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]'

In [1]:
```
model_name = 'gpt-3.5-turbo'
file_db = "~/Downloads/chinook.sqlite"
```

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

openai_api_key = s.get_api_key(provider="OPENAI")
```

openai_api_key

In [3]:
```
from vanna.openai import OpenAI_Chat
from vanna.chromadb.chromadb_vector import ChromaDB_VectorStore
```

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

config = {
    'api_key': openai_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]:
```
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 [ ]:

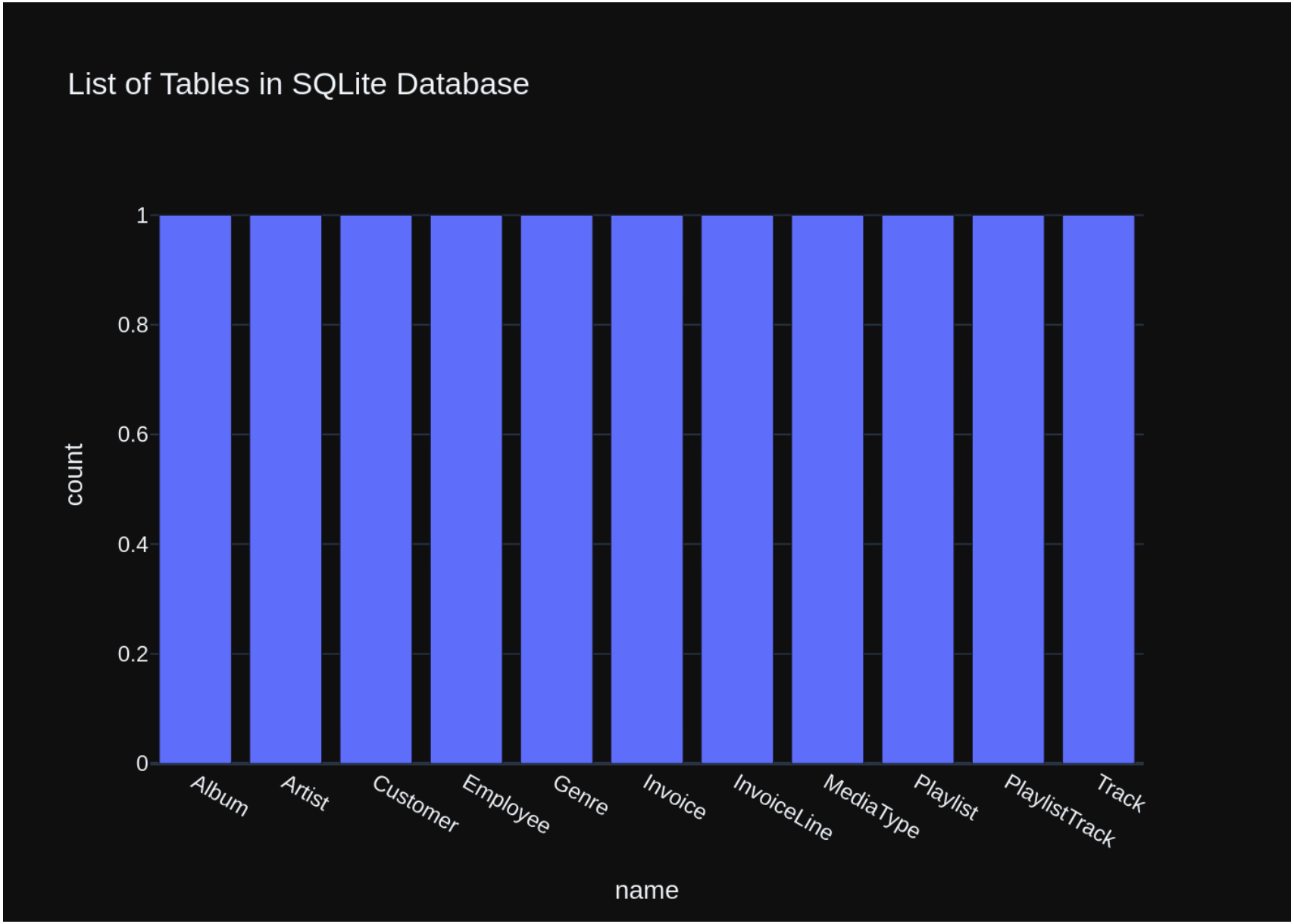
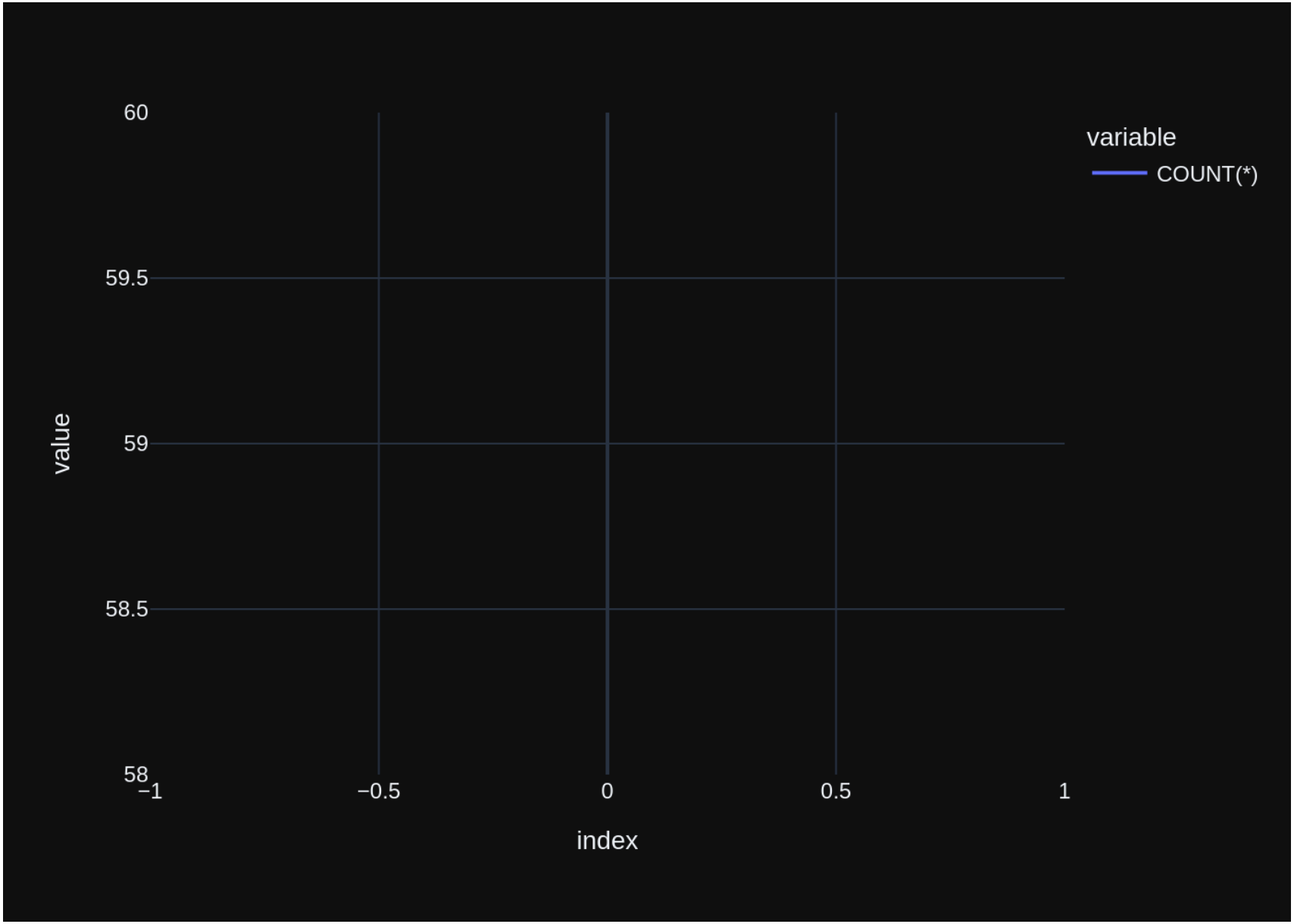
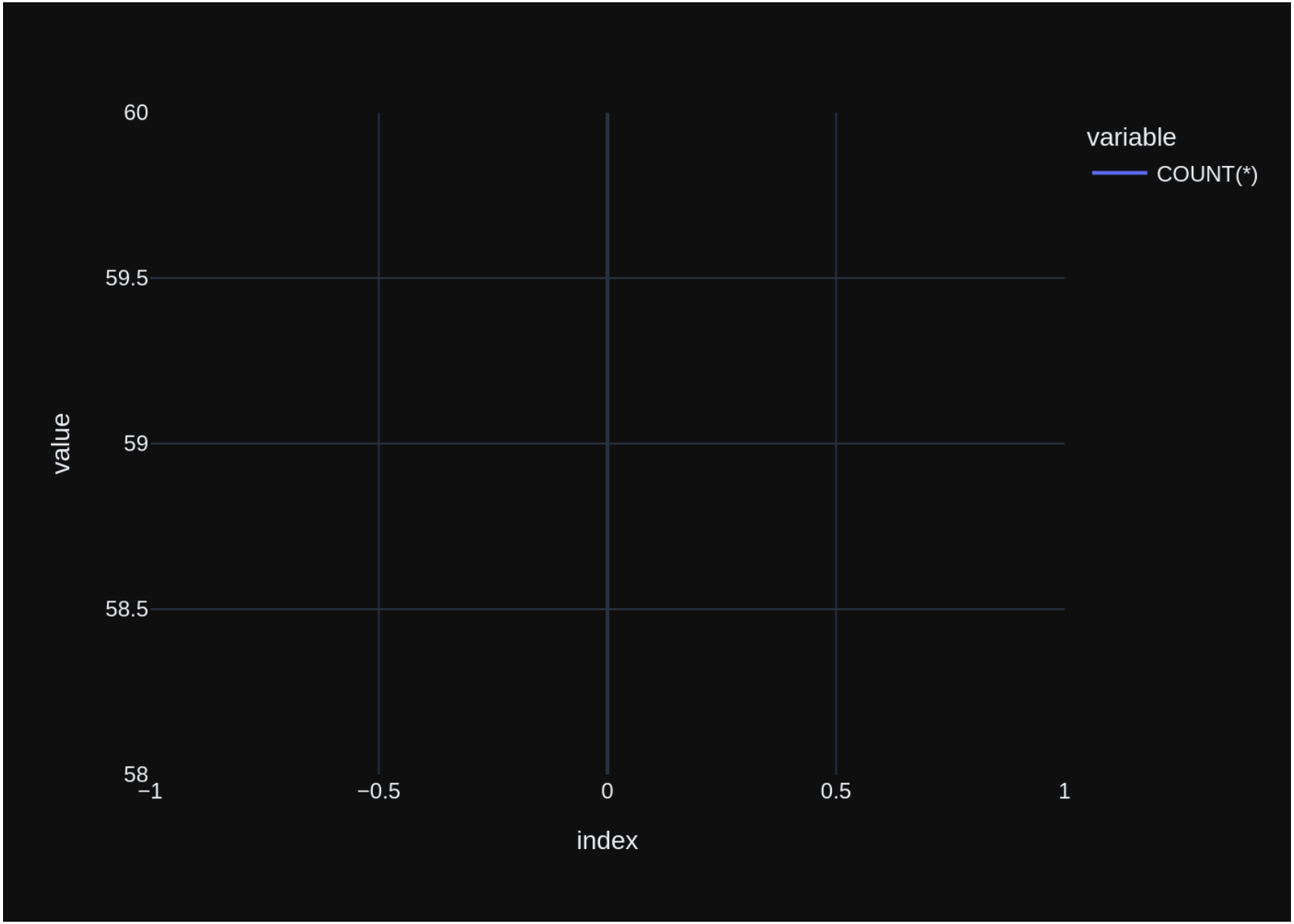
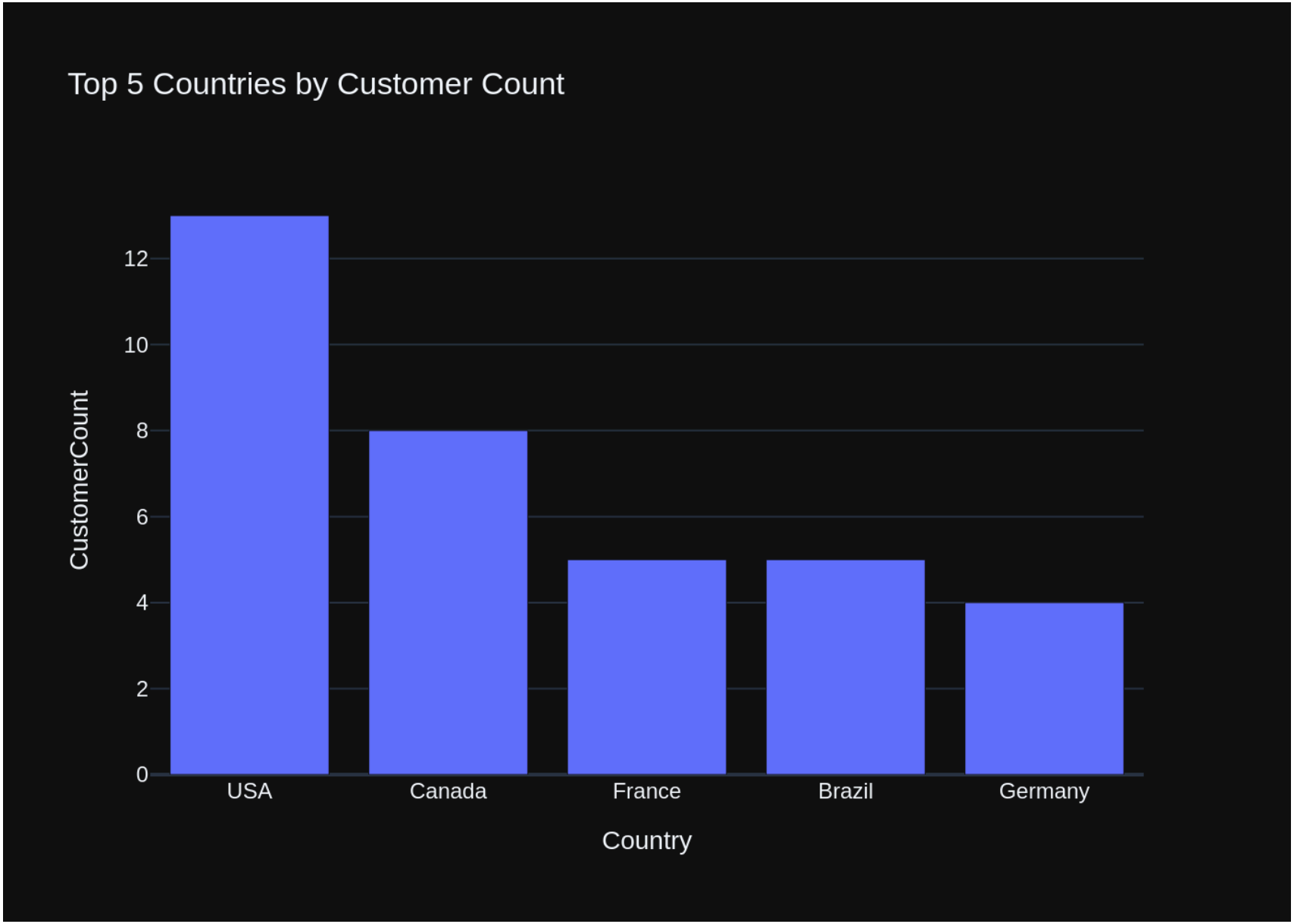
Asking the AI

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()
```

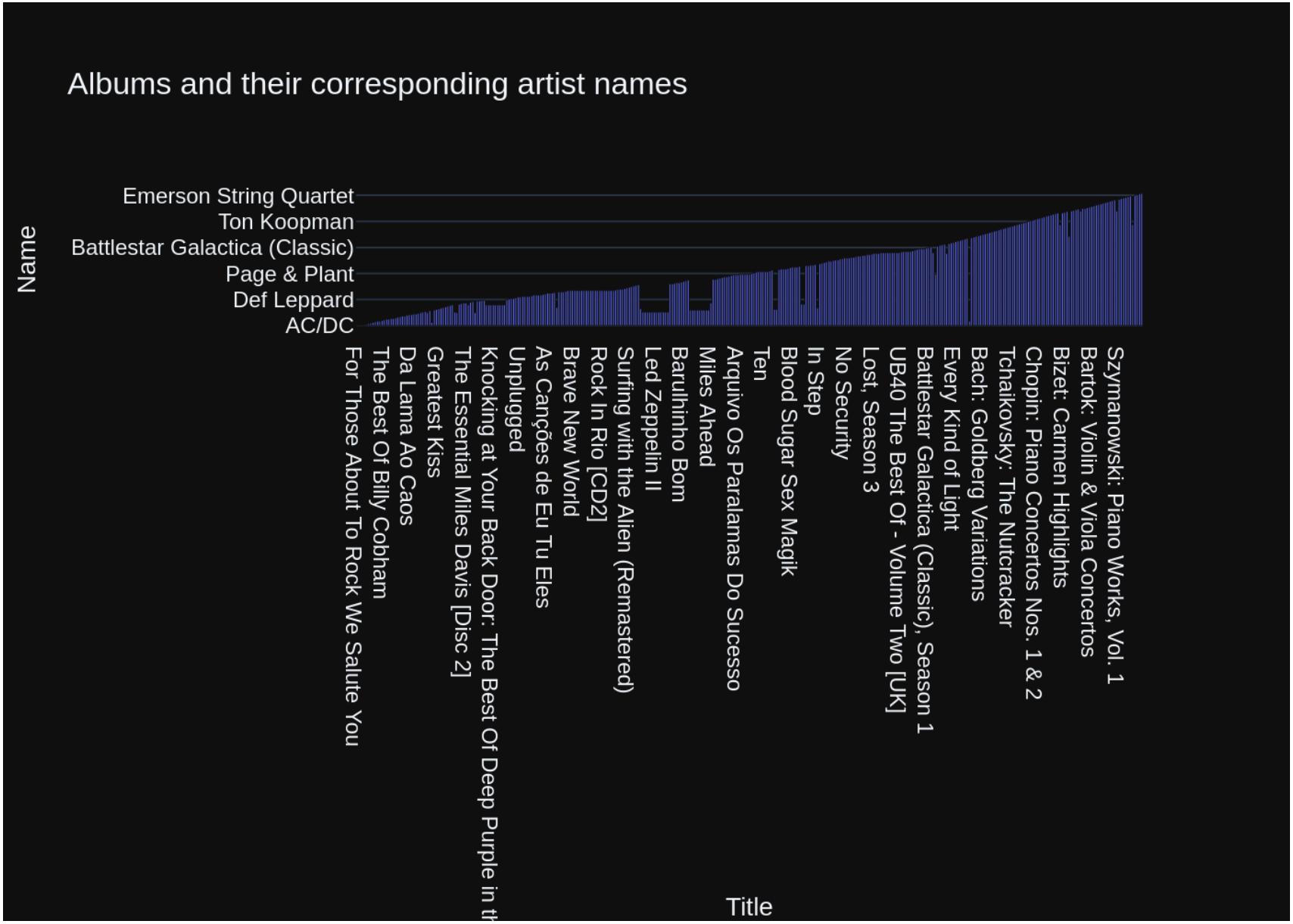
In [18]:
```
vn.ask(question="Show me a list of tables in the SQLite database")
```

```
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 gpt-3.5-turbo for 948.25 tokens (approx)
```sql
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 gpt-3.5-turbo for 168.5 tokens (approx)
```


List of Tables in SQLite Database
count
1
0.8
0.6
0.4
0.2
0
Album
Artist
Customer
Employee
Genre
Invoice
InvoiceLine
MediaType
Playlist
PlaylistTrack
Track
name

Out[18]:
```
("SELECT name\nFROM sqlite_master\nWHERE 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': [{'alignmentgroup': 'True',
              'hovertemplate': 'name=%{x}<br>count=%{y}<extra></extra>',
              'legendgroup': '',
              'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
              'name': '',
              'offsetgroup': '',
              'orientation': 'v',
              'showlegend': False,
              'textposition': 'auto',
              'type': 'bar',
              'x': array(['Album', 'Artist', 'Customer', 'Employee', 'Genre', 'Invoice',
                          'InvoiceLine', 'MediaType', 'Playlist', 'PlaylistTrack', 'Track'],
                         dtype=object),
              'xaxis': 'x',
              'y': array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]),
              'yaxis': 'y'}],
    'layout': {'barmode': 'relative',
               'legend': {'tracegroupgap': 0},
               'template': '...',
               'title': {'text': 'List of Tables in SQLite Database'},
               'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'name'}},
               'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'count'}}}
})
```

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 format 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 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 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 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 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\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 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_CustomerSupportRepId ON Customer (SupportRepId)\n\nCREATE TABLE Playlist\n(\n   PlaylistId 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 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 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 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\nFROM sqlite_master\nWHERE type = 'tabl
e';"}, {'role': 'user', 'content': 'How many records are in table called customer'}]
Using model gpt-3.5-turbo for 1175.25 tokens (approx)
SELECT COUNT(*)
FROM Customer;
SELECT COUNT(*)
FROM Customer;
SELECT COUNT(*)
FROM Customer;
   COUNT(*)
0        59
Using model gpt-3.5-turbo for 163.25 tokens (approx)
```


60
59.5
59
58.5
58
value
−1
−0.5
0
0.5
1
index
variable
COUNT(*)

Out[19]:
```
('SELECT COUNT(*)\nFROM Customer;',
   COUNT(*)
0        59,
Figure({
    'data': [{'hovertemplate': 'variable=COUNT(*)<br>index=%{x}<br>value=%{y}<extra></extra>',
              'legendgroup': 'COUNT(*)',
              'line': {'color': '#636efa', 'dash': 'solid'},
              'marker': {'symbol': 'circle'},
              'mode': 'lines',
              'name': 'COUNT(*)',
              'orientation': 'v',
              'showlegend': True,
              'type': 'scatter',
              'x': array([0]),
              'xaxis': 'x',
              'y': array([59]),
              'yaxis': 'y'}],
    'layout': {'legend': {'title': {'text': 'variable'}, 'tracegroupgap': 0},
               'margin': {'t': 60},
               'template': '...',
               'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'index'}},
               'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'value'}}}
})
```

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 format 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 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 INDEX IFK_CustomerSupportRepId ON Customer (SupportRepId)\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 UPDATE NO ACTION\n)\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_InvoiceLineInvoiceId ON InvoiceLine (InvoiceId)\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_InvoiceLineTrackId ON InvoiceLine (TrackId)\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\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\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': 'How many records are in table called customer'}, {'role': 'assistant', 'content': 'SELECT COUNT(*)\nFROM Custom

```
er;'}, {'role': 'user', 'content': 'Show me a list of tables in the SQLite database'}, {'role': 'assistan
t', 'content': "SELECT name\nFROM sqlite_master\nWHERE type = 'table';"}, {'role': 'user', 'content': 'How
many customers are there'}]
Using model gpt-3.5-turbo for 1095.5 tokens (approx)
SELECT COUNT(*)
FROM Customer;
SELECT COUNT(*)
FROM Customer;
SELECT COUNT(*)
FROM Customer;
   COUNT(*)
0        59
Using model gpt-3.5-turbo for 159.0 tokens (approx)
```


60
59.5
59
58.5
58
value
−1
−0.5
0
0.5
1
index
variable
COUNT(*)

Out[20]:
```
('SELECT COUNT(*)\nFROM Customer;',
   COUNT(*)
0        59,
Figure({
    'data': [{'hovertemplate': 'variable=COUNT(*)<br>index=%{x}<br>value=%{y}<extra></extra>',
              'legendgroup': 'COUNT(*)',
              'line': {'color': '#636efa', 'dash': 'solid'},
              'marker': {'symbol': 'circle'},
              'mode': 'lines',
              'name': 'COUNT(*)',
              'orientation': 'v',
              'showlegend': True,
              'type': 'scatter',
              'x': array([0]),
              'xaxis': 'x',
              'y': array([59]),
              'yaxis': 'y'}],
    'layout': {'legend': {'title': {'text': 'variable'}, 'tracegroupgap': 0},
               'margin': {'t': 60},
               'template': '...',
               'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'index'}},
               'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'value'}}}
})
```

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 format 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 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 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 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 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 MediaType\n(\n   MediaTypeId INTEGER NOT NULL,\n   Name NVARCHAR(120),\n   CONSTRAINT PK_MediaType PRIMARY KEY (MediaTypeId)\n)\n\nCREATE TABLE Playlist\n(\n   PlaylistId INTEGER NOT NULL,\n   Name NVARCHAR(120),\n   CONSTRAINT PK_Playlist PRIMARY KEY (PlaylistId)\n)\n\nCREATE INDEX IFK_CustomerSupportRepId ON Customer (SupportRepId)\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 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 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 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 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(*)\nFROM Customer;'}, {'role': 'user', 'content': 'How many records are in table c
alled customer'}, {'role': 'assistant', 'content': 'SELECT COUNT(*)\nFROM Customer;'}, {'role': 'user', 'co
ntent': 'Show me a list of tables in the SQLite database'}, {'role': 'assistant', 'content': "SELECT name\n
FROM sqlite_master\nWHERE type = 'table';"}, {'role': 'user', 'content': 'what are the top 5 countries that
customers come from?'}]
Using model gpt-3.5-turbo for 1231.25 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 gpt-3.5-turbo for 192.75 tokens (approx)
```


Top 5 Countries by Customer Count
CustomerCount
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 by Customer Count'},
                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'Country'}},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'CustomerCount'}}}
 }))
```

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\nFROM sqlite_master\nWHERE type = 'table';"}, {'role': 'user', 'content': 'what are the top 5 cou
ntries that customers come from?'}, {'role': 'assistant', 'content': 'SELECT Country, COUNT(*) AS CustomerC
ount\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(*)\nFROM C
ustomer;'}, {'role': 'user', 'content': 'How many customers are there'}, {'role': 'assistant', 'content':
'SELECT COUNT(*)\nFROM Customer;'}, {'role': 'user', 'content': '  \n    List all albums and their correspo
nding artist names  \n'}]
Using model gpt-3.5-turbo for 801.5 tokens (approx)
SELECT Album.Title, Artist.Name
FROM Album
JOIN Artist ON Album.ArtistId = Artist.ArtistId;
SELECT Album.Title, Artist.Name
FROM Album
JOIN Artist ON Album.ArtistId = Artist.ArtistId;
```

```
SELECT Album.Title, Artist.Name
FROM Album
JOIN Artist ON Album.ArtistId = Artist.ArtistId;
                                                 Title  \
0                For For Those About To Rock We Salute You
1                                     Balls to the Wall
2                                     Restless and Wild
3                                     Let There Be Rock
4                                              Big Ones
..                                                  ...
342                             Respighi:Pines of Rome
343  Schubert: The Late String Quartets & String Qu...
344                                Monteverdi: L'Orfeo
345                              Mozart: Chamber Music
346  Koyaanisqatsi (Soundtrack from the Motion Pict...

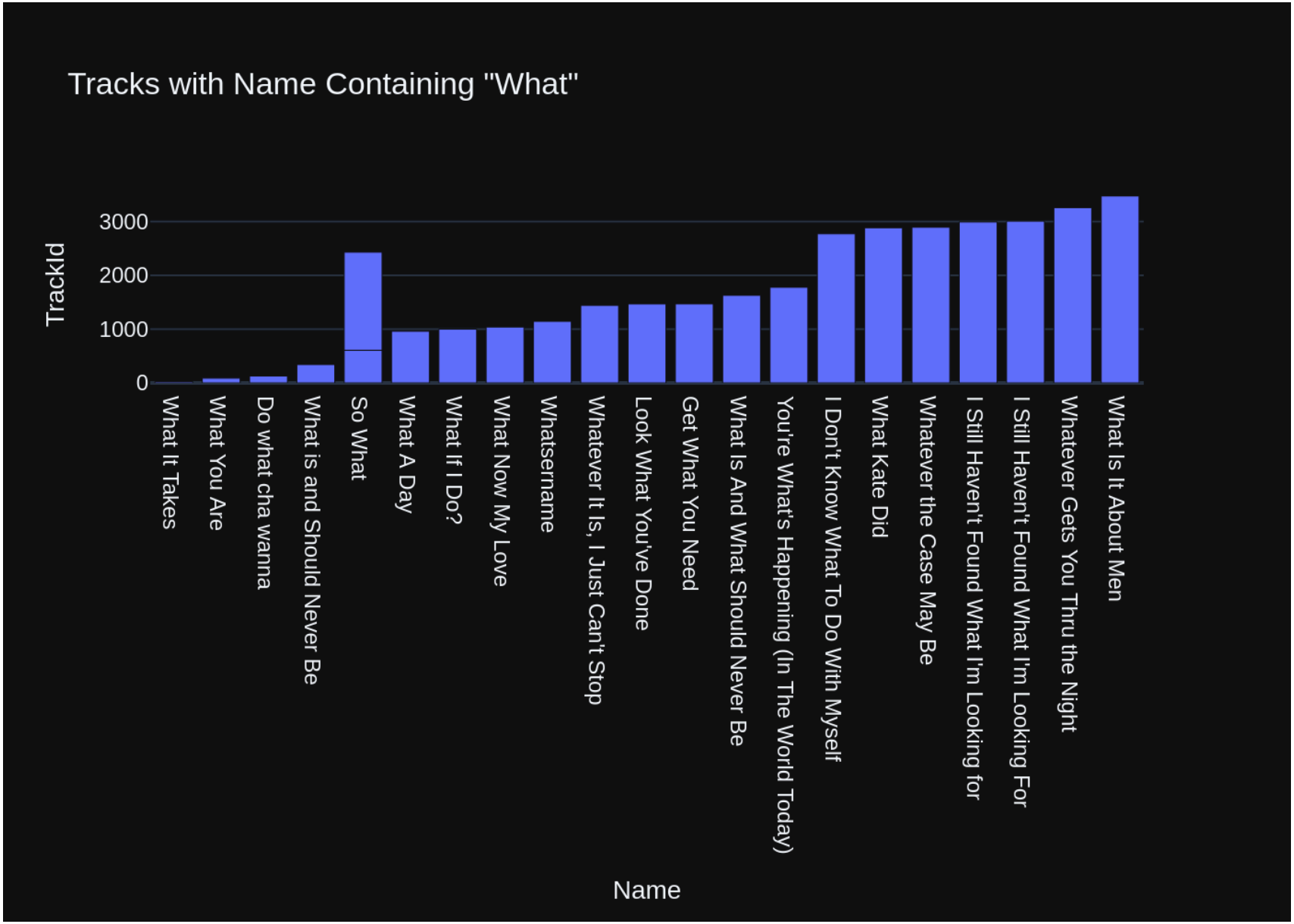
                                                  Name
0                                                AC/DC
1                                              Accept
2                                              Accept
3                                                AC/DC
4                                            Aerosmith
..                                                 ...
342                                    Eugene Ormandy
343                             Emerson String Quartet
344  C. Monteverdi, Nigel Rogers - Chiaroscuro; Lon...
345                                     Nash Ensemble
346                              Philip Glass Ensemble

[347 rows x 2 columns]
Using model gpt-3.5-turbo for 186.25 tokens (approx)
```


Albums and their corresponding artist names
Name
Emerson String Quartet
Ton Koopman
Battlestar Galactica (Classic)
Page & Plant
Def Leppard
AC/DC
Szymanowski: Piano Works, Vol. 1
Bartok: Violin & Viola Concertos
Bizet: Carmen Highlights
Chopin: Piano Concertos Nos. 1 & 2
Tchaikovsky: The Nutcracker
Bach: Goldberg Variations
Every Kind of Light
Battlestar Galactica (Classic), Season 1
UB40 The Best Of - Volume Two [UK]
Lost, Season 3
No Security
In Step
Blood Sugar Sex Magik
Ten
Arquivo Os Paralamas Do Sucesso
Miles Ahead
Barulhinho Bom
Led Zeppelin II
Surfing with the Alien (Remastered)
Rock In Rio [CD2]
Brave New World
As Canções de Eu Tu Eles
Unplugged
Knocking at Your Back Door: The Best Of Deep Purple in th
The Essential Miles Davis [Disc 2]
Greatest Kiss
Da Lama Ao Caos
The Best Of Billy Cobham
For Those About To Rock We Salute You
Title

Out[22]: ('SELECT Album.Title, Artist.Name\nFROM Album\nJOIN Artist ON Album.ArtistId = Artist.ArtistId;',

```
                                                 Title  \
0                For For Those About To Rock We Salute You
1                                     Balls to the Wall
2                                     Restless and Wild
3                                     Let There Be Rock
4                                              Big Ones
..                                                  ...
342                             Respighi:Pines of Rome
343  Schubert: The Late String Quartets & String Qu...
344                                Monteverdi: L'Orfeo
345                              Mozart: Chamber Music
346  Koyaanisqatsi (Soundtrack from the Motion Pict...

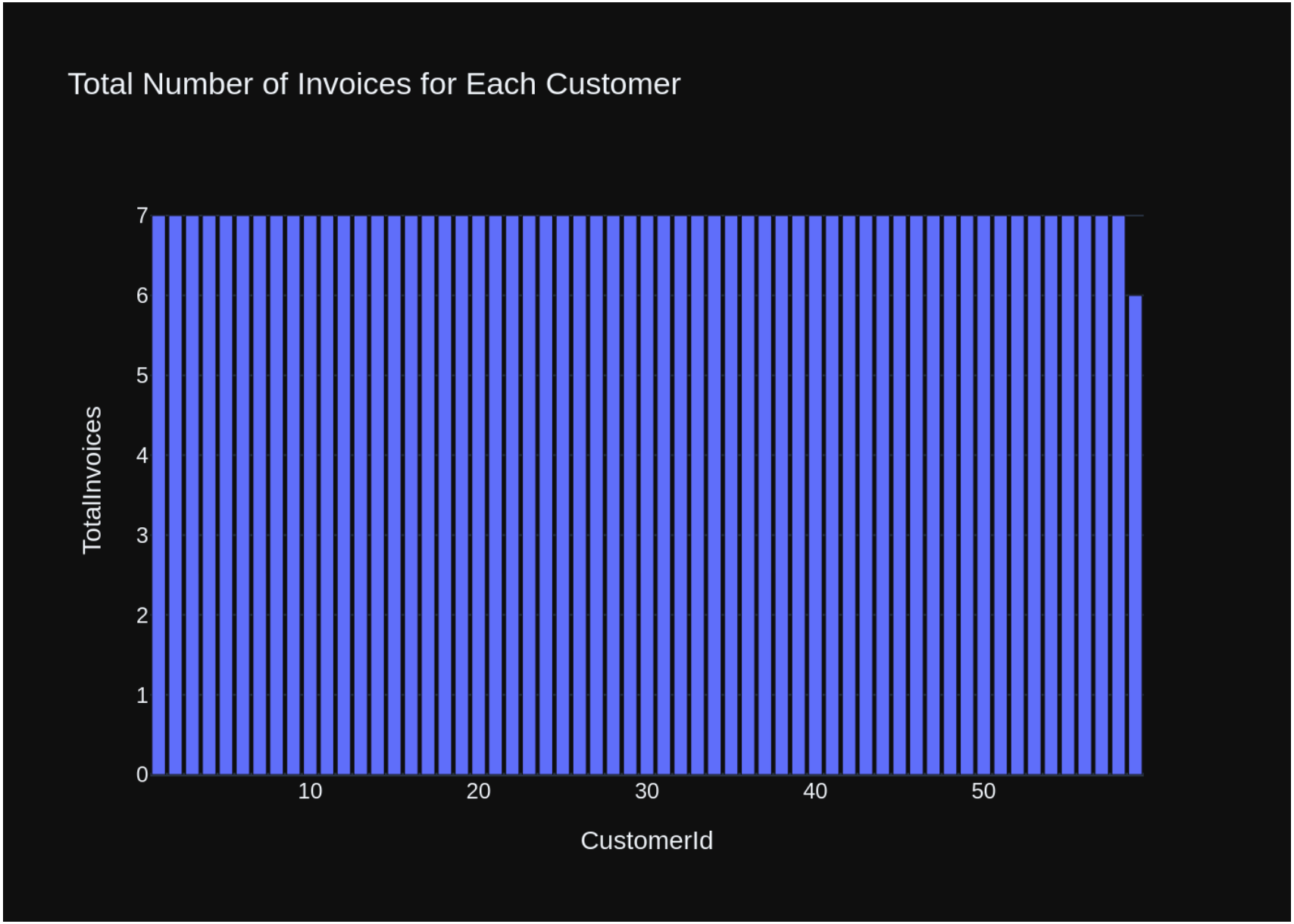
                                                  Name
0                                                AC/DC
1                                               Accept
2                                               Accept
3                                                AC/DC
4                                            Aerosmith
..                                                 ...
342                                      Eugene Ormandy
343                             Emerson String Quartet
344  C. Monteverdi, Nigel Rogers - Chiaroscuro; Lon...
345                                       Nash Ensemble
346                              Philip Glass Ensemble

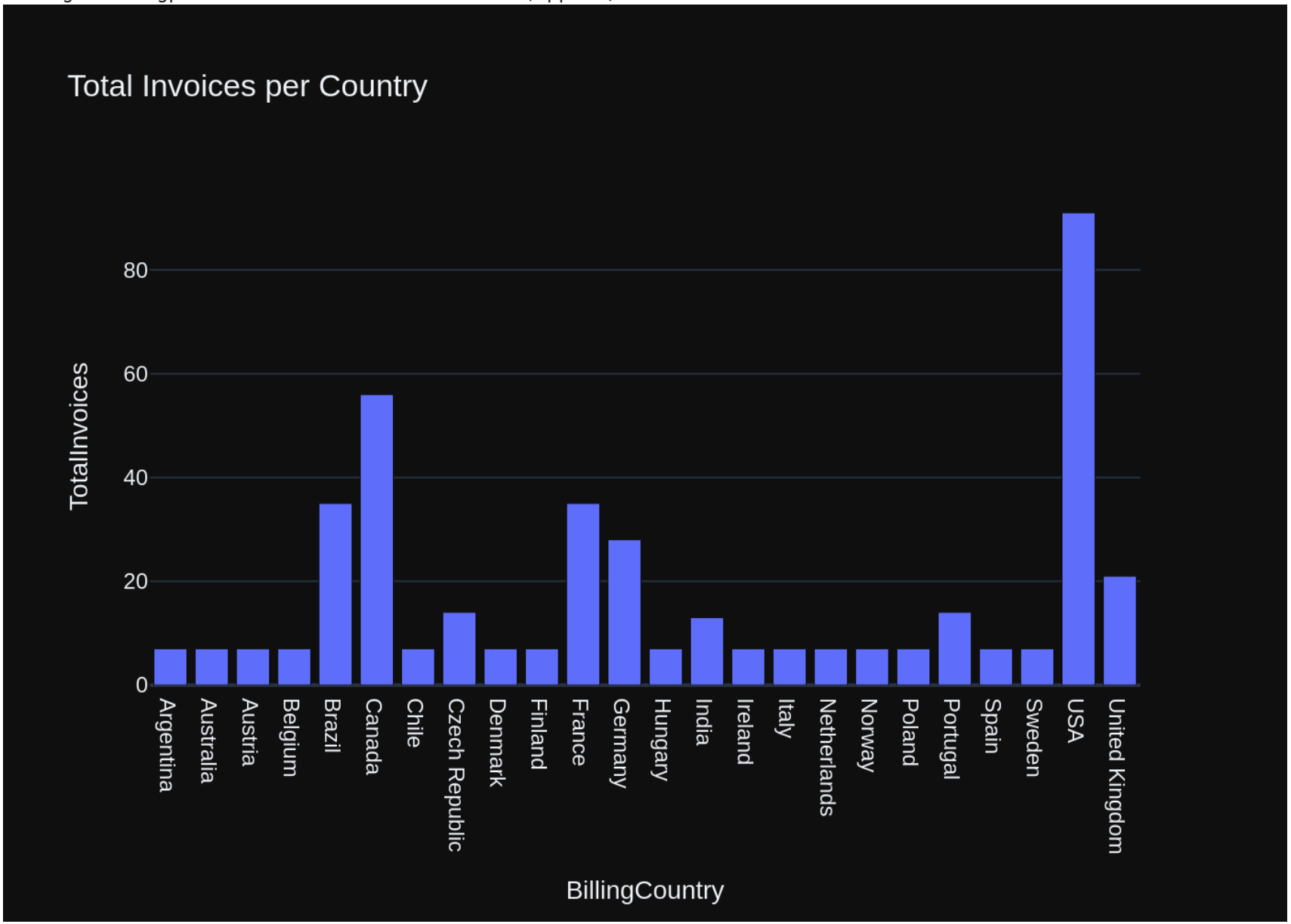
[347 rows x 2 columns],
Figure({
    'data': [{'alignmentgroup': 'True',
              'hovertemplate': 'Title=%{x}<br>Name=%{y}<extra></extra>',
              'legendgroup': '',
              'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
              'name': '',
              'offsetgroup': '',
              'orientation': 'v',
              'showlegend': False,
              'textposition': 'auto',
              'type': 'bar',
              'x': array(['For Those About To Rock We Salute You', 'Balls to the Wall',
                          'Restless and Wild', ..., "Monteverdi: L'Orfeo",
                          'Mozart: Chamber Music',
```

```
                            'Koyaanisqatsi (Soundtrack from the Motion Picture)'], dtype=object),
                'xaxis': 'x',
                'y': array(['AC/DC', 'Accept', 'Accept', ...,
                            'C. Monteverdi, Nigel Rogers - Chiaroscuro; London Baroque; London Cornett & Sa
ckbu',
                            'Nash Ensemble', 'Philip Glass Ensemble'], dtype=object),
                'yaxis': 'y'}],
     'layout': {'barmode': 'relative',
                'legend': {'tracegroupgap': 0},
                'template': '...',
                'title': {'text': 'Albums and their corresponding artist names'},
                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'Title'}},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'Name'}}}
 }))
```

In [23]:
```
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 format 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 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_TrackAlbumId ON Track (AlbumId)\n\nCREATE INDEX IFK_PlaylistTrackTrackId ON PlaylistTrack (TrackId)\n\nCREATE INDEX IFK_TrackMediaTypeId ON Track (MediaTypeId)\n\nCREATE INDEX IFK_InvoiceLineTrackId 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 (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 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\nCREATE TABLE Playlist\n(\n   PlaylistId 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 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   List all albums and their corresponding artist names\n'}, {'role': 'assistant', 'content': 'SELECT Album.Title, Artist.Name\nFROM Album\nJOIN Artist ON Album.ArtistId = Artist.ArtistId;'}, {'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 customers are there'}, {'role': 'assistant', 'content': 'SELECT COUNT(*)\nFROM Customer;'}, {'role': 'user', 'content': 'Show me a list of tables in the SQLite database'}, {'role': 'assistant', 'content': "SELECT name\nFROM sqlite_master\nWHERE type = 'table';"}, {'role': 'user', 'content': 'How many records are in table called customer'}, {'role': 'assistant', 'content': 'SELECT COUNT(*)\nFROM Customer;'}, {'role': 'user', 'content': ' \n   Find all tracks with a name containing "What" (case-insensitive)\n'}]
Using model gpt-3.5-turbo for 825.25 tokens (approx)
```
SELECT *
FROM Track
WHERE Name LIKE '%What%';
SELECT *
FROM Track
```

```
WHERE Name LIKE '%What%';
SELECT *
FROM Track
WHERE Name LIKE '%What%';
```

	TrackId	Name	AlbumId
0	26	What It Takes	5
1	88	What You Are	10
2	130	Do what cha wanna	13
3	342	What is and Should Never Be	30
4	607	So What	48
5	960	What A Day	76
6	1000	What If I Do?	80
7	1039	What Now My Love	83
8	1145	Whatsername	89
9	1440	Whatever It Is, I Just Can't Stop	116
10	1469	Look What You've Done	119
11	1470	Get What You Need	119
12	1628	What Is And What Should Never Be	133
13	1778	You're What's Happening (In The World Today)	146
14	1823	So What	149
15	2772	I Don't Know What To Do With Myself	223
16	2884	What Kate Did	231
17	2893	Whatever the Case May Be	230
18	2992	I Still Haven't Found What I'm Looking for	237
19	3007	I Still Haven't Found What I'm Looking For	238
20	3258	Whatever Gets You Thru the Night	255
21	3475	What Is It About Men	322

	MediaTypeId	GenreId	Composer
0	1	1	Steven Tyler, Joe Perry, Desmond Child
1	1	1	Audioslave/Chris Cornell
2	1	2	George Duke
3	1	1	Jimmy Page/Robert Plant
4	1	2	Miles Davis
5	1	1	Mike Bordin, Billy Gould, Mike Patton
6	1	1	Dave Grohl, Taylor Hawkins, Nate Mendel, Chris...
7	1	12	carl sigman/gilbert becaud/pierre leroyer
8	1	4	Green Day
9	1	1	Jay Kay/Kay, Jay
10	1	4	N. Cester
11	1	4	C. Cester/C. Muncey/N. Cester
12	1	1	Jimmy Page, Robert Plant

```
13            1       14              Allen Story/George Gordy/Robert Gordy
14            1        3                                   Culmer/Exalt
15            1        7                                           None
16            3       19                                           None
17            3       19                                           None
18            1        1     Bono/Clayton, Adam/Mullen Jr., Larry/The Edge
19            1        1                                             U2
20            2        9                                           None
21            2        9  Delroy "Chris" Cooper, Donovan Jackson, Earl C...

    Milliseconds      Bytes  UnitPrice
0         310622   10144730       0.99
1         249391    5988186       0.99
2         274155    9018565       0.99
3         260675    8497116       0.99
4         564009   18360449       0.99
5         158275    5203430       0.99
6         302994    9929799       0.99
7         149995    4913383       0.99
8         252316    8244843       0.99
9         247222    8249453       0.99
10        230974    7517083       0.99
11        247719    8043765       0.99
12        287973    9369385       0.99
13        142027    4631104       0.99
14        189152    6162894       0.99
15        221387    7251478       0.99
16       2610250  484583988       1.99
17       2616410  183867185       1.99
18        353567   11542247       0.99
19        280764    9306737       0.99
20        215084    3499018       0.99
21        209573    3426106       0.99
Using model gpt-3.5-turbo for 223.25 tokens (approx)
```


Tracks with Name 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 *\nFROM Track\nWHERE Name LIKE '%What%';",

	TrackId	Name	AlbumId	MediaTypeId	GenreId	Composer
0	26	What It Takes	5	1	1	Steven Tyler, Joe Perry, Desmond Child
1	88	What You Are	10	1	1	Audioslave/Chris Cornell
2	130	Do what cha wanna	13	1	2	George Duke
3	342	What is and Should Never Be	30	1	1	Jimmy Page/Robert Plant
4	607	So What	48	1	2	Miles Davis
5	960	What A Day	76	1	1	Mike Bordin, Billy Gould, Mike Patton
6	1000	What If I Do?	80	1	1	Dave Grohl, Taylor Hawkins, Nate Mendel, Chris...
7	1039	What Now My Love	83	1	12	carl sigman/gilbert becaud/pierre leroyer
8	1145	Whatsername	89	1	4	Green Day
9	1440	Whatever It Is, I Just Can't Stop	116	1	1	Jay Kay/Kay, Jay
10	1469	Look What You've Done	119	1	4	N. Cester
11	1470	Get What You Need	119	1	4	C. Cester/C. Muncey/N. Cester
12	1628	What Is And What Should Never Be	133	1	1	Jimmy Page, Robert Plant
13	1778	You're What's Happening (In The World Today)	146	1	14	Allen Story/George Gordy/Robert Gordy
14	1823	So What	149	1	3	Culmer/Exalt
15	2772	I Don't Know What To Do With Myself	223	1	7	None
16	2884	What Kate Did	231			
17	2893	Whatever the Case May Be	230			
18	2992	I Still Haven't Found What I'm Looking for	237			
19	3007	I Still Haven't Found What I'm Looking For	238			
20	3258	Whatever Gets You Thru the Night	255			
21	3475	What Is It About Men	322			

```
16            3       19                                                None
17            3       19                                                None
18            1        1      Bono/Clayton, Adam/Mullen Jr., Larry/The Edge
19            1        1                                                  U2
20            2        9                                                None
21            2        9  Delroy "Chris" Cooper, Donovan Jackson, Earl C...

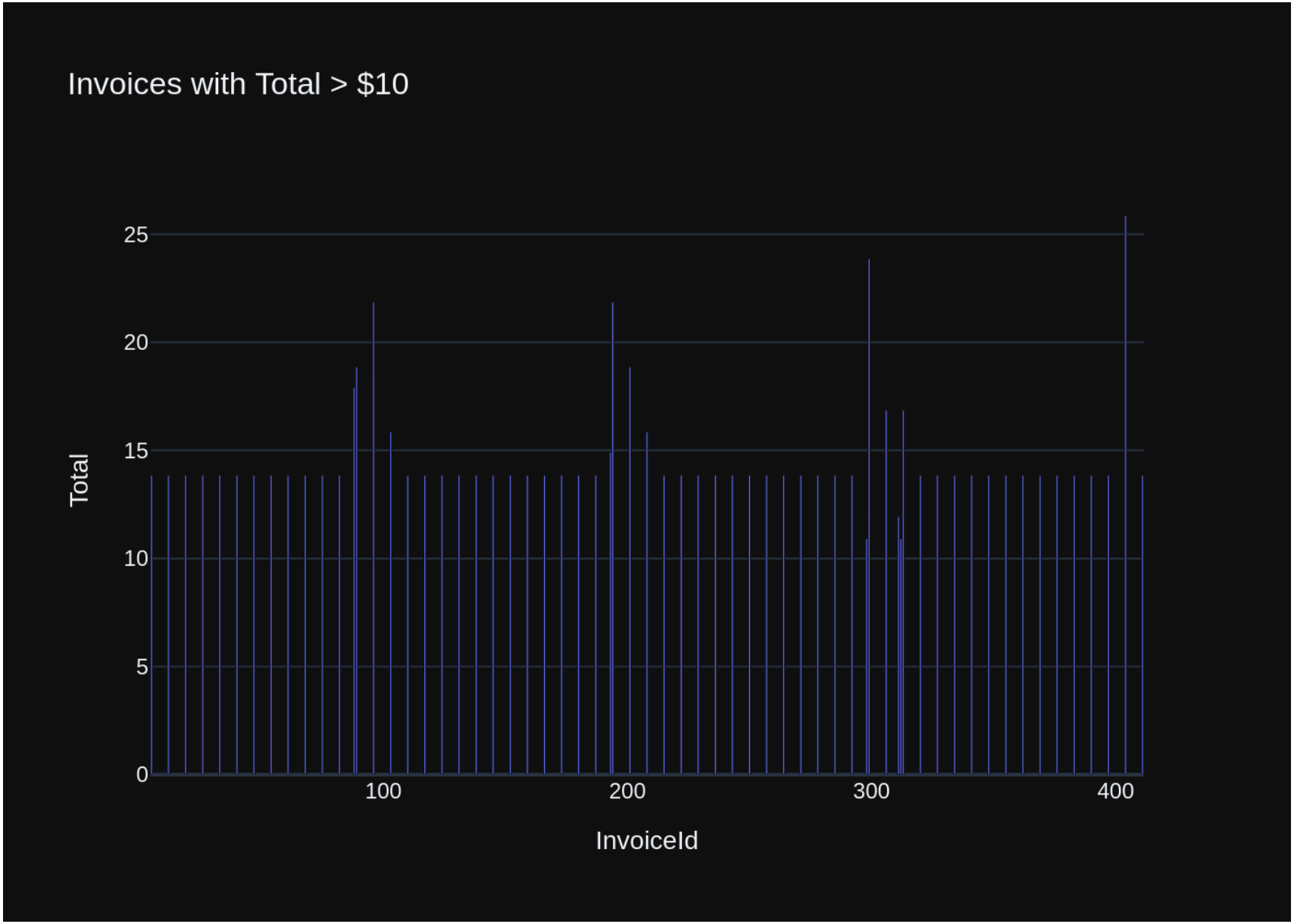
    Milliseconds      Bytes  UnitPrice
0         310622   10144730       0.99
1         249391    5988186       0.99
2         274155    9018565       0.99
3         260675    8497116       0.99
4         564009   18360449       0.99
5         158275    5203430       0.99
6         302994    9929799       0.99
7         149995    4913383       0.99
8         252316    8244843       0.99
9         247222    8249453       0.99
10        230974    7517083       0.99
11        247719    8043765       0.99
12        287973    9369385       0.99
13        142027    4631104       0.99
14        189152    6162894       0.99
15        221387    7251478       0.99
16       2610250  484583988       1.99
17       2616410  183867185       1.99
18        353567   11542247       0.99
19        280764    9306737       0.99
20        215084    3499018       0.99
21        209573    3426106       0.99  ,
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 with Name Containing "What"'},
                    'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'Name'}},
                    '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 format 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(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 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 INDEX IFK_CustomerSupportRepId 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 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\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': 'How many customers are there'}, {'role': 'assistant', 'content': 'SELECT COUNT(*)\nFROM Customer;'}, {'role': 'user', 'content': 'How many records are in table called customer'}, {'role': 'assistant', 'content': 'SELECT COUNT(*)\nFROM Customer;'}, {'role': 'user', 'content': 'what are the top 5 countries that customers com

```
e from?'}, {'role': 'assistant', 'content': 'SELECT Country, COUNT(*) AS CustomerCount\nFROM Customer\nGROU
P 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 Album.Title, Artist.Na
me\nFROM Album\nJOIN Artist ON Album.ArtistId = Artist.ArtistId;'}, {'role': 'user', 'content': '  \n    Fi
nd all tracks with a name containing "What" (case-insensitive)\n'}, {'role': 'assistant', 'content': "SELEC
T *\nFROM Track\nWHERE Name LIKE '%What%';"}, {'role': 'user', 'content': 'Show me a list of tables in the
SQLite database'}, {'role': 'assistant', 'content': "SELECT name\nFROM sqlite_master\nWHERE type = 'tabl
e';"}, {'role': 'user', 'content': '  \n    Get the total number of invoices for each customer\n'}]
Using model gpt-3.5-turbo for 1173.5 tokens (approx)
SELECT CustomerId, COUNT(*) AS TotalInvoices
FROM Invoice
GROUP BY CustomerId;
SELECT CustomerId, COUNT(*) AS TotalInvoices
FROM Invoice
GROUP BY CustomerId;
SELECT CustomerId, COUNT(*) AS TotalInvoices
FROM Invoice
GROUP BY CustomerId;
    CustomerId  TotalInvoices
0            1             7
1            2             7
2            3             7
3            4             7
4            5             7
5            6             7
6            7             7
7            8             7
8            9             7
9           10             7
10          11             7
11          12             7
12          13             7
13          14             7
14          15             7
15          16             7
16          17             7
17          18             7
18          19             7
19          20             7
20          21             7
21          22             7
22          23             7
```

```
23          24              7
24          25              7
25          26              7
26          27              7
27          28              7
28          29              7
29          30              7
30          31              7
31          32              7
32          33              7
33          34              7
34          35              7
35          36              7
36          37              7
37          38              7
38          39              7
39          40              7
40          41              7
41          42              7
42          43              7
43          44              7
44          45              7
45          46              7
46          47              7
47          48              7
48          49              7
49          50              7
50          51              7
51          52              7
52          53              7
53          54              7
54          55              7
55          56              7
56          57              7
57          58              7
58          59              6
Using model gpt-3.5-turbo for 185.5 tokens (approx)
```


Total Number of Invoices for Each Customer
TotalInvoices
0
1
2
3
4
5
6
7
10
20
30
40
50
CustomerId

Out[24]: ('SELECT CustomerId, COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY CustomerId;',

	CustomerId	TotalInvoices
0	1	7
1	2	7
2	3	7
3	4	7
4	5	7
5	6	7
6	7	7
7	8	7
8	9	7
9	10	7
10	11	7
11	12	7
12	13	7
13	14	7
14	15	7
15	16	7
16	17	7
17	18	7
18	19	7
19	20	7
20	21	7
21	22	7
22	23	7
23	24	7
24	25	7
25	26	7
26	27	7
27	28	7
28	29	7
29	30	7
30	31	7
31	32	7
32	33	7
33	34	7
34	35	7
35	36	7
36	37	7
37	38	7
38	39	7
39	40	7

```
40          41              7
41          42              7
42          43              7
43          44              7
44          45              7
45          46              7
46          47              7
47          48              7
48          49              7
49          50              7
50          51              7
51          52              7
52          53              7
53          54              7
54          55              7
55          56              7
56          57              7
57          58              7
58          59              6,
Figure({
    'data': [{'alignmentgroup': 'True',
              'hovertemplate': 'CustomerId=%{x}<br>TotalInvoices=%{y}<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 for Each Customer'},
                    'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'CustomerId'}},
                    'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'TotalInvoices'}}}
})
```

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 CustomerI

```
d, COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY CustomerId;'}, {'role': 'user', 'content': 'what are t
he 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': 'How many records are in table called customer'}, {'role': 'assistant', 'content': 'SELECT C
OUNT(*)\nFROM Customer;'}, {'role': 'user', 'content': 'How many customers are there'}, {'role': 'assistan
t', 'content': 'SELECT COUNT(*)\nFROM Customer;'}, {'role': 'user', 'content': '  \n    List all albums and
their corresponding artist names  \n'}, {'role': 'assistant', 'content': 'SELECT Album.Title, Artist.Name\n
FROM Album\nJOIN Artist ON Album.ArtistId = Artist.ArtistId;'}, {'role': 'user', 'content': '  \n    Find a
ll tracks with a name containing "What" (case-insensitive)\n'}, {'role': 'assistant', 'content': "SELECT *
\nFROM Track\nWHERE Name LIKE '%What%';"}, {'role': 'user', 'content': 'Show me a list of tables in the SQL
ite database'}, {'role': 'assistant', 'content': "SELECT name\nFROM sqlite_master\nWHERE type = 'table';"},
{'role': 'user', 'content': '  \n    Find the total number of invoices per country:\n'}]
Using model gpt-3.5-turbo for 1257.25 tokens (approx)
SELECT BillingCountry, COUNT(*) AS TotalInvoices
FROM Invoice
GROUP BY BillingCountry;
SELECT BillingCountry, COUNT(*) AS TotalInvoices
FROM Invoice
GROUP BY BillingCountry;
SELECT BillingCountry, COUNT(*) AS TotalInvoices
FROM Invoice
GROUP BY BillingCountry;
    BillingCountry  TotalInvoices
0        Argentina             7
1        Australia             7
2          Austria             7
3          Belgium             7
4           Brazil            35
5           Canada            56
6            Chile             7
7   Czech Republic            14
8          Denmark             7
9          Finland             7
10          France            35
11         Germany            28
12         Hungary             7
13           India            13
14         Ireland             7
15           Italy             7
16     Netherlands             7
17          Norway             7
18          Poland             7
```

```
19        Portugal             14
20           Spain              7
21          Sweden              7
22             USA             91
23  United Kingdom             21
Using model gpt-3.5-turbo for 187.5 tokens (approx)
```



Out[25]: ('SELECT BillingCountry, COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY BillingCountry;',

	BillingCountry	TotalInvoices
0	Argentina	7
1	Australia	7
2	Austria	7
3	Belgium	7
4	Brazil	35
5	Canada	56
6	Chile	7
7	Czech Republic	14
8	Denmark	7
9	Finland	7
10	France	35
11	Germany	28
12	Hungary	7
13	India	13
14	Ireland	7
15	Italy	7
16	Netherlands	7
17	Norway	7
18	Poland	7
19	Portugal	14
20	Spain	7
21	Sweden	7
22	USA	91
23	United Kingdom	21,

```
Figure({
    'data': [{'alignmentgroup': 'True',
              'hovertemplate': 'BillingCountry=%{x}<br>TotalInvoices=%{y}<extra></extra>',
              'legendgroup': '',
              'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
              'name': '',
              'offsetgroup': '',
              'orientation': 'v',
              'showlegend': False,
              'textposition': 'auto',
              'type': 'bar',
              'x': array(['Argentina', 'Australia', 'Austria', 'Belgium', 'Brazil', 'Canada',
                          'Chile', 'Czech Republic', 'Denmark', 'Finland', 'France', 'Germany',
                          'Hungary', 'India', 'Ireland', 'Italy', 'Netherlands', 'Norway',
                          'Poland', 'Portugal', 'Spain', 'Sweden', 'USA', 'United Kingdom'],
                         dtype=object),
```

```
                  'xaxis': 'x',
                  'y': array([ 7,  7,  7,  7, 35, 56,  7, 14,  7,  7, 35, 28,  7, 13,  7,  7,  7,  7,
                               7, 14,  7,  7, 91, 21]),
                  'yaxis': 'y'}],
      'layout': {'barmode': 'relative',
                 'legend': {'tracegroupgap': 0},
                 'template': '...',
                 'title': {'text': 'Total Invoices per Country'},
                 'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'BillingCountry'}},
                 'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'TotalInvoices'}}}
 }))
```

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 format instructions. \n===Tables \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_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 UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)\n\nCREATE INDEX IFK_InvoiceLineTrackId ON InvoiceLine (TrackId)\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 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 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 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 CustomerId, COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY CustomerId;'}, {'role': 'user', 'content': ' \n   Find the total number of invoices per country:\n'}, {'role': 'assistant', 'content': 'SELECT BillingCountry, COUNT

```
(*) AS TotalInvoices\nFROM Invoice\nGROUP BY BillingCountry;'}, {'role': 'user', 'content': 'How many recor
ds are in table called customer'}, {'role': 'assistant', 'content': 'SELECT COUNT(*)\nFROM Customer;'}, {'r
ole': 'user', 'content': 'How many customers are there'}, {'role': 'assistant', 'content': 'SELECT COUNT(*)
\nFROM Customer;'}, {'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    List all albums and their
corresponding artist names  \n'}, {'role': 'assistant', 'content': 'SELECT Album.Title, Artist.Name\nFROM A
lbum\nJOIN Artist ON Album.ArtistId = Artist.ArtistId;'}, {'role': 'user', 'content': '  \n    Find all tra
cks with a name containing "What" (case-insensitive)\n'}, {'role': 'assistant', 'content': "SELECT *\nFROM
Track\nWHERE Name LIKE '%What%';"}, {'role': 'user', 'content': 'Show me a list of tables in the SQLite dat
abase'}, {'role': 'assistant', 'content': "SELECT name\nFROM sqlite_master\nWHERE type = 'table';"}, {'rol
e': 'user', 'content': '  \n    List all invoices with a total exceeding $10:\n'}]
Using model gpt-3.5-turbo for 1241.25 tokens (approx)
SELECT *
FROM Invoice
WHERE Total > 10;
SELECT *
FROM Invoice
WHERE Total > 10;
SELECT *
FROM Invoice
WHERE Total > 10;
    InvoiceId  CustomerId          InvoiceDate             BillingAddress  \
0           5          23  2009-01-11 00:00:00            69 Salem Street
1          12           2  2009-02-11 00:00:00    Theodor-Heuss-Straße 34
2          19          40  2009-03-14 00:00:00             8, Rue Hanovre
3          26          19  2009-04-14 00:00:00            1 Infinite Loop
4          33          57  2009-05-15 00:00:00            Calle Lira, 198
..        ...         ...                  ...                        ...
59        383          10  2013-08-12 00:00:00  Rua Dr. Falcão Filho, 155
60        390          48  2013-09-12 00:00:00      Lijnbaansgracht 120bg
61        397          27  2013-10-13 00:00:00            1033 N Park Ave
62        404           6  2013-11-13 00:00:00              Rilská 3174/6
63        411          44  2013-12-14 00:00:00            Porthaninkatu 9

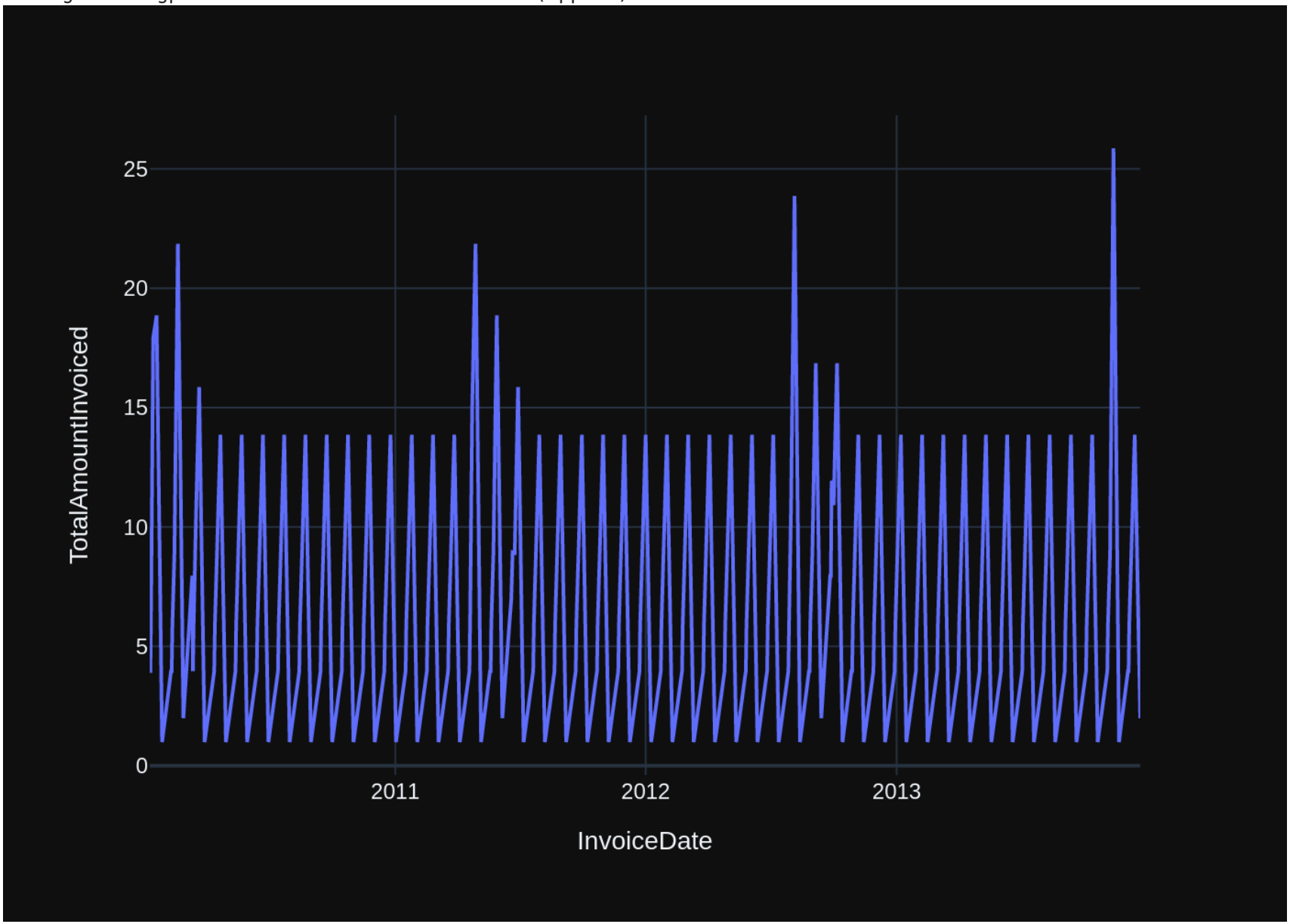
   BillingCity BillingState  BillingCountry BillingPostalCode  Total
0       Boston           MA             USA              2113  13.86
1    Stuttgart         None         Germany             70174  13.86
2        Paris         None          France             75002  13.86
3    Cupertino           CA             USA             95014  13.86
4     Santiago         None           Chile              None  13.86
..         ...          ...             ...               ...    ...
```

```
59   São Paulo           SP          Brazil         01007-010  13.86
60   Amsterdam           VV     Netherlands              1016  13.86
61      Tucson           AZ             USA             85719  13.86
62      Prague         None  Czech Republic             14300  25.86
63    Helsinki         None         Finland             00530  13.86

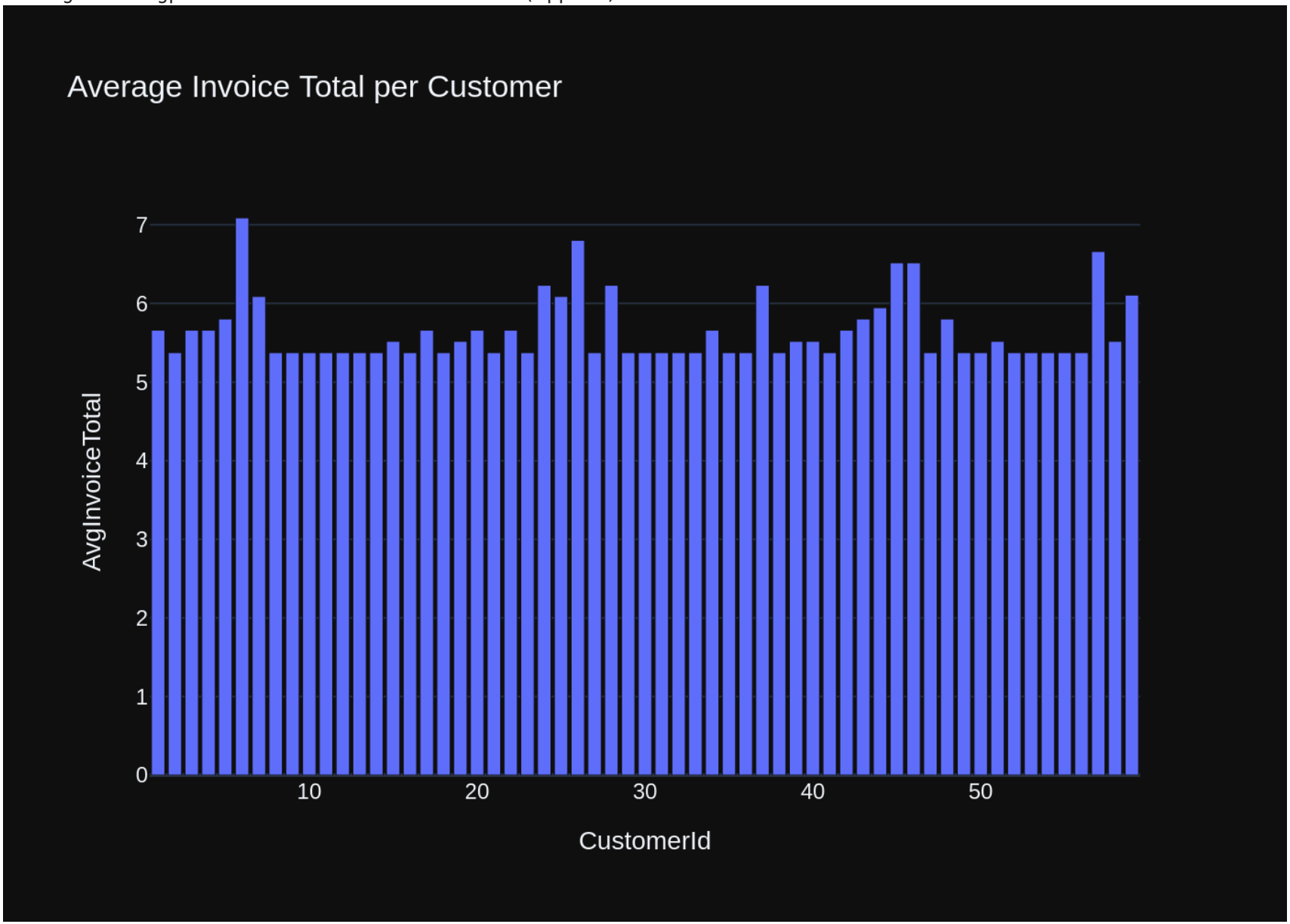
[64 rows x 9 columns]
Using model gpt-3.5-turbo for 228.25 tokens (approx)
```


Invoices with Total > $10
Total
25
20
15
10
5
0
100
200
300
400
InvoiceId

Out[26]:
```
('SELECT *\nFROM Invoice\nWHERE Total > 10;',
    InvoiceId  CustomerId          InvoiceDate             BillingAddress  \
0           5          23  2009-01-11 00:00:00            69 Salem Street
1          12           2  2009-02-11 00:00:00    Theodor-Heuss-Straße 34
2          19          40  2009-03-14 00:00:00             8, Rue Hanovre
3          26          19  2009-04-14 00:00:00            1 Infinite Loop
4          33          57  2009-05-15 00:00:00            Calle Lira, 198
..        ...         ...                  ...                        ...
59        383          10  2013-08-12 00:00:00  Rua Dr. Falcão Filho, 155
60        390          48  2013-09-12 00:00:00      Lijnbaansgracht 120bg
61        397          27  2013-10-13 00:00:00            1033 N Park Ave
62        404           6  2013-11-13 00:00:00              Rilská 3174/6
63        411          44  2013-12-14 00:00:00            Porthaninkatu 9

   BillingCity BillingState  BillingCountry BillingPostalCode  Total
0       Boston           MA             USA              2113  13.86
1    Stuttgart         None         Germany             70174  13.86
2        Paris         None          France             75002  13.86
3    Cupertino           CA             USA             95014  13.86
4     Santiago         None           Chile              None  13.86
..         ...          ...             ...               ...    ...
59   São Paulo           SP          Brazil         01007-010  13.86
60   Amsterdam           VV     Netherlands              1016  13.86
61      Tucson           AZ             USA             85719  13.86
62      Prague         None  Czech Republic             14300  25.86
63    Helsinki         None         Finland             00530  13.86

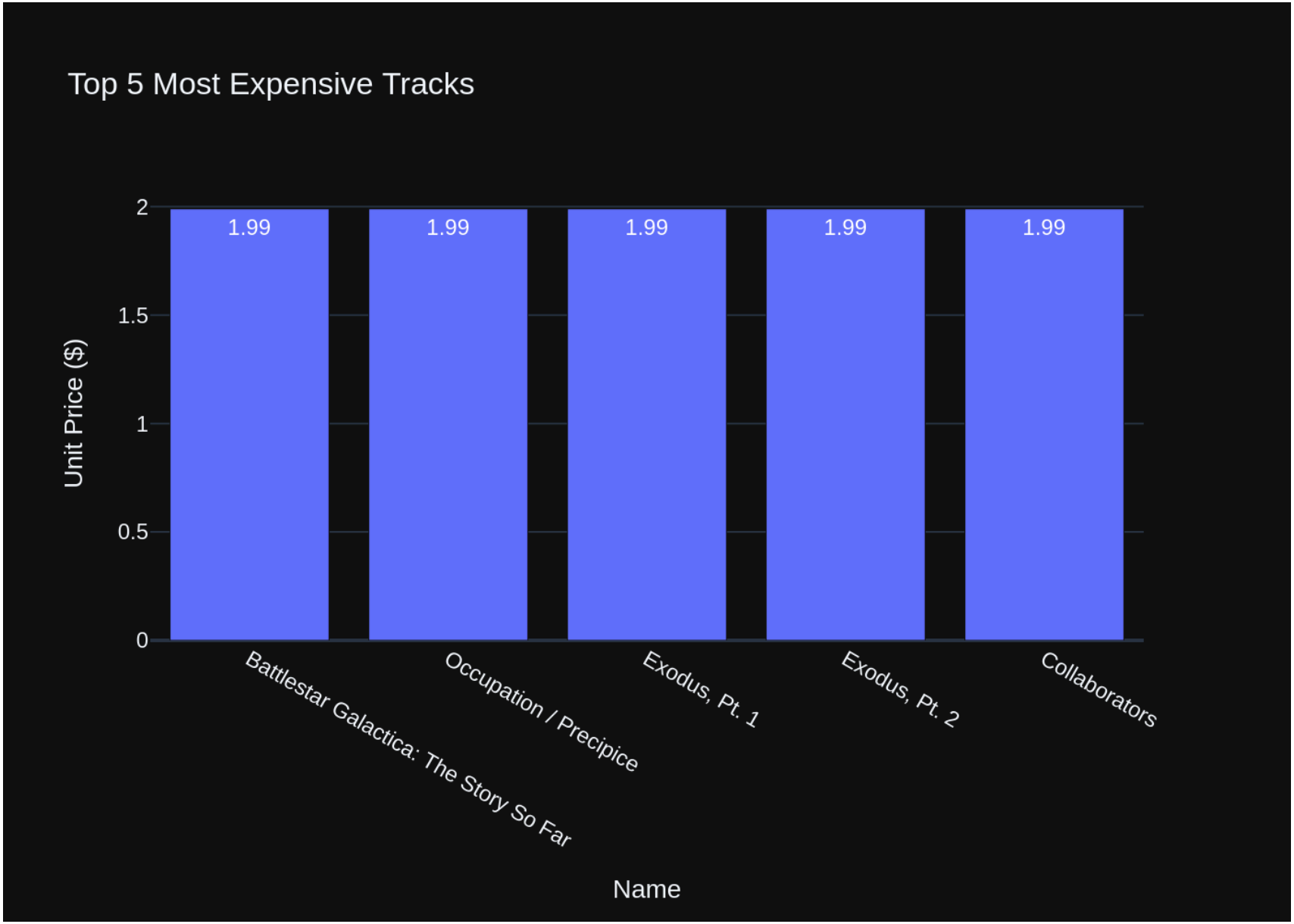
[64 rows x 9 columns],
Figure({
    'data': [{'alignmentgroup': 'True',
              'hovertemplate': 'InvoiceId=%{x}<br>Total=%{y}<extra></extra>',
              'legendgroup': '',
              'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
              'name': '',
              'offsetgroup': '',
              'orientation': 'v',
              'showlegend': False,
              'textposition': 'auto',
              'type': 'bar',
              'x': array([  5,  12,  19,  26,  33,  40,  47,  54,  61,  68,  75,  82,  88,  89,
                           96, 103, 110, 117, 124, 131, 138, 145, 152, 159, 166, 173, 180, 187,
                          193, 194, 201, 208, 215, 222, 229, 236, 243, 250, 257, 264, 271, 278,
```

```
                           285, 292, 298, 299, 306, 311, 312, 313, 320, 327, 334, 341, 348, 355,
                           362, 369, 376, 383, 390, 397, 404, 411]),
                'xaxis': 'x',
                'y': array([13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86,
                           13.86, 13.86, 17.91, 18.86, 21.86, 15.86, 13.86, 13.86, 13.86, 13.86,
                           13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 14.91, 21.86,
                           18.86, 15.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86,
                           13.86, 13.86, 13.86, 13.86, 10.91, 23.86, 16.86, 11.94, 10.91, 16.86,
                           13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86,
                           13.86, 13.86, 25.86, 13.86]),
                'yaxis': 'y'}],
     'layout': {'barmode': 'relative',
                'legend': {'tracegroupgap': 0},
                'template': '...',
                'title': {'text': 'Invoices with Total > $10'},
                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'InvoiceId'}},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'Total'}}}
 }))
```

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

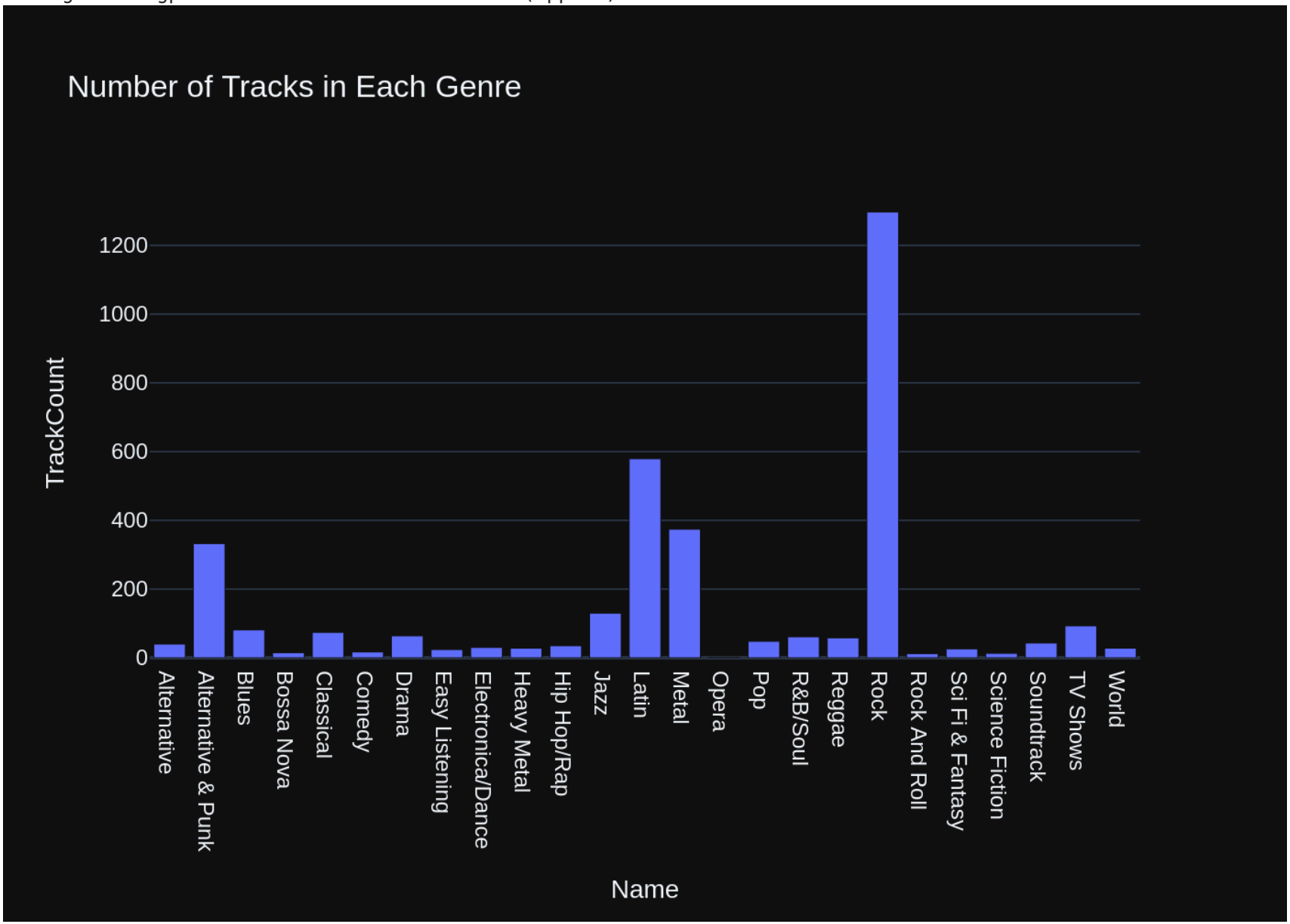
vn.ask(question=question)
```

```
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 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 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_InvoiceLineInvoiceId ON InvoiceLine (InvoiceId)\n\nCREATE INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)\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 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 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 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 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 *\nFROM Invoice\nWHERE T
otal > 10;'}, {'role': 'user', 'content': '  \n    Find the total number of invoices per country:\n'}, {'ro
le': 'assistant', 'content': 'SELECT BillingCountry, COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY Bill
ingCountry;'}, {'role': 'user', 'content': '  \n    Get the total number of invoices for each customer\n'},
{'role': 'assistant', 'content': 'SELECT CustomerId, COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY Cust
omerId;'}, {'role': 'user', 'content': 'How many records are in table called customer'}, {'role': 'assistan
t', 'content': 'SELECT COUNT(*)\nFROM Customer;'}, {'role': 'user', 'content': 'How many customers are ther
e'}, {'role': 'assistant', 'content': 'SELECT COUNT(*)\nFROM 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    Find all tracks with a name containing "What" (case-insensitive)\n'}, {'role':
'assistant', 'content': "SELECT *\nFROM Track\nWHERE Name LIKE '%What%';"}, {'role': 'user', 'content': '
\n    List all albums and their corresponding artist names  \n'}, {'role': 'assistant', 'content': 'SELECT
Album.Title, Artist.Name\nFROM Album\nJOIN Artist ON Album.ArtistId = Artist.ArtistId;'}, {'role': 'user',
'content': 'Show me a list of tables in the SQLite database'}, {'role': 'assistant', 'content': "SELECT nam
e\nFROM sqlite_master\nWHERE type = 'table';"}, {'role': 'user', 'content': '  \n    Find all invoices sinc
e 2010 and the total amount invoiced:\n'}]
Using model gpt-3.5-turbo for 1395.25 tokens (approx)
SELECT InvoiceDate, SUM(Total) AS TotalAmountInvoiced
FROM Invoice
WHERE InvoiceDate >= '2010-01-01'
GROUP BY InvoiceDate;
SELECT InvoiceDate, SUM(Total) AS TotalAmountInvoiced
FROM Invoice
WHERE InvoiceDate >= '2010-01-01'
GROUP BY InvoiceDate;
SELECT InvoiceDate, SUM(Total) AS TotalAmountInvoiced
FROM Invoice
WHERE InvoiceDate >= '2010-01-01'
GROUP BY InvoiceDate;
             InvoiceDate  TotalAmountInvoiced
0    2010-01-08 00:00:00                3.96
1    2010-01-09 00:00:00                3.96
2    2010-01-10 00:00:00                6.94
3    2010-01-13 00:00:00               17.91
4    2010-01-18 00:00:00               18.86
..                   ...                 ...
277  2013-12-05 00:00:00                3.96
278  2013-12-06 00:00:00                5.94
279  2013-12-09 00:00:00                8.91
```

```
280  2013-12-14 00:00:00                13.86
281  2013-12-22 00:00:00                 1.99

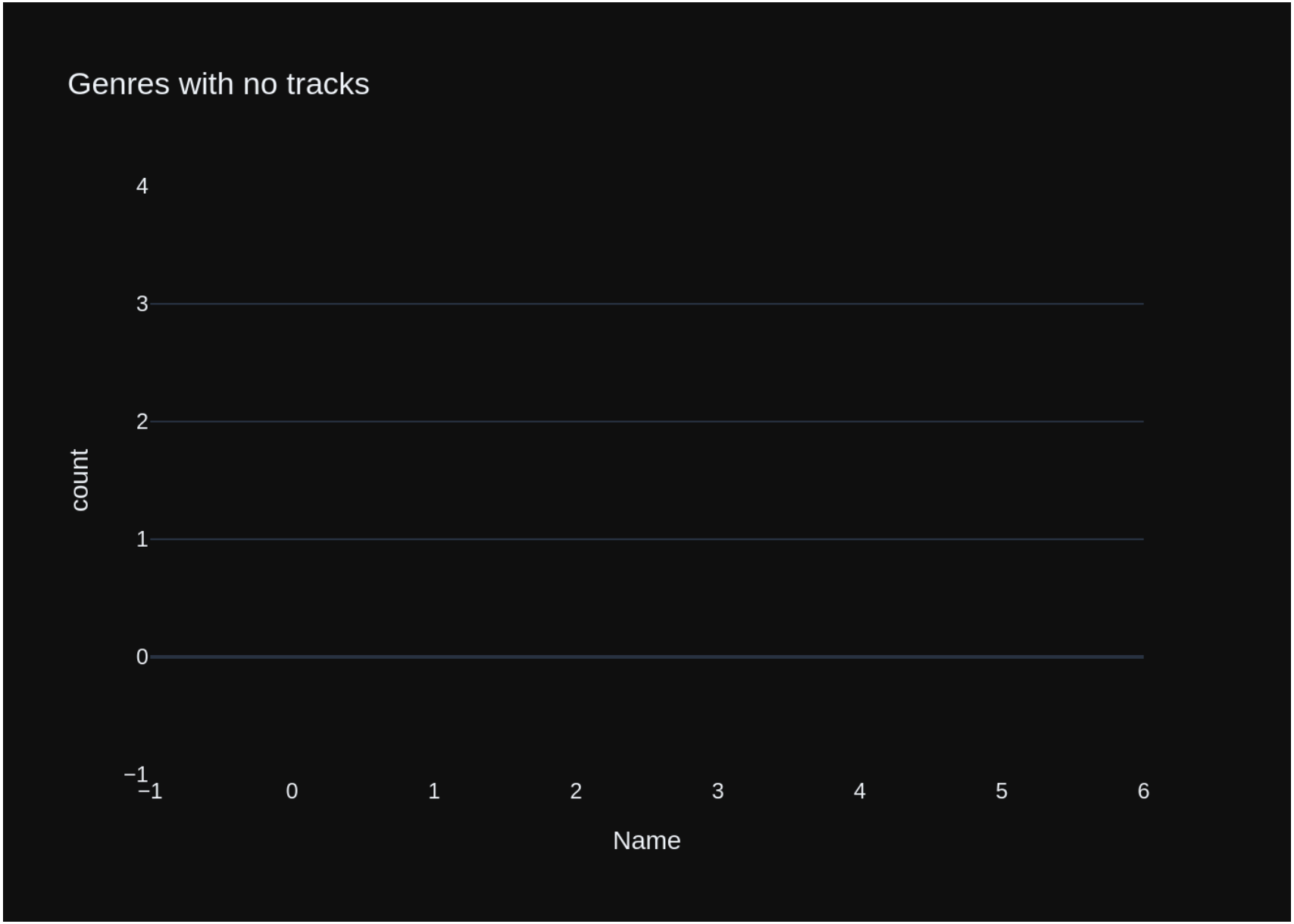
[282 rows x 2 columns]
Using model gpt-3.5-turbo for 202.75 tokens (approx)
```



Out[27]:
```
("SELECT InvoiceDate, SUM(Total) AS TotalAmountInvoiced\nFROM Invoice\nWHERE InvoiceDate >= '2010-01-01'\n
GROUP BY InvoiceDate;",
              InvoiceDate  TotalAmountInvoiced
 0    2010-01-08 00:00:00                3.96
 1    2010-01-09 00:00:00                3.96
 2    2010-01-10 00:00:00                6.94
 3    2010-01-13 00:00:00               17.91
 4    2010-01-18 00:00:00               18.86
 ..                   ...                 ...
 277  2013-12-05 00:00:00                3.96
 278  2013-12-06 00:00:00                5.94
 279  2013-12-09 00:00:00                8.91
 280  2013-12-14 00:00:00               13.86
 281  2013-12-22 00:00:00                1.99

 [282 rows x 2 columns],
 Figure({
     'data': [{'hovertemplate': 'InvoiceDate=%{x}<br>TotalAmountInvoiced=%{y}<extra></extra>',
               'legendgroup': '',
               'line': {'color': '#636efa', 'dash': 'solid'},
               'marker': {'symbol': 'circle'},
               'mode': 'lines',
               'name': '',
               'orientation': 'v',
               'showlegend': False,
               'type': 'scatter',
               'x': array(['2010-01-08 00:00:00', '2010-01-09 00:00:00', '2010-01-10 00:00:00',
                           ..., '2013-12-09 00:00:00', '2013-12-14 00:00:00',
                           '2013-12-22 00:00:00'], dtype=object),
               'xaxis': 'x',
               'y': array([ 3.96,  3.96,  6.94, ...,  8.91, 13.86,  1.99]),
               'yaxis': 'y'}],
     'layout': {'legend': {'tracegroupgap': 0},
                'margin': {'t': 60},
                'template': '...',
                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'InvoiceDate'}},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'TotalAmountInvoiced'}}}
 }))
```

In [28]:
```
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 format 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 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 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 INDEX IFK_CustomerSupportRepId ON Customer (SupportRepId)\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 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   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 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_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 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 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 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': '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    Find all invoices
since 2010 and the total amount invoiced:\n'}, {'role': 'assistant', 'content': "SELECT InvoiceDate, SUM(To
tal) AS TotalAmountInvoiced\nFROM Invoice\nWHERE InvoiceDate >= '2010-01-01'\nGROUP BY InvoiceDate;"}, {'ro
le': 'user', 'content': '  \n    Get the total number of invoices for each customer\n'}, {'role': 'assistan
t', 'content': 'SELECT CustomerId, COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY CustomerId;'}, {'rol
e': 'user', 'content': '  \n    List all albums and their corresponding artist names  \n'}, {'role': 'assis
tant', 'content': 'SELECT Album.Title, Artist.Name\nFROM Album\nJOIN Artist ON Album.ArtistId = Artist.Arti
stId;'}, {'role': 'user', 'content': '  \n    Find the total number of invoices per country:\n'}, {'role':
'assistant', 'content': 'SELECT BillingCountry, COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY BillingCo
untry;'}, {'role': 'user', 'content': '  \n    List all invoices with a total exceeding $10:\n'}, {'role':
'assistant', 'content': 'SELECT *\nFROM Invoice\nWHERE Total > 10;'}, {'role': 'user', 'content': 'How many
records are in table called customer'}, {'role': 'assistant', 'content': 'SELECT COUNT(*)\nFROM Custome
r;'}, {'role': 'user', 'content': 'How many customers are there'}, {'role': 'assistant', 'content': 'SELECT
COUNT(*)\nFROM Customer;'}, {'role': 'user', 'content': '  \n    Find all tracks with a name containing "Wh
at" (case-insensitive)\n'}, {'role': 'assistant', 'content': "SELECT *\nFROM Track\nWHERE Name LIKE '%Wha
t%';"}, {'role': 'user', 'content': 'Show me a list of tables in the SQLite database'}, {'role': 'assistan
t', 'content': "SELECT name\nFROM sqlite_master\nWHERE type = 'table';"}, {'role': 'user', 'content': "  \n
List all employees and their reporting manager's name (if any):\n"}]
Using model gpt-3.5-turbo for 1408.25 tokens (approx)
SELECT e.FirstName || ' ' || e.LastName AS EmployeeName,
       m.FirstName || ' ' || m.LastName AS ManagerName
FROM Employee e
LEFT JOIN Employee m ON e.ReportsTo = m.EmployeeId;
SELECT e.FirstName || ' ' || e.LastName AS EmployeeName,
       m.FirstName || ' ' || m.LastName AS ManagerName
FROM Employee e
LEFT JOIN Employee m ON e.ReportsTo = m.EmployeeId;
SELECT e.FirstName || ' ' || e.LastName AS EmployeeName,
       m.FirstName || ' ' || m.LastName AS ManagerName
FROM Employee e
LEFT JOIN Employee m ON e.ReportsTo = m.EmployeeId;
       EmployeeName       ManagerName
0      Andrew Adams              None
1     Nancy Edwards      Andrew Adams
2      Jane Peacock     Nancy Edwards
3     Margaret Park     Nancy Edwards
4     Steve Johnson     Nancy Edwards
5  Michael Mitchell      Andrew Adams
6       Robert King  Michael Mitchell
```

```
7    Laura Callahan  Michael Mitchell
Using model gpt-3.5-turbo for 214.25 tokens (approx)
```

Out[28]:
```
("SELECT e.FirstName || ' ' || e.LastName AS EmployeeName, \n       m.FirstName || ' ' || m.LastName AS Ma
nagerName\nFROM Employee e\nLEFT JOIN Employee m ON e.ReportsTo = m.EmployeeId;",
        EmployeeName       ManagerName
 0      Andrew Adams              None
 1     Nancy Edwards      Andrew Adams
 2      Jane Peacock     Nancy Edwards
 3     Margaret Park     Nancy Edwards
 4     Steve Johnson     Nancy Edwards
 5  Michael Mitchell      Andrew Adams
 6       Robert King  Michael Mitchell
 7    Laura Callahan  Michael Mitchell,
 Figure({
     'data': [{'alignmentgroup': 'True',
               'hovertemplate': 'EmployeeName=%{x}<br>ManagerName=%{y}<extra></extra>',
               'legendgroup': '',
               'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
               'name': '',
               'offsetgroup': '',
               'orientation': 'v',
               'showlegend': False,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Andrew Adams', 'Nancy Edwards', 'Jane Peacock', 'Margaret Park',
                           'Steve Johnson', 'Michael Mitchell', 'Robert King', 'Laura Callahan'],
                          dtype=object),
               'xaxis': 'x',
               'y': array([None, 'Andrew Adams', 'Nancy Edwards', 'Nancy Edwards', 'Nancy Edwards',
                           'Andrew Adams', 'Michael Mitchell', 'Michael Mitchell'], dtype=object),
               'yaxis': 'y'}],
     'layout': {'barmode': 'relative',
                'legend': {'tracegroupgap': 0},
                'template': '...',
                'title': {'text': 'Employee Reporting Managers'},
                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'EmployeeName'}},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'ManagerName'}}}
 }))
```

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 format 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 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   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 INDEX IFK_CustomerSupportRepId ON Customer (SupportRepId)\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 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 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 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\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 CustomerId, COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY CustomerId;'}, {'role': 'user', 'content': ' \n   Find all invoices since 2010 and the total amount invoiced:\n'}, {'role': 'assistant', 'content': "SELECT InvoiceD

ate, SUM(Total) AS TotalAmountInvoiced\nFROM Invoice\nWHERE InvoiceDate >= '2010-01-01'\nGROUP BY InvoiceDa
te;"}, {'role': 'user', 'content': ' \n   Find the total number of invoices per country:\n'}, {'role': 'a
ssistant', 'content': 'SELECT BillingCountry, COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY BillingCoun
try;'}, {'role': 'user', 'content': ' \n   List all invoices with a total exceeding $10:\n'}, {'role': 'a
ssistant', 'content': 'SELECT *\nFROM Invoice\nWHERE Total > 10;'}, {'role': 'user', 'content': 'How many c
ustomers are there'}, {'role': 'assistant', 'content': 'SELECT COUNT(*)\nFROM Customer;'}, {'role': 'user',
'content': 'what are the top 5 countries that customers come from?'}, {'role': 'assistant', 'content': 'SEL
ECT 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(*)\nFROM Customer;'}, {'role': 'user', 'content': " \n   List all employees and
their reporting manager's name (if any):\n"}, {'role': 'assistant', 'content': "SELECT e.FirstName || ' ' |
| e.LastName AS EmployeeName, \n   m.FirstName || ' ' || m.LastName AS ManagerName\nFROM Employee e\nLE
FT JOIN Employee m ON e.ReportsTo = m.EmployeeId;"}, {'role': 'user', 'content': ' \n   Find all tracks w
ith a name containing "What" (case-insensitive)\n'}, {'role': 'assistant', 'content': "SELECT *\nFROM Track
\nWHERE Name LIKE '%What%';"}, {'role': 'user', 'content': ' \n   List all albums and their corresponding
artist names \n'}, {'role': 'assistant', 'content': 'SELECT Album.Title, Artist.Name\nFROM Album\nJOIN Art
ist ON Album.ArtistId = Artist.ArtistId;'}, {'role': 'user', 'content': ' \n   Get the average invoice to
tal for each customer:\n'}]
Using model gpt-3.5-turbo for 1350.25 tokens (approx)
SELECT CustomerId, AVG(Total) AS AvgInvoiceTotal
FROM Invoice
GROUP BY CustomerId;
SELECT CustomerId, AVG(Total) AS AvgInvoiceTotal
FROM Invoice
GROUP BY CustomerId;
SELECT CustomerId, AVG(Total) AS AvgInvoiceTotal
FROM Invoice
GROUP BY CustomerId;

	CustomerId	AvgInvoiceTotal
0	1	5.660000
1	2	5.374286
2	3	5.660000
3	4	5.660000
4	5	5.802857
5	6	7.088571
6	7	6.088571
7	8	5.374286
8	9	5.374286
9	10	5.374286
10	11	5.374286
11	12	5.374286
12	13	5.374286

13	14	5.374286
14	15	5.517143
15	16	5.374286
16	17	5.660000
17	18	5.374286
18	19	5.517143
19	20	5.660000
20	21	5.374286
21	22	5.660000
22	23	5.374286
23	24	6.231429
24	25	6.088571
25	26	6.802857
26	27	5.374286
27	28	6.231429
28	29	5.374286
29	30	5.374286
30	31	5.374286
31	32	5.374286
32	33	5.374286
33	34	5.660000
34	35	5.374286
35	36	5.374286
36	37	6.231429
37	38	5.374286
38	39	5.517143
39	40	5.517143
40	41	5.374286
41	42	5.660000
42	43	5.802857
43	44	5.945714
44	45	6.517143
45	46	6.517143
46	47	5.374286
47	48	5.802857
48	49	5.374286
49	50	5.374286
50	51	5.517143
51	52	5.374286
52	53	5.374286
53	54	5.374286
54	55	5.374286

```
55          56          5.374286
56          57          6.660000
57          58          5.517143
58          59          6.106667
Using model gpt-3.5-turbo for 188.0 tokens (approx)
```



Out[29]: ('SELECT CustomerId, AVG(Total) AS AvgInvoiceTotal\nFROM Invoice\nGROUP BY CustomerId;',

	CustomerId	AvgInvoiceTotal
0	1	5.660000
1	2	5.374286
2	3	5.660000
3	4	5.660000
4	5	5.802857
5	6	7.088571
6	7	6.088571
7	8	5.374286
8	9	5.374286
9	10	5.374286
10	11	5.374286
11	12	5.374286
12	13	5.374286
13	14	5.374286
14	15	5.517143
15	16	5.374286
16	17	5.660000
17	18	5.374286
18	19	5.517143
19	20	5.660000
20	21	5.374286
21	22	5.660000
22	23	5.374286
23	24	6.231429
24	25	6.088571
25	26	6.802857
26	27	5.374286
27	28	6.231429
28	29	5.374286
29	30	5.374286
30	31	5.374286
31	32	5.374286
32	33	5.374286
33	34	5.660000
34	35	5.374286
35	36	5.374286
36	37	6.231429
37	38	5.374286
38	39	5.517143
39	40	5.517143

```
40          41          5.374286
41          42          5.660000
42          43          5.802857
43          44          5.945714
44          45          6.517143
45          46          6.517143
46          47          5.374286
47          48          5.802857
48          49          5.374286
49          50          5.374286
50          51          5.517143
51          52          5.374286
52          53          5.374286
53          54          5.374286
54          55          5.374286
55          56          5.374286
56          57          6.660000
57          58          5.517143
58          59          6.106667,
Figure({
    'data': [{'alignmentgroup': 'True',
              'hovertemplate': 'CustomerId=%{x}<br>AvgInvoiceTotal=%{y}<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([5.66      , 5.37428571, 5.66      , 5.66      , 5.80285714, 7.08857143,
                          6.08857143, 5.37428571, 5.37428571, 5.37428571, 5.37428571, 5.37428571,
                          5.37428571, 5.37428571, 5.51714286, 5.37428571, 5.66      , 5.37428571,
                          5.51714286, 5.66      , 5.37428571, 5.66      , 5.37428571, 6.23142857,
                          6.08857143, 6.80285714, 5.37428571, 6.23142857, 5.37428571, 5.37428571,
                          5.37428571, 5.37428571, 5.37428571, 5.66      , 5.37428571, 5.37428571,
                          6.23142857, 5.37428571, 5.51714286, 5.51714286, 5.37428571, 5.66      ,
```

```
                                5.80285714, 5.94571429, 6.51714286, 6.51714286, 5.37428571, 5.80285714,
                                5.37428571, 5.37428571, 5.51714286, 5.37428571, 5.37428571, 5.37428571,
                                5.37428571, 5.37428571, 6.66      , 5.51714286, 6.10666667]),
                   'yaxis': 'y'}],
        'layout': {'barmode': 'relative',
                   'legend': {'tracegroupgap': 0},
                   'template': '...',
                   'title': {'text': 'Average Invoice Total per Customer'},
                   'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'CustomerId'}},
                   'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'AvgInvoiceTotal'}}}
    }))
```

In [30]:
```
question = """
    Find the top 5 most expensive tracks (based on unit price):
"""

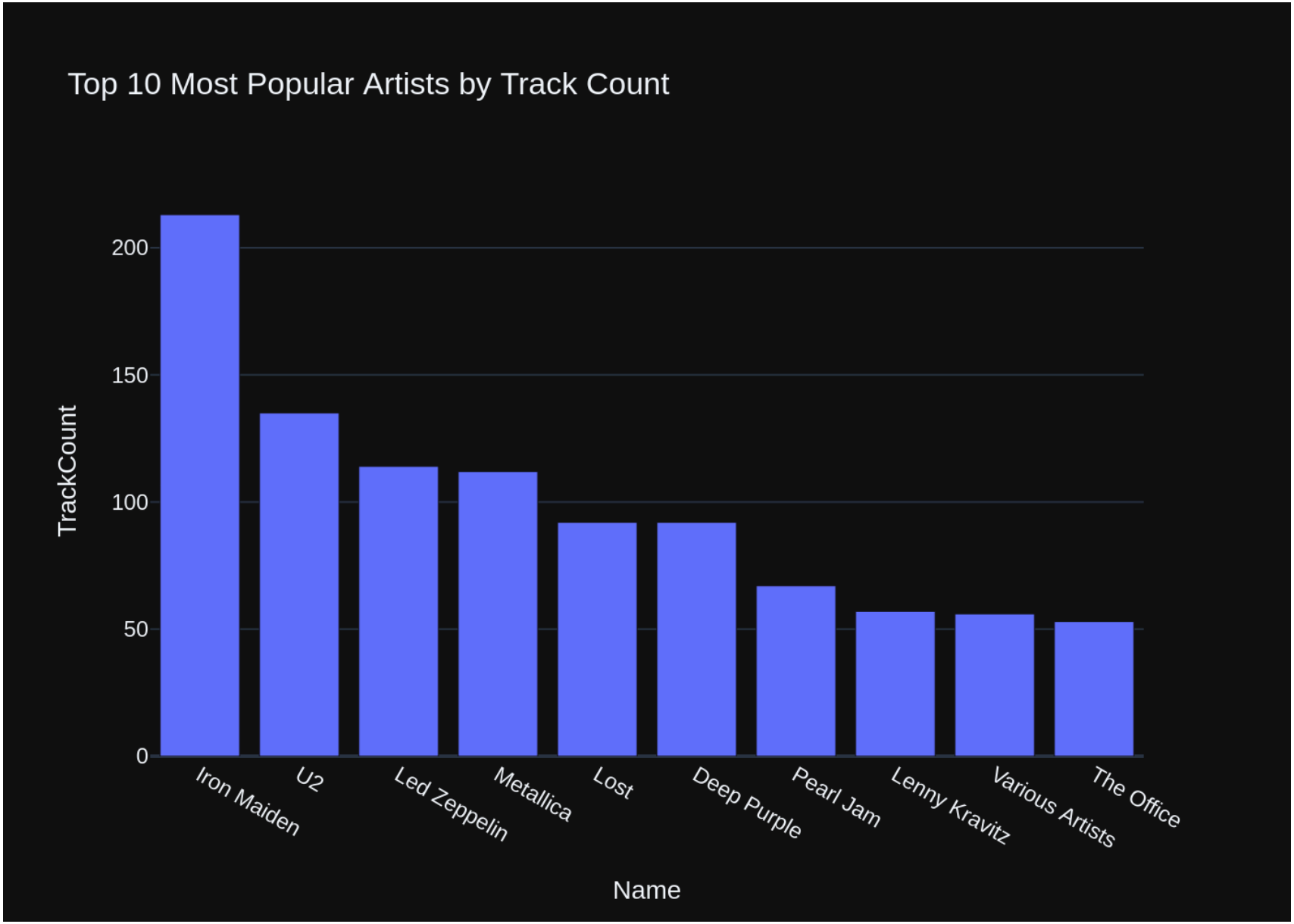
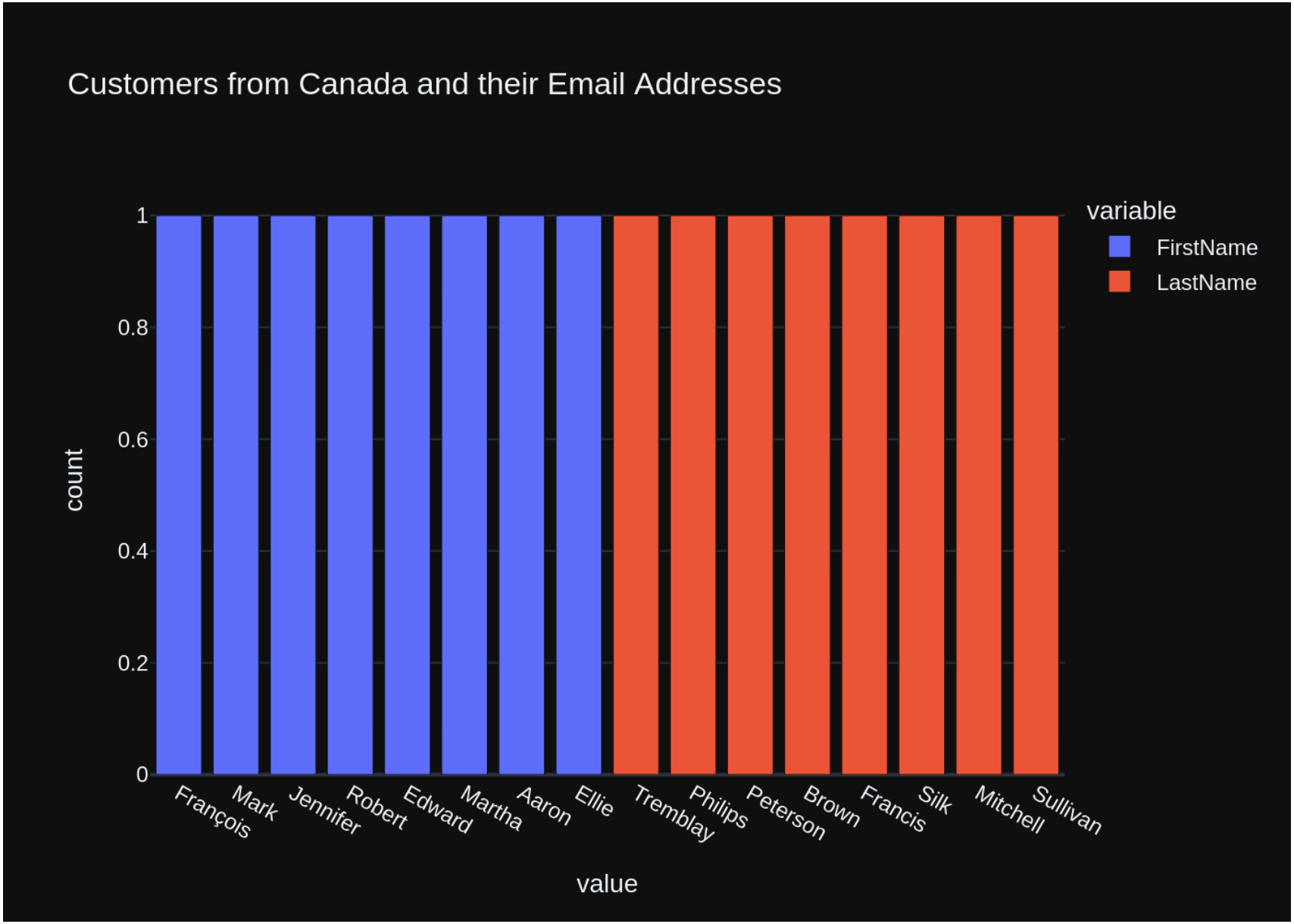
vn.ask(question=question)
```

```
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 format instructions. \n===Tables \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_TrackAlbumId ON Track (AlbumId)\n\nCREATE INDEX IFK_TrackGenreId ON Track (GenreId)\n\nCREATE INDEX IFK_PlaylistTrackTrackId ON PlaylistTrack (TrackId)\n\nCREATE INDEX IFK_InvoiceLineTrackId ON InvoiceLine (TrackId)\n\nCREATE INDEX IFK_TrackMediaTypeId ON Track (MediaTypeId)\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 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 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 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   Find all tracks with a name containing "What" (case-insensitive)\n'}, {'role': 'assistant', 'content': "SELECT *\nFROM Track\nWHERE Name LIKE '%What%';"}, {'role': 'user', 'content': ' \n   List all invoices with a total exceeding $10:\n'}, {'role': 'assistant', 'content': 'SELECT *\nFROM Invoice\nWHERE Total > 10;'}, {'role': 'user', 'content': ' \n   List all albums and their corresponding artist names \n'}, {'role': 'assistant', 'content': 'SELECT Album.Title, Artist.Name\nFROM Album\nJOIN Artist ON Album.ArtistId = Artist.ArtistId;'}, {'role': 'user', 'content': ' \n   Find all invoices since 2010 and the total amount invoiced:\n'}, {'role': 'assistant', 'content': "SELECT InvoiceDate, SUM(Total) AS TotalAmountInvoiced\nFROM Invoice\nWHERE InvoiceDate >= '2010-01-01'\nGROUP BY InvoiceDate;"}, {'role': 'user', 'content': ' \n   Get the average invoice total for each customer:\n'}, {'role': 'assistant', 'content': 'SELECT CustomerId, AVG(Total) AS AvgInvoiceTotal\nFROM Invoice\nGROUP BY CustomerId;'}, {'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   Find the total num

```
ber of invoices per country:\n'}, {'role': 'assistant', 'content': 'SELECT BillingCountry, COUNT(*) AS Tota
lInvoices\nFROM Invoice\nGROUP BY BillingCountry;'}, {'role': 'user', 'content': '  \n    Get the total num
ber of invoices for each customer\n'}, {'role': 'assistant', 'content': 'SELECT CustomerId, COUNT(*) AS Tot
alInvoices\nFROM Invoice\nGROUP BY CustomerId;'}, {'role': 'user', 'content': 'How many customers are ther
e'}, {'role': 'assistant', 'content': 'SELECT COUNT(*)\nFROM Customer;'}, {'role': 'user', 'content': 'Show
me a list of tables in the SQLite database'}, {'role': 'assistant', 'content': "SELECT name\nFROM sqlite_ma
ster\nWHERE type = 'table';"}, {'role': 'user', 'content': '  \n    Find the top 5 most expensive tracks (b
ased on unit price):\n'}]
Using model gpt-3.5-turbo for 1089.0 tokens (approx)
SELECT *
FROM Track
ORDER BY UnitPrice DESC
LIMIT 5;
SELECT *
FROM Track
ORDER BY UnitPrice DESC
LIMIT 5;
SELECT *
FROM Track
ORDER BY UnitPrice DESC
LIMIT 5;
   TrackId                                    Name  AlbumId  MediaTypeId  \
0     2819  Battlestar Galactica: The Story So Far      226            3
1     2820                  Occupation / Precipice      227            3
2     2821                            Exodus, Pt. 1      227            3
3     2822                            Exodus, Pt. 2      227            3
4     2823                            Collaborators      227            3

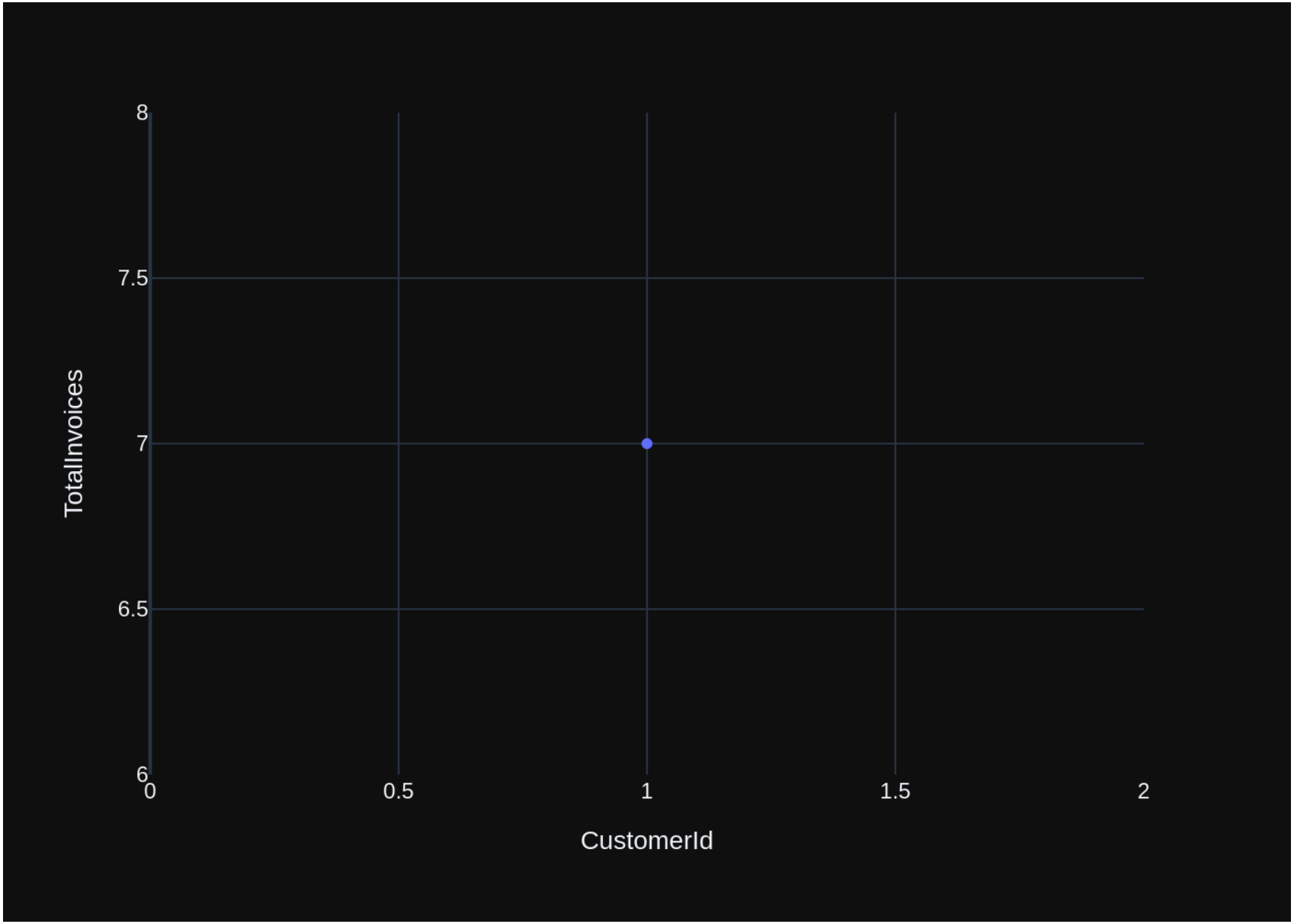
   GenreId Composer  Milliseconds       Bytes  UnitPrice
0       18     None       2622250   490750393       1.99
1       19     None       5286953  1054423946       1.99
2       19     None       2621708   475079441       1.99
3       19     None       2618000   466820021       1.99
4       19     None       2626626   483484911       1.99
Using model gpt-3.5-turbo for 223.75 tokens (approx)
```


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

Out[30]:
```
('SELECT *\nFROM Track\nORDER BY UnitPrice DESC\nLIMIT 5;',
   TrackId                                  Name  AlbumId  MediaTypeId  \
0     2819  Battlestar Galactica: The Story So Far      226            3
1     2820                 Occupation / Precipice      227            3
2     2821                         Exodus, Pt. 1      227            3
3     2822                         Exodus, Pt. 2      227            3
4     2823                         Collaborators      227            3

   GenreId Composer  Milliseconds       Bytes  UnitPrice
0       18     None       2622250   490750393       1.99
1       19     None       5286953  1054423946       1.99
2       19     None       2621708   475079441       1.99
3       19     None       2618000   466820021       1.99
4       19     None       2626626   483484911       1.99  ,
Figure({
    'data': [{'alignmentgroup': 'True',
              'hovertemplate': 'Name=%{x}<br>Unit Price ($)=%{text}<extra></extra>',
              'legendgroup': '',
              'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
              'name': '',
              'offsetgroup': '',
              'orientation': 'v',
              'showlegend': False,
              'text': array([1.99, 1.99, 1.99, 1.99, 1.99]),
              'textposition': 'auto',
              'type': 'bar',
              'x': array(['Battlestar Galactica: The Story So Far', 'Occupation / Precipice',
                          'Exodus, Pt. 1', 'Exodus, Pt. 2', 'Collaborators'], dtype=object),
              'xaxis': 'x',
              'y': array([1.99, 1.99, 1.99, 1.99, 1.99]),
              'yaxis': 'y'}],
    'layout': {'barmode': 'relative',
               'legend': {'tracegroupgap': 0},
               'template': '...',
               'title': {'text': 'Top 5 Most Expensive Tracks'},
               'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'Name'}},
               'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'Unit Price ($)'}}}
})
```

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 format instructions. \n===Tables \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_TrackGenreId ON Track (GenreId)\n\nCREATE TABLE Genre\n(\n   GenreId INTEGER NOT NULL,\n   Name NVARCHAR(120),\n   CONSTRAINT PK_Genre PRIMARY KEY (GenreId)\n)\n\nCREATE 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 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\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 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 Playlist\n(\n   PlaylistId 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 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   Find the top 5 most expensive tracks (based on unit price):\n'}, {'role': 'assistant', 'content': 'SELECT *\nFROM Track\nORDER BY UnitPrice DESC\nLIMIT 5;'}, {'role': 'user', 'content': ' \n   List all albums and their corresponding artist names \n'}, {'role': 'assistant', 'content': 'SELECT Album.Title, Artist.Name\nFROM Album\nJOIN Artist ON Album.ArtistId = Artist.ArtistId;'}, {'role': 'user', 'content': ' \n   Find all tracks with a name containing "What" (case-insensitive)\n'}, {'role': 'assistant', 'content': "SELECT *\nFROM Track\nWHERE Name LIKE '%What%';"}, {'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   Find the total number of invoices per country:\n'}, {'role': 'assistant', 'content': 'SELECT BillingCountry, COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY BillingCountry;'}, {'role': 'user', 'content': ' \n   Find all invoices since 2010 and the total amount invoiced:\n'}, {'role': 'assistant', 'content': "SELECT InvoiceDate, SUM(Total) AS TotalAmountInvoiced\nFROM Invoice\nWHERE InvoiceDate >= '2010-01-01'\nGROUP BY InvoiceDate;"}, {'role': 'user', 'content': ' \n   List all invoices with a total exceeding $10:\n'}, {'role': 'assistant', 'content': 'SELECT *\nFROM Invoice\nWHERE Total > 10;'}, {'role': 'user', 'content': ' \n   Get the total number of invoices for each customer\n'}, {'role': 'assistant', 'cont

```
ent': 'SELECT CustomerId, COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY CustomerId;'}, {'role': 'user',
'content': 'Show me a list of tables in the SQLite database'}, {'role': 'assistant', 'content': "SELECT nam
e\nFROM sqlite_master\nWHERE type = 'table';"}, {'role': 'user', 'content': 'How many records are in table
called customer'}, {'role': 'assistant', 'content': 'SELECT COUNT(*)\nFROM Customer;'}, {'role': 'user', 'c
ontent': '  \n    List all genres and the number of tracks in each genre:\n'}]
Using model gpt-3.5-turbo for 1022.75 tokens (approx)
SELECT Genre.Name, COUNT(*) AS TrackCount
FROM Track
JOIN Genre ON Track.GenreId = Genre.GenreId
GROUP BY Genre.Name;
SELECT Genre.Name, COUNT(*) AS TrackCount
FROM Track
JOIN Genre ON Track.GenreId = Genre.GenreId
GROUP BY Genre.Name;
SELECT Genre.Name, COUNT(*) AS TrackCount
FROM Track
JOIN Genre ON Track.GenreId = Genre.GenreId
GROUP BY Genre.Name;
                   Name  TrackCount
0          Alternative          40
1   Alternative & Punk         332
2                Blues          81
3           Bossa Nova          15
4            Classical          74
5               Comedy          17
6                Drama          64
7       Easy Listening          24
8    Electronica/Dance          30
9           Heavy Metal          28
10          Hip Hop/Rap          35
11                 Jazz         130
12                Latin         579
13                Metal         374
14                Opera           1
15                  Pop          48
16            R&B/Soul          61
17              Reggae          58
18                Rock        1297
19       Rock And Roll          12
20    Sci Fi & Fantasy          26
21     Science Fiction          13
22          Soundtrack          43
```

```
23            TV Shows          93
24               World          28
Using model gpt-3.5-turbo for 195.5 tokens (approx)
```



Out[31]:
```
('SELECT Genre.Name, COUNT(*) AS TrackCount\nFROM Track\nJOIN Genre ON Track.GenreId = Genre.GenreId\nGROU
P BY Genre.Name;',
                    Name  TrackCount
 0          Alternative          40
 1   Alternative & Punk         332
 2                Blues          81
 3           Bossa Nova          15
 4            Classical          74
 5               Comedy          17
 6                Drama          64
 7       Easy Listening          24
 8    Electronica/Dance          30
 9          Heavy Metal          28
 10         Hip Hop/Rap          35
 11                Jazz         130
 12               Latin         579
 13               Metal         374
 14               Opera           1
 15                 Pop          48
 16           R&B/Soul          61
 17              Reggae          58
 18                Rock        1297
 19       Rock And Roll          12
 20    Sci Fi & Fantasy          26
 21     Science Fiction          13
 22          Soundtrack          43
 23            TV Shows          93
 24               World          28,
 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(['Alternative', 'Alternative & Punk', 'Blues', 'Bossa Nova', 'Classical',
                           'Comedy', 'Drama', 'Easy Listening', 'Electronica/Dance', 'Heavy Metal',
                           'Hip Hop/Rap', 'Jazz', 'Latin', 'Metal', 'Opera', 'Pop', 'R&B/Soul',
```

```
                            'Reggae', 'Rock', 'Rock And Roll', 'Sci Fi & Fantasy',
                            'Science Fiction', 'Soundtrack', 'TV Shows', 'World'], dtype=object),
                'xaxis': 'x',
                'y': array([  40,  332,   81,   15,   74,   17,   64,   24,   30,   28,   35,  130,
                             579,  374,    1,   48,   61,   58, 1297,   12,   26,   13,   43,   93,
                              28]),
                'yaxis': 'y'}],
     'layout': {'barmode': 'relative',
                'legend': {'tracegroupgap': 0},
                'template': '...',
                'title': {'text': 'Number of Tracks in Each Genre'},
                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'Name'}},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'TrackCount'}}}
 }))
```

In [32]:
```
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 format 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 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_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 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\nCREATE INDEX IFK_AlbumArtistId ON Album (ArtistId)\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 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 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 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   List all genres and the number of tracks in each genre:\n'}, {'role': 'assistant', 'content': 'SELECT Genre.Name, COUNT(*) AS TrackCount\nFROM Track\nJOIN Genre ON Track.GenreId = Genre.GenreId\nGROUP BY Genre.Name;'}, {'role': 'user', 'content': ' \n   List all albums and their corresponding artist names \n'}, {'role': 'assistant', 'content': 'SELECT Album.Title, Artist.Name\nFROM Album\nJOIN Artist ON Album.ArtistId = Artist.ArtistId;'}, {'role': 'user', 'content': ' \n   Find all tracks with a name containing "What" (case-insensitive)\n'}, {'role': 'assistant', 'content': "SELECT *\nFROM Track\nWHERE Name LIKE '%What%';"}, {'role': 'user', 'content': ' \n   Find the top 5 most expensive tracks (based on unit price):\n'}, {'role': 'assistant', 'content': 'SELECT *\nFROM Track\nORDER BY UnitPrice DESC\nLIMIT 5;'}, {'role': 'user', 'content': ' \n   Find all invoices since 2010 and the total amount invoiced:\n'}, {'role': 'assistant', 'content': "SELECT InvoiceDate, SUM(Total) AS TotalAmountInvoiced\nFROM Invoice\nWHERE InvoiceDate >= '2010-01-01'\nGROUP BY InvoiceDate;"}, {'role': 'user', 'content': 'Show me a list of tables in the SQLite database'}, {'role': 'assistant', 'content': "SELECT name\nFROM sqlite_master\nWHERE 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   List all invoices with a total exceedin

```
g $10:\n'}, {'role': 'assistant', 'content': 'SELECT *\nFROM Invoice\nWHERE Total > 10;'}, {'role': 'user',
'content': "  \n    List all employees and their reporting manager's name (if any):\n"}, {'role': 'assistan
t', 'content': "SELECT e.FirstName || ' ' || e.LastName AS EmployeeName, \n       m.FirstName || ' ' || m.L
astName AS ManagerName\nFROM Employee e\nLEFT JOIN Employee m ON e.ReportsTo = m.EmployeeId;"}, {'role': 'u
ser', 'content': '  \n    Find the total number of invoices per country:\n'}, {'role': 'assistant', 'conten
t': 'SELECT BillingCountry, COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY BillingCountry;'}, {'role':
'user', 'content': '  \n    Get all genres that do not have any tracks associated with them:\n'}]
Using model gpt-3.5-turbo for 1078.0 tokens (approx)
```sql
SELECT Genre.Name
FROM Genre
LEFT JOIN Track ON Genre.GenreId = Track.GenreId
WHERE Track.GenreId IS NULL;
```
SELECT Genre.Name
FROM Genre
LEFT JOIN Track ON Genre.GenreId = Track.GenreId
WHERE Track.GenreId IS NULL;
SELECT Genre.Name
FROM Genre
LEFT JOIN Track ON Genre.GenreId = Track.GenreId
WHERE Track.GenreId IS NULL;
Empty DataFrame
Columns: [Name]
Index: []
Using model gpt-3.5-turbo for 188.25 tokens (approx)
```


Genres with no tracks
4
3
2
1
0
−1
count
−1
0
1
2
3
4
5
6
Name

Out[32]:
```
('SELECT Genre.Name\nFROM Genre\nLEFT JOIN Track ON Genre.GenreId = Track.GenreId\nWHERE Track.GenreId IS
NULL;',
 Empty DataFrame
 Columns: [Name]
 Index: [],
 Figure({
     'data': [{'alignmentgroup': 'True',
               'hovertemplate': 'Name=%{x}<br>count=%{y}<extra></extra>',
               'legendgroup': '',
               'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
               'name': '',
               'offsetgroup': '',
               'orientation': 'v',
               'showlegend': False,
               'textposition': 'auto',
               'type': 'bar',
               'x': array([], dtype=object),
               'xaxis': 'x',
               'y': array([], dtype=int64),
               'yaxis': 'y'}],
     'layout': {'barmode': 'relative',
                'legend': {'tracegroupgap': 0},
                'template': '...',
                'title': {'text': 'Genres with no tracks'},
                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'Name'}},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'count'}}}
 }))
```

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 format 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 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 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 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 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 INDEX IFK_CustomerSupportRepId ON Customer (SupportRepId)\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 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 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\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 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 TABLE Playlist\n(\n   PlaylistId 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 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 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': 'what are the top 5 countries th
at customers come from?'}, {'role': 'assistant', 'content': 'SELECT Country, COUNT(*) AS CustomerCount\nFRO
M Customer\nGROUP BY Country\nORDER BY CustomerCount DESC\nLIMIT 5;'}, {'role': 'user', 'content': 'How man
y customers are there'}, {'role': 'assistant', 'content': 'SELECT COUNT(*)\nFROM Customer;'}, {'role': 'use
r', 'content': '  \n    Get the total number of invoices for each customer\n'}, {'role': 'assistant', 'cont
ent': 'SELECT CustomerId, COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY CustomerId;'}, {'role': 'user',
'content': 'How many records are in table called customer'}, {'role': 'assistant', 'content': 'SELECT COUNT
(*)\nFROM Customer;'}, {'role': 'user', 'content': '  \n    Get the average invoice total for each custome
r:\n'}, {'role': 'assistant', 'content': 'SELECT CustomerId, AVG(Total) AS AvgInvoiceTotal\nFROM Invoice\nG
ROUP BY CustomerId;'}, {'role': 'user', 'content': '  \n    List all invoices with a total exceeding $1
0:\n'}, {'role': 'assistant', 'content': 'SELECT *\nFROM Invoice\nWHERE Total > 10;'}, {'role': 'user', 'co
ntent': '  \n    Find the total number of invoices per country:\n'}, {'role': 'assistant', 'content': 'SELE
CT BillingCountry, COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY BillingCountry;'}, {'role': 'user', 'c
ontent': '  \n    Find all invoices since 2010 and the total amount invoiced:\n'}, {'role': 'assistant', 'c
ontent': "SELECT InvoiceDate, SUM(Total) AS TotalAmountInvoiced\nFROM Invoice\nWHERE InvoiceDate >= '2010-0
1-01'\nGROUP BY InvoiceDate;"}, {'role': 'user', 'content': "  \n    List all employees and their reporting
manager's name (if any):\n"}, {'role': 'assistant', 'content': "SELECT e.FirstName || ' ' || e.LastName AS
EmployeeName, \n       m.FirstName || ' ' || m.LastName AS ManagerName\nFROM Employee e\nLEFT JOIN Employee
m ON e.ReportsTo = m.EmployeeId;"}, {'role': 'user', 'content': '  \n    List all albums and their correspo
nding artist names  \n'}, {'role': 'assistant', 'content': 'SELECT Album.Title, Artist.Name\nFROM Album\nJO
IN Artist ON Album.ArtistId = Artist.ArtistId;'}, {'role': 'user', 'content': '  \n    List all customers w
ho have not placed any orders:\n'}]
Using model gpt-3.5-turbo for 1502.5 tokens (approx)
SELECT Customer.*
FROM Customer
LEFT JOIN Invoice ON Customer.CustomerId = Invoice.CustomerId
WHERE Invoice.CustomerId IS NULL;
SELECT Customer.*
FROM Customer
LEFT JOIN Invoice ON Customer.CustomerId = Invoice.CustomerId
WHERE Invoice.CustomerId IS NULL;
SELECT Customer.*
FROM Customer
LEFT JOIN Invoice ON Customer.CustomerId = Invoice.CustomerId
WHERE Invoice.CustomerId IS NULL;
Empty DataFrame
Columns: [CustomerId, FirstName, LastName, Company, Address, City, State, Country, PostalCode, Phone, Fax,
Email, SupportRepId]
Index: []
Using model gpt-3.5-turbo for 261.0 tokens (approx)
```


Customers with No Orders
4
3
2
1
0
−1
count
−1
0
1
2
3
4
5
6
CustomerId

Out[33]:
```
('SELECT Customer.*\nFROM Customer\nLEFT JOIN Invoice ON Customer.CustomerId = Invoice.CustomerId\nWHERE I
nvoice.CustomerId IS NULL;',
 Empty DataFrame
 Columns: [CustomerId, FirstName, LastName, Company, Address, City, State, Country, PostalCode, Phone, Fa
x, Email, SupportRepId]
 Index: [],
 Figure({
     'data': [{'alignmentgroup': 'True',
               'hovertemplate': 'CustomerId=%{x}<br>count=%{y}<extra></extra>',
               'legendgroup': '',
               'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
               'name': '',
               'offsetgroup': '',
               'orientation': 'v',
               'showlegend': False,
               'textposition': 'auto',
               'type': 'bar',
               'x': array([], dtype=object),
               'xaxis': 'x',
               'y': array([], dtype=int64),
               'yaxis': 'y'}],
     'layout': {'barmode': 'relative',
                'legend': {'tracegroupgap': 0},
                'template': '...',
                'title': {'text': 'Customers with No Orders'},
                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'CustomerId'}},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'count'}}}
 }))
```

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

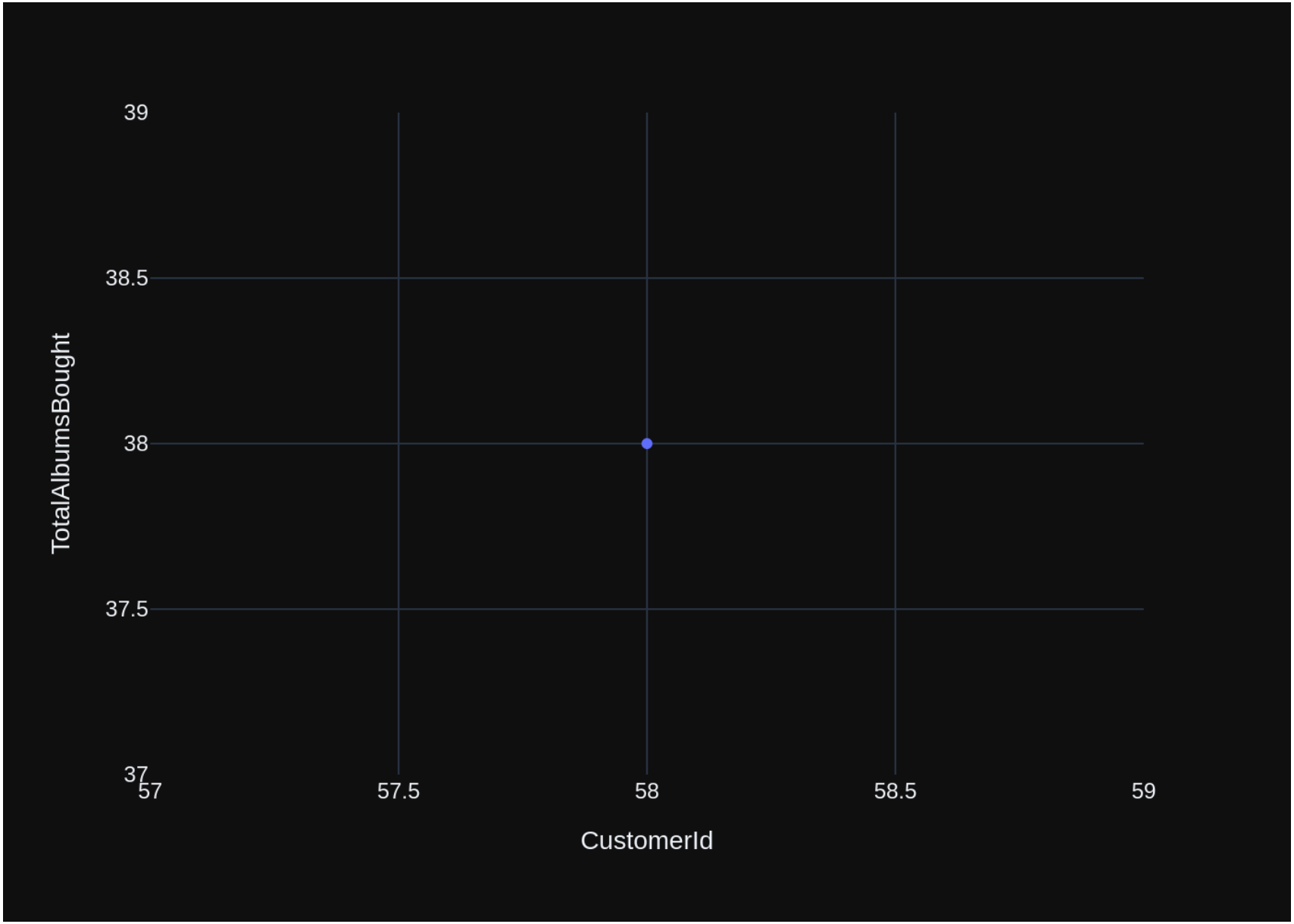
vn.ask(question=question)
```

```
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 format instructions. \n===Tables \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_AlbumArtistId ON Album (ArtistId)\n\nCREATE INDEX IFK_TrackAlbumId ON Track (AlbumId)\n\nCREATE TABLE Artist\n(\n   ArtistId INTEGER NOT NULL,\n   Name NVARCHAR(120),\n   CONSTRAINT PK_Artist PRIMARY KEY (ArtistId)\n)\n\nCREATE INDEX IFK_TrackGenreId ON Track (GenreId)\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\nCREATE INDEX IFK_PlaylistTrackTrackId ON PlaylistTrack (TrackId)\n\nCREATE INDEX IFK_TrackMediaTypeId ON Track (MediaTypeId)\n\nCREATE TABLE Playlist\n(\n   PlaylistId 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 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 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   Find the top 5 most expensive tracks (based on unit price):\n'}, {'role': 'assistant', 'content': 'SELECT *\nFROM Track\nORDER BY UnitPrice DESC\nLIMIT 5;'}, {'role': 'user', 'content': ' \n   List all genres and the number of tracks in each genre:\n'}, {'role': 'assistant', 'content': 'SELECT Genre.Name, COUNT(*) AS TrackCount\nFROM Track\nJOIN Genre ON Track.GenreId = Genre.GenreId\nGROUP BY Genre.Name;'}, {'role': 'user', 'content': ' \n   List all albums and their corresponding artist names\n\n'}, {'role': 'assistant', 'content': 'SELECT Album.Title, Artist.Name\nFROM Album\nJOIN Artist ON Album.ArtistId = Artist.ArtistId;'}, {'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   Find all tracks with a name containing "What" (case-insensitive)\n'}, {'role': 'assistant', 'content': "SELECT *\nFROM Track\nWHERE Name LIKE '%What%';"}, {'role': 'user', 'content': ' \n   List all invoices with a total exceeding $10:\n'}, {'role': 'assistant', 'content': 'SELECT *\nFROM Invoice\nWHERE Total > 10;'}, {'role': 'user', 'content': ' \n   Get the average invoice total for each customer:\n'}, {'role': 'assistant', 'content': 'SELECT CustomerId, AVG(Total) AS AvgInvoiceTotal\nFROM Invoice\nGROUP BY CustomerId;'}, {'role': 'user', 'content': 'How many customers are there'}, {'role': 'assistant', 'content': 'SELECT COUNT(*)\nFROM Custome

```
r;'}, {'role': 'user', 'content': '  \n    Get the total number of invoices for each customer\n'}, {'role':
'assistant', 'content': 'SELECT CustomerId, COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY CustomerI
d;'}, {'role': 'user', 'content': '  \n    Find the total number of invoices per country:\n'}, {'role': 'as
sistant', 'content': 'SELECT BillingCountry, COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY BillingCount
ry;'}, {'role': 'user', 'content': '  \n    Get the top 10 most popular artists (based on the number of tra
cks):\n'}]
Using model gpt-3.5-turbo for 1030.25 tokens (approx)
SELECT Artist.Name, COUNT(*) AS TrackCount
FROM Artist
JOIN Album ON Artist.ArtistId = Album.ArtistId
JOIN Track ON Album.AlbumId = Track.AlbumId
GROUP BY Artist.Name
ORDER BY TrackCount DESC
LIMIT 10;
SELECT Artist.Name, COUNT(*) AS TrackCount
FROM Artist
JOIN Album ON Artist.ArtistId = Album.ArtistId
JOIN Track ON Album.AlbumId = Track.AlbumId
GROUP BY Artist.Name
ORDER BY TrackCount DESC
LIMIT 10;
SELECT Artist.Name, COUNT(*) AS TrackCount
FROM Artist
JOIN Album ON Artist.ArtistId = Album.ArtistId
JOIN Track ON Album.AlbumId = Track.AlbumId
GROUP BY Artist.Name
ORDER BY TrackCount DESC
LIMIT 10;
              Name  TrackCount
0      Iron Maiden         213
1               U2         135
2     Led Zeppelin         114
3        Metallica         112
4             Lost          92
5      Deep Purple          92
6        Pearl Jam          67
7    Lenny Kravitz          57
8  Various Artists          56
9       The Office          53
Using model gpt-3.5-turbo for 219.75 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 Artist.Name, COUNT(*) AS TrackCount\nFROM Artist\nJOIN Album ON Artist.ArtistId = Album.ArtistId
\nJOIN Track ON Album.AlbumId = Track.AlbumId\nGROUP BY Artist.Name\nORDER BY TrackCount DESC\nLIMIT 10;',
               Name  TrackCount
 0      Iron Maiden         213
 1               U2         135
 2     Led Zeppelin         114
 3        Metallica         112
 4             Lost          92
 5      Deep Purple          92
 6        Pearl Jam          67
 7    Lenny Kravitz          57
 8  Various Artists          56
 9       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], 'title': {'text': 'Name'}},
                '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 format 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 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 INDEX IFK_CustomerSupportRepId ON Customer (SupportRepId)\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 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 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 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 INDEX IFK_InvoiceLineInvoiceId ON InvoiceLine (InvoiceId)\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 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 (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 \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': '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 customers are there'}, {'role': 'assistant', 'content': 'SELECT COUNT(*)\nFROM Customer;'}, {'role': 'user', 'content': ' \n   Get the total number of invoices for each customer\n'}, {'role': 'assistant', 'content': 'SELECT CustomerId, COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY CustomerId;'}, {'role': 'user', 'content': ' \n   Find the total number

```
of invoices per country:\n'}, {'role': 'assistant', 'content': 'SELECT BillingCountry, COUNT(*) AS TotalInv
oices\nFROM Invoice\nGROUP BY BillingCountry;'}, {'role': 'user', 'content': 'How many records are in table
called customer'}, {'role': 'assistant', 'content': 'SELECT COUNT(*)\nFROM Customer;'}, {'role': 'user', 'c
ontent': "  \n    List all employees and their reporting manager's name (if any):\n"}, {'role': 'assistan
t', 'content': "SELECT e.FirstName || ' ' || e.LastName AS EmployeeName, \n       m.FirstName || ' ' || m.L
astName AS ManagerName\nFROM Employee e\nLEFT JOIN Employee m ON e.ReportsTo = m.EmployeeId;"}, {'role': 'u
ser', 'content': '  \n    Get the average invoice total for each customer:\n'}, {'role': 'assistant', 'cont
ent': 'SELECT CustomerId, AVG(Total) AS AvgInvoiceTotal\nFROM Invoice\nGROUP BY CustomerId;'}, {'role': 'us
er', 'content': '  \n    List all invoices with a total exceeding $10:\n'}, {'role': 'assistant', 'conten
t': 'SELECT *\nFROM Invoice\nWHERE Total > 10;'}, {'role': 'user', 'content': '  \n    Find all invoices si
nce 2010 and the total amount invoiced:\n'}, {'role': 'assistant', 'content': "SELECT InvoiceDate, SUM(Tota
l) AS TotalAmountInvoiced\nFROM Invoice\nWHERE InvoiceDate >= '2010-01-01'\nGROUP BY InvoiceDate;"}, {'rol
e': 'user', 'content': '  \n    Get the top 10 most popular artists (based on the number of tracks):\n'},
{'role': 'assistant', 'content': 'SELECT Artist.Name, COUNT(*) AS TrackCount\nFROM Artist\nJOIN Album ON Ar
tist.ArtistId = Album.ArtistId\nJOIN Track ON Album.AlbumId = Track.AlbumId\nGROUP BY Artist.Name\nORDER BY
TrackCount DESC\nLIMIT 10;'}, {'role': 'user', 'content': '  \n     List all customers from Canada and thei
r email addresses:\n'}]
Using model gpt-3.5-turbo for 1321.25 tokens (approx)
SELECT FirstName, LastName, Email
FROM Customer
WHERE Country = 'Canada';
SELECT FirstName, LastName, Email
FROM Customer
WHERE Country = 'Canada';
SELECT FirstName, LastName, Email
FROM Customer
WHERE Country = 'Canada';
  FirstName  LastName                    Email
0  François  Tremblay     ftremblay@gmail.com
1      Mark   Philips      mphilips12@shaw.ca
2  Jennifer  Peterson     jenniferp@rogers.ca
3    Robert     Brown        robbrown@shaw.ca
4    Edward   Francis     edfrancis@yachoo.ca
5    Martha      Silk    marthasilk@gmail.com
6     Aaron  Mitchell  aaronmitchell@yahoo.ca
7     Ellie  Sullivan  ellie.sullivan@shaw.ca
Using model gpt-3.5-turbo for 189.75 tokens (approx)
```


Customers from Canada and their Email Addresses
variable
FirstName
LastName
count
1
0.8
0.6
0.4
0.2
0
François
Mark
Jennifer
Robert
Edward
Martha
Aaron
Ellie
Tremblay
Philips
Peterson
Brown
Francis
Silk
Mitchell
Sullivan
value

```
Out[35]: ("SELECT FirstName, LastName, Email\nFROM Customer\nWHERE Country = 'Canada';",
            FirstName  LastName                  Email
          0  François  Tremblay     ftremblay@gmail.com
          1      Mark   Philips      mphilips12@shaw.ca
          2  Jennifer  Peterson     jenniferp@rogers.ca
          3    Robert     Brown        robbrown@shaw.ca
          4    Edward   Francis     edfrancis@yachoo.ca
          5    Martha      Silk    marthasilk@gmail.com
          6     Aaron  Mitchell  aaronmitchell@yahoo.ca
          7     Ellie  Sullivan  ellie.sullivan@shaw.ca,
          Figure({
              'data': [{'alignmentgroup': 'True',
                        'hovertemplate': 'variable=FirstName<br>value=%{x}<br>count=%{y}<extra></extra>',
                        'legendgroup': 'FirstName',
                        'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
                        'name': 'FirstName',
                        'offsetgroup': 'FirstName',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['François', 'Mark', 'Jennifer', 'Robert', 'Edward', 'Martha', 'Aaron',
                                    'Ellie'], dtype=object),
                        'xaxis': 'x',
                        'y': array([1, 1, 1, 1, 1, 1, 1, 1]),
                        'yaxis': 'y'},
                       {'alignmentgroup': 'True',
                        'hovertemplate': 'variable=LastName<br>value=%{x}<br>count=%{y}<extra></extra>',
                        'legendgroup': 'LastName',
                        'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
                        'name': 'LastName',
                        'offsetgroup': 'LastName',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Tremblay', 'Philips', 'Peterson', 'Brown', 'Francis', 'Silk',
                                    'Mitchell', 'Sullivan'], dtype=object),
                        'xaxis': 'x',
                        'y': array([1, 1, 1, 1, 1, 1, 1, 1]),
                        'yaxis': 'y'}],
              'layout': {'barmode': 'relative',
```

```
                    'legend': {'title': {'text': 'variable'}, 'tracegroupgap': 0},
                    'template': '...',
                    'title': {'text': 'Customers from Canada and their Email Addresses'},
                    'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'value'}},
                    'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'count'}}}
})
```

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 format 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(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 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 INDEX IFK_CustomerSupportRepId ON Customer (SupportRepId)\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 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 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\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 CustomerId, COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY CustomerId;'}, {'role': 'user', 'content': ' \n   List all invoices with a total exceeding $10:\n'}, {'role': 'assistant', 'content': 'SELECT *\nFROM Invoice

```
\nWHERE Total > 10;'}, {'role': 'user', 'content': '  \n    Find the total number of invoices per countr
y:\n'}, {'role': 'assistant', 'content': 'SELECT BillingCountry, COUNT(*) AS TotalInvoices\nFROM Invoice\nG
ROUP BY BillingCountry;'}, {'role': 'user', 'content': '  \n    Get the average invoice total for each cust
omer:\n'}, {'role': 'assistant', 'content': 'SELECT CustomerId, AVG(Total) AS AvgInvoiceTotal\nFROM Invoice
\nGROUP BY CustomerId;'}, {'role': 'user', 'content': '  \n    Find all invoices since 2010 and the total a
mount invoiced:\n'}, {'role': 'assistant', 'content': "SELECT InvoiceDate, SUM(Total) AS TotalAmountInvoice
d\nFROM Invoice\nWHERE InvoiceDate >= '2010-01-01'\nGROUP BY InvoiceDate;"}, {'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': '  \n    Find the top 5 most expensive tracks (based on unit price):\n'}, {'role':
'assistant', 'content': 'SELECT *\nFROM Track\nORDER BY UnitPrice DESC\nLIMIT 5;'}, {'role': 'user', 'conte
nt': 'How many customers are there'}, {'role': 'assistant', 'content': 'SELECT COUNT(*)\nFROM Customer;'},
{'role': 'user', 'content': 'How many records are in table called customer'}, {'role': 'assistant', 'conten
t': 'SELECT COUNT(*)\nFROM Customer;'}, {'role': 'user', 'content': '  \n     List all customers from Canad
a and their email addresses:\n'}, {'role': 'assistant', 'content': "SELECT FirstName, LastName, Email\nFROM
Customer\nWHERE Country = 'Canada';"}, {'role': 'user', 'content': '  \n     Find the customer with the mos
t invoices \n'}]
Using model gpt-3.5-turbo for 1317.5 tokens (approx)
SELECT CustomerId, COUNT(*) AS TotalInvoices
FROM Invoice
GROUP BY CustomerId
ORDER BY TotalInvoices DESC
LIMIT 1;
SELECT CustomerId, COUNT(*) AS TotalInvoices
FROM Invoice
GROUP BY CustomerId
ORDER BY TotalInvoices DESC
LIMIT 1;
SELECT CustomerId, COUNT(*) AS TotalInvoices
FROM Invoice
GROUP BY CustomerId
ORDER BY TotalInvoices DESC
LIMIT 1;
   CustomerId  TotalInvoices
0           1              7
Using model gpt-3.5-turbo for 192.5 tokens (approx)
```


8
7.5
7
6.5
6
TotalInvoices
0
0.5
1
1.5
2
CustomerId

Out[36]:
```
('SELECT CustomerId, COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY CustomerId\nORDER BY TotalInvoices
DESC\nLIMIT 1;',
    CustomerId  TotalInvoices
 0           1              7,
 Figure({
     'data': [{'hovertemplate': 'CustomerId=%{x}<br>TotalInvoices=%{y}<extra></extra>',
               'legendgroup': '',
               'marker': {'color': '#636efa', 'symbol': 'circle'},
               'mode': 'markers',
               'name': '',
               'orientation': 'v',
               'showlegend': False,
               'type': 'scatter',
               'x': array([1]),
               'xaxis': 'x',
               'y': array([7]),
               'yaxis': 'y'}],
     'layout': {'legend': {'tracegroupgap': 0},
                'margin': {'t': 60},
                'template': '...',
                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'CustomerId'}},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'TotalInvoices'}}}
 }))
```

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 format instructions. \n===Tables \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 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\nCREATE INDEX IFK_AlbumArtistId ON Album (ArtistId)\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_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(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_TrackAlbumId ON Track (AlbumId)\n\nCREATE INDEX IFK_InvoiceLineInvoiceId ON InvoiceLine (InvoiceId)\n\nCREATE INDEX IFK_InvoiceLineTrackId 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 (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 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   Find the customer with the most invoices \n'}, {'role': 'assistant', 'content': 'SELECT CustomerId, COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY CustomerId\nORDER BY TotalInvoices DESC\nLIMIT 1;'}, {'role': 'user', 'content': ' \n   Get the total number of invoices for each customer\n'}, {'role': 'assistant', 'content': 'SELECT CustomerId, COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY CustomerId;'}, {'role': 'user', 'content': ' \n   Get the top 10 most popular artists (based on the number of tracks):\n'}, {'role': 'assistant', 'content': 'SELECT Artist.Name, COUNT(*) AS TrackCount\nFROM Artist\nJOIN Album ON Artist.ArtistId = Album.ArtistId\nJOIN Track ON Album.AlbumId = Track.AlbumId\nGROUP BY Artist.Name\nORDER BY TrackCount DESC\nLIMIT 10;'}, {'role': 'user', 'content': ' \n   Find all invoices since 2010 and the total amount invoice

```
d:\n'}, {'role': 'assistant', 'content': "SELECT InvoiceDate, SUM(Total) AS TotalAmountInvoiced\nFROM Invoi
ce\nWHERE InvoiceDate >= '2010-01-01'\nGROUP BY InvoiceDate;"}, {'role': 'user', 'content': '  \n    List a
ll invoices with a total exceeding $10:\n'}, {'role': 'assistant', 'content': 'SELECT *\nFROM Invoice\nWHER
E Total > 10;'}, {'role': 'user', 'content': '  \n    Find the top 5 most expensive tracks (based on unit p
rice):\n'}, {'role': 'assistant', 'content': 'SELECT *\nFROM Track\nORDER BY UnitPrice DESC\nLIMIT 5;'},
{'role': 'user', 'content': '  \n    Get the average invoice total for each customer:\n'}, {'role': 'assist
ant', 'content': 'SELECT CustomerId, AVG(Total) AS AvgInvoiceTotal\nFROM Invoice\nGROUP BY CustomerId;'},
{'role': 'user', 'content': '  \n    Find the total number of invoices per country:\n'}, {'role': 'assistan
t', 'content': 'SELECT BillingCountry, COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY BillingCountry;'},
{'role': 'user', 'content': '  \n    List all albums and their corresponding artist names  \n'}, {'role':
'assistant', 'content': 'SELECT Album.Title, Artist.Name\nFROM Album\nJOIN Artist ON Album.ArtistId = Artis
t.ArtistId;'}, {'role': 'user', 'content': '  \n    List all genres and the number of tracks in each genr
e:\n'}, {'role': 'assistant', 'content': 'SELECT Genre.Name, COUNT(*) AS TrackCount\nFROM Track\nJOIN Genre
ON Track.GenreId = Genre.GenreId\nGROUP BY Genre.Name;'}, {'role': 'user', 'content': '  \n     Find the cu
stomer who bought the most albums in total quantity (across all invoices): \n'}]
Using model gpt-3.5-turbo for 1280.75 tokens (approx)
Intermediate SQL
```sql
-- Find the total quantity of tracks bought per customer
SELECT Invoice.CustomerId, SUM(InvoiceLine.Quantity) AS TotalQuantity
FROM Invoice
JOIN InvoiceLine ON Invoice.InvoiceId = InvoiceLine.InvoiceId
GROUP BY Invoice.CustomerId;
```

Final SQL
```sql
-- Find the customer who bought the most albums in total quantity
SELECT CustomerId, SUM(TotalQuantity) AS TotalAlbumsBought
FROM (
    SELECT Invoice.CustomerId, SUM(InvoiceLine.Quantity) AS TotalQuantity
    FROM Invoice
    JOIN InvoiceLine ON Invoice.InvoiceId = InvoiceLine.InvoiceId
    GROUP BY Invoice.CustomerId
)
GROUP BY CustomerId
ORDER BY TotalAlbumsBought DESC
LIMIT 1;
```
SELECT CustomerId, SUM(TotalQuantity) AS TotalAlbumsBought
FROM (
    SELECT Invoice.CustomerId, SUM(InvoiceLine.Quantity) AS TotalQuantity
```

```
    FROM Invoice
    JOIN InvoiceLine ON Invoice.InvoiceId = InvoiceLine.InvoiceId
    GROUP BY Invoice.CustomerId
)
GROUP BY CustomerId
ORDER BY TotalAlbumsBought DESC
LIMIT 1;
SELECT CustomerId, SUM(TotalQuantity) AS TotalAlbumsBought
FROM (
    SELECT Invoice.CustomerId, SUM(InvoiceLine.Quantity) AS TotalQuantity
    FROM Invoice
    JOIN InvoiceLine ON Invoice.InvoiceId = InvoiceLine.InvoiceId
    GROUP BY Invoice.CustomerId
)
GROUP BY CustomerId
ORDER BY TotalAlbumsBought DESC
LIMIT 1;
   CustomerId  TotalAlbumsBought
0          58                 38
Using model gpt-3.5-turbo for 256.5 tokens (approx)
```


39
38.5
38
37.5
37
TotalAlbumsBought
57
57.5
58
58.5
59
CustomerId

Out[37]:
```
('SELECT CustomerId, SUM(TotalQuantity) AS TotalAlbumsBought\nFROM (\n    SELECT Invoice.CustomerId, SUM(I
nvoiceLine.Quantity) AS TotalQuantity\n    FROM Invoice\n    JOIN InvoiceLine ON Invoice.InvoiceId = Invoi
ceLine.InvoiceId\n    GROUP BY Invoice.CustomerId\n)\nGROUP BY CustomerId\nORDER BY TotalAlbumsBought DESC
\nLIMIT 1;',
    CustomerId  TotalAlbumsBought
 0          58                38,
 Figure({
     'data': [{'hovertemplate': 'CustomerId=%{x}<br>TotalAlbumsBought=%{y}<extra></extra>',
               'legendgroup': '',
               'marker': {'color': '#636efa', 'symbol': 'circle'},
               'mode': 'markers',
               'name': '',
               'orientation': 'v',
               'showlegend': False,
               'type': 'scatter',
               'x': array([58]),
               'xaxis': 'x',
               'y': array([38]),
               'yaxis': 'y'}],
     'layout': {'legend': {'tracegroupgap': 0},
                'margin': {'t': 60},
                'template': '...',
                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'CustomerId'}},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'TotalAlbumsBought'}}}
 }))
```

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

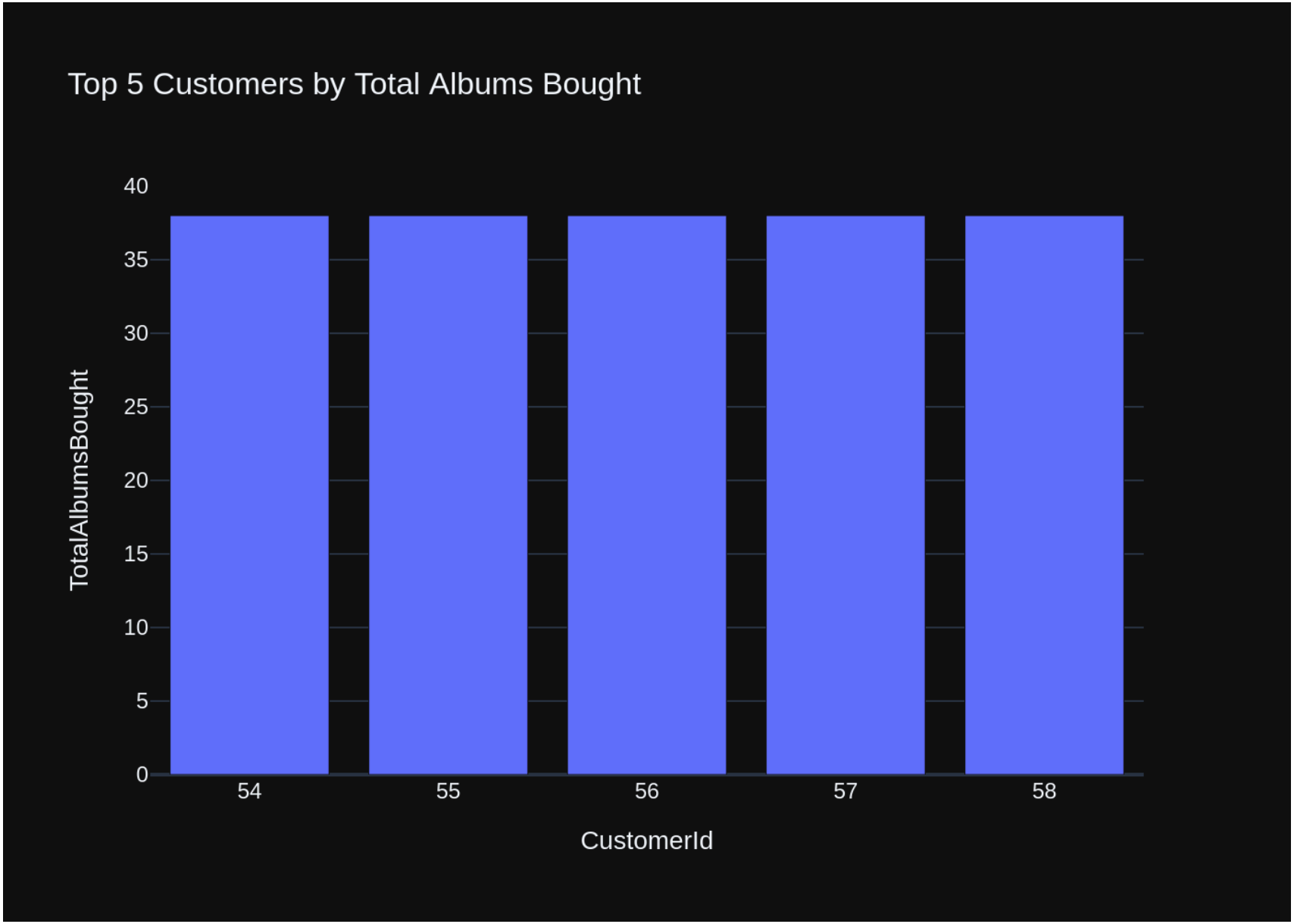
vn.ask(question=question)
```

```
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 format instructions. \n===Tables \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 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\nCREATE INDEX IFK_AlbumArtistId ON Album (ArtistId)\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 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_InvoiceCustomerId 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 (InvoiceId)\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 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   Find the customer who bought the most albums in total quantity (across all invoices): \n'}, {'role': 'assistant', 'content': 'SELECT CustomerId, SUM(TotalQuantity) AS TotalAlbumsBought\nFROM (\n   SELECT Invoice.CustomerId, SUM(InvoiceLine.Quantity) AS TotalQuantity\n   FROM Invoice\n   JOIN InvoiceLine ON Invoice.InvoiceId = InvoiceLine.InvoiceId\n   GROUP BY Invoice.CustomerId\n)\nGROUP BY CustomerId\nORDER BY TotalAlbumsBought DESC\nLIMIT 1;'}, {'role': 'user', 'content': ' \n   Get the top 10 most popular artists (based on the number of tracks):\n'}, {'role': 'assistant', 'content': 'SELECT Artist.Name, COUNT(*) AS TrackCount\nFROM Artist\nJOIN Album ON Artist.ArtistId = Album.ArtistId\nJOIN Track ON Album.AlbumId = Track.AlbumId\nGROUP BY Artist.Name\nORDER BY TrackCount DESC\nLIMIT 10;'}, {'role': 'user', 'content': ' \n   Find the customer with the most invoices \n'}, {'role': 'assistant', 'content': 'SELECT CustomerId, COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY CustomerId\nORDER BY TotalInvoices DESC\nLIMIT 1;'}, {'role': 'user', 'content': ' \n   Find the top 5 most expens

```
ive tracks (based on unit price):\n'}, {'role': 'assistant', 'content': 'SELECT *\nFROM Track\nORDER BY Uni
tPrice DESC\nLIMIT 5;'}, {'role': 'user', 'content': '  \n    List all invoices with a total exceeding $1
0:\n'}, {'role': 'assistant', 'content': 'SELECT *\nFROM Invoice\nWHERE Total > 10;'}, {'role': 'user', 'co
ntent': '  \n    Get the total number of invoices for each customer\n'}, {'role': 'assistant', 'content':
'SELECT CustomerId, COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY CustomerId;'}, {'role': 'user', 'cont
ent': '  \n    List all albums and their corresponding artist names  \n'}, {'role': 'assistant', 'content':
'SELECT Album.Title, Artist.Name\nFROM Album\nJOIN Artist ON Album.ArtistId = Artist.ArtistId;'}, {'role':
'user', 'content': '  \n    Get the average invoice total for each customer:\n'}, {'role': 'assistant', 'co
ntent': 'SELECT CustomerId, AVG(Total) AS AvgInvoiceTotal\nFROM Invoice\nGROUP BY CustomerId;'}, {'role':
'user', 'content': 'what are the top 5 countries that customers come from?'}, {'role': 'assistant', 'conten
t': 'SELECT Country, COUNT(*) AS CustomerCount\nFROM Customer\nGROUP BY Country\nORDER BY CustomerCount DES
C\nLIMIT 5;'}, {'role': 'user', 'content': '  \n    Find all invoices since 2010 and the total amount invoi
ced:\n'}, {'role': 'assistant', 'content': "SELECT InvoiceDate, SUM(Total) AS TotalAmountInvoiced\nFROM Inv
oice\nWHERE InvoiceDate >= '2010-01-01'\nGROUP BY InvoiceDate;"}, {'role': 'user', 'content': '  \n     Fin
d the top 5 customer who bought the most albums in total quantity (across all invoices):\n'}]
Using model gpt-3.5-turbo for 1286.5 tokens (approx)
SELECT CustomerId, SUM(Quantity) AS TotalAlbumsBought
FROM (
    SELECT Invoice.CustomerId, SUM(InvoiceLine.Quantity) AS Quantity
    FROM Invoice
    JOIN InvoiceLine ON Invoice.InvoiceId = InvoiceLine.InvoiceId
    GROUP BY Invoice.CustomerId, InvoiceLine.TrackId
)
GROUP BY CustomerId
ORDER BY TotalAlbumsBought DESC
LIMIT 5;
SELECT CustomerId, SUM(Quantity) AS TotalAlbumsBought
FROM (
    SELECT Invoice.CustomerId, SUM(InvoiceLine.Quantity) AS Quantity
    FROM Invoice
    JOIN InvoiceLine ON Invoice.InvoiceId = InvoiceLine.InvoiceId
    GROUP BY Invoice.CustomerId, InvoiceLine.TrackId
)
GROUP BY CustomerId
ORDER BY TotalAlbumsBought DESC
LIMIT 5;
SELECT CustomerId, SUM(Quantity) AS TotalAlbumsBought
FROM (
    SELECT Invoice.CustomerId, SUM(InvoiceLine.Quantity) AS Quantity
    FROM Invoice
    JOIN InvoiceLine ON Invoice.InvoiceId = InvoiceLine.InvoiceId
    GROUP BY Invoice.CustomerId, InvoiceLine.TrackId
```

```
)
GROUP BY CustomerId
ORDER BY TotalAlbumsBought DESC
LIMIT 5;
   CustomerId  TotalAlbumsBought
0          58                 38
1          57                 38
2          56                 38
3          55                 38
4          54                 38
Using model gpt-3.5-turbo for 260.5 tokens (approx)
```


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

Out[38]:
```
('SELECT CustomerId, SUM(Quantity) AS TotalAlbumsBought\nFROM (\n    SELECT Invoice.CustomerId, SUM(Invoic
eLine.Quantity) AS Quantity\n    FROM Invoice\n    JOIN InvoiceLine ON Invoice.InvoiceId = InvoiceLine.Inv
oiceId\n    GROUP BY Invoice.CustomerId, InvoiceLine.TrackId\n)\nGROUP BY CustomerId\nORDER BY TotalAlbums
Bought DESC\nLIMIT 5;',
    CustomerId  TotalAlbumsBought
 0          58                38
 1          57                38
 2          56                38
 3          55                38
 4          54                38,
 Figure({
     'data': [{'alignmentgroup': 'True',
               'hovertemplate': 'CustomerId=%{x}<br>TotalAlbumsBought=%{y}<extra></extra>',
               'legendgroup': '',
               'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
               'name': '',
               'offsetgroup': '',
               'orientation': 'v',
               'showlegend': False,
               'textposition': 'auto',
               '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': 'CustomerId'}},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'TotalAlbumsBought'}}}
 }))
```

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

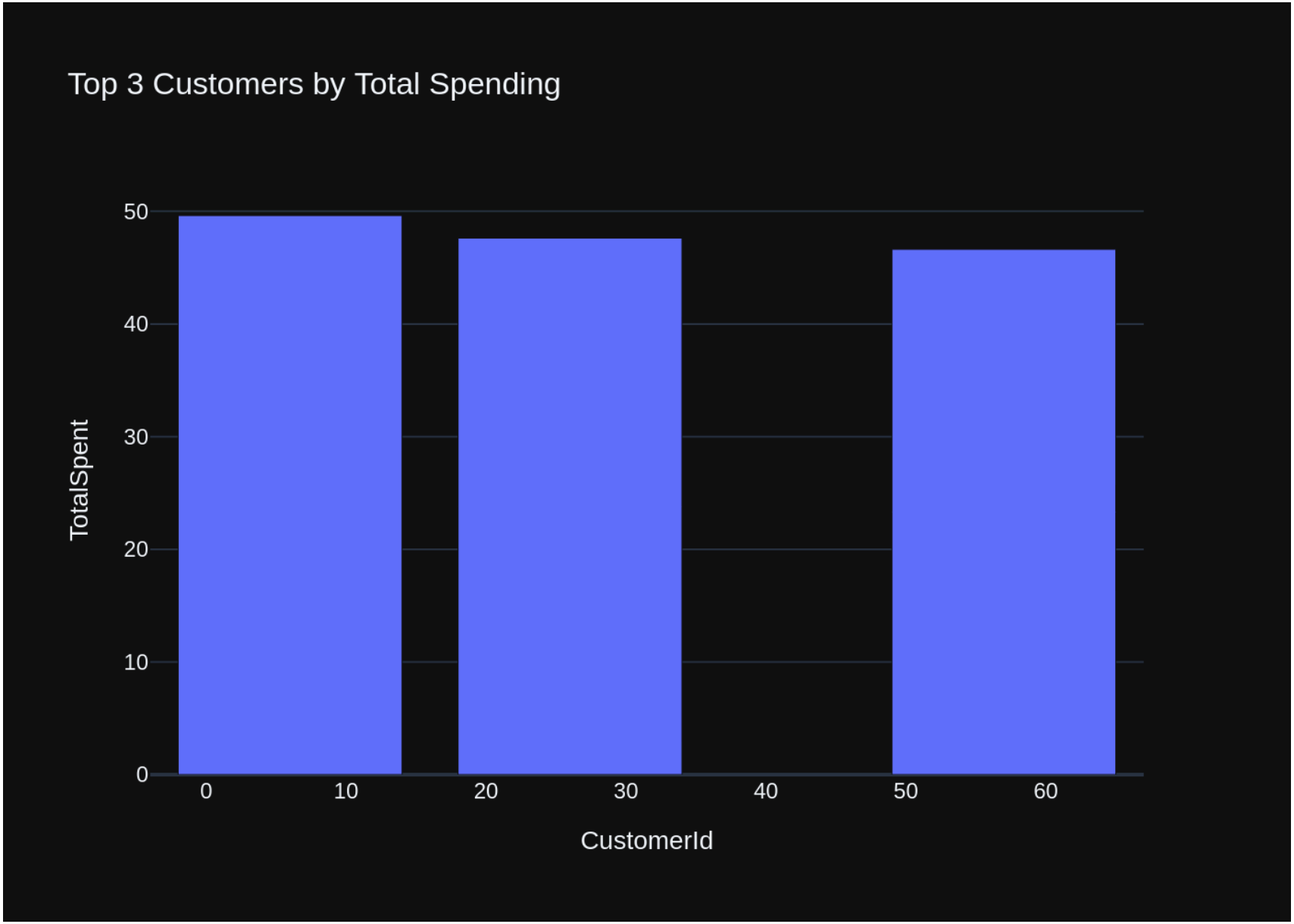
vn.ask(question=question)
```

```
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 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 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 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 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 (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_InvoiceCustomerId 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 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 (TrackId) \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 INDEX IFK_InvoiceLineTrackId 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 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   F

ind the top 5 customer who bought the most albums in total quantity (across all invoices):\n'}, {'role': 'a
ssistant', 'content': 'SELECT CustomerId, SUM(Quantity) AS TotalAlbumsBought\nFROM (\n   SELECT Invoice.Cu
stomerId, SUM(InvoiceLine.Quantity) AS Quantity\n   FROM Invoice\n   JOIN InvoiceLine ON Invoice.InvoiceI
d = InvoiceLine.InvoiceId\n   GROUP BY Invoice.CustomerId, InvoiceLine.TrackId\n)\nGROUP BY CustomerId\nOR
DER BY TotalAlbumsBought DESC\nLIMIT 5;'}, {'role': 'user', 'content': ' \n   Find the top 5 most expensi
ve tracks (based on unit price):\n'}, {'role': 'assistant', 'content': 'SELECT *\nFROM Track\nORDER BY Unit
Price 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 CustomerId, SUM(Tot
alQuantity) AS TotalAlbumsBought\nFROM (\n   SELECT Invoice.CustomerId, SUM(InvoiceLine.Quantity) AS Total
Quantity\n   FROM Invoice\n   JOIN InvoiceLine ON Invoice.InvoiceId = InvoiceLine.InvoiceId\n   GROUP BY
Invoice.CustomerId\n)\nGROUP BY CustomerId\nORDER BY TotalAlbumsBought DESC\nLIMIT 1;'}, {'role': 'user',
'content': ' \n   Find the customer with the most invoices \n'}, {'role': 'assistant', 'content': 'SELEC
T CustomerId, COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY CustomerId\nORDER BY TotalInvoices DESC\nLI
MIT 1;'}, {'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 average invoice total for e
ach customer:\n'}, {'role': 'assistant', 'content': 'SELECT CustomerId, AVG(Total) AS AvgInvoiceTotal\nFROM
Invoice\nGROUP BY CustomerId;'}, {'role': 'user', 'content': ' \n   Get the top 10 most popular artists
(based on the number of tracks):\n'}, {'role': 'assistant', 'content': 'SELECT Artist.Name, COUNT(*) AS Tra
ckCount\nFROM Artist\nJOIN Album ON Artist.ArtistId = Album.ArtistId\nJOIN Track ON Album.AlbumId = Track.A
lbumId\nGROUP BY Artist.Name\nORDER BY TrackCount DESC\nLIMIT 10;'}, {'role': 'user', 'content': ' \n   G
et the total number of invoices for each customer\n'}, {'role': 'assistant', 'content': 'SELECT CustomerId,
COUNT(*) AS TotalInvoices\nFROM Invoice\nGROUP BY CustomerId;'}, {'role': 'user', 'content': ' \n   List
all invoices with a total exceeding $10:\n'}, {'role': 'assistant', 'content': 'SELECT *\nFROM Invoice\nWHE
RE Total > 10;'}, {'role': 'user', 'content': ' \n   Find all invoices since 2010 and the total amount in
voiced:\n'}, {'role': 'assistant', 'content': "SELECT InvoiceDate, SUM(Total) AS TotalAmountInvoiced\nFROM
Invoice\nWHERE InvoiceDate >= '2010-01-01'\nGROUP BY InvoiceDate;"}, {'role': 'user', 'content': ' \n
Find the top 3 customers who spent the most money overall:\n'}]
Using model gpt-3.5-turbo for 1614.25 tokens (approx)
```
SELECT CustomerId, SUM(Total) AS TotalSpent
FROM Invoice
GROUP BY CustomerId
ORDER BY TotalSpent DESC
LIMIT 3;
SELECT CustomerId, SUM(Total) AS TotalSpent
FROM Invoice
GROUP BY CustomerId
ORDER BY TotalSpent DESC
LIMIT 3;
SELECT CustomerId, SUM(Total) AS TotalSpent
FROM Invoice
GROUP BY CustomerId
```

```
ORDER BY TotalSpent DESC
LIMIT 3;
   CustomerId  TotalSpent
0           6       49.62
1          26       47.62
2          57       46.62
Using model gpt-3.5-turbo for 195.25 tokens (approx)
```


Top 3 Customers by Total Spending
TotalSpent
50
40
30
20
10
0
0
10
20
30
40
50
60
CustomerId

Out[39]:
```
('SELECT CustomerId, SUM(Total) AS TotalSpent\nFROM Invoice\nGROUP BY CustomerId\nORDER BY TotalSpent DESC
\nLIMIT 3;',
    CustomerId  TotalSpent
 0           6       49.62
 1          26       47.62
 2          57       46.62,
 Figure({
     'data': [{'alignmentgroup': 'True',
               'hovertemplate': 'CustomerId=%{x}<br>TotalSpent=%{y}<extra></extra>',
               'legendgroup': '',
               'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
               'name': '',
               'offsetgroup': '',
               'orientation': 'v',
               'showlegend': False,
               'textposition': 'auto',
               'type': 'bar',
               'x': array([ 6, 26, 57]),
               'xaxis': 'x',
               'y': array([49.62, 47.62, 46.62]),
               'yaxis': 'y'}],
     'layout': {'barmode': 'relative',
                'legend': {'tracegroupgap': 0},
                'template': '...',
                'title': {'text': 'Top 3 Customers by Total Spending'},
                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'CustomerId'}},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'TotalSpent'}}}
 }))
```

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
```

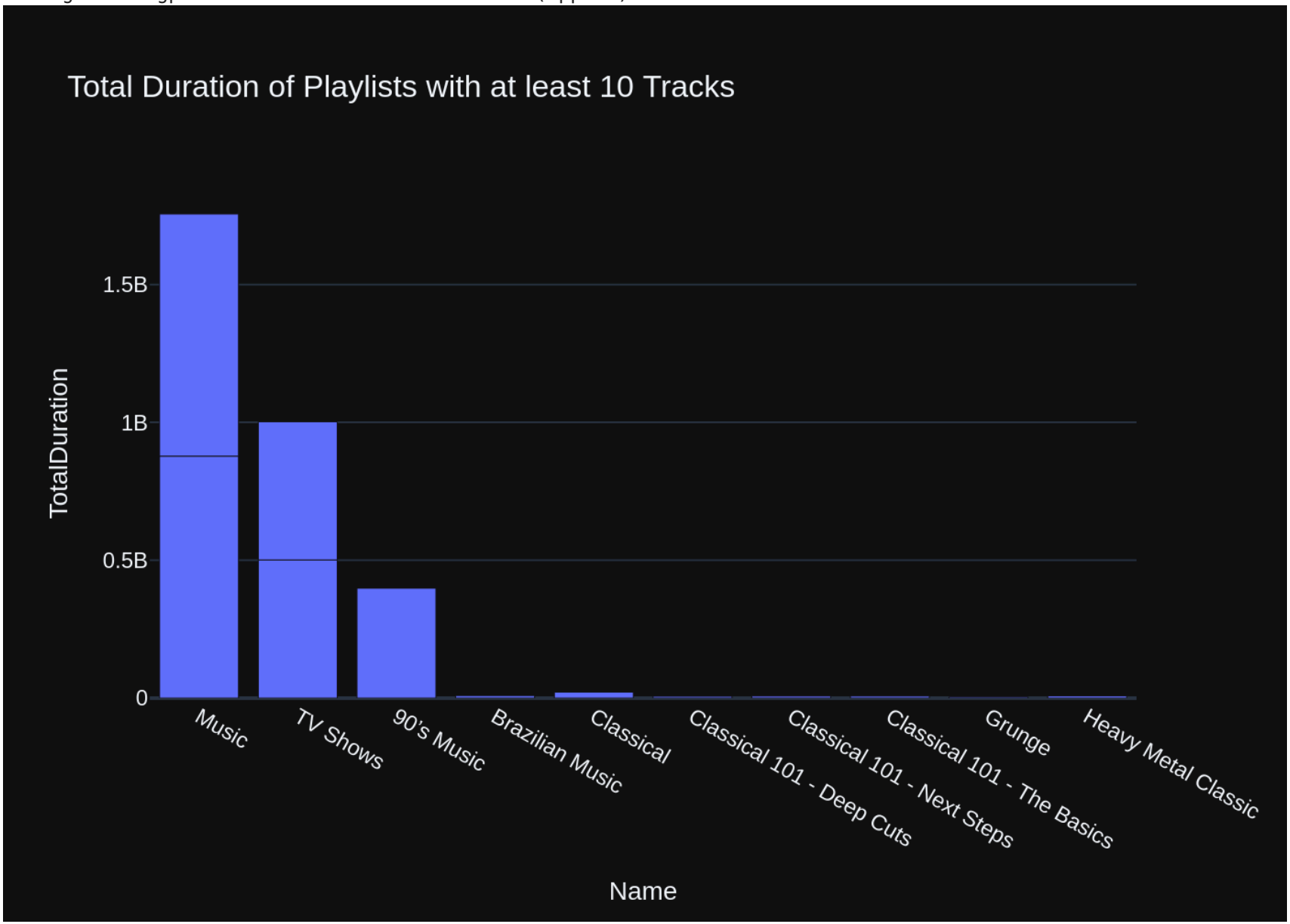
[{'role': 'system', 'content': "You are a SQLite expert. Please help to generate a SQL query to answer the question. Your response should ONLY be based on the given context and follow the response guidelines and format instructions. \n===Tables \nCREATE INDEX IFK_PlaylistTrackTrackId ON PlaylistTrack (TrackId)\n\nCREATE TABLE Playlist\n(\n   PlaylistId INTEGER NOT NULL,\n   Name NVARCHAR(120),\n   CONSTRAINT PK_Playlist PRIMARY KEY (PlaylistId)\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 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 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_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 (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\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceLineTrackId 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 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   List all genres and the number of tracks in each genre:\n'}, {'role': 'assistant', 'content': 'SELECT Genre.Name, COUNT(*) AS TrackCount\nFROM Track\nJOIN Genre ON Track.GenreId = Genre.GenreId\nGROUP BY Genre.Name;'}, {'role': 'user', 'content': ' \n   Get the top 10 most popular artists (based on the number of tracks):\n'}, {'role': 'assistant', 'content': 'SELECT Artist.Name, COUNT(*) AS TrackCount\nFROM Artist\nJOIN Album ON Artist.ArtistId = Album.ArtistId\nJOIN Track ON Album.AlbumId = Track.AlbumId\nGROUP BY Artist.Name\nORDER BY TrackCount DESC\nLIMIT 10;'}, {'role': 'user', 'content': ' \n   Find the top 5 customer who bought the most albums in total quantity (across all invoices):\n'}, {'role': 'assistant', 'content': 'SELECT CustomerId, SUM(Quantity) AS TotalAlbumsBought \nFROM (\n   SELECT Invoice.CustomerId, SUM(InvoiceLine.Quantity) AS Quantity\n   FROM Invoice\n   JOIN InvoiceLine ON Invoice.InvoiceId = InvoiceLine.InvoiceId\n   GROUP BY Invoice.CustomerId, InvoiceLine.TrackId\n)\nGROUP BY CustomerId\nORDER BY TotalAlbumsBought 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 CustomerId, SUM(TotalQuantity) AS TotalAlbumsBought\nFROM (\n   SELECT Invoice.CustomerId, SUM(InvoiceLine.Quantity) AS TotalQuantity\n   FROM Invoice\n   JOIN InvoiceLine ON Invoice.InvoiceId = InvoiceLine.InvoiceId\n   GROUP BY Invoice.CustomerId\n)\nGROUP BY CustomerId\nORDER BY TotalAlbumsBought DESC\nLIMIT 1;'}, {'role': 'user', 'content': ' \n   Find all tracks with a name containing "Wh

```
at" (case-insensitive)\n'}, {'role': 'assistant', 'content': "SELECT *\nFROM Track\nWHERE Name LIKE '%Wha
t%';"}, {'role': 'user', 'content': '  \n    List all albums and their corresponding artist names  \n'},
{'role': 'assistant', 'content': 'SELECT Album.Title, Artist.Name\nFROM Album\nJOIN Artist ON Album.ArtistI
d = Artist.ArtistId;'}, {'role': 'user', 'content': '  \n    Find the top 5 most expensive tracks (based on
unit price):\n'}, {'role': 'assistant', 'content': 'SELECT *\nFROM Track\nORDER BY UnitPrice DESC\nLIMIT
5;'}, {'role': 'user', 'content': '  \n    Find all invoices since 2010 and the total amount invoiced:\n'},
{'role': 'assistant', 'content': "SELECT InvoiceDate, SUM(Total) AS TotalAmountInvoiced\nFROM Invoice\nWHER
E InvoiceDate >= '2010-01-01'\nGROUP BY InvoiceDate;"}, {'role': 'user', 'content': '  \n     Find the top
3 customers who spent the most money overall:\n'}, {'role': 'assistant', 'content': 'SELECT CustomerId, SUM
(Total) AS TotalSpent\nFROM Invoice\nGROUP BY CustomerId\nORDER BY TotalSpent DESC\nLIMIT 3;'}, {'role': 'u
ser', 'content': '  \n    List all invoices with a total exceeding $10:\n'}, {'role': 'assistant', 'conten
t': 'SELECT *\nFROM Invoice\nWHERE Total > 10;'}, {'role': 'user', 'content': '  \n     Get all playlists c
ontaining at least 10 tracks and the total duration of those tracks:\n'}]
```


```
Using model gpt-3.5-turbo for 1230.25 tokens (approx)
SELECT Playlist.PlaylistId, Playlist.Name, SUM(Track.Milliseconds) AS TotalDuration
FROM Playlist
JOIN PlaylistTrack ON Playlist.PlaylistId = PlaylistTrack.PlaylistId
JOIN Track ON PlaylistTrack.TrackId = Track.TrackId
GROUP BY Playlist.PlaylistId, Playlist.Name
HAVING COUNT(Track.TrackId) >= 10;
SELECT Playlist.PlaylistId, Playlist.Name, SUM(Track.Milliseconds) AS TotalDuration
FROM Playlist
JOIN PlaylistTrack ON Playlist.PlaylistId = PlaylistTrack.PlaylistId
JOIN Track ON PlaylistTrack.TrackId = Track.TrackId
GROUP BY Playlist.PlaylistId, Playlist.Name
HAVING COUNT(Track.TrackId) >= 10;
SELECT Playlist.PlaylistId, Playlist.Name, SUM(Track.Milliseconds) AS TotalDuration
FROM Playlist
JOIN PlaylistTrack ON Playlist.PlaylistId = PlaylistTrack.PlaylistId
JOIN Track ON PlaylistTrack.TrackId = Track.TrackId
GROUP BY Playlist.PlaylistId, Playlist.Name
HAVING COUNT(Track.TrackId) >= 10;
```

	PlaylistId	Name	TotalDuration
0	1	Music	877683083
1	3	TV Shows	501094957
2	5	90’s Music	398705153
3	8	Music	877683083
4	10	TV Shows	501094957
5	11	Brazilian Music	9486559
6	12	Classical	21770592
7	13	Classical 101 - Deep Cuts	6755730
8	14	Classical 101 - Next Steps	7575051

```
9           15  Classical 101 - The Basics        7439811
10          16                      Grunge        4122018
11          17         Heavy Metal Classic        8206312
Using model gpt-3.5-turbo for 256.25 tokens (approx)
```



Out[40]:
```
('SELECT Playlist.PlaylistId, Playlist.Name, SUM(Track.Milliseconds) AS TotalDuration\nFROM Playlist\nJOIN
PlaylistTrack ON Playlist.PlaylistId = PlaylistTrack.PlaylistId\nJOIN Track ON PlaylistTrack.TrackId = Tra
ck.TrackId\nGROUP BY Playlist.PlaylistId, Playlist.Name\nHAVING COUNT(Track.TrackId) >= 10;',
     PlaylistId                        Name  TotalDuration
 0            1                       Music      877683083
 1            3                   TV Shows      501094957
 2            5                  90’s Music      398705153
 3            8                       Music      877683083
 4           10                   TV Shows      501094957
 5           11              Brazilian Music        9486559
 6           12                    Classical       21770592
 7           13   Classical 101 - Deep Cuts        6755730
 8           14  Classical 101 - Next Steps        7575051
 9           15  Classical 101 - The Basics        7439811
 10          16                      Grunge        4122018
 11          17         Heavy Metal Classic        8206312,
 Figure({
     'data': [{'alignmentgroup': 'True',
               'hovertemplate': 'Name=%{x}<br>TotalDuration=%{y}<extra></extra>',
               'legendgroup': '',
               'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
               'name': '',
               'offsetgroup': '',
               'orientation': 'v',
               'showlegend': False,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Music', 'TV Shows', '90’s Music', 'Music', 'TV Shows',
                           'Brazilian Music', 'Classical', 'Classical 101 - Deep Cuts',
                           'Classical 101 - Next Steps', 'Classical 101 - The Basics', 'Grunge',
                           'Heavy Metal Classic'], dtype=object),
               'xaxis': 'x',
               'y': array([877683083, 501094957, 398705153, 877683083, 501094957,   9486559,
                            21770592,   6755730,   7575051,   7439811,   4122018,   8206312]),
               'yaxis': 'y'}],
     'layout': {'barmode': 'relative',
                'legend': {'tracegroupgap': 0},
                'template': '...',
                'title': {'text': 'Total Duration of Playlists with at least 10 Tracks'},
                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'Name'}},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'TotalDuration'}}}
 }))
```

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

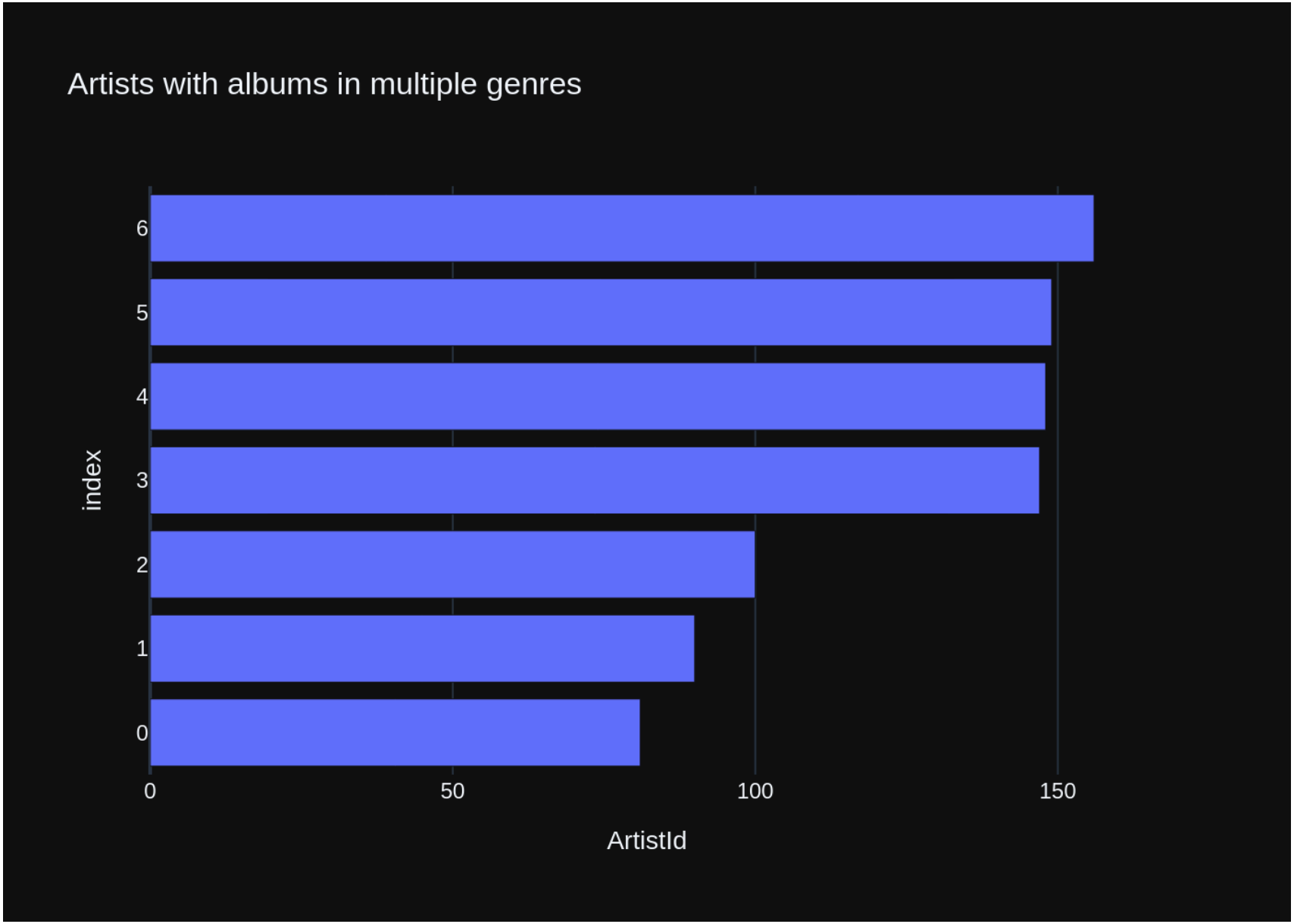
vn.ask(question=question)
```

```
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 format 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 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,\nFOREIGN 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_TrackGenreId ON Track (GenreId)\n\nCREATE INDEX IFK_TrackAlbumId ON Track (AlbumId)\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\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 (ArtistId)\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   TrackId 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   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 \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 top 10 most popular artists (based on the number of tracks):\n'}, {'role': 'assistant', 'content': 'SELECT Artist.Name, COUNT(*) AS TrackCount\nFROM Artist\nJOIN Album ON Artist.ArtistId = Album.ArtistId\nJOIN Track ON Album.AlbumId = Track.AlbumId\nGROUP BY Artist.Name\nORDER BY TrackCount DESC\nLIMIT 10;'}, {'role': 'user', 'content': ' \n   List all albums and their corresponding artist names \n'}, {'role': 'assistant', 'content': 'SELECT Album.Title, Artist.Name\nFROM Album\nJOIN Artist ON Album.ArtistId = Artist.ArtistId;'}, {'role': 'user', 'content': ' \n   List all genres and the number of tracks in each genre:\n'}, {'role': 'assistant', 'content': 'SELECT Genre.Name, COUNT(*) AS TrackCount\nFROM Track\nJOIN Genre ON Track.GenreId = Genre.GenreId\nGROUP BY Genre.Name;'}, {'role': 'user', 'content': ' \n   Get all playlists containing at least 10 tracks and the total duration of those tracks:\n'}, {'role': 'assistant', 'content': 'SELECT Playlist.PlaylistId, Playlist.Name, SUM(Track.Milliseconds) AS TotalDuration\nFROM Playlist\nJOIN PlaylistTrack ON Playlist.PlaylistId = PlaylistTrack.PlaylistId\nJOIN Track ON PlaylistTrack.TrackId = Track.TrackId\nGROUP BY Playlist.PlaylistId, Playlist.Name\nHAVING COUNT(Track.TrackId) >= 10;'}, {'role': 'user', 'content': ' \n   Find the top 5 customer who bought the most albums in total quantity (across all invoices):\n'}, {'role': 'assistant', 'content': 'SELECT CustomerId, SUM(Quantity) AS TotalAlbumsBought\nFROM (\n   SELECT Invoice.CustomerId, SUM(InvoiceLine.Quantity) AS Quantity\n   FROM Invoice\n

JOIN InvoiceLine ON Invoice.InvoiceId = InvoiceLine.InvoiceId\n   GROUP BY Invoice.CustomerId, InvoiceLin
e.TrackId\n)\nGROUP BY CustomerId\nORDER BY TotalAlbumsBought DESC\nLIMIT 5;'}, {'role': 'user', 'content':
' \n   Find the customer who bought the most albums in total quantity (across all invoices): \n'}, {'rol
e': 'assistant', 'content': 'SELECT CustomerId, SUM(TotalQuantity) AS TotalAlbumsBought\nFROM (\n   SELECT
Invoice.CustomerId, SUM(InvoiceLine.Quantity) AS TotalQuantity\n   FROM Invoice\n   JOIN InvoiceLine ON I
nvoice.InvoiceId = InvoiceLine.InvoiceId\n   GROUP BY Invoice.CustomerId\n)\nGROUP BY CustomerId\nORDER BY
TotalAlbumsBought DESC\nLIMIT 1;'}, {'role': 'user', 'content': ' \n   Find the top 5 most expensive trac
ks (based on unit price):\n'}, {'role': 'assistant', 'content': 'SELECT *\nFROM Track\nORDER BY UnitPrice D
ESC\nLIMIT 5;'}, {'role': 'user', 'content': ' \n   Find all tracks with a name containing "What" (case-i
nsensitive)\n'}, {'role': 'assistant', 'content': "SELECT *\nFROM Track\nWHERE Name LIKE '%What%';"}, {'rol
e': 'user', 'content': 'Show me a list of tables in the SQLite database'}, {'role': 'assistant', 'content':
"SELECT name\nFROM sqlite_master\nWHERE type = 'table';"}, {'role': 'user', 'content': 'what are the top 5
countries that customers come from?'}, {'role': 'assistant', 'content': 'SELECT Country, COUNT(*) AS Custom
erCount\nFROM Customer\nGROUP BY Country\nORDER BY CustomerCount DESC\nLIMIT 5;'}, {'role': 'user', 'conten
t': ' \n   Identify artists who have albums with tracks appearing in multiple genres:\n'}]
Using model gpt-3.5-turbo for 1290.25 tokens (approx)
Intermediate SQL
```sql
SELECT DISTINCT ArtistId
FROM Album
WHERE AlbumId IN (
    SELECT DISTINCT AlbumId
    FROM Track
    GROUP BY AlbumId
    HAVING COUNT(DISTINCT GenreId) > 1
);
```
```
SELECT DISTINCT ArtistId
FROM Album
WHERE AlbumId IN (
    SELECT DISTINCT AlbumId
    FROM Track
    GROUP BY AlbumId
    HAVING COUNT(DISTINCT GenreId) > 1
);
SELECT DISTINCT ArtistId
FROM Album
WHERE AlbumId IN (
    SELECT DISTINCT AlbumId
    FROM Track
    GROUP BY AlbumId
    HAVING COUNT(DISTINCT GenreId) > 1
```

```
);
   ArtistId
0        81
1        90
2       100
3       147
4       148
5       149
6       156
Using model gpt-3.5-turbo for 205.25 tokens (approx)
```


Artists with albums in multiple genres
index
6
5
4
3
2
1
0
0
50
100
150
ArtistId

Out[41]:
```
('SELECT DISTINCT ArtistId\nFROM Album\nWHERE AlbumId IN (\n    SELECT DISTINCT AlbumId\n    FROM Track\n
GROUP BY AlbumId\n    HAVING COUNT(DISTINCT GenreId) > 1\n);',
    ArtistId
 0        81
 1        90
 2       100
 3       147
 4       148
 5       149
 6       156,
 Figure({
     'data': [{'alignmentgroup': 'True',
               'hovertemplate': 'ArtistId=%{x}<br>index=%{y}<extra></extra>',
               'legendgroup': '',
               'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
               'name': '',
               'offsetgroup': '',
               'orientation': 'h',
               'showlegend': False,
               'textposition': 'auto',
               'type': 'bar',
               'x': array([ 81,  90, 100, 147, 148, 149, 156]),
               'xaxis': 'x',
               'y': array([0, 1, 2, 3, 4, 5, 6]),
               'yaxis': 'y'}],
     'layout': {'barmode': 'relative',
                'legend': {'tracegroupgap': 0},
                'template': '...',
                'title': {'text': 'Artists with albums in multiple genres'},
                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'ArtistId'}},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'index'}}}
 }))
```

Check completion time

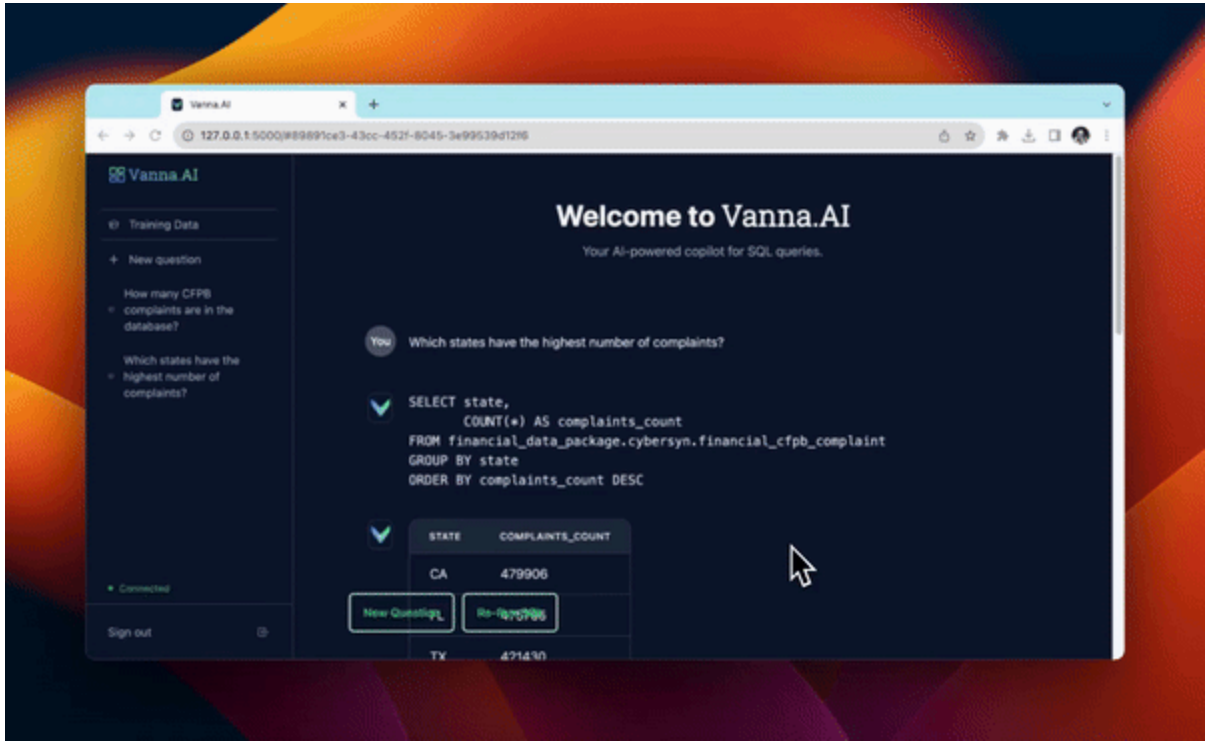
In [42]:
```
ts_stop = time()

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 'gpt-3.5-turbo' LLM took : 71.73 sec
```

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

```
2024-06-20 20:16:41.085922
```

Launch the User Interface



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