Generating SQL for SQLite using Ollama, ChromaDB

This notebook runs through the process of using the `vanna` Python package to generate SQL using AI (RAG + LLMs) including connecting to a database and training. If you're not ready to train on your own database, you can still try it using a sample SQLite database.

Which LLM do you want to use?

- OpenAI via Vanna.AI (Recommended)

  Use Vanna.AI for free to generate your queries
- OpenAI

  Use OpenAI with your own API key
- Azure OpenAI

  If you have OpenAI models deployed on Azure
- [Selected] Ollama

  Use Ollama locally for free. Requires additional setup.
- Mistral via Mistral API

  If you have a Mistral API key
- Other LLM

  If you have a different LLM model

Where do you want to store the 'training' data?

- Vanna Hosted Vector DB (Recommended)

  Use Vanna.AIs hosted vector database (pgvector) for free. This is usable across machines with no additional setup.
- [Selected] ChromaDB

  Use ChromaDBs open-source vector database for free locally. No additional setup is necessary -- all database files will be created and stored locally.
- Marqo

  Use Marqo locally for free. Requires additional setup. Or use their hosted option.
- Other VectorDB

Use any other vector database. Requires additional setup.

Setup

!pip install 'vanna[chromadb,anthropic]'

```
In [1]: model_name = 'claude-3-5-sonnet-20240620'
        file_db = "~/Downloads/chinook.sqlite"
```

```
In [2]: from api_key_store import ApiKeyStore
        s = ApiKeyStore()

        anthropic_api_key = s.get_api_key(provider="ANTHROPIC")
```

anthropic_api_key

```
In [3]: from vanna.anthropic import Anthropic_Chat
        from vanna.chromadb.chromadb_vector import ChromaDB_VectorStore
```

```
In [4]: class MyVanna(ChromaDB_VectorStore, Anthropic_Chat):
            def __init__(self, config=None):
                ChromaDB_VectorStore.__init__(self, config=config)
                Anthropic_Chat.__init__(self, config=config)

        config = {
            'api_key': anthropic_api_key,
            'model': model_name
        }
        vn = MyVanna(config=config)
```

Which database do you want to query?

- Postgres
- Microsoft SQL Server
- DuckDB
- Snowflake
- BigQuery
- [Selected] SQLite

- Other Database

  Use Vanna to generate queries for any SQL database

In [5]:
```
import os
import re
from time import time
```

In [6]:
```
# file_db = "./db/gpt3sql.sqlite"

file_db = os.path.abspath(os.path.expanduser(file_db))
vn.connect_to_sqlite(file_db)
```

In [7]:
```
vn.run_sql_is_set
```

Out[7]:
```
True
```

In [8]:
```
clean_and_train = True  # False
```

In [9]:
```
hostname = os.uname().nodename
print("Hostname:", hostname)
```

```
Hostname: papa-game
```

In [10]:
```
def remove_collections(collection_name=None, ACCEPTED_TYPES = ["sql", "ddl", "documentation"]):
    if not collection_name:
        collections = ACCEPTED_TYPES
    elif isinstance(collection_name, str):
        collections = [collection_name]
    elif isinstance(collection_name, list):
        collections = collection_name
    else:
        print(f"\t{collection_name} is unknown: Skipped")
        return

    for c in collections:
        if not c in ACCEPTED_TYPES:
            print(f"\t{c} is unknown: Skipped")
            continue
        # print(f"vn.remove_collection('{c}')")
        vn.remove_collection(c)
```

In [11]:
```
def strip_brackets(ddl):
    """
    This function removes square brackets from table and column names in a DDL script.

    Args:
        ddl (str): The DDL script containing square brackets.

    Returns:
        str: The DDL script with square brackets removed.
    """
    # Use regular expressions to match and replace square brackets
    pattern = r"\[([^\]]+)]"  # Match any character except ] within square brackets
    return re.sub(pattern, r"\1", ddl)
```

In [12]:
```
if clean_and_train:
    remove_collections()
```

Training

You only need to train once. Do not train again unless you want to add more training data.

In [13]:
```
# show training data
training_data = vn.get_training_data()
training_data
```

Out[13]:

id	question	content	training_data_type

In [14]:
```
df_ddl = vn.run_sql("SELECT type, sql FROM sqlite_master WHERE sql is not null")
```

In [15]:
```
df_ddl
```

Out[15]:

	type	sql
0	table	CREATE TABLE [Album]\n(\n [AlbumId] INTEGER...
1	table	CREATE TABLE [Artist]\n(\n [ArtistId] INTEG...
2	table	CREATE TABLE [Customer]\n(\n [CustomerId] I...
3	table	CREATE TABLE [Employee]\n(\n [EmployeeId] I...
4	table	CREATE TABLE [Genre]\n(\n [GenreId] INTEGER...
5	table	CREATE TABLE [Invoice]\n(\n [InvoiceId] INT...
6	table	CREATE TABLE [InvoiceLine]\n(\n [InvoiceLin...
7	table	CREATE TABLE [MediaType]\n(\n [MediaTypeId]...
8	table	CREATE TABLE [Playlist]\n(\n [PlaylistId] I...
9	table	CREATE TABLE [PlaylistTrack]\n(\n [Playlist...
10	table	CREATE TABLE [Track]\n(\n [TrackId] INTEGER...
11	index	CREATE INDEX [IFK_AlbumArtistId] ON [Album] ([...
12	index	CREATE INDEX [IFK_CustomerSupportRepId] ON [Cu...
13	index	CREATE INDEX [IFK_EmployeeReportsTo] ON [Emplo...
14	index	CREATE INDEX [IFK_InvoiceCustomerId] ON [Invoi...
15	index	CREATE INDEX [IFK_InvoiceLineInvoiceId] ON [In...
16	index	CREATE INDEX [IFK_InvoiceLineTrackId] ON [Invo...
17	index	CREATE INDEX [IFK_PlaylistTrackTrackId] ON [Pl...
18	index	CREATE INDEX [IFK_TrackAlbumId] ON [Track] ([A...
19	index	CREATE INDEX [IFK_TrackGenreId] ON [Track] ([G...
20	index	CREATE INDEX [IFK_TrackMediaTypeId] ON [Track]...

In [16]:

```python
if clean_and_train:
    for ddl in df_ddl['sql'].to_list():
        ddl = strip_brackets(ddl)
        vn.train(ddl=ddl)
```

```
# Sometimes you may want to add documentation about your business terminology or definitions.
vn.train(documentation="In the chinook database invoice means order")
```

```
Adding ddl: 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
)
Adding ddl: CREATE TABLE Artist
(
    ArtistId INTEGER  NOT NULL,
    Name NVARCHAR(120),
    CONSTRAINT PK_Artist PRIMARY KEY  (ArtistId)
)
Adding ddl: CREATE TABLE Customer
(
    CustomerId INTEGER  NOT NULL,
    FirstName NVARCHAR(40)  NOT NULL,
    LastName NVARCHAR(20)  NOT NULL,
    Company NVARCHAR(80),
    Address NVARCHAR(70),
    City NVARCHAR(40),
    State NVARCHAR(40),
    Country NVARCHAR(40),
    PostalCode NVARCHAR(10),
    Phone NVARCHAR(24),
    Fax NVARCHAR(24),
    Email NVARCHAR(60)  NOT NULL,
    SupportRepId INTEGER,
    CONSTRAINT PK_Customer PRIMARY KEY  (CustomerId),
    FOREIGN KEY (SupportRepId) REFERENCES Employee (EmployeeId)
                ON DELETE NO ACTION ON UPDATE NO ACTION
)
Adding ddl: CREATE TABLE Employee
(
    EmployeeId INTEGER  NOT NULL,
    LastName NVARCHAR(20)  NOT NULL,
    FirstName NVARCHAR(20)  NOT NULL,
    Title NVARCHAR(30),
    ReportsTo INTEGER,
    BirthDate DATETIME,
    HireDate DATETIME,
    Address NVARCHAR(70),
    City NVARCHAR(40),
    State NVARCHAR(40),
    Country NVARCHAR(40),
    PostalCode NVARCHAR(10),
    Phone NVARCHAR(24),
    Fax NVARCHAR(24),
    Email NVARCHAR(60),
    CONSTRAINT PK_Employee PRIMARY KEY  (EmployeeId),
    FOREIGN KEY (ReportsTo) REFERENCES Employee (EmployeeId)
                ON DELETE NO ACTION ON UPDATE NO ACTION
)
Adding ddl: CREATE TABLE Genre
(
    GenreId INTEGER  NOT NULL,
    Name NVARCHAR(120),
    CONSTRAINT PK_Genre PRIMARY KEY  (GenreId)
)
Adding ddl: CREATE TABLE Invoice
(
    InvoiceId INTEGER  NOT NULL,
    CustomerId INTEGER  NOT NULL,
    InvoiceDate DATETIME  NOT NULL,
    BillingAddress NVARCHAR(70),
    BillingCity NVARCHAR(40),
    BillingState NVARCHAR(40),
    BillingCountry NVARCHAR(40),
    BillingPostalCode NVARCHAR(10),
    Total NUMERIC(10,2)  NOT NULL,
    CONSTRAINT PK_Invoice PRIMARY KEY  (InvoiceId),
    FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId)
                ON DELETE NO ACTION ON UPDATE NO ACTION
)
Adding ddl: CREATE TABLE InvoiceLine
(
    InvoiceLineId INTEGER  NOT NULL,
    InvoiceId INTEGER  NOT NULL,
    TrackId INTEGER  NOT NULL,
    UnitPrice NUMERIC(10,2)  NOT NULL,
    Quantity INTEGER  NOT NULL,
    CONSTRAINT PK_InvoiceLine PRIMARY KEY  (InvoiceLineId),
    FOREIGN KEY (InvoiceId) REFERENCES Invoice (InvoiceId)
                ON DELETE NO ACTION ON UPDATE NO ACTION,
    FOREIGN KEY (TrackId) REFERENCES Track (TrackId)
                ON DELETE NO ACTION ON UPDATE NO ACTION
)
Adding ddl: CREATE TABLE MediaType
(
    MediaTypeId INTEGER  NOT NULL,
    Name NVARCHAR(120),
    CONSTRAINT PK_MediaType PRIMARY KEY  (MediaTypeId)
)
Adding ddl: CREATE TABLE Playlist
(
    PlaylistId INTEGER  NOT NULL,
    Name NVARCHAR(120),
    CONSTRAINT PK_Playlist PRIMARY KEY  (PlaylistId)
)
Adding ddl: CREATE TABLE PlaylistTrack
(
    PlaylistId INTEGER  NOT NULL,
    TrackId INTEGER  NOT NULL,
    CONSTRAINT PK_PlaylistTrack PRIMARY KEY  (PlaylistId, TrackId),
    FOREIGN KEY (PlaylistId) REFERENCES Playlist (PlaylistId)
                ON DELETE NO ACTION ON UPDATE NO ACTION,
    FOREIGN KEY (TrackId) REFERENCES Track (TrackId)
                ON DELETE NO ACTION ON UPDATE NO ACTION
)
Adding ddl: CREATE TABLE Track
(
    TrackId INTEGER  NOT NULL,
    Name NVARCHAR(200)  NOT NULL,
    AlbumId INTEGER,
    MediaTypeId INTEGER  NOT NULL,
    GenreId INTEGER,
    Composer NVARCHAR(220),
    Milliseconds INTEGER  NOT NULL,
    Bytes INTEGER,
    UnitPrice NUMERIC(10,2)  NOT NULL,
    CONSTRAINT PK_Track PRIMARY KEY  (TrackId),
    FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId)
                ON DELETE NO ACTION ON UPDATE NO ACTION,
    FOREIGN KEY (GenreId) REFERENCES Genre (GenreId)
```

```
                ON DELETE NO ACTION ON UPDATE NO ACTION,
    FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId)
                ON DELETE NO ACTION ON UPDATE NO ACTION
)
Adding ddl: CREATE INDEX IFK_AlbumArtistId ON Album (ArtistId)
Adding ddl: CREATE INDEX IFK_CustomerSupportRepId ON Customer (SupportRepId)
Adding ddl: CREATE INDEX IFK_EmployeeReportsTo ON Employee (ReportsTo)
Adding ddl: CREATE INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)
Adding ddl: CREATE INDEX IFK_InvoiceLineInvoiceId ON InvoiceLine (InvoiceId)
Adding ddl: CREATE INDEX IFK_InvoiceLineTrackId ON InvoiceLine (TrackId)
Adding ddl: CREATE INDEX IFK_PlaylistTrackTrackId ON PlaylistTrack (TrackId)
Adding ddl: CREATE INDEX IFK_TrackAlbumId ON Track (AlbumId)
Adding ddl: CREATE INDEX IFK_TrackGenreId ON Track (GenreId)
Adding ddl: CREATE INDEX IFK_TrackMediaTypeId ON Track (MediaTypeId)
Adding documentation....
```

In [ ]:

Asking the AI

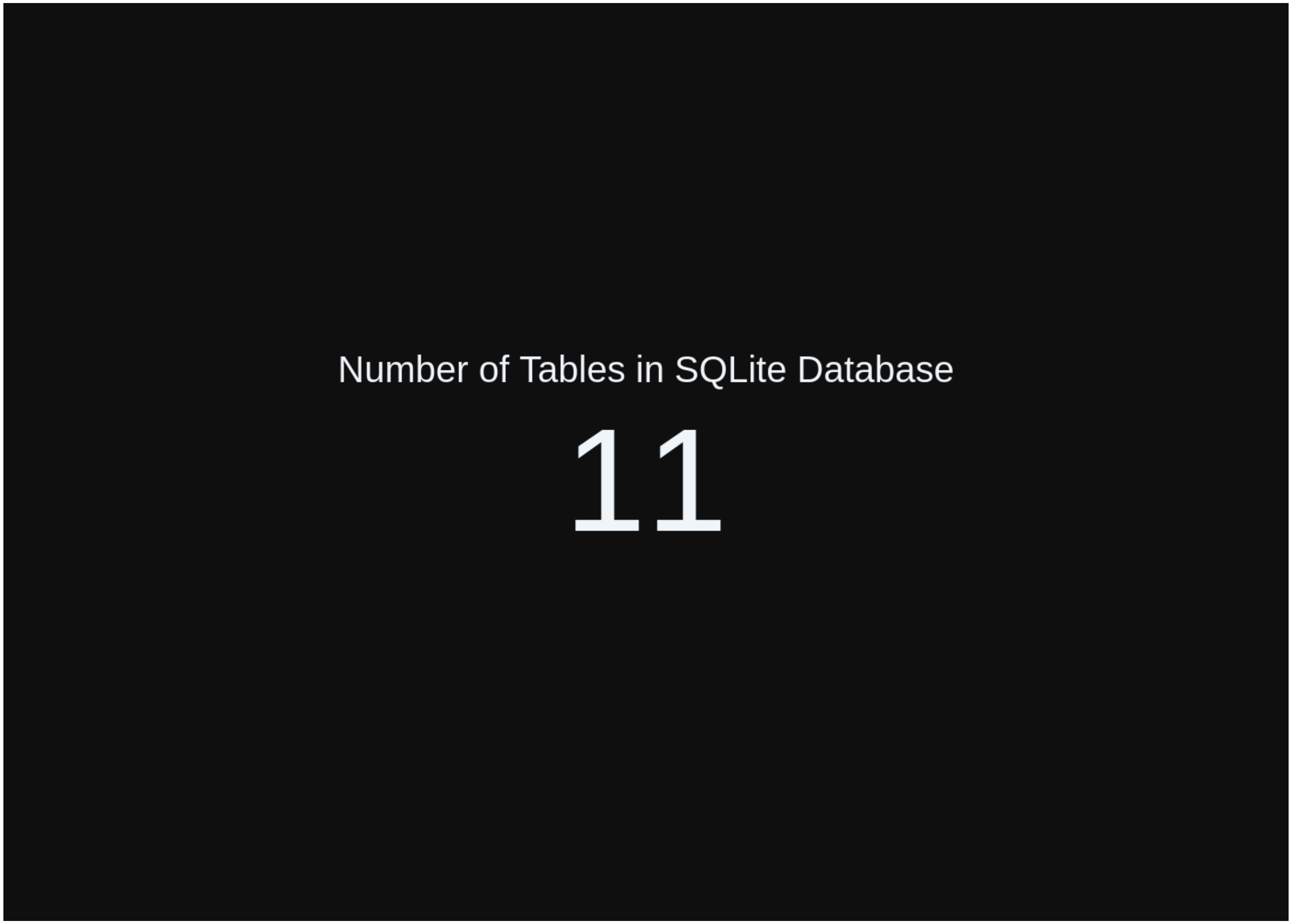
Whenever you ask a new question, it will find the 10 most relevant pieces of training data and use it as part of the LLM prompt to generate the SQL.

In [17]:
```
ts_start = time()
```

In [18]:
```
vn.ask(question="Show me a list of tables in the SQLite database")
```

```
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
[{'role': 'system', 'content': "You are a SQLite expert. Please help to generate a SQL query to answer the
question. Your response should ONLY be based on the given context and follow the response guidelines and fo
rmat instructions. \n===Tables \nCREATE TABLE Playlist\n(\n    PlaylistId INTEGER  NOT NULL,\n    Name NVAR
CHAR(120),\n    CONSTRAINT PK_Playlist PRIMARY KEY  (PlaylistId)\n)\n\nCREATE TABLE InvoiceLine\n(\n    Inv
oiceLineId INTEGER  NOT NULL,\n    InvoiceId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    UnitPr
ice NUMERIC(10,2)  NOT NULL,\n    Quantity INTEGER  NOT NULL,\n    CONSTRAINT PK_InvoiceLine PRIMARY KEY
(InvoiceLineId),\n    FOREIGN KEY (InvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON U
PDATE NO ACTION,\n    FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE
NO ACTION\n)\n\nCREATE TABLE PlaylistTrack\n(\n    PlaylistId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT
NULL,\n    CONSTRAINT PK_PlaylistTrack PRIMARY KEY  (PlaylistId, TrackId),\n    FOREIGN KEY (PlaylistId) RE
FERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (TrackId) RE
FERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Track\n(\n    Tra
ckId INTEGER  NOT NULL,\n    Name NVARCHAR(200)  NOT NULL,\n    AlbumId INTEGER,\n    MediaTypeId INTEGER
NOT NULL,\n    GenreId INTEGER,\n    Composer NVARCHAR(220),\n    Milliseconds INTEGER  NOT NULL,\n    Byte
s INTEGER,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (TrackId),\n    FO
REIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN
KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (M
ediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE T
ABLE MediaType\n(\n    MediaTypeId INTEGER  NOT NULL,\n    Name NVARCHAR(120),\n    CONSTRAINT PK_MediaType
PRIMARY KEY  (MediaTypeId)\n)\n\nCREATE TABLE Artist\n(\n    ArtistId INTEGER  NOT NULL,\n    Name NVARCHAR
(120),\n    CONSTRAINT PK_Artist PRIMARY KEY  (ArtistId)\n)\n\nCREATE TABLE Album\n(\n    AlbumId INTEGER
NOT NULL,\n    Title NVARCHAR(160)  NOT NULL,\n    ArtistId INTEGER  NOT NULL,\n    CONSTRAINT PK_Album PRI
MARY KEY  (AlbumId),\n    FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON
UPDATE NO ACTION\n)\n\nCREATE TABLE Genre\n(\n    GenreId INTEGER  NOT NULL,\n    Name NVARCHAR(120),\n
CONSTRAINT PK_Genre PRIMARY KEY  (GenreId)\n)\n\nCREATE TABLE Invoice\n(\n    InvoiceId INTEGER  NOT NUL
L,\n    CustomerId INTEGER  NOT NULL,\n    InvoiceDate DATETIME  NOT NULL,\n    BillingAddress NVARCHAR(7
0),\n    BillingCity NVARCHAR(40),\n    BillingState NVARCHAR(40),\n    BillingCountry NVARCHAR(40),\n    B
illingPostalCode NVARCHAR(10),\n    Total NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Invoice PRIMARY KEY
(InvoiceId),\n    FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UP
DATE NO ACTION\n)\n\nCREATE INDEX IFK_EmployeeReportsTo ON Employee (ReportsTo)\n\n\n===Additional Context
\n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided context is s
ufficient, please generate a valid SQL query without any explanations for the question. \n2. If the provide
d context is almost sufficient but requires knowledge of a specific string in a particular column, please g
enerate an intermediate SQL query to find the distinct strings in that column. Prepend the query with a com
ment saying intermediate_sql \n3. If the provided context is insufficient, please explain why it can't be g
enerated. \n4. Please use the most relevant table(s). \n5. If the question has been asked and answered befo
re, please repeat the answer exactly as it was given before. \n"}, {'role': 'user', 'content': 'Show me a l
ist of tables in the SQLite database'}]
Using model claude-3-5-sonnet-20240620 for 948.25 tokens (approx)
Here's a SQL query to show a list of tables in the SQLite database:

SELECT name FROM sqlite_master WHERE type='table';
SELECT name FROM sqlite_master WHERE type='table';
SELECT name FROM sqlite_master WHERE type='table';
             name
0           Album
1          Artist
2        Customer
3        Employee
4           Genre
5         Invoice
6     InvoiceLine
7       MediaType
8        Playlist
9   PlaylistTrack
10          Track
Using model claude-3-5-sonnet-20240620 for 168.0 tokens (approx)
```

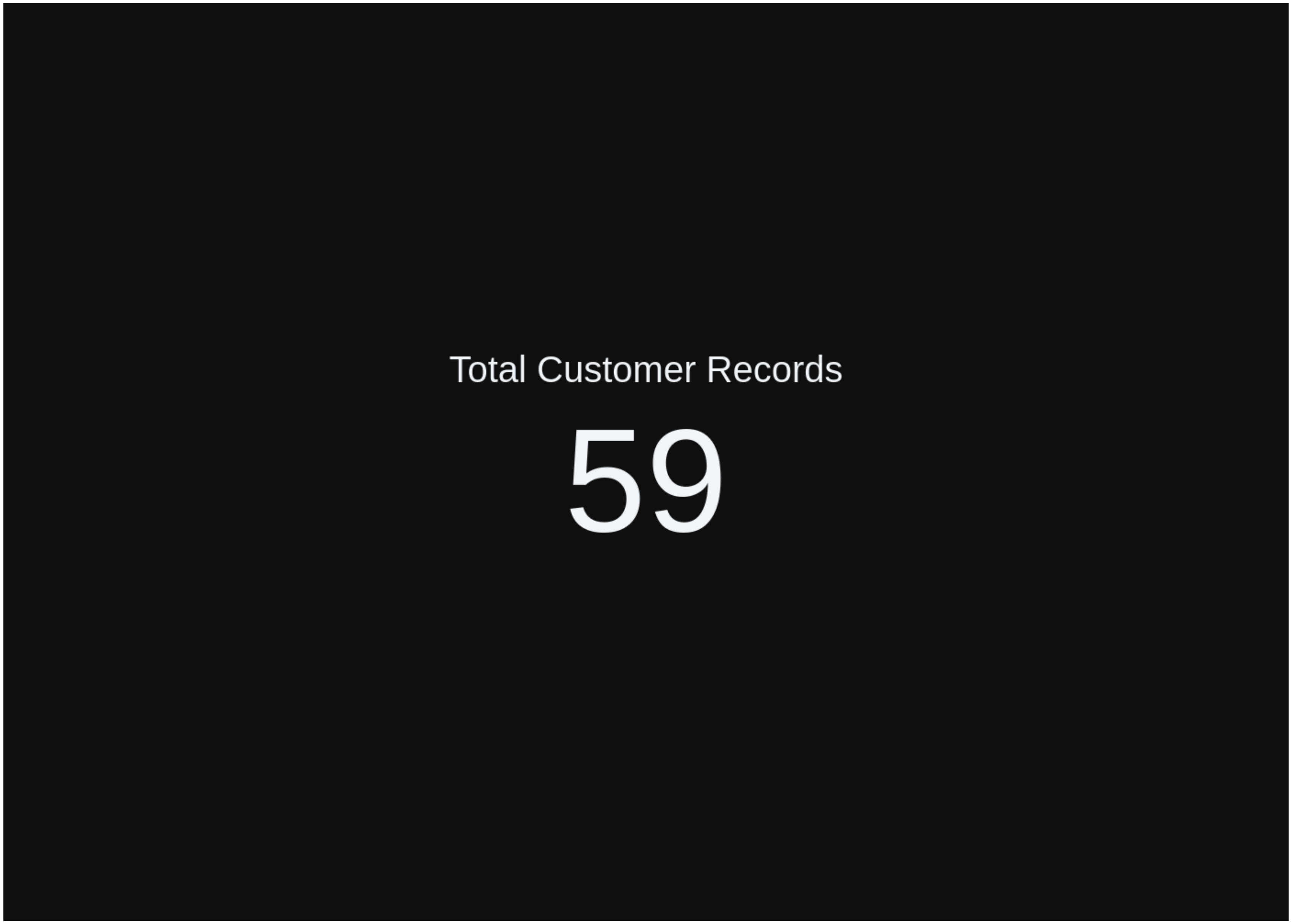

Number of Tables in SQLite Database
11

Out[18]:
```
("SELECT name FROM sqlite_master WHERE type='table';",
              name
 0           Album
 1          Artist
 2        Customer
 3        Employee
 4           Genre
 5         Invoice
 6     InvoiceLine
 7       MediaType
 8        Playlist
 9   PlaylistTrack
 10          Track,
 Figure({
     'data': [{'mode': 'number',
               'title': {'text': 'Number of Tables in SQLite Database'},
               'type': 'indicator',
               'value': 11}],
     'layout': {'template': '...'}
 }))
```

In [19]:
```
vn.ask(question="How many records are in table called customer")
```

```
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
[{'role': 'system', 'content': "You are a SQLite expert. Please help to generate a SQL query to answer the
question. Your response should ONLY be based on the given context and follow the response guidelines and fo
rmat instructions. \n===Tables \nCREATE TABLE Customer\n(\n    CustomerId INTEGER  NOT NULL,\n    FirstName
NVARCHAR(40)  NOT NULL,\n    LastName NVARCHAR(20)  NOT NULL,\n    Company NVARCHAR(80),\n    Address NVARC
HAR(70),\n    City NVARCHAR(40),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVARCH
AR(10),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60)  NOT NULL,\n    SupportRepI
d INTEGER,\n    CONSTRAINT PK_Customer PRIMARY KEY  (CustomerId),\n    FOREIGN KEY (SupportRepId) REFERENCE
S Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Invoice\n(\n    In
voiceId INTEGER  NOT NULL,\n    CustomerId INTEGER  NOT NULL,\n    InvoiceDate DATETIME  NOT NULL,\n    Bil
lingAddress NVARCHAR(70),\n    BillingCity NVARCHAR(40),\n    BillingState NVARCHAR(40),\n    BillingCountr
y NVARCHAR(40),\n    BillingPostalCode NVARCHAR(10),\n    Total NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK
_Invoice PRIMARY KEY  (InvoiceId),\n    FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON
DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE InvoiceLine\n(\n    InvoiceLineId INTEGER  NOT NUL
L,\n    InvoiceId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    UnitPrice NUMERIC(10,2)  NOT NUL
L,\n    Quantity INTEGER  NOT NULL,\n    CONSTRAINT PK_InvoiceLine PRIMARY KEY  (InvoiceLineId),\n    FOREI
GN KEY (InvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREI
GN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABL
E Album\n(\n    AlbumId INTEGER  NOT NULL,\n    Title NVARCHAR(160)  NOT NULL,\n    ArtistId INTEGER  NOT N
ULL,\n    CONSTRAINT PK_Album PRIMARY KEY  (AlbumId),\n    FOREIGN KEY (ArtistId) REFERENCES Artist (Artist
Id) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceCustomerId ON Invoice (Cust
omerId)\n\nCREATE TABLE Employee\n(\n    EmployeeId INTEGER  NOT NULL,\n    LastName NVARCHAR(20)  NOT NUL
L,\n    FirstName NVARCHAR(20)  NOT NULL,\n    Title NVARCHAR(30),\n    ReportsTo INTEGER,\n    BirthDate D
ATETIME,\n    HireDate DATETIME,\n    Address NVARCHAR(70),\n    City NVARCHAR(40),\n    State NVARCHAR(4
0),\n    Country NVARCHAR(40),\n    PostalCode NVARCHAR(10),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(2
4),\n    Email NVARCHAR(60),\n    CONSTRAINT PK_Employee PRIMARY KEY  (EmployeeId),\n    FOREIGN KEY (Repor
tsTo) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Tra
ck\n(\n    TrackId INTEGER  NOT NULL,\n    Name NVARCHAR(200)  NOT NULL,\n    AlbumId INTEGER,\n    MediaTy
peId INTEGER  NOT NULL,\n    GenreId INTEGER,\n    Composer NVARCHAR(220),\n    Milliseconds INTEGER  NOT N
ULL,\n    Bytes INTEGER,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (Tra
ckId),\n    FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTIO
N,\n    FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n
FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION
\n)\n\nCREATE INDEX IFK_CustomerSupportRepId ON Customer (SupportRepId)\n\nCREATE TABLE Playlist\n(\n    Pl
aylistId INTEGER  NOT NULL,\n    Name NVARCHAR(120),\n    CONSTRAINT PK_Playlist PRIMARY KEY  (PlaylistId)
\n)\n\nCREATE TABLE PlaylistTrack\n(\n    PlaylistId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n
CONSTRAINT PK_PlaylistTrack PRIMARY KEY  (PlaylistId, TrackId),\n    FOREIGN KEY (PlaylistId) REFERENCES Pl
aylist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (TrackId) REFERENCES Tr
ack (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n===Additional Context \n\nIn the chinoo
k database invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficient, please
generate a valid SQL query without any explanations for the question. \n2. If the provided context is almos
t sufficient but requires knowledge of a specific string in a particular column, please generate an interme
diate SQL query to find the distinct strings in that column. Prepend the query with a comment saying interm
ediate_sql \n3. If the provided context is insufficient, please explain why it can't be generated. \n4. Ple
ase use the most relevant table(s). \n5. If the question has been asked and answered before, please repeat
the answer exactly as it was given before. \n"}, {'role': 'user', 'content': 'Show me a list of tables in t
he SQLite database'}, {'role': 'assistant', 'content': "SELECT name FROM sqlite_master WHERE type='tabl
e';"}, {'role': 'user', 'content': 'How many records are in table called customer'}]
Using model claude-3-5-sonnet-20240620 for 1174.75 tokens (approx)
SELECT COUNT(*) FROM Customer;
SELECT COUNT(*) FROM Customer;
SELECT COUNT(*) FROM Customer;
   COUNT(*)
0        59
Using model claude-3-5-sonnet-20240620 for 163.25 tokens (approx)
```



Total Customer Records

59

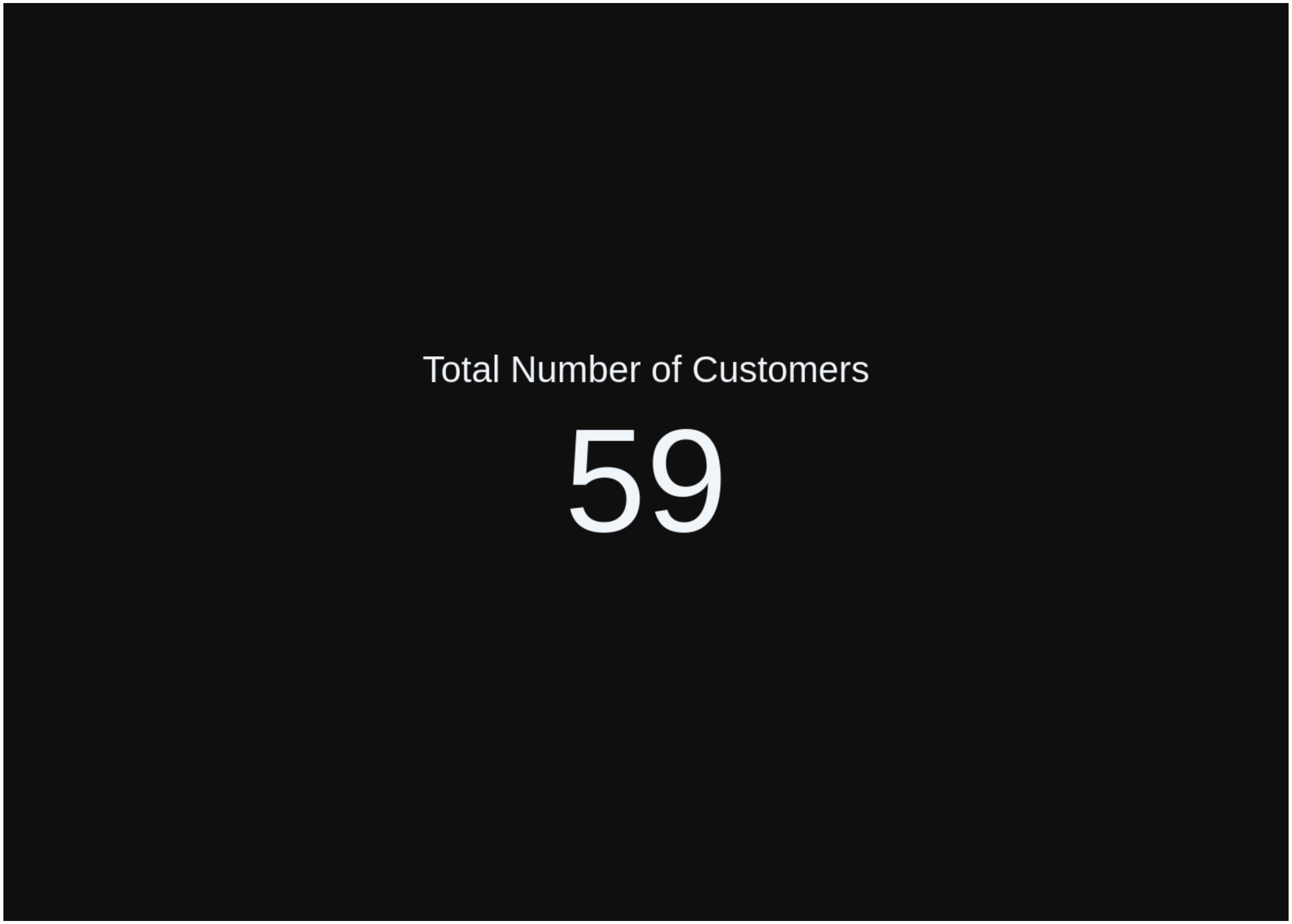
```
Out[19]: ('SELECT COUNT(*) FROM Customer;',
             COUNT(*)
          0        59,
          Figure({
              'data': [{'mode': 'number', 'title': {'text': 'Total Customer Records'}, 'type': 'indicator', 'valu
         e': 59}],
              'layout': {'template': '...'}
          }))
```

```
In [20]: vn.ask(question="How many customers are there")
```

```
Number of requested results 10 is greater than number of elements in index 2, updating n_results = 2
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

[{'role': 'system', 'content': "You are a SQLite expert. Please help to generate a SQL query to answer the question. Your response should ONLY be based on the given context and follow the response guidelines and fo rmat instructions. \n===Tables \nCREATE TABLE Customer\n(\n   CustomerId INTEGER NOT NULL,\n   FirstName NVARCHAR(40) NOT NULL,\n   LastName NVARCHAR(20) NOT NULL,\n   Company NVARCHAR(80),\n   Address NVARC HAR(70),\n   City NVARCHAR(40),\n   State NVARCHAR(40),\n   Country NVARCHAR(40),\n   PostalCode NVARCH AR(10),\n   Phone NVARCHAR(24),\n   Fax NVARCHAR(24),\n   Email NVARCHAR(60) NOT NULL,\n   SupportRepI d INTEGER,\n   CONSTRAINT PK_Customer PRIMARY KEY (CustomerId),\n   FOREIGN KEY (SupportRepId) REFERENCE S Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_CustomerSuppor tRepId ON Customer (SupportRepId)\n\nCREATE TABLE Invoice\n(\n   InvoiceId INTEGER NOT NULL,\n   Custome rId INTEGER NOT NULL,\n   InvoiceDate DATETIME NOT NULL,\n   BillingAddress NVARCHAR(70),\n   BillingC ity NVARCHAR(40),\n   BillingState NVARCHAR(40),\n   BillingCountry NVARCHAR(40),\n   BillingPostalCode NVARCHAR(10),\n   Total NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Invoice PRIMARY KEY (InvoiceId),\n FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n) \n\nCREATE INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)\n\nCREATE TABLE InvoiceLine\n(\n   InvoiceL ineId INTEGER NOT NULL,\n   InvoiceId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   UnitPrice N UMERIC(10,2) NOT NULL,\n   Quantity INTEGER NOT NULL,\n   CONSTRAINT PK_InvoiceLine PRIMARY KEY (Invoi ceLineId),\n   FOREIGN KEY (InvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACT ION\n)\n\nCREATE INDEX IFK_InvoiceLineInvoiceId ON InvoiceLine (InvoiceId)\n\nCREATE TABLE Track\n(\n   Tr ackId INTEGER NOT NULL,\n   Name NVARCHAR(200) NOT NULL,\n   AlbumId INTEGER,\n   MediaTypeId INTEGER NOT NULL,\n   GenreId INTEGER,\n   Composer NVARCHAR(220),\n   Milliseconds INTEGER NOT NULL,\n   Byte s INTEGER,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Track PRIMARY KEY (TrackId),\n   FO REIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (M ediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE I NDEX IFK_InvoiceLineTrackId ON InvoiceLine (TrackId)\n\nCREATE TABLE Album\n(\n   AlbumId INTEGER NOT NUL L,\n   Title NVARCHAR(160) NOT NULL,\n   ArtistId INTEGER NOT NULL,\n   CONSTRAINT PK_Album PRIMARY KE Y (AlbumId),\n   FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Employee\n(\n   EmployeeId INTEGER NOT NULL,\n   LastName NVARCHAR(20) NOT NULL,\n   FirstName NVARCHAR(20) NOT NULL,\n   Title NVARCHAR(30),\n   ReportsTo INTEGER,\n   BirthDat e DATETIME,\n   HireDate DATETIME,\n   Address NVARCHAR(70),\n   City NVARCHAR(40),\n   State NVARCHAR (40),\n   Country NVARCHAR(40),\n   PostalCode NVARCHAR(10),\n   Phone NVARCHAR(24),\n   Fax NVARCHAR(2 4),\n   Email NVARCHAR(60),\n   CONSTRAINT PK_Employee PRIMARY KEY (EmployeeId),\n   FOREIGN KEY (Repor tsTo) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n===Additional Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided cont ext is sufficient, please generate a valid SQL query without any explanations for the question. \n2. If the provided context is almost sufficient but requires knowledge of a specific string in a particular column, p lease generate an intermediate SQL query to find the distinct strings in that column. Prepend the query wit h a comment saying intermediate_sql \n3. If the provided context is insufficient, please explain why it ca n't be generated. \n4. Please use the most relevant table(s). \n5. If the question has been asked and answe red before, please repeat the answer exactly as it was given before. \n"}, {'role': 'user', 'content': 'How many records are in table called customer'}, {'role': 'assistant', 'content': 'SELECT COUNT(*) FROM Custome

```
r;'}, {'role': 'user', 'content': 'Show me a list of tables in the SQLite database'}, {'role': 'assistant',
'content': "SELECT name FROM sqlite_master WHERE type='table';"}, {'role': 'user', 'content': 'How many cus
tomers are there'}]
Using model claude-3-5-sonnet-20240620 for 1095.0 tokens (approx)
SELECT COUNT(*) FROM Customer;
SELECT COUNT(*) FROM Customer;
SELECT COUNT(*) FROM Customer;
   COUNT(*)
0        59
Using model claude-3-5-sonnet-20240620 for 159.0 tokens (approx)
```



Total Number of Customers

59

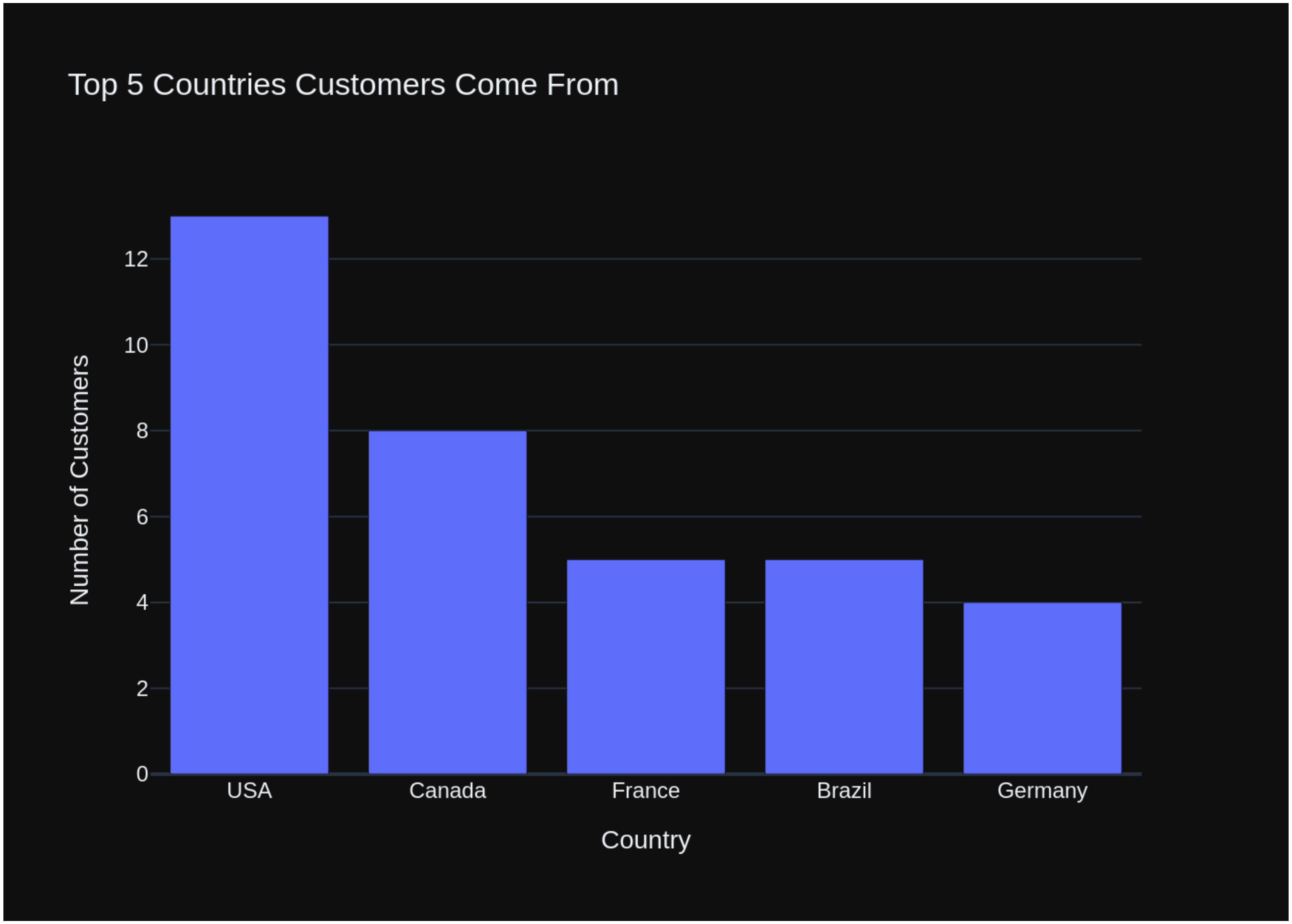
Out[20]:
```
('SELECT COUNT(*) FROM Customer;',
    COUNT(*)
 0        59,
 Figure({
     'data': [{'mode': 'number', 'title': {'text': 'Total Number of Customers'}, 'type': 'indicator', 'val
ue': 59}],
     'layout': {'template': '...'}
 }))
```

In [ ]:

In [21]:
```
vn.ask(question="what are the top 5 countries that customers come from?")
```

```
Number of requested results 10 is greater than number of elements in index 3, updating n_results = 3
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
[{'role': 'system', 'content': "You are a SQLite expert. Please help to generate a SQL query to answer the
question. Your response should ONLY be based on the given context and follow the response guidelines and fo
rmat instructions. \n===Tables \nCREATE TABLE Customer\n(\n    CustomerId INTEGER  NOT NULL,\n    FirstName
NVARCHAR(40)  NOT NULL,\n    LastName NVARCHAR(20)  NOT NULL,\n    Company NVARCHAR(80),\n    Address NVARC
HAR(70),\n    City NVARCHAR(40),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVARCH
AR(10),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60)  NOT NULL,\n    SupportRepI
d INTEGER,\n    CONSTRAINT PK_Customer PRIMARY KEY  (CustomerId),\n    FOREIGN KEY (SupportRepId) REFERENCE
S Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Invoice\n(\n    In
voiceId INTEGER  NOT NULL,\n    CustomerId INTEGER  NOT NULL,\n    InvoiceDate DATETIME  NOT NULL,\n    Bil
lingAddress NVARCHAR(70),\n    BillingCity NVARCHAR(40),\n    BillingState NVARCHAR(40),\n    BillingCountr
y NVARCHAR(40),\n    BillingPostalCode NVARCHAR(10),\n    Total NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK
_Invoice PRIMARY KEY  (InvoiceId),\n    FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON
DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE InvoiceLine\n(\n    InvoiceLineId INTEGER  NOT NUL
L,\n    InvoiceId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    UnitPrice NUMERIC(10,2)  NOT NUL
L,\n    Quantity INTEGER  NOT NULL,\n    CONSTRAINT PK_InvoiceLine PRIMARY KEY  (InvoiceLineId),\n    FOREI
GN KEY (InvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREI
GN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABL
E Employee\n(\n    EmployeeId INTEGER  NOT NULL,\n    LastName NVARCHAR(20)  NOT NULL,\n    FirstName NVARC
HAR(20)  NOT NULL,\n    Title NVARCHAR(30),\n    ReportsTo INTEGER,\n    BirthDate DATETIME,\n    HireDate
DATETIME,\n    Address NVARCHAR(70),\n    City NVARCHAR(40),\n    State NVARCHAR(40),\n    Country NVARCHAR
(40),\n    PostalCode NVARCHAR(10),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(6
0),\n    CONSTRAINT PK_Employee PRIMARY KEY  (EmployeeId),\n    FOREIGN KEY (ReportsTo) REFERENCES Employee
(EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE MediaType\n(\n    MediaTypeId
INTEGER  NOT NULL,\n    Name NVARCHAR(120),\n    CONSTRAINT PK_MediaType PRIMARY KEY  (MediaTypeId)\n)\n\nC
REATE TABLE Playlist\n(\n    PlaylistId INTEGER  NOT NULL,\n    Name NVARCHAR(120),\n    CONSTRAINT PK_Play
list PRIMARY KEY  (PlaylistId)\n)\n\nCREATE INDEX IFK_CustomerSupportRepId ON Customer (SupportRepId)\n\nCR
EATE TABLE PlaylistTrack\n(\n    PlaylistId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    CONSTRA
INT PK_PlaylistTrack PRIMARY KEY  (PlaylistId, TrackId),\n    FOREIGN KEY (PlaylistId) REFERENCES Playlist
(PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (TrackId) REFERENCES Track (Tr
ackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Track\n(\n    TrackId INTEGER  NOT
NULL,\n    Name NVARCHAR(200)  NOT NULL,\n    AlbumId INTEGER,\n    MediaTypeId INTEGER  NOT NULL,\n    Gen
reId INTEGER,\n    Composer NVARCHAR(220),\n    Milliseconds INTEGER  NOT NULL,\n    Bytes INTEGER,\n    Un
itPrice NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (TrackId),\n    FOREIGN KEY (AlbumI
d) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (GenreId) REF
ERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (MediaTypeId) REFER
ENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Album\n(\n
AlbumId INTEGER  NOT NULL,\n    Title NVARCHAR(160)  NOT NULL,\n    ArtistId INTEGER  NOT NULL,\n    CONSTR
AINT PK_Album PRIMARY KEY  (AlbumId),\n    FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\tON DEL
ETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n===Additional Context \n\nIn the chinook database invoice means o
rder\n\n===Response Guidelines \n1. If the provided context is sufficient, please generate a valid SQL quer
y without any explanations for the question. \n2. If the provided context is almost sufficient but requires
knowledge of a specific string in a particular column, please generate an intermediate SQL query to find th
e distinct strings in that column. Prepend the query with a comment saying intermediate_sql \n3. If the pro
vided context is insufficient, please explain why it can't be generated. \n4. Please use the most relevant
table(s). \n5. If the question has been asked and answered before, please repeat the answer exactly as it w
as given before. \n"}, {'role': 'user', 'content': 'How many customers are there'}, {'role': 'assistant',
'content': 'SELECT COUNT(*) FROM Customer;'}, {'role': 'user', 'content': 'How many records are in table ca
lled customer'}, {'role': 'assistant', 'content': 'SELECT COUNT(*) FROM Customer;'}, {'role': 'user', 'cont
ent': 'Show me a list of tables in the SQLite database'}, {'role': 'assistant', 'content': "SELECT name FRO
M sqlite_master WHERE type='table';"}, {'role': 'user', 'content': 'what are the top 5 countries that custo
mers come from?'}]
Using model claude-3-5-sonnet-20240620 for 1230.75 tokens (approx)
SELECT Country, COUNT(*) as CustomerCount
FROM Customer
GROUP BY Country
ORDER BY CustomerCount DESC
LIMIT 5;
SELECT Country, COUNT(*) as CustomerCount
FROM Customer
GROUP BY Country
ORDER BY CustomerCount DESC
LIMIT 5;
SELECT Country, COUNT(*) as CustomerCount
FROM Customer
GROUP BY Country
ORDER BY CustomerCount DESC
LIMIT 5;
   Country  CustomerCount
0      USA             13
1   Canada              8
2   France              5
3   Brazil              5
4  Germany              4
Using model claude-3-5-sonnet-20240620 for 192.75 tokens (approx)
```


Top 5 Countries Customers Come From
Number of Customers
0
2
4
6
8
10
12
USA
Canada
France
Brazil
Germany
Country

Out[21]:
```
('SELECT Country, COUNT(*) as CustomerCount\nFROM Customer\nGROUP BY Country\nORDER BY CustomerCount DESC
\nLIMIT 5;',
    Country  CustomerCount
 0      USA             13
 1   Canada              8
 2   France              5
 3   Brazil              5
 4  Germany              4,
 Figure({
     'data': [{'alignmentgroup': 'True',
               'hovertemplate': 'Country=%{x}<br>CustomerCount=%{y}<extra></extra>',
               'legendgroup': '',
               'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
               'name': '',
               'offsetgroup': '',
               'orientation': 'v',
               'showlegend': False,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['USA', 'Canada', 'France', 'Brazil', 'Germany'], dtype=object),
               'xaxis': 'x',
               'y': array([13,  8,  5,  5,  4]),
               'yaxis': 'y'}],
     'layout': {'barmode': 'relative',
                'legend': {'tracegroupgap': 0},
                'template': '...',
                'title': {'text': 'Top 5 Countries Customers Come From'},
                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'Country'}},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'Number of Customers'}}}
 }))
```

More SQL questions

see `sample-sql-queries-sqlite-chinook.ipynb`

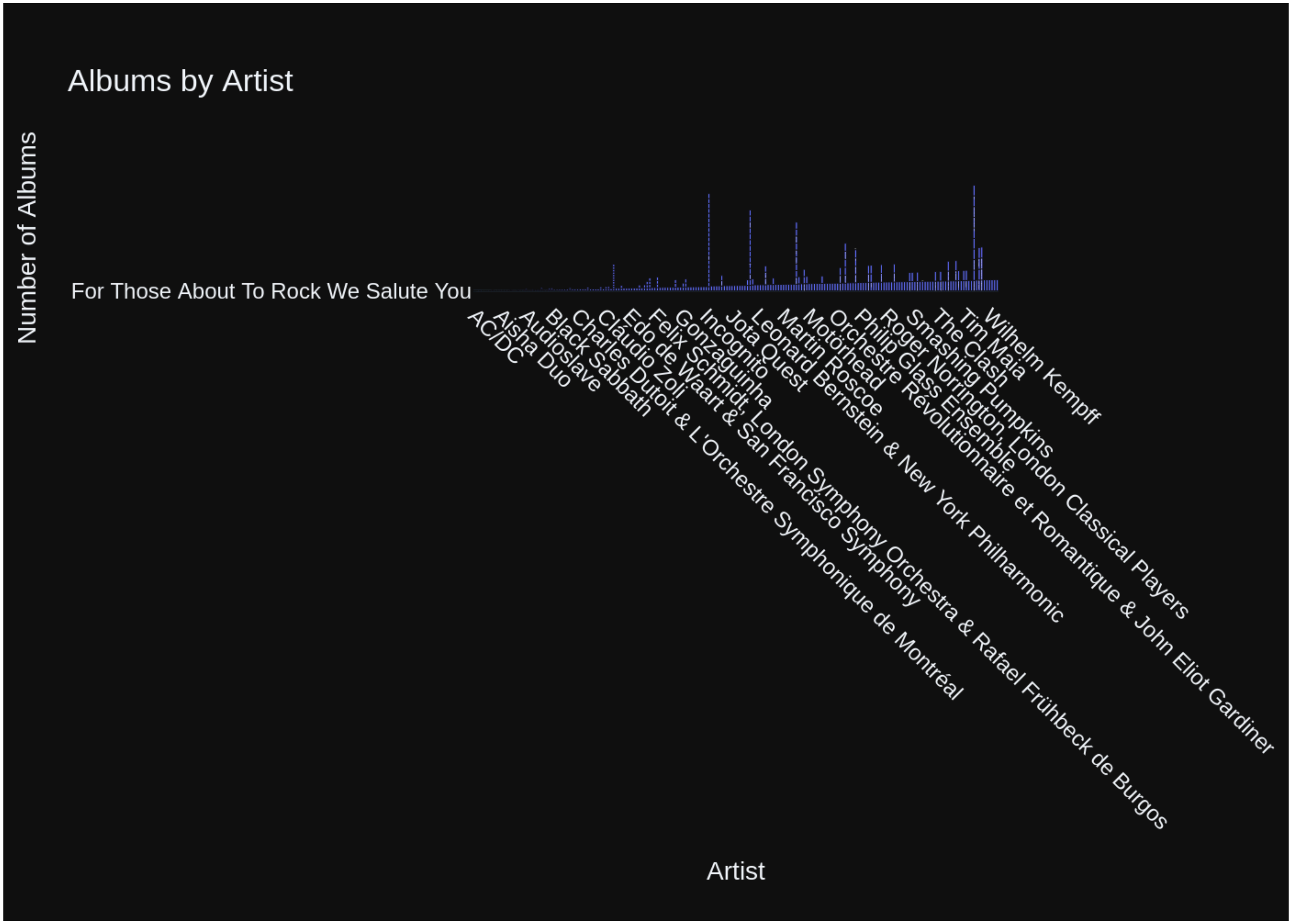
In [22]:
```
question = """
    List all albums and their corresponding artist names
"""

vn.ask(question=question)
```

```
Number of requested results 10 is greater than number of elements in index 4, updating n_results = 4
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
[{'role': 'system', 'content': "You are a SQLite expert. Please help to generate a SQL query to answer the
question. Your response should ONLY be based on the given context and follow the response guidelines and fo
rmat instructions. \n===Tables \nCREATE INDEX IFK_AlbumArtistId ON Album (ArtistId)\n\nCREATE TABLE Album\n
(\n    AlbumId INTEGER  NOT NULL,\n    Title NVARCHAR(160)  NOT NULL,\n    ArtistId INTEGER  NOT NULL,\n
CONSTRAINT PK_Album PRIMARY KEY  (AlbumId),\n    FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\t
ON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Track\n(\n    TrackId INTEGER  NOT NULL,\n    Na
me NVARCHAR(200)  NOT NULL,\n    AlbumId INTEGER,\n    MediaTypeId INTEGER  NOT NULL,\n    GenreId INTEGE
R,\n    Composer NVARCHAR(220),\n    Milliseconds INTEGER  NOT NULL,\n    Bytes INTEGER,\n    UnitPrice NUM
ERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (TrackId),\n    FOREIGN KEY (AlbumId) REFERENCE
S Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (GenreId) REFERENCES Genr
e (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (MediaTypeId) REFERENCES MediaT
ype (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_TrackAlbumId ON Trac
k (AlbumId)\n\nCREATE TABLE Artist\n(\n    ArtistId INTEGER  NOT NULL,\n    Name NVARCHAR(120),\n    CONSTR
AINT PK_Artist PRIMARY KEY  (ArtistId)\n)\n\nCREATE INDEX IFK_TrackGenreId ON Track (GenreId)\n\nCREATE IND
EX IFK_PlaylistTrackTrackId ON PlaylistTrack (TrackId)\n\nCREATE INDEX IFK_TrackMediaTypeId ON Track (Media
TypeId)\n\nCREATE TABLE Playlist\n(\n    PlaylistId INTEGER  NOT NULL,\n    Name NVARCHAR(120),\n    CONSTR
AINT PK_Playlist PRIMARY KEY  (PlaylistId)\n)\n\nCREATE TABLE PlaylistTrack\n(\n    PlaylistId INTEGER  NOT
NULL,\n    TrackId INTEGER  NOT NULL,\n    CONSTRAINT PK_PlaylistTrack PRIMARY KEY  (PlaylistId, TrackI
d),\n    FOREIGN KEY (PlaylistId) REFERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO A
CTION,\n    FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION
\n)\n\n\n===Additional Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n
1. If the provided context is sufficient, please generate a valid SQL query without any explanations for th
e question. \n2. If the provided context is almost sufficient but requires knowledge of a specific string i
n a particular column, please generate an intermediate SQL query to find the distinct strings in that colum
n. Prepend the query with a comment saying intermediate_sql \n3. If the provided context is insufficient, p
lease explain why it can't be generated. \n4. Please use the most relevant table(s). \n5. If the question h
as been asked and answered before, please repeat the answer exactly as it was given before. \n"}, {'role':
'user', 'content': 'Show me a list of tables in the SQLite database'}, {'role': 'assistant', 'content': "SE
LECT name FROM sqlite_master WHERE type='table';"}, {'role': 'user', 'content': 'what are the top 5 countri
es that customers come from?'}, {'role': 'assistant', 'content': 'SELECT Country, COUNT(*) as CustomerCount
\nFROM Customer\nGROUP BY Country\nORDER BY CustomerCount DESC\nLIMIT 5;'}, {'role': 'user', 'content': 'Ho
w many records are in table called customer'}, {'role': 'assistant', 'content': 'SELECT COUNT(*) FROM Custo
mer;'}, {'role': 'user', 'content': 'How many customers are there'}, {'role': 'assistant', 'content': 'SELE
CT COUNT(*) FROM Customer;'}, {'role': 'user', 'content': '  \n    List all albums and their corresponding
artist names  \n'}]
Using model claude-3-5-sonnet-20240620 for 801.0 tokens (approx)
SELECT
    Album.Title AS AlbumTitle,
    Artist.Name AS ArtistName
FROM
    Album
JOIN
```

```
    Artist ON Album.ArtistId = Artist.ArtistId
ORDER BY
    Artist.Name, Album.Title;
SELECT
    Album.Title AS AlbumTitle,
    Artist.Name AS ArtistName
FROM
    Album
JOIN
    Artist ON Album.ArtistId = Artist.ArtistId
ORDER BY
    Artist.Name, Album.Title;
SELECT
    Album.Title AS AlbumTitle,
    Artist.Name AS ArtistName
FROM
    Album
JOIN
    Artist ON Album.ArtistId = Artist.ArtistId
ORDER BY
    Artist.Name, Album.Title;
                                AlbumTitle  \
0    For Those About To Rock We Salute You
1                        Let There Be Rock
2            A Copland Celebration, Vol. I
3                                   Worlds
4        The World of Classical Favourites
..                                     ...
342                     Vinicius De Moraes
343              Bach: Goldberg Variations
344       Bartok: Violin & Viola Concertos
345                 Bach: The Cello Suites
346                    Ao Vivo [IMPORT]

                                            ArtistName
0                                                AC/DC
1                                                AC/DC
2            Aaron Copland & London Symphony Orchestra
3                                       Aaron Goldberg
4    Academy of St. Martin in the Fields & Sir Nevi...
..                                                 ...
342                                 Vinícius De Moraes
```

```
343                                        Wilhelm Kempff
344                                        Yehudi Menuhin
345                                              Yo-Yo Ma
346                                        Zeca Pagodinho

[347 rows x 2 columns]
Using model claude-3-5-sonnet-20240620 for 210.5 tokens (approx)
```


Albums by Artist
Number of Albums
For Those About To Rock We Salute You
AC/DC
Aisha Duo
Audioslave
Black Sabbath
Charles Dutoit & L'Orchestre Symphonique de Montréal
Cláudio Zoli
Edo de Waart & San Francisco Symphony
Felix Schmidt, London Symphony Orchestra & Rafael Frühbeck de Burgos
Gonzaguinha
Incognito
Jota Quest
Leonard Bernstein & New York Philharmonic
Martin Roscoe
Motörhead
Orchestre Révolutionnaire et Romantique & John Eliot Gardiner
Philip Glass Ensemble
Roger Norrington, London Classical Players
Smashing Pumpkins
The Clash
Tim Maia
Wilhelm Kempff
Artist

```
Out[22]: ('SELECT \n    Album.Title AS AlbumTitle, \n    Artist.Name AS ArtistName\nFROM \n    Album\nJOIN \n    Ar
         tist ON Album.ArtistId = Artist.ArtistId\nORDER BY \n    Artist.Name, Album.Title;',
                                          AlbumTitle  \
          0    For Those About To Rock We Salute You
          1                       Let There Be Rock
          2            A Copland Celebration, Vol. I
          3                                  Worlds
          4        The World of Classical Favourites
          ..                                    ...
          342                     Vinicius De Moraes
          343              Bach: Goldberg Variations
          344       Bartok: Violin & Viola Concertos
          345                 Bach: The Cello Suites
          346                    Ao Vivo [IMPORT]

                                                    ArtistName
          0                                             AC/DC
          1                                             AC/DC
          2            Aaron Copland & London Symphony Orchestra
          3                                     Aaron Goldberg
          4    Academy of St. Martin in the Fields & Sir Nevi...
          ..                                              ...
          342                               Vinícius De Moraes
          343                                   Wilhelm Kempff
          344                                   Yehudi Menuhin
          345                                        Yo-Yo Ma
          346                                   Zeca Pagodinho

          [347 rows x 2 columns],
          Figure({
              'data': [{'alignmentgroup': 'True',
                        'hovertemplate': 'ArtistName=%{x}<br>AlbumTitle=%{y}<extra></extra>',
                        'legendgroup': '',
                        'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
                        'name': '',
                        'offsetgroup': '',
                        'orientation': 'v',
                        'showlegend': False,
                        'textposition': 'outside',
                        'texttemplate': '%{y}',
                        'type': 'bar',
                        'x': array(['AC/DC', 'AC/DC', 'Aaron Copland & London Symphony Orchestra', ...,
                                    'Yehudi Menuhin', 'Yo-Yo Ma', 'Zeca Pagodinho'], dtype=object),
                        'xaxis': 'x',
                        'y': array(['For Those About To Rock We Salute You', 'Let There Be Rock',
                                    'A Copland Celebration, Vol. I', ...,
                                    'Bartok: Violin & Viola Concertos', 'Bach: The Cello Suites',
                                    'Ao Vivo [IMPORT]'], dtype=object),
                        'yaxis': 'y'}],
              'layout': {'barmode': 'relative',
                         'legend': {'tracegroupgap': 0},
                         'template': '...',
                         'title': {'text': 'Albums by Artist'},
                         'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'tickangle': 45, 'title': {'text': 'Artis
         t'}},
                         'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'Number of Albums'}}}
          }))
```

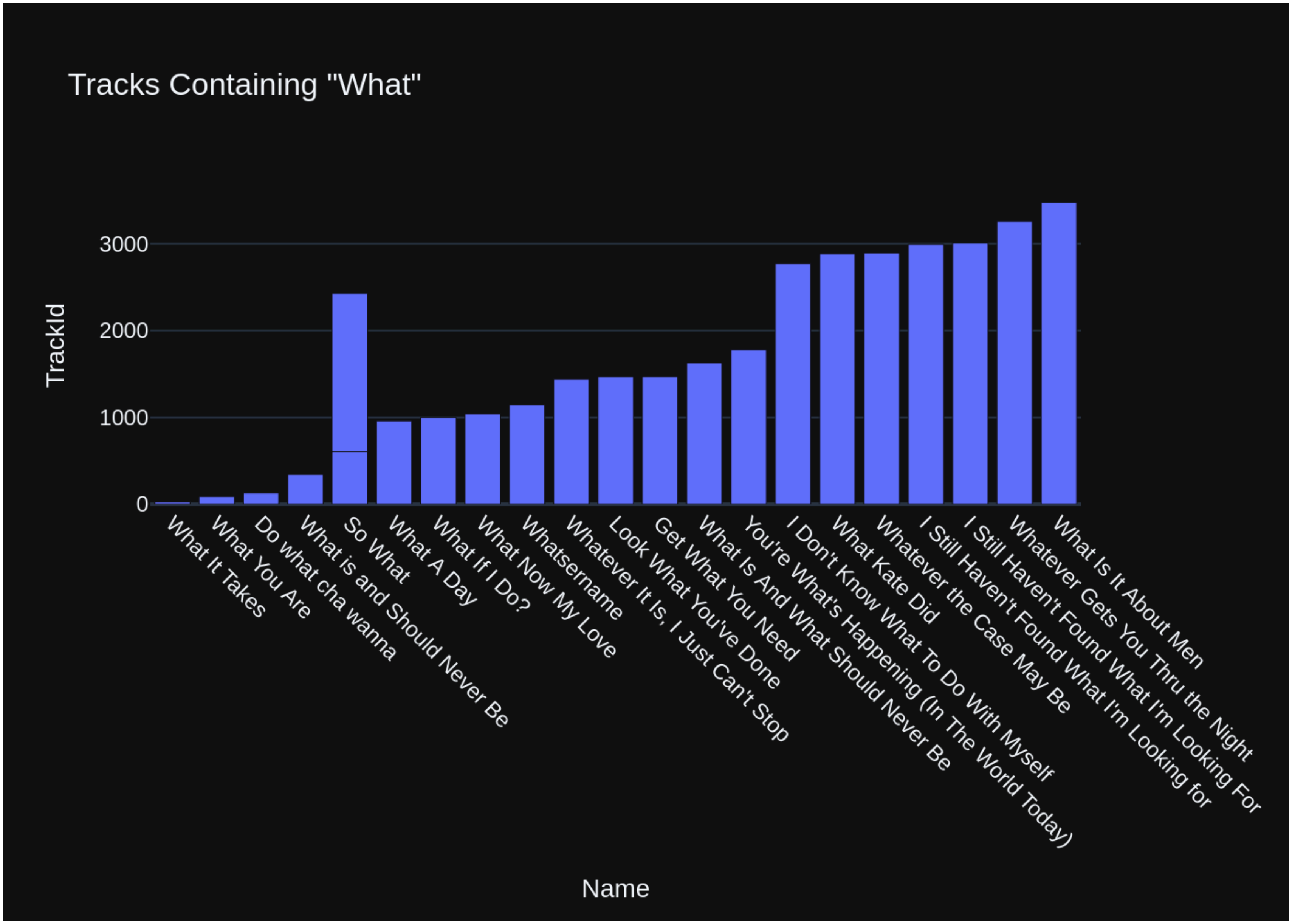
In [23]:

```python
question = """
    Find all tracks with a name containing "What" (case-insensitive)
"""

vn.ask(question=question)
```

```
Number of requested results 10 is greater than number of elements in index 5, updating n_results = 5
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
[{'role': 'system', 'content': "You are a SQLite expert. Please help to generate a SQL query to answer the
question. Your response should ONLY be based on the given context and follow the response guidelines and fo
rmat instructions. \n===Tables \nCREATE INDEX IFK_TrackGenreId ON Track (GenreId)\n\nCREATE TABLE Track\n
(\n    TrackId INTEGER  NOT NULL,\n    Name NVARCHAR(200)  NOT NULL,\n    AlbumId INTEGER,\n    MediaTypeId
INTEGER  NOT NULL,\n    GenreId INTEGER,\n    Composer NVARCHAR(220),\n    Milliseconds INTEGER  NOT NUL
L,\n    Bytes INTEGER,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (Track
Id),\n    FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n
FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIG
N KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n
CREATE INDEX IFK_TrackAlbumId ON Track (AlbumId)\n\nCREATE INDEX IFK_PlaylistTrackTrackId ON PlaylistTrack
(TrackId)\n\nCREATE INDEX IFK_TrackMediaTypeId ON Track (MediaTypeId)\n\nCREATE INDEX IFK_InvoiceLineTrackI
d ON InvoiceLine (TrackId)\n\nCREATE TABLE PlaylistTrack\n(\n    PlaylistId INTEGER  NOT NULL,\n    TrackId
INTEGER  NOT NULL,\n    CONSTRAINT PK_PlaylistTrack PRIMARY KEY  (PlaylistId, TrackId),\n    FOREIGN KEY (P
laylistId) REFERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY
(TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_A
lbumArtistId ON Album (ArtistId)\n\nCREATE TABLE Album\n(\n    AlbumId INTEGER  NOT NULL,\n    Title NVARCH
AR(160)  NOT NULL,\n    ArtistId INTEGER  NOT NULL,\n    CONSTRAINT PK_Album PRIMARY KEY  (AlbumId),\n    F
OREIGN KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREA
TE TABLE Playlist\n(\n    PlaylistId INTEGER  NOT NULL,\n    Name NVARCHAR(120),\n    CONSTRAINT PK_Playlis
t PRIMARY KEY  (PlaylistId)\n)\n\n\n===Additional Context \n\nIn the chinook database invoice means order\n
\n===Response Guidelines \n1. If the provided context is sufficient, please generate a valid SQL query with
out any explanations for the question. \n2. If the provided context is almost sufficient but requires knowl
edge of a specific string in a particular column, please generate an intermediate SQL query to find the dis
tinct strings in that column. Prepend the query with a comment saying intermediate_sql \n3. If the provided
context is insufficient, please explain why it can't be generated. \n4. Please use the most relevant table
(s). \n5. If the question has been asked and answered before, please repeat the answer exactly as it was gi
ven before. \n"}, {'role': 'user', 'content': '  \n    List all albums and their corresponding artist names
\n'}, {'role': 'assistant', 'content': 'SELECT \n    Album.Title AS AlbumTitle, \n    Artist.Name AS Artist
Name\nFROM \n    Album\nJOIN \n    Artist ON Album.ArtistId = Artist.ArtistId\nORDER BY \n    Artist.Name,
Album.Title;'}, {'role': 'user', 'content': 'Show me a list of tables in the SQLite database'}, {'role': 'a
ssistant', 'content': "SELECT name FROM sqlite_master WHERE type='table';"}, {'role': 'user', 'content': 'w
hat are the top 5 countries that customers come from?'}, {'role': 'assistant', 'content': 'SELECT Country,
COUNT(*) as CustomerCount\nFROM Customer\nGROUP BY Country\nORDER BY CustomerCount DESC\nLIMIT 5;'}, {'rol
e': 'user', 'content': 'How many customers are there'}, {'role': 'assistant', 'content': 'SELECT COUNT(*) F
ROM Customer;'}, {'role': 'user', 'content': 'How many records are in table called customer'}, {'role': 'as
sistant', 'content': 'SELECT COUNT(*) FROM Customer;'}, {'role': 'user', 'content': '  \n    Find all track
s with a name containing "What" (case-insensitive)\n'}]
Using model claude-3-5-sonnet-20240620 for 846.5 tokens (approx)
SELECT TrackId, Name
FROM Track
WHERE Name LIKE '%What%' COLLATE NOCASE;
SELECT TrackId, Name
```

```
FROM Track
WHERE Name LIKE '%What%' COLLATE NOCASE;
SELECT TrackId, Name
FROM Track
WHERE Name LIKE '%What%' COLLATE NOCASE;
    TrackId                                             Name
0        26                                    What It Takes
1        88                                     What You Are
2       130                                 Do what cha wanna
3       342                     What is and Should Never Be
4       607                                          So What
5       960                                       What A Day
6      1000                                    What If I Do?
7      1039                                 What Now My Love
8      1145                                      Whatsername
9      1440               Whatever It Is, I Just Can't Stop
10     1469                            Look What You've Done
11     1470                                Get What You Need
12     1628                 What Is And What Should Never Be
13     1778  You're What's Happening (In The World Today)
14     1823                                          So What
15     2772             I Don't Know What To Do With Myself
16     2884                                    What Kate Did
17     2893                        Whatever the Case May Be
18     2992    I Still Haven't Found What I'm Looking for
19     3007    I Still Haven't Found What I'm Looking For
20     3258                 Whatever Gets You Thru the Night
21     3475                            What Is It About Men
Using model claude-3-5-sonnet-20240620 for 185.0 tokens (approx)
```


Tracks Containing "What"
TrackId
3000
2000
1000
0
What It Takes
What You Are
Do what cha wanna
What is and Should Never Be
So What
What A Day
What If I Do?
What Now My Love
Whatsername
Whatever It Is, I Just Can't Stop
Look What You've Done
Get What You Need
What Is And What Should Never Be
You're What's Happening (In The World Today)
I Don't Know What To Do With Myself
What Kate Did
Whatever the Case May Be
I Still Haven't Found What I'm Looking for
I Still Haven't Found What I'm Looking For
Whatever Gets You Thru the Night
What Is It About Men
Name

```
Out[23]: ("SELECT TrackId, Name\nFROM Track\nWHERE Name LIKE '%What%' COLLATE NOCASE;",
              TrackId                                          Name
          0        26                                  What It Takes
          1        88                                   What You Are
          2       130                              Do what cha wanna
          3       342                   What is and Should Never Be
          4       607                                        So What
          5       960                                     What A Day
          6      1000                                  What If I Do?
          7      1039                               What Now My Love
          8      1145                                    Whatsername
          9      1440             Whatever It Is, I Just Can't Stop
          10     1469                          Look What You've Done
          11     1470                              Get What You Need
          12     1628              What Is And What Should Never Be
          13     1778  You're What's Happening (In The World Today)
          14     1823                                        So What
          15     2772            I Don't Know What To Do With Myself
          16     2884                                  What Kate Did
          17     2893                     Whatever the Case May Be
          18     2992    I Still Haven't Found What I'm Looking for
          19     3007    I Still Haven't Found What I'm Looking For
          20     3258              Whatever Gets You Thru the Night
          21     3475                           What Is It About Men,
          Figure({
              'data': [{'alignmentgroup': 'True',
                        'hovertemplate': 'Name=%{x}<br>TrackId=%{y}<extra></extra>',
                        'legendgroup': '',
                        'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
                        'name': '',
                        'offsetgroup': '',
                        'orientation': 'v',
                        'showlegend': False,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['What It Takes', 'What You Are', 'Do what cha wanna',
                                    'What is and Should Never Be', 'So What', 'What A Day', 'What If I Do?',
                                    'What Now My Love', 'Whatsername', "Whatever It Is, I Just Can't Stop",
                                    "Look What You've Done", 'Get What You Need',
                                    'What Is And What Should Never Be',
                                    "You're What's Happening (In The World Today)", 'So What',
                                    "I Don't Know What To Do With Myself", 'What Kate Did',
                                  'Whatever the Case May Be',
                                  "I Still Haven't Found What I'm Looking for",
                                  "I Still Haven't Found What I'm Looking For",
                                  'Whatever Gets You Thru the Night', 'What Is It About Men'],
                                 dtype=object),
                       'xaxis': 'x',
                       'y': array([  26,   88,  130,  342,  607,  960, 1000, 1039, 1145, 1440, 1469, 1470,
                                  1628, 1778, 1823, 2772, 2884, 2893, 2992, 3007, 3258, 3475]),
                       'yaxis': 'y'}],
             'layout': {'barmode': 'relative',
                        'legend': {'tracegroupgap': 0},
                        'template': '...',
                        'title': {'text': 'Tracks Containing "What"'},
                        'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'tickangle': 45, 'title': {'text': 'Nam
e'}},
                        'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'TrackId'}}}
 }))
```

In [24]:

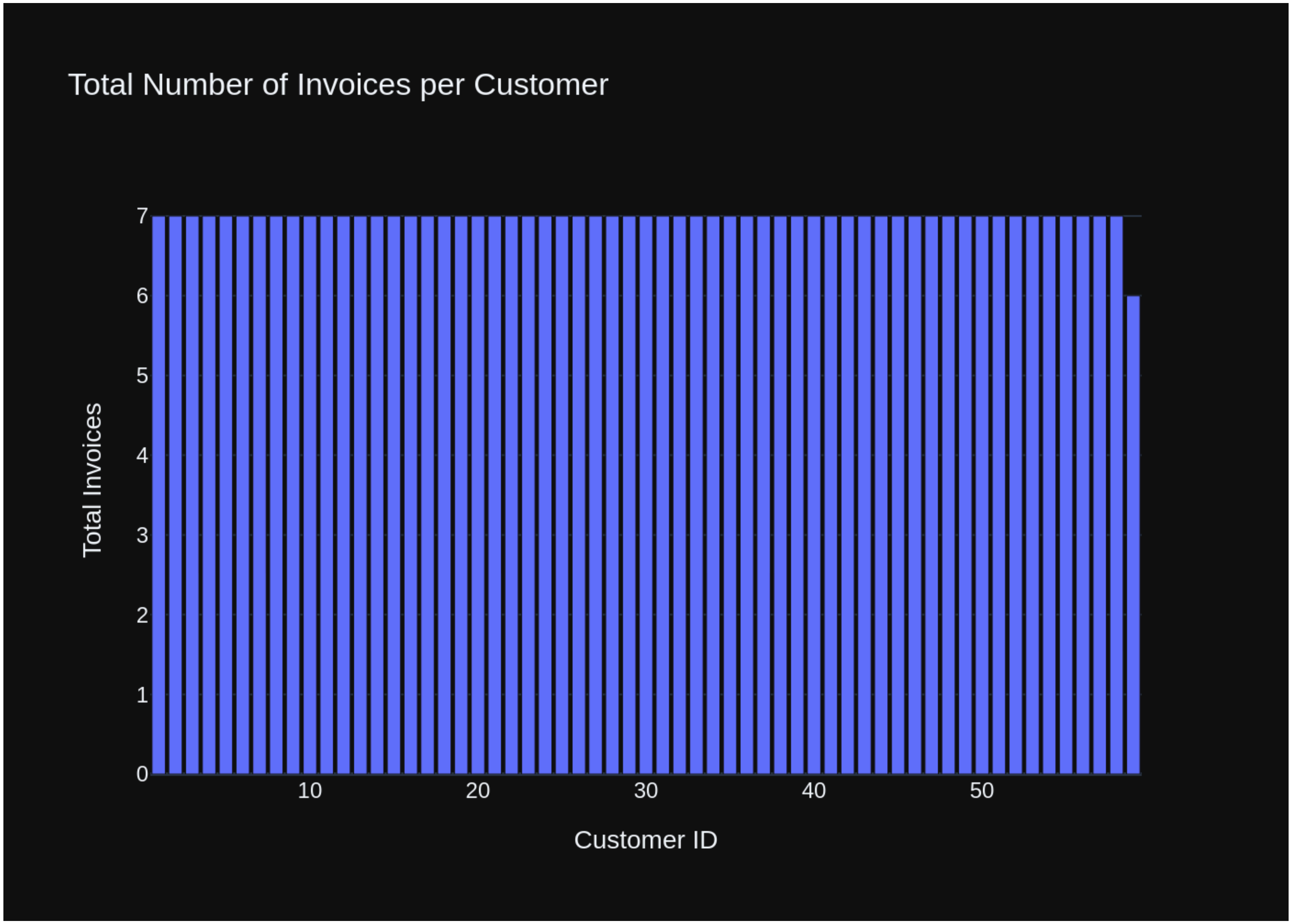
```
question = """
    Get the total number of invoices for each customer
"""

vn.ask(question=question)
```

```
Number of requested results 10 is greater than number of elements in index 6, updating n_results = 6
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
[{'role': 'system', 'content': "You are a SQLite expert. Please help to generate a SQL query to answer the
question. Your response should ONLY be based on the given context and follow the response guidelines and fo
rmat instructions. \n===Tables \nCREATE INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)\n\nCREATE TABLE
Invoice\n(\n    InvoiceId INTEGER  NOT NULL,\n    CustomerId INTEGER  NOT NULL,\n    InvoiceDate DATETIME
NOT NULL,\n    BillingAddress NVARCHAR(70),\n    BillingCity NVARCHAR(40),\n    BillingState NVARCHAR(4
0),\n    BillingCountry NVARCHAR(40),\n    BillingPostalCode NVARCHAR(10),\n    Total NUMERIC(10,2)  NOT NU
LL,\n    CONSTRAINT PK_Invoice PRIMARY KEY  (InvoiceId),\n    FOREIGN KEY (CustomerId) REFERENCES Customer
(CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceLineInvoiceId ON I
nvoiceLine (InvoiceId)\n\nCREATE TABLE InvoiceLine\n(\n    InvoiceLineId INTEGER  NOT NULL,\n    InvoiceId
INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    Quantity IN
TEGER  NOT NULL,\n    CONSTRAINT PK_InvoiceLine PRIMARY KEY  (InvoiceLineId),\n    FOREIGN KEY (InvoiceId)
REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (TrackId) RE
FERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceLineTr
ackId ON InvoiceLine (TrackId)\n\nCREATE TABLE Customer\n(\n    CustomerId INTEGER  NOT NULL,\n    FirstNam
e NVARCHAR(40)  NOT NULL,\n    LastName NVARCHAR(20)  NOT NULL,\n    Company NVARCHAR(80),\n    Address NVA
RCHAR(70),\n    City NVARCHAR(40),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVAR
CHAR(10),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60)  NOT NULL,\n    SupportRe
pId INTEGER,\n    CONSTRAINT PK_Customer PRIMARY KEY  (CustomerId),\n    FOREIGN KEY (SupportRepId) REFEREN
CES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_CustomerSupp
ortRepId ON Customer (SupportRepId)\n\nCREATE INDEX IFK_EmployeeReportsTo ON Employee (ReportsTo)\n\nCREATE
TABLE Employee\n(\n    EmployeeId INTEGER  NOT NULL,\n    LastName NVARCHAR(20)  NOT NULL,\n    FirstName N
VARCHAR(20)  NOT NULL,\n    Title NVARCHAR(30),\n    ReportsTo INTEGER,\n    BirthDate DATETIME,\n    HireD
ate DATETIME,\n    Address NVARCHAR(70),\n    City NVARCHAR(40),\n    State NVARCHAR(40),\n    Country NVAR
CHAR(40),\n    PostalCode NVARCHAR(10),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(24),\n    Email NVARCHAR
(60),\n    CONSTRAINT PK_Employee PRIMARY KEY  (EmployeeId),\n    FOREIGN KEY (ReportsTo) REFERENCES Employ
ee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Track\n(\n    TrackId INTE
GER  NOT NULL,\n    Name NVARCHAR(200)  NOT NULL,\n    AlbumId INTEGER,\n    MediaTypeId INTEGER  NOT NUL
L,\n    GenreId INTEGER,\n    Composer NVARCHAR(220),\n    Milliseconds INTEGER  NOT NULL,\n    Bytes INTEG
ER,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (TrackId),\n    FOREIGN K
EY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (Ge
nreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (MediaTyp
eId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n===Additional
Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided cont
ext is sufficient, please generate a valid SQL query without any explanations for the question. \n2. If the
provided context is almost sufficient but requires knowledge of a specific string in a particular column, p
lease generate an intermediate SQL query to find the distinct strings in that column. Prepend the query wit
h a comment saying intermediate_sql \n3. If the provided context is insufficient, please explain why it ca
n't be generated. \n4. Please use the most relevant table(s). \n5. If the question has been asked and answe
red before, please repeat the answer exactly as it was given before. \n"}, {'role': 'user', 'content': 'How
many records are in table called customer'}, {'role': 'assistant', 'content': 'SELECT COUNT(*) FROM Custome
r;'}, {'role': 'user', 'content': 'How many customers are there'}, {'role': 'assistant', 'content': 'SELECT
COUNT(*) FROM Customer;'}, {'role': 'user', 'content': 'what are the top 5 countries that customers come fr
om?'}, {'role': 'assistant', 'content': 'SELECT Country, COUNT(*) as CustomerCount\nFROM Customer\nGROUP BY
Country\nORDER BY CustomerCount DESC\nLIMIT 5;'}, {'role': 'user', 'content': '  \n    List all albums and
their corresponding artist names  \n'}, {'role': 'assistant', 'content': 'SELECT \n    Album.Title AS Album
Title, \n    Artist.Name AS ArtistName\nFROM \n    Album\nJOIN \n    Artist ON Album.ArtistId = Artist.Arti
stId\nORDER BY \n    Artist.Name, Album.Title;'}, {'role': 'user', 'content': '  \n    Find all tracks with
a name containing "What" (case-insensitive)\n'}, {'role': 'assistant', 'content': "SELECT TrackId, Name\nFR
OM Track\nWHERE Name LIKE '%What%' COLLATE NOCASE;"}, {'role': 'user', 'content': 'Show me a list of tables
in the SQLite database'}, {'role': 'assistant', 'content': "SELECT name FROM sqlite_master WHERE type='tabl
e';"}, {'role': 'user', 'content': '  \n    Get the total number of invoices for each customer\n'}]
Using model claude-3-5-sonnet-20240620 for 1201.5 tokens (approx)
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    COUNT(Invoice.InvoiceId) AS TotalInvoices
FROM
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
GROUP BY
    Customer.CustomerId
ORDER BY
    TotalInvoices DESC;
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    COUNT(Invoice.InvoiceId) AS TotalInvoices
FROM
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
GROUP BY
    Customer.CustomerId
ORDER BY
    TotalInvoices DESC;
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    COUNT(Invoice.InvoiceId) AS TotalInvoices
FROM
```

```
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
GROUP BY
    Customer.CustomerId
ORDER BY
    TotalInvoices DESC;
    CustomerId  FirstName      LastName  TotalInvoices
0            1       Luís     Gonçalves              7
1            2     Leonie        Köhler              7
2            3   François      Tremblay              7
3            4      Bjørn        Hansen              7
4            5  František   Wichterlová              7
5            6     Helena          Holý              7
6            7     Astrid        Gruber              7
7            8       Daan       Peeters              7
8            9       Kara       Nielsen              7
9           10    Eduardo       Martins              7
10          11  Alexandre         Rocha              7
11          12    Roberto       Almeida              7
12          13   Fernanda         Ramos              7
13          14       Mark       Philips              7
14          15   Jennifer      Peterson              7
15          16      Frank        Harris              7
16          17       Jack         Smith              7
17          18   Michelle        Brooks              7
18          19        Tim         Goyer              7
19          20        Dan        Miller              7
20          21      Kathy         Chase              7
21          22    Heather       Leacock              7
22          23       John        Gordon              7
23          24      Frank       Ralston              7
24          25     Victor       Stevens              7
25          26    Richard    Cunningham              7
26          27    Patrick          Gray              7
27          28     Julia        Barnett              7
28          29     Robert         Brown              7
29          30     Edward       Francis              7
30          31     Martha          Silk              7
31          32      Aaron      Mitchell              7
32          33     Ellie       Sullivan              7
33          34       João     Fernandes              7
```

```
34          35   Madalena       Sampaio              7
35          36     Hannah     Schneider              7
36          37       Fynn    Zimmermann              7
37          38     Niklas      Schröder              7
38          39    Camille       Bernard              7
39          40  Dominique      Lefebvre              7
40          41       Marc        Dubois              7
41          42      Wyatt        Girard              7
42          43   Isabelle       Mercier              7
43          44      Terhi    Hämäläinen              7
44          45   Ladislav        Kovács              7
45          46       Hugh      O'Reilly              7
46          47      Lucas       Mancini              7
47          48   Johannes  Van der Berg              7
48          49  Stanisław        Wójcik              7
49          50   Enrique          Muñoz              7
50          51     Joakim     Johansson              7
51          52       Emma         Jones              7
52          53       Phil        Hughes              7
53          54      Steve        Murray              7
54          55       Mark        Taylor              7
55          56      Diego     Gutiérrez              7
56          57       Luis         Rojas              7
57          58      Manoj        Pareek              7
58          59       Puja    Srivastava              6
Using model claude-3-5-sonnet-20240620 for 248.25 tokens (approx)
```


Total Number of Invoices per Customer
Total Invoices
0
1
2
3
4
5
6
7
10
20
30
40
50
Customer ID

```
Out[24]: ('SELECT \n    Customer.CustomerId,\n    Customer.FirstName,\n    Customer.LastName,\n    COUNT(Invoice.In
         voiceId) AS TotalInvoices\nFROM \n    Customer\nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.C
         ustomerId\nGROUP BY \n    Customer.CustomerId\nORDER BY \n    TotalInvoices DESC;',
              CustomerId  FirstName      LastName  TotalInvoices
          0            1       Luís     Gonçalves              7
          1            2     Leonie        Köhler              7
          2            3   François      Tremblay              7
          3            4      Bjørn        Hansen              7
          4            5  František   Wichterlová              7
          5            6     Helena          Holý              7
          6            7     Astrid        Gruber              7
          7            8       Daan       Peeters              7
          8            9       Kara       Nielsen              7
          9           10    Eduardo       Martins              7
          10          11  Alexandre         Rocha              7
          11          12    Roberto       Almeida              7
          12          13   Fernanda         Ramos              7
          13          14       Mark       Philips              7
          14          15   Jennifer      Peterson              7
          15          16      Frank        Harris              7
          16          17       Jack         Smith              7
          17          18   Michelle        Brooks              7
          18          19        Tim         Goyer              7
          19          20        Dan        Miller              7
          20          21      Kathy         Chase              7
          21          22    Heather       Leacock              7
          22          23       John        Gordon              7
          23          24      Frank       Ralston              7
          24          25     Victor       Stevens              7
          25          26    Richard    Cunningham              7
          26          27    Patrick          Gray              7
          27          28      Julia       Barnett              7
          28          29     Robert         Brown              7
          29          30     Edward       Francis              7
          30          31     Martha          Silk              7
          31          32      Aaron      Mitchell              7
          32          33      Ellie      Sullivan              7
          33          34       João     Fernandes              7
          34          35   Madalena       Sampaio              7
          35          36     Hannah     Schneider              7
          36          37       Fynn    Zimmermann              7
          37          38    Niklas      Schröder              7
```

```
38          39    Camille       Bernard              7
39          40  Dominique      Lefebvre              7
40          41       Marc        Dubois              7
41          42      Wyatt        Girard              7
42          43   Isabelle       Mercier              7
43          44      Terhi    Hämäläinen              7
44          45   Ladislav        Kovács              7
45          46       Hugh      O'Reilly              7
46          47      Lucas       Mancini              7
47          48   Johannes  Van der Berg              7
48          49  Stanisław        Wójcik              7
49          50    Enrique         Muñoz              7
50          51     Joakim     Johansson              7
51          52       Emma         Jones              7
52          53       Phil        Hughes              7
53          54      Steve        Murray              7
54          55       Mark        Taylor              7
55          56      Diego     Gutiérrez              7
56          57       Luis         Rojas              7
57          58      Manoj        Pareek              7
58          59       Puja    Srivastava              6,
Figure({
    'data': [{'alignmentgroup': 'True',
              'customdata': array([['Luís', 'Gonçalves'],
                                   ['Leonie', 'Köhler'],
                                   ['François', 'Tremblay'],
                                   ['Bjørn', 'Hansen'],
                                   ['František', 'Wichterlová'],
                                   ['Helena', 'Holý'],
                                   ['Astrid', 'Gruber'],
                                   ['Daan', 'Peeters'],
                                   ['Kara', 'Nielsen'],
                                   ['Eduardo', 'Martins'],
                                   ['Alexandre', 'Rocha'],
                                   ['Roberto', 'Almeida'],
                                   ['Fernanda', 'Ramos'],
                                   ['Mark', 'Philips'],
                                   ['Jennifer', 'Peterson'],
                                   ['Frank', 'Harris'],
                                   ['Jack', 'Smith'],
                                   ['Michelle', 'Brooks'],
                                   ['Tim', 'Goyer'],
                                   ['Dan', 'Miller'],
                                   ['Kathy', 'Chase'],
                                   ['Heather', 'Leacock'],
                                   ['John', 'Gordon'],
                                   ['Frank', 'Ralston'],
                                   ['Victor', 'Stevens'],
                                   ['Richard', 'Cunningham'],
                                   ['Patrick', 'Gray'],
                                   ['Julia', 'Barnett'],
                                   ['Robert', 'Brown'],
                                   ['Edward', 'Francis'],
                                   ['Martha', 'Silk'],
                                   ['Aaron', 'Mitchell'],
                                   ['Ellie', 'Sullivan'],
                                   ['João', 'Fernandes'],
                                   ['Madalena', 'Sampaio'],
                                   ['Hannah', 'Schneider'],
                                   ['Fynn', 'Zimmermann'],
                                   ['Niklas', 'Schröder'],
                                   ['Camille', 'Bernard'],
                                   ['Dominique', 'Lefebvre'],
                                   ['Marc', 'Dubois'],
                                   ['Wyatt', 'Girard'],
                                   ['Isabelle', 'Mercier'],
                                   ['Terhi', 'Hämäläinen'],
                                   ['Ladislav', 'Kovács'],
                                   ['Hugh', "O'Reilly"],
                                   ['Lucas', 'Mancini'],
                                   ['Johannes', 'Van der Berg'],
                                   ['Stanisław', 'Wójcik'],
                                   ['Enrique', 'Muñoz'],
                                   ['Joakim', 'Johansson'],
                                   ['Emma', 'Jones'],
                                   ['Phil', 'Hughes'],
                                   ['Steve', 'Murray'],
                                   ['Mark', 'Taylor'],
                                   ['Diego', 'Gutiérrez'],
                                   ['Luis', 'Rojas'],
                                   ['Manoj', 'Pareek'],
                                   ['Puja', 'Srivastava']], dtype=object),
              'hovertemplate': ('Customer ID=%{x}<br>Total Invo' ... '{customdata[1]}<extra></extra>'),
              'legendgroup': '',
              'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
              'name': '',
              'offsetgroup': '',
              'orientation': 'v',
              'showlegend': False,
              'textposition': 'auto',
              'type': 'bar',
              'x': array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
                          19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36,
                          37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54,
                          55, 56, 57, 58, 59]),
              'xaxis': 'x',
              'y': array([7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
                          7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
                          7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 6]),
              'yaxis': 'y'}],
    'layout': {'barmode': 'relative',
               'legend': {'tracegroupgap': 0},
               'template': '...',
               'title': {'text': 'Total Number of Invoices per Customer'},
               'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'Customer ID'}},
               'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'Total Invoices'}}}
})
```

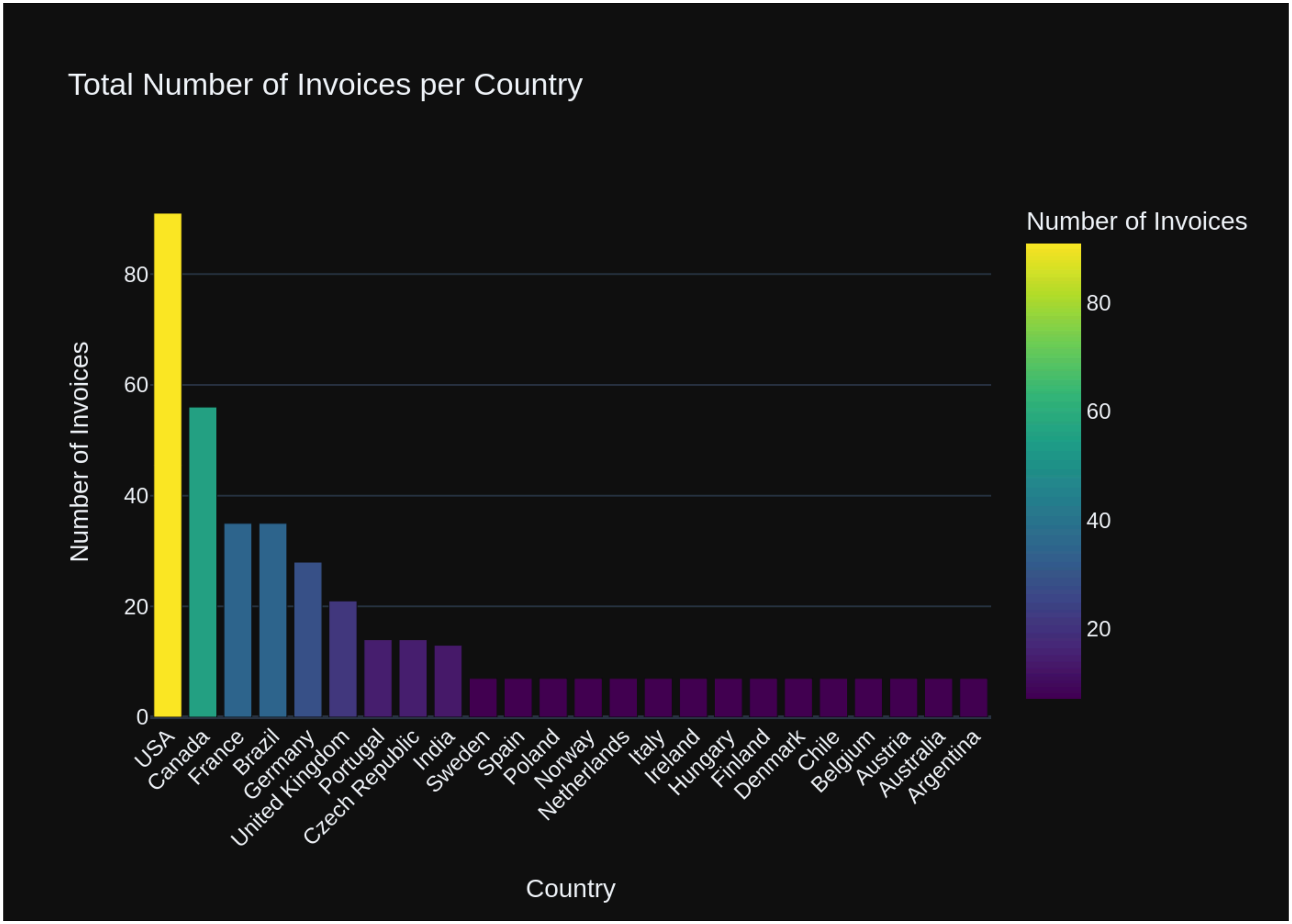
In [25]:
```
question = """
    Find the total number of invoices per country:
"""

vn.ask(question=question)
```

```
Number of requested results 10 is greater than number of elements in index 7, updating n_results = 7
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

[{'role': 'system', 'content': "You are a SQLite expert. Please help to generate a SQL query to answer the question. Your response should ONLY be based on the given context and follow the response guidelines and format instructions. \n===Tables \nCREATE TABLE Invoice\n(\n   InvoiceId INTEGER NOT NULL,\n   CustomerId INTEGER NOT NULL,\n   InvoiceDate DATETIME NOT NULL,\n   BillingAddress NVARCHAR(70),\n   BillingCity NVARCHAR(40),\n   BillingState NVARCHAR(40),\n   BillingCountry NVARCHAR(40),\n   BillingPostalCode NVARCHAR(10),\n   Total NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Invoice PRIMARY KEY (InvoiceId),\n   FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceLineInvoiceId ON InvoiceLine (InvoiceId)\n\nCREATE INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)\n\nCREATE TABLE InvoiceLine\n(\n   InvoiceLineId INTEGER NOT NULL,\n   InvoiceId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   Quantity INTEGER NOT NULL,\n   CONSTRAINT PK_InvoiceLine PRIMARY KEY (InvoiceLineId),\n   FOREIGN KEY (InvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceLineTrackId ON InvoiceLine (TrackId)\n\nCREATE TABLE Customer\n(\n   CustomerId INTEGER NOT NULL,\n   FirstName NVARCHAR(40) NOT NULL,\n   LastName NVARCHAR(20) NOT NULL,\n   Company NVARCHAR(80),\n   Address NVARCHAR(70),\n   City NVARCHAR(40),\n   State NVARCHAR(40),\n   Country NVARCHAR(40),\n   PostalCode NVARCHAR(10),\n   Phone NVARCHAR(24),\n   Fax NVARCHAR(24),\n   Email NVARCHAR(60) NOT NULL,\n   SupportRepId INTEGER,\n   CONSTRAINT PK_Customer PRIMARY KEY (CustomerId),\n   FOREIGN KEY (SupportRepId) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Employee\n(\n   EmployeeId INTEGER NOT NULL,\n   LastName NVARCHAR(20) NOT NULL,\n   FirstName NVARCHAR(20) NOT NULL,\n   Title NVARCHAR(30),\n   ReportsTo INTEGER,\n   BirthDate DATETIME,\n   HireDate DATETIME,\n   Address NVARCHAR(70),\n   City NVARCHAR(40),\n   State NVARCHAR(40),\n   Country NVARCHAR(40),\n   PostalCode NVARCHAR(10),\n   Phone NVARCHAR(24),\n   Fax NVARCHAR(24),\n   Email NVARCHAR(60),\n   CONSTRAINT PK_Employee PRIMARY KEY (EmployeeId),\n   FOREIGN KEY (ReportsTo) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Track\n(\n   TrackId INTEGER NOT NULL,\n   Name NVARCHAR(200) NOT NULL,\n   AlbumId INTEGER,\n   MediaTypeId INTEGER NOT NULL,\n   GenreId INTEGER,\n   Composer NVARCHAR(220),\n   Milliseconds INTEGER NOT NULL,\n   Bytes INTEGER,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Track PRIMARY KEY (TrackId),\n   FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_EmployeeReportsTo ON Employee (ReportsTo)\n\nCREATE TABLE Album\n(\n   AlbumId INTEGER NOT NULL,\n   Title NVARCHAR(160) NOT NULL,\n   ArtistId INTEGER NOT NULL,\n   CONSTRAINT PK_Album PRIMARY KEY (AlbumId),\n   FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n===Additional Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficient, please generate a valid SQL query without any explanations for the question. \n2. If the provided context is almost sufficient but requires knowledge of a specific string in a particular column, please generate an intermediate SQL query to find the distinct strings in that column. Prepend the query with a comment saying intermediate_sql \n3. If the provided context is insufficient, please explain why it can't be generated. \n4. Please use the most relevant table(s). \n5. If the question has been asked and answered before, please repeat the answer exactly as it was given before. \n"}, {'role': 'user', 'content': ' \n   Get the total number of invoices for each customer\n'}, {'role': 'assistant', 'content': 'SELECT \n   Cust

```
omer.CustomerId,\n    Customer.FirstName,\n    Customer.LastName,\n    COUNT(Invoice.InvoiceId) AS TotalInv
oices\nFROM \n    Customer\nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId\nGROUP BY
\n    Customer.CustomerId\nORDER BY \n    TotalInvoices DESC;'}, {'role': 'user', 'content': 'what are the
top 5 countries that customers come from?'}, {'role': 'assistant', 'content': 'SELECT Country, COUNT(*) as
CustomerCount\nFROM Customer\nGROUP BY Country\nORDER BY CustomerCount DESC\nLIMIT 5;'}, {'role': 'user',
'content': 'How many records are in table called customer'}, {'role': 'assistant', 'content': 'SELECT COUNT
(*) FROM Customer;'}, {'role': 'user', 'content': 'How many customers are there'}, {'role': 'assistant', 'c
ontent': 'SELECT COUNT(*) FROM Customer;'}, {'role': 'user', 'content': '  \n    Find all tracks with a nam
e containing "What" (case-insensitive)\n'}, {'role': 'assistant', 'content': "SELECT TrackId, Name\nFROM Tr
ack\nWHERE Name LIKE '%What%' COLLATE NOCASE;"}, {'role': 'user', 'content': '  \n    List all albums and t
heir corresponding artist names  \n'}, {'role': 'assistant', 'content': 'SELECT \n    Album.Title AS AlbumT
itle, \n    Artist.Name AS ArtistName\nFROM \n    Album\nJOIN \n    Artist ON Album.ArtistId = Artist.Artis
tId\nORDER BY \n    Artist.Name, Album.Title;'}, {'role': 'user', 'content': 'Show me a list of tables in t
he SQLite database'}, {'role': 'assistant', 'content': "SELECT name FROM sqlite_master WHERE type='tabl
e';"}, {'role': 'user', 'content': '  \n    Find the total number of invoices per country:\n'}]
Using model claude-3-5-sonnet-20240620 for 1335.5 tokens (approx)
SELECT
    BillingCountry,
    COUNT(*) AS TotalInvoices
FROM
    Invoice
GROUP BY
    BillingCountry
ORDER BY
    TotalInvoices DESC;
SELECT
    BillingCountry,
    COUNT(*) AS TotalInvoices
FROM
    Invoice
GROUP BY
    BillingCountry
ORDER BY
    TotalInvoices DESC;
SELECT
    BillingCountry,
    COUNT(*) AS TotalInvoices
FROM
    Invoice
GROUP BY
    BillingCountry
ORDER BY
```

```
    TotalInvoices DESC;
    BillingCountry  TotalInvoices
0              USA             91
1           Canada             56
2           France             35
3           Brazil             35
4          Germany             28
5   United Kingdom             21
6         Portugal             14
7   Czech Republic             14
8            India             13
9           Sweden              7
10           Spain              7
11          Poland              7
12          Norway              7
13     Netherlands              7
14           Italy              7
15         Ireland              7
16         Hungary              7
17         Finland              7
18         Denmark              7
19           Chile              7
20         Belgium              7
21         Austria              7
22       Australia              7
23       Argentina              7
Using model claude-3-5-sonnet-20240620 for 200.75 tokens (approx)
```

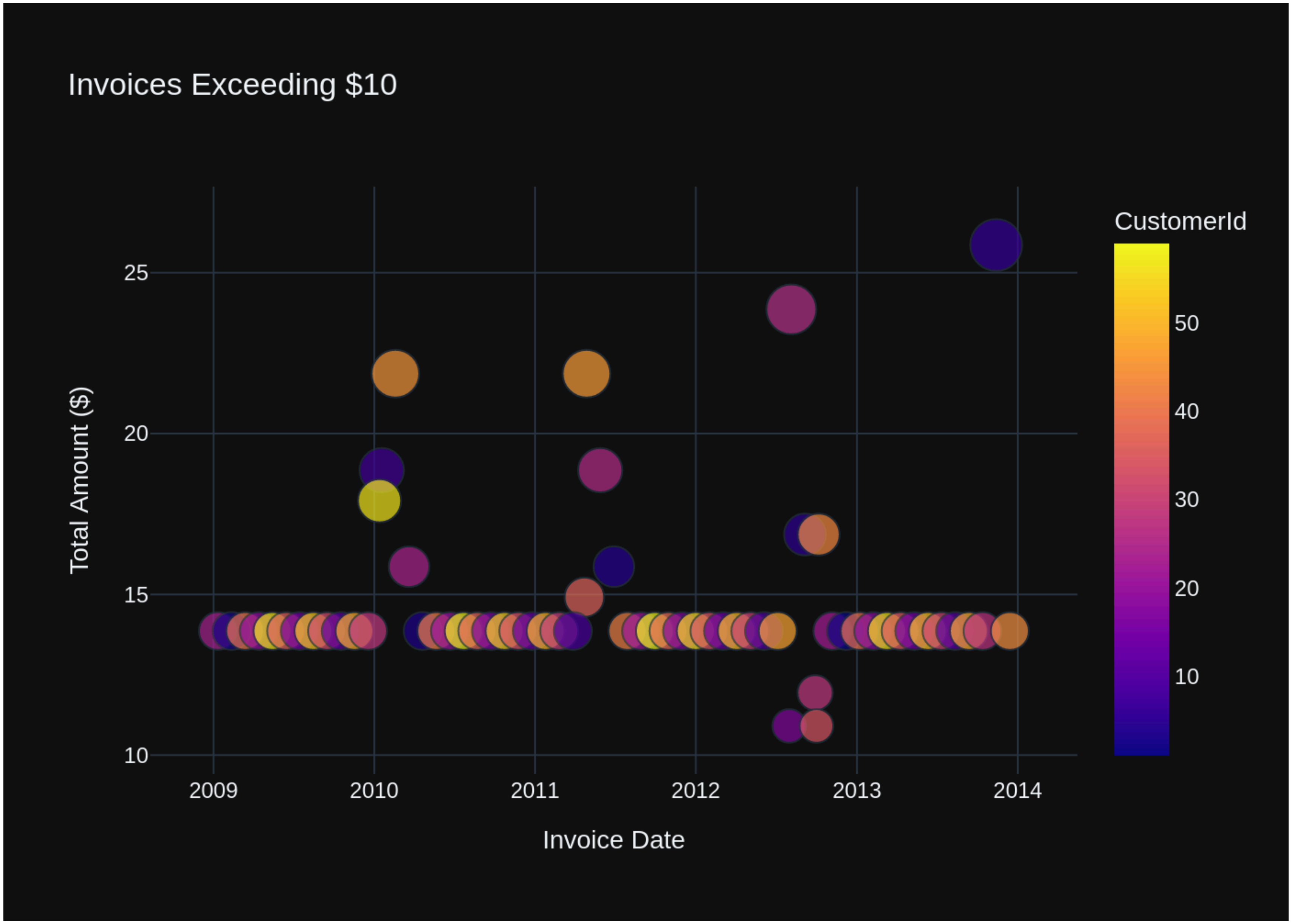

Total Number of Invoices per Country
Number of Invoices
0
20
40
60
80
Country
USA
Canada
France
Brazil
Germany
United Kingdom
Portugal
Czech Republic
India
Sweden
Spain
Poland
Norway
Netherlands
Italy
Ireland
Hungary
Finland
Denmark
Chile
Belgium
Austria
Australia
Argentina
Number of Invoices
20
40
60
80

```
Out[25]: ('SELECT \n    BillingCountry, \n    COUNT(*) AS TotalInvoices\nFROM \n    Invoice\nGROUP BY \n    Billing
         Country\nORDER BY \n    TotalInvoices DESC;',
              BillingCountry  TotalInvoices
          0             USA             91
          1          Canada             56
          2          France             35
          3          Brazil             35
          4         Germany             28
          5   United Kingdom            21
          6        Portugal             14
          7   Czech Republic            14
          8           India             13
          9          Sweden              7
          10          Spain              7
          11         Poland              7
          12         Norway              7
          13    Netherlands              7
          14          Italy              7
          15        Ireland              7
          16        Hungary              7
          17        Finland              7
          18        Denmark              7
          19          Chile              7
          20        Belgium              7
          21        Austria              7
          22      Australia              7
          23      Argentina              7,
          Figure({
              'data': [{'alignmentgroup': 'True',
                        'hovertemplate': 'Country=%{x}<br>Number of Invoices=%{marker.color}<extra></extra>',
                        'legendgroup': '',
                        'marker': {'color': array([91, 56, 35, 35, 28, 21, 14, 14, 13,  7,  7,  7,  7,  7,  7,  7,
         7,  7,
                                                  7,  7,  7,  7,  7,  7]),
                                   'coloraxis': 'coloraxis',
                                   'pattern': {'shape': ''}},
                        'name': '',
                        'offsetgroup': '',
                        'orientation': 'v',
                        'showlegend': False,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['USA', 'Canada', 'France', 'Brazil', 'Germany', 'United Kingdom',
                                    'Portugal', 'Czech Republic', 'India', 'Sweden', 'Spain', 'Poland',
                                    'Norway', 'Netherlands', 'Italy', 'Ireland', 'Hungary', 'Finland',
                                    'Denmark', 'Chile', 'Belgium', 'Austria', 'Australia', 'Argentina'],
                                   dtype=object),
                        'xaxis': 'x',
                        'y': array([91, 56, 35, 35, 28, 21, 14, 14, 13,  7,  7,  7,  7,  7,  7,  7,  7,  7,
                                     7,  7,  7,  7,  7,  7]),
                        'yaxis': 'y'}],
              'layout': {'barmode': 'relative',
                         'coloraxis': {'colorbar': {'title': {'text': 'Number of Invoices'}},
                                       'colorscale': [[0.0, '#440154'], [0.1111111111111111,
                                                      '#482878'], [0.2222222222222222,
                                                      '#3e4989'], [0.3333333333333333,
                                                      '#31688e'], [0.4444444444444444,
                                                      '#26828e'], [0.5555555555555556,
                                                      '#1f9e89'], [0.6666666666666666,
                                                      '#35b779'], [0.7777777777777778,
                                                      '#6ece58'], [0.8888888888888888,
                                                      '#b5de2b'], [1.0, '#fde725']]},
                         'legend': {'tracegroupgap': 0},
                         'template': '...',
                         'title': {'text': 'Total Number of Invoices per Country'},
                         'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'tickangle': -45, 'title': {'text': 'Countr
         y'}},
                         'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'Number of Invoices'}}}
          }))
```

In [26]:
```
question = """
    List all invoices with a total exceeding $10:
"""

vn.ask(question=question)
```

```
Number of requested results 10 is greater than number of elements in index 8, updating n_results = 8
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
[{'role': 'system', 'content': "You are a SQLite expert. Please help to generate a SQL query to answer the
question. Your response should ONLY be based on the given context and follow the response guidelines and fo
rmat instructions. \n===Tables \nCREATE TABLE InvoiceLine\n(\n    InvoiceLineId INTEGER  NOT NULL,\n    Inv
oiceId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    Quan
tity INTEGER  NOT NULL,\n    CONSTRAINT PK_InvoiceLine PRIMARY KEY  (InvoiceLineId),\n    FOREIGN KEY (Invo
iceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (Trac
kId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_Invoic
eLineInvoiceId ON InvoiceLine (InvoiceId)\n\nCREATE TABLE Invoice\n(\n    InvoiceId INTEGER  NOT NULL,\n
CustomerId INTEGER  NOT NULL,\n    InvoiceDate DATETIME  NOT NULL,\n    BillingAddress NVARCHAR(70),\n    B
illingCity NVARCHAR(40),\n    BillingState NVARCHAR(40),\n    BillingCountry NVARCHAR(40),\n    BillingPost
alCode NVARCHAR(10),\n    Total NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Invoice PRIMARY KEY  (InvoiceI
d),\n    FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO A
CTION\n)\n\nCREATE INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)\n\nCREATE INDEX IFK_InvoiceLineTrack
Id ON InvoiceLine (TrackId)\n\nCREATE TABLE Track\n(\n    TrackId INTEGER  NOT NULL,\n    Name NVARCHAR(20
0)  NOT NULL,\n    AlbumId INTEGER,\n    MediaTypeId INTEGER  NOT NULL,\n    GenreId INTEGER,\n    Composer
NVARCHAR(220),\n    Milliseconds INTEGER  NOT NULL,\n    Bytes INTEGER,\n    UnitPrice NUMERIC(10,2)  NOT N
ULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (TrackId),\n    FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId)
\n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\t
ON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId)
\n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_EmployeeReportsTo ON Employee (Reports
To)\n\nCREATE TABLE Customer\n(\n    CustomerId INTEGER  NOT NULL,\n    FirstName NVARCHAR(40)  NOT NULL,\n
LastName NVARCHAR(20)  NOT NULL,\n    Company NVARCHAR(80),\n    Address NVARCHAR(70),\n    City NVARCHAR(4
0),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVARCHAR(10),\n    Phone NVARCHAR(2
4),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60)  NOT NULL,\n    SupportRepId INTEGER,\n    CONSTRAINT PK
_Customer PRIMARY KEY  (CustomerId),\n    FOREIGN KEY (SupportRepId) REFERENCES Employee (EmployeeId) \n\t
\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Employee\n(\n    EmployeeId INTEGER  NOT NUL
L,\n    LastName NVARCHAR(20)  NOT NULL,\n    FirstName NVARCHAR(20)  NOT NULL,\n    Title NVARCHAR(30),\n
ReportsTo INTEGER,\n    BirthDate DATETIME,\n    HireDate DATETIME,\n    Address NVARCHAR(70),\n    City NV
ARCHAR(40),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVARCHAR(10),\n    Phone NV
ARCHAR(24),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60),\n    CONSTRAINT PK_Employee PRIMARY KEY  (Emplo
yeeId),\n    FOREIGN KEY (ReportsTo) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE N
O ACTION\n)\n\nCREATE INDEX IFK_CustomerSupportRepId ON Customer (SupportRepId)\n\n\n===Additional Context
\n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided context is s
ufficient, please generate a valid SQL query without any explanations for the question. \n2. If the provide
d context is almost sufficient but requires knowledge of a specific string in a particular column, please g
enerate an intermediate SQL query to find the distinct strings in that column. Prepend the query with a com
ment saying intermediate_sql \n3. If the provided context is insufficient, please explain why it can't be g
enerated. \n4. Please use the most relevant table(s). \n5. If the question has been asked and answered befo
re, please repeat the answer exactly as it was given before. \n"}, {'role': 'user', 'content': '  \n    Get
the total number of invoices for each customer\n'}, {'role': 'assistant', 'content': 'SELECT \n    Custome
r.CustomerId,\n    Customer.FirstName,\n    Customer.LastName,\n    COUNT(Invoice.InvoiceId) AS TotalInvoic
es\nFROM \n    Customer\nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId\nGROUP BY \n
Customer.CustomerId\nORDER BY \n    TotalInvoices DESC;'}, {'role': 'user', 'content': '  \n    Find the to
tal number of invoices per country:\n'}, {'role': 'assistant', 'content': 'SELECT \n    BillingCountry, \n
COUNT(*) AS TotalInvoices\nFROM \n    Invoice\nGROUP BY \n    BillingCountry\nORDER BY \n    TotalInvoices
DESC;'}, {'role': 'user', 'content': 'How many records are in table called customer'}, {'role': 'assistan
t', 'content': 'SELECT COUNT(*) FROM Customer;'}, {'role': 'user', 'content': 'How many customers are ther
e'}, {'role': 'assistant', 'content': 'SELECT COUNT(*) FROM Customer;'}, {'role': 'user', 'content': 'what
are the top 5 countries that customers come from?'}, {'role': 'assistant', 'content': 'SELECT Country, COUN
T(*) as CustomerCount\nFROM Customer\nGROUP BY Country\nORDER BY CustomerCount DESC\nLIMIT 5;'}, {'role':
'user', 'content': '  \n    List all albums and their corresponding artist names  \n'}, {'role': 'assistan
t', 'content': 'SELECT \n    Album.Title AS AlbumTitle, \n    Artist.Name AS ArtistName\nFROM \n    Album\n
JOIN \n    Artist ON Album.ArtistId = Artist.ArtistId\nORDER BY \n    Artist.Name, Album.Title;'}, {'role':
'user', 'content': 'Show me a list of tables in the SQLite database'}, {'role': 'assistant', 'content': "SE
LECT name FROM sqlite_master WHERE type='table';"}, {'role': 'user', 'content': '  \n    Find all tracks wi
th a name containing "What" (case-insensitive)\n'}, {'role': 'assistant', 'content': "SELECT TrackId, Name
\nFROM Track\nWHERE Name LIKE '%What%' COLLATE NOCASE;"}, {'role': 'user', 'content': '  \n    List all inv
oices with a total exceeding $10:\n'}]
Using model claude-3-5-sonnet-20240620 for 1332.75 tokens (approx)
SELECT
    InvoiceId,
    CustomerId,
    InvoiceDate,
    Total
FROM
    Invoice
WHERE
    Total > 10
ORDER BY
    Total DESC;
SELECT
    InvoiceId,
    CustomerId,
    InvoiceDate,
    Total
FROM
    Invoice
WHERE
    Total > 10
ORDER BY
    Total DESC;
SELECT
    InvoiceId,
    CustomerId,
    InvoiceDate,
    Total
FROM
    Invoice
WHERE
    Total > 10
ORDER BY
    Total DESC;
    InvoiceId  CustomerId          InvoiceDate  Total
0         404           6  2013-11-13 00:00:00  25.86
1         299          26  2012-08-05 00:00:00  23.86
2          96          45  2010-02-18 00:00:00  21.86
3         194          46  2011-04-28 00:00:00  21.86
4          89           7  2010-01-18 00:00:00  18.86
..        ...         ...                  ...    ...
59        397          27  2013-10-13 00:00:00  13.86
60        411          44  2013-12-14 00:00:00  13.86
61        311          28  2012-09-28 00:00:00  11.94
62        298          17  2012-07-31 00:00:00  10.91
63        312          34  2012-10-01 00:00:00  10.91

[64 rows x 4 columns]
Using model claude-3-5-sonnet-20240620 for 209.75 tokens (approx)
```


Invoices Exceeding $10
Total Amount ($)
25
20
15
10
Invoice Date
2009
2010
2011
2012
2013
2014
CustomerId
50
40
30
20
10

```
Out[26]: ('SELECT \n    InvoiceId, \n    CustomerId, \n    InvoiceDate, \n    Total\nFROM \n    Invoice\nWHERE \n
         Total > 10\nORDER BY \n    Total DESC;',
              InvoiceId  CustomerId          InvoiceDate  Total
          0         404           6  2013-11-13 00:00:00  25.86
          1         299          26  2012-08-05 00:00:00  23.86
          2          96          45  2010-02-18 00:00:00  21.86
          3         194          46  2011-04-28 00:00:00  21.86
          4          89           7  2010-01-18 00:00:00  18.86
          ..        ...         ...                  ...    ...
          59        397          27  2013-10-13 00:00:00  13.86
          60        411          44  2013-12-14 00:00:00  13.86
          61        311          28  2012-09-28 00:00:00  11.94
          62        298          17  2012-07-31 00:00:00  10.91
          63        312          34  2012-10-01 00:00:00  10.91

          [64 rows x 4 columns],
          Figure({
              'data': [{'customdata': array([[404],
                                             [299],
                                             [ 96],
                                             [194],
                                             [ 89],
                                             [201],
                                             [ 88],
                                             [306],
                                             [313],
                                             [103],
                                             [208],
                                             [193],
                                             [  5],
                                             [ 12],
                                             [ 19],
                                             [ 26],
                                             [ 33],
                                             [ 40],
                                             [ 47],
                                             [ 54],
                                             [ 61],
                                             [ 68],
                                             [ 75],
                                             [ 82],
                                             [110],
                                    [117],
                                    [124],
                                    [131],
                                    [138],
                                    [145],
                                    [152],
                                    [159],
                                    [166],
                                    [173],
                                    [180],
                                    [187],
                                    [215],
                                    [222],
                                    [229],
                                    [236],
                                    [243],
                                    [250],
                                    [257],
                                    [264],
                                    [271],
                                    [278],
                                    [285],
                                    [292],
                                    [320],
                                    [327],
                                    [334],
                                    [341],
                                    [348],
                                    [355],
                                    [362],
                                    [369],
                                    [376],
                                    [383],
                                    [390],
                                    [397],
                                    [411],
                                    [311],
                                    [298],
                                    [312]]),
                  'hovertemplate': ('InvoiceDate=%{x}<br>Total=%{ma' ... '%{marker.color}<extra></extra>'),
                  'legendgroup': '',
                  'marker': {'color': array([ 6, 26, 45, 46,  7, 25, 57,  5, 43, 24,  4, 37, 23,  2, 40, 19,
57, 36,
                                           15, 53, 32, 11, 49, 28,  3, 41, 20, 58, 37, 16, 54, 33, 12, 50,
29,  8,
                                           42, 21, 59, 38, 17, 55, 34, 13, 51, 30,  9, 47, 22,  1, 39, 18,
56, 35,
                                           14, 52, 31, 10, 48, 27, 44, 28, 17, 34]),
                          'coloraxis': 'coloraxis',
                          'size': array([25.86, 23.86, 21.86, 21.86, 18.86, 18.86, 17.91, 16.86, 16.86, 1
5.86,
                                          15.86, 14.91, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 1
3.86,
                                          13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 1
3.86,
                                          13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 1
3.86,
                                          13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 1
3.86,
                                          13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 1
3.86,
                                          13.86, 11.94, 10.91, 10.91]),
                          'sizemode': 'area',
                          'sizeref': 0.06465,
                          'symbol': 'circle'},
               'mode': 'markers',
               'name': '',
               'orientation': 'v',
               'showlegend': False,
               'type': 'scatter',
               'x': array(['2013-11-13 00:00:00', '2012-08-05 00:00:00', '2010-02-18 00:00:00',
                            '2011-04-28 00:00:00', '2010-01-18 00:00:00', '2011-05-29 00:00:00',
                            '2010-01-13 00:00:00', '2012-09-05 00:00:00', '2012-10-06 00:00:00',
                            '2010-03-21 00:00:00', '2011-06-29 00:00:00', '2011-04-23 00:00:00',
                            '2009-01-11 00:00:00', '2009-02-11 00:00:00', '2009-03-14 00:00:00',
                            '2009-04-14 00:00:00', '2009-05-15 00:00:00', '2009-06-15 00:00:00',
                            '2009-07-16 00:00:00', '2009-08-16 00:00:00', '2009-09-16 00:00:00',
                            '2009-10-17 00:00:00', '2009-11-17 00:00:00', '2009-12-18 00:00:00',
                            '2010-04-21 00:00:00', '2010-05-22 00:00:00', '2010-06-22 00:00:00',
                            '2010-07-23 00:00:00', '2010-08-23 00:00:00', '2010-09-23 00:00:00',
                            '2010-10-24 00:00:00', '2010-11-24 00:00:00', '2010-12-25 00:00:00',
                            '2011-01-25 00:00:00', '2011-02-25 00:00:00', '2011-03-28 00:00:00',
                            '2011-07-30 00:00:00', '2011-08-30 00:00:00', '2011-09-30 00:00:00',
                            '2011-10-31 00:00:00', '2011-12-01 00:00:00', '2012-01-01 00:00:00',
                                   '2012-02-01 00:00:00', '2012-03-03 00:00:00', '2012-04-03 00:00:00',
                                   '2012-05-04 00:00:00', '2012-06-04 00:00:00', '2012-07-05 00:00:00',
                                   '2012-11-06 00:00:00', '2012-12-07 00:00:00', '2013-01-07 00:00:00',
                                   '2013-02-07 00:00:00', '2013-03-10 00:00:00', '2013-04-10 00:00:00',
                                   '2013-05-11 00:00:00', '2013-06-11 00:00:00', '2013-07-12 00:00:00',
                                   '2013-08-12 00:00:00', '2013-09-12 00:00:00', '2013-10-13 00:00:00',
                                   '2013-12-14 00:00:00', '2012-09-28 00:00:00', '2012-07-31 00:00:00',
                                   '2012-10-01 00:00:00'], dtype=object),
                      'xaxis': 'x',
                      'y': array([25.86, 23.86, 21.86, 21.86, 18.86, 18.86, 17.91, 16.86, 16.86, 15.86,
                                  15.86, 14.91, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86,
                                  13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86,
                                  13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86,
                                  13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86,
                                  13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86,
                                  13.86, 11.94, 10.91, 10.91]),
                      'yaxis': 'y'}],
             'layout': {'coloraxis': {'colorbar': {'title': {'text': 'CustomerId'}},
                                      'colorscale': [[0.0, '#0d0887'], [0.1111111111111111,
                                                     '#46039f'], [0.2222222222222222,
                                                     '#7201a8'], [0.3333333333333333,
                                                     '#9c179e'], [0.4444444444444444,
                                                     '#bd3786'], [0.5555555555555556,
                                                     '#d8576b'], [0.6666666666666666,
                                                     '#ed7953'], [0.7777777777777778,
                                                     '#fb9f3a'], [0.8888888888888888,
                                                     '#fdca26'], [1.0, '#f0f921']]},
                        'legend': {'itemsizing': 'constant', 'tracegroupgap': 0},
                        'template': '...',
                        'title': {'text': 'Invoices Exceeding $10'},
                        'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'Invoice Date'}},
                        'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'Total Amount ($)'}}}
         }))
```

In [27]:
```
question = """
    Find all invoices since 2010 and the total amount invoiced:
"""

vn.ask(question=question)
```

```
Number of requested results 10 is greater than number of elements in index 9, updating n_results = 9
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
[{'role': 'system', 'content': "You are a SQLite expert. Please help to generate a SQL query to answer the
question. Your response should ONLY be based on the given context and follow the response guidelines and fo
rmat instructions. \n===Tables \nCREATE TABLE Invoice\n(\n    InvoiceId INTEGER  NOT NULL,\n    CustomerId
INTEGER  NOT NULL,\n    InvoiceDate DATETIME  NOT NULL,\n    BillingAddress NVARCHAR(70),\n    BillingCity
NVARCHAR(40),\n    BillingState NVARCHAR(40),\n    BillingCountry NVARCHAR(40),\n    BillingPostalCode NVAR
CHAR(10),\n    Total NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Invoice PRIMARY KEY  (InvoiceId),\n    FOR
EIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n
CREATE TABLE InvoiceLine\n(\n    InvoiceLineId INTEGER  NOT NULL,\n    InvoiceId INTEGER  NOT NULL,\n    Tr
ackId INTEGER  NOT NULL,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    Quantity INTEGER  NOT NULL,\n    CONS
TRAINT PK_InvoiceLine PRIMARY KEY  (InvoiceLineId),\n    FOREIGN KEY (InvoiceId) REFERENCES Invoice (Invoic
eId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (TrackId) REFERENCES Track (TrackId)
\n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceLineInvoiceId ON InvoiceLine (I
nvoiceId)\n\nCREATE INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)\n\nCREATE INDEX IFK_InvoiceLineTrac
kId ON InvoiceLine (TrackId)\n\nCREATE TABLE Customer\n(\n    CustomerId INTEGER  NOT NULL,\n    FirstName
NVARCHAR(40)  NOT NULL,\n    LastName NVARCHAR(20)  NOT NULL,\n    Company NVARCHAR(80),\n    Address NVARC
HAR(70),\n    City NVARCHAR(40),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVARCH
AR(10),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60)  NOT NULL,\n    SupportRepI
d INTEGER,\n    CONSTRAINT PK_Customer PRIMARY KEY  (CustomerId),\n    FOREIGN KEY (SupportRepId) REFERENCE
S Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Employee\n(\n    E
mployeeId INTEGER  NOT NULL,\n    LastName NVARCHAR(20)  NOT NULL,\n    FirstName NVARCHAR(20)  NOT NULL,\n
Title NVARCHAR(30),\n    ReportsTo INTEGER,\n    BirthDate DATETIME,\n    HireDate DATETIME,\n    Address N
VARCHAR(70),\n    City NVARCHAR(40),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NV
ARCHAR(10),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60),\n    CONSTRAINT PK_Emp
loyee PRIMARY KEY  (EmployeeId),\n    FOREIGN KEY (ReportsTo) REFERENCES Employee (EmployeeId) \n\t\tON DEL
ETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Track\n(\n    TrackId INTEGER  NOT NULL,\n    Name NVA
RCHAR(200)  NOT NULL,\n    AlbumId INTEGER,\n    MediaTypeId INTEGER  NOT NULL,\n    GenreId INTEGER,\n
Composer NVARCHAR(220),\n    Milliseconds INTEGER  NOT NULL,\n    Bytes INTEGER,\n    UnitPrice NUMERIC(10,
2)  NOT NULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (TrackId),\n    FOREIGN KEY (AlbumId) REFERENCES Album
(AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (GenreId) REFERENCES Genre (Genre
Id) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (MediaTypeId) REFERENCES MediaType (Med
iaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE PlaylistTrack\n(\n    PlaylistId
INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    CONSTRAINT PK_PlaylistTrack PRIMARY KEY  (PlaylistI
d, TrackId),\n    FOREIGN KEY (PlaylistId) REFERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UP
DATE NO ACTION,\n    FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE N
O ACTION\n)\n\nCREATE TABLE Album\n(\n    AlbumId INTEGER  NOT NULL,\n    Title NVARCHAR(160)  NOT NULL,\n
ArtistId INTEGER  NOT NULL,\n    CONSTRAINT PK_Album PRIMARY KEY  (AlbumId),\n    FOREIGN KEY (ArtistId) RE
FERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n===Additional Context \n\n
In the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided context is suffi
cient, please generate a valid SQL query without any explanations for the question. \n2. If the provided co
ntext is almost sufficient but requires knowledge of a specific string in a particular column, please gener
ate an intermediate SQL query to find the distinct strings in that column. Prepend the query with a comment
saying intermediate_sql \n3. If the provided context is insufficient, please explain why it can't be genera
ted. \n4. Please use the most relevant table(s). \n5. If the question has been asked and answered before, p
lease repeat the answer exactly as it was given before. \n"}, {'role': 'user', 'content': '  \n    List all
invoices with a total exceeding $10:\n'}, {'role': 'assistant', 'content': 'SELECT \n    InvoiceId, \n    C
ustomerId, \n    InvoiceDate, \n    Total\nFROM \n    Invoice\nWHERE \n    Total > 10\nORDER BY \n    Total
DESC;'}, {'role': 'user', 'content': '  \n    Find the total number of invoices per country:\n'}, {'role':
'assistant', 'content': 'SELECT \n    BillingCountry, \n    COUNT(*) AS TotalInvoices\nFROM \n    Invoice\n
GROUP BY \n    BillingCountry\nORDER BY \n    TotalInvoices DESC;'}, {'role': 'user', 'content': '  \n    G
et the total number of invoices for each customer\n'}, {'role': 'assistant', 'content': 'SELECT \n    Custo
mer.CustomerId,\n    Customer.FirstName,\n    Customer.LastName,\n    COUNT(Invoice.InvoiceId) AS TotalInvo
ices\nFROM \n    Customer\nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId\nGROUP BY \n
Customer.CustomerId\nORDER BY \n    TotalInvoices DESC;'}, {'role': 'user', 'content': 'How many records ar
e in table called customer'}, {'role': 'assistant', 'content': 'SELECT COUNT(*) FROM Customer;'}, {'role':
'user', 'content': 'How many customers are there'}, {'role': 'assistant', 'content': 'SELECT COUNT(*) FROM
Customer;'}, {'role': 'user', 'content': 'what are the top 5 countries that customers come from?'}, {'rol
e': 'assistant', 'content': 'SELECT Country, COUNT(*) as CustomerCount\nFROM Customer\nGROUP BY Country\nOR
DER BY CustomerCount DESC\nLIMIT 5;'}, {'role': 'user', 'content': '  \n    List all albums and their corre
sponding artist names  \n'}, {'role': 'assistant', 'content': 'SELECT \n    Album.Title AS AlbumTitle, \n
Artist.Name AS ArtistName\nFROM \n    Album\nJOIN \n    Artist ON Album.ArtistId = Artist.ArtistId\nORDER B
Y \n    Artist.Name, Album.Title;'}, {'role': 'user', 'content': '  \n    Find all tracks with a name conta
ining "What" (case-insensitive)\n'}, {'role': 'assistant', 'content': "SELECT TrackId, Name\nFROM Track\nWH
ERE Name LIKE '%What%' COLLATE NOCASE;"}, {'role': 'user', 'content': 'Show me a list of tables in the SQLi
te database'}, {'role': 'assistant', 'content': "SELECT name FROM sqlite_master WHERE type='table';"}, {'ro
le': 'user', 'content': '  \n    Find all invoices since 2010 and the total amount invoiced:\n'}]
Using model claude-3-5-sonnet-20240620 for 1510.5 tokens (approx)
SELECT
    COUNT(*) AS InvoiceCount,
    SUM(Total) AS TotalAmount
FROM
    Invoice
WHERE
    InvoiceDate >= '2010-01-01';
SELECT
    COUNT(*) AS InvoiceCount,
    SUM(Total) AS TotalAmount
FROM
    Invoice
WHERE
    InvoiceDate >= '2010-01-01';
SELECT
    COUNT(*) AS InvoiceCount,
    SUM(Total) AS TotalAmount
FROM
```

```
    Invoice
WHERE
    InvoiceDate >= '2010-01-01';
   InvoiceCount  TotalAmount
0           329      1879.14
Using model claude-3-5-sonnet-20240620 for 200.0 tokens (approx)
```

Total Amount Invoiced
Invoice Count: 329
$1879

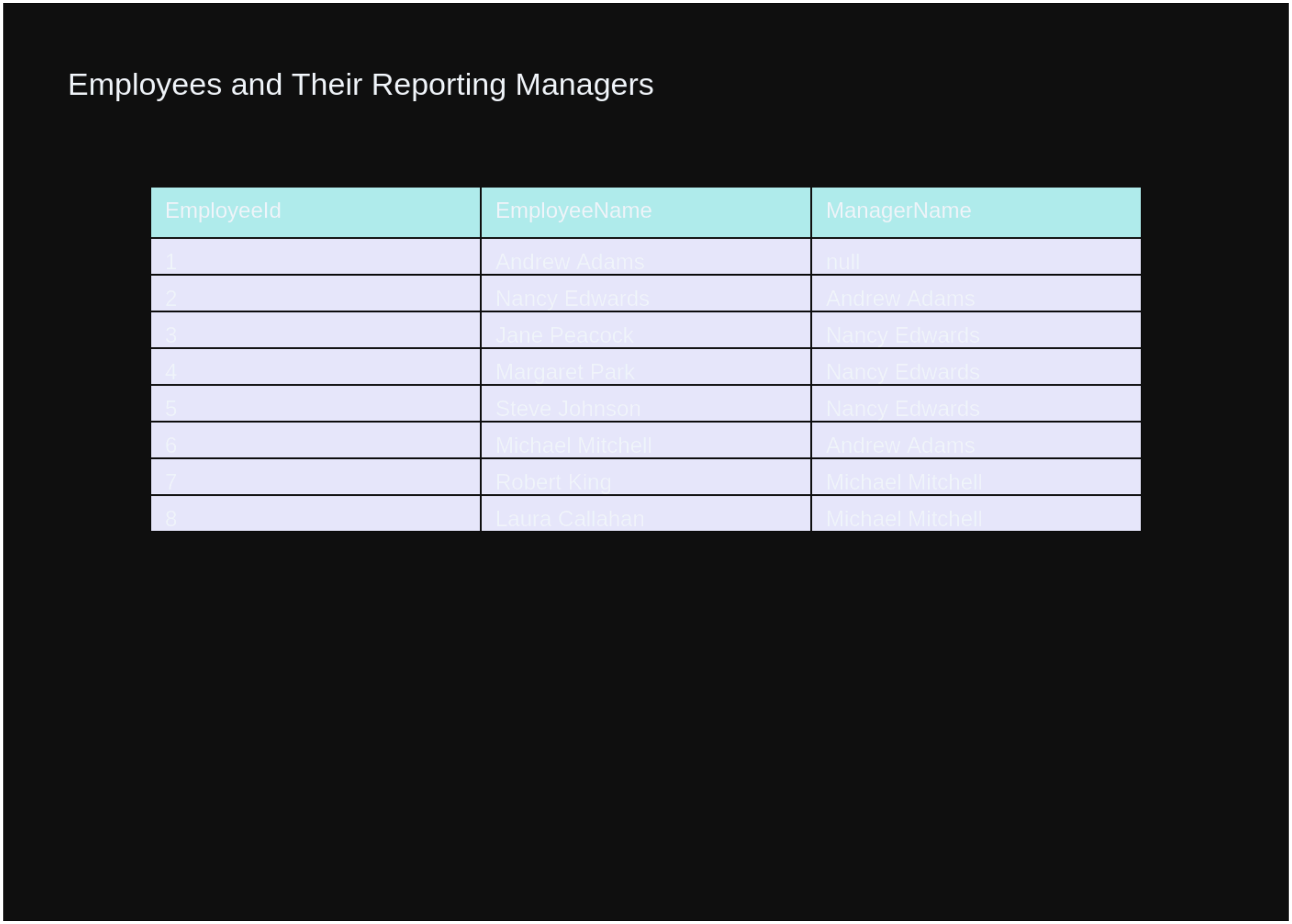
Out[27]:
```
("SELECT \n    COUNT(*) AS InvoiceCount,\n    SUM(Total) AS TotalAmount\nFROM \n    Invoice\nWHERE \n    I
nvoiceDate >= '2010-01-01';",
    InvoiceCount  TotalAmount
 0           329      1879.14,
 Figure({
     'data': [{'mode': 'number',
               'number': {'prefix': '$'},
               'title': {'text': ('Total Amount Invoiced<br><span' ... "ray'>Invoice Count: 329</span>")},
               'type': 'indicator',
               'value': 1879.14}],
     'layout': {'template': '...'}
 }))
```

In [28]:
```python
question = """
    List all employees and their reporting manager's name (if any):
"""

vn.ask(question=question)
```

```
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
[{'role': 'system', 'content': "You are a SQLite expert. Please help to generate a SQL query to answer the
question. Your response should ONLY be based on the given context and follow the response guidelines and fo
rmat instructions. \n===Tables \nCREATE INDEX IFK_EmployeeReportsTo ON Employee (ReportsTo)\n\nCREATE TABLE
Employee\n(\n    EmployeeId INTEGER  NOT NULL,\n    LastName NVARCHAR(20)  NOT NULL,\n    FirstName NVARCHA
R(20)  NOT NULL,\n    Title NVARCHAR(30),\n    ReportsTo INTEGER,\n    BirthDate DATETIME,\n    HireDate DA
TETIME,\n    Address NVARCHAR(70),\n    City NVARCHAR(40),\n    State NVARCHAR(40),\n    Country NVARCHAR(4
0),\n    PostalCode NVARCHAR(10),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(6
0),\n    CONSTRAINT PK_Employee PRIMARY KEY  (EmployeeId),\n    FOREIGN KEY (ReportsTo) REFERENCES Employee
(EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Customer\n(\n    CustomerId I
NTEGER  NOT NULL,\n    FirstName NVARCHAR(40)  NOT NULL,\n    LastName NVARCHAR(20)  NOT NULL,\n    Company
NVARCHAR(80),\n    Address NVARCHAR(70),\n    City NVARCHAR(40),\n    State NVARCHAR(40),\n    Country NVAR
CHAR(40),\n    PostalCode NVARCHAR(10),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(24),\n    Email NVARCHAR
(60)  NOT NULL,\n    SupportRepId INTEGER,\n    CONSTRAINT PK_Customer PRIMARY KEY  (CustomerId),\n    FORE
IGN KEY (SupportRepId) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n
\nCREATE INDEX IFK_CustomerSupportRepId ON Customer (SupportRepId)\n\nCREATE TABLE Invoice\n(\n    InvoiceI
d INTEGER  NOT NULL,\n    CustomerId INTEGER  NOT NULL,\n    InvoiceDate DATETIME  NOT NULL,\n    BillingAd
dress NVARCHAR(70),\n    BillingCity NVARCHAR(40),\n    BillingState NVARCHAR(40),\n    BillingCountry NVAR
CHAR(40),\n    BillingPostalCode NVARCHAR(10),\n    Total NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Invoi
ce PRIMARY KEY  (InvoiceId),\n    FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE
NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE InvoiceLine\n(\n    InvoiceLineId INTEGER  NOT NULL,\n
InvoiceId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    Q
uantity INTEGER  NOT NULL,\n    CONSTRAINT PK_InvoiceLine PRIMARY KEY  (InvoiceLineId),\n    FOREIGN KEY (I
nvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (T
rackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Track\n
(\n    TrackId INTEGER  NOT NULL,\n    Name NVARCHAR(200)  NOT NULL,\n    AlbumId INTEGER,\n    MediaTypeId
INTEGER  NOT NULL,\n    GenreId INTEGER,\n    Composer NVARCHAR(220),\n    Milliseconds INTEGER  NOT NUL
L,\n    Bytes INTEGER,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (Track
Id),\n    FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n
FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIG
N KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n
CREATE INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)\n\nCREATE TABLE Artist\n(\n    ArtistId INTEGER
NOT NULL,\n    Name NVARCHAR(120),\n    CONSTRAINT PK_Artist PRIMARY KEY  (ArtistId)\n)\n\nCREATE TABLE Pla
ylistTrack\n(\n    PlaylistId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    CONSTRAINT PK_Playlis
tTrack PRIMARY KEY  (PlaylistId, TrackId),\n    FOREIGN KEY (PlaylistId) REFERENCES Playlist (PlaylistId)
\n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\t
ON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n===Additional Context \n\nIn the chinook database invoice m
eans order\n\n===Response Guidelines \n1. If the provided context is sufficient, please generate a valid SQ
L query without any explanations for the question. \n2. If the provided context is almost sufficient but re
quires knowledge of a specific string in a particular column, please generate an intermediate SQL query to
find the distinct strings in that column. Prepend the query with a comment saying intermediate_sql \n3. If
the provided context is insufficient, please explain why it can't be generated. \n4. Please use the most re
levant table(s). \n5. If the question has been asked and answered before, please repeat the answer exactly
as it was given before. \n"}, {'role': 'user', 'content': 'what are the top 5 countries that customers come
from?'}, {'role': 'assistant', 'content': 'SELECT Country, COUNT(*) as CustomerCount\nFROM Customer\nGROUP
BY Country\nORDER BY CustomerCount DESC\nLIMIT 5;'}, {'role': 'user', 'content': '  \n    Get the total num
ber of invoices for each customer\n'}, {'role': 'assistant', 'content': 'SELECT \n    Customer.CustomerI
d,\n    Customer.FirstName,\n    Customer.LastName,\n    COUNT(Invoice.InvoiceId) AS TotalInvoices\nFROM \n
Customer\nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId\nGROUP BY \n    Customer.Cust
omerId\nORDER BY \n    TotalInvoices DESC;'}, {'role': 'user', 'content': '  \n    List all albums and thei
r corresponding artist names  \n'}, {'role': 'assistant', 'content': 'SELECT \n    Album.Title AS AlbumTitl
e, \n    Artist.Name AS ArtistName\nFROM \n    Album\nJOIN \n    Artist ON Album.ArtistId = Artist.ArtistId
\nORDER BY \n    Artist.Name, Album.Title;'}, {'role': 'user', 'content': '  \n    List all invoices with a
total exceeding $10:\n'}, {'role': 'assistant', 'content': 'SELECT \n    InvoiceId, \n    CustomerId, \n
InvoiceDate, \n    Total\nFROM \n    Invoice\nWHERE \n    Total > 10\nORDER BY \n    Total DESC;'}, {'rol
e': 'user', 'content': '  \n    Find all invoices since 2010 and the total amount invoiced:\n'}, {'role':
'assistant', 'content': "SELECT \n    COUNT(*) AS InvoiceCount,\n    SUM(Total) AS TotalAmount\nFROM \n
Invoice\nWHERE \n    InvoiceDate >= '2010-01-01';"}, {'role': 'user', 'content': '  \n    Find the total nu
mber of invoices per country:\n'}, {'role': 'assistant', 'content': 'SELECT \n    BillingCountry, \n    COU
NT(*) AS TotalInvoices\nFROM \n    Invoice\nGROUP BY \n    BillingCountry\nORDER BY \n    TotalInvoices DES
C;'}, {'role': 'user', 'content': 'How many records are in table called customer'}, {'role': 'assistant',
'content': 'SELECT COUNT(*) FROM Customer;'}, {'role': 'user', 'content': 'How many customers are there'},
{'role': 'assistant', 'content': 'SELECT COUNT(*) FROM Customer;'}, {'role': 'user', 'content': '  \n    Fi
nd all tracks with a name containing "What" (case-insensitive)\n'}, {'role': 'assistant', 'content': "SELEC
T TrackId, Name\nFROM Track\nWHERE Name LIKE '%What%' COLLATE NOCASE;"}, {'role': 'user', 'content': 'Show
me a list of tables in the SQLite database'}, {'role': 'assistant', 'content': "SELECT name FROM sqlite_mas
ter WHERE type='table';"}, {'role': 'user', 'content': "  \n    List all employees and their reporting mana
ger's name (if any):\n"}]
Using model claude-3-5-sonnet-20240620 for 1524.25 tokens (approx)
SELECT
    e.EmployeeId,
    e.FirstName || ' ' || e.LastName AS EmployeeName,
    m.FirstName || ' ' || m.LastName AS ManagerName
FROM
    Employee e
LEFT JOIN
    Employee m ON e.ReportsTo = m.EmployeeId
ORDER BY
    e.EmployeeId;
SELECT
    e.EmployeeId,
    e.FirstName || ' ' || e.LastName AS EmployeeName,
    m.FirstName || ' ' || m.LastName AS ManagerName
FROM
    Employee e
```

```
LEFT JOIN
    Employee m ON e.ReportsTo = m.EmployeeId
ORDER BY
    e.EmployeeId;
SELECT
    e.EmployeeId,
    e.FirstName || ' ' || e.LastName AS EmployeeName,
    m.FirstName || ' ' || m.LastName AS ManagerName
FROM
    Employee e
LEFT JOIN
    Employee m ON e.ReportsTo = m.EmployeeId
ORDER BY
    e.EmployeeId;
   EmployeeId      EmployeeName       ManagerName
0           1      Andrew Adams              None
1           2     Nancy Edwards      Andrew Adams
2           3      Jane Peacock     Nancy Edwards
3           4     Margaret Park     Nancy Edwards
4           5     Steve Johnson     Nancy Edwards
5           6  Michael Mitchell      Andrew Adams
6           7       Robert King  Michael Mitchell
7           8    Laura Callahan  Michael Mitchell
Using model claude-3-5-sonnet-20240620 for 234.0 tokens (approx)
```



Employees and Their Reporting Managers

EmployeeId	EmployeeName	ManagerName
1	Andrew Adams	null
2	Nancy Edwards	Andrew Adams
3	Jane Peacock	Nancy Edwards
4	Margaret Park	Nancy Edwards
5	Steve Johnson	Nancy Edwards
6	Michael Mitchell	Andrew Adams
7	Robert King	Michael Mitchell
8	Laura Callahan	Michael Mitchell

```
Out[28]: ("SELECT \n    e.EmployeeId,\n    e.FirstName || ' ' || e.LastName AS EmployeeName,\n    m.FirstName || '
         ' || m.LastName AS ManagerName\nFROM \n    Employee e\nLEFT JOIN \n    Employee m ON e.ReportsTo = m.Emplo
         yeeId\nORDER BY \n    e.EmployeeId;",
             EmployeeId      EmployeeName       ManagerName
          0           1      Andrew Adams              None
          1           2     Nancy Edwards      Andrew Adams
          2           3      Jane Peacock     Nancy Edwards
          3           4     Margaret Park     Nancy Edwards
          4           5     Steve Johnson     Nancy Edwards
          5           6  Michael Mitchell      Andrew Adams
          6           7       Robert King  Michael Mitchell
          7           8    Laura Callahan  Michael Mitchell,
          Figure({
              'data': [{'cells': {'align': 'left',
                                  'fill': {'color': 'lavender'},
                                  'values': [[1, 2, 3, 4, 5, 6, 7, 8], ['Andrew Adams',
                                             'Nancy Edwards', 'Jane Peacock', 'Margaret
                                             Park', 'Steve Johnson', 'Michael Mitchell',
                                             'Robert King', 'Laura Callahan'], [None, 'Andrew
                                             Adams', 'Nancy Edwards', 'Nancy Edwards', 'Nancy
                                             Edwards', 'Andrew Adams', 'Michael Mitchell',
                                             'Michael Mitchell']]},
                        'header': {'align': 'left',
                                   'fill': {'color': 'paleturquoise'},
                                   'values': [EmployeeId, EmployeeName, ManagerName]},
                        'type': 'table'}],
              'layout': {'template': '...', 'title': {'text': 'Employees and Their Reporting Managers'}}
          }))
```

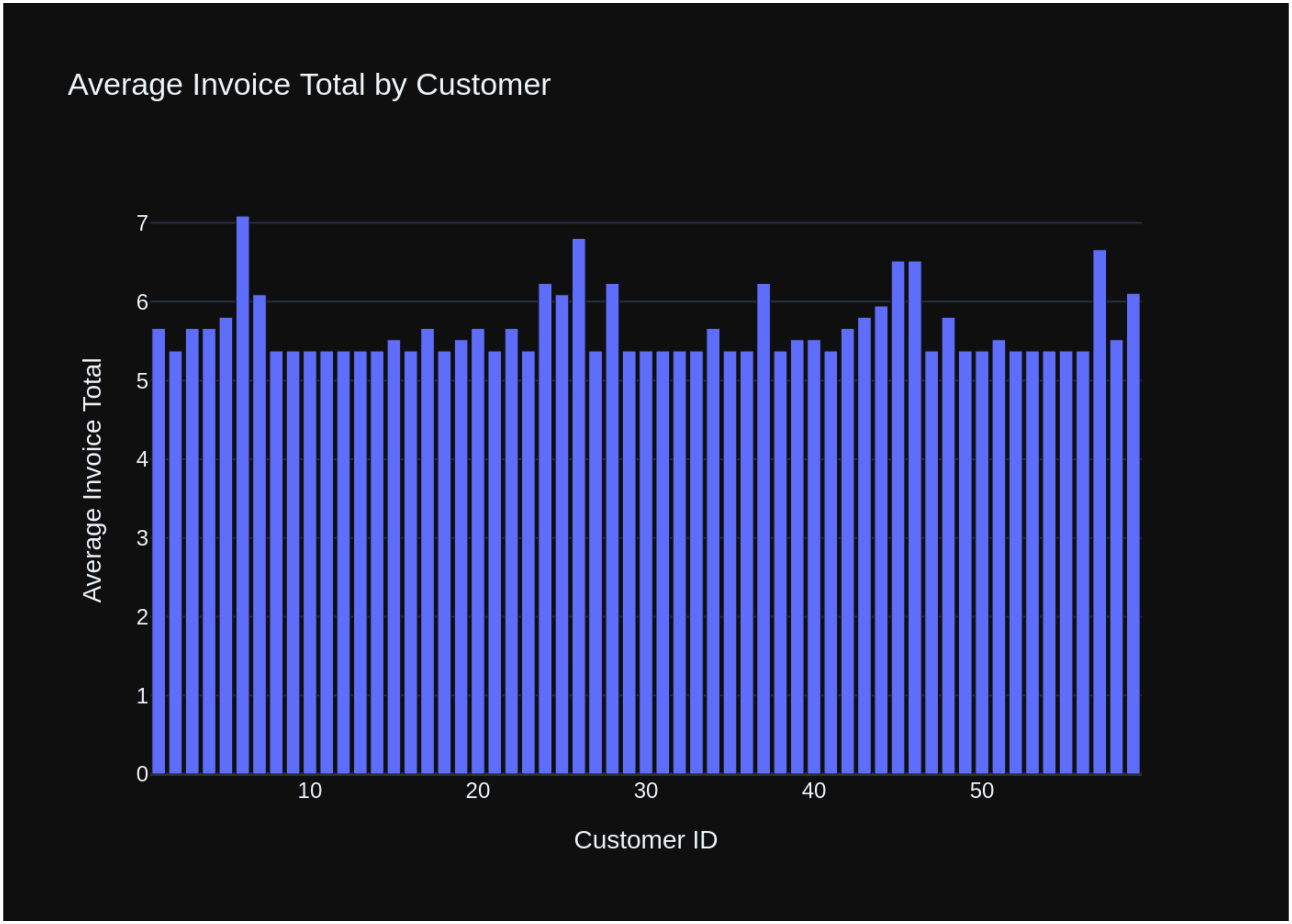
```
In [29]: question = """
             Get the average invoice total for each customer:
         """

         vn.ask(question=question)
```

```
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
[{'role': 'system', 'content': "You are a SQLite expert. Please help to generate a SQL query to answer the
question. Your response should ONLY be based on the given context and follow the response guidelines and fo
rmat instructions. \n===Tables \nCREATE INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)\n\nCREATE INDEX
IFK_InvoiceLineInvoiceId ON InvoiceLine (InvoiceId)\n\nCREATE TABLE Invoice\n(\n    InvoiceId INTEGER  NOT
NULL,\n    CustomerId INTEGER  NOT NULL,\n    InvoiceDate DATETIME  NOT NULL,\n    BillingAddress NVARCHAR
(70),\n    BillingCity NVARCHAR(40),\n    BillingState NVARCHAR(40),\n    BillingCountry NVARCHAR(40),\n
BillingPostalCode NVARCHAR(10),\n    Total NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Invoice PRIMARY KEY
(InvoiceId),\n    FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UP
DATE NO ACTION\n)\n\nCREATE TABLE InvoiceLine\n(\n    InvoiceLineId INTEGER  NOT NULL,\n    InvoiceId INTEG
ER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    Quantity INTEGER
NOT NULL,\n    CONSTRAINT PK_InvoiceLine PRIMARY KEY  (InvoiceLineId),\n    FOREIGN KEY (InvoiceId) REFEREN
CES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (TrackId) REFERENCE
S Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceLineTrackId O
N InvoiceLine (TrackId)\n\nCREATE INDEX IFK_CustomerSupportRepId ON Customer (SupportRepId)\n\nCREATE TABLE
Customer\n(\n    CustomerId INTEGER  NOT NULL,\n    FirstName NVARCHAR(40)  NOT NULL,\n    LastName NVARCHA
R(20)  NOT NULL,\n    Company NVARCHAR(80),\n    Address NVARCHAR(70),\n    City NVARCHAR(40),\n    State N
VARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVARCHAR(10),\n    Phone NVARCHAR(24),\n    Fax NVA
RCHAR(24),\n    Email NVARCHAR(60)  NOT NULL,\n    SupportRepId INTEGER,\n    CONSTRAINT PK_Customer PRIMAR
Y KEY  (CustomerId),\n    FOREIGN KEY (SupportRepId) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO AC
TION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_EmployeeReportsTo ON Employee (ReportsTo)\n\nCREATE TABLE T
rack\n(\n    TrackId INTEGER  NOT NULL,\n    Name NVARCHAR(200)  NOT NULL,\n    AlbumId INTEGER,\n    Media
TypeId INTEGER  NOT NULL,\n    GenreId INTEGER,\n    Composer NVARCHAR(220),\n    Milliseconds INTEGER  NOT
NULL,\n    Bytes INTEGER,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (Tr
ackId),\n    FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTIO
N,\n    FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n
FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION
\n)\n\nCREATE TABLE Employee\n(\n    EmployeeId INTEGER  NOT NULL,\n    LastName NVARCHAR(20)  NOT NULL,\n
FirstName NVARCHAR(20)  NOT NULL,\n    Title NVARCHAR(30),\n    ReportsTo INTEGER,\n    BirthDate DATETIM
E,\n    HireDate DATETIME,\n    Address NVARCHAR(70),\n    City NVARCHAR(40),\n    State NVARCHAR(40),\n
Country NVARCHAR(40),\n    PostalCode NVARCHAR(10),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(24),\n    Em
ail NVARCHAR(60),\n    CONSTRAINT PK_Employee PRIMARY KEY  (EmployeeId),\n    FOREIGN KEY (ReportsTo) REFER
ENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n===Additional Context \n
\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided context is suf
ficient, please generate a valid SQL query without any explanations for the question. \n2. If the provided
context is almost sufficient but requires knowledge of a specific string in a particular column, please gen
erate an intermediate SQL query to find the distinct strings in that column. Prepend the query with a comme
nt saying intermediate_sql \n3. If the provided context is insufficient, please explain why it can't be gen
erated. \n4. Please use the most relevant table(s). \n5. If the question has been asked and answered befor
e, please repeat the answer exactly as it was given before. \n"}, {'role': 'user', 'content': '  \n    Get
the total number of invoices for each customer\n'}, {'role': 'assistant', 'content': 'SELECT \n    Custome
r.CustomerId,\n    Customer.FirstName,\n    Customer.LastName,\n    COUNT(Invoice.InvoiceId) AS TotalInvoic
es\nFROM \n    Customer\nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId\nGROUP BY \n
Customer.CustomerId\nORDER BY \n    TotalInvoices DESC;'}, {'role': 'user', 'content': '  \n    Find all in
voices since 2010 and the total amount invoiced:\n'}, {'role': 'assistant', 'content': "SELECT \n    COUNT
(*) AS InvoiceCount,\n    SUM(Total) AS TotalAmount\nFROM \n    Invoice\nWHERE \n    InvoiceDate >= '2010-0
1-01';"}, {'role': 'user', 'content': '  \n    List all invoices with a total exceeding $10:\n'}, {'role':
'assistant', 'content': 'SELECT \n    InvoiceId, \n    CustomerId, \n    InvoiceDate, \n    Total\nFROM \n
Invoice\nWHERE \n    Total > 10\nORDER BY \n    Total DESC;'}, {'role': 'user', 'content': '  \n    Find th
e total number of invoices per country:\n'}, {'role': 'assistant', 'content': 'SELECT \n    BillingCountry,
\n    COUNT(*) AS TotalInvoices\nFROM \n    Invoice\nGROUP BY \n    BillingCountry\nORDER BY \n    TotalInv
oices DESC;'}, {'role': 'user', 'content': 'How many customers are there'}, {'role': 'assistant', 'conten
t': 'SELECT COUNT(*) FROM Customer;'}, {'role': 'user', 'content': 'How many records are in table called cu
stomer'}, {'role': 'assistant', 'content': 'SELECT COUNT(*) FROM Customer;'}, {'role': 'user', 'content':
'what are the top 5 countries that customers come from?'}, {'role': 'assistant', 'content': 'SELECT Countr
y, COUNT(*) as CustomerCount\nFROM Customer\nGROUP BY Country\nORDER BY CustomerCount DESC\nLIMIT 5;'}, {'r
ole': 'user', 'content': "  \n    List all employees and their reporting manager's name (if any):\n"}, {'ro
le': 'assistant', 'content': "SELECT \n    e.EmployeeId,\n    e.FirstName || ' ' || e.LastName AS EmployeeN
ame,\n    m.FirstName || ' ' || m.LastName AS ManagerName\nFROM \n    Employee e\nLEFT JOIN \n    Employee
m ON e.ReportsTo = m.EmployeeId\nORDER BY \n    e.EmployeeId;"}, {'role': 'user', 'content': '  \n    Find
all tracks with a name containing "What" (case-insensitive)\n'}, {'role': 'assistant', 'content': "SELECT T
rackId, Name\nFROM Track\nWHERE Name LIKE '%What%' COLLATE NOCASE;"}, {'role': 'user', 'content': '  \n
List all albums and their corresponding artist names  \n'}, {'role': 'assistant', 'content': 'SELECT \n
Album.Title AS AlbumTitle, \n    Artist.Name AS ArtistName\nFROM \n    Album\nJOIN \n    Artist ON Album.Ar
tistId = Artist.ArtistId\nORDER BY \n    Artist.Name, Album.Title;'}, {'role': 'user', 'content': '  \n
Get the average invoice total for each customer:\n'}]
Using model claude-3-5-sonnet-20240620 for 1480.75 tokens (approx)
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    AVG(Invoice.Total) AS AverageInvoiceTotal
FROM
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
GROUP BY
    Customer.CustomerId
ORDER BY
    AverageInvoiceTotal DESC;
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    AVG(Invoice.Total) AS AverageInvoiceTotal
```

```
FROM
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
GROUP BY
    Customer.CustomerId
ORDER BY
    AverageInvoiceTotal DESC;
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    AVG(Invoice.Total) AS AverageInvoiceTotal
FROM
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
GROUP BY
    Customer.CustomerId
ORDER BY
    AverageInvoiceTotal DESC;
    CustomerId  FirstName      LastName  AverageInvoiceTotal
0            6     Helena          Holý             7.088571
1           26    Richard    Cunningham             6.802857
2           57       Luis         Rojas             6.660000
3           45   Ladislav        Kovács             6.517143
4           46       Hugh      O'Reilly             6.517143
5           24      Frank       Ralston             6.231429
6           28      Julia       Barnett             6.231429
7           37       Fynn    Zimmermann             6.231429
8           59       Puja    Srivastava             6.106667
9            7     Astrid        Gruber             6.088571
10          25     Victor       Stevens             6.088571
11          44      Terhi    Hämäläinen             5.945714
12           5  František   Wichterlová             5.802857
13          43   Isabelle       Mercier             5.802857
14          48   Johannes  Van der Berg             5.802857
15           1       Luís     Gonçalves             5.660000
16           3   François      Tremblay             5.660000
17           4      Bjørn        Hansen             5.660000
18          17       Jack         Smith             5.660000
19          20        Dan        Miller             5.660000
```

```
20          22    Heather       Leacock             5.660000
21          34       João     Fernandes             5.660000
22          42      Wyatt        Girard             5.660000
23          15   Jennifer      Peterson             5.517143
24          19        Tim         Goyer             5.517143
25          39    Camille       Bernard             5.517143
26          40  Dominique      Lefebvre             5.517143
27          51     Joakim     Johansson             5.517143
28          58      Manoj        Pareek             5.517143
29           2     Leonie        Köhler             5.374286
30           8       Daan       Peeters             5.374286
31           9       Kara       Nielsen             5.374286
32          10    Eduardo       Martins             5.374286
33          11  Alexandre         Rocha             5.374286
34          12    Roberto       Almeida             5.374286
35          13   Fernanda         Ramos             5.374286
36          14       Mark       Philips             5.374286
37          16      Frank        Harris             5.374286
38          18   Michelle        Brooks             5.374286
39          21      Kathy         Chase             5.374286
40          23       John        Gordon             5.374286
41          27    Patrick          Gray             5.374286
42          29     Robert         Brown             5.374286
43          30     Edward       Francis             5.374286
44          31     Martha          Silk             5.374286
45          32      Aaron      Mitchell             5.374286
46          33      Ellie      Sullivan             5.374286
47          35   Madalena       Sampaio             5.374286
48          36     Hannah     Schneider             5.374286
49          38     Niklas       Schröder             5.374286
50          41       Marc        Dubois             5.374286
51          47      Lucas       Mancini             5.374286
52          49  Stanisław        Wójcik             5.374286
53          50    Enrique         Muñoz             5.374286
54          52       Emma         Jones             5.374286
55          53       Phil        Hughes             5.374286
56          54      Steve        Murray             5.374286
57          55       Mark        Taylor             5.374286
58          56      Diego     Gutiérrez             5.374286
Using model claude-3-5-sonnet-20240620 for 256.25 tokens (approx)
```


Average Invoice Total by Customer
Average Invoice Total
0
1
2
3
4
5
6
7
10
20
30
40
50
Customer ID

```
Out[29]: ('SELECT \n    Customer.CustomerId,\n    Customer.FirstName,\n    Customer.LastName,\n    AVG(Invoice.Tota
         l) AS AverageInvoiceTotal\nFROM \n    Customer\nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.C
         ustomerId\nGROUP BY \n    Customer.CustomerId\nORDER BY \n    AverageInvoiceTotal DESC;',
              CustomerId  FirstName     LastName  AverageInvoiceTotal
          0            6     Helena         Holý             7.088571
          1           26    Richard   Cunningham             6.802857
          2           57       Luis        Rojas             6.660000
          3           45   Ladislav       Kovács             6.517143
          4           46       Hugh     O'Reilly             6.517143
          5           24      Frank      Ralston             6.231429
          6           28      Julia      Barnett             6.231429
          7           37       Fynn   Zimmermann             6.231429
          8           59       Puja   Srivastava             6.106667
          9            7     Astrid       Gruber             6.088571
          10          25     Victor      Stevens             6.088571
          11          44      Terhi   Hämäläinen             5.945714
          12           5  František  Wichterlová             5.802857
          13          43   Isabelle      Mercier             5.802857
          14          48   Johannes  Van der Berg            5.802857
          15           1       Luís    Gonçalves             5.660000
          16           3   François     Tremblay             5.660000
          17           4      Bjørn       Hansen             5.660000
          18          17       Jack        Smith             5.660000
          19          20        Dan       Miller             5.660000
          20          22    Heather      Leacock             5.660000
          21          34       João    Fernandes             5.660000
          22          42      Wyatt       Girard             5.660000
          23          15   Jennifer     Peterson             5.517143
          24          19        Tim        Goyer             5.517143
          25          39    Camille      Bernard             5.517143
          26          40  Dominique     Lefebvre             5.517143
          27          51     Joakim    Johansson             5.517143
          28          58      Manoj       Pareek             5.517143
          29           2     Leonie       Köhler             5.374286
          30           8       Daan      Peeters             5.374286
          31           9       Kara      Nielsen             5.374286
          32          10    Eduardo      Martins             5.374286
          33          11  Alexandre        Rocha             5.374286
          34          12    Roberto      Almeida             5.374286
          35          13   Fernanda        Ramos             5.374286
          36          14       Mark      Philips             5.374286
          37          16      Frank       Harris             5.374286
```

```
38          18   Michelle        Brooks             5.374286
39          21      Kathy         Chase             5.374286
40          23       John        Gordon             5.374286
41          27    Patrick          Gray             5.374286
42          29     Robert         Brown             5.374286
43          30     Edward       Francis             5.374286
44          31     Martha          Silk             5.374286
45          32      Aaron      Mitchell             5.374286
46          33      Ellie      Sullivan             5.374286
47          35   Madalena       Sampaio             5.374286
48          36     Hannah     Schneider             5.374286
49          38     Niklas      Schröder             5.374286
50          41       Marc        Dubois             5.374286
51          47      Lucas       Mancini             5.374286
52          49  Stanisław        Wójcik             5.374286
53          50    Enrique         Muñoz             5.374286
54          52       Emma         Jones             5.374286
55          53       Phil        Hughes             5.374286
56          54      Steve        Murray             5.374286
57          55       Mark        Taylor             5.374286
58          56      Diego     Gutiérrez             5.374286,
Figure({
    'data': [{'alignmentgroup': 'True',
              'customdata': array([['Helena', 'Holý'],
                                   ['Richard', 'Cunningham'],
                                   ['Luis', 'Rojas'],
                                   ['Ladislav', 'Kovács'],
                                   ['Hugh', "O'Reilly"],
                                   ['Frank', 'Ralston'],
                                   ['Julia', 'Barnett'],
                                   ['Fynn', 'Zimmermann'],
                                   ['Puja', 'Srivastava'],
                                   ['Astrid', 'Gruber'],
                                   ['Victor', 'Stevens'],
                                   ['Terhi', 'Hämäläinen'],
                                   ['František', 'Wichterlová'],
                                   ['Isabelle', 'Mercier'],
                                   ['Johannes', 'Van der Berg'],
                                   ['Luís', 'Gonçalves'],
                                   ['François', 'Tremblay'],
                                   ['Bjørn', 'Hansen'],
                                   ['Jack', 'Smith'],
                                   ['Dan', 'Miller'],
                                   ['Heather', 'Leacock'],
                                   ['João', 'Fernandes'],
                                   ['Wyatt', 'Girard'],
                                   ['Jennifer', 'Peterson'],
                                   ['Tim', 'Goyer'],
                                   ['Camille', 'Bernard'],
                                   ['Dominique', 'Lefebvre'],
                                   ['Joakim', 'Johansson'],
                                   ['Manoj', 'Pareek'],
                                   ['Leonie', 'Köhler'],
                                   ['Daan', 'Peeters'],
                                   ['Kara', 'Nielsen'],
                                   ['Eduardo', 'Martins'],
                                   ['Alexandre', 'Rocha'],
                                   ['Roberto', 'Almeida'],
                                   ['Fernanda', 'Ramos'],
                                   ['Mark', 'Philips'],
                                   ['Frank', 'Harris'],
                                   ['Michelle', 'Brooks'],
                                   ['Kathy', 'Chase'],
                                   ['John', 'Gordon'],
                                   ['Patrick', 'Gray'],
                                   ['Robert', 'Brown'],
                                   ['Edward', 'Francis'],
                                   ['Martha', 'Silk'],
                                   ['Aaron', 'Mitchell'],
                                   ['Ellie', 'Sullivan'],
                                   ['Madalena', 'Sampaio'],
                                   ['Hannah', 'Schneider'],
                                   ['Niklas', 'Schröder'],
                                   ['Marc', 'Dubois'],
                                   ['Lucas', 'Mancini'],
                                   ['Stanisław', 'Wójcik'],
                                   ['Enrique', 'Muñoz'],
                                   ['Emma', 'Jones'],
                                   ['Phil', 'Hughes'],
                                   ['Steve', 'Murray'],
                                   ['Mark', 'Taylor'],
                                   ['Diego', 'Gutiérrez']], dtype=object),
              'hovertemplate': ('CustomerId=%{x}<br>Average Inv' ... '{customdata[1]}<extra></extra>'),
              'legendgroup': '',
              'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
              'name': '',
              'offsetgroup': '',
              'orientation': 'v',
              'showlegend': False,
              'textposition': 'auto',
              'type': 'bar',
              'x': array([ 6, 26, 57, 45, 46, 24, 28, 37, 59,  7, 25, 44,  5, 43, 48,  1,  3,  4,
                          17, 20, 22, 34, 42, 15, 19, 39, 40, 51, 58,  2,  8,  9, 10, 11, 12, 13,
                          14, 16, 18, 21, 23, 27, 29, 30, 31, 32, 33, 35, 36, 38, 41, 47, 49, 50,
                          52, 53, 54, 55, 56]),
              'xaxis': 'x',
              'y': array([7.08857143, 6.80285714, 6.66      , 6.51714286, 6.51714286, 6.23142857,
                          6.23142857, 6.23142857, 6.10666667, 6.08857143, 6.08857143, 5.94571429,
                          5.80285714, 5.80285714, 5.80285714, 5.66      , 5.66      , 5.66      ,
                          5.66      , 5.66      , 5.66      , 5.66      , 5.66      , 5.51714286,
                          5.51714286, 5.51714286, 5.51714286, 5.51714286, 5.51714286, 5.37428571,
                          5.37428571, 5.37428571, 5.37428571, 5.37428571, 5.37428571, 5.37428571,
                          5.37428571, 5.37428571, 5.37428571, 5.37428571, 5.37428571, 5.37428571,
                          5.37428571, 5.37428571, 5.37428571, 5.37428571, 5.37428571, 5.37428571,
                          5.37428571, 5.37428571, 5.37428571, 5.37428571, 5.37428571, 5.37428571,
                          5.37428571, 5.37428571, 5.37428571, 5.37428571, 5.37428571]),
              'yaxis': 'y'}],
   'layout': {'barmode': 'relative',
               'legend': {'tracegroupgap': 0},
               'template': '...',
               'title': {'text': 'Average Invoice Total by Customer'},
               'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'Customer ID'}},
               'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'Average Invoice Tota
l'}}}
}))
```

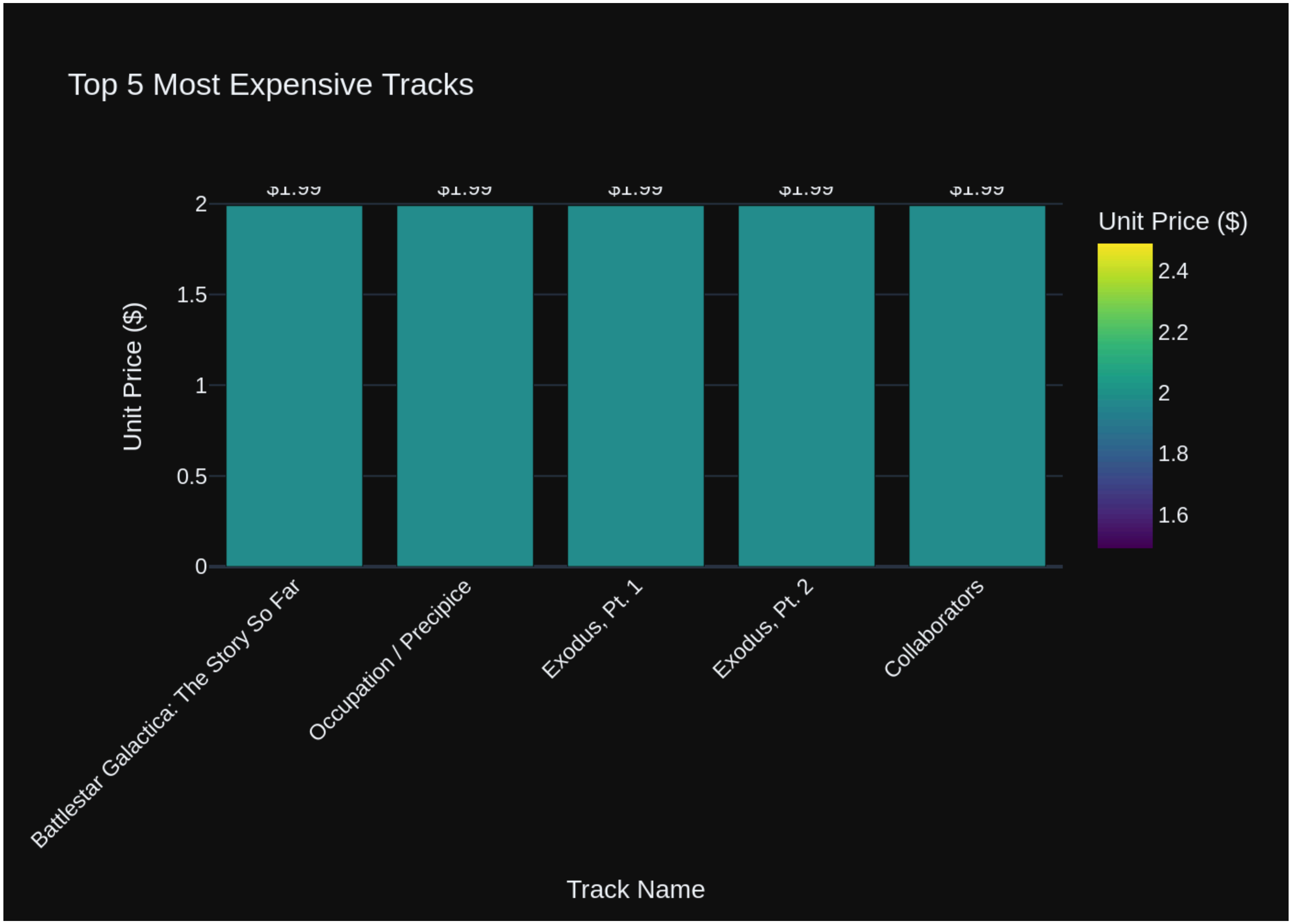
In [30]:

```
question = """
    Find the top 5 most expensive tracks (based on unit price):
"""

vn.ask(question=question)
```

```
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
[{'role': 'system', 'content': "You are a SQLite expert. Please help to generate a SQL query to answer the
question. Your response should ONLY be based on the given context and follow the response guidelines and fo
rmat instructions. \n===Tables \nCREATE TABLE Track\n(\n    TrackId INTEGER  NOT NULL,\n    Name NVARCHAR(2
00)  NOT NULL,\n    AlbumId INTEGER,\n    MediaTypeId INTEGER  NOT NULL,\n    GenreId INTEGER,\n    Compose
r NVARCHAR(220),\n    Milliseconds INTEGER  NOT NULL,\n    Bytes INTEGER,\n    UnitPrice NUMERIC(10,2)  NOT
NULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (TrackId),\n    FOREIGN KEY (AlbumId) REFERENCES Album (AlbumI
d) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n
\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaType
Id) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_TrackAlbumId ON Track (AlbumId)\n
\nCREATE INDEX IFK_TrackGenreId ON Track (GenreId)\n\nCREATE INDEX IFK_PlaylistTrackTrackId ON PlaylistTrac
k (TrackId)\n\nCREATE INDEX IFK_InvoiceLineTrackId ON InvoiceLine (TrackId)\n\nCREATE INDEX IFK_TrackMediaT
ypeId ON Track (MediaTypeId)\n\nCREATE TABLE InvoiceLine\n(\n    InvoiceLineId INTEGER  NOT NULL,\n    Invo
iceId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    Quant
ity INTEGER  NOT NULL,\n    CONSTRAINT PK_InvoiceLine PRIMARY KEY  (InvoiceLineId),\n    FOREIGN KEY (Invoi
ceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (Track
Id) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE PlaylistTra
ck\n(\n    PlaylistId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    CONSTRAINT PK_PlaylistTrack P
RIMARY KEY  (PlaylistId, TrackId),\n    FOREIGN KEY (PlaylistId) REFERENCES Playlist (PlaylistId) \n\t\tON
DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE
NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_AlbumArtistId ON Album (ArtistId)\n\nCREATE TABLE Albu
m\n(\n    AlbumId INTEGER  NOT NULL,\n    Title NVARCHAR(160)  NOT NULL,\n    ArtistId INTEGER  NOT NULL,\n
CONSTRAINT PK_Album PRIMARY KEY  (AlbumId),\n    FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\t
ON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n\n===Additional Context \n\nIn the chinook database invoice m
eans order\n\n===Response Guidelines \n1. If the provided context is sufficient, please generate a valid SQ
L query without any explanations for the question. \n2. If the provided context is almost sufficient but re
quires knowledge of a specific string in a particular column, please generate an intermediate SQL query to
find the distinct strings in that column. Prepend the query with a comment saying intermediate_sql \n3. If
the provided context is insufficient, please explain why it can't be generated. \n4. Please use the most re
levant table(s). \n5. If the question has been asked and answered before, please repeat the answer exactly
as it was given before. \n"}, {'role': 'user', 'content': '  \n    Find all tracks with a name containing
"What" (case-insensitive)\n'}, {'role': 'assistant', 'content': "SELECT TrackId, Name\nFROM Track\nWHERE Na
me LIKE '%What%' COLLATE NOCASE;"}, {'role': 'user', 'content': '  \n    List all albums and their correspo
nding artist names  \n'}, {'role': 'assistant', 'content': 'SELECT \n    Album.Title AS AlbumTitle, \n    A
rtist.Name AS ArtistName\nFROM \n    Album\nJOIN \n    Artist ON Album.ArtistId = Artist.ArtistId\nORDER BY
\n    Artist.Name, Album.Title;'}, {'role': 'user', 'content': '  \n    List all invoices with a total exce
eding $10:\n'}, {'role': 'assistant', 'content': 'SELECT \n    InvoiceId, \n    CustomerId, \n    InvoiceDa
te, \n    Total\nFROM \n    Invoice\nWHERE \n    Total > 10\nORDER BY \n    Total DESC;'}, {'role': 'user',
'content': '  \n    Get the average invoice total for each customer:\n'}, {'role': 'assistant', 'content':
'SELECT \n    Customer.CustomerId,\n    Customer.FirstName,\n    Customer.LastName,\n    AVG(Invoice.Total)
AS AverageInvoiceTotal\nFROM \n    Customer\nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.Custo
merId\nGROUP BY \n    Customer.CustomerId\nORDER BY \n    AverageInvoiceTotal DESC;'}, {'role': 'user', 'co
ntent': '  \n    Find all invoices since 2010 and the total amount invoiced:\n'}, {'role': 'assistant', 'co
ntent': "SELECT \n    COUNT(*) AS InvoiceCount,\n    SUM(Total) AS TotalAmount\nFROM \n    Invoice\nWHERE 
\n    InvoiceDate >= '2010-01-01';"}, {'role': 'user', 'content': 'what are the top 5 countries that custom
ers come from?'}, {'role': 'assistant', 'content': 'SELECT Country, COUNT(*) as CustomerCount\nFROM Custome
r\nGROUP BY Country\nORDER BY CustomerCount DESC\nLIMIT 5;'}, {'role': 'user', 'content': '  \n    Find the 
total number of invoices per country:\n'}, {'role': 'assistant', 'content': 'SELECT \n    BillingCountry, 
\n    COUNT(*) AS TotalInvoices\nFROM \n    Invoice\nGROUP BY \n    BillingCountry\nORDER BY \n    TotalInv
oices DESC;'}, {'role': 'user', 'content': '  \n    Get the total number of invoices for each customer\n'}, 
{'role': 'assistant', 'content': 'SELECT \n    Customer.CustomerId,\n    Customer.FirstName,\n    Customer.
LastName,\n    COUNT(Invoice.InvoiceId) AS TotalInvoices\nFROM \n    Customer\nLEFT JOIN \n    Invoice ON C
ustomer.CustomerId = Invoice.CustomerId\nGROUP BY \n    Customer.CustomerId\nORDER BY \n    TotalInvoices D
ESC;'}, {'role': 'user', 'content': 'Show me a list of tables in the SQLite database'}, {'role': 'assistan
t', 'content': "SELECT name FROM sqlite_master WHERE type='table';"}, {'role': 'user', 'content': 'How many 
customers are there'}, {'role': 'assistant', 'content': 'SELECT COUNT(*) FROM Customer;'}, {'role': 'user', 
'content': '  \n    Find the top 5 most expensive tracks (based on unit price):\n'}]
Using model claude-3-5-sonnet-20240620 for 1255.75 tokens (approx)
SELECT 
    TrackId,
    Name,
    UnitPrice
FROM 
    Track
ORDER BY 
    UnitPrice DESC
LIMIT 5;
SELECT 
    TrackId,
    Name,
    UnitPrice
FROM 
    Track
ORDER BY 
    UnitPrice DESC
LIMIT 5;
SELECT 
    TrackId,
    Name,
    UnitPrice
FROM 
    Track
ORDER BY 
    UnitPrice DESC
LIMIT 5;
```

```
   TrackId                                Name  UnitPrice
0     2819  Battlestar Galactica: The Story So Far       1.99
1     2820                 Occupation / Precipice       1.99
2     2821                         Exodus, Pt. 1       1.99
3     2822                         Exodus, Pt. 2       1.99
4     2823                          Collaborators       1.99
Using model claude-3-5-sonnet-20240620 for 197.0 tokens (approx)
```


Top 5 Most Expensive Tracks
$1.99
$1.99
$1.99
$1.99
$1.99
Unit Price ($)
0
0.5
1
1.5
2
Battlestar Galactica: The Story So Far
Occupation / Precipice
Exodus, Pt. 1
Exodus, Pt. 2
Collaborators
Track Name
Unit Price ($)
1.6
1.8
2
2.2
2.4

```
Out[30]: ('SELECT \n    TrackId,\n    Name,\n    UnitPrice\nFROM \n    Track\nORDER BY \n    UnitPrice DESC\nLIMIT
         5;',
             TrackId                                      Name  UnitPrice
          0     2819  Battlestar Galactica: The Story So Far       1.99
          1     2820                  Occupation / Precipice       1.99
          2     2821                           Exodus, Pt. 1       1.99
          3     2822                           Exodus, Pt. 2       1.99
          4     2823                           Collaborators       1.99,
          Figure({
              'data': [{'alignmentgroup': 'True',
                        'hovertemplate': 'Track Name=%{x}<br>Unit Price ($)=%{marker.color}<extra></extra>',
                        'legendgroup': '',
                        'marker': {'color': array([1.99, 1.99, 1.99, 1.99, 1.99]),
                                   'coloraxis': 'coloraxis',
                                   'pattern': {'shape': ''}},
                        'name': '',
                        'offsetgroup': '',
                        'orientation': 'v',
                        'showlegend': False,
                        'text': array([1.99, 1.99, 1.99, 1.99, 1.99]),
                        'textposition': 'outside',
                        'texttemplate': '$%{text:.2f}',
                        'type': 'bar',
                        'x': array(['Battlestar Galactica: The Story So Far', 'Occupation / Precipice',
                                    'Exodus, Pt. 1', 'Exodus, Pt. 2', 'Collaborators'], dtype=object),
                        'xaxis': 'x',
                        'y': array([1.99, 1.99, 1.99, 1.99, 1.99]),
                        'yaxis': 'y'}],
              'layout': {'barmode': 'relative',
                         'coloraxis': {'colorbar': {'title': {'text': 'Unit Price ($)'}},
                                       'colorscale': [[0.0, '#440154'], [0.1111111111111111,
                                                      '#482878'], [0.2222222222222222,
                                                      '#3e4989'], [0.3333333333333333,
                                                      '#31688e'], [0.4444444444444444,
                                                      '#26828e'], [0.5555555555555556,
                                                      '#1f9e89'], [0.6666666666666666,
                                                      '#35b779'], [0.7777777777777778,
                                                      '#6ece58'], [0.8888888888888888,
                                                      '#b5de2b'], [1.0, '#fde725']]},
                         'legend': {'tracegroupgap': 0},
                         'template': '...',
                         'title': {'text': 'Top 5 Most Expensive Tracks'},
                         'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'tickangle': -45, 'title': {'text': 'Track
         Name'}},
                         'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'Unit Price ($)'}}}
          }))
```

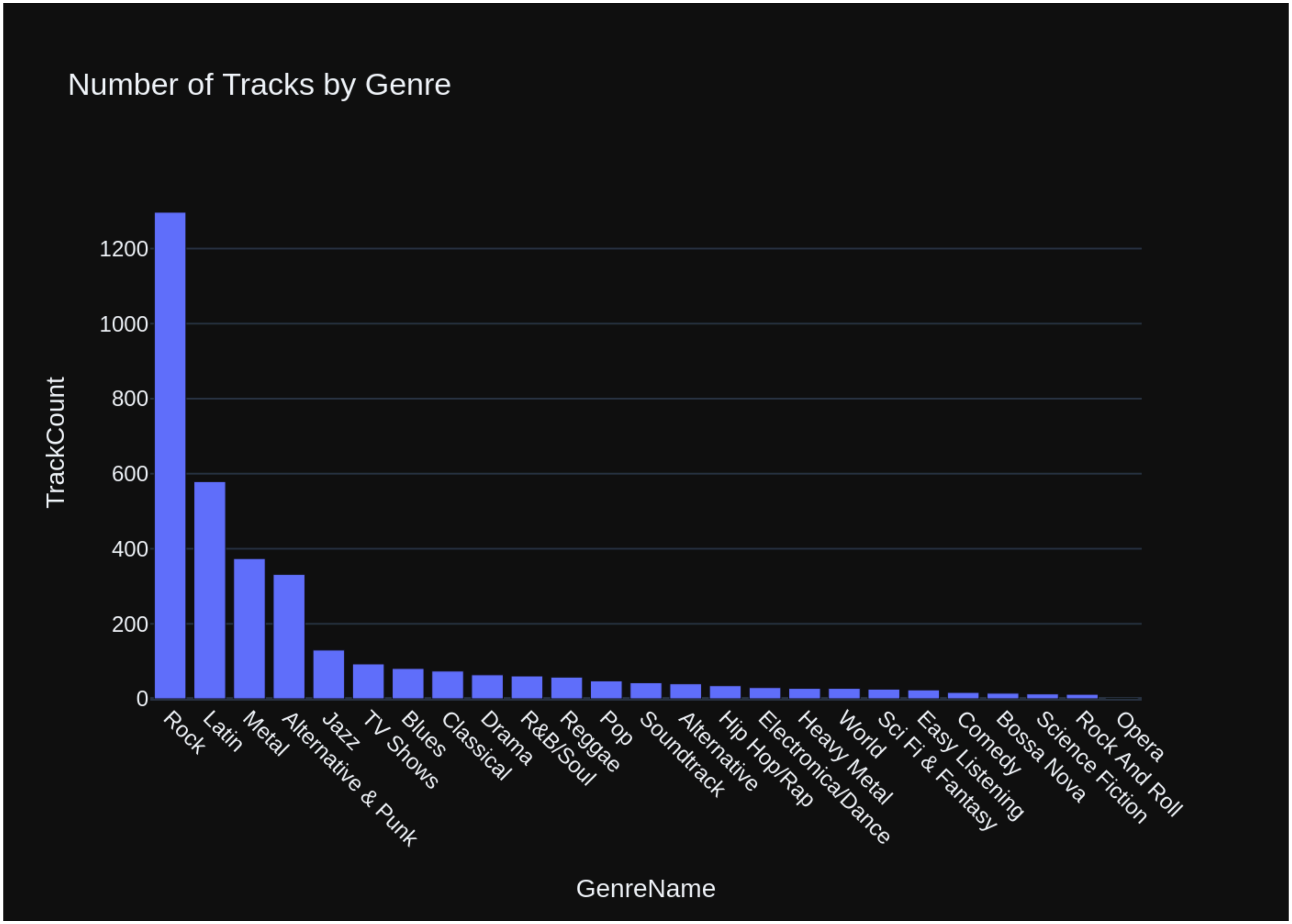
In [31]:

```
question = """
    List all genres and the number of tracks in each genre:
"""

vn.ask(question=question)
```

```
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
[{'role': 'system', 'content': "You are a SQLite expert. Please help to generate a SQL query to answer the
question. Your response should ONLY be based on the given context and follow the response guidelines and fo
rmat instructions. \n===Tables \nCREATE TABLE Track\n(\n    TrackId INTEGER  NOT NULL,\n    Name NVARCHAR(2
00)  NOT NULL,\n    AlbumId INTEGER,\n    MediaTypeId INTEGER  NOT NULL,\n    GenreId INTEGER,\n    Compose
r NVARCHAR(220),\n    Milliseconds INTEGER  NOT NULL,\n    Bytes INTEGER,\n    UnitPrice NUMERIC(10,2)  NOT
NULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (TrackId),\n    FOREIGN KEY (AlbumId) REFERENCES Album (AlbumI
d) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n
\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaType
Id) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_TrackGenreId ON Track (GenreId)\n
\nCREATE TABLE Genre\n(\n    GenreId INTEGER  NOT NULL,\n    Name NVARCHAR(120),\n    CONSTRAINT PK_Genre P
RIMARY KEY  (GenreId)\n)\n\nCREATE INDEX IFK_PlaylistTrackTrackId ON PlaylistTrack (TrackId)\n\nCREATE INDE
X IFK_TrackAlbumId ON Track (AlbumId)\n\nCREATE INDEX IFK_TrackMediaTypeId ON Track (MediaTypeId)\n\nCREATE
TABLE Album\n(\n    AlbumId INTEGER  NOT NULL,\n    Title NVARCHAR(160)  NOT NULL,\n    ArtistId INTEGER  N
OT NULL,\n    CONSTRAINT PK_Album PRIMARY KEY  (AlbumId),\n    FOREIGN KEY (ArtistId) REFERENCES Artist (Ar
tistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE PlaylistTrack\n(\n    PlaylistId I
NTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    CONSTRAINT PK_PlaylistTrack PRIMARY KEY  (PlaylistI
d, TrackId),\n    FOREIGN KEY (PlaylistId) REFERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UP
DATE NO ACTION,\n    FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE N
O ACTION\n)\n\nCREATE INDEX IFK_AlbumArtistId ON Album (ArtistId)\n\nCREATE TABLE Playlist\n(\n    Playlist
Id INTEGER  NOT NULL,\n    Name NVARCHAR(120),\n    CONSTRAINT PK_Playlist PRIMARY KEY  (PlaylistId)\n)\n\n
\n===Additional Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If t
he provided context is sufficient, please generate a valid SQL query without any explanations for the quest
ion. \n2. If the provided context is almost sufficient but requires knowledge of a specific string in a par
ticular column, please generate an intermediate SQL query to find the distinct strings in that column. Prep
end the query with a comment saying intermediate_sql \n3. If the provided context is insufficient, please e
xplain why it can't be generated. \n4. Please use the most relevant table(s). \n5. If the question has been
asked and answered before, please repeat the answer exactly as it was given before. \n"}, {'role': 'user',
'content': '  \n    Find the top 5 most expensive tracks (based on unit price):\n'}, {'role': 'assistant',
'content': 'SELECT \n    TrackId,\n    Name,\n    UnitPrice\nFROM \n    Track\nORDER BY \n    UnitPrice DES
C\nLIMIT 5;'}, {'role': 'user', 'content': '  \n    List all albums and their corresponding artist names
\n'}, {'role': 'assistant', 'content': 'SELECT \n    Album.Title AS AlbumTitle, \n    Artist.Name AS Artist
Name\nFROM \n    Album\nJOIN \n    Artist ON Album.ArtistId = Artist.ArtistId\nORDER BY \n    Artist.Name,
Album.Title;'}, {'role': 'user', 'content': '  \n    Find all tracks with a name containing "What" (case-in
sensitive)\n'}, {'role': 'assistant', 'content': "SELECT TrackId, Name\nFROM Track\nWHERE Name LIKE '%Wha
t%' COLLATE NOCASE;"}, {'role': 'user', 'content': 'what are the top 5 countries that customers come fro
m?'}, {'role': 'assistant', 'content': 'SELECT Country, COUNT(*) as CustomerCount\nFROM Customer\nGROUP BY
Country\nORDER BY CustomerCount DESC\nLIMIT 5;'}, {'role': 'user', 'content': '  \n    Find all invoices si
nce 2010 and the total amount invoiced:\n'}, {'role': 'assistant', 'content': "SELECT \n    COUNT(*) AS Inv
oiceCount,\n    SUM(Total) AS TotalAmount\nFROM \n    Invoice\nWHERE \n    InvoiceDate >= '2010-01-01';"},
{'role': 'user', 'content': '  \n    Find the total number of invoices per country:\n'}, {'role': 'assistan
t', 'content': 'SELECT \n    BillingCountry, \n    COUNT(*) AS TotalInvoices\nFROM \n    Invoice\nGROUP BY
\n    BillingCountry\nORDER BY \n    TotalInvoices DESC;'}, {'role': 'user', 'content': '  \n    List all i
nvoices with a total exceeding $10:\n'}, {'role': 'assistant', 'content': 'SELECT \n    InvoiceId, \n    Cu
stomerId, \n    InvoiceDate, \n    Total\nFROM \n    Invoice\nWHERE \n    Total > 10\nORDER BY \n    Total 
DESC;'}, {'role': 'user', 'content': 'Show me a list of tables in the SQLite database'}, {'role': 'assistan
t', 'content': "SELECT name FROM sqlite_master WHERE type='table';"}, {'role': 'user', 'content': '  \n    
Get the total number of invoices for each customer\n'}, {'role': 'assistant', 'content': 'SELECT \n    Cust
omer.CustomerId,\n    Customer.FirstName,\n    Customer.LastName,\n    COUNT(Invoice.InvoiceId) AS TotalInv
oices\nFROM \n    Customer\nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId\nGROUP BY 
\n    Customer.CustomerId\nORDER BY \n    TotalInvoices DESC;'}, {'role': 'user', 'content': 'How many reco
rds are in table called customer'}, {'role': 'assistant', 'content': 'SELECT COUNT(*) FROM Customer;'}, {'r
ole': 'user', 'content': '  \n    List all genres and the number of tracks in each genre:\n'}]
Using model claude-3-5-sonnet-20240620 for 1150.25 tokens (approx)
SELECT
    Genre.Name AS GenreName,
    COUNT(Track.TrackId) AS TrackCount
FROM
    Genre
LEFT JOIN
    Track ON Genre.GenreId = Track.GenreId
GROUP BY
    Genre.GenreId
ORDER BY
    TrackCount DESC;
SELECT
    Genre.Name AS GenreName,
    COUNT(Track.TrackId) AS TrackCount
FROM
    Genre
LEFT JOIN
    Track ON Genre.GenreId = Track.GenreId
GROUP BY
    Genre.GenreId
ORDER BY
    TrackCount DESC;
SELECT
    Genre.Name AS GenreName,
    COUNT(Track.TrackId) AS TrackCount
FROM
    Genre
LEFT JOIN
    Track ON Genre.GenreId = Track.GenreId
GROUP BY
    Genre.GenreId
```

```
ORDER BY
    TrackCount DESC;
             GenreName  TrackCount
0                 Rock        1297
1                Latin         579
2                Metal         374
3   Alternative & Punk         332
4                 Jazz         130
5             TV Shows          93
6                Blues          81
7            Classical          74
8                Drama          64
9             R&B/Soul          61
10              Reggae          58
11                 Pop          48
12          Soundtrack          43
13         Alternative          40
14         Hip Hop/Rap          35
15   Electronica/Dance          30
16         Heavy Metal          28
17               World          28
18    Sci Fi & Fantasy          26
19      Easy Listening          24
20              Comedy          17
21          Bossa Nova          15
22     Science Fiction          13
23       Rock And Roll          12
24               Opera           1
Using model claude-3-5-sonnet-20240620 for 217.25 tokens (approx)
```


Number of Tracks by Genre
TrackCount
0
200
400
600
800
1000
1200
GenreName
Rock
Latin
Metal
Alternative & Punk
Jazz
TV Shows
Blues
Classical
Drama
R&B/Soul
Reggae
Pop
Soundtrack
Alternative
Hip Hop/Rap
Electronica/Dance
Heavy Metal
World
Sci Fi & Fantasy
Easy Listening
Comedy
Bossa Nova
Science Fiction
Rock And Roll
Opera

```
Out[31]: ('SELECT \n    Genre.Name AS GenreName,\n    COUNT(Track.TrackId) AS TrackCount\nFROM \n    Genre\nLEFT JO
         IN \n    Track ON Genre.GenreId = Track.GenreId\nGROUP BY \n    Genre.GenreId\nORDER BY \n    TrackCount D
         ESC;',
                       GenreName  TrackCount
          0                 Rock        1297
          1                Latin         579
          2                Metal         374
          3   Alternative & Punk         332
          4                 Jazz         130
          5             TV Shows          93
          6                Blues          81
          7            Classical          74
          8                Drama          64
          9             R&B/Soul          61
          10              Reggae          58
          11                 Pop          48
          12          Soundtrack          43
          13         Alternative          40
          14         Hip Hop/Rap          35
          15   Electronica/Dance          30
          16         Heavy Metal          28
          17               World          28
          18    Sci Fi & Fantasy          26
          19      Easy Listening          24
          20              Comedy          17
          21          Bossa Nova          15
          22     Science Fiction          13
          23       Rock And Roll          12
          24               Opera           1,
          Figure({
              'data': [{'alignmentgroup': 'True',
                        'hovertemplate': 'GenreName=%{x}<br>TrackCount=%{y}<extra></extra>',
                        'legendgroup': '',
                        'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
                        'name': '',
                        'offsetgroup': '',
                        'orientation': 'v',
                        'showlegend': False,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Rock', 'Latin', 'Metal', 'Alternative & Punk', 'Jazz', 'TV Shows',
                                    'Blues', 'Classical', 'Drama', 'R&B/Soul', 'Reggae', 'Pop',
                                    'Soundtrack', 'Alternative', 'Hip Hop/Rap', 'Electronica/Dance',
                                    'Heavy Metal', 'World', 'Sci Fi & Fantasy', 'Easy Listening', 'Comedy',
                                    'Bossa Nova', 'Science Fiction', 'Rock And Roll', 'Opera'], dtype=object),
                        'xaxis': 'x',
                        'y': array([1297,  579,  374,  332,  130,   93,   81,   74,   64,   61,   58,   48,
                                      43,   40,   35,   30,   28,   28,   26,   24,   17,   15,   13,   12,
                                       1]),
                        'yaxis': 'y'}],
              'layout': {'barmode': 'relative',
                         'legend': {'tracegroupgap': 0},
                         'template': '...',
                         'title': {'text': 'Number of Tracks by Genre'},
                         'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'tickangle': 45, 'title': {'text': 'GenreNa
         me'}},
                         'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'TrackCount'}}}
          }))
```

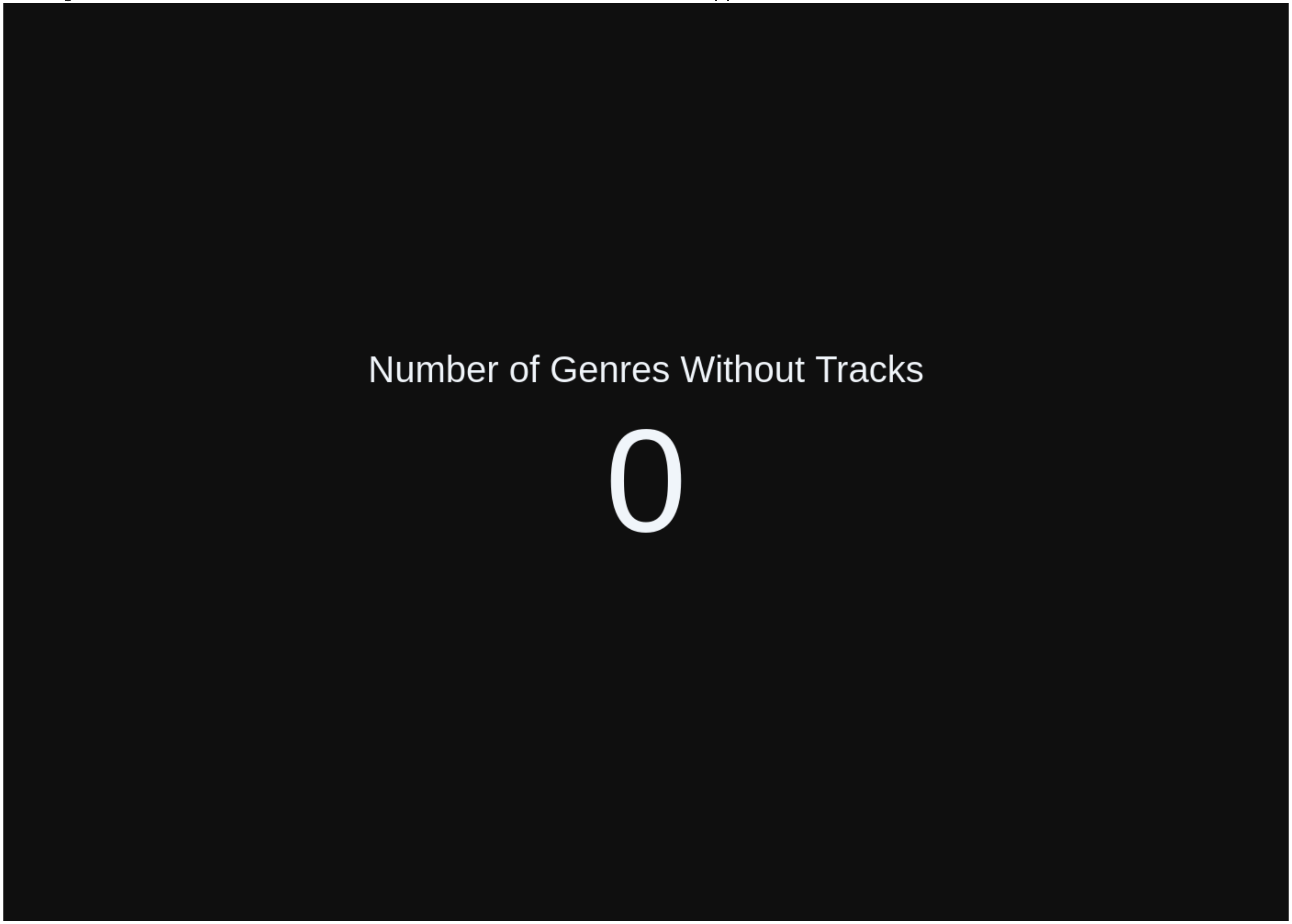
In [32]:

```python
question = """
    Get all genres that do not have any tracks associated with them:
"""

vn.ask(question=question)
```

```
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
[{'role': 'system', 'content': "You are a SQLite expert. Please help to generate a SQL query to answer the
question. Your response should ONLY be based on the given context and follow the response guidelines and fo
rmat instructions. \n===Tables \nCREATE INDEX IFK_TrackGenreId ON Track (GenreId)\n\nCREATE TABLE Track\n
(\n    TrackId INTEGER  NOT NULL,\n    Name NVARCHAR(200)  NOT NULL,\n    AlbumId INTEGER,\n    MediaTypeId
INTEGER  NOT NULL,\n    GenreId INTEGER,\n    Composer NVARCHAR(220),\n    Milliseconds INTEGER  NOT NUL
L,\n    Bytes INTEGER,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (Track
Id),\n    FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n
FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIG
N KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n
CREATE INDEX IFK_PlaylistTrackTrackId ON PlaylistTrack (TrackId)\n\nCREATE INDEX IFK_TrackAlbumId ON Track
(AlbumId)\n\nCREATE INDEX IFK_TrackMediaTypeId ON Track (MediaTypeId)\n\nCREATE TABLE Genre\n(\n    GenreId
INTEGER  NOT NULL,\n    Name NVARCHAR(120),\n    CONSTRAINT PK_Genre PRIMARY KEY  (GenreId)\n)\n\nCREATE TA
BLE Album\n(\n    AlbumId INTEGER  NOT NULL,\n    Title NVARCHAR(160)  NOT NULL,\n    ArtistId INTEGER  NOT
NULL,\n    CONSTRAINT PK_Album PRIMARY KEY  (AlbumId),\n    FOREIGN KEY (ArtistId) REFERENCES Artist (Artis
tId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_AlbumArtistId ON Album (ArtistId)
\n\nCREATE TABLE PlaylistTrack\n(\n    PlaylistId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    C
ONSTRAINT PK_PlaylistTrack PRIMARY KEY  (PlaylistId, TrackId),\n    FOREIGN KEY (PlaylistId) REFERENCES Pla
ylist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (TrackId) REFERENCES Tra
ck (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Artist\n(\n    ArtistId INTEG
ER  NOT NULL,\n    Name NVARCHAR(120),\n    CONSTRAINT PK_Artist PRIMARY KEY  (ArtistId)\n)\n\n\n===Additio
nal Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided
context is sufficient, please generate a valid SQL query without any explanations for the question. \n2. If
the provided context is almost sufficient but requires knowledge of a specific string in a particular colum
n, please generate an intermediate SQL query to find the distinct strings in that column. Prepend the query
with a comment saying intermediate_sql \n3. If the provided context is insufficient, please explain why it
can't be generated. \n4. Please use the most relevant table(s). \n5. If the question has been asked and ans
wered before, please repeat the answer exactly as it was given before. \n"}, {'role': 'user', 'content': '
\n    List all genres and the number of tracks in each genre:\n'}, {'role': 'assistant', 'content': 'SELECT
\n    Genre.Name AS GenreName,\n    COUNT(Track.TrackId) AS TrackCount\nFROM \n    Genre\nLEFT JOIN \n    T
rack ON Genre.GenreId = Track.GenreId\nGROUP BY \n    Genre.GenreId\nORDER BY \n    TrackCount DESC;'}, {'r
ole': 'user', 'content': '  \n    List all albums and their corresponding artist names  \n'}, {'role': 'ass
istant', 'content': 'SELECT \n    Album.Title AS AlbumTitle, \n    Artist.Name AS ArtistName\nFROM \n    Al
bum\nJOIN \n    Artist ON Album.ArtistId = Artist.ArtistId\nORDER BY \n    Artist.Name, Album.Title;'}, {'r
ole': 'user', 'content': '  \n    Find all tracks with a name containing "What" (case-insensitive)\n'}, {'r
ole': 'assistant', 'content': "SELECT TrackId, Name\nFROM Track\nWHERE Name LIKE '%What%' COLLATE NOCAS
E;"}, {'role': 'user', 'content': '  \n    Find the top 5 most expensive tracks (based on unit price):\n'},
{'role': 'assistant', 'content': 'SELECT \n    TrackId,\n    Name,\n    UnitPrice\nFROM \n    Track\nORDER
BY \n    UnitPrice DESC\nLIMIT 5;'}, {'role': 'user', 'content': 'Show me a list of tables in the SQLite da
tabase'}, {'role': 'assistant', 'content': "SELECT name FROM sqlite_master WHERE type='table';"}, {'role':
'user', 'content': '  \n    Find all invoices since 2010 and the total amount invoiced:\n'}, {'role': 'assi
stant', 'content': "SELECT \n    COUNT(*) AS InvoiceCount,\n    SUM(Total) AS TotalAmount\nFROM \n    Invoi
ce\nWHERE \n    InvoiceDate >= '2010-01-01';"}, {'role': 'user', 'content': 'what are the top 5 countries t
hat customers come from?'}, {'role': 'assistant', 'content': 'SELECT Country, COUNT(*) as CustomerCount\nFR
OM Customer\nGROUP BY Country\nORDER BY CustomerCount DESC\nLIMIT 5;'}, {'role': 'user', 'content': '  \n
List all invoices with a total exceeding $10:\n'}, {'role': 'assistant', 'content': 'SELECT \n    InvoiceI
d, \n    CustomerId, \n    InvoiceDate, \n    Total\nFROM \n    Invoice\nWHERE \n    Total > 10\nORDER BY
\n    Total DESC;'}, {'role': 'user', 'content': 'How many customers are there'}, {'role': 'assistant', 'co
ntent': 'SELECT COUNT(*) FROM Customer;'}, {'role': 'user', 'content': '  \n    Get the average invoice tot
al for each customer:\n'}, {'role': 'assistant', 'content': 'SELECT \n    Customer.CustomerId,\n    Custome
r.FirstName,\n    Customer.LastName,\n    AVG(Invoice.Total) AS AverageInvoiceTotal\nFROM \n    Customer\nL
EFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId\nGROUP BY \n    Customer.CustomerId\nORD
ER BY \n    AverageInvoiceTotal DESC;'}, {'role': 'user', 'content': '  \n    Get all genres that do not ha
ve any tracks associated with them:\n'}]
Using model claude-3-5-sonnet-20240620 for 1165.75 tokens (approx)
SELECT
    Genre.GenreId,
    Genre.Name
FROM
    Genre
LEFT JOIN
    Track ON Genre.GenreId = Track.GenreId
WHERE
    Track.TrackId IS NULL;
SELECT
    Genre.GenreId,
    Genre.Name
FROM
    Genre
LEFT JOIN
    Track ON Genre.GenreId = Track.GenreId
WHERE
    Track.TrackId IS NULL;
SELECT
    Genre.GenreId,
    Genre.Name
FROM
    Genre
LEFT JOIN
    Track ON Genre.GenreId = Track.GenreId
WHERE
    Track.TrackId IS NULL;
Empty DataFrame
Columns: [GenreId, Name]
```

```
Index: []
Using model claude-3-5-sonnet-20240620 for 203.25 tokens (approx)
```



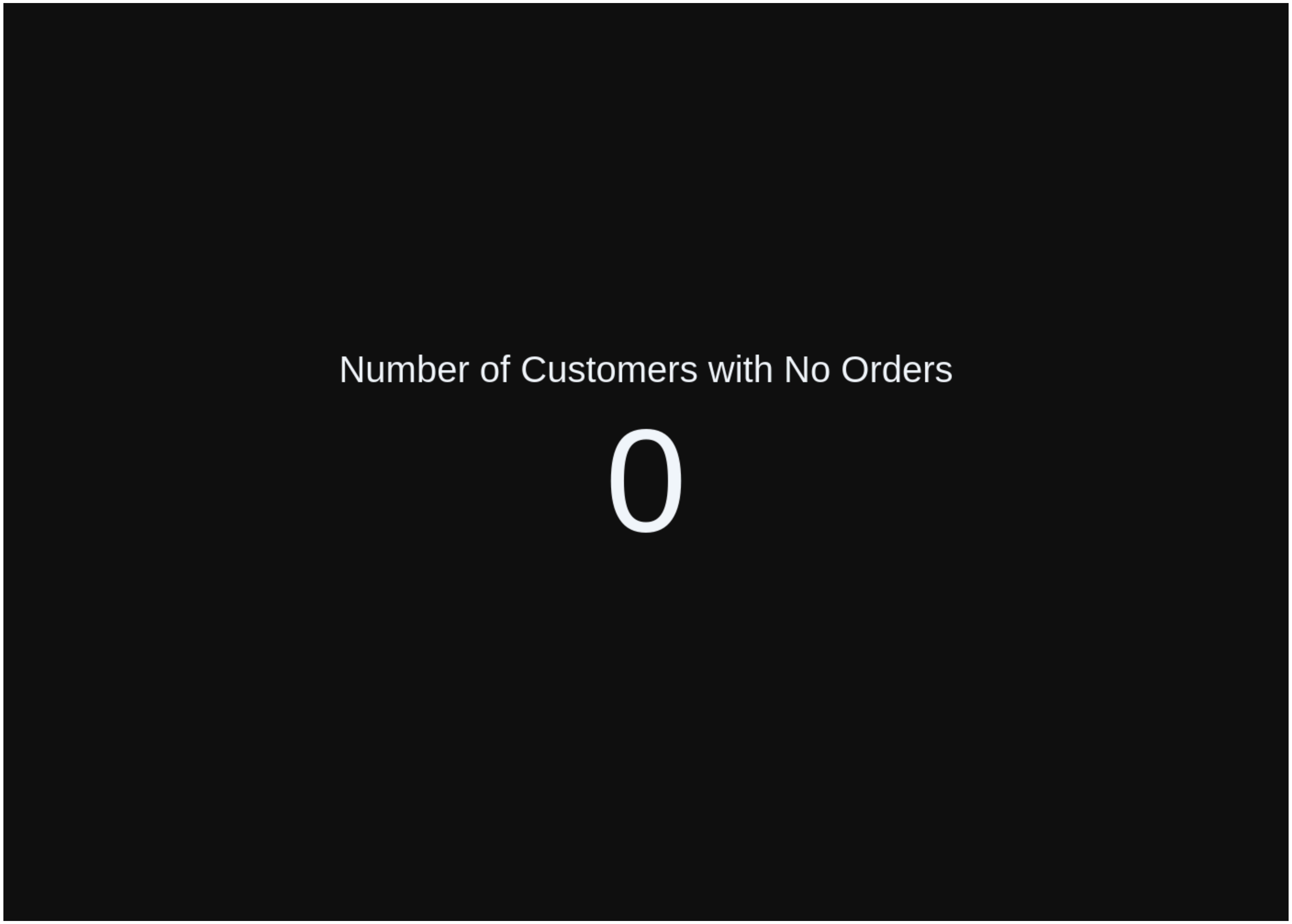
```
Out[32]: ('SELECT \n    Genre.GenreId,\n    Genre.Name\nFROM \n    Genre\nLEFT JOIN \n    Track ON Genre.GenreId =
         Track.GenreId\nWHERE \n    Track.TrackId IS NULL;',
          Empty DataFrame
          Columns: [GenreId, Name]
          Index: [],
          Figure({
              'data': [{'mode': 'number', 'title': {'text': 'Number of Genres Without Tracks'}, 'type': 'indicato
         r', 'value': 0}],
              'layout': {'template': '...'}
          }))
```

```
In [33]: question = """
             List all customers who have not placed any orders:
         """

         vn.ask(question=question)
```

```
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
[{'role': 'system', 'content': "You are a SQLite expert. Please help to generate a SQL query to answer the
question. Your response should ONLY be based on the given context and follow the response guidelines and fo
rmat instructions. \n===Tables \nCREATE TABLE Customer\n(\n    CustomerId INTEGER  NOT NULL,\n    FirstName
NVARCHAR(40)  NOT NULL,\n    LastName NVARCHAR(20)  NOT NULL,\n    Company NVARCHAR(80),\n    Address NVARC
HAR(70),\n    City NVARCHAR(40),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVARCH
AR(10),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60)  NOT NULL,\n    SupportRepI
d INTEGER,\n    CONSTRAINT PK_Customer PRIMARY KEY  (CustomerId),\n    FOREIGN KEY (SupportRepId) REFERENCE
S Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Invoice\n(\n    In
voiceId INTEGER  NOT NULL,\n    CustomerId INTEGER  NOT NULL,\n    InvoiceDate DATETIME  NOT NULL,\n    Bil
lingAddress NVARCHAR(70),\n    BillingCity NVARCHAR(40),\n    BillingState NVARCHAR(40),\n    BillingCountr
y NVARCHAR(40),\n    BillingPostalCode NVARCHAR(10),\n    Total NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK
_Invoice PRIMARY KEY  (InvoiceId),\n    FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON
DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE InvoiceLine\n(\n    InvoiceLineId INTEGER  NOT NUL
L,\n    InvoiceId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    UnitPrice NUMERIC(10,2)  NOT NUL
L,\n    Quantity INTEGER  NOT NULL,\n    CONSTRAINT PK_InvoiceLine PRIMARY KEY  (InvoiceLineId),\n    FOREI
GN KEY (InvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREI
GN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABL
E Employee\n(\n    EmployeeId INTEGER  NOT NULL,\n    LastName NVARCHAR(20)  NOT NULL,\n    FirstName NVARC
HAR(20)  NOT NULL,\n    Title NVARCHAR(30),\n    ReportsTo INTEGER,\n    BirthDate DATETIME,\n    HireDate
DATETIME,\n    Address NVARCHAR(70),\n    City NVARCHAR(40),\n    State NVARCHAR(40),\n    Country NVARCHAR
(40),\n    PostalCode NVARCHAR(10),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(6
0),\n    CONSTRAINT PK_Employee PRIMARY KEY  (EmployeeId),\n    FOREIGN KEY (ReportsTo) REFERENCES Employee
(EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_CustomerSupportRepId ON C
ustomer (SupportRepId)\n\nCREATE TABLE PlaylistTrack\n(\n    PlaylistId INTEGER  NOT NULL,\n    TrackId INT
EGER  NOT NULL,\n    CONSTRAINT PK_PlaylistTrack PRIMARY KEY  (PlaylistId, TrackId),\n    FOREIGN KEY (Play
listId) REFERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (T
rackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Album\n
(\n    AlbumId INTEGER  NOT NULL,\n    Title NVARCHAR(160)  NOT NULL,\n    ArtistId INTEGER  NOT NULL,\n
CONSTRAINT PK_Album PRIMARY KEY  (AlbumId),\n    FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\t
ON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)\n
\nCREATE TABLE Track\n(\n    TrackId INTEGER  NOT NULL,\n    Name NVARCHAR(200)  NOT NULL,\n    AlbumId INT
EGER,\n    MediaTypeId INTEGER  NOT NULL,\n    GenreId INTEGER,\n    Composer NVARCHAR(220),\n    Milliseco
nds INTEGER  NOT NULL,\n    Bytes INTEGER,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Track
PRIMARY KEY  (TrackId),\n    FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON
UPDATE NO ACTION,\n    FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE
NO ACTION,\n    FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON U
PDATE NO ACTION\n)\n\nCREATE TABLE Playlist\n(\n    PlaylistId INTEGER  NOT NULL,\n    Name NVARCHAR(12
0),\n    CONSTRAINT PK_Playlist PRIMARY KEY  (PlaylistId)\n)\n\n\n===Additional Context \n\nIn the chinook
database invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficient, please g
enerate a valid SQL query without any explanations for the question. \n2. If the provided context is almost
sufficient but requires knowledge of a specific string in a particular column, please generate an intermedi
ate SQL query to find the distinct strings in that column. Prepend the query with a comment saying intermed
iate_sql \n3. If the provided context is insufficient, please explain why it can't be generated. \n4. Pleas
e use the most relevant table(s). \n5. If the question has been asked and answered before, please repeat th
e answer exactly as it was given before. \n"}, {'role': 'user', 'content': '  \n    Get the total number of
invoices for each customer\n'}, {'role': 'assistant', 'content': 'SELECT \n    Customer.CustomerId,\n    Cu
stomer.FirstName,\n    Customer.LastName,\n    COUNT(Invoice.InvoiceId) AS TotalInvoices\nFROM \n    Custom
er\nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId\nGROUP BY \n    Customer.CustomerId
\nORDER BY \n    TotalInvoices DESC;'}, {'role': 'user', 'content': 'How many customers are there'}, {'rol
e': 'assistant', 'content': 'SELECT COUNT(*) FROM Customer;'}, {'role': 'user', 'content': 'what are the to
p 5 countries that customers come from?'}, {'role': 'assistant', 'content': 'SELECT Country, COUNT(*) as Cu
stomerCount\nFROM Customer\nGROUP BY Country\nORDER BY CustomerCount DESC\nLIMIT 5;'}, {'role': 'user', 'co
ntent': 'How many records are in table called customer'}, {'role': 'assistant', 'content': 'SELECT COUNT(*)
FROM Customer;'}, {'role': 'user', 'content': '  \n    Get the average invoice total for each custome
r:\n'}, {'role': 'assistant', 'content': 'SELECT \n    Customer.CustomerId,\n    Customer.FirstName,\n    C
ustomer.LastName,\n    AVG(Invoice.Total) AS AverageInvoiceTotal\nFROM \n    Customer\nLEFT JOIN \n    Invo
ice ON Customer.CustomerId = Invoice.CustomerId\nGROUP BY \n    Customer.CustomerId\nORDER BY \n    Average
InvoiceTotal DESC;'}, {'role': 'user', 'content': '  \n    List all invoices with a total exceeding $1
0:\n'}, {'role': 'assistant', 'content': 'SELECT \n    InvoiceId, \n    CustomerId, \n    InvoiceDate, \n
Total\nFROM \n    Invoice\nWHERE \n    Total > 10\nORDER BY \n    Total DESC;'}, {'role': 'user', 'conten
t': '  \n    Find all invoices since 2010 and the total amount invoiced:\n'}, {'role': 'assistant', 'conten
t': "SELECT \n    COUNT(*) AS InvoiceCount,\n    SUM(Total) AS TotalAmount\nFROM \n    Invoice\nWHERE \n
InvoiceDate >= '2010-01-01';"}, {'role': 'user', 'content': '  \n    Find the total number of invoices per
country:\n'}, {'role': 'assistant', 'content': 'SELECT \n    BillingCountry, \n    COUNT(*) AS TotalInvoice
s\nFROM \n    Invoice\nGROUP BY \n    BillingCountry\nORDER BY \n    TotalInvoices DESC;'}, {'role': 'use
r', 'content': "  \n    List all employees and their reporting manager's name (if any):\n"}, {'role': 'assi
stant', 'content': "SELECT \n    e.EmployeeId,\n    e.FirstName || ' ' || e.LastName AS EmployeeName,\n
m.FirstName || ' ' || m.LastName AS ManagerName\nFROM \n    Employee e\nLEFT JOIN \n    Employee m ON e.Rep
ortsTo = m.EmployeeId\nORDER BY \n    e.EmployeeId;"}, {'role': 'user', 'content': '  \n    List all albums
and their corresponding artist names  \n'}, {'role': 'assistant', 'content': 'SELECT \n    Album.Title AS A
lbumTitle, \n    Artist.Name AS ArtistName\nFROM \n    Album\nJOIN \n    Artist ON Album.ArtistId = Artist.
ArtistId\nORDER BY \n    Artist.Name, Album.Title;'}, {'role': 'user', 'content': '  \n    List all custome
rs who have not placed any orders:\n'}]
Using model claude-3-5-sonnet-20240620 for 1677.0 tokens (approx)
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    Customer.Email
FROM
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
WHERE
```

```
    Invoice.InvoiceId IS NULL;
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    Customer.Email
FROM
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
WHERE
    Invoice.InvoiceId IS NULL;
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    Customer.Email
FROM
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
WHERE
    Invoice.InvoiceId IS NULL;
Empty DataFrame
Columns: [CustomerId, FirstName, LastName, Email]
Index: []
Using model claude-3-5-sonnet-20240620 for 231.0 tokens (approx)
```


Number of Customers with No Orders
0

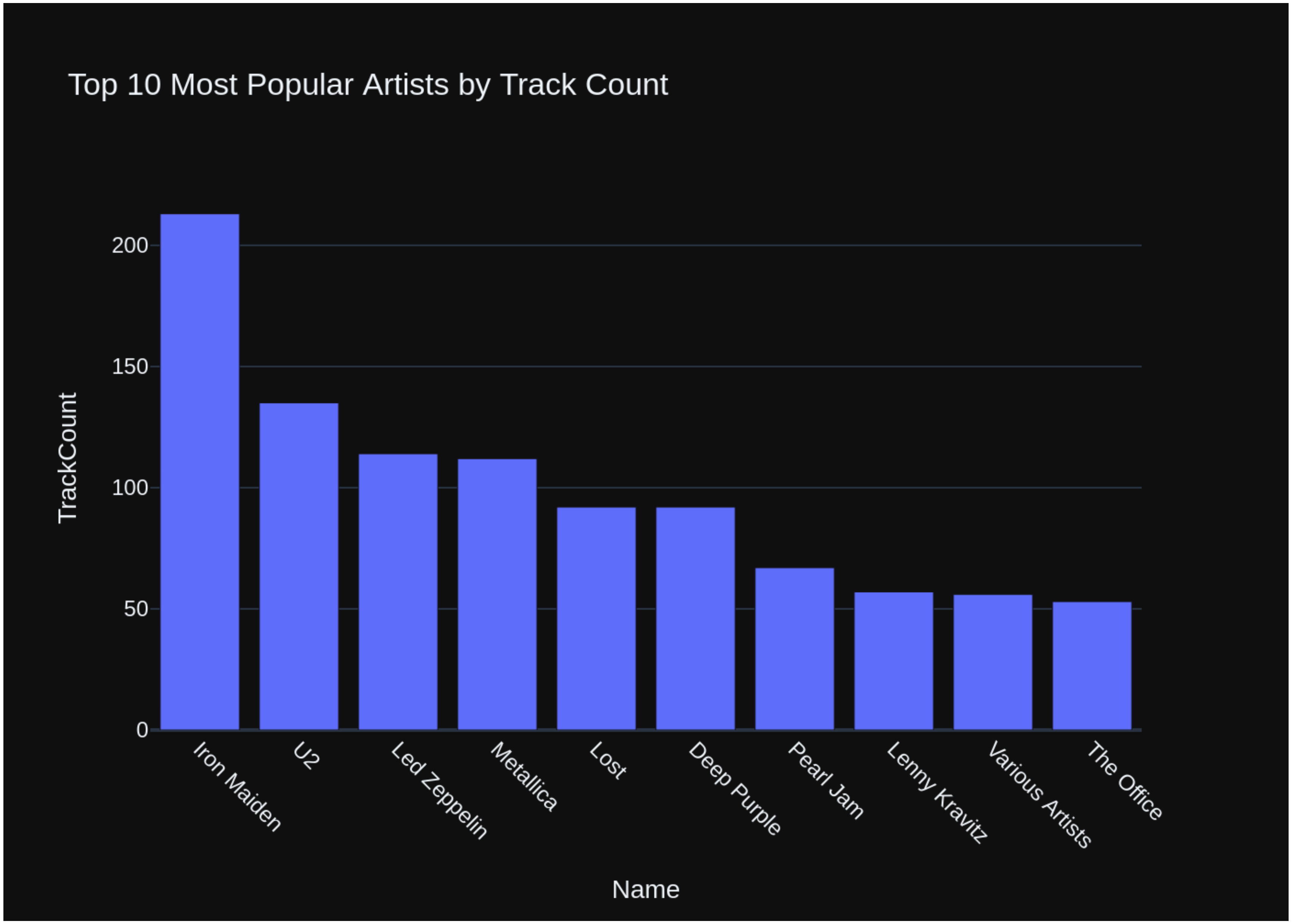
Out[33]:
```
('SELECT \n    Customer.CustomerId,\n    Customer.FirstName,\n    Customer.LastName,\n    Customer.Email\n
FROM \n    Customer\nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId\nWHERE \n    Invo
ice.InvoiceId IS NULL;',
 Empty DataFrame
 Columns: [CustomerId, FirstName, LastName, Email]
 Index: [],
 Figure({
     'data': [{'mode': 'number', 'title': {'text': 'Number of Customers with No Orders'}, 'type': 'indicat
or', 'value': 0}],
     'layout': {'template': '...'}
 }))
```

In [34]:
```python
question = """
    Get the top 10 most popular artists (based on the number of tracks):
"""

vn.ask(question=question)
```

```
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
[{'role': 'system', 'content': "You are a SQLite expert. Please help to generate a SQL query to answer the
question. Your response should ONLY be based on the given context and follow the response guidelines and fo
rmat instructions. \n===Tables \nCREATE TABLE Track\n(\n    TrackId INTEGER  NOT NULL,\n    Name NVARCHAR(2
00)  NOT NULL,\n    AlbumId INTEGER,\n    MediaTypeId INTEGER  NOT NULL,\n    GenreId INTEGER,\n    Compose
r NVARCHAR(220),\n    Milliseconds INTEGER  NOT NULL,\n    Bytes INTEGER,\n    UnitPrice NUMERIC(10,2)  NOT
NULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (TrackId),\n    FOREIGN KEY (AlbumId) REFERENCES Album (AlbumI
d) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n
\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaType
Id) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_AlbumArtistId ON Album (ArtistId)
\n\nCREATE INDEX IFK_TrackAlbumId ON Track (AlbumId)\n\nCREATE TABLE Artist\n(\n    ArtistId INTEGER  NOT N
ULL,\n    Name NVARCHAR(120),\n    CONSTRAINT PK_Artist PRIMARY KEY  (ArtistId)\n)\n\nCREATE INDEX IFK_Trac
kGenreId ON Track (GenreId)\n\nCREATE TABLE Album\n(\n    AlbumId INTEGER  NOT NULL,\n    Title NVARCHAR(16
0)  NOT NULL,\n    ArtistId INTEGER  NOT NULL,\n    CONSTRAINT PK_Album PRIMARY KEY  (AlbumId),\n    FOREIG
N KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE IN
DEX IFK_PlaylistTrackTrackId ON PlaylistTrack (TrackId)\n\nCREATE INDEX IFK_TrackMediaTypeId ON Track (Medi
aTypeId)\n\nCREATE TABLE Playlist\n(\n    PlaylistId INTEGER  NOT NULL,\n    Name NVARCHAR(120),\n    CONST
RAINT PK_Playlist PRIMARY KEY  (PlaylistId)\n)\n\nCREATE TABLE PlaylistTrack\n(\n    PlaylistId INTEGER  NO
T NULL,\n    TrackId INTEGER  NOT NULL,\n    CONSTRAINT PK_PlaylistTrack PRIMARY KEY  (PlaylistId, TrackI
d),\n    FOREIGN KEY (PlaylistId) REFERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO A
CTION,\n    FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION
\n)\n\n\n===Additional Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n
1. If the provided context is sufficient, please generate a valid SQL query without any explanations for th
e question. \n2. If the provided context is almost sufficient but requires knowledge of a specific string i
n a particular column, please generate an intermediate SQL query to find the distinct strings in that colum
n. Prepend the query with a comment saying intermediate_sql \n3. If the provided context is insufficient, p
lease explain why it can't be generated. \n4. Please use the most relevant table(s). \n5. If the question h
as been asked and answered before, please repeat the answer exactly as it was given before. \n"}, {'role':
'user', 'content': '  \n    Find the top 5 most expensive tracks (based on unit price):\n'}, {'role': 'assi
stant', 'content': 'SELECT \n    TrackId,\n    Name,\n    UnitPrice\nFROM \n    Track\nORDER BY \n    UnitP
rice DESC\nLIMIT 5;'}, {'role': 'user', 'content': '  \n    List all genres and the number of tracks in eac
h genre:\n'}, {'role': 'assistant', 'content': 'SELECT \n    Genre.Name AS GenreName,\n    COUNT(Track.Trac
kId) AS TrackCount\nFROM \n    Genre\nLEFT JOIN \n    Track ON Genre.GenreId = Track.GenreId\nGROUP BY \n
Genre.GenreId\nORDER BY \n    TrackCount DESC;'}, {'role': 'user', 'content': '  \n    List all albums and
their corresponding artist names  \n'}, {'role': 'assistant', 'content': 'SELECT \n    Album.Title AS Album
Title, \n    Artist.Name AS ArtistName\nFROM \n    Album\nJOIN \n    Artist ON Album.ArtistId = Artist.Arti
stId\nORDER BY \n    Artist.Name, Album.Title;'}, {'role': 'user', 'content': 'what are the top 5 countries
that customers come from?'}, {'role': 'assistant', 'content': 'SELECT Country, COUNT(*) as CustomerCount\nF
ROM Customer\nGROUP BY Country\nORDER BY CustomerCount DESC\nLIMIT 5;'}, {'role': 'user', 'content': '  \n
List all invoices with a total exceeding $10:\n'}, {'role': 'assistant', 'content': 'SELECT \n    InvoiceI
d, \n    CustomerId, \n    InvoiceDate, \n    Total\nFROM \n    Invoice\nWHERE \n    Total > 10\nORDER BY
\n    Total DESC;'}, {'role': 'user', 'content': 'How many customers are there'}, {'role': 'assistant', 'co
ntent': 'SELECT COUNT(*) FROM Customer;'}, {'role': 'user', 'content': '  \n    Find all tracks with a name
containing "What" (case-insensitive)\n'}, {'role': 'assistant', 'content': "SELECT TrackId, Name\nFROM Trac
k\nWHERE Name LIKE '%What%' COLLATE NOCASE;"}, {'role': 'user', 'content': '  \n    Get the average invoice
total for each customer:\n'}, {'role': 'assistant', 'content': 'SELECT \n    Customer.CustomerId,\n    Cust
omer.FirstName,\n    Customer.LastName,\n    AVG(Invoice.Total) AS AverageInvoiceTotal\nFROM \n    Customer
\nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId\nGROUP BY \n    Customer.CustomerId\n
ORDER BY \n    AverageInvoiceTotal DESC;'}, {'role': 'user', 'content': 'How many records are in table call
ed customer'}, {'role': 'assistant', 'content': 'SELECT COUNT(*) FROM Customer;'}, {'role': 'user', 'conten
t': '  \n    Find the total number of invoices per country:\n'}, {'role': 'assistant', 'content': 'SELECT 
\n    BillingCountry, \n    COUNT(*) AS TotalInvoices\nFROM \n    Invoice\nGROUP BY \n    BillingCountry\nO
RDER BY \n    TotalInvoices DESC;'}, {'role': 'user', 'content': '  \n    Get the top 10 most popular artis
ts (based on the number of tracks):\n'}]
Using model claude-3-5-sonnet-20240620 for 1164.5 tokens (approx)
SELECT
    Artist.ArtistId,
    Artist.Name,
    COUNT(Track.TrackId) AS TrackCount
FROM
    Artist
JOIN
    Album ON Artist.ArtistId = Album.ArtistId
JOIN
    Track ON Album.AlbumId = Track.AlbumId
GROUP BY
    Artist.ArtistId
ORDER BY
    TrackCount DESC
LIMIT 10;
SELECT
    Artist.ArtistId,
    Artist.Name,
    COUNT(Track.TrackId) AS TrackCount
FROM
    Artist
JOIN
    Album ON Artist.ArtistId = Album.ArtistId
JOIN
    Track ON Album.AlbumId = Track.AlbumId
GROUP BY
    Artist.ArtistId
ORDER BY
    TrackCount DESC
LIMIT 10;
```

```
SELECT
    Artist.ArtistId,
    Artist.Name,
    COUNT(Track.TrackId) AS TrackCount
FROM
    Artist
JOIN
    Album ON Artist.ArtistId = Album.ArtistId
JOIN
    Track ON Album.AlbumId = Track.AlbumId
GROUP BY
    Artist.ArtistId
ORDER BY
    TrackCount DESC
LIMIT 10;
   ArtistId             Name  TrackCount
0        90      Iron Maiden         213
1       150               U2         135
2        22     Led Zeppelin         114
3        50        Metallica         112
4       149             Lost          92
5        58      Deep Purple          92
6       118        Pearl Jam          67
7       100    Lenny Kravitz          57
8        21  Various Artists          56
9       156       The Office          53
Using model claude-3-5-sonnet-20240620 for 242.75 tokens (approx)
```


Top 10 Most Popular Artists by Track Count
TrackCount
200
150
100
50
0
Iron Maiden
U2
Led Zeppelin
Metallica
Lost
Deep Purple
Pearl Jam
Lenny Kravitz
Various Artists
The Office
Name

Out[34]:
```
('SELECT \n    Artist.ArtistId,\n    Artist.Name,\n    COUNT(Track.TrackId) AS TrackCount\nFROM \n    Arti
st\nJOIN \n    Album ON Artist.ArtistId = Album.ArtistId\nJOIN \n    Track ON Album.AlbumId = Track.AlbumI
d\nGROUP BY \n    Artist.ArtistId\nORDER BY \n    TrackCount DESC\nLIMIT 10;',
    ArtistId             Name  TrackCount
 0        90      Iron Maiden         213
 1       150               U2         135
 2        22     Led Zeppelin         114
 3        50        Metallica         112
 4       149             Lost          92
 5        58      Deep Purple          92
 6       118        Pearl Jam          67
 7       100    Lenny Kravitz          57
 8        21  Various Artists          56
 9       156       The Office          53,
 Figure({
     'data': [{'alignmentgroup': 'True',
               'hovertemplate': 'Name=%{x}<br>TrackCount=%{y}<extra></extra>',
               'legendgroup': '',
               'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
               'name': '',
               'offsetgroup': '',
               'orientation': 'v',
               'showlegend': False,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Iron Maiden', 'U2', 'Led Zeppelin', 'Metallica', 'Lost', 'Deep Purple',
                           'Pearl Jam', 'Lenny Kravitz', 'Various Artists', 'The Office'],
                          dtype=object),
               'xaxis': 'x',
               'y': array([213, 135, 114, 112,  92,  92,  67,  57,  56,  53]),
               'yaxis': 'y'}],
     'layout': {'barmode': 'relative',
                'legend': {'tracegroupgap': 0},
                'template': '...',
                'title': {'text': 'Top 10 Most Popular Artists by Track Count'},
                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'tickangle': 45, 'title': {'text': 'Nam
e'}},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'TrackCount'}}}
 }))
```

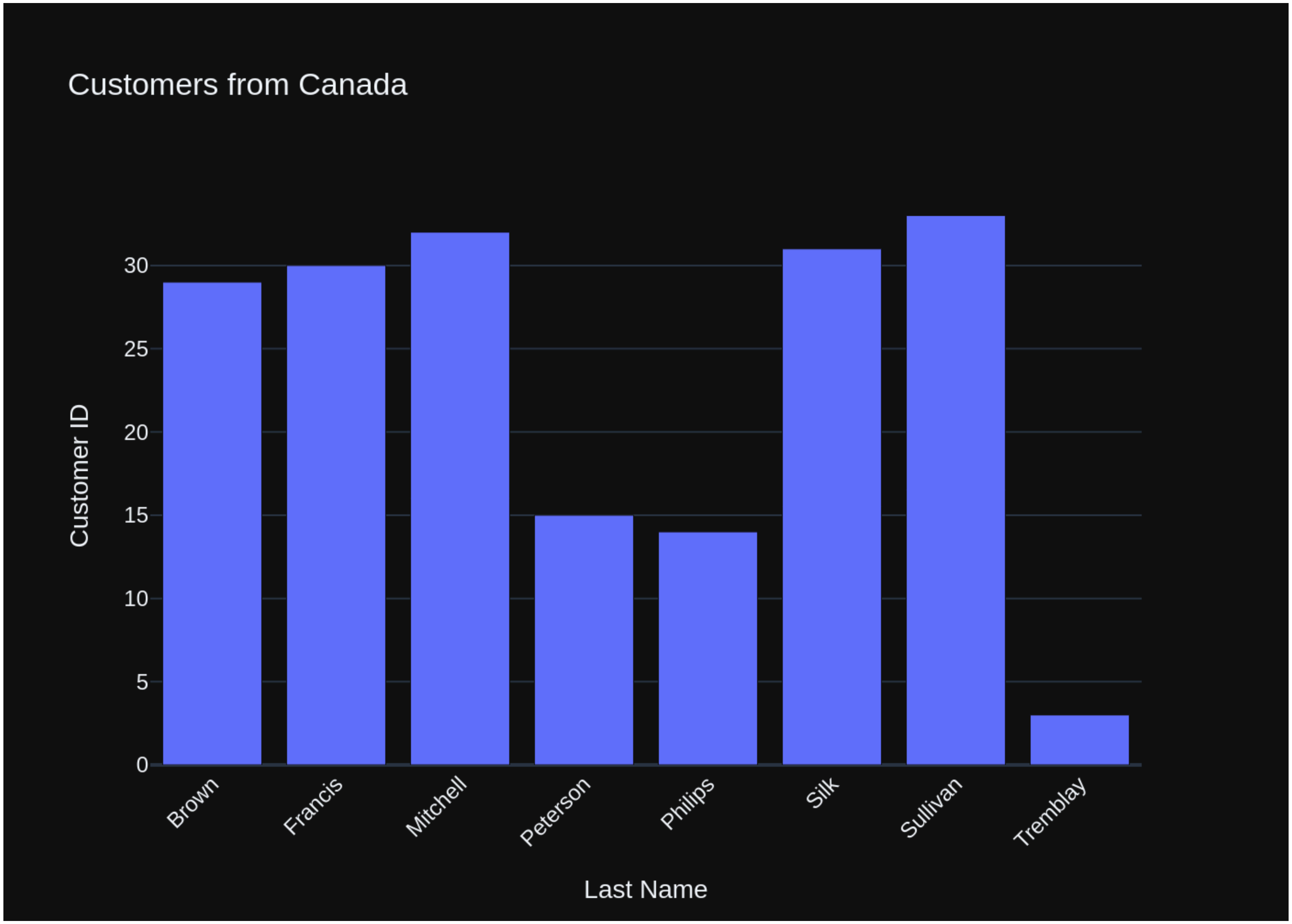
In [35]:
```
question = """
     List all customers from Canada and their email addresses:
```

```
"""

vn.ask(question=question)
```

```
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
[{'role': 'system', 'content': "You are a SQLite expert. Please help to generate a SQL query to answer the
question. Your response should ONLY be based on the given context and follow the response guidelines and fo
rmat instructions. \n===Tables \nCREATE TABLE Customer\n(\n    CustomerId INTEGER  NOT NULL,\n    FirstName
NVARCHAR(40)  NOT NULL,\n    LastName NVARCHAR(20)  NOT NULL,\n    Company NVARCHAR(80),\n    Address NVARC
HAR(70),\n    City NVARCHAR(40),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVARCH
AR(10),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60)  NOT NULL,\n    SupportRepI
d INTEGER,\n    CONSTRAINT PK_Customer PRIMARY KEY  (CustomerId),\n    FOREIGN KEY (SupportRepId) REFERENCE
S Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_CustomerSuppor
tRepId ON Customer (SupportRepId)\n\nCREATE TABLE Invoice\n(\n    InvoiceId INTEGER  NOT NULL,\n    Custome
rId INTEGER  NOT NULL,\n    InvoiceDate DATETIME  NOT NULL,\n    BillingAddress NVARCHAR(70),\n    BillingC
ity NVARCHAR(40),\n    BillingState NVARCHAR(40),\n    BillingCountry NVARCHAR(40),\n    BillingPostalCode
NVARCHAR(10),\n    Total NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Invoice PRIMARY KEY  (InvoiceId),\n
FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)
\n\nCREATE INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)\n\nCREATE TABLE Employee\n(\n    EmployeeId
INTEGER  NOT NULL,\n    LastName NVARCHAR(20)  NOT NULL,\n    FirstName NVARCHAR(20)  NOT NULL,\n    Title
NVARCHAR(30),\n    ReportsTo INTEGER,\n    BirthDate DATETIME,\n    HireDate DATETIME,\n    Address NVARCHA
R(70),\n    City NVARCHAR(40),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVARCHAR
(10),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60),\n    CONSTRAINT PK_Employee
PRIMARY KEY  (EmployeeId),\n    FOREIGN KEY (ReportsTo) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO
ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE InvoiceLine\n(\n    InvoiceLineId INTEGER  NOT NULL,\n    Inv
oiceId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    Quan
tity INTEGER  NOT NULL,\n    CONSTRAINT PK_InvoiceLine PRIMARY KEY  (InvoiceLineId),\n    FOREIGN KEY (Invo
iceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (Trac
kId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_Invoic
eLineTrackId ON InvoiceLine (TrackId)\n\nCREATE INDEX IFK_InvoiceLineInvoiceId ON InvoiceLine (InvoiceId)\n
\nCREATE INDEX IFK_EmployeeReportsTo ON Employee (ReportsTo)\n\nCREATE TABLE PlaylistTrack\n(\n    Playlist
Id INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    CONSTRAINT PK_PlaylistTrack PRIMARY KEY  (Playli
stId, TrackId),\n    FOREIGN KEY (PlaylistId) REFERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON
UPDATE NO ACTION,\n    FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE
NO ACTION\n)\n\n\n===Additional Context \n\nIn the chinook database invoice means order\n\n===Response Guid
elines \n1. If the provided context is sufficient, please generate a valid SQL query without any explanatio
ns for the question. \n2. If the provided context is almost sufficient but requires knowledge of a specific
string in a particular column, please generate an intermediate SQL query to find the distinct strings in th
at column. Prepend the query with a comment saying intermediate_sql \n3. If the provided context is insuffi
cient, please explain why it can't be generated. \n4. Please use the most relevant table(s). \n5. If the qu
estion has been asked and answered before, please repeat the answer exactly as it was given before. \n"},
{'role': 'user', 'content': 'what are the top 5 countries that customers come from?'}, {'role': 'assistan
t', 'content': 'SELECT Country, COUNT(*) as CustomerCount\nFROM Customer\nGROUP BY Country\nORDER BY Custom
erCount DESC\nLIMIT 5;'}, {'role': 'user', 'content': '  \n    Get the total number of invoices for each cu
stomer\n'}, {'role': 'assistant', 'content': 'SELECT \n    Customer.CustomerId,\n    Customer.FirstName,\n
Customer.LastName,\n    COUNT(Invoice.InvoiceId) AS TotalInvoices\nFROM \n    Customer\nLEFT JOIN \n    Inv
oice ON Customer.CustomerId = Invoice.CustomerId\nGROUP BY \n    Customer.CustomerId\nORDER BY \n    TotalI
nvoices DESC;'}, {'role': 'user', 'content': 'How many customers are there'}, {'role': 'assistant', 'conten
t': 'SELECT COUNT(*) FROM Customer;'}, {'role': 'user', 'content': '  \n    Find the total number of invoic
es per country:\n'}, {'role': 'assistant', 'content': 'SELECT \n    BillingCountry, \n    COUNT(*) AS Total
Invoices\nFROM \n    Invoice\nGROUP BY \n    BillingCountry\nORDER BY \n    TotalInvoices DESC;'}, {'role':
'user', 'content': 'How many records are in table called customer'}, {'role': 'assistant', 'content': 'SELE
CT COUNT(*) FROM Customer;'}, {'role': 'user', 'content': "  \n    List all employees and their reporting m
anager's name (if any):\n"}, {'role': 'assistant', 'content': "SELECT \n    e.EmployeeId,\n    e.FirstName
|| ' ' || e.LastName AS EmployeeName,\n    m.FirstName || ' ' || m.LastName AS ManagerName\nFROM \n    Empl
oyee e\nLEFT JOIN \n    Employee m ON e.ReportsTo = m.EmployeeId\nORDER BY \n    e.EmployeeId;"}, {'role':
'user', 'content': '  \n    Get the average invoice total for each customer:\n'}, {'role': 'assistant', 'co
ntent': 'SELECT \n    Customer.CustomerId,\n    Customer.FirstName,\n    Customer.LastName,\n    AVG(Invoic
e.Total) AS AverageInvoiceTotal\nFROM \n    Customer\nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invo
ice.CustomerId\nGROUP BY \n    Customer.CustomerId\nORDER BY \n    AverageInvoiceTotal DESC;'}, {'role': 'u
ser', 'content': '  \n    List all invoices with a total exceeding $10:\n'}, {'role': 'assistant', 'conten
t': 'SELECT \n    InvoiceId, \n    CustomerId, \n    InvoiceDate, \n    Total\nFROM \n    Invoice\nWHERE \n
Total > 10\nORDER BY \n    Total DESC;'}, {'role': 'user', 'content': '  \n    Find all invoices since 2010
and the total amount invoiced:\n'}, {'role': 'assistant', 'content': "SELECT \n    COUNT(*) AS InvoiceCoun
t,\n    SUM(Total) AS TotalAmount\nFROM \n    Invoice\nWHERE \n    InvoiceDate >= '2010-01-01';"}, {'role':
'user', 'content': '  \n    Find the top 5 most expensive tracks (based on unit price):\n'}, {'role': 'assi
stant', 'content': 'SELECT \n    TrackId,\n    Name,\n    UnitPrice\nFROM \n    Track\nORDER BY \n    UnitP
rice DESC\nLIMIT 5;'}, {'role': 'user', 'content': '  \n     List all customers from Canada and their email
addresses:\n'}]
Using model claude-3-5-sonnet-20240620 for 1446.0 tokens (approx)
SELECT
    CustomerId,
    FirstName,
    LastName,
    Email
FROM
    Customer
WHERE
    Country = 'Canada'
ORDER BY
    LastName, FirstName;
SELECT
    CustomerId,
    FirstName,
    LastName,
    Email
FROM
    Customer
WHERE
```

```
    Country = 'Canada'
ORDER BY
    LastName, FirstName;
SELECT
    CustomerId,
    FirstName,
    LastName,
    Email
FROM
    Customer
WHERE
    Country = 'Canada'
ORDER BY
    LastName, FirstName;
   CustomerId FirstName  LastName                    Email
0          29    Robert     Brown        robbrown@shaw.ca
1          30    Edward   Francis     edfrancis@yachoo.ca
2          32     Aaron  Mitchell  aaronmitchell@yahoo.ca
3          15  Jennifer  Peterson     jenniferp@rogers.ca
4          14      Mark   Philips      mphilips12@shaw.ca
5          31    Martha      Silk    marthasilk@gmail.com
6          33     Ellie  Sullivan  ellie.sullivan@shaw.ca
7           3  François  Tremblay     ftremblay@gmail.com
Using model claude-3-5-sonnet-20240620 for 214.0 tokens (approx)
```

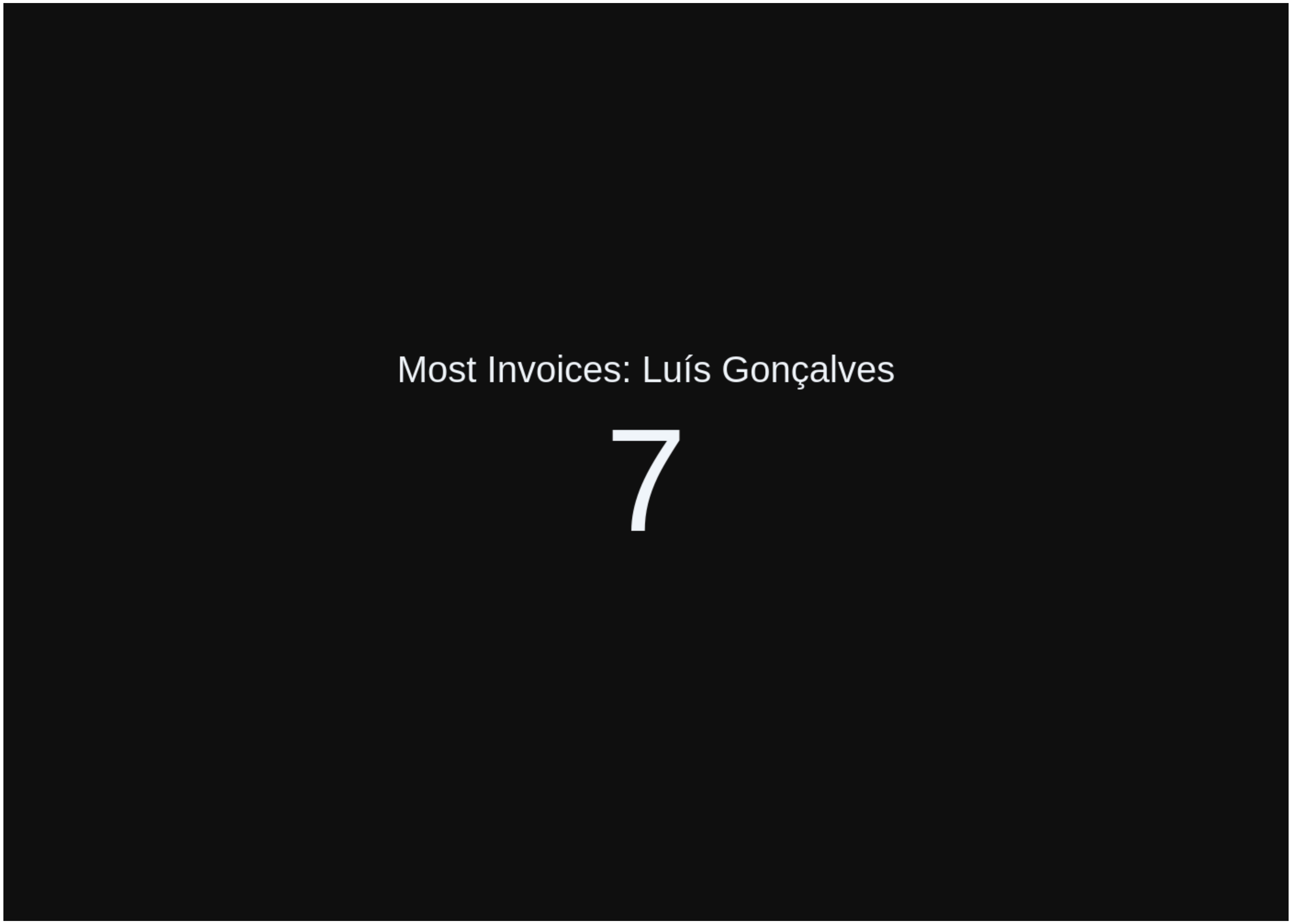

Customers from Canada
Customer ID
Last Name
0
5
10
15
20
25
30
Brown
Francis
Mitchell
Peterson
Philips
Silk
Sullivan
Tremblay

```
Out[35]: ("SELECT \n    CustomerId,\n    FirstName,\n    LastName,\n    Email\nFROM \n    Customer\nWHERE \n    Cou
         ntry = 'Canada'\nORDER BY \n    LastName, FirstName;",
             CustomerId FirstName  LastName                   Email
          0          29    Robert     Brown        robbrown@shaw.ca
          1          30    Edward   Francis     edfrancis@yachoo.ca
          2          32     Aaron  Mitchell  aaronmitchell@yahoo.ca
          3          15  Jennifer  Peterson     jenniferp@rogers.ca
          4          14      Mark   Philips      mphilips12@shaw.ca
          5          31    Martha      Silk    marthasilk@gmail.com
          6          33     Ellie  Sullivan  ellie.sullivan@shaw.ca
          7           3  François  Tremblay     ftremblay@gmail.com,
          Figure({
              'data': [{'alignmentgroup': 'True',
                        'customdata': array([['Robert', 'robbrown@shaw.ca'],
                                             ['Edward', 'edfrancis@yachoo.ca'],
                                             ['Aaron', 'aaronmitchell@yahoo.ca'],
                                             ['Jennifer', 'jenniferp@rogers.ca'],
                                             ['Mark', 'mphilips12@shaw.ca'],
                                             ['Martha', 'marthasilk@gmail.com'],
                                             ['Ellie', 'ellie.sullivan@shaw.ca'],
                                             ['François', 'ftremblay@gmail.com']], dtype=object),
                        'hovertemplate': ('Last Name=%{x}<br>Customer ID=' ... '{customdata[1]}<extra></extra>'),
                        'legendgroup': '',
                        'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
                        'name': '',
                        'offsetgroup': '',
                        'orientation': 'v',
                        'showlegend': False,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Brown', 'Francis', 'Mitchell', 'Peterson', 'Philips', 'Silk',
                                    'Sullivan', 'Tremblay'], dtype=object),
                        'xaxis': 'x',
                        'y': array([29, 30, 32, 15, 14, 31, 33,  3]),
                        'yaxis': 'y'}],
              'layout': {'barmode': 'relative',
                         'legend': {'tracegroupgap': 0},
                         'template': '...',
                         'title': {'text': 'Customers from Canada'},
                         'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'tickangle': -45, 'title': {'text': 'Last N
         ame'}},
                    'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'Customer ID'}}}
})) 
```

In [36]:
```
question = """
     Find the customer with the most invoices
"""

vn.ask(question=question)
```

```
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
[{'role': 'system', 'content': "You are a SQLite expert. Please help to generate a SQL query to answer the
question. Your response should ONLY be based on the given context and follow the response guidelines and fo
rmat instructions. \n===Tables \nCREATE INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)\n\nCREATE TABLE
Invoice\n(\n    InvoiceId INTEGER  NOT NULL,\n    CustomerId INTEGER  NOT NULL,\n    InvoiceDate DATETIME
NOT NULL,\n    BillingAddress NVARCHAR(70),\n    BillingCity NVARCHAR(40),\n    BillingState NVARCHAR(4
0),\n    BillingCountry NVARCHAR(40),\n    BillingPostalCode NVARCHAR(10),\n    Total NUMERIC(10,2)  NOT NU
LL,\n    CONSTRAINT PK_Invoice PRIMARY KEY  (InvoiceId),\n    FOREIGN KEY (CustomerId) REFERENCES Customer
(CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceLineInvoiceId ON I
nvoiceLine (InvoiceId)\n\nCREATE TABLE InvoiceLine\n(\n    InvoiceLineId INTEGER  NOT NULL,\n    InvoiceId
INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    Quantity IN
TEGER  NOT NULL,\n    CONSTRAINT PK_InvoiceLine PRIMARY KEY  (InvoiceLineId),\n    FOREIGN KEY (InvoiceId)
REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (TrackId) RE
FERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceLineTr
ackId ON InvoiceLine (TrackId)\n\nCREATE TABLE Customer\n(\n    CustomerId INTEGER  NOT NULL,\n    FirstNam
e NVARCHAR(40)  NOT NULL,\n    LastName NVARCHAR(20)  NOT NULL,\n    Company NVARCHAR(80),\n    Address NVA
RCHAR(70),\n    City NVARCHAR(40),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVAR
CHAR(10),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60)  NOT NULL,\n    SupportRe
pId INTEGER,\n    CONSTRAINT PK_Customer PRIMARY KEY  (CustomerId),\n    FOREIGN KEY (SupportRepId) REFEREN
CES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_CustomerSupp
ortRepId ON Customer (SupportRepId)\n\nCREATE TABLE Employee\n(\n    EmployeeId INTEGER  NOT NULL,\n    Las
tName NVARCHAR(20)  NOT NULL,\n    FirstName NVARCHAR(20)  NOT NULL,\n    Title NVARCHAR(30),\n    ReportsT
o INTEGER,\n    BirthDate DATETIME,\n    HireDate DATETIME,\n    Address NVARCHAR(70),\n    City NVARCHAR(4
0),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVARCHAR(10),\n    Phone NVARCHAR(2
4),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60),\n    CONSTRAINT PK_Employee PRIMARY KEY  (EmployeeI
d),\n    FOREIGN KEY (ReportsTo) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO AC
TION\n)\n\nCREATE INDEX IFK_EmployeeReportsTo ON Employee (ReportsTo)\n\nCREATE TABLE Track\n(\n    TrackId
INTEGER  NOT NULL,\n    Name NVARCHAR(200)  NOT NULL,\n    AlbumId INTEGER,\n    MediaTypeId INTEGER  NOT N
ULL,\n    GenreId INTEGER,\n    Composer NVARCHAR(220),\n    Milliseconds INTEGER  NOT NULL,\n    Bytes INT
EGER,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (TrackId),\n    FOREIGN
KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (G
enreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (MediaTy
peId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n===Additiona
l Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided co
ntext is sufficient, please generate a valid SQL query without any explanations for the question. \n2. If t
he provided context is almost sufficient but requires knowledge of a specific string in a particular colum
n, please generate an intermediate SQL query to find the distinct strings in that column. Prepend the query
with a comment saying intermediate_sql \n3. If the provided context is insufficient, please explain why it
can't be generated. \n4. Please use the most relevant table(s). \n5. If the question has been asked and ans
wered before, please repeat the answer exactly as it was given before. \n"}, {'role': 'user', 'content': '
\n    Get the total number of invoices for each customer\n'}, {'role': 'assistant', 'content': 'SELECT \n
Customer.CustomerId,\n    Customer.FirstName,\n    Customer.LastName,\n    COUNT(Invoice.InvoiceId) AS Tota
lInvoices\nFROM \n    Customer\nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId\nGROUP
BY \n    Customer.CustomerId\nORDER BY \n    TotalInvoices DESC;'}, {'role': 'user', 'content': '  \n    Li
st all invoices with a total exceeding $10:\n'}, {'role': 'assistant', 'content': 'SELECT \n    InvoiceId,
\n    CustomerId, \n    InvoiceDate, \n    Total\nFROM \n    Invoice\nWHERE \n    Total > 10\nORDER BY \n
Total DESC;'}, {'role': 'user', 'content': '  \n    Get the average invoice total for each customer:\n'},
{'role': 'assistant', 'content': 'SELECT \n    Customer.CustomerId,\n    Customer.FirstName,\n    Customer.
LastName,\n    AVG(Invoice.Total) AS AverageInvoiceTotal\nFROM \n    Customer\nLEFT JOIN \n    Invoice ON C
ustomer.CustomerId = Invoice.CustomerId\nGROUP BY \n    Customer.CustomerId\nORDER BY \n    AverageInvoiceT
otal DESC;'}, {'role': 'user', 'content': '  \n    Find the total number of invoices per country:\n'}, {'ro
le': 'assistant', 'content': 'SELECT \n    BillingCountry, \n    COUNT(*) AS TotalInvoices\nFROM \n    Invo
ice\nGROUP BY \n    BillingCountry\nORDER BY \n    TotalInvoices DESC;'}, {'role': 'user', 'content': '  \n
Find all invoices since 2010 and the total amount invoiced:\n'}, {'role': 'assistant', 'content': "SELECT
\n    COUNT(*) AS InvoiceCount,\n    SUM(Total) AS TotalAmount\nFROM \n    Invoice\nWHERE \n    InvoiceDate
>= '2010-01-01';"}, {'role': 'user', 'content': 'what are the top 5 countries that customers come from?'},
{'role': 'assistant', 'content': 'SELECT Country, COUNT(*) as CustomerCount\nFROM Customer\nGROUP BY Countr
y\nORDER BY CustomerCount DESC\nLIMIT 5;'}, {'role': 'user', 'content': '  \n    Find the top 5 most expens
ive tracks (based on unit price):\n'}, {'role': 'assistant', 'content': 'SELECT \n    TrackId,\n    Name,\n
UnitPrice\nFROM \n    Track\nORDER BY \n    UnitPrice DESC\nLIMIT 5;'}, {'role': 'user', 'content': 'How ma
ny customers are there'}, {'role': 'assistant', 'content': 'SELECT COUNT(*) FROM Customer;'}, {'role': 'use
r', 'content': 'How many records are in table called customer'}, {'role': 'assistant', 'content': 'SELECT C
OUNT(*) FROM Customer;'}, {'role': 'user', 'content': '  \n     List all customers from Canada and their em
ail addresses:\n'}, {'role': 'assistant', 'content': "SELECT \n    CustomerId,\n    FirstName,\n    LastNam
e,\n    Email\nFROM \n    Customer\nWHERE \n    Country = 'Canada'\nORDER BY \n    LastName, FirstName;"},
{'role': 'user', 'content': '  \n     Find the customer with the most invoices \n'}]
Using model claude-3-5-sonnet-20240620 for 1486.0 tokens (approx)
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    COUNT(Invoice.InvoiceId) AS InvoiceCount
FROM
    Customer
JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
GROUP BY
    Customer.CustomerId
ORDER BY
    InvoiceCount DESC
LIMIT 1;
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    COUNT(Invoice.InvoiceId) AS InvoiceCount
FROM
    Customer
JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
GROUP BY
    Customer.CustomerId
ORDER BY
    InvoiceCount DESC
LIMIT 1;
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    COUNT(Invoice.InvoiceId) AS InvoiceCount
FROM
    Customer
JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
GROUP BY
    Customer.CustomerId
ORDER BY
    InvoiceCount DESC
LIMIT 1;
   CustomerId FirstName   LastName  InvoiceCount
0           1      Luís  Gonçalves             7
Using model claude-3-5-sonnet-20240620 for 245.5 tokens (approx)
```



Most Invoices: Luís Gonçalves

7

```
Out[36]: ('SELECT \n    Customer.CustomerId,\n    Customer.FirstName,\n    Customer.LastName,\n    COUNT(Invoice.In
         voiceId) AS InvoiceCount\nFROM \n    Customer\nJOIN \n    Invoice ON Customer.CustomerId = Invoice.Custome
         rId\nGROUP BY \n    Customer.CustomerId\nORDER BY \n    InvoiceCount DESC\nLIMIT 1;',
             CustomerId FirstName   LastName  InvoiceCount
          0           1      Luís  Gonçalves             7,
          Figure({
              'data': [{'domain': {'x': [0, 1], 'y': [0, 1]},
                        'mode': 'number',
                        'title': {'text': 'Most Invoices: Luís Gonçalves'},
                        'type': 'indicator',
                        'value': 7}],
              'layout': {'template': '...'}
          }))
```

```
In [ ]:
```

Advanced SQL questions

```
In [37]: question = """
              Find the customer who bought the most albums in total quantity (across all invoices):
         """

         vn.ask(question=question)
```

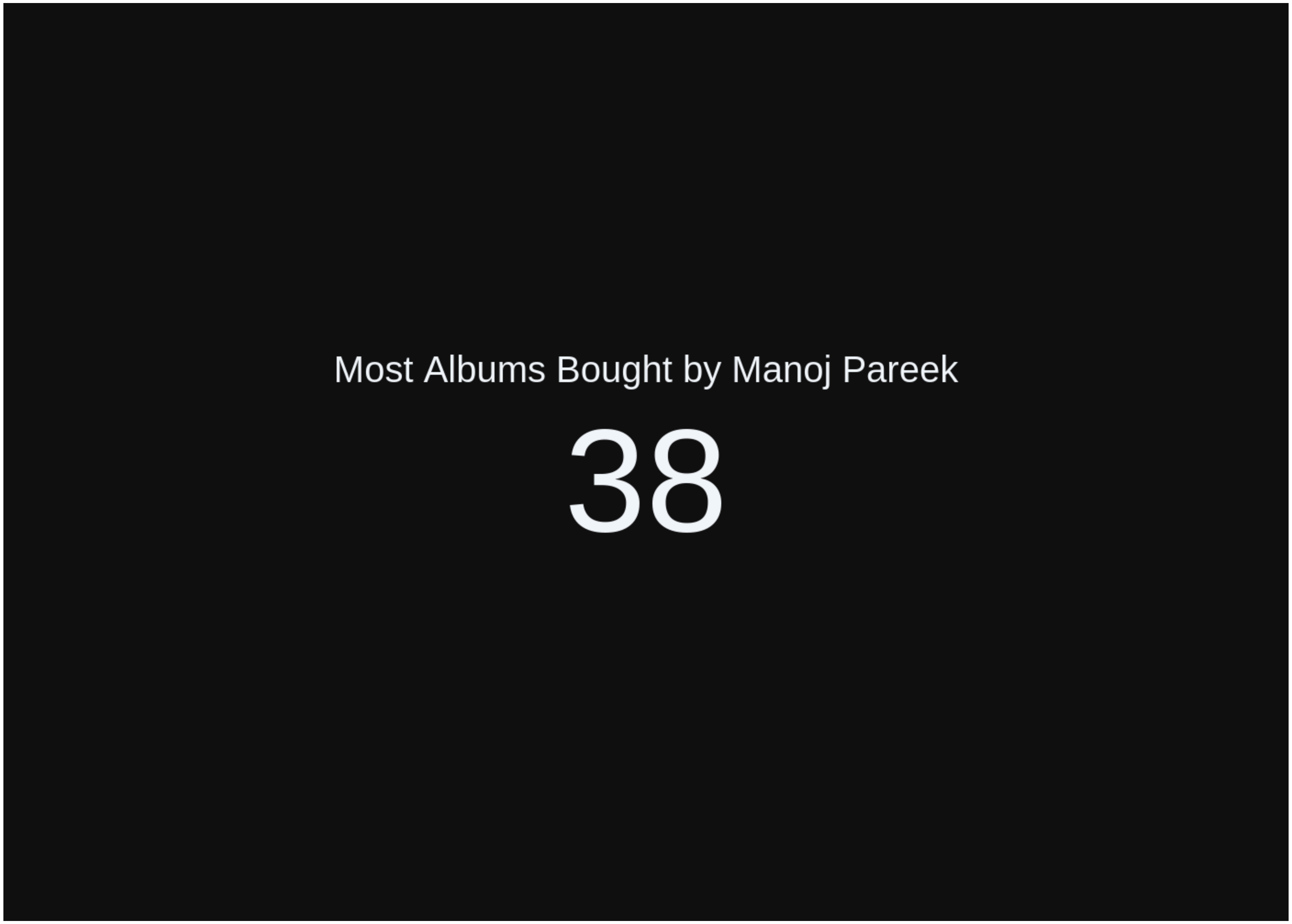
```
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
[{'role': 'system', 'content': "You are a SQLite expert. Please help to generate a SQL query to answer the
question. Your response should ONLY be based on the given context and follow the response guidelines and fo
rmat instructions. \n===Tables \nCREATE TABLE Track\n(\n    TrackId INTEGER  NOT NULL,\n    Name NVARCHAR(2
00)  NOT NULL,\n    AlbumId INTEGER,\n    MediaTypeId INTEGER  NOT NULL,\n    GenreId INTEGER,\n    Compose
r NVARCHAR(220),\n    Milliseconds INTEGER  NOT NULL,\n    Bytes INTEGER,\n    UnitPrice NUMERIC(10,2)  NOT
NULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (TrackId),\n    FOREIGN KEY (AlbumId) REFERENCES Album (AlbumI
d) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n
\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaType
Id) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Album\n(\n    AlbumId INTEGER  NOT NUL
L,\n    Title NVARCHAR(160)  NOT NULL,\n    ArtistId INTEGER  NOT NULL,\n    CONSTRAINT PK_Album PRIMARY KE
Y  (AlbumId),\n    FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON UPDATE
NO ACTION\n)\n\nCREATE INDEX IFK_AlbumArtistId ON Album (ArtistId)\n\nCREATE TABLE InvoiceLine\n(\n    Invo
iceLineId INTEGER  NOT NULL,\n    InvoiceId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    UnitPri
ce NUMERIC(10,2)  NOT NULL,\n    Quantity INTEGER  NOT NULL,\n    CONSTRAINT PK_InvoiceLine PRIMARY KEY  (I
nvoiceLineId),\n    FOREIGN KEY (InvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPD
ATE NO ACTION,\n    FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO
ACTION\n)\n\nCREATE INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)\n\nCREATE TABLE Invoice\n(\n    Inv
oiceId INTEGER  NOT NULL,\n    CustomerId INTEGER  NOT NULL,\n    InvoiceDate DATETIME  NOT NULL,\n    Bill
ingAddress NVARCHAR(70),\n    BillingCity NVARCHAR(40),\n    BillingState NVARCHAR(40),\n    BillingCountry
NVARCHAR(40),\n    BillingPostalCode NVARCHAR(10),\n    Total NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_I
nvoice PRIMARY KEY  (InvoiceId),\n    FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DE
LETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_TrackAlbumId ON Track (AlbumId)\n\nCREATE INDEX I
FK_InvoiceLineInvoiceId ON InvoiceLine (InvoiceId)\n\nCREATE INDEX IFK_InvoiceLineTrackId ON InvoiceLine (T
rackId)\n\nCREATE TABLE PlaylistTrack\n(\n    PlaylistId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NUL
L,\n    CONSTRAINT PK_PlaylistTrack PRIMARY KEY  (PlaylistId, TrackId),\n    FOREIGN KEY (PlaylistId) REFER
ENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (TrackId) REFER
ENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n===Additional Context \n\nIn th
e chinook database invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficien
t, please generate a valid SQL query without any explanations for the question. \n2. If the provided contex
t is almost sufficient but requires knowledge of a specific string in a particular column, please generate
an intermediate SQL query to find the distinct strings in that column. Prepend the query with a comment say
ing intermediate_sql \n3. If the provided context is insufficient, please explain why it can't be generate
d. \n4. Please use the most relevant table(s). \n5. If the question has been asked and answered before, ple
ase repeat the answer exactly as it was given before. \n"}, {'role': 'user', 'content': '  \n     Find the
customer with the most invoices \n'}, {'role': 'assistant', 'content': 'SELECT \n    Customer.CustomerId,\n
Customer.FirstName,\n    Customer.LastName,\n    COUNT(Invoice.InvoiceId) AS InvoiceCount\nFROM \n    Custo
mer\nJOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId\nGROUP BY \n    Customer.CustomerId\nOR
DER BY \n    InvoiceCount DESC\nLIMIT 1;'}, {'role': 'user', 'content': '  \n    Get the total number of in
voices for each customer\n'}, {'role': 'assistant', 'content': 'SELECT \n    Customer.CustomerId,\n    Cust
omer.FirstName,\n    Customer.LastName,\n    COUNT(Invoice.InvoiceId) AS TotalInvoices\nFROM \n    Customer
\nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId\nGROUP BY \n    Customer.CustomerId\n
ORDER BY \n    TotalInvoices DESC;'}, {'role': 'user', 'content': '  \n    Get the top 10 most popular arti
sts (based on the number of tracks):\n'}, {'role': 'assistant', 'content': 'SELECT \n    Artist.ArtistId,\n
Artist.Name,\n    COUNT(Track.TrackId) AS TrackCount\nFROM \n    Artist\nJOIN \n    Album ON Artist.ArtistI
d = Album.ArtistId\nJOIN \n    Track ON Album.AlbumId = Track.AlbumId\nGROUP BY \n    Artist.ArtistId\nORDE
R BY \n    TrackCount DESC\nLIMIT 10;'}, {'role': 'user', 'content': '  \n    Find all invoices since 2010
and the total amount invoiced:\n'}, {'role': 'assistant', 'content': "SELECT \n    COUNT(*) AS InvoiceCoun
t,\n    SUM(Total) AS TotalAmount\nFROM \n    Invoice\nWHERE \n    InvoiceDate >= '2010-01-01';"}, {'role':
'user', 'content': '  \n    List all invoices with a total exceeding $10:\n'}, {'role': 'assistant', 'conte
nt': 'SELECT \n    InvoiceId, \n    CustomerId, \n    InvoiceDate, \n    Total\nFROM \n    Invoice\nWHERE
\n    Total > 10\nORDER BY \n    Total DESC;'}, {'role': 'user', 'content': '  \n    Find the top 5 most ex
pensive tracks (based on unit price):\n'}, {'role': 'assistant', 'content': 'SELECT \n    TrackId,\n    Nam
e,\n    UnitPrice\nFROM \n    Track\nORDER BY \n    UnitPrice DESC\nLIMIT 5;'}, {'role': 'user', 'content':
'  \n    Get the average invoice total for each customer:\n'}, {'role': 'assistant', 'content': 'SELECT \n
Customer.CustomerId,\n    Customer.FirstName,\n    Customer.LastName,\n    AVG(Invoice.Total) AS AverageInv
oiceTotal\nFROM \n    Customer\nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId\nGROUP
BY \n    Customer.CustomerId\nORDER BY \n    AverageInvoiceTotal DESC;'}, {'role': 'user', 'content': '  \n
Find the total number of invoices per country:\n'}, {'role': 'assistant', 'content': 'SELECT \n    BillingC
ountry, \n    COUNT(*) AS TotalInvoices\nFROM \n    Invoice\nGROUP BY \n    BillingCountry\nORDER BY \n
TotalInvoices DESC;'}, {'role': 'user', 'content': '  \n    List all albums and their corresponding artist
names  \n'}, {'role': 'assistant', 'content': 'SELECT \n    Album.Title AS AlbumTitle, \n    Artist.Name AS
ArtistName\nFROM \n    Album\nJOIN \n    Artist ON Album.ArtistId = Artist.ArtistId\nORDER BY \n    Artist.
Name, Album.Title;'}, {'role': 'user', 'content': '  \n    List all genres and the number of tracks in each
genre:\n'}, {'role': 'assistant', 'content': 'SELECT \n    Genre.Name AS GenreName,\n    COUNT(Track.TrackI
d) AS TrackCount\nFROM \n    Genre\nLEFT JOIN \n    Track ON Genre.GenreId = Track.GenreId\nGROUP BY \n
Genre.GenreId\nORDER BY \n    TrackCount DESC;'}, {'role': 'user', 'content': '  \n     Find the customer w
ho bought the most albums in total quantity (across all invoices): \n'}]
Using model claude-3-5-sonnet-20240620 for 1533.75 tokens (approx)
SELECT
    c.CustomerId,
    c.FirstName,
    c.LastName,
    SUM(il.Quantity) AS TotalAlbumsBought
FROM
    Customer c
JOIN Invoice i ON c.CustomerId = i.CustomerId
JOIN InvoiceLine il ON i.InvoiceId = il.InvoiceId
JOIN Track t ON il.TrackId = t.TrackId
WHERE
    t.AlbumId IS NOT NULL
GROUP BY
    c.CustomerId
ORDER BY
    TotalAlbumsBought DESC
```

```
LIMIT 1;
SELECT
    c.CustomerId,
    c.FirstName,
    c.LastName,
    SUM(il.Quantity) AS TotalAlbumsBought
FROM
    Customer c
JOIN Invoice i ON c.CustomerId = i.CustomerId
JOIN InvoiceLine il ON i.InvoiceId = il.InvoiceId
JOIN Track t ON il.TrackId = t.TrackId
WHERE
    t.AlbumId IS NOT NULL
GROUP BY
    c.CustomerId
ORDER BY
    TotalAlbumsBought DESC
LIMIT 1;
```

```
SELECT
    c.CustomerId,
    c.FirstName,
    c.LastName,
    SUM(il.Quantity) AS TotalAlbumsBought
FROM
    Customer c
JOIN Invoice i ON c.CustomerId = i.CustomerId
JOIN InvoiceLine il ON i.InvoiceId = il.InvoiceId
JOIN Track t ON il.TrackId = t.TrackId
WHERE
    t.AlbumId IS NOT NULL
GROUP BY
    c.CustomerId
ORDER BY
    TotalAlbumsBought DESC
LIMIT 1;
```

```
   CustomerId FirstName LastName  TotalAlbumsBought
0          58     Manoj   Pareek                 38
Using model claude-3-5-sonnet-20240620 for 282.25 tokens (approx)
```



Most Albums Bought by Manoj Pareek

38

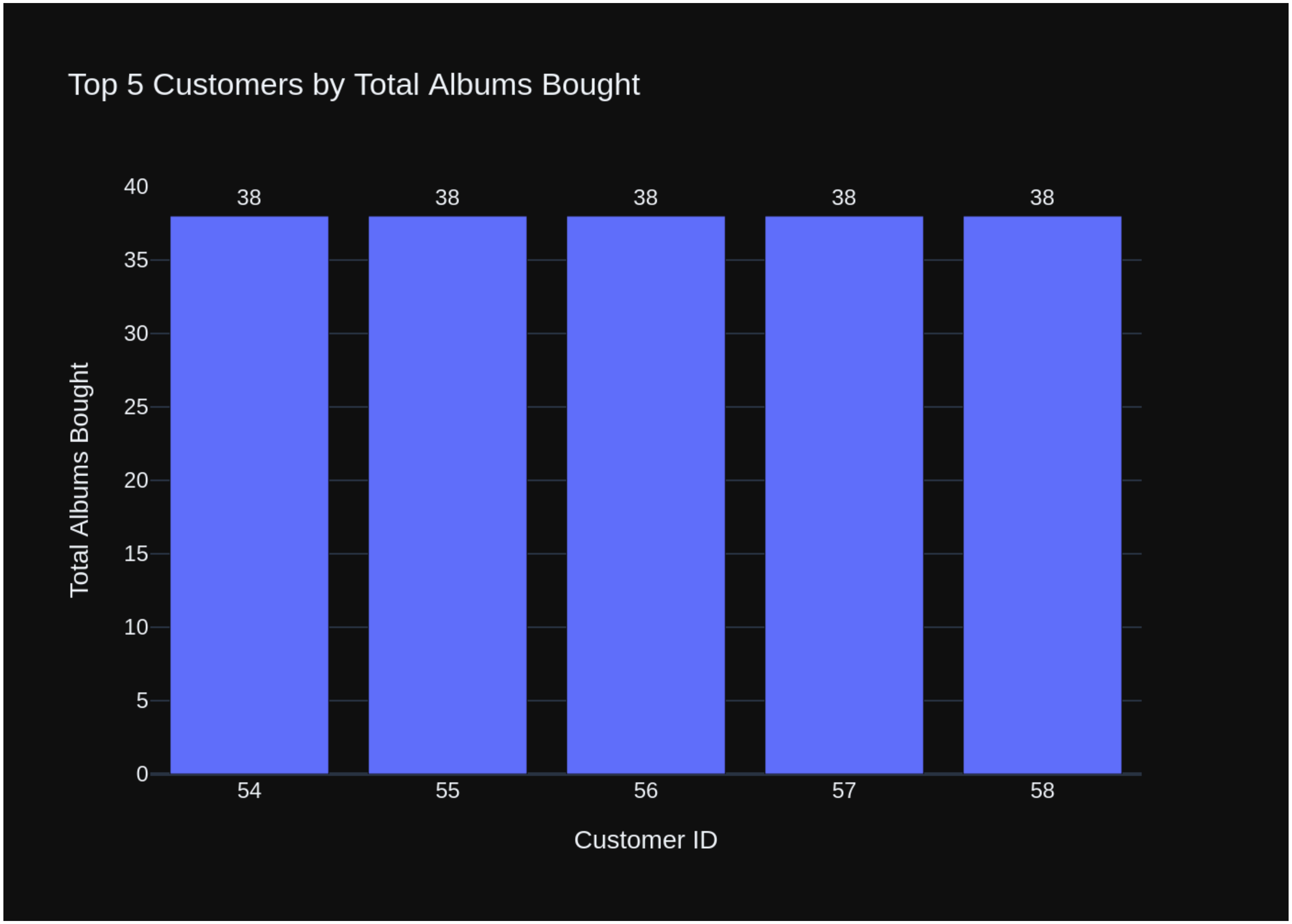
Out[37]:
```
('SELECT \n    c.CustomerId,\n    c.FirstName,\n    c.LastName,\n    SUM(il.Quantity) AS TotalAlbumsBought
\nFROM \n    Customer c\nJOIN Invoice i ON c.CustomerId = i.CustomerId\nJOIN InvoiceLine il ON i.InvoiceId
= il.InvoiceId\nJOIN Track t ON il.TrackId = t.TrackId\nWHERE \n    t.AlbumId IS NOT NULL\nGROUP BY \n
c.CustomerId\nORDER BY \n    TotalAlbumsBought DESC\nLIMIT 1;',
    CustomerId FirstName LastName  TotalAlbumsBought
 0          58     Manoj   Pareek                 38,
 Figure({
     'data': [{'domain': {'x': [0, 1], 'y': [0, 1]},
               'mode': 'number',
               'title': {'text': 'Most Albums Bought by Manoj Pareek'},
               'type': 'indicator',
               'value': 38}],
     'layout': {'template': '...'}
 }))
```

In [38]:
```python
question = """
     Find the top 5 customer who bought the most albums in total quantity (across all invoices):
"""

vn.ask(question=question)
```

```
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
[{'role': 'system', 'content': "You are a SQLite expert. Please help to generate a SQL query to answer the
question. Your response should ONLY be based on the given context and follow the response guidelines and fo
rmat instructions. \n===Tables \nCREATE TABLE Track\n(\n    TrackId INTEGER  NOT NULL,\n    Name NVARCHAR(2
00)  NOT NULL,\n    AlbumId INTEGER,\n    MediaTypeId INTEGER  NOT NULL,\n    GenreId INTEGER,\n    Compose
r NVARCHAR(220),\n    Milliseconds INTEGER  NOT NULL,\n    Bytes INTEGER,\n    UnitPrice NUMERIC(10,2)  NOT
NULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (TrackId),\n    FOREIGN KEY (AlbumId) REFERENCES Album (AlbumI
d) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n
\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaType
Id) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Album\n(\n    AlbumId INTEGER  NOT NUL
L,\n    Title NVARCHAR(160)  NOT NULL,\n    ArtistId INTEGER  NOT NULL,\n    CONSTRAINT PK_Album PRIMARY KE
Y  (AlbumId),\n    FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON UPDATE
NO ACTION\n)\n\nCREATE INDEX IFK_AlbumArtistId ON Album (ArtistId)\n\nCREATE TABLE InvoiceLine\n(\n    Invo
iceLineId INTEGER  NOT NULL,\n    InvoiceId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    UnitPri
ce NUMERIC(10,2)  NOT NULL,\n    Quantity INTEGER  NOT NULL,\n    CONSTRAINT PK_InvoiceLine PRIMARY KEY  (I
nvoiceLineId),\n    FOREIGN KEY (InvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPD
ATE NO ACTION,\n    FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO
ACTION\n)\n\nCREATE TABLE Invoice\n(\n    InvoiceId INTEGER  NOT NULL,\n    CustomerId INTEGER  NOT NULL,\n
InvoiceDate DATETIME  NOT NULL,\n    BillingAddress NVARCHAR(70),\n    BillingCity NVARCHAR(40),\n    Billi
ngState NVARCHAR(40),\n    BillingCountry NVARCHAR(40),\n    BillingPostalCode NVARCHAR(10),\n    Total NUM
ERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Invoice PRIMARY KEY  (InvoiceId),\n    FOREIGN KEY (CustomerId) RE
FERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_Invoice
CustomerId ON Invoice (CustomerId)\n\nCREATE INDEX IFK_TrackAlbumId ON Track (AlbumId)\n\nCREATE INDEX IFK_
InvoiceLineTrackId ON InvoiceLine (TrackId)\n\nCREATE INDEX IFK_InvoiceLineInvoiceId ON InvoiceLine (Invoic
eId)\n\nCREATE TABLE Artist\n(\n    ArtistId INTEGER  NOT NULL,\n    Name NVARCHAR(120),\n    CONSTRAINT PK
_Artist PRIMARY KEY  (ArtistId)\n)\n\n\n===Additional Context \n\nIn the chinook database invoice means ord
er\n\n===Response Guidelines \n1. If the provided context is sufficient, please generate a valid SQL query
without any explanations for the question. \n2. If the provided context is almost sufficient but requires k
nowledge of a specific string in a particular column, please generate an intermediate SQL query to find the
distinct strings in that column. Prepend the query with a comment saying intermediate_sql \n3. If the provi
ded context is insufficient, please explain why it can't be generated. \n4. Please use the most relevant ta
ble(s). \n5. If the question has been asked and answered before, please repeat the answer exactly as it was
given before. \n"}, {'role': 'user', 'content': '  \n     Find the customer who bought the most albums in t
otal quantity (across all invoices): \n'}, {'role': 'assistant', 'content': 'SELECT \n    c.CustomerId,\n
c.FirstName,\n    c.LastName,\n    SUM(il.Quantity) AS TotalAlbumsBought\nFROM \n    Customer c\nJOIN Invoi
ce i ON c.CustomerId = i.CustomerId\nJOIN InvoiceLine il ON i.InvoiceId = il.InvoiceId\nJOIN Track t ON il.
TrackId = t.TrackId\nWHERE \n    t.AlbumId IS NOT NULL\nGROUP BY \n    c.CustomerId\nORDER BY \n    TotalAl
bumsBought DESC\nLIMIT 1;'}, {'role': 'user', 'content': '  \n    Get the top 10 most popular artists (base
d on the number of tracks):\n'}, {'role': 'assistant', 'content': 'SELECT \n    Artist.ArtistId,\n    Artis
t.Name,\n    COUNT(Track.TrackId) AS TrackCount\nFROM \n    Artist\nJOIN \n    Album ON Artist.ArtistId = A
lbum.ArtistId\nJOIN \n    Track ON Album.AlbumId = Track.AlbumId\nGROUP BY \n    Artist.ArtistId\nORDER BY
\n    TrackCount DESC\nLIMIT 10;'}, {'role': 'user', 'content': '  \n     Find the customer with the most i
nvoices \n'}, {'role': 'assistant', 'content': 'SELECT \n    Customer.CustomerId,\n    Customer.FirstNam
e,\n    Customer.LastName,\n    COUNT(Invoice.InvoiceId) AS InvoiceCount\nFROM \n    Customer\nJOIN \n    I
nvoice ON Customer.CustomerId = Invoice.CustomerId\nGROUP BY \n    Customer.CustomerId\nORDER BY \n    Invo
iceCount DESC\nLIMIT 1;'}, {'role': 'user', 'content': '  \n    Find the top 5 most expensive tracks (based
on unit price):\n'}, {'role': 'assistant', 'content': 'SELECT \n    TrackId,\n    Name,\n    UnitPrice\nFRO
M \n    Track\nORDER BY \n    UnitPrice DESC\nLIMIT 5;'}, {'role': 'user', 'content': '  \n    List all inv
oices with a total exceeding $10:\n'}, {'role': 'assistant', 'content': 'SELECT \n    InvoiceId, \n    Cust
omerId, \n    InvoiceDate, \n    Total\nFROM \n    Invoice\nWHERE \n    Total > 10\nORDER BY \n    Total DE
SC;'}, {'role': 'user', 'content': '  \n    Get the total number of invoices for each customer\n'}, {'rol
e': 'assistant', 'content': 'SELECT \n    Customer.CustomerId,\n    Customer.FirstName,\n    Customer.LastN
ame,\n    COUNT(Invoice.InvoiceId) AS TotalInvoices\nFROM \n    Customer\nLEFT JOIN \n    Invoice ON Custom
er.CustomerId = Invoice.CustomerId\nGROUP BY \n    Customer.CustomerId\nORDER BY \n    TotalInvoices DES
C;'}, {'role': 'user', 'content': '  \n    Get the average invoice total for each customer:\n'}, {'role':
'assistant', 'content': 'SELECT \n    Customer.CustomerId,\n    Customer.FirstName,\n    Customer.LastNam
e,\n    AVG(Invoice.Total) AS AverageInvoiceTotal\nFROM \n    Customer\nLEFT JOIN \n    Invoice ON Custome
r.CustomerId = Invoice.CustomerId\nGROUP BY \n    Customer.CustomerId\nORDER BY \n    AverageInvoiceTotal D
ESC;'}, {'role': 'user', 'content': '  \n    List all albums and their corresponding artist names  \n'},
{'role': 'assistant', 'content': 'SELECT \n    Album.Title AS AlbumTitle, \n    Artist.Name AS ArtistName\n
FROM \n    Album\nJOIN \n    Artist ON Album.ArtistId = Artist.ArtistId\nORDER BY \n    Artist.Name, Album.
Title;'}, {'role': 'user', 'content': '  \n    Find all invoices since 2010 and the total amount invoice
d:\n'}, {'role': 'assistant', 'content': "SELECT \n    COUNT(*) AS InvoiceCount,\n    SUM(Total) AS TotalAm
ount\nFROM \n    Invoice\nWHERE \n    InvoiceDate >= '2010-01-01';"}, {'role': 'user', 'content': 'what are
the top 5 countries that customers come from?'}, {'role': 'assistant', 'content': 'SELECT Country, COUNT(*)
as CustomerCount\nFROM Customer\nGROUP BY Country\nORDER BY CustomerCount DESC\nLIMIT 5;'}, {'role': 'use
r', 'content': '  \n     Find the top 5 customer who bought the most albums in total quantity (across all i
nvoices):\n'}]
Using model claude-3-5-sonnet-20240620 for 1515.75 tokens (approx)
SELECT
    c.CustomerId,
    c.FirstName,
    c.LastName,
    SUM(il.Quantity) AS TotalAlbumsBought
FROM
    Customer c
JOIN Invoice i ON c.CustomerId = i.CustomerId
JOIN InvoiceLine il ON i.InvoiceId = il.InvoiceId
JOIN Track t ON il.TrackId = t.TrackId
WHERE
    t.AlbumId IS NOT NULL
GROUP BY
    c.CustomerId
ORDER BY
    TotalAlbumsBought DESC
```

```
LIMIT 5;
SELECT
    c.CustomerId,
    c.FirstName,
    c.LastName,
    SUM(il.Quantity) AS TotalAlbumsBought
FROM
    Customer c
JOIN Invoice i ON c.CustomerId = i.CustomerId
JOIN InvoiceLine il ON i.InvoiceId = il.InvoiceId
JOIN Track t ON il.TrackId = t.TrackId
WHERE
    t.AlbumId IS NOT NULL
GROUP BY
    c.CustomerId
ORDER BY
    TotalAlbumsBought DESC
LIMIT 5;
SELECT
    c.CustomerId,
    c.FirstName,
    c.LastName,
    SUM(il.Quantity) AS TotalAlbumsBought
FROM
    Customer c
JOIN Invoice i ON c.CustomerId = i.CustomerId
JOIN InvoiceLine il ON i.InvoiceId = il.InvoiceId
JOIN Track t ON il.TrackId = t.TrackId
WHERE
    t.AlbumId IS NOT NULL
GROUP BY
    c.CustomerId
ORDER BY
    TotalAlbumsBought DESC
LIMIT 5;
   CustomerId FirstName   LastName  TotalAlbumsBought
0          58     Manoj     Pareek                 38
1          57      Luis      Rojas                 38
2          56     Diego  Gutiérrez                 38
3          55      Mark     Taylor                 38
4          54     Steve     Murray                 38
Using model claude-3-5-sonnet-20240620 for 283.5 tokens (approx)
```


Top 5 Customers by Total Albums Bought
Total Albums Bought
Customer ID
0
5
10
15
20
25
30
35
40
38
38
38
38
38
54
55
56
57
58

```
Out[38]: ('SELECT \n    c.CustomerId,\n    c.FirstName,\n    c.LastName,\n    SUM(il.Quantity) AS TotalAlbumsBought
         \nFROM \n    Customer c\nJOIN Invoice i ON c.CustomerId = i.CustomerId\nJOIN InvoiceLine il ON i.InvoiceId
         = il.InvoiceId\nJOIN Track t ON il.TrackId = t.TrackId\nWHERE \n    t.AlbumId IS NOT NULL\nGROUP BY \n
         c.CustomerId\nORDER BY \n    TotalAlbumsBought DESC\nLIMIT 5;',
             CustomerId FirstName   LastName  TotalAlbumsBought
          0          58     Manoj     Pareek                 38
          1          57      Luis      Rojas                 38
          2          56     Diego  Gutiérrez                 38
          3          55      Mark     Taylor                 38
          4          54     Steve     Murray                 38,
          Figure({
              'data': [{'alignmentgroup': 'True',
                        'customdata': array([['Manoj', 'Pareek'],
                                             ['Luis', 'Rojas'],
                                             ['Diego', 'Gutiérrez'],
                                             ['Mark', 'Taylor'],
                                             ['Steve', 'Murray']], dtype=object),
                        'hovertemplate': ('Customer ID=%{x}<br>Total Albu' ... '{customdata[1]}<extra></extra>'),
                        'legendgroup': '',
                        'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
                        'name': '',
                        'offsetgroup': '',
                        'orientation': 'v',
                        'showlegend': False,
                        'text': array([38., 38., 38., 38., 38.]),
                        'textposition': 'outside',
                        'texttemplate': '%{text}',
                        'type': 'bar',
                        'x': array([58, 57, 56, 55, 54]),
                        'xaxis': 'x',
                        'y': array([38, 38, 38, 38, 38]),
                        'yaxis': 'y'}],
              'layout': {'barmode': 'relative',
                         'legend': {'tracegroupgap': 0},
                         'template': '...',
                         'title': {'text': 'Top 5 Customers by Total Albums Bought'},
                         'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'Customer ID'}},
                         'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'Total Albums Bought'}}}
          }))
```

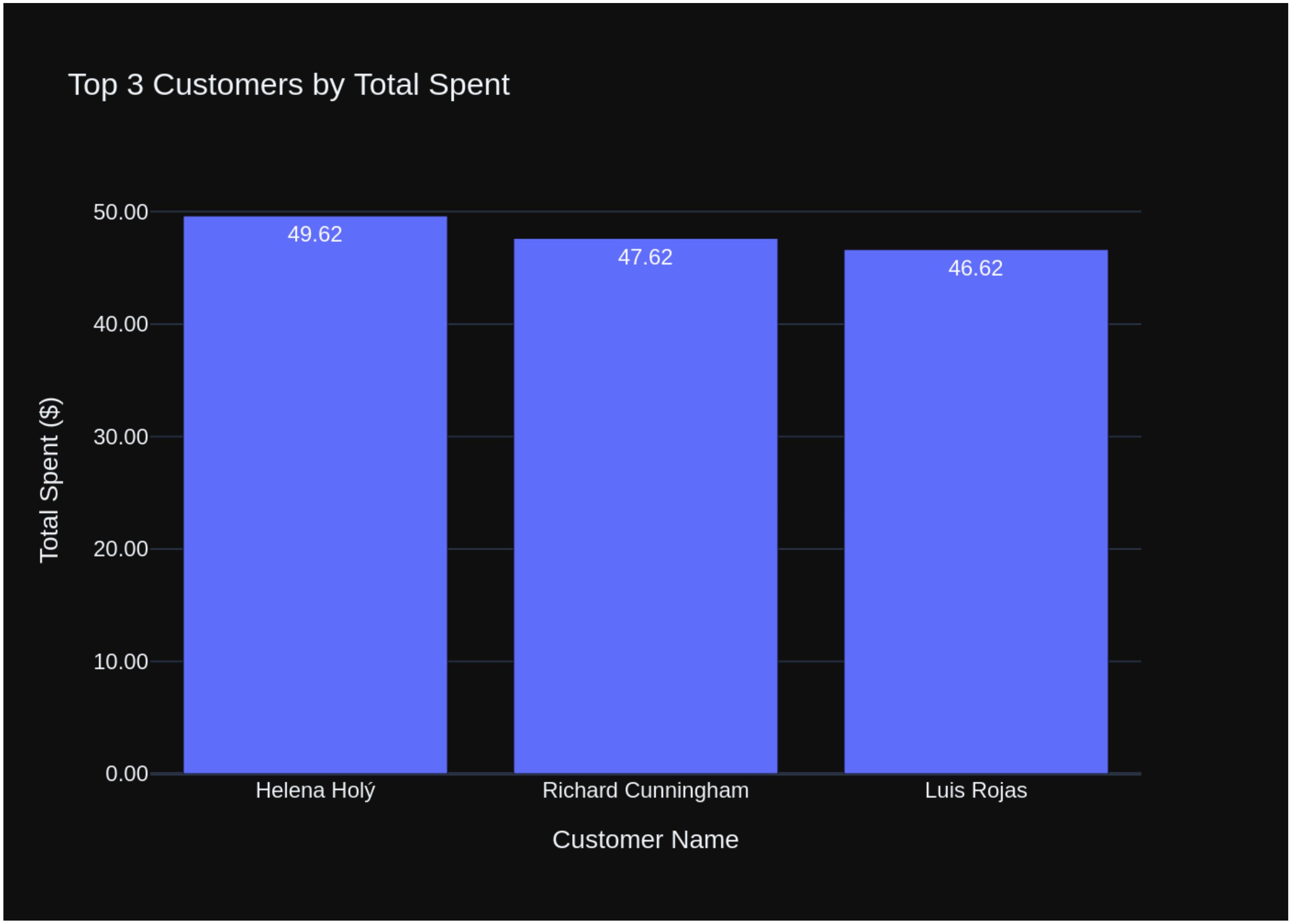
```
In [39]: question = """
              Find the top 3 customers who spent the most money overall:
```

```
"""

vn.ask(question=question)
```

```
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
[{'role': 'system', 'content': "You are a SQLite expert. Please help to generate a SQL query to answer the
question. Your response should ONLY be based on the given context and follow the response guidelines and fo
rmat instructions. \n===Tables \nCREATE TABLE Invoice\n(\n    InvoiceId INTEGER  NOT NULL,\n    CustomerId
INTEGER  NOT NULL,\n    InvoiceDate DATETIME  NOT NULL,\n    BillingAddress NVARCHAR(70),\n    BillingCity
NVARCHAR(40),\n    BillingState NVARCHAR(40),\n    BillingCountry NVARCHAR(40),\n    BillingPostalCode NVAR
CHAR(10),\n    Total NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Invoice PRIMARY KEY  (InvoiceId),\n    FOR
EIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n
CREATE TABLE InvoiceLine\n(\n    InvoiceLineId INTEGER  NOT NULL,\n    InvoiceId INTEGER  NOT NULL,\n    Tr
ackId INTEGER  NOT NULL,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    Quantity INTEGER  NOT NULL,\n    CONS
TRAINT PK_InvoiceLine PRIMARY KEY  (InvoiceLineId),\n    FOREIGN KEY (InvoiceId) REFERENCES Invoice (Invoic
eId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (TrackId) REFERENCES Track (TrackId)
\n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Customer\n(\n    CustomerId INTEGER  NOT N
ULL,\n    FirstName NVARCHAR(40)  NOT NULL,\n    LastName NVARCHAR(20)  NOT NULL,\n    Company NVARCHAR(8
0),\n    Address NVARCHAR(70),\n    City NVARCHAR(40),\n    State NVARCHAR(40),\n    Country NVARCHAR(4
0),\n    PostalCode NVARCHAR(10),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60)
NOT NULL,\n    SupportRepId INTEGER,\n    CONSTRAINT PK_Customer PRIMARY KEY  (CustomerId),\n    FOREIGN KE
Y (SupportRepId) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREAT
E INDEX IFK_CustomerSupportRepId ON Customer (SupportRepId)\n\nCREATE TABLE Track\n(\n    TrackId INTEGER
NOT NULL,\n    Name NVARCHAR(200)  NOT NULL,\n    AlbumId INTEGER,\n    MediaTypeId INTEGER  NOT NULL,\n
GenreId INTEGER,\n    Composer NVARCHAR(220),\n    Milliseconds INTEGER  NOT NULL,\n    Bytes INTEGER,\n
UnitPrice NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (TrackId),\n    FOREIGN KEY (Album
Id) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (GenreId) RE
FERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (MediaTypeId) REFE
RENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_Invoice
CustomerId ON Invoice (CustomerId)\n\nCREATE INDEX IFK_EmployeeReportsTo ON Employee (ReportsTo)\n\nCREATE
TABLE PlaylistTrack\n(\n    PlaylistId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    CONSTRAINT P
K_PlaylistTrack PRIMARY KEY  (PlaylistId, TrackId),\n    FOREIGN KEY (PlaylistId) REFERENCES Playlist (Play
listId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (TrackId) REFERENCES Track (TrackI
d) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Employee\n(\n    EmployeeId INTEGER  NO
T NULL,\n    LastName NVARCHAR(20)  NOT NULL,\n    FirstName NVARCHAR(20)  NOT NULL,\n    Title NVARCHAR(3
0),\n    ReportsTo INTEGER,\n    BirthDate DATETIME,\n    HireDate DATETIME,\n    Address NVARCHAR(70),\n
City NVARCHAR(40),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVARCHAR(10),\n    P
hone NVARCHAR(24),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60),\n    CONSTRAINT PK_Employee PRIMARY KEY
(EmployeeId),\n    FOREIGN KEY (ReportsTo) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UP
DATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceLineTrackId ON InvoiceLine (TrackId)\n\n\n===Additional Contex
t \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided context is
sufficient, please generate a valid SQL query without any explanations for the question. \n2. If the provid
ed context is almost sufficient but requires knowledge of a specific string in a particular column, please
generate an intermediate SQL query to find the distinct strings in that column. Prepend the query with a co
mment saying intermediate_sql \n3. If the provided context is insufficient, please explain why it can't be
generated. \n4. Please use the most relevant table(s). \n5. If the question has been asked and answered bef
ore, please repeat the answer exactly as it was given before. \n"}, {'role': 'user', 'content': '  \n     F
ind the top 5 customer who bought the most albums in total quantity (across all invoices):\n'}, {'role': 'a
ssistant', 'content': 'SELECT \n    c.CustomerId,\n    c.FirstName,\n    c.LastName,\n    SUM(il.Quantity)
AS TotalAlbumsBought\nFROM \n    Customer c\nJOIN Invoice i ON c.CustomerId = i.CustomerId\nJOIN InvoiceLin
e il ON i.InvoiceId = il.InvoiceId\nJOIN Track t ON il.TrackId = t.TrackId\nWHERE \n    t.AlbumId IS NOT NU
LL\nGROUP BY \n    c.CustomerId\nORDER BY \n    TotalAlbumsBought DESC\nLIMIT 5;'}, {'role': 'user', 'conte
nt': '  \n    Find the top 5 most expensive tracks (based on unit price):\n'}, {'role': 'assistant', 'conte
nt': 'SELECT \n    TrackId,\n    Name,\n    UnitPrice\nFROM \n    Track\nORDER BY \n    UnitPrice DESC\nLIM
IT 5;'}, {'role': 'user', 'content': '  \n     Find the customer who bought the most albums in total quanti
ty (across all invoices): \n'}, {'role': 'assistant', 'content': 'SELECT \n    c.CustomerId,\n    c.FirstNa
me,\n    c.LastName,\n    SUM(il.Quantity) AS TotalAlbumsBought\nFROM \n    Customer c\nJOIN Invoice i ON
c.CustomerId = i.CustomerId\nJOIN InvoiceLine il ON i.InvoiceId = il.InvoiceId\nJOIN Track t ON il.TrackId
= t.TrackId\nWHERE \n    t.AlbumId IS NOT NULL\nGROUP BY \n    c.CustomerId\nORDER BY \n    TotalAlbumsBoug
ht DESC\nLIMIT 1;'}, {'role': 'user', 'content': '  \n     Find the customer with the most invoices \n'},
{'role': 'assistant', 'content': 'SELECT \n    Customer.CustomerId,\n    Customer.FirstName,\n    Customer.
LastName,\n    COUNT(Invoice.InvoiceId) AS InvoiceCount\nFROM \n    Customer\nJOIN \n    Invoice ON Custome
r.CustomerId = Invoice.CustomerId\nGROUP BY \n    Customer.CustomerId\nORDER BY \n    InvoiceCount DESC\nLI
MIT 1;'}, {'role': 'user', 'content': '  \n    Get the average invoice total for each customer:\n'}, {'rol
e': 'assistant', 'content': 'SELECT \n    Customer.CustomerId,\n    Customer.FirstName,\n    Customer.LastN
ame,\n    AVG(Invoice.Total) AS AverageInvoiceTotal\nFROM \n    Customer\nLEFT JOIN \n    Invoice ON Custom
er.CustomerId = Invoice.CustomerId\nGROUP BY \n    Customer.CustomerId\nORDER BY \n    AverageInvoiceTotal
DESC;'}, {'role': 'user', 'content': 'what are the top 5 countries that customers come from?'}, {'role': 'a
ssistant', 'content': 'SELECT Country, COUNT(*) as CustomerCount\nFROM Customer\nGROUP BY Country\nORDER BY
CustomerCount DESC\nLIMIT 5;'}, {'role': 'user', 'content': '  \n    Get the top 10 most popular artists (b
ased on the number of tracks):\n'}, {'role': 'assistant', 'content': 'SELECT \n    Artist.ArtistId,\n    Ar
tist.Name,\n    COUNT(Track.TrackId) AS TrackCount\nFROM \n    Artist\nJOIN \n    Album ON Artist.ArtistId
= Album.ArtistId\nJOIN \n    Track ON Album.AlbumId = Track.AlbumId\nGROUP BY \n    Artist.ArtistId\nORDER
BY \n    TrackCount DESC\nLIMIT 10;'}, {'role': 'user', 'content': '  \n    Get the total number of invoice
s for each customer\n'}, {'role': 'assistant', 'content': 'SELECT \n    Customer.CustomerId,\n    Customer.
FirstName,\n    Customer.LastName,\n    COUNT(Invoice.InvoiceId) AS TotalInvoices\nFROM \n    Customer\nLEF
T JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId\nGROUP BY \n    Customer.CustomerId\nORDER
BY \n    TotalInvoices DESC;'}, {'role': 'user', 'content': '  \n    List all invoices with a total exceedi
ng $10:\n'}, {'role': 'assistant', 'content': 'SELECT \n    InvoiceId, \n    CustomerId, \n    InvoiceDate,
\n    Total\nFROM \n    Invoice\nWHERE \n    Total > 10\nORDER BY \n    Total DESC;'}, {'role': 'user', 'co
ntent': 'How many customers are there'}, {'role': 'assistant', 'content': 'SELECT COUNT(*) FROM Custome
r;'}, {'role': 'user', 'content': '  \n     Find the top 3 customers who spent the most money overall:\n'}]
Using model claude-3-5-sonnet-20240620 for 1796.75 tokens (approx)
SELECT
    c.CustomerId,
    c.FirstName,
    c.LastName,
    SUM(i.Total) AS TotalSpent
FROM
```

```
    Customer c
JOIN
    Invoice i ON c.CustomerId = i.CustomerId
GROUP BY
    c.CustomerId
ORDER BY
    TotalSpent DESC
LIMIT 3;
SELECT
    c.CustomerId,
    c.FirstName,
    c.LastName,
    SUM(i.Total) AS TotalSpent
FROM
    Customer c
JOIN
    Invoice i ON c.CustomerId = i.CustomerId
GROUP BY
    c.CustomerId
ORDER BY
    TotalSpent DESC
LIMIT 3;
SELECT
    c.CustomerId,
    c.FirstName,
    c.LastName,
    SUM(i.Total) AS TotalSpent
FROM
    Customer c
JOIN
    Invoice i ON c.CustomerId = i.CustomerId
GROUP BY
    c.CustomerId
ORDER BY
    TotalSpent DESC
LIMIT 3;
   CustomerId FirstName    LastName  TotalSpent
0           6    Helena        Holý       49.62
1          26   Richard  Cunningham       47.62
2          57      Luis       Rojas       46.62
Using model claude-3-5-sonnet-20240620 for 235.5 tokens (approx)
```


Top 3 Customers by Total Spent
Total Spent ($)
50.00
40.00
30.00
20.00
10.00
0.00
49.62
47.62
46.62
Helena Holý
Richard Cunningham
Luis Rojas
Customer Name

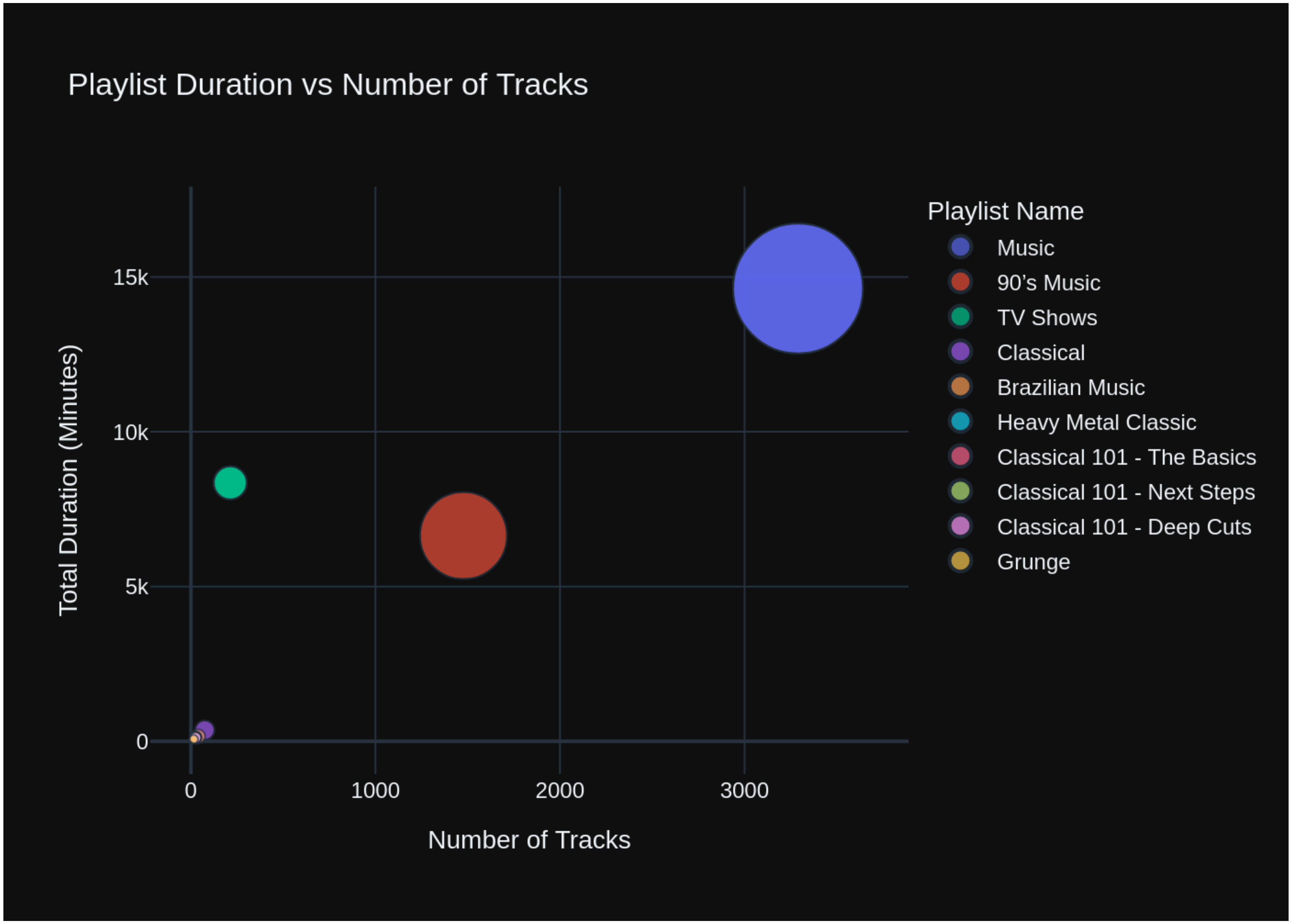
Out[39]:
```
('SELECT \n    c.CustomerId,\n    c.FirstName,\n    c.LastName,\n    SUM(i.Total) AS TotalSpent\nFROM \n
Customer c\nJOIN \n    Invoice i ON c.CustomerId = i.CustomerId\nGROUP BY \n    c.CustomerId\nORDER BY \n
TotalSpent DESC\nLIMIT 3;',
    CustomerId FirstName    LastName  TotalSpent
 0           6    Helena        Holý       49.62
 1          26   Richard  Cunningham       47.62
 2          57      Luis       Rojas       46.62,
 Figure({
     'data': [{'text': array([49.62, 47.62, 46.62]),
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Helena Holý', 'Richard Cunningham', 'Luis Rojas'], dtype=object),
               'y': array([49.62, 47.62, 46.62])}],
     'layout': {'template': '...',
                'title': {'text': 'Top 3 Customers by Total Spent'},
                'xaxis': {'title': {'text': 'Customer Name'}},
                'yaxis': {'tickformat': ',.2f', 'title': {'text': 'Total Spent ($)'}}}
 }))
```

In [40]:
```
question = """
     Get all playlists containing at least 10 tracks and the total duration of those tracks:
"""

vn.ask(question=question)
```

```
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
[{'role': 'system', 'content': "You are a SQLite expert. Please help to generate a SQL query to answer the
question. Your response should ONLY be based on the given context and follow the response guidelines and fo
rmat instructions. \n===Tables \nCREATE INDEX IFK_PlaylistTrackTrackId ON PlaylistTrack (TrackId)\n\nCREATE
TABLE Playlist\n(\n    PlaylistId INTEGER  NOT NULL,\n    Name NVARCHAR(120),\n    CONSTRAINT PK_Playlist P
RIMARY KEY  (PlaylistId)\n)\n\nCREATE TABLE Track\n(\n    TrackId INTEGER  NOT NULL,\n    Name NVARCHAR(20
0)  NOT NULL,\n    AlbumId INTEGER,\n    MediaTypeId INTEGER  NOT NULL,\n    GenreId INTEGER,\n    Composer
NVARCHAR(220),\n    Milliseconds INTEGER  NOT NULL,\n    Bytes INTEGER,\n    UnitPrice NUMERIC(10,2)  NOT N
ULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (TrackId),\n    FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId)
\n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\t
ON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId)
\n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE PlaylistTrack\n(\n    PlaylistId INTEGER
NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    CONSTRAINT PK_PlaylistTrack PRIMARY KEY  (PlaylistId, TrackI
d),\n    FOREIGN KEY (PlaylistId) REFERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO A
CTION,\n    FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION
\n)\n\nCREATE INDEX IFK_TrackGenreId ON Track (GenreId)\n\nCREATE INDEX IFK_TrackAlbumId ON Track (AlbumId)
\n\nCREATE INDEX IFK_TrackMediaTypeId ON Track (MediaTypeId)\n\nCREATE INDEX IFK_AlbumArtistId ON Album (Ar
tistId)\n\nCREATE TABLE Album\n(\n    AlbumId INTEGER  NOT NULL,\n    Title NVARCHAR(160)  NOT NULL,\n    A
rtistId INTEGER  NOT NULL,\n    CONSTRAINT PK_Album PRIMARY KEY  (AlbumId),\n    FOREIGN KEY (ArtistId) REF
ERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceLineT
rackId ON InvoiceLine (TrackId)\n\n\n===Additional Context \n\nIn the chinook database invoice means order
\n\n===Response Guidelines \n1. If the provided context is sufficient, please generate a valid SQL query wi
thout any explanations for the question. \n2. If the provided context is almost sufficient but requires kno
wledge of a specific string in a particular column, please generate an intermediate SQL query to find the d
istinct strings in that column. Prepend the query with a comment saying intermediate_sql \n3. If the provid
ed context is insufficient, please explain why it can't be generated. \n4. Please use the most relevant tab
le(s). \n5. If the question has been asked and answered before, please repeat the answer exactly as it was
given before. \n"}, {'role': 'user', 'content': '  \n    List all genres and the number of tracks in each g
enre:\n'}, {'role': 'assistant', 'content': 'SELECT \n    Genre.Name AS GenreName,\n    COUNT(Track.TrackI
d) AS TrackCount\nFROM \n    Genre\nLEFT JOIN \n    Track ON Genre.GenreId = Track.GenreId\nGROUP BY \n
Genre.GenreId\nORDER BY \n    TrackCount DESC;'}, {'role': 'user', 'content': '  \n    Get the top 10 most
popular artists (based on the number of tracks):\n'}, {'role': 'assistant', 'content': 'SELECT \n    Artis
t.ArtistId,\n    Artist.Name,\n    COUNT(Track.TrackId) AS TrackCount\nFROM \n    Artist\nJOIN \n    Album
ON Artist.ArtistId = Album.ArtistId\nJOIN \n    Track ON Album.AlbumId = Track.AlbumId\nGROUP BY \n    Arti
st.ArtistId\nORDER BY \n    TrackCount DESC\nLIMIT 10;'}, {'role': 'user', 'content': '  \n     Find the to
p 5 customer who bought the most albums in total quantity (across all invoices):\n'}, {'role': 'assistant',
'content': 'SELECT \n    c.CustomerId,\n    c.FirstName,\n    c.LastName,\n    SUM(il.Quantity) AS TotalAlb
umsBought\nFROM \n    Customer c\nJOIN Invoice i ON c.CustomerId = i.CustomerId\nJOIN InvoiceLine il ON i.I
nvoiceId = il.InvoiceId\nJOIN Track t ON il.TrackId = t.TrackId\nWHERE \n    t.AlbumId IS NOT NULL\nGROUP B
Y \n    c.CustomerId\nORDER BY \n    TotalAlbumsBought DESC\nLIMIT 5;'}, {'role': 'user', 'content': '  \n
Find the customer who bought the most albums in total quantity (across all invoices): \n'}, {'role': 'assis
tant', 'content': 'SELECT \n    c.CustomerId,\n    c.FirstName,\n    c.LastName,\n    SUM(il.Quantity) AS T
otalAlbumsBought\nFROM \n    Customer c\nJOIN Invoice i ON c.CustomerId = i.CustomerId\nJOIN InvoiceLine il
ON i.InvoiceId = il.InvoiceId\nJOIN Track t ON il.TrackId = t.TrackId\nWHERE \n    t.AlbumId IS NOT NULL\nG
ROUP BY \n    c.CustomerId\nORDER BY \n    TotalAlbumsBought DESC\nLIMIT 1;'}, {'role': 'user', 'content':
'  \n    List all albums and their corresponding artist names  \n'}, {'role': 'assistant', 'content': 'SELE
CT \n    Album.Title AS AlbumTitle, \n    Artist.Name AS ArtistName\nFROM \n    Album\nJOIN \n    Artist ON
Album.ArtistId = Artist.ArtistId\nORDER BY \n    Artist.Name, Album.Title;'}, {'role': 'user', 'content': '
\n    Find all tracks with a name containing "What" (case-insensitive)\n'}, {'role': 'assistant', 'conten
t': "SELECT TrackId, Name\nFROM Track\nWHERE Name LIKE '%What%' COLLATE NOCASE;"}, {'role': 'user', 'conten
t': '  \n    Find the top 5 most expensive tracks (based on unit price):\n'}, {'role': 'assistant', 'conten
t': 'SELECT \n    TrackId,\n    Name,\n    UnitPrice\nFROM \n    Track\nORDER BY \n    UnitPrice DESC\nLIMI
T 5;'}, {'role': 'user', 'content': '  \n    Find all invoices since 2010 and the total amount invoice
d:\n'}, {'role': 'assistant', 'content': "SELECT \n    COUNT(*) AS InvoiceCount,\n    SUM(Total) AS TotalAm
ount\nFROM \n    Invoice\nWHERE \n    InvoiceDate >= '2010-01-01';"}, {'role': 'user', 'content': '  \n
List all invoices with a total exceeding $10:\n'}, {'role': 'assistant', 'content': 'SELECT \n    InvoiceI
d, \n    CustomerId, \n    InvoiceDate, \n    Total\nFROM \n    Invoice\nWHERE \n    Total > 10\nORDER BY
\n    Total DESC;'}, {'role': 'user', 'content': 'Show me a list of tables in the SQLite database'}, {'rol
e': 'assistant', 'content': "SELECT name FROM sqlite_master WHERE type='table';"}, {'role': 'user', 'conten
t': '  \n     Get all playlists containing at least 10 tracks and the total duration of those tracks:\n'}]
Using model claude-3-5-sonnet-20240620 for 1334.0 tokens (approx)
SELECT
    p.PlaylistId,
    p.Name AS PlaylistName,
    COUNT(pt.TrackId) AS TrackCount,
    SUM(t.Milliseconds) / 60000 AS TotalDurationMinutes
FROM
    Playlist p
JOIN
    PlaylistTrack pt ON p.PlaylistId = pt.PlaylistId
JOIN
    Track t ON pt.TrackId = t.TrackId
GROUP BY
    p.PlaylistId
HAVING
    COUNT(pt.TrackId) >= 10
ORDER BY
    TrackCount DESC;
SELECT
    p.PlaylistId,
    p.Name AS PlaylistName,
    COUNT(pt.TrackId) AS TrackCount,
    SUM(t.Milliseconds) / 60000 AS TotalDurationMinutes
FROM
    Playlist p
```

```
JOIN
    PlaylistTrack pt ON p.PlaylistId = pt.PlaylistId
JOIN
    Track t ON pt.TrackId = t.TrackId
GROUP BY
    p.PlaylistId
HAVING
    COUNT(pt.TrackId) >= 10
ORDER BY
    TrackCount DESC;
SELECT
    p.PlaylistId,
    p.Name AS PlaylistName,
    COUNT(pt.TrackId) AS TrackCount,
    SUM(t.Milliseconds) / 60000 AS TotalDurationMinutes
FROM
    Playlist p
JOIN
    PlaylistTrack pt ON p.PlaylistId = pt.PlaylistId
JOIN
    Track t ON pt.TrackId = t.TrackId
GROUP BY
    p.PlaylistId
HAVING
    COUNT(pt.TrackId) >= 10
ORDER BY
    TrackCount DESC;
    PlaylistId                PlaylistName  TrackCount  TotalDurationMinutes
0            8                       Music        3290                 14628
1            1                       Music        3290                 14628
2            5                  90’s Music        1477                  6645
3           10                    TV Shows         213                  8351
4            3                    TV Shows         213                  8351
5           12                   Classical          75                   362
6           11             Brazilian Music          39                   158
7           17         Heavy Metal Classic          26                   136
8           15  Classical 101 - The Basics          25                   123
9           14  Classical 101 - Next Steps          25                   126
10          13   Classical 101 - Deep Cuts          25                   112
11          16                      Grunge          15                    68
Using model claude-3-5-sonnet-20240620 for 286.0 tokens (approx)
```


Playlist Duration vs Number of Tracks
Total Duration (Minutes)
15k
10k
5k
0
0
1000
2000
3000
Number of Tracks
Playlist Name
Music
90’s Music
TV Shows
Classical
Brazilian Music
Heavy Metal Classic
Classical 101 - The Basics
Classical 101 - Next Steps
Classical 101 - Deep Cuts
Grunge

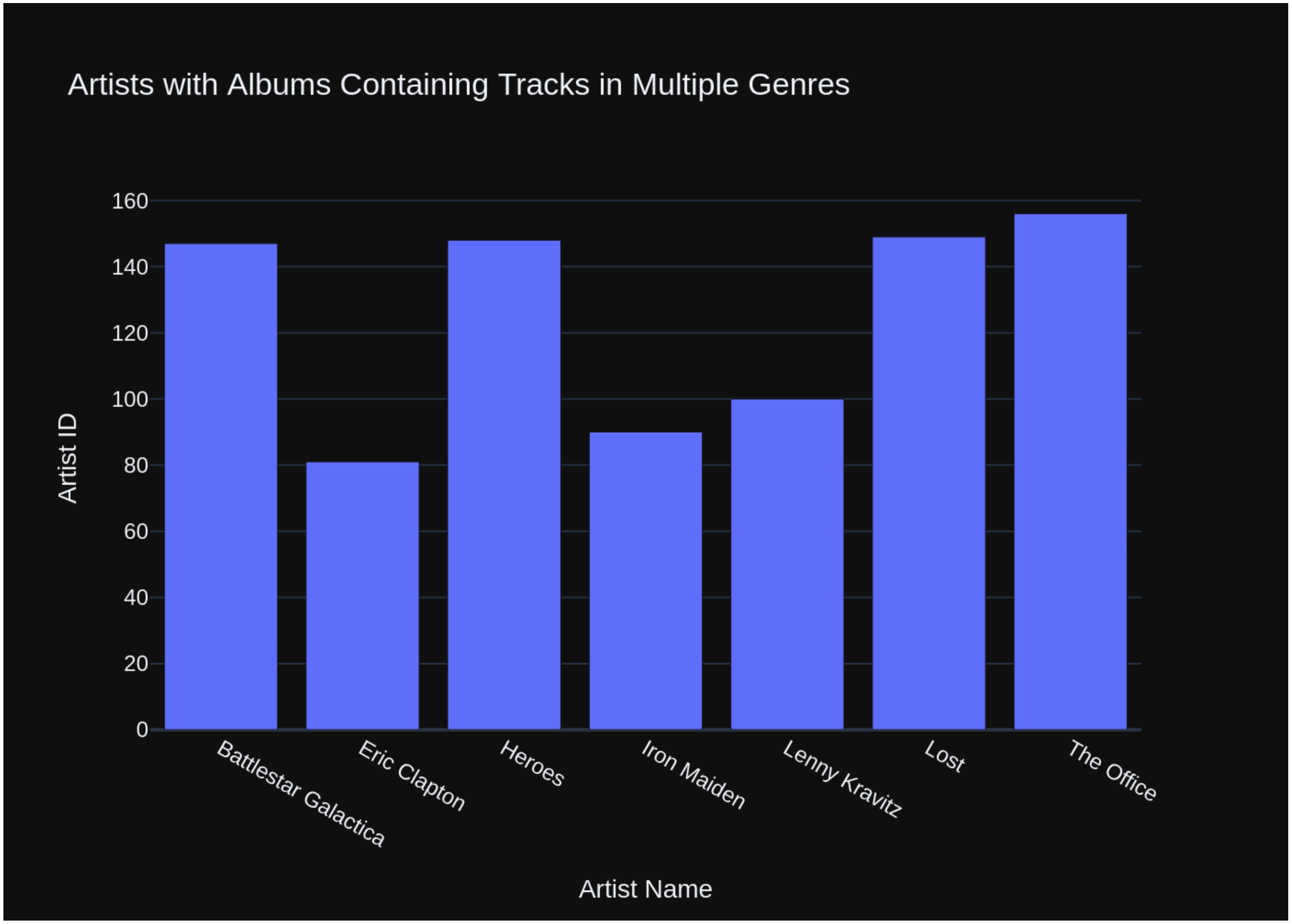
```
Out[40]: ('SELECT \n    p.PlaylistId,\n    p.Name AS PlaylistName,\n    COUNT(pt.TrackId) AS TrackCount,\n    SUM
         (t.Milliseconds) / 60000 AS TotalDurationMinutes\nFROM \n    Playlist p\nJOIN \n    PlaylistTrack pt ON p.
         PlaylistId = pt.PlaylistId\nJOIN \n    Track t ON pt.TrackId = t.TrackId\nGROUP BY \n    p.PlaylistId\nHAV
         ING \n    COUNT(pt.TrackId) >= 10\nORDER BY \n    TrackCount DESC;',
              PlaylistId                PlaylistName  TrackCount  TotalDurationMinutes
          0            8                       Music        3290                 14628
          1            1                       Music        3290                 14628
          2            5                  90’s Music        1477                  6645
          3           10                    TV Shows         213                  8351
          4            3                    TV Shows         213                  8351
          5           12                   Classical          75                   362
          6           11             Brazilian Music          39                   158
          7           17         Heavy Metal Classic          26                   136
          8           15  Classical 101 - The Basics          25                   123
          9           14  Classical 101 - Next Steps          25                   126
          10          13   Classical 101 - Deep Cuts          25                   112
          11          16                      Grunge          15                    68,
          Figure({
              'data': [{'customdata': array([[8],
                                             [1]]),
                        'hovertemplate': ('PlaylistName=Music<br>Number o' ... '{customdata[0]}<extra></extra>'),
                        'legendgroup': 'Music',
                        'marker': {'color': '#636efa',
                                   'size': array([3290, 3290]),
                                   'sizemode': 'area',
                                   'sizeref': 1.316,
                                   'symbol': 'circle'},
                        'mode': 'markers',
                        'name': 'Music',
                        'orientation': 'v',
                        'showlegend': True,
                        'type': 'scatter',
                        'x': array([3290, 3290]),
                        'xaxis': 'x',
                        'y': array([14628, 14628]),
                        'yaxis': 'y'},
                       {'customdata': array([[5]]),
                        'hovertemplate': ('PlaylistName=90’s Music<br>Num' ... '{customdata[0]}<extra></extra>'),
                        'legendgroup': '90’s Music',
                        'marker': {'color': '#EF553B',
                                   'size': array([1477]),
                                   'sizemode': 'area',
                                   'sizeref': 1.316,
                                   'symbol': 'circle'},
                        'mode': 'markers',
                        'name': '90’s Music',
                        'orientation': 'v',
                        'showlegend': True,
                        'type': 'scatter',
                        'x': array([1477]),
                        'xaxis': 'x',
                        'y': array([6645]),
                        'yaxis': 'y'},
                       {'customdata': array([[10],
                                             [ 3]]),
                        'hovertemplate': ('PlaylistName=TV Shows<br>Numbe' ... '{customdata[0]}<extra></extra>'),
                        'legendgroup': 'TV Shows',
                        'marker': {'color': '#00cc96',
                                   'size': array([213, 213]),
                                   'sizemode': 'area',
                                   'sizeref': 1.316,
                                   'symbol': 'circle'},
                        'mode': 'markers',
                        'name': 'TV Shows',
                        'orientation': 'v',
                        'showlegend': True,
                        'type': 'scatter',
                        'x': array([213, 213]),
                        'xaxis': 'x',
                        'y': array([8351, 8351]),
                        'yaxis': 'y'},
                       {'customdata': array([[12]]),
                        'hovertemplate': ('PlaylistName=Classical<br>Numb' ... '{customdata[0]}<extra></extra>'),
                        'legendgroup': 'Classical',
                        'marker': {'color': '#ab63fa',
                                   'size': array([75]),
                                   'sizemode': 'area',
                                   'sizeref': 1.316,
                                   'symbol': 'circle'},
                        'mode': 'markers',
                        'name': 'Classical',
                        'orientation': 'v',
                        'showlegend': True,
                        'type': 'scatter',
                        'x': array([75]),
                        'xaxis': 'x',
                        'y': array([362]),
                        'yaxis': 'y'},
                       {'customdata': array([[11]]),
                        'hovertemplate': ('PlaylistName=Brazilian Music<b' ... '{customdata[0]}<extra></extra>'),
                        'legendgroup': 'Brazilian Music',
                        'marker': {'color': '#FFA15A',
                                   'size': array([39]),
                                   'sizemode': 'area',
                                   'sizeref': 1.316,
                                   'symbol': 'circle'},
                        'mode': 'markers',
                        'name': 'Brazilian Music',
                        'orientation': 'v',
                        'showlegend': True,
                        'type': 'scatter',
                        'x': array([39]),
                        'xaxis': 'x',
                        'y': array([158]),
                        'yaxis': 'y'},
                       {'customdata': array([[17]]),
                        'hovertemplate': ('PlaylistName=Heavy Metal Class' ... '{customdata[0]}<extra></extra>'),
                        'legendgroup': 'Heavy Metal Classic',
                        'marker': {'color': '#19d3f3',
                                   'size': array([26]),
                                   'sizemode': 'area',
                                   'sizeref': 1.316,
                                   'symbol': 'circle'},
                        'mode': 'markers',
                        'name': 'Heavy Metal Classic',
                        'orientation': 'v',
                        'showlegend': True,
                        'type': 'scatter',
                        'x': array([26]),
                        'xaxis': 'x',
                        'y': array([136]),
                        'yaxis': 'y'},
                       {'customdata': array([[15]]),
                        'hovertemplate': ('PlaylistName=Classical 101 - T' ... '{customdata[0]}<extra></extra>'),
                        'legendgroup': 'Classical 101 - The Basics',
                        'marker': {'color': '#FF6692',
                                   'size': array([25]),
                                   'sizemode': 'area',
                                   'sizeref': 1.316,
                                   'symbol': 'circle'},
                        'mode': 'markers',
                        'name': 'Classical 101 - The Basics',
                        'orientation': 'v',
                        'showlegend': True,
                        'type': 'scatter',
                        'x': array([25]),
                        'xaxis': 'x',
                        'y': array([123]),
                        'yaxis': 'y'},
                       {'customdata': array([[14]]),
                        'hovertemplate': ('PlaylistName=Classical 101 - N' ... '{customdata[0]}<extra></extra>'),
                        'legendgroup': 'Classical 101 - Next Steps',
                        'marker': {'color': '#B6E880',
                                   'size': array([25]),
                                   'sizemode': 'area',
                                   'sizeref': 1.316,
                                   'symbol': 'circle'},
                        'mode': 'markers',
                        'name': 'Classical 101 - Next Steps',
                        'orientation': 'v',
                        'showlegend': True,
                        'type': 'scatter',
                        'x': array([25]),
                        'xaxis': 'x',
                        'y': array([126]),
                        'yaxis': 'y'},
                       {'customdata': array([[13]]),
                        'hovertemplate': ('PlaylistName=Classical 101 - D' ... '{customdata[0]}<extra></extra>'),
                        'legendgroup': 'Classical 101 - Deep Cuts',
                        'marker': {'color': '#FF97FF',
                                   'size': array([25]),
                                   'sizemode': 'area',
                                   'sizeref': 1.316,
                                   'symbol': 'circle'},
                        'mode': 'markers',
                        'name': 'Classical 101 - Deep Cuts',
                        'orientation': 'v',
                        'showlegend': True,
                        'type': 'scatter',
                        'x': array([25]),
                        'xaxis': 'x',
                        'y': array([112]),
                        'yaxis': 'y'},
                       {'customdata': array([[16]]),
                        'hovertemplate': ('PlaylistName=Grunge<br>Number ' ... '{customdata[0]}<extra></extra>'),
                        'legendgroup': 'Grunge',
                        'marker': {'color': '#FECB52',
                                   'size': array([15]),
                                   'sizemode': 'area',
                                   'sizeref': 1.316,
                                   'symbol': 'circle'},
                        'mode': 'markers',
                        'name': 'Grunge',
                        'orientation': 'v',
                        'showlegend': True,
                        'type': 'scatter',
                        'x': array([15]),
                        'xaxis': 'x',
                        'y': array([68]),
                        'yaxis': 'y'}],
              'layout': {'legend': {'itemsizing': 'constant', 'title': {'text': 'Playlist Name'}, 'tracegroupgap':
         0},
                         'template': '...',
                         'title': {'text': 'Playlist Duration vs Number of Tracks'},
                         'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'Number of Tracks'}},
                         'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'Total Duration (Minute
         s)'}}}
          }))
```

In [41]:
```
question = """
     Identify artists who have albums with tracks appearing in multiple genres:
"""

vn.ask(question=question)
```

```
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
[{'role': 'system', 'content': "You are a SQLite expert. Please help to generate a SQL query to answer the
question. Your response should ONLY be based on the given context and follow the response guidelines and fo
rmat instructions. \n===Tables \nCREATE INDEX IFK_AlbumArtistId ON Album (ArtistId)\n\nCREATE TABLE Track\n
(\n    TrackId INTEGER  NOT NULL,\n    Name NVARCHAR(200)  NOT NULL,\n    AlbumId INTEGER,\n    MediaTypeId
INTEGER  NOT NULL,\n    GenreId INTEGER,\n    Composer NVARCHAR(220),\n    Milliseconds INTEGER  NOT NUL
L,\n    Bytes INTEGER,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (Track
Id),\n    FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n
FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIG
N KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n
CREATE INDEX IFK_TrackGenreId ON Track (GenreId)\n\nCREATE INDEX IFK_TrackAlbumId ON Track (AlbumId)\n\nCRE
ATE TABLE Album\n(\n    AlbumId INTEGER  NOT NULL,\n    Title NVARCHAR(160)  NOT NULL,\n    ArtistId INTEGE
R  NOT NULL,\n    CONSTRAINT PK_Album PRIMARY KEY  (AlbumId),\n    FOREIGN KEY (ArtistId) REFERENCES Artist
(ArtistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_TrackMediaTypeId ON Track
(MediaTypeId)\n\nCREATE INDEX IFK_PlaylistTrackTrackId ON PlaylistTrack (TrackId)\n\nCREATE TABLE Artist\n
(\n    ArtistId INTEGER  NOT NULL,\n    Name NVARCHAR(120),\n    CONSTRAINT PK_Artist PRIMARY KEY  (ArtistI
d)\n)\n\nCREATE TABLE Genre\n(\n    GenreId INTEGER  NOT NULL,\n    Name NVARCHAR(120),\n    CONSTRAINT PK_
Genre PRIMARY KEY  (GenreId)\n)\n\nCREATE TABLE PlaylistTrack\n(\n    PlaylistId INTEGER  NOT NULL,\n    Tr
ackId INTEGER  NOT NULL,\n    CONSTRAINT PK_PlaylistTrack PRIMARY KEY  (PlaylistId, TrackId),\n    FOREIGN
KEY (PlaylistId) REFERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREI
GN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n===Additi
onal Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided
context is sufficient, please generate a valid SQL query without any explanations for the question. \n2. If
the provided context is almost sufficient but requires knowledge of a specific string in a particular colum
n, please generate an intermediate SQL query to find the distinct strings in that column. Prepend the query
with a comment saying intermediate_sql \n3. If the provided context is insufficient, please explain why it
can't be generated. \n4. Please use the most relevant table(s). \n5. If the question has been asked and ans
wered before, please repeat the answer exactly as it was given before. \n"}, {'role': 'user', 'content': '
\n    Get the top 10 most popular artists (based on the number of tracks):\n'}, {'role': 'assistant', 'cont
ent': 'SELECT \n    Artist.ArtistId,\n    Artist.Name,\n    COUNT(Track.TrackId) AS TrackCount\nFROM \n
Artist\nJOIN \n    Album ON Artist.ArtistId = Album.ArtistId\nJOIN \n    Track ON Album.AlbumId = Track.Alb
umId\nGROUP BY \n    Artist.ArtistId\nORDER BY \n    TrackCount DESC\nLIMIT 10;'}, {'role': 'user', 'conten
t': '  \n    List all albums and their corresponding artist names  \n'}, {'role': 'assistant', 'content':
'SELECT \n    Album.Title AS AlbumTitle, \n    Artist.Name AS ArtistName\nFROM \n    Album\nJOIN \n    Arti
st ON Album.ArtistId = Artist.ArtistId\nORDER BY \n    Artist.Name, Album.Title;'}, {'role': 'user', 'conte
nt': '  \n    List all genres and the number of tracks in each genre:\n'}, {'role': 'assistant', 'content':
'SELECT \n    Genre.Name AS GenreName,\n    COUNT(Track.TrackId) AS TrackCount\nFROM \n    Genre\nLEFT JOIN
\n    Track ON Genre.GenreId = Track.GenreId\nGROUP BY \n    Genre.GenreId\nORDER BY \n    TrackCount DES
C;'}, {'role': 'user', 'content': '  \n     Find the top 5 customer who bought the most albums in total qua
ntity (across all invoices):\n'}, {'role': 'assistant', 'content': 'SELECT \n    c.CustomerId,\n    c.First
Name,\n    c.LastName,\n    SUM(il.Quantity) AS TotalAlbumsBought\nFROM \n    Customer c\nJOIN Invoice i ON
c.CustomerId = i.CustomerId\nJOIN InvoiceLine il ON i.InvoiceId = il.InvoiceId\nJOIN Track t ON il.TrackId
= t.TrackId\nWHERE \n    t.AlbumId IS NOT NULL\nGROUP BY \n    c.CustomerId\nORDER BY \n    TotalAlbumsBoug
ht DESC\nLIMIT 5;'}, {'role': 'user', 'content': '  \n     Find the customer who bought the most albums in
total quantity (across all invoices): \n'}, {'role': 'assistant', 'content': 'SELECT \n    c.CustomerId,\n
c.FirstName,\n    c.LastName,\n    SUM(il.Quantity) AS TotalAlbumsBought\nFROM \n    Customer c\nJOIN Invoi
ce i ON c.CustomerId = i.CustomerId\nJOIN InvoiceLine il ON i.InvoiceId = il.InvoiceId\nJOIN Track t ON il.
TrackId = t.TrackId\nWHERE \n    t.AlbumId IS NOT NULL\nGROUP BY \n    c.CustomerId\nORDER BY \n    TotalAl
bumsBought DESC\nLIMIT 1;'}, {'role': 'user', 'content': '  \n     Get all playlists containing at least 10
tracks and the total duration of those tracks:\n'}, {'role': 'assistant', 'content': 'SELECT \n    p.Playli
stId,\n    p.Name AS PlaylistName,\n    COUNT(pt.TrackId) AS TrackCount,\n    SUM(t.Milliseconds) / 60000 A
S TotalDurationMinutes\nFROM \n    Playlist p\nJOIN \n    PlaylistTrack pt ON p.PlaylistId = pt.PlaylistId
\nJOIN \n    Track t ON pt.TrackId = t.TrackId\nGROUP BY \n    p.PlaylistId\nHAVING \n    COUNT(pt.TrackId)
>= 10\nORDER BY \n    TrackCount DESC;'}, {'role': 'user', 'content': '  \n    Find the top 5 most expensiv
e tracks (based on unit price):\n'}, {'role': 'assistant', 'content': 'SELECT \n    TrackId,\n    Name,\n
UnitPrice\nFROM \n    Track\nORDER BY \n    UnitPrice DESC\nLIMIT 5;'}, {'role': 'user', 'content': '  \n
Find all tracks with a name containing "What" (case-insensitive)\n'}, {'role': 'assistant', 'content': "SEL
ECT TrackId, Name\nFROM Track\nWHERE Name LIKE '%What%' COLLATE NOCASE;"}, {'role': 'user', 'content': 'Sho
w me a list of tables in the SQLite database'}, {'role': 'assistant', 'content': "SELECT name FROM sqlite_m
aster WHERE type='table';"}, {'role': 'user', 'content': 'what are the top 5 countries that customers come
from?'}, {'role': 'assistant', 'content': 'SELECT Country, COUNT(*) as CustomerCount\nFROM Customer\nGROUP
BY Country\nORDER BY CustomerCount DESC\nLIMIT 5;'}, {'role': 'user', 'content': '  \n     Identify artists
who have albums with tracks appearing in multiple genres:\n'}]
Using model claude-3-5-sonnet-20240620 for 1405.75 tokens (approx)
SELECT DISTINCT
    ar.ArtistId,
    ar.Name AS ArtistName
FROM
    Artist ar
JOIN Album al ON ar.ArtistId = al.ArtistId
JOIN Track t ON al.AlbumId = t.AlbumId
GROUP BY
    ar.ArtistId, al.AlbumId
HAVING
    COUNT(DISTINCT t.GenreId) > 1
ORDER BY
    ar.Name;
SELECT DISTINCT
    ar.ArtistId,
    ar.Name AS ArtistName
FROM
    Artist ar
JOIN Album al ON ar.ArtistId = al.ArtistId
JOIN Track t ON al.AlbumId = t.AlbumId
GROUP BY
```

```
    ar.ArtistId, al.AlbumId
HAVING
    COUNT(DISTINCT t.GenreId) > 1
ORDER BY
    ar.Name;
SELECT DISTINCT
    ar.ArtistId,
    ar.Name AS ArtistName
FROM
    Artist ar
JOIN Album al ON ar.ArtistId = al.ArtistId
JOIN Track t ON al.AlbumId = t.AlbumId
GROUP BY
    ar.ArtistId, al.AlbumId
HAVING
    COUNT(DISTINCT t.GenreId) > 1
ORDER BY
    ar.Name;
   ArtistId            ArtistName
0       147  Battlestar Galactica
1        81          Eric Clapton
2       148                Heroes
3        90           Iron Maiden
4       100         Lenny Kravitz
5       149                  Lost
6       156            The Office
Using model claude-3-5-sonnet-20240620 for 237.25 tokens (approx)
```


Artists with Albums Containing Tracks in Multiple Genres
Artist ID
160
140
120
100
80
60
40
20
0
Battlestar Galactica
Eric Clapton
Heroes
Iron Maiden
Lenny Kravitz
Lost
The Office
Artist Name

Out[41]:
```
('SELECT DISTINCT \n    ar.ArtistId,\n    ar.Name AS ArtistName\nFROM \n    Artist ar\nJOIN Album al ON a
r.ArtistId = al.ArtistId\nJOIN Track t ON al.AlbumId = t.AlbumId\nGROUP BY \n    ar.ArtistId, al.AlbumId\n
HAVING \n    COUNT(DISTINCT t.GenreId) > 1\nORDER BY \n    ar.Name;',
    ArtistId           ArtistName
 0       147  Battlestar Galactica
 1        81          Eric Clapton
 2       148                Heroes
 3        90           Iron Maiden
 4       100         Lenny Kravitz
 5       149                  Lost
 6       156            The Office,
 Figure({
     'data': [{'alignmentgroup': 'True',
               'hovertemplate': 'ArtistName=%{x}<br>ArtistId=%{y}<extra></extra>',
               'legendgroup': '',
               'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
               'name': '',
               'offsetgroup': '',
               'orientation': 'v',
               'showlegend': False,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Battlestar Galactica', 'Eric Clapton', 'Heroes', 'Iron Maiden',
                           'Lenny Kravitz', 'Lost', 'The Office'], dtype=object),
               'xaxis': 'x',
               'y': array([147,  81, 148,  90, 100, 149, 156]),
               'yaxis': 'y'}],
     'layout': {'barmode': 'relative',
                'legend': {'tracegroupgap': 0},
                'template': '...',
                'title': {'text': 'Artists with Albums Containing Tracks in Multiple Genres'},
                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'Artist Name'}},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'Artist ID'}}}
 }))
```

Check completion time

In [42]:
```python
ts_stop = time()
```

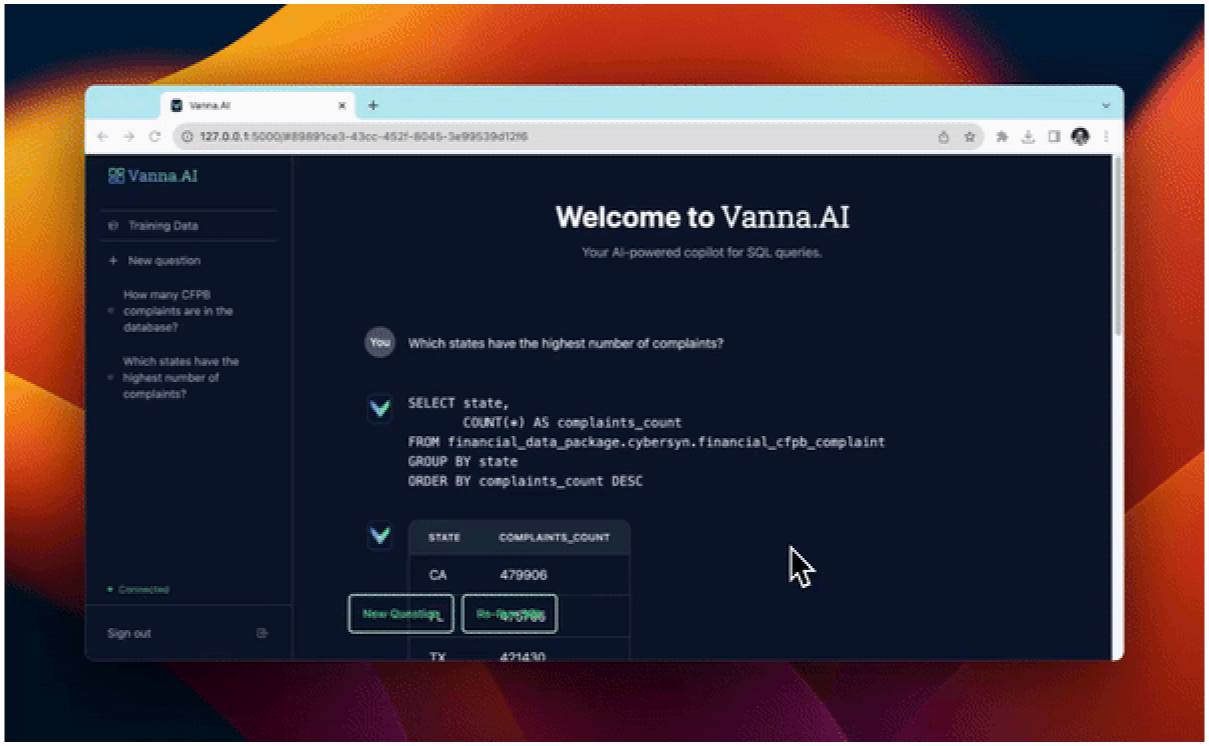
```
elapsed_time = ts_stop - ts_start
print(f"test running on '{hostname}' with '{model_name}' LLM took : {elapsed_time:.2f} sec")
```

```
test running on 'papa-game' with 'claude-3-5-sonnet-20240620' LLM took : 88.94 sec
```

In [43]:
```
from datetime import datetime
print(datetime.now())
```

```
2024-06-21 21:00:21.041741
```

Launch the User Interface


Vanna.AI
127.0.0.1:5000/#89891ce3-43cc-452f-8045-3e99539d12f6
Vanna.AI
Training Data
New question
How many CFPB complaints are in the database?
Which states have the highest number of complaints?
Connected
Sign out
Welcome to Vanna.AI
Your AI-powered copilot for SQL queries.
You
New Question
Re-Run SQL

Which states have the highest number of complaints?

```
SELECT state,
       COUNT(*) AS complaints_count
FROM financial_data_package.cybersyn.financial_cfpb_complaint
GROUP BY state
ORDER BY complaints_count DESC
```

STATE	COMPLAINTS_COUNT
CA	479906
FL	[illegible]
TX	421430

from vanna.flask import VannaFlaskApp app = VannaFlaskApp(vn) app.run()

Next Steps

Using Vanna via Jupyter notebooks is great for getting started but check out additional customizable interfaces like the

- Streamlit app
- Flask app
- Slackbot