DBeaver for Exploring Enterprise IT

DBeaver is a tool for connecting to, querying, and managing various relational databases. It appears quite fit for allowing consultants to survey what data is collected and used. DBeaver is an open-source application with commercial support. I'm guessing commercial users would buy either the "Enterprise" or "Cloud" version. Consultants to those companies could then use DBeaver in the short term to explore data for use in analyses, creating CSV files (or JSON, etc.) for download.

For long-term use, consultants would likely skip DBeaver and go straight to running against the DBs using the DBs native SQL interface (or similar). DBeaver helps in that work too by providing information you'd need to make the connection; see section *Using DBeaver Connection Settings to Learn How to Connect* below. My *dbeaver-exp* git repository has a Jupyter notebook that demonstrates connecting to DBeaver demo SQLite database. See https://github.com/pdenno/dbeaver-exp/blob/main/sqlite-connection.ipynb.) The demo DB is unprotected. Before trying things like this with a commercial concern, consultants should work out the details with the customer.

I think it makes sense to have people doing consulting-like work to start with DBeaver because it provides read permission across many DBs for exploration… I guess(!); I'm just learning it today!
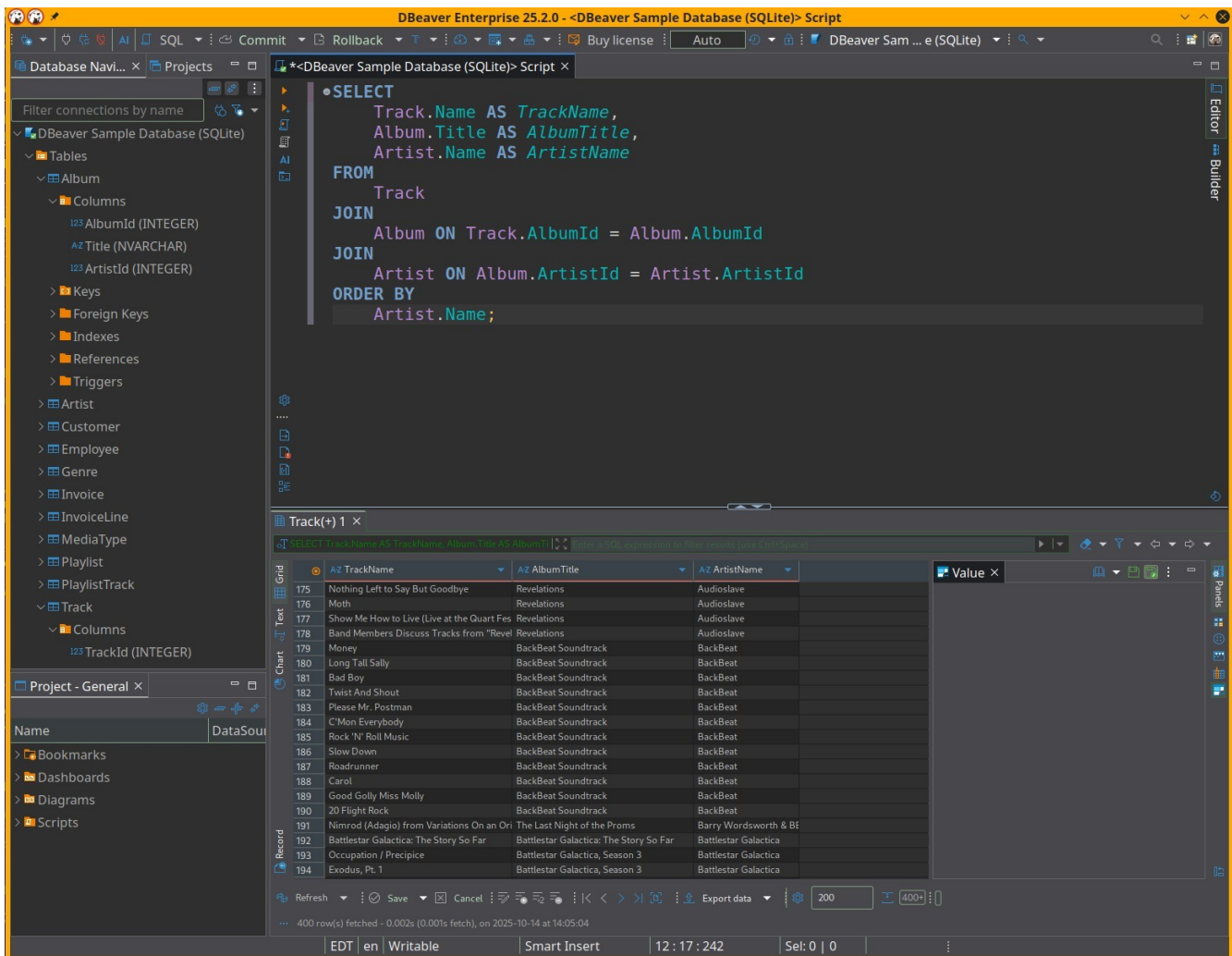
Figure 1: Here I am exploring the DBeaver demo DB with the Enterprise edition. I chose the "DBeaver Sample Database (SQLite) found in upper left of the figure. I wrote the SQL at the top center, and ran it using one of the buttons to the left of the SQL editor (hover over those buttons to see what they do). It returned the data in the bottom center.

Though DBeaver has an AI tool to help you write SQL (see the little blue "AI" button to the left of the SQL editing region), I doubt it is active in most deployment because it sends the request outside (to OpenAI).

Below I right clicked on the data region and selected the "Export data" menu item to save the result of the query.
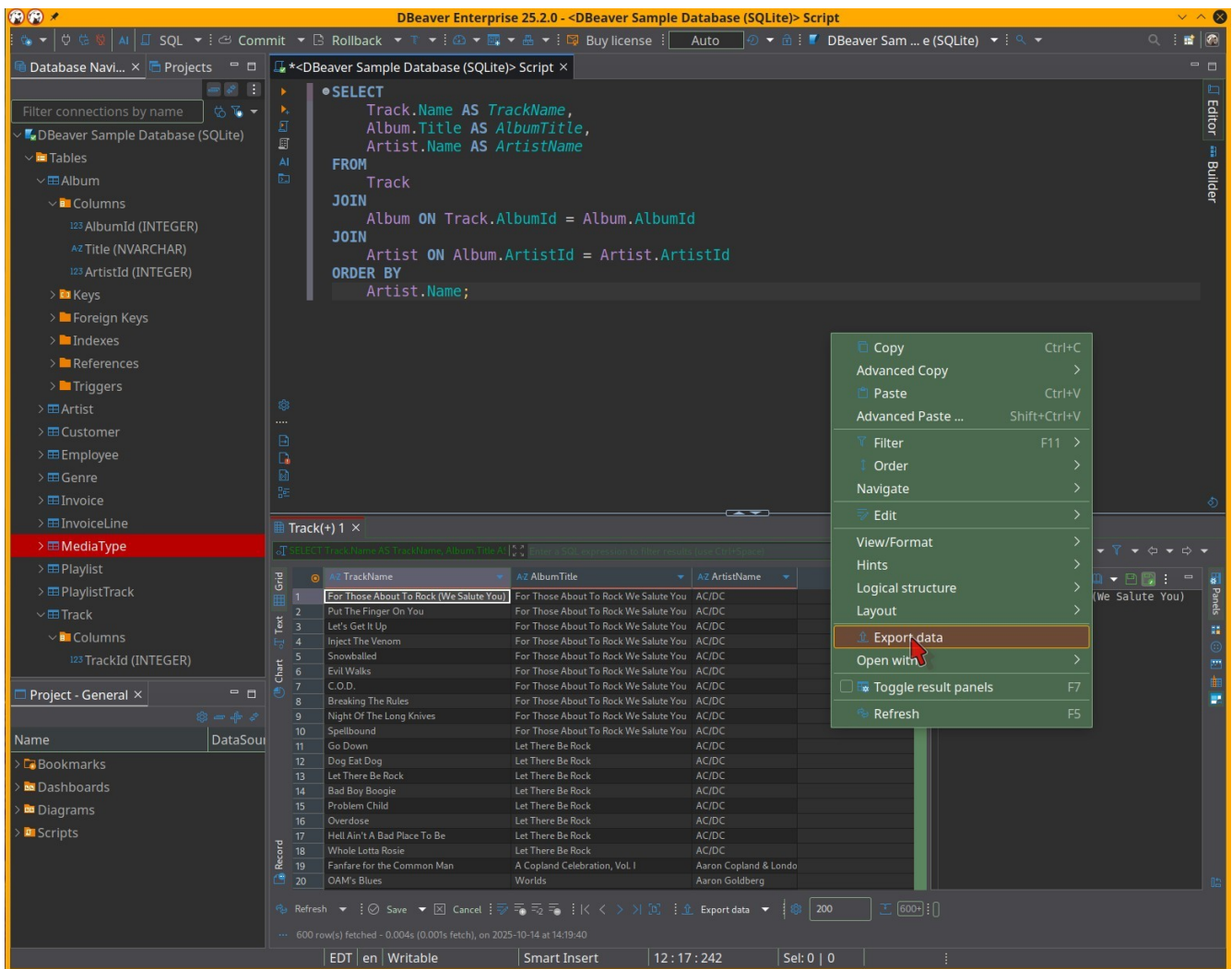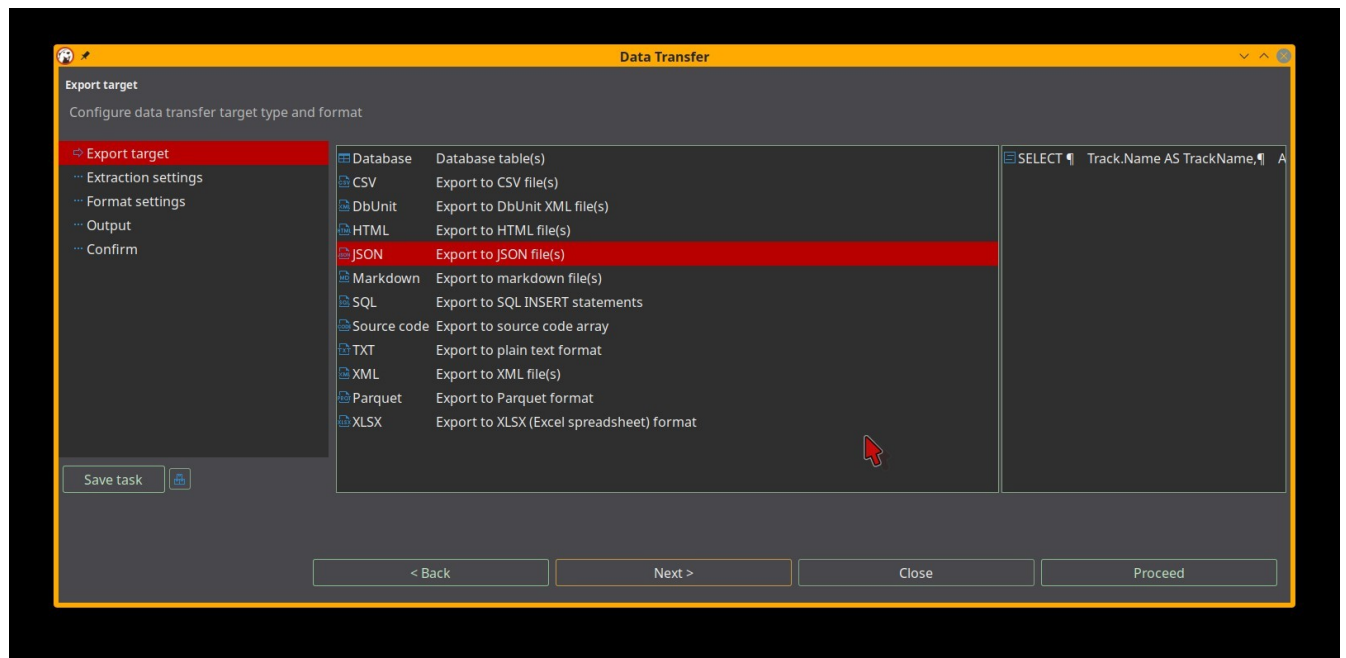
Figure 2: First step in exporting data

Figure 3: There are several dialogs to control how data is written; lots of features.

Using DBeaver *Connection Settings* to Learn How to Connect

You can reach the "Connection settings" dialog in the figure below by right clicking on the database ("DBeaver Sample Database (SQLite)" in red in the upper left) and selecting "Edit connection". For this example at least, the multi-tab dialog provided sufficient information to allow me to connect to the database with an application program. I'll send an example Jupyter notebook where I connected to this SQLite database with Python.
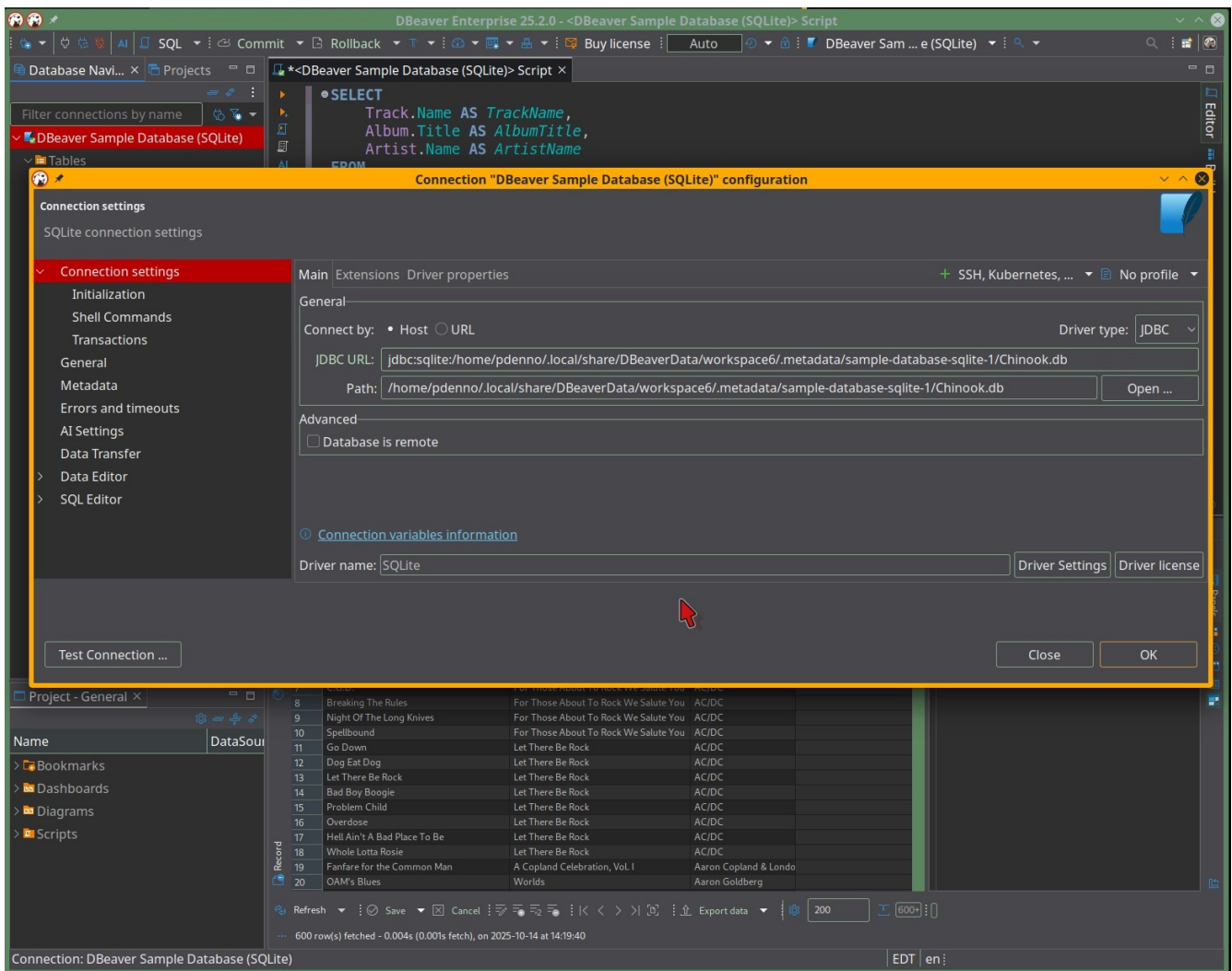
Figure 4: Information needed by applications that intend to query this database.

Below are screenshots from the Jupyter notebook I mentioned above.

Sqlite DB Demo using DBeaver Connection Setting Info

This is a notebook to demonstrate that the information in DBeaver about the example music album DB is sufficient to make a connection and run a query. DBeaver provides extensive dialogs providing information about DBs. This one is sqlite3. Others are likely to be similar, but not identical.

```
import sqlite3
import pandas as pd
```

```
# Identify the DB file; It is the 'Path' under the DBeaver "Connection settings"
database_file = "/home/pdenno/.local/share/DBeaverData/workspace6/.metadata/sample-database-sqlite-1/Chinook.db"
```

Figure 5: Connection information (database_file)

In [8]:
```python
try:
    # Connect to the SQLite database.
    connection = sqlite3.connect(database_file)
    print(f"Successfully connected to {database_file}")

    # Create a cursor object to execute SQL commands
    cursor = connection.cursor()

    # Query the data
    cursor.execute('''
    SELECT
    Track.Name AS TrackName,
    Album.Title AS AlbumTitle,
    Artist.Name AS ArtistName,
    Track.Milliseconds AS TrackMillis
    FROM
        Track
    JOIN
        Album ON Track.AlbumId = Album.AlbumId
    JOIN
        Artist ON Album.ArtistId = Artist.ArtistId
    ORDER BY
        Artist.Name;''')
    rows = cursor.fetchall()
    print(f"Retrieved {len(rows)} tuples")
    for row in rows:
        print(row)

except sqlite3.Error as e:
    print(f"An error occurred: {e}")

finally:
    # Close the connection
    if connection:
        connection.close()
        print(f"Connection to {database_file} closed.")
```

```
Successfully connected to /home/pdenno/.local/share/DBeaverData/workspace6/.metadata/sample-database-sqlite-1/Ch
inook.db
Retrieved 3503 tuples
('For Those About To Rock (We Salute You)', 'For Those About To Rock We Salute You', 'AC/DC', 343719)
('Put The Finger On You', 'For Those About To Rock We Salute You', 'AC/DC', 205662)
```

Figure 6: A query against the DBeaver demo notebook.