France|21000|16.86
103|24|2010-03-21 00:00:00|162 E Superior Street|Chicago|IL|USA|60611|15.86
sqlite> .tables
Album      Employee   InvoiceLine PlaylistTrack
Artist      Genre      MediaType  Track
Customer   Invoice     Playlist
sqlite> SELECT
  track.Name AS TrackName,
  album.Title AS AlbumTitle,
  artist.Name AS ArtistName
FROM
  track
INNER JOIN
  album ON track.AlbumId = album.AlbumId
INNER JOIN
  artist ON album.ArtistId = artist.ArtistId
LIMIT 10;
For Those About To Rock (We Salute You)|For Those About To Rock We Salute You|AC/DC
Balls to the Wall|Balls to the Wall|Accept
Fast As a Shark|Restless and Wild|Accept
Restless and Wild|Restless and Wild|Accept
Princess of the Dawn|Restless and Wild|Accept
Put The Finger On You|For Those About To Rock We Salute You|AC/DC
Let's Get It Up|For Those About To Rock We Salute You|AC/DC
Inject The Venom|For Those About To Rock We Salute You|AC/DC
Snowballed|For Those About To Rock We Salute You|AC/DC
Evil Walks|For Those About To Rock We Salute You|AC/DC
sqlite>
```

4. Try to create a query statement that does the join using _unindexed_ records for all joins.



```
Restless and Wild|Restless and Wild|Accept
Princess of the Dawn|Restless and Wild|Accept
Put The Finger On You|For Those About To Rock We Salute You|AC/DC
Let's Get It Up|For Those About To Rock We Salute You|AC/DC
Inject The Venom|For Those About To Rock We Salute You|AC/DC
Snowballed|For Those About To Rock We Salute You|AC/DC
Evil Walks|For Those About To Rock We Salute You|AC/DC
sqlite> SELECT
  track.Name AS TrackName,
  album.Title AS AlbumTitle,
  artist.Name AS ArtistName
FROM
  track
CROSS JOIN
  album
CROSS JOIN
  artist
LIMIT 10;
For Those About To Rock (We Salute You)|For Those About To Rock We Salute You|AC/DC
For Those About To Rock (We Salute You)|For Those About To Rock We Salute You|Accept
For Those About To Rock (We Salute You)|For Those About To Rock We Salute You|Aerosmith
For Those About To Rock (We Salute You)|For Those About To Rock We Salute You|Alanis Morissette
For Those About To Rock (We Salute You)|For Those About To Rock We Salute You|Alice In Chains
For Those About To Rock (We Salute You)|For Those About To Rock We Salute You|Antônio Carlos Jobim
For Those About To Rock (We Salute You)|For Those About To Rock We Salute You|Apocalyptic
For Those About To Rock (We Salute You)|For Those About To Rock We Salute You|Audioslave
For Those About To Rock (We Salute You)|For Those About To Rock We Salute You|BackBeat
For Those About To Rock (We Salute You)|For Those About To Rock We Salute You|Billy Cobham
sqlite>
```

5. Find a way to time the query from #4 using tools available. Find the average of 10 queries. Vary the details a bit to avoid cache effects.

One way to time query is. timer ON

Average of 10 queries = 0.0005

6. Create some index capabilities for the tables that will speed up the queries.

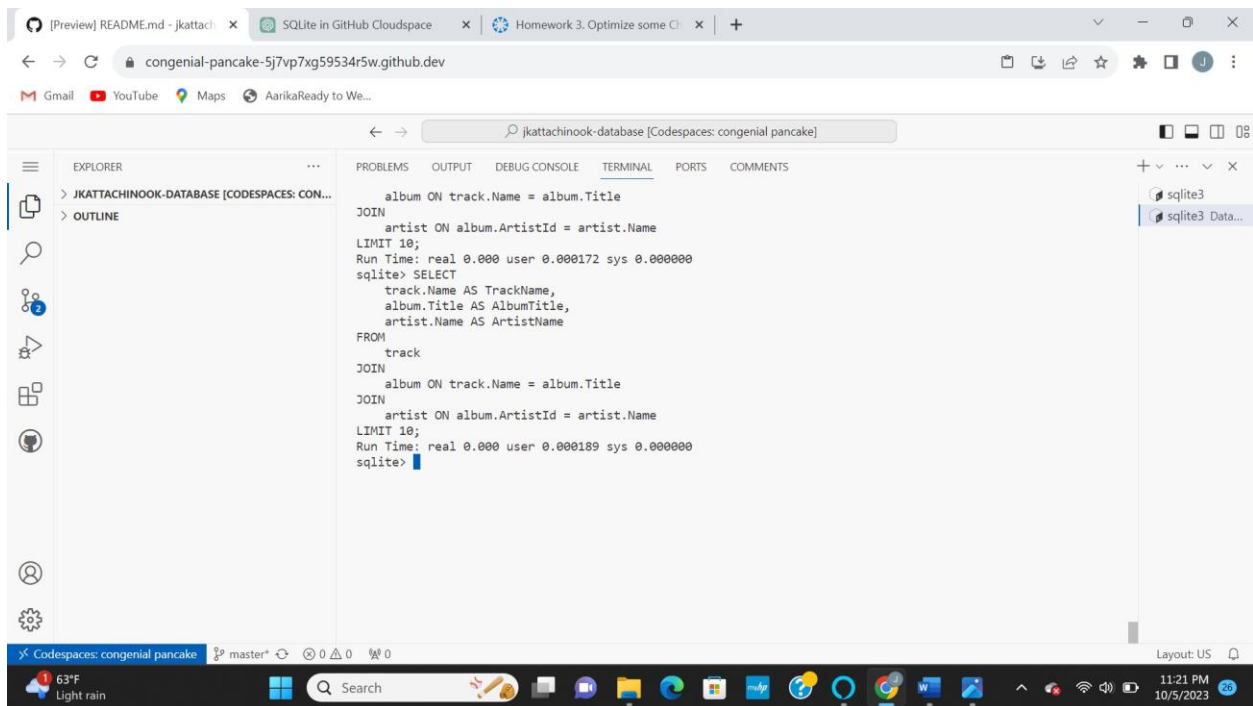
I have created 2 Index capabilities they are

```
CREATE INDEX idx_media_type_id ON media_types (MediaTypeId);
```

```
CREATE INDEX idx_genre_id ON genres (GenreId);
```

7. Time the queries again, to see if the effect worked.

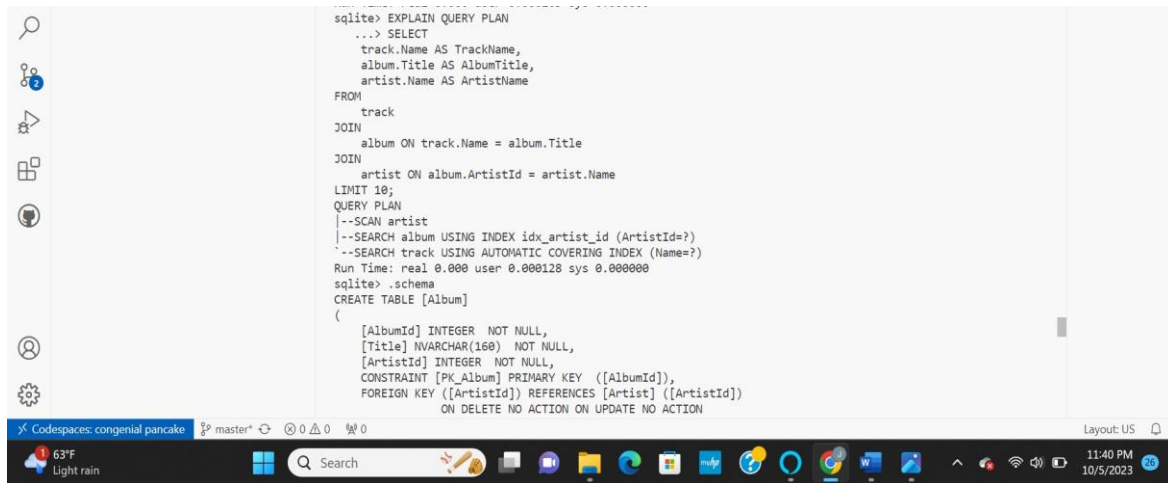
Yes, There is an effect it went from 0.001 to 0.000



The screenshot shows a VS Code editor with a SQLite database named 'jkattachinook-database'. The Explorer pane on the left shows the database structure. The main editor area displays a SQL query and its execution results. The query is a JOIN between the 'album' and 'track' tables, filtered by 'album.ArtistId = artist.Name', with a LIMIT of 10. The execution results show the query was run successfully, with a real time of 0.000189 seconds, user time of 0.000172 seconds, and system time of 0.000000 seconds. The results are displayed in a table with columns: track.Name AS TrackName, album.Title AS AlbumTitle, and artist.Name AS ArtistName. The table contains 10 rows of data.

```
album ON track.Name = album.Title
JOIN
  artist ON album.ArtistId = artist.Name
LIMIT 10;
Run Time: real 0.000 user 0.000172 sys 0.000000
sqlite> SELECT
  track.Name AS TrackName,
  album.Title AS AlbumTitle,
  artist.Name AS ArtistName
FROM
  track
JOIN
  album ON track.Name = album.Title
JOIN
  artist ON album.ArtistId = artist.Name
LIMIT 10;
Run Time: real 0.000 user 0.000189 sys 0.000000
sqlite>
```


8. Look at the query plan to make sure the index capabilities are being used.



The screenshot shows a SQLite query plan in a code editor. The query is a SELECT statement with three columns: track.Name AS TrackName, album.Title AS AlbumTitle, and artist.Name AS ArtistName. It joins the track, album, and artist tables. The query plan shows that the artist table is scanned, the album table is searched using the idx_artist_id index, and the track table is searched using an automatic covering index. The query is limited to 10 rows. The schema for the album table is also shown, indicating it has a primary key on AlbumId and a foreign key on ArtistId.

```
sqlite> EXPLAIN QUERY PLAN
...> SELECT
  track.Name AS TrackName,
  album.Title AS AlbumTitle,
  artist.Name AS ArtistName
FROM
  track
JOIN
  album ON track.Name = album.Title
JOIN
  artist ON album.ArtistId = artist.Name
LIMIT 10;
QUERY PLAN
|--SCAN artist
|--SEARCH album USING INDEX idx_artist_id (ArtistId=?)
|--SEARCH track USING AUTOMATIC COVERING INDEX (Name=?)
Run Time: real 0.000 user 0.000128 sys 0.000000
sqlite> .schema
CREATE TABLE [Album]
(
  [AlbumId] INTEGER NOT NULL,
  [Title] NVARCHAR(160) NOT NULL,
  [ArtistId] INTEGER NOT NULL,
  CONSTRAINT [PK_Album] PRIMARY KEY ([AlbumId]),
  FOREIGN KEY ([ArtistId]) REFERENCES [Artist] ([ArtistId])
  ON DELETE NO ACTION ON UPDATE NO ACTION
)
```