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

In [1]:
```
!pip install 'vanna[chromadb,gemini]'
```

```
Requirement already satisfied: vanna[chromadb,gemini] in /home/papagame/anaconda3/envs/vanna/lib/python3.1
1/site-packages (0.5.5)
Requirement already satisfied: requests in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-packages
(from vanna[chromadb,gemini]) (2.32.3)
Requirement already satisfied: tabulate in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-packages
(from vanna[chromadb,gemini]) (0.9.0)
Requirement already satisfied: plotly in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-packages
(from vanna[chromadb,gemini]) (5.22.0)
Requirement already satisfied: pandas in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-packages
(from vanna[chromadb,gemini]) (2.2.2)
Requirement already satisfied: sqlparse in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-packages
(from vanna[chromadb,gemini]) (0.5.0)
Requirement already satisfied: kaleido in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-packages
(from vanna[chromadb,gemini]) (0.2.1)
Requirement already satisfied: flask in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-packages (f
rom vanna[chromadb,gemini]) (3.0.3)
Requirement already satisfied: flask-sock in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-packag
es (from vanna[chromadb,gemini]) (0.7.0)
Requirement already satisfied: sqlalchemy in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-packag
es (from vanna[chromadb,gemini]) (2.0.30)
Requirement already satisfied: chromadb in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-packages
(from vanna[chromadb,gemini]) (0.5.0)
Requirement already satisfied: google-generativeai in /home/papagame/anaconda3/envs/vanna/lib/python3.11/si
te-packages (from vanna[chromadb,gemini]) (0.7.0)
Requirement already satisfied: build>=1.0.3 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-pack
ages (from chromadb->vanna[chromadb,gemini]) (1.2.1)
Requirement already satisfied: pydantic>=1.9 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-pac
kages (from chromadb->vanna[chromadb,gemini]) (2.7.3)
Requirement already satisfied: chroma-hnswlib==0.7.3 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/
site-packages (from chromadb->vanna[chromadb,gemini]) (0.7.3)
Requirement already satisfied: fastapi>=0.95.2 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-p
ackages (from chromadb->vanna[chromadb,gemini]) (0.111.0)
Requirement already satisfied: uvicorn>=0.18.3 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-p
ackages (from uvicorn[standard]>=0.18.3->chromadb->vanna[chromadb,gemini]) (0.30.1)
Requirement already satisfied: numpy>=1.22.5 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-pac
kages (from chromadb->vanna[chromadb,gemini]) (1.26.4)
Requirement already satisfied: posthog>=2.4.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-pa
ckages (from chromadb->vanna[chromadb,gemini]) (3.5.0)
Requirement already satisfied: typing-extensions>=4.5.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.
11/site-packages (from chromadb->vanna[chromadb,gemini]) (4.12.1)
Requirement already satisfied: onnxruntime>=1.14.1 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/si
te-packages (from chromadb->vanna[chromadb,gemini]) (1.18.0)
```

```
Requirement already satisfied: opentelemetry-api>=1.2.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.
11/site-packages (from chromadb->vanna[chromadb,gemini]) (1.25.0)
Requirement already satisfied: opentelemetry-exporter-otlp-proto-grpc>=1.2.0 in /home/papagame/anaconda3/en
vs/vanna/lib/python3.11/site-packages (from chromadb->vanna[chromadb,gemini]) (1.25.0)
Requirement already satisfied: opentelemetry-instrumentation-fastapi>=0.41b0 in /home/papagame/anaconda3/en
vs/vanna/lib/python3.11/site-packages (from chromadb->vanna[chromadb,gemini]) (0.46b0)
Requirement already satisfied: opentelemetry-sdk>=1.2.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.
11/site-packages (from chromadb->vanna[chromadb,gemini]) (1.25.0)
Requirement already satisfied: tokenizers>=0.13.2 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/sit
e-packages (from chromadb->vanna[chromadb,gemini]) (0.19.1)
Requirement already satisfied: pypika>=0.48.9 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-pa
ckages (from chromadb->vanna[chromadb,gemini]) (0.48.9)
Requirement already satisfied: tqdm>=4.65.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-pack
ages (from chromadb->vanna[chromadb,gemini]) (4.66.4)
Requirement already satisfied: overrides>=7.3.1 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-
packages (from chromadb->vanna[chromadb,gemini]) (7.7.0)
Requirement already satisfied: importlib-resources in /home/papagame/anaconda3/envs/vanna/lib/python3.11/si
te-packages (from chromadb->vanna[chromadb,gemini]) (6.4.0)
Requirement already satisfied: grpcio>=1.58.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-pa
ckages (from chromadb->vanna[chromadb,gemini]) (1.64.1)
Requirement already satisfied: bcrypt>=4.0.1 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-pac
kages (from chromadb->vanna[chromadb,gemini]) (4.1.3)
Requirement already satisfied: typer>=0.9.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-pack
ages (from chromadb->vanna[chromadb,gemini]) (0.12.3)
Requirement already satisfied: kubernetes>=28.1.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/sit
e-packages (from chromadb->vanna[chromadb,gemini]) (29.0.0)
Requirement already satisfied: tenacity>=8.2.3 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-p
ackages (from chromadb->vanna[chromadb,gemini]) (8.3.0)
Requirement already satisfied: PyYAML>=6.0.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-pac
kages (from chromadb->vanna[chromadb,gemini]) (6.0.1)
Requirement already satisfied: mmh3>=4.0.1 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-packa
ges (from chromadb->vanna[chromadb,gemini]) (4.1.0)
Requirement already satisfied: orjson>=3.9.12 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-pa
ckages (from chromadb->vanna[chromadb,gemini]) (3.10.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /home/papagame/anaconda3/envs/vanna/lib/python3.
11/site-packages (from requests->vanna[chromadb,gemini]) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-pack
ages (from requests->vanna[chromadb,gemini]) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/sit
e-packages (from requests->vanna[chromadb,gemini]) (2.2.1)
Requirement already satisfied: certifi>=2017.4.17 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/sit
e-packages (from requests->vanna[chromadb,gemini]) (2024.6.2)
```

```
Requirement already satisfied: Werkzeug>=3.0.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-p
ackages (from flask->vanna[chromadb,gemini]) (3.0.3)
Requirement already satisfied: Jinja2>=3.1.2 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-pac
kages (from flask->vanna[chromadb,gemini]) (3.1.4)
Requirement already satisfied: itsdangerous>=2.1.2 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/si
te-packages (from flask->vanna[chromadb,gemini]) (2.2.0)
Requirement already satisfied: click>=8.1.3 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-pack
ages (from flask->vanna[chromadb,gemini]) (8.1.7)
Requirement already satisfied: blinker>=1.6.2 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-pa
ckages (from flask->vanna[chromadb,gemini]) (1.8.2)
Requirement already satisfied: simple-websocket>=0.5.1 in /home/papagame/anaconda3/envs/vanna/lib/python3.1
1/site-packages (from flask-sock->vanna[chromadb,gemini]) (1.0.0)
Requirement already satisfied: google-ai-generativelanguage==0.6.5 in /home/papagame/anaconda3/envs/vanna/l
ib/python3.11/site-packages (from google-generativeai->vanna[chromadb,gemini]) (0.6.5)
Requirement already satisfied: google-api-core in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-p
ackages (from google-generativeai->vanna[chromadb,gemini]) (2.19.0)
Requirement already satisfied: google-api-python-client in /home/papagame/anaconda3/envs/vanna/lib/python3.
11/site-packages (from google-generativeai->vanna[chromadb,gemini]) (2.134.0)
Requirement already satisfied: google-auth>=2.15.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/si
te-packages (from google-generativeai->vanna[chromadb,gemini]) (2.29.0)
Requirement already satisfied: protobuf in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-packages
(from google-generativeai->vanna[chromadb,gemini]) (4.25.3)
Requirement already satisfied: proto-plus<2.0.0dev,>=1.22.3 in /home/papagame/anaconda3/envs/vanna/lib/pyth
on3.11/site-packages (from google-ai-generativelanguage==0.6.5->google-generativeai->vanna[chromadb,gemin
i]) (1.24.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /home/papagame/anaconda3/envs/vanna/lib/python3.1
1/site-packages (from pandas->vanna[chromadb,gemini]) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-pack
ages (from pandas->vanna[chromadb,gemini]) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-pa
ckages (from pandas->vanna[chromadb,gemini]) (2024.1)
Requirement already satisfied: packaging in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-package
s (from plotly->vanna[chromadb,gemini]) (24.0)
Requirement already satisfied: greenlet!=0.4.17 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-
packages (from sqlalchemy->vanna[chromadb,gemini]) (3.0.3)
Requirement already satisfied: pyproject_hooks in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-p
ackages (from build>=1.0.3->chromadb->vanna[chromadb,gemini]) (1.1.0)
Requirement already satisfied: starlette<0.38.0,>=0.37.2 in /home/papagame/anaconda3/envs/vanna/lib/python
3.11/site-packages (from fastapi>=0.95.2->chromadb->vanna[chromadb,gemini]) (0.37.2)
Requirement already satisfied: fastapi-cli>=0.0.2 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/sit
e-packages (from fastapi>=0.95.2->chromadb->vanna[chromadb,gemini]) (0.0.4)
Requirement already satisfied: httpx>=0.23.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-pac
```

```
kages (from fastapi>=0.95.2->chromadb->vanna[chromadb,gemini]) (0.27.0)
Requirement already satisfied: python-multipart>=0.0.7 in /home/papagame/anaconda3/envs/vanna/lib/python3.1
1/site-packages (from fastapi>=0.95.2->chromadb->vanna[chromadb,gemini]) (0.0.9)
Requirement already satisfied: ujson!=4.0.2,!=4.1.0,!=4.2.0,!=4.3.0,!=5.0.0,!=5.1.0,>=4.0.1 in /home/papaga
me/anaconda3/envs/vanna/lib/python3.11/site-packages (from fastapi>=0.95.2->chromadb->vanna[chromadb,gemin
i]) (5.10.0)
Requirement already satisfied: email_validator>=2.0.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.1
1/site-packages (from fastapi>=0.95.2->chromadb->vanna[chromadb,gemini]) (2.1.1)
Requirement already satisfied: googleapis-common-protos<2.0.dev0,>=1.56.2 in /home/papagame/anaconda3/envs/
vanna/lib/python3.11/site-packages (from google-api-core->google-generativeai->vanna[chromadb,gemini]) (1.6
3.1)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.1
1/site-packages (from google-auth>=2.15.0->google-generativeai->vanna[chromadb,gemini]) (5.3.3)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/
site-packages (from google-auth>=2.15.0->google-generativeai->vanna[chromadb,gemini]) (0.4.0)
Requirement already satisfied: rsa<5,>=3.1.4 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-pac
kages (from google-auth>=2.15.0->google-generativeai->vanna[chromadb,gemini]) (4.9)
Requirement already satisfied: MarkupSafe>=2.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-p
ackages (from Jinja2>=3.1.2->flask->vanna[chromadb,gemini]) (2.1.5)
Requirement already satisfied: six>=1.9.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-packag
es (from kubernetes>=28.1.0->chromadb->vanna[chromadb,gemini]) (1.16.0)
Requirement already satisfied: websocket-client!=0.40.0,!=0.41.*,!=0.42.*,>=0.32.0 in /home/papagame/anacon
da3/envs/vanna/lib/python3.11/site-packages (from kubernetes>=28.1.0->chromadb->vanna[chromadb,gemini]) (1.
8.0)
Requirement already satisfied: requests-oauthlib in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site
-packages (from kubernetes>=28.1.0->chromadb->vanna[chromadb,gemini]) (2.0.0)
Requirement already satisfied: oauthlib>=3.2.2 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-p
ackages (from kubernetes>=28.1.0->chromadb->vanna[chromadb,gemini]) (3.2.2)
Requirement already satisfied: coloredlogs in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-packa
ges (from onnxruntime>=1.14.1->chromadb->vanna[chromadb,gemini]) (15.0.1)
Requirement already satisfied: flatbuffers in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-packa
ges (from onnxruntime>=1.14.1->chromadb->vanna[chromadb,gemini]) (24.3.25)
Requirement already satisfied: sympy in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-packages (f
rom onnxruntime>=1.14.1->chromadb->vanna[chromadb,gemini]) (1.12.1)
Requirement already satisfied: deprecated>=1.2.6 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site
-packages (from opentelemetry-api>=1.2.0->chromadb->vanna[chromadb,gemini]) (1.2.14)
Requirement already satisfied: importlib-metadata<=7.1,>=6.0 in /home/papagame/anaconda3/envs/vanna/lib/pyt
hon3.11/site-packages (from opentelemetry-api>=1.2.0->chromadb->vanna[chromadb,gemini]) (7.1.0)
Requirement already satisfied: opentelemetry-exporter-otlp-proto-common==1.25.0 in /home/papagame/anaconda
3/envs/vanna/lib/python3.11/site-packages (from opentelemetry-exporter-otlp-proto-grpc>=1.2.0->chromadb->va
nna[chromadb,gemini]) (1.25.0)
Requirement already satisfied: opentelemetry-proto==1.25.0 in /home/papagame/anaconda3/envs/vanna/lib/pytho
```

```
n3.11/site-packages (from opentelemetry-exporter-otlp-proto-grpc>=1.2.0->chromadb->vanna[chromadb,gemini])
(1.25.0)
Requirement already satisfied: opentelemetry-instrumentation-asgi==0.46b0 in /home/papagame/anaconda3/envs/
vanna/lib/python3.11/site-packages (from opentelemetry-instrumentation-fastapi>=0.41b0->chromadb->vanna[chr
omadb,gemini]) (0.46b0)
Requirement already satisfied: opentelemetry-instrumentation==0.46b0 in /home/papagame/anaconda3/envs/vann
a/lib/python3.11/site-packages (from opentelemetry-instrumentation-fastapi>=0.41b0->chromadb->vanna[chromad
b,gemini]) (0.46b0)
Requirement already satisfied: opentelemetry-semantic-conventions==0.46b0 in /home/papagame/anaconda3/envs/
vanna/lib/python3.11/site-packages (from opentelemetry-instrumentation-fastapi>=0.41b0->chromadb->vanna[chr
omadb,gemini]) (0.46b0)
Requirement already satisfied: opentelemetry-util-http==0.46b0 in /home/papagame/anaconda3/envs/vanna/lib/p
ython3.11/site-packages (from opentelemetry-instrumentation-fastapi>=0.41b0->chromadb->vanna[chromadb,gemin
i]) (0.46b0)
Requirement already satisfied: setuptools>=16.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-
packages (from opentelemetry-instrumentation==0.46b0->opentelemetry-instrumentation-fastapi>=0.41b0->chroma
db->vanna[chromadb,gemini]) (69.5.1)
Requirement already satisfied: wrapt<2.0.0,>=1.0.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/si
te-packages (from opentelemetry-instrumentation==0.46b0->opentelemetry-instrumentation-fastapi>=0.41b0->chr
omadb->vanna[chromadb,gemini]) (1.16.0)
Requirement already satisfied: asgiref~=3.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-pack
ages (from opentelemetry-instrumentation-asgi==0.46b0->opentelemetry-instrumentation-fastapi>=0.41b0->chrom
adb->vanna[chromadb,gemini]) (3.8.1)
Requirement already satisfied: monotonic>=1.5 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-pa
ckages (from posthog>=2.4.0->chromadb->vanna[chromadb,gemini]) (1.6)
Requirement already satisfied: backoff>=1.10.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-p
ackages (from posthog>=2.4.0->chromadb->vanna[chromadb,gemini]) (2.2.1)
Requirement already satisfied: annotated-types>=0.4.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.1
1/site-packages (from pydantic>=1.9->chromadb->vanna[chromadb,gemini]) (0.7.0)
Requirement already satisfied: pydantic-core==2.18.4 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/
site-packages (from pydantic>=1.9->chromadb->vanna[chromadb,gemini]) (2.18.4)
Requirement already satisfied: wsproto in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-packages
(from simple-websocket>=0.5.1->flask-sock->vanna[chromadb,gemini]) (1.2.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.16.4 in /home/papagame/anaconda3/envs/vanna/lib/pyth
on3.11/site-packages (from tokenizers>=0.13.2->chromadb->vanna[chromadb,gemini]) (0.23.2)
Requirement already satisfied: shellingham>=1.3.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/sit
e-packages (from typer>=0.9.0->chromadb->vanna[chromadb,gemini]) (1.5.4)
Requirement already satisfied: rich>=10.11.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-pac
kages (from typer>=0.9.0->chromadb->vanna[chromadb,gemini]) (13.7.1)
Requirement already satisfied: h11>=0.8 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-packages
(from uvicorn>=0.18.3->uvicorn[standard]>=0.18.3->chromadb->vanna[chromadb,gemini]) (0.14.0)
Requirement already satisfied: httptools>=0.5.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-
```

```
packages (from uvicorn[standard]>=0.18.3->chromadb->vanna[chromadb,gemini]) (0.6.1)
Requirement already satisfied: python-dotenv>=0.13 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/si
te-packages (from uvicorn[standard]>=0.18.3->chromadb->vanna[chromadb,gemini]) (1.0.1)
Requirement already satisfied: uvloop!=0.15.0,!=0.15.1,>=0.14.0 in /home/papagame/anaconda3/envs/vanna/lib/
python3.11/site-packages (from uvicorn[standard]>=0.18.3->chromadb->vanna[chromadb,gemini]) (0.19.0)
Requirement already satisfied: watchfiles>=0.13 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-
packages (from uvicorn[standard]>=0.18.3->chromadb->vanna[chromadb,gemini]) (0.22.0)
Requirement already satisfied: websockets>=10.4 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-
packages (from uvicorn[standard]>=0.18.3->chromadb->vanna[chromadb,gemini]) (12.0)
Requirement already satisfied: httplib2<1.dev0,>=0.19.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.
11/site-packages (from google-api-python-client->google-generativeai->vanna[chromadb,gemini]) (0.22.0)
Requirement already satisfied: google-auth-httplib2<1.0.0,>=0.2.0 in /home/papagame/anaconda3/envs/vanna/li
b/python3.11/site-packages (from google-api-python-client->google-generativeai->vanna[chromadb,gemini]) (0.
2.0)
Requirement already satisfied: uritemplate<5,>=3.0.1 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/
site-packages (from google-api-python-client->google-generativeai->vanna[chromadb,gemini]) (4.1.1)
Requirement already satisfied: dnspython>=2.0.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-
packages (from email_validator>=2.0.0->fastapi>=0.95.2->chromadb->vanna[chromadb,gemini]) (2.6.1)
Requirement already satisfied: grpcio-status<2.0.dev0,>=1.33.2 in /home/papagame/anaconda3/envs/vanna/lib/p
ython3.11/site-packages (from google-api-core[grpc]!=2.0.*,!=2.1.*,!=2.10.*,!=2.2.*,!=2.3.*,!=2.4.*,!=2.5.
*,!=2.6.*,!=2.7.*,!=2.8.*,!=2.9.*,<3.0.0dev,>=1.34.1->google-ai-generativelanguage==0.6.5->google-generativ
eai->vanna[chromadb,gemini]) (1.62.2)
Requirement already satisfied: pyparsing!=3.0.0,!=3.0.1,!=3.0.2,!=3.0.3,<4,>=2.4.2 in /home/papagame/anacon
da3/envs/vanna/lib/python3.11/site-packages (from httplib2<1.dev0,>=0.19.0->google-api-python-client->googl
e-generativeai->vanna[chromadb,gemini]) (3.1.2)
Requirement already satisfied: anyio in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-packages (f
rom httpx>=0.23.0->fastapi>=0.95.2->chromadb->vanna[chromadb,gemini]) (4.4.0)
Requirement already satisfied: httpcore==1.* in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-pac
kages (from httpx>=0.23.0->fastapi>=0.95.2->chromadb->vanna[chromadb,gemini]) (1.0.5)
Requirement already satisfied: sniffio in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-packages
(from httpx>=0.23.0->fastapi>=0.95.2->chromadb->vanna[chromadb,gemini]) (1.3.1)
Requirement already satisfied: filelock in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-packages
(from huggingface-hub<1.0,>=0.16.4->tokenizers>=0.13.2->chromadb->vanna[chromadb,gemini]) (3.14.0)
Requirement already satisfied: fsspec>=2023.5.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-
packages (from huggingface-hub<1.0,>=0.16.4->tokenizers>=0.13.2->chromadb->vanna[chromadb,gemini]) (2024.6.
0)
Requirement already satisfied: zipp>=0.5 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-package
s (from importlib-metadata<=7.1,>=6.0->opentelemetry-api>=1.2.0->chromadb->vanna[chromadb,gemini]) (3.19.2)
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/s
ite-packages (from pyasn1-modules>=0.2.1->google-auth>=2.15.0->google-generativeai->vanna[chromadb,gemini])
(0.6.0)
Requirement already satisfied: markdown-it-py>=2.2.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/
```

```
site-packages (from rich>=10.11.0->typer>=0.9.0->chromadb->vanna[chromadb,gemini]) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.1
1/site-packages (from rich>=10.11.0->typer>=0.9.0->chromadb->vanna[chromadb,gemini]) (2.18.0)
Requirement already satisfied: humanfriendly>=9.1 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/sit
e-packages (from coloredlogs->onnxruntime>=1.14.1->chromadb->vanna[chromadb,gemini]) (10.0)
Requirement already satisfied: mpmath<1.4.0,>=1.1.0 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/s
ite-packages (from sympy->onnxruntime>=1.14.1->chromadb->vanna[chromadb,gemini]) (1.3.0)
Requirement already satisfied: mdurl~=0.1 in /home/papagame/anaconda3/envs/vanna/lib/python3.11/site-packag
es (from markdown-it-py>=2.2.0->rich>=10.11.0->typer>=0.9.0->chromadb->vanna[chromadb,gemini]) (0.1.2)
```

In [2]:
```
model_name = 'gemini-1.5-flash'
file_db = "~/Downloads/chinook.sqlite"
```

Gemini Help

- How to get started
- Vertex AI API for Gemini

In [3]:
```
from api_key_store import ApiKeyStore
s = ApiKeyStore()

google_api_key = s.get_api_key(provider="GOOGLE/VERTEX_AI")
```

google_api_key

In [4]:
```
from vanna.google import GoogleGeminiChat
from vanna.chromadb.chromadb_vector import ChromaDB_VectorStore
```

In [5]:
```
class MyVanna(ChromaDB_VectorStore, GoogleGeminiChat):
    def __init__(self, config=None):
        ChromaDB_VectorStore.__init__(self, config=config)
        GoogleGeminiChat.__init__(self, config=config)

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

```
/home/papagame/anaconda3/envs/vanna/lib/python3.11/site-packages/tqdm/auto.py:21: TqdmWarning: IProgress no
t found. Please update jupyter and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user_instal
l.html
  from .autonotebook import tqdm as notebook_tqdm
```

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 [6]:
```
import os
import re
from time import time
```

In [7]:
```
# file_db = "./db/gpt3sql.sqlite"

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

In [8]:
```
vn.run_sql_is_set
```

Out[8]:
```
True
```

In [9]:
```
clean_and_train = True  # False
```

In [10]:
```
hostname = os.uname().nodename
print("Hostname:", hostname)
```

```
Hostname: papa-game
```

In [11]:
```
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 [12]:
```
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 [13]:
```
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 [14]:
```
# show training data
training_data = vn.get_training_data()
training_data
```

Out[14]:

id	question	content	training_data_type

In [15]:
```
df_ddl = vn.run_sql("SELECT type, sql FROM sqlite_master WHERE sql is not null")
```

In [16]:
```
df_ddl
```

Out[16]:

	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 [17]:

```
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 [18]:
```
ts_start = time()
```

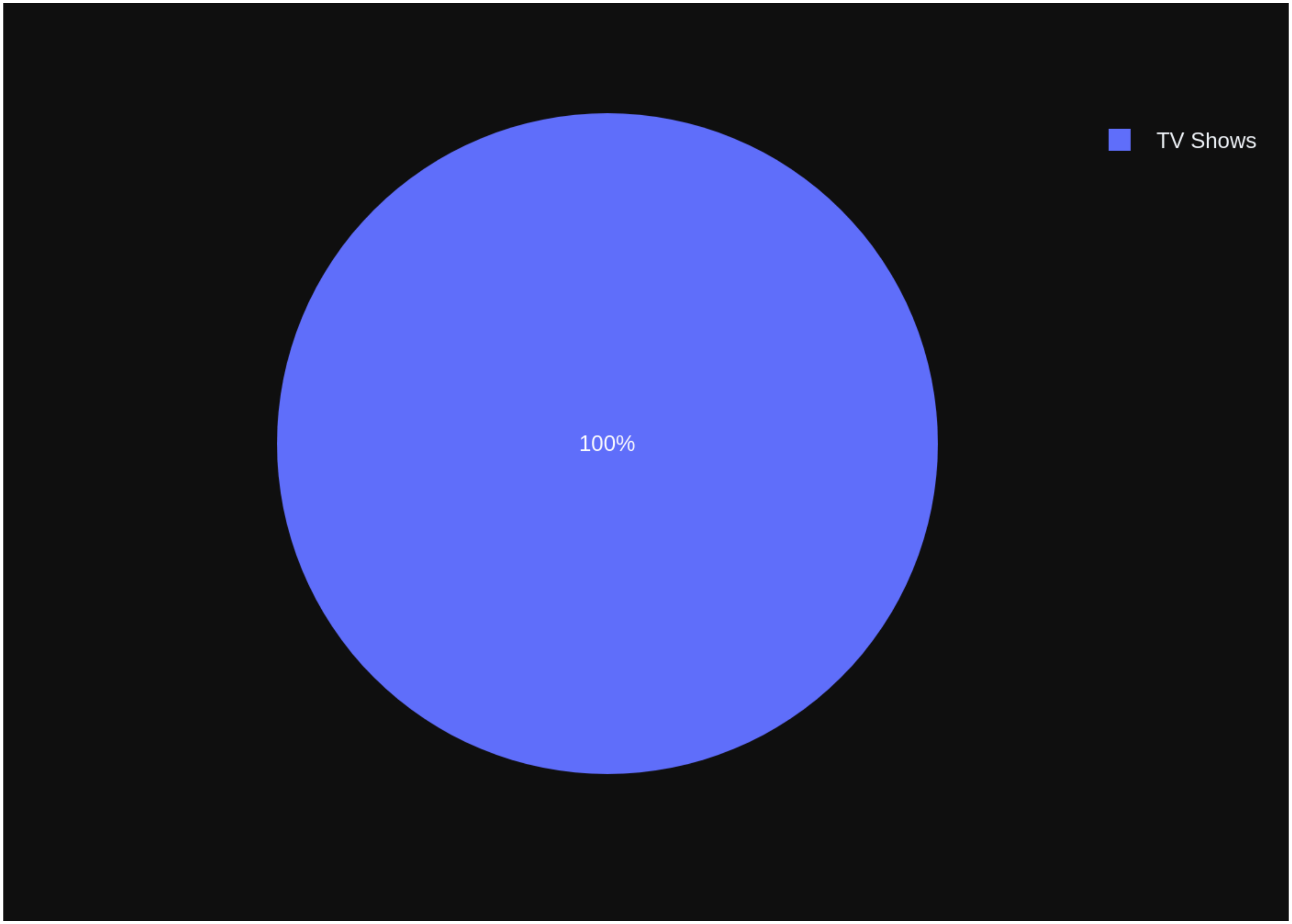
In [19]:
```
vn.ask(question="Show me a list of tables in the SQLite database")
```

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

~~~
["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 Playlist\n(\n    PlaylistId INTEGER  NOT NULL,\n    Name NVARCHAR(120),\n    CONSTRAINT PK_P
laylist PRIMARY KEY  (PlaylistId)\n)\n\nCREATE TABLE InvoiceLine\n(\n    InvoiceLineId INTEGER  NOT NULL,\n
InvoiceId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    Q
uantity INTEGER  NOT NULL,\n    CONSTRAINT PK_InvoiceLine PRIMARY KEY  (InvoiceLineId),\n    FOREIGN KEY (I
nvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (T
rackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Playlis
tTrack\n(\n    PlaylistId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    CONSTRAINT PK_PlaylistTra
ck 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 D
ELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Track\n(\n    TrackId INTEGER  NOT NULL,\n    Name N
VARCHAR(200)  NOT NULL,\n    AlbumId INTEGER,\n    MediaTypeId INTEGER  NOT NULL,\n    GenreId INTEGER,\n
Composer NVARCHAR(220),\n    Milliseconds INTEGER  NOT NULL,\n    Bytes INTEGER,\n    UnitPrice NUMERIC(10,
2)  NOT NULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (TrackId),\n    FOREIGN KEY (AlbumId) REFERENCES Album
(AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (GenreId) REFERENCES Genre (Genre
Id) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (MediaTypeId) REFERENCES MediaType (Med
iaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE MediaType\n(\n    MediaTypeId IN
TEGER  NOT NULL,\n    Name NVARCHAR(120),\n    CONSTRAINT PK_MediaType PRIMARY KEY  (MediaTypeId)\n)\n\nCRE
ATE TABLE Artist\n(\n    ArtistId INTEGER  NOT NULL,\n    Name NVARCHAR(120),\n    CONSTRAINT PK_Artist PRI
MARY 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 PRIMARY KEY  (AlbumId),\n    FOREIGN KE
Y (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  (Ge
nreId)\n)\n\nCREATE TABLE Invoice\n(\n    InvoiceId INTEGER  NOT NULL,\n    CustomerId INTEGER  NOT NULL,\n
InvoiceDate DATETIME  NOT NULL,\n    BillingAddress NVARCHAR(70),\n    BillingCity NVARCHAR(40),\n    Billi
ngState NVARCHAR(40),\n    BillingCountry NVARCHAR(40),\n    BillingPostalCode NVARCHAR(10),\n    Total NUM
ERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Invoice PRIMARY KEY  (InvoiceId),\n    FOREIGN KEY (CustomerId) RE
FERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_Employe
eReportsTo ON Employee (ReportsTo)\n\n\n===Additional Context \n\nIn the chinook database invoice means ord
er\n\n===Response Guidelines \n1. If the provided context is sufficient, please generate a valid SQL query
without any explanations for the question. \n2. If the provided context is almost sufficient but requires k
nowledge of a specific string in a particular column, please generate an intermediate SQL query to find the
distinct strings in that column. Prepend the query with a comment saying intermediate_sql \n3. If the provi
ded context is insufficient, please explain why it can't be generated. \n4. Please use the most relevant ta
ble(s). \n5. If the question has been asked and answered before, please repeat the answer exactly as it was
given before. \n", 'Show me a list of tables in the SQLite database']
```sql
SELECT
    Name
FROM
    Playlist
~~~

~~~
JOIN
    PlaylistTrack ON Playlist.PlaylistId = PlaylistTrack.PlaylistId
WHERE
    PlaylistTrack.TrackId IN (
        SELECT
            TrackId
        FROM
            InvoiceLine
        WHERE
            UnitPrice > 0.99
    );
```
SELECT
    Name
FROM
    Playlist
JOIN
    PlaylistTrack ON Playlist.PlaylistId = PlaylistTrack.PlaylistId
WHERE
    PlaylistTrack.TrackId IN (
        SELECT
            TrackId
        FROM
            InvoiceLine
        WHERE
            UnitPrice > 0.99
    );
SELECT
    Name
FROM
    Playlist
JOIN
    PlaylistTrack ON Playlist.PlaylistId = PlaylistTrack.PlaylistId
WHERE
    PlaylistTrack.TrackId IN (
        SELECT
            TrackId
        FROM
            InvoiceLine
        WHERE
            UnitPrice > 0.99
    );
~~~

```
          Name
0     TV Shows
1     TV Shows
2     TV Shows
3     TV Shows
4     TV Shows
..         ...
201   TV Shows
202   TV Shows
203   TV Shows
204   TV Shows
205   TV Shows

[206 rows x 1 columns]
```


100%
TV Shows

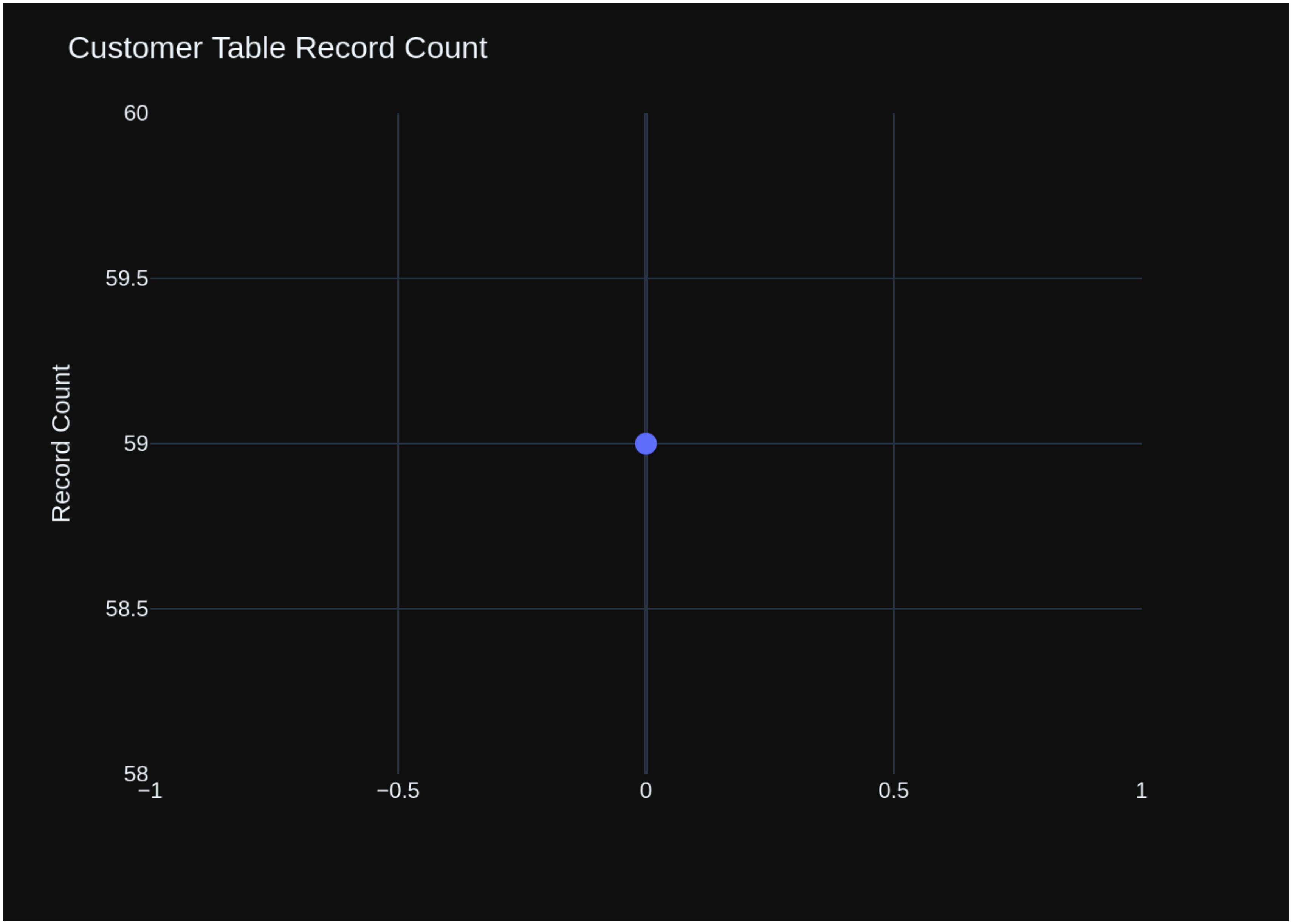
Out[19]:
```
('SELECT \n    Name \nFROM \n    Playlist\nJOIN \n    PlaylistTrack ON Playlist.PlaylistId = PlaylistTrac
k.PlaylistId\nWHERE \n    PlaylistTrack.TrackId IN (\n        SELECT \n            TrackId \n        FROM
\n            InvoiceLine\n        WHERE \n            UnitPrice > 0.99\n    );',
           Name
 0     TV Shows
 1     TV Shows
 2     TV Shows
 3     TV Shows
 4     TV Shows
 ..         ...
 201   TV Shows
 202   TV Shows
 203   TV Shows
 204   TV Shows
 205   TV Shows

 [206 rows x 1 columns],
 Figure({
     'data': [{'domain': {'x': [0.0, 1.0], 'y': [0.0, 1.0]},
               'hovertemplate': 'Name=%{label}<extra></extra>',
               'labels': array(['TV Shows', 'TV Shows', 'TV Shows', ..., 'TV Shows', 'TV Shows',
                                'TV Shows'], dtype=object),
               'legendgroup': '',
               'name': '',
               'showlegend': True,
               'type': 'pie'}],
     'layout': {'legend': {'tracegroupgap': 0}, 'margin': {'t': 60}, 'template': '...'}
 }))
```

In [20]:
```
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
```

```
["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    L
astName NVARCHAR(20)  NOT NULL,\n    Company NVARCHAR(80),\n    Address NVARCHAR(70),\n    City NVARCHAR(4
0),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVARCHAR(10),\n    Phone NVARCHAR(2
4),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60)  NOT NULL,\n    SupportRepId INTEGER,\n    CONSTRAINT PK
_Customer PRIMARY KEY  (CustomerId),\n    FOREIGN KEY (SupportRepId) REFERENCES Employee (EmployeeId) \n\t
\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Invoice\n(\n    InvoiceId INTEGER  NOT NULL,\n
CustomerId INTEGER  NOT NULL,\n    InvoiceDate DATETIME  NOT NULL,\n    BillingAddress NVARCHAR(70),\n    B
illingCity NVARCHAR(40),\n    BillingState NVARCHAR(40),\n    BillingCountry NVARCHAR(40),\n    BillingPost
alCode NVARCHAR(10),\n    Total NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Invoice PRIMARY KEY  (InvoiceI
d),\n    FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO A
CTION\n)\n\nCREATE TABLE InvoiceLine\n(\n    InvoiceLineId INTEGER  NOT NULL,\n    InvoiceId INTEGER  NOT N
ULL,\n    TrackId INTEGER  NOT NULL,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    Quantity INTEGER  NOT NUL
L,\n    CONSTRAINT PK_InvoiceLine PRIMARY KEY  (InvoiceLineId),\n    FOREIGN KEY (InvoiceId) REFERENCES Inv
oice (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  N
OT NULL,\n    Title NVARCHAR(160)  NOT NULL,\n    ArtistId INTEGER  NOT NULL,\n    CONSTRAINT PK_Album PRIM
ARY KEY  (AlbumId),\n    FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON U
PDATE 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    Post
alCode NVARCHAR(10),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60),\n    CONSTRAI
NT 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 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_CustomerSupportRepId
ON Customer (SupportRepId)\n\nCREATE TABLE Playlist\n(\n    PlaylistId INTEGER  NOT NULL,\n    Name NVARCHA
R(120),\n    CONSTRAINT PK_Playlist PRIMARY KEY  (PlaylistId)\n)\n\nCREATE TABLE PlaylistTrack\n(\n    Play
listId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    CONSTRAINT PK_PlaylistTrack PRIMARY KEY  (Pl
aylistId, TrackId),\n    FOREIGN KEY (PlaylistId) REFERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTIO
N ON UPDATE NO ACTION,\n    FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON U
PDATE NO ACTION\n)\n\n\n\n===Additional Context \n\nIn the chinook database invoice means order\n\n===Respons
e Guidelines \n1. If the provided context is sufficient, please generate a valid SQL query without any expl
anations for the question. \n2. If the provided context is almost sufficient but requires knowledge of a sp
ecific string in a particular column, please generate an intermediate SQL query to find the distinct string
s 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", 'Show me a list of tables in the SQLite database', 'SELECT \n    Name \nFROM \n    Playlist\nJOIN \n
PlaylistTrack ON Playlist.PlaylistId = PlaylistTrack.PlaylistId\nWHERE \n    PlaylistTrack.TrackId IN (\n
SELECT \n            TrackId \n        FROM \n            InvoiceLine\n        WHERE \n            UnitPric
e > 0.99\n    );', 'How many records are in table called customer']
```sql
SELECT
    COUNT(*)
FROM
    Customer;
```
SELECT
    COUNT(*)
FROM
    Customer;
SELECT
    COUNT(*)
FROM
    Customer;
   COUNT(*)
0        59
~~~


Customer Table Record Count
Record Count
60
59.5
59
58.5
58
−1
−0.5
0
0.5
1

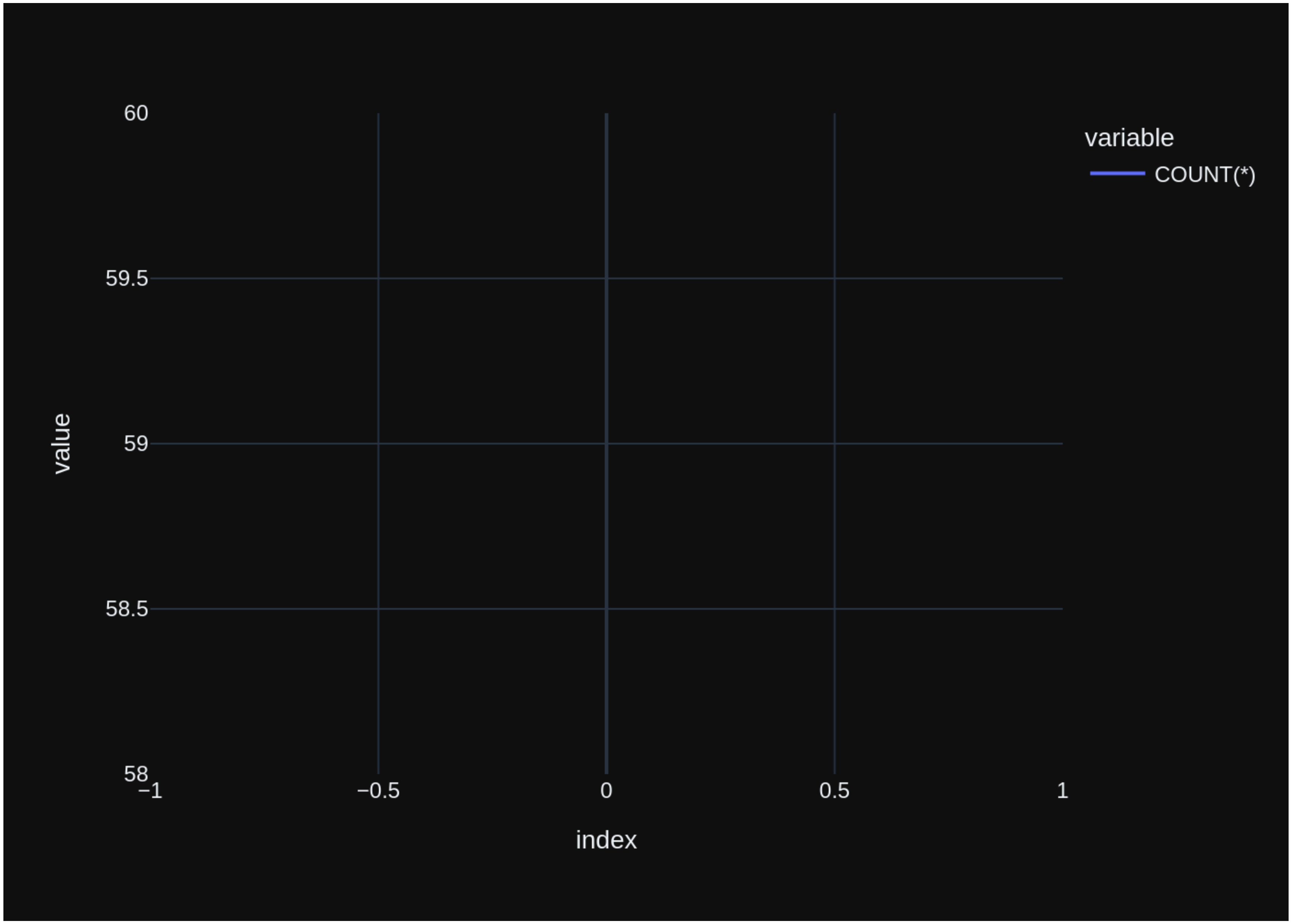
```
Out[20]: ('SELECT \n    COUNT(*) \nFROM \n    Customer;',
             COUNT(*)
          0        59,
          Figure({
              'data': [{'hovertemplate': 'index=%{x}<br>COUNT(*)=%{y}<extra></extra>',
                        'legendgroup': '',
                        'marker': {'color': '#636efa', 'size': 12, 'symbol': 'circle'},
                        'mode': 'markers',
                        'name': '',
                        'orientation': 'v',
                        'showlegend': False,
                        'type': 'scatter',
                        'x': array([0]),
                        'xaxis': 'x',
                        'y': array([59]),
                        'yaxis': 'y'}],
              'layout': {'legend': {'tracegroupgap': 0},
                         'margin': {'t': 60},
                         'template': '...',
                         'title': {'text': 'Customer Table Record Count'},
                         'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': ''}},
                         'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'Record Count'}}}
          }))
```

```
In [21]: 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
```

```
["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    L
astName NVARCHAR(20)  NOT NULL,\n    Company NVARCHAR(80),\n    Address NVARCHAR(70),\n    City NVARCHAR(4
0),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVARCHAR(10),\n    Phone NVARCHAR(2
4),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60)  NOT NULL,\n    SupportRepId INTEGER,\n    CONSTRAINT PK
_Customer PRIMARY KEY  (CustomerId),\n    FOREIGN KEY (SupportRepId) REFERENCES Employee (EmployeeId) \n\t
\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_CustomerSupportRepId ON Customer (SupportR
epId)\n\nCREATE TABLE Invoice\n(\n    InvoiceId INTEGER  NOT NULL,\n    CustomerId INTEGER  NOT NULL,\n    
InvoiceDate DATETIME  NOT NULL,\n    BillingAddress NVARCHAR(70),\n    BillingCity NVARCHAR(40),\n    Billi
ngState NVARCHAR(40),\n    BillingCountry NVARCHAR(40),\n    BillingPostalCode NVARCHAR(10),\n    Total NUM
ERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Invoice PRIMARY KEY  (InvoiceId),\n    FOREIGN KEY (CustomerId) RE
FERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_Invoice
CustomerId ON Invoice (CustomerId)\n\nCREATE TABLE InvoiceLine\n(\n    InvoiceLineId INTEGER  NOT NULL,\n    
InvoiceId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    Q
uantity INTEGER  NOT NULL,\n    CONSTRAINT PK_InvoiceLine PRIMARY KEY  (InvoiceLineId),\n    FOREIGN KEY (I
nvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (T
rackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_Inv
oiceLineInvoiceId 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 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_InvoiceLineTrackId O
N 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 KE
Y (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 NVARCHA
R(20)  NOT NULL,\n    Title NVARCHAR(30),\n    ReportsTo INTEGER,\n    BirthDate DATETIME,\n    HireDate DA
TETIME,\n    Address NVARCHAR(70),\n    City NVARCHAR(40),\n    State NVARCHAR(40),\n    Country NVARCHAR(4
0),\n    PostalCode NVARCHAR(10),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(6
0),\n    CONSTRAINT PK_Employee PRIMARY KEY  (EmployeeId),\n    FOREIGN KEY (ReportsTo) REFERENCES Employee
(EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n===Additional Context \n\nIn the chinook
database invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficient, please g
enerate a valid SQL query without any explanations for the question. \n2. If the provided context is almost
sufficient but requires knowledge of a specific string in a particular column, please generate an intermedi
ate SQL query to find the distinct strings in that column. Prepend the query with a comment saying intermed
iate_sql \n3. If the provided context is insufficient, please explain why it can't be generated. \n4. Pleas
e use the most relevant table(s). \n5. If the question has been asked and answered before, please repeat th
e answer exactly as it was given before. \n", 'How many records are in table called customer', 'SELECT \n  
COUNT(*) \nFROM \n    Customer;', 'Show me a list of tables in the SQLite database', 'SELECT \n    Name \nF
```

~~~
ROM \n    Playlist\nJOIN \n    PlaylistTrack ON Playlist.PlaylistId = PlaylistTrack.PlaylistId\nWHERE \n
PlaylistTrack.TrackId IN (\n        SELECT \n            TrackId \n        FROM \n            InvoiceLine\n
WHERE \n            UnitPrice > 0.99\n    );', 'How many customers are there']
```sql
SELECT
    COUNT(*)
FROM
    Customer;
```
SELECT
    COUNT(*)
FROM
    Customer;
SELECT
    COUNT(*)
FROM
    Customer;
   COUNT(*)
0        59
~~~


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

```
Out[21]: ('SELECT \n    COUNT(*) \nFROM \n    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 [22]: 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
```

```
["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    L
astName NVARCHAR(20)  NOT NULL,\n    Company NVARCHAR(80),\n    Address NVARCHAR(70),\n    City NVARCHAR(4
0),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVARCHAR(10),\n    Phone NVARCHAR(2
4),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60)  NOT NULL,\n    SupportRepId INTEGER,\n    CONSTRAINT PK
_Customer PRIMARY KEY  (CustomerId),\n    FOREIGN KEY (SupportRepId) REFERENCES Employee (EmployeeId) \n\t
\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Invoice\n(\n    InvoiceId INTEGER  NOT NULL,\n
CustomerId INTEGER  NOT NULL,\n    InvoiceDate DATETIME  NOT NULL,\n    BillingAddress NVARCHAR(70),\n    B
illingCity NVARCHAR(40),\n    BillingState NVARCHAR(40),\n    BillingCountry NVARCHAR(40),\n    BillingPost
alCode NVARCHAR(10),\n    Total NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Invoice PRIMARY KEY  (InvoiceI
d),\n    FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO A
CTION\n)\n\nCREATE TABLE InvoiceLine\n(\n    InvoiceLineId INTEGER  NOT NULL,\n    InvoiceId INTEGER  NOT N
ULL,\n    TrackId INTEGER  NOT NULL,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    Quantity INTEGER  NOT NUL
L,\n    CONSTRAINT PK_InvoiceLine PRIMARY KEY  (InvoiceLineId),\n    FOREIGN KEY (InvoiceId) REFERENCES Inv
oice (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 INTE
GER  NOT NULL,\n    LastName NVARCHAR(20)  NOT NULL,\n    FirstName NVARCHAR(20)  NOT NULL,\n    Title NVAR
CHAR(30),\n    ReportsTo INTEGER,\n    BirthDate DATETIME,\n    HireDate DATETIME,\n    Address NVARCHAR(7
0),\n    City NVARCHAR(40),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVARCHAR(1
0),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60),\n    CONSTRAINT PK_Employee PR
IMARY KEY  (EmployeeId),\n    FOREIGN KEY (ReportsTo) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO A
CTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE MediaType\n(\n    MediaTypeId INTEGER  NOT NULL,\n    Name NVA
RCHAR(120),\n    CONSTRAINT PK_MediaType PRIMARY KEY  (MediaTypeId)\n)\n\nCREATE TABLE Playlist\n(\n    Pla
ylistId 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 AC
TION ON UPDATE NO ACTION,\n    FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION O
N 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 NVARCHA
R(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 D
ELETE 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  (AlbumI
d),\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 \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", 'How many
customers are there', 'SELECT \n    COUNT(*) \nFROM \n    Customer;', 'How many records are in table called
customer', 'SELECT \n    COUNT(*) \nFROM \n    Customer;', 'Show me a list of tables in the SQLite databas
e', 'SELECT \n    Name \nFROM \n    Playlist\nJOIN \n    PlaylistTrack ON Playlist.PlaylistId = PlaylistTra
ck.PlaylistId\nWHERE \n    PlaylistTrack.TrackId IN (\n        SELECT \n            TrackId \n        FROM
\n            InvoiceLine\n        WHERE \n            UnitPrice > 0.99\n    );', 'what are the top 5 count
ries that customers come from?']
403 Generative Language API has not been used in project 124236468554 before or it is disabled. Enable it b
y visiting https://console.developers.google.com/apis/api/generativelanguage.googleapis.com/overview?projec
t=124236468554 then retry. If you enabled this API recently, wait a few minutes for the action to propagate
to our systems and retry. [links {
  description: "Google developers console API activation"
  url: "https://console.developers.google.com/apis/api/generativelanguage.googleapis.com/overview?project=1
24236468554"
}
, reason: "SERVICE_DISABLED"
domain: "googleapis.com"
metadata {
  key: "service"
  value: "generativelanguage.googleapis.com"
}
metadata {
  key: "consumer"
  value: "projects/124236468554"
}
]
```

Out[22]:
```
(None, None, None)
```

More SQL questions

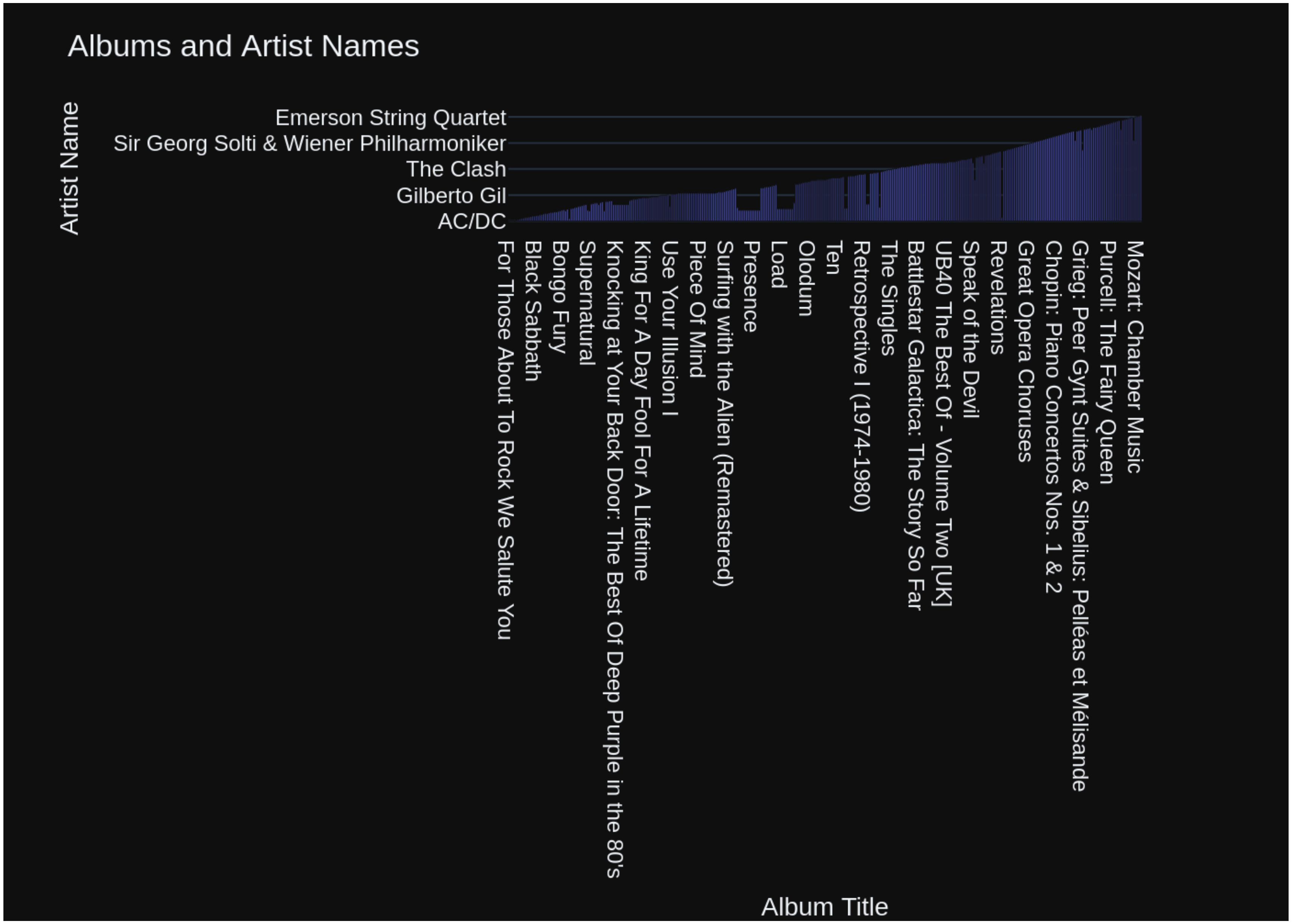
see `sample-sql-queries-sqlite-chinook.ipynb`

In [23]:
```
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 3, updating n_results = 3
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

~~~
["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 Album\n(\n    AlbumId INTEGER  NOT NUL
L,\n    Title NVARCHAR(160)  NOT NULL,\n    ArtistId INTEGER  NOT NULL,\n    CONSTRAINT PK_Album PRIMARY KE
Y  (AlbumId),\n    FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON UPDATE
NO ACTION\n)\n\nCREATE TABLE 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 A
CTION 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 A
CTION ON UPDATE NO ACTION\n)\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  (ArtistI
d)\n)\n\nCREATE INDEX IFK_TrackGenreId ON Track (GenreId)\n\nCREATE INDEX IFK_PlaylistTrackTrackId ON Playl
istTrack (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  (Pla
ylistId)\n)\n\nCREATE TABLE PlaylistTrack\n(\n    PlaylistId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT N
ULL,\n    CONSTRAINT PK_PlaylistTrack PRIMARY KEY  (PlaylistId, TrackId),\n    FOREIGN KEY (PlaylistId) REF
ERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (TrackId) REF
ERENCES 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 sufficie
nt, please generate a valid SQL query without any explanations for the question. \n2. If the provided conte
xt is almost sufficient but requires knowledge of a specific string in a particular column, please generate
an intermediate SQL query to find the distinct strings in that column. Prepend the query with a comment say
ing intermediate_sql \n3. If the provided context is insufficient, please explain why it can't be generate
d. \n4. Please use the most relevant table(s). \n5. If the question has been asked and answered before, ple
ase repeat the answer exactly as it was given before. \n", 'Show me a list of tables in the SQLite databas
e', 'SELECT \n    Name \nFROM \n    Playlist\nJOIN \n    PlaylistTrack ON Playlist.PlaylistId = PlaylistTra
ck.PlaylistId\nWHERE \n    PlaylistTrack.TrackId IN (\n        SELECT \n            TrackId \n        FROM
\n            InvoiceLine\n        WHERE \n            UnitPrice > 0.99\n    );', 'How many records are in
table called customer', 'SELECT \n    COUNT(*) \nFROM \n    Customer;', 'How many customers are there', 'SE
LECT \n    COUNT(*) \nFROM \n    Customer;', '  \n    List all albums and their corresponding artist names
\n']
```sql
SELECT
    Album.Title AS AlbumTitle,
    Artist.Name AS ArtistName
FROM
    Album
JOIN
    Artist ON Album.ArtistId = Artist.ArtistId;
```
~~~

```
SELECT
    Album.Title AS AlbumTitle,
    Artist.Name AS ArtistName
FROM
    Album
JOIN
    Artist ON Album.ArtistId = Artist.ArtistId;
SELECT
    Album.Title AS AlbumTitle,
    Artist.Name AS ArtistName
FROM
    Album
JOIN
    Artist ON Album.ArtistId = Artist.ArtistId;
                                          AlbumTitle  \
0                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...   

                                          ArtistName  
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]
```


Albums and Artist Names
Artist Name
Emerson String Quartet
Sir Georg Solti & Wiener Philharmoniker
The Clash
Gilberto Gil
AC/DC
For Those About To Rock We Salute You
Black Sabbath
Bongo Fury
Supernatural
Knocking at Your Back Door: The Best Of Deep Purple in the 80's
King For A Day Fool For A Lifetime
Use Your Illusion I
Piece Of Mind
Surfing with the Alien (Remastered)
Presence
Load
Olodum
Ten
Retrospective I (1974-1980)
The Singles
Battlestar Galactica: The Story So Far
UB40 The Best Of - Volume Two [UK]
Speak of the Devil
Revelations
Great Opera Choruses
Chopin: Piano Concertos Nos. 1 & 2
Grieg: Peer Gynt Suites & Sibelius: Pelléas et Mélisande
Purcell: The Fairy Queen
Mozart: Chamber Music
Album Title

```
Out[23]: ('SELECT \n    Album.Title AS AlbumTitle, \n    Artist.Name AS ArtistName \nFROM \n    Album \nJOIN \n
         Artist ON Album.ArtistId = Artist.ArtistId;',
                                                    AlbumTitle  \
          0                 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...

                                                    ArtistName
          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': 'AlbumTitle=%{x}<br>ArtistName=%{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},
                         'margin': {'t': 60},
                         'template': '...',
                         'title': {'text': 'Albums and Artist Names'},
                         'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'Album Title'}},
                         'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'Artist Name'}}}
          }))
```

In [24]:

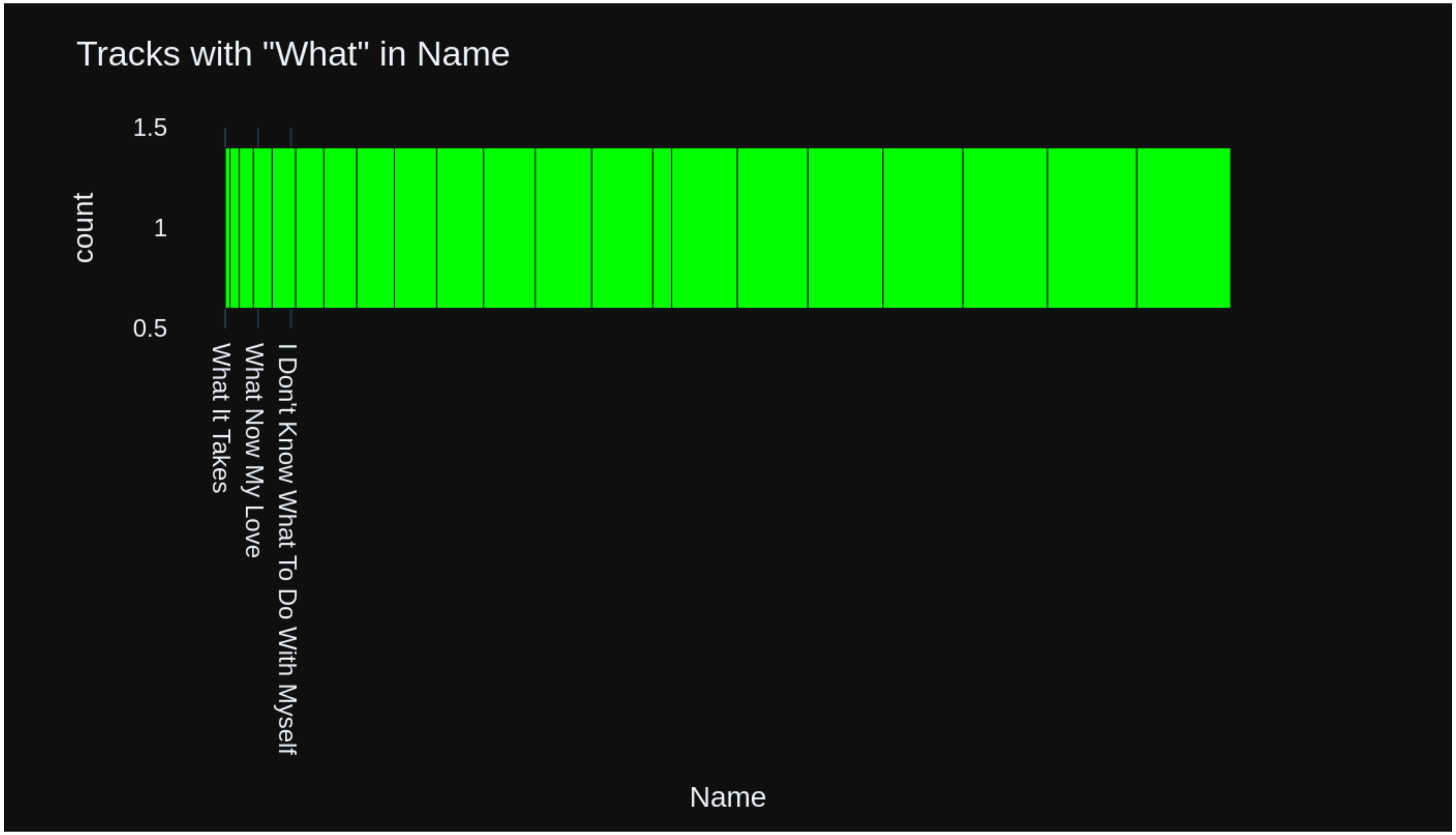
```
question = """
    Find all tracks with a name containing "What" (case-insensitive)
"""

vn.ask(question=question)
```

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

```
["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 NUL
L,\n    Name NVARCHAR(200)  NOT NULL,\n    AlbumId INTEGER,\n    MediaTypeId INTEGER  NOT NULL,\n    GenreI
d INTEGER,\n    Composer NVARCHAR(220),\n    Milliseconds INTEGER  NOT NULL,\n    Bytes INTEGER,\n    UnitP
rice NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (TrackId),\n    FOREIGN KEY (AlbumId) R
EFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (GenreId) REFEREN
CES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (MediaTypeId) REFERENCE
S 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\n
CREATE TABLE PlaylistTrack\n(\n    PlaylistId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    CONST
RAINT PK_PlaylistTrack PRIMARY KEY  (PlaylistId, TrackId),\n    FOREIGN KEY (PlaylistId) REFERENCES Playlis
t (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 (Arti
stId)\n\nCREATE TABLE Album\n(\n    AlbumId INTEGER  NOT NULL,\n    Title NVARCHAR(160)  NOT NULL,\n    Art
istId INTEGER  NOT NULL,\n    CONSTRAINT PK_Album PRIMARY KEY  (AlbumId),\n    FOREIGN KEY (ArtistId) REFER
ENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Playlist\n(\n    P
laylistId 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 \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", '  \n    L
ist all albums and their corresponding artist names  \n', 'SELECT \n    Album.Title AS AlbumTitle, \n    Ar
tist.Name AS ArtistName \nFROM \n    Album \nJOIN \n    Artist ON Album.ArtistId = Artist.ArtistId;', 'Show
me a list of tables in the SQLite database', 'SELECT \n    Name \nFROM \n    Playlist\nJOIN \n    PlaylistT
rack ON Playlist.PlaylistId = PlaylistTrack.PlaylistId\nWHERE \n    PlaylistTrack.TrackId IN (\n        SEL
ECT \n            TrackId \n        FROM \n            InvoiceLine\n        WHERE \n            UnitPrice >
0.99\n    );', 'How many customers are there', 'SELECT \n    COUNT(*) \nFROM \n    Customer;', 'How many re
cords are in table called customer', 'SELECT \n    COUNT(*) \nFROM \n    Customer;', '  \n    Find all trac
ks with a name containing "What" (case-insensitive)\n']
```

```sql
SELECT
    Name
FROM
    Track
WHERE
    LOWER(Name) LIKE '%what%';
```

```
SELECT
    Name
FROM
    Track
WHERE
    LOWER(Name) LIKE '%what%';
SELECT
    Name
FROM
    Track
WHERE
    LOWER(Name) LIKE '%what%';
                                                Name
0                                    What It Takes
1                                     What You Are
2                                Do what cha wanna
3                      What is and Should Never Be
4                                          So What
5                                       What A Day
6                                    What If I Do?
7                                 What Now My Love
8                                      Whatsername
9                Whatever It Is, I Just Can't Stop
10                             Look What You've Done
11                                Get What You Need
12                  What Is And What Should Never Be
13  You're What's Happening (In The World Today)
14                                         So What
15            I Don't Know What To Do With Myself
16                                   What Kate Did
17                       Whatever the Case May Be
18    I Still Haven't Found What I'm Looking for
19    I Still Haven't Found What I'm Looking For
20                  Whatever Gets You Thru the Night
21                            What Is It About Men
```


Tracks with "What" in Name
count
1.5
1
0.5
What It Takes
What Now My Love
I Don't Know What To Do With Myself
Name

```
Out[24]:  ("SELECT \n    Name \nFROM \n    Track\nWHERE \n    LOWER(Name) LIKE '%what%';",
                                                Name
           0                                 What It Takes
           1                                  What You Are
           2                             Do what cha wanna
           3                   What is and Should Never Be
           4                                       So What
           5                                    What A Day
           6                                 What If I Do?
           7                              What Now My Love
           8                                   Whatsername
           9             Whatever It Is, I Just Can't Stop
           10                        Look What You've Done
           11                            Get What You Need
           12             What Is And What Should Never Be
           13  You're What's Happening (In The World Today)
           14                                      So What
           15           I Don't Know What To Do With Myself
           16                                What Kate Did
           17                     Whatever the Case May Be
           18    I Still Haven't Found What I'm Looking for
           19    I Still Haven't Found What I'm Looking For
           20             Whatever Gets You Thru the Night
           21                         What Is It About Men,
           Figure({
               'data': [{'alignmentgroup': 'True',
                         'hovertemplate': 'Name=%{x}<br>count=%{y}<extra></extra>',
                         'legendgroup': '',
                         'marker': {'color': 'rgba(0, 255, 0, 1)', 'pattern': {'shape': ''}},
                         'name': '',
                         'offsetgroup': '',
                         'orientation': 'h',
                         '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([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]),
                         'yaxis': 'y'}],
               'layout': {'barmode': 'relative',
                          'height': 400,
                          'legend': {'tracegroupgap': 0},
                          'margin': {'t': 60},
                          'template': '...',
                          'title': {'text': 'Tracks with "What" in Name'},
                          'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'Name'}},
                          'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'count'}}}
           })
```

In [25]:

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

vn.ask(question=question)
```

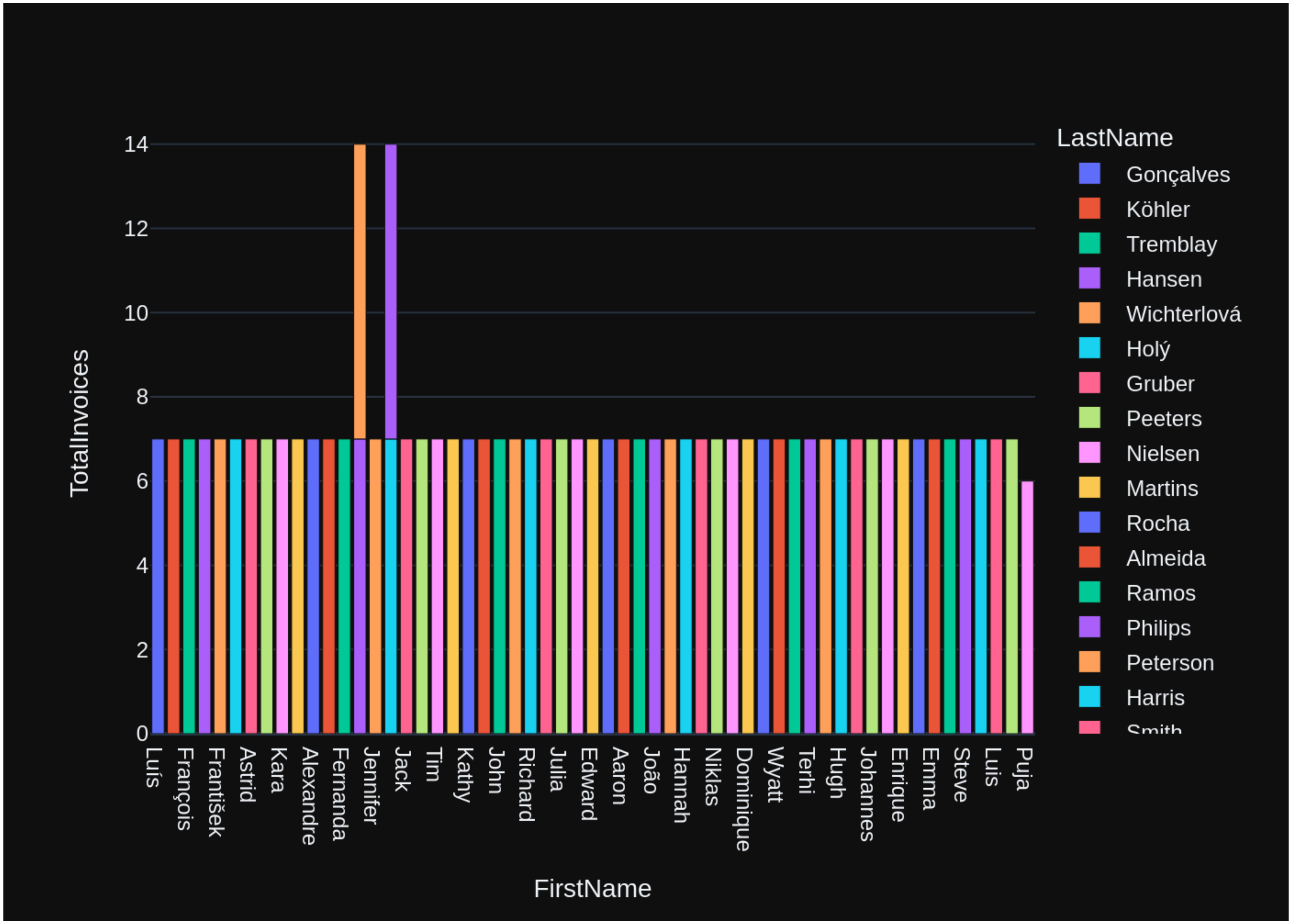
```
Number of requested results 10 is greater than number of elements in index 5, updating n_results = 5
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
["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 INTE
GER  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(4
0),\n    BillingPostalCode NVARCHAR(10),\n    Total NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Invoice PRI
MARY KEY  (InvoiceId),\n    FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO AC
TION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceLineInvoiceId ON InvoiceLine (InvoiceId)\n\nCREATE T
ABLE InvoiceLine\n(\n    InvoiceLineId INTEGER  NOT NULL,\n    InvoiceId INTEGER  NOT NULL,\n    TrackId IN
TEGER  NOT NULL,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    Quantity INTEGER  NOT NULL,\n    CONSTRAINT P
K_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\n
CREATE TABLE Customer\n(\n    CustomerId INTEGER  NOT NULL,\n    FirstName NVARCHAR(40)  NOT NULL,\n    Las
tName NVARCHAR(20)  NOT NULL,\n    Company NVARCHAR(80),\n    Address NVARCHAR(70),\n    City NVARCHAR(4
0),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVARCHAR(10),\n    Phone NVARCHAR(2
4),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60)  NOT NULL,\n    SupportRepId INTEGER,\n    CONSTRAINT PK
_Customer PRIMARY KEY  (CustomerId),\n    FOREIGN KEY (SupportRepId) REFERENCES Employee (EmployeeId) \n\t
\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_CustomerSupportRepId ON Customer (SupportR
epId)\n\nCREATE INDEX IFK_EmployeeReportsTo ON Employee (ReportsTo)\n\nCREATE TABLE Employee\n(\n    Employ
eeId INTEGER  NOT NULL,\n    LastName NVARCHAR(20)  NOT NULL,\n    FirstName NVARCHAR(20)  NOT NULL,\n    T
itle NVARCHAR(30),\n    ReportsTo INTEGER,\n    BirthDate DATETIME,\n    HireDate DATETIME,\n    Address NV
ARCHAR(70),\n    City NVARCHAR(40),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVA
RCHAR(10),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60),\n    CONSTRAINT PK_Empl
oyee PRIMARY KEY  (EmployeeId),\n    FOREIGN KEY (ReportsTo) REFERENCES Employee (EmployeeId) \n\t\tON DELE
TE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Track\n(\n    TrackId INTEGER  NOT NULL,\n    Name NVAR
CHAR(200)  NOT NULL,\n    AlbumId INTEGER,\n    MediaTypeId INTEGER  NOT NULL,\n    GenreId INTEGER,\n    C
omposer NVARCHAR(220),\n    Milliseconds INTEGER  NOT NULL,\n    Bytes INTEGER,\n    UnitPrice NUMERIC(10,
2)  NOT NULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (TrackId),\n    FOREIGN KEY (AlbumId) REFERENCES Album
(AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (GenreId) REFERENCES Genre (Genre
Id) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (MediaTypeId) REFERENCES MediaType (Med
iaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n===Additional Context \n\nIn the chinook da
tabase invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficient, please gen
erate a valid SQL query without any explanations for the question. \n2. If the provided context is almost s
ufficient but requires knowledge of a specific string in a particular column, please generate an intermedia
te SQL query to find the distinct strings in that column. Prepend the query with a comment saying intermedi
ate_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", 'How many customers are there', 'SELECT \n    COUNT(*) \nFROM
\n    Customer;', 'How many records are in table called customer', 'SELECT \n    COUNT(*) \nFROM \n    Cust
omer;', 'Show me a list of tables in the SQLite database', 'SELECT \n    Name \nFROM \n    Playlist\nJOIN
\n    PlaylistTrack ON Playlist.PlaylistId = PlaylistTrack.PlaylistId\nWHERE \n    PlaylistTrack.TrackId IN
```

~~~
(\n        SELECT \n            TrackId \n        FROM \n            InvoiceLine\n        WHERE \n
UnitPrice > 0.99\n    );', '  \n    List all albums and their corresponding artist names  \n', 'SELECT \n
Album.Title AS AlbumTitle, \n    Artist.Name AS ArtistName \nFROM \n    Album \nJOIN \n    Artist ON Album.
ArtistId = Artist.ArtistId;', '  \n    Find all tracks with a name containing "What" (case-insensitive)\n',
"SELECT \n    Name \nFROM \n    Track\nWHERE \n    LOWER(Name) LIKE '%what%';", '  \n    Get the total numb
er of invoices for each customer\n']
```sql
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    COUNT(Invoice.InvoiceId) AS TotalInvoices
FROM
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
GROUP BY
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName;
```
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    COUNT(Invoice.InvoiceId) AS TotalInvoices
FROM
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
GROUP BY
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName;
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    COUNT(Invoice.InvoiceId) AS TotalInvoices
FROM
    Customer
LEFT JOIN
~~~

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

```
36          37       Fynn    Zimmermann               7
37          38    Niklas       Schröder               7
38          39   Camille        Bernard               7
39          40 Dominique       Lefebvre               7
40          41      Marc         Dubois               7
41          42     Wyatt         Girard               7
42          43  Isabelle        Mercier               7
43          44     Terhi     Hämäläinen               7
44          45  Ladislav         Kovács               7
45          46      Hugh       O'Reilly               7
46          47     Lucas        Mancini               7
47          48  Johannes   Van der Berg               7
48          49 Stanisław         Wójcik               7
49          50   Enrique          Muñoz               7
50          51    Joakim      Johansson               7
51          52      Emma          Jones               7
52          53      Phil         Hughes               7
53          54     Steve         Murray               7
54          55      Mark         Taylor               7
55          56     Diego      Gutiérrez               7
56          57      Luis          Rojas               7
57          58     Manoj         Pareek               7
58          59      Puja     Srivastava               6
```


14
12
10
8
6
4
2
0
TotalInvoices
Luís
François
František
Astrid
Kara
Alexandre
Fernanda
Jennifer
Jack
Tim
Kathy
John
Richard
Julia
Edward
Aaron
João
Hannah
Niklas
Dominique
Wyatt
Terhi
Hugh
Johannes
Enrique
Emma
Steve
Luis
Puja
FirstName
LastName
Gonçalves
Köhler
Tremblay
Hansen
Wichterlová
Holý
Gruber
Peeters
Nielsen
Martins
Rocha
Almeida
Ramos
Philips
Peterson
Harris
Smith

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

```
 38          39    Camille       Bernard              7
 39          40  Dominique      Lefebvre              7
 40          41       Marc        Dubois              7
 41          42      Wyatt        Girard              7
 42          43   Isabelle       Mercier              7
 43          44      Terhi    Hämäläinen              7
 44          45   Ladislav        Kovács              7
 45          46       Hugh      O'Reilly              7
 46          47      Lucas       Mancini              7
 47          48   Johannes  Van der Berg              7
 48          49  Stanisław        Wójcik              7
 49          50    Enrique         Muñoz              7
 50          51     Joakim     Johansson              7
 51          52       Emma         Jones              7
 52          53       Phil        Hughes              7
 53          54      Steve        Murray              7
 54          55       Mark        Taylor              7
 55          56      Diego     Gutiérrez              7
 56          57       Luis         Rojas              7
 57          58      Manoj        Pareek              7
 58          59       Puja    Srivastava              6,
 Figure({
     'data': [{'alignmentgroup': 'True',
               'hovertemplate': 'LastName=Gonçalves<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra
>',
               'legendgroup': 'Gonçalves',
               'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
               'name': 'Gonçalves',
               'offsetgroup': 'Gonçalves',
               'orientation': 'v',
               'showlegend': True,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Luís'], dtype=object),
               'xaxis': 'x',
               'y': array([7]),
               'yaxis': 'y'},
              {'alignmentgroup': 'True',
               'hovertemplate': 'LastName=Köhler<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
               'legendgroup': 'Köhler',
               'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
               'name': 'Köhler',
               'offsetgroup': 'Köhler',
               'orientation': 'v',
               'showlegend': True,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Leonie'], dtype=object),
               'xaxis': 'x',
               'y': array([7]),
               'yaxis': 'y'},
              {'alignmentgroup': 'True',
               'hovertemplate': 'LastName=Tremblay<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra
>',
               'legendgroup': 'Tremblay',
               'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
               'name': 'Tremblay',
               'offsetgroup': 'Tremblay',
               'orientation': 'v',
               'showlegend': True,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['François'], dtype=object),
               'xaxis': 'x',
               'y': array([7]),
               'yaxis': 'y'},
              {'alignmentgroup': 'True',
               'hovertemplate': 'LastName=Hansen<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
               'legendgroup': 'Hansen',
               'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
               'name': 'Hansen',
               'offsetgroup': 'Hansen',
               'orientation': 'v',
               'showlegend': True,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Bjørn'], dtype=object),
               'xaxis': 'x',
               'y': array([7]),
               'yaxis': 'y'},
              {'alignmentgroup': 'True',
               'hovertemplate': 'LastName=Wichterlová<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extr
a>',
               'legendgroup': 'Wichterlová',
               'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
               'name': 'Wichterlová',
               'offsetgroup': 'Wichterlová',
               'orientation': 'v',
               'showlegend': True,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['František'], dtype=object),
               'xaxis': 'x',
               'y': array([7]),
               'yaxis': 'y'},
              {'alignmentgroup': 'True',
               'hovertemplate': 'LastName=Holý<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
               'legendgroup': 'Holý',
               'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
               'name': 'Holý',
               'offsetgroup': 'Holý',
               'orientation': 'v',
               'showlegend': True,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Helena'], dtype=object),
               'xaxis': 'x',
               'y': array([7]),
               'yaxis': 'y'},
              {'alignmentgroup': 'True',
               'hovertemplate': 'LastName=Gruber<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
               'legendgroup': 'Gruber',
               'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
               'name': 'Gruber',
               'offsetgroup': 'Gruber',
               'orientation': 'v',
               'showlegend': True,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Astrid'], dtype=object),
               'xaxis': 'x',
               'y': array([7]),
               'yaxis': 'y'},
              {'alignmentgroup': 'True',
               'hovertemplate': 'LastName=Peeters<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
               'legendgroup': 'Peeters',
               'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
               'name': 'Peeters',
               'offsetgroup': 'Peeters',
               'orientation': 'v',
               'showlegend': True,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Daan'], dtype=object),
               'xaxis': 'x',
               'y': array([7]),
               'yaxis': 'y'},
              {'alignmentgroup': 'True',
               'hovertemplate': 'LastName=Nielsen<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
               'legendgroup': 'Nielsen',
               'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
               'name': 'Nielsen',
               'offsetgroup': 'Nielsen',
               'orientation': 'v',
               'showlegend': True,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Kara'], dtype=object),
               'xaxis': 'x',
               'y': array([7]),
               'yaxis': 'y'},
              {'alignmentgroup': 'True',
               'hovertemplate': 'LastName=Martins<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
               'legendgroup': 'Martins',
               'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
               'name': 'Martins',
               'offsetgroup': 'Martins',
               'orientation': 'v',
               'showlegend': True,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Eduardo'], dtype=object),
               'xaxis': 'x',
               'y': array([7]),
               'yaxis': 'y'},
              {'alignmentgroup': 'True',
               'hovertemplate': 'LastName=Rocha<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
               'legendgroup': 'Rocha',
               'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
               'name': 'Rocha',
               'offsetgroup': 'Rocha',
               'orientation': 'v',
               'showlegend': True,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Alexandre'], dtype=object),
               'xaxis': 'x',
               'y': array([7]),
               'yaxis': 'y'},
              {'alignmentgroup': 'True',
               'hovertemplate': 'LastName=Almeida<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
               'legendgroup': 'Almeida',
               'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
               'name': 'Almeida',
               'offsetgroup': 'Almeida',
               'orientation': 'v',
               'showlegend': True,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Roberto'], dtype=object),
               'xaxis': 'x',
               'y': array([7]),
               'yaxis': 'y'},
              {'alignmentgroup': 'True',
               'hovertemplate': 'LastName=Ramos<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
               'legendgroup': 'Ramos',
               'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
               'name': 'Ramos',
               'offsetgroup': 'Ramos',
               'orientation': 'v',
               'showlegend': True,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Fernanda'], dtype=object),
               'xaxis': 'x',
               'y': array([7]),
               'yaxis': 'y'},
              {'alignmentgroup': 'True',
               'hovertemplate': 'LastName=Philips<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
               'legendgroup': 'Philips',
               'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
               'name': 'Philips',
               'offsetgroup': 'Philips',
               'orientation': 'v',
               'showlegend': True,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Mark'], dtype=object),
               'xaxis': 'x',
               'y': array([7]),
               'yaxis': 'y'},
              {'alignmentgroup': 'True',
               'hovertemplate': 'LastName=Peterson<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra
>',
               'legendgroup': 'Peterson',
               'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
               'name': 'Peterson',
               'offsetgroup': 'Peterson',
               'orientation': 'v',
               'showlegend': True,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Jennifer'], dtype=object),
               'xaxis': 'x',
               'y': array([7]),
               'yaxis': 'y'},
              {'alignmentgroup': 'True',
               'hovertemplate': 'LastName=Harris<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
               'legendgroup': 'Harris',
               'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
               'name': 'Harris',
               'offsetgroup': 'Harris',
               'orientation': 'v',
               'showlegend': True,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Frank'], dtype=object),
               'xaxis': 'x',
               'y': array([7]),
               'yaxis': 'y'},
              {'alignmentgroup': 'True',
               'hovertemplate': 'LastName=Smith<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
               'legendgroup': 'Smith',
               'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
               'name': 'Smith',
               'offsetgroup': 'Smith',
               'orientation': 'v',
               'showlegend': True,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Jack'], dtype=object),
               'xaxis': 'x',
               'y': array([7]),
               'yaxis': 'y'},
              {'alignmentgroup': 'True',
               'hovertemplate': 'LastName=Brooks<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
               'legendgroup': 'Brooks',
               'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
               'name': 'Brooks',
               'offsetgroup': 'Brooks',
               'orientation': 'v',
               'showlegend': True,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Michelle'], dtype=object),
               'xaxis': 'x',
               'y': array([7]),
               'yaxis': 'y'},
              {'alignmentgroup': 'True',
               'hovertemplate': 'LastName=Goyer<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
               'legendgroup': 'Goyer',
               'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
               'name': 'Goyer',
               'offsetgroup': 'Goyer',
               'orientation': 'v',
               'showlegend': True,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Tim'], dtype=object),
               'xaxis': 'x',
               'y': array([7]),
               'yaxis': 'y'},
              {'alignmentgroup': 'True',
               'hovertemplate': 'LastName=Miller<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
               'legendgroup': 'Miller',
               'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
               'name': 'Miller',
               'offsetgroup': 'Miller',
               'orientation': 'v',
               'showlegend': True,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Dan'], dtype=object),
               'xaxis': 'x',
               'y': array([7]),
               'yaxis': 'y'},
              {'alignmentgroup': 'True',
               'hovertemplate': 'LastName=Chase<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
               'legendgroup': 'Chase',
               'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
               'name': 'Chase',
               'offsetgroup': 'Chase',
               'orientation': 'v',
               'showlegend': True,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Kathy'], dtype=object),
               'xaxis': 'x',
               'y': array([7]),
               'yaxis': 'y'},
              {'alignmentgroup': 'True',
               'hovertemplate': 'LastName=Leacock<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
               'legendgroup': 'Leacock',
               'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
               'name': 'Leacock',
               'offsetgroup': 'Leacock',
               'orientation': 'v',
               'showlegend': True,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Heather'], dtype=object),
               'xaxis': 'x',
               'y': array([7]),
               'yaxis': 'y'},
              {'alignmentgroup': 'True',
               'hovertemplate': 'LastName=Gordon<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
               'legendgroup': 'Gordon',
               'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
               'name': 'Gordon',
               'offsetgroup': 'Gordon',
               'orientation': 'v',
               'showlegend': True,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['John'], dtype=object),
               'xaxis': 'x',
               'y': array([7]),
               'yaxis': 'y'},
              {'alignmentgroup': 'True',
               'hovertemplate': 'LastName=Ralston<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
               'legendgroup': 'Ralston',
               'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
               'name': 'Ralston',
               'offsetgroup': 'Ralston',
               'orientation': 'v',
               'showlegend': True,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Frank'], dtype=object),
               'xaxis': 'x',
               'y': array([7]),
               'yaxis': 'y'},
              {'alignmentgroup': 'True',
               'hovertemplate': 'LastName=Stevens<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
               'legendgroup': 'Stevens',
               'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
               'name': 'Stevens',
               'offsetgroup': 'Stevens',
               'orientation': 'v',
               'showlegend': True,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Victor'], dtype=object),
               'xaxis': 'x',
               'y': array([7]),
               'yaxis': 'y'},
              {'alignmentgroup': 'True',
               'hovertemplate': 'LastName=Cunningham<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra
```

```
                  >',
                               'legendgroup': 'Cunningham',
                               'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
                               'name': 'Cunningham',
                               'offsetgroup': 'Cunningham',
                               'orientation': 'v',
                               'showlegend': True,
                               'textposition': 'auto',
                               'type': 'bar',
                               'x': array(['Richard'], dtype=object),
                               'xaxis': 'x',
                               'y': array([7]),
                               'yaxis': 'y'},
                              {'alignmentgroup': 'True',
                               'hovertemplate': 'LastName=Gray<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
                               'legendgroup': 'Gray',
                               'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
                               'name': 'Gray',
                               'offsetgroup': 'Gray',
                               'orientation': 'v',
                               'showlegend': True,
                               'textposition': 'auto',
                               'type': 'bar',
                               'x': array(['Patrick'], dtype=object),
                               'xaxis': 'x',
                               'y': array([7]),
                               'yaxis': 'y'},
                              {'alignmentgroup': 'True',
                               'hovertemplate': 'LastName=Barnett<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
                               'legendgroup': 'Barnett',
                               'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
                               'name': 'Barnett',
                               'offsetgroup': 'Barnett',
                               'orientation': 'v',
                               'showlegend': True,
                               'textposition': 'auto',
                               'type': 'bar',
                               'x': array(['Julia'], dtype=object),
                               'xaxis': 'x',
                               'y': array([7]),
                               'yaxis': 'y'},
                              {'alignmentgroup': 'True',
 'hovertemplate': 'LastName=Brown<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
 'legendgroup': 'Brown',
 'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
 'name': 'Brown',
 'offsetgroup': 'Brown',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Robert'], dtype=object),
 'xaxis': 'x',
 'y': array([7]),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'LastName=Francis<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
 'legendgroup': 'Francis',
 'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
 'name': 'Francis',
 'offsetgroup': 'Francis',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Edward'], dtype=object),
 'xaxis': 'x',
 'y': array([7]),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'LastName=Silk<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
 'legendgroup': 'Silk',
 'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
 'name': 'Silk',
 'offsetgroup': 'Silk',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Martha'], dtype=object),
 'xaxis': 'x',
 'y': array([7]),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
                              'hovertemplate': 'LastName=Mitchell<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra
               >',
                              'legendgroup': 'Mitchell',
                              'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
                              'name': 'Mitchell',
                              'offsetgroup': 'Mitchell',
                              'orientation': 'v',
                              'showlegend': True,
                              'textposition': 'auto',
                              'type': 'bar',
                              'x': array(['Aaron'], dtype=object),
                              'xaxis': 'x',
                              'y': array([7]),
                              'yaxis': 'y'},
                             {'alignmentgroup': 'True',
                              'hovertemplate': 'LastName=Sullivan<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra
               >',
                              'legendgroup': 'Sullivan',
                              'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
                              'name': 'Sullivan',
                              'offsetgroup': 'Sullivan',
                              'orientation': 'v',
                              'showlegend': True,
                              'textposition': 'auto',
                              'type': 'bar',
                              'x': array(['Ellie'], dtype=object),
                              'xaxis': 'x',
                              'y': array([7]),
                              'yaxis': 'y'},
                             {'alignmentgroup': 'True',
                              'hovertemplate': 'LastName=Fernandes<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra
               >',
                              'legendgroup': 'Fernandes',
                              'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
                              'name': 'Fernandes',
                              'offsetgroup': 'Fernandes',
                              'orientation': 'v',
                              'showlegend': True,
                              'textposition': 'auto',
                              'type': 'bar',
                              'x': array(['João'], dtype=object),
                              'xaxis': 'x',
                              'y': array([7]),
                              'yaxis': 'y'},
                             {'alignmentgroup': 'True',
                              'hovertemplate': 'LastName=Sampaio<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
                              'legendgroup': 'Sampaio',
                              'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
                              'name': 'Sampaio',
                              'offsetgroup': 'Sampaio',
                              'orientation': 'v',
                              'showlegend': True,
                              'textposition': 'auto',
                              'type': 'bar',
                              'x': array(['Madalena'], dtype=object),
                              'xaxis': 'x',
                              'y': array([7]),
                              'yaxis': 'y'},
                             {'alignmentgroup': 'True',
                              'hovertemplate': 'LastName=Schneider<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra
               >',
                              'legendgroup': 'Schneider',
                              'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
                              'name': 'Schneider',
                              'offsetgroup': 'Schneider',
                              'orientation': 'v',
                              'showlegend': True,
                              'textposition': 'auto',
                              'type': 'bar',
                              'x': array(['Hannah'], dtype=object),
                              'xaxis': 'x',
                              'y': array([7]),
                              'yaxis': 'y'},
                             {'alignmentgroup': 'True',
                              'hovertemplate': 'LastName=Zimmermann<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra
               >',
                              'legendgroup': 'Zimmermann',
                              'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
                              'name': 'Zimmermann',
                              'offsetgroup': 'Zimmermann',
                              'orientation': 'v',
                              'showlegend': True,
                              'textposition': 'auto',
                              'type': 'bar',
                              'x': array(['Fynn'], dtype=object),
                              'xaxis': 'x',
                              'y': array([7]),
                              'yaxis': 'y'},
                             {'alignmentgroup': 'True',
                              'hovertemplate': 'LastName=Schröder<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra
               >',
                              'legendgroup': 'Schröder',
                              'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
                              'name': 'Schröder',
                              'offsetgroup': 'Schröder',
                              'orientation': 'v',
                              'showlegend': True,
                              'textposition': 'auto',
                              'type': 'bar',
                              'x': array(['Niklas'], dtype=object),
                              'xaxis': 'x',
                              'y': array([7]),
                              'yaxis': 'y'},
                             {'alignmentgroup': 'True',
                              'hovertemplate': 'LastName=Bernard<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
                              'legendgroup': 'Bernard',
                              'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
                              'name': 'Bernard',
                              'offsetgroup': 'Bernard',
                              'orientation': 'v',
                              'showlegend': True,
                              'textposition': 'auto',
                              'type': 'bar',
                              'x': array(['Camille'], dtype=object),
                              'xaxis': 'x',
                              'y': array([7]),
                              'yaxis': 'y'},
                             {'alignmentgroup': 'True',
                              'hovertemplate': 'LastName=Lefebvre<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra
               >',
                              'legendgroup': 'Lefebvre',
                              'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
                              'name': 'Lefebvre',
                              'offsetgroup': 'Lefebvre',
                              'orientation': 'v',
                              'showlegend': True,
                              'textposition': 'auto',
                              'type': 'bar',
                              'x': array(['Dominique'], dtype=object),
                              'xaxis': 'x',
                              'y': array([7]),
                              'yaxis': 'y'},
                             {'alignmentgroup': 'True',
                              'hovertemplate': 'LastName=Dubois<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
                              'legendgroup': 'Dubois',
                              'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
                              'name': 'Dubois',
                              'offsetgroup': 'Dubois',
                              'orientation': 'v',
                              'showlegend': True,
                              'textposition': 'auto',
                              'type': 'bar',
                              'x': array(['Marc'], dtype=object),
                              'xaxis': 'x',
                              'y': array([7]),
                              'yaxis': 'y'},
                             {'alignmentgroup': 'True',
                              'hovertemplate': 'LastName=Girard<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
                              'legendgroup': 'Girard',
                              'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
                              'name': 'Girard',
                              'offsetgroup': 'Girard',
                              'orientation': 'v',
                              'showlegend': True,
                              'textposition': 'auto',
                              'type': 'bar',
                              'x': array(['Wyatt'], dtype=object),
                              'xaxis': 'x',
                              'y': array([7]),
                              'yaxis': 'y'},
                             {'alignmentgroup': 'True',
                              'hovertemplate': 'LastName=Mercier<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
                              'legendgroup': 'Mercier',
                              'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
                              'name': 'Mercier',
                              'offsetgroup': 'Mercier',
                              'orientation': 'v',
                              'showlegend': True,
                               'textposition': 'auto',
                               'type': 'bar',
                               'x': array(['Isabelle'], dtype=object),
                               'xaxis': 'x',
                               'y': array([7]),
                               'yaxis': 'y'},
                              {'alignmentgroup': 'True',
                               'hovertemplate': 'LastName=Hämäläinen<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra
>',
                               'legendgroup': 'Hämäläinen',
                               'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
                               'name': 'Hämäläinen',
                               'offsetgroup': 'Hämäläinen',
                               'orientation': 'v',
                               'showlegend': True,
                               'textposition': 'auto',
                               'type': 'bar',
                               'x': array(['Terhi'], dtype=object),
                               'xaxis': 'x',
                               'y': array([7]),
                               'yaxis': 'y'},
                              {'alignmentgroup': 'True',
                               'hovertemplate': 'LastName=Kovács<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
                               'legendgroup': 'Kovács',
                               'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
                               'name': 'Kovács',
                               'offsetgroup': 'Kovács',
                               'orientation': 'v',
                               'showlegend': True,
                               'textposition': 'auto',
                               'type': 'bar',
                               'x': array(['Ladislav'], dtype=object),
                               'xaxis': 'x',
                               'y': array([7]),
                               'yaxis': 'y'},
                              {'alignmentgroup': 'True',
                               'hovertemplate': "LastName=O'Reilly<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra
>",
                               'legendgroup': "O'Reilly",
                               'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
                               'name': "O'Reilly",
                               'offsetgroup': "O'Reilly",
                               'orientation': 'v',
                               'showlegend': True,
                               'textposition': 'auto',
                               'type': 'bar',
                               'x': array(['Hugh'], dtype=object),
                               'xaxis': 'x',
                               'y': array([7]),
                               'yaxis': 'y'},
                              {'alignmentgroup': 'True',
                               'hovertemplate': 'LastName=Mancini<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
                               'legendgroup': 'Mancini',
                               'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
                               'name': 'Mancini',
                               'offsetgroup': 'Mancini',
                               'orientation': 'v',
                               'showlegend': True,
                               'textposition': 'auto',
                               'type': 'bar',
                               'x': array(['Lucas'], dtype=object),
                               'xaxis': 'x',
                               'y': array([7]),
                               'yaxis': 'y'},
                              {'alignmentgroup': 'True',
                               'hovertemplate': 'LastName=Van der Berg<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></ext
                ra>',
                               'legendgroup': 'Van der Berg',
                               'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
                               'name': 'Van der Berg',
                               'offsetgroup': 'Van der Berg',
                               'orientation': 'v',
                               'showlegend': True,
                               'textposition': 'auto',
                               'type': 'bar',
                               'x': array(['Johannes'], dtype=object),
                               'xaxis': 'x',
                               'y': array([7]),
                               'yaxis': 'y'},
                              {'alignmentgroup': 'True',
                               'hovertemplate': 'LastName=Wójcik<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
                               'legendgroup': 'Wójcik',
                               'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
                               'name': 'Wójcik',
                               'offsetgroup': 'Wójcik',
                               'orientation': 'v',
                               'showlegend': True,
                               'textposition': 'auto',
                               'type': 'bar',
                               'x': array(['Stanisław'], dtype=object),
                               'xaxis': 'x',
                               'y': array([7]),
                               'yaxis': 'y'},
                              {'alignmentgroup': 'True',
                               'hovertemplate': 'LastName=Muñoz<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
                               'legendgroup': 'Muñoz',
                               'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
                               'name': 'Muñoz',
                               'offsetgroup': 'Muñoz',
                               'orientation': 'v',
                               'showlegend': True,
                               'textposition': 'auto',
                               'type': 'bar',
                               'x': array(['Enrique'], dtype=object),
                               'xaxis': 'x',
                               'y': array([7]),
                               'yaxis': 'y'},
                              {'alignmentgroup': 'True',
                               'hovertemplate': 'LastName=Johansson<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra
                >',
                               'legendgroup': 'Johansson',
                               'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
                               'name': 'Johansson',
                               'offsetgroup': 'Johansson',
                               'orientation': 'v',
                               'showlegend': True,
                               'textposition': 'auto',
                               'type': 'bar',
                               'x': array(['Joakim'], dtype=object),
                               'xaxis': 'x',
                               'y': array([7]),
                               'yaxis': 'y'},
                              {'alignmentgroup': 'True',
                               'hovertemplate': 'LastName=Jones<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
                               'legendgroup': 'Jones',
                               'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
                               'name': 'Jones',
                               'offsetgroup': 'Jones',
                               'orientation': 'v',
                               'showlegend': True,
                               'textposition': 'auto',
                               'type': 'bar',
                               'x': array(['Emma'], dtype=object),
                               'xaxis': 'x',
                               'y': array([7]),
                               'yaxis': 'y'},
                              {'alignmentgroup': 'True',
                               'hovertemplate': 'LastName=Hughes<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
                               'legendgroup': 'Hughes',
                               'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
                               'name': 'Hughes',
                               'offsetgroup': 'Hughes',
                               'orientation': 'v',
                               'showlegend': True,
                               'textposition': 'auto',
                               'type': 'bar',
                               'x': array(['Phil'], dtype=object),
                               'xaxis': 'x',
                               'y': array([7]),
                               'yaxis': 'y'},
                              {'alignmentgroup': 'True',
                               'hovertemplate': 'LastName=Murray<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
                               'legendgroup': 'Murray',
                               'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
                               'name': 'Murray',
                               'offsetgroup': 'Murray',
                               'orientation': 'v',
                               'showlegend': True,
                               'textposition': 'auto',
                               'type': 'bar',
                               'x': array(['Steve'], dtype=object),
                               'xaxis': 'x',
                               'y': array([7]),
                               'yaxis': 'y'},
                              {'alignmentgroup': 'True',
                               'hovertemplate': 'LastName=Taylor<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
                               'legendgroup': 'Taylor',
                               'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
                               'name': 'Taylor',
                               'offsetgroup': 'Taylor',
                               'orientation': 'v',
                               'showlegend': True,
                               'textposition': 'auto',
                               'type': 'bar',
                               'x': array(['Mark'], dtype=object),
                               'xaxis': 'x',
                               'y': array([7]),
                               'yaxis': 'y'},
                              {'alignmentgroup': 'True',
                               'hovertemplate': 'LastName=Gutiérrez<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra
 >',
                               'legendgroup': 'Gutiérrez',
                               'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
                               'name': 'Gutiérrez',
                               'offsetgroup': 'Gutiérrez',
                               'orientation': 'v',
                               'showlegend': True,
                               'textposition': 'auto',
                               'type': 'bar',
                               'x': array(['Diego'], dtype=object),
                               'xaxis': 'x',
                               'y': array([7]),
                               'yaxis': 'y'},
                              {'alignmentgroup': 'True',
                               'hovertemplate': 'LastName=Rojas<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
                               'legendgroup': 'Rojas',
                               'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
                               'name': 'Rojas',
                               'offsetgroup': 'Rojas',
                               'orientation': 'v',
                               'showlegend': True,
                               'textposition': 'auto',
                               'type': 'bar',
                               'x': array(['Luis'], dtype=object),
                               'xaxis': 'x',
                               'y': array([7]),
                               'yaxis': 'y'},
                              {'alignmentgroup': 'True',
                               'hovertemplate': 'LastName=Pareek<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
                               'legendgroup': 'Pareek',
                               'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
                               'name': 'Pareek',
                               'offsetgroup': 'Pareek',
                               'orientation': 'v',
                               'showlegend': True,
                               'textposition': 'auto',
                               'type': 'bar',
                               'x': array(['Manoj'], dtype=object),
                               'xaxis': 'x',
                               'y': array([7]),
                               'yaxis': 'y'},
                              {'alignmentgroup': 'True',
                               'hovertemplate': 'LastName=Srivastava<br>FirstName=%{x}<br>TotalInvoices=%{y}<extra></extra
                >',
                               'legendgroup': 'Srivastava',
                               'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
                               'name': 'Srivastava',
                               'offsetgroup': 'Srivastava',
                               'orientation': 'v',
                               'showlegend': True,
                               'textposition': 'auto',
                               'type': 'bar',
                               'x': array(['Puja'], dtype=object),
                               'xaxis': 'x',
                               'y': array([6]),
                               'yaxis': 'y'}],
                     'layout': {'barmode': 'relative',
                                'legend': {'title': {'text': 'LastName'}, 'tracegroupgap': 0},
                                'margin': {'t': 60},
                                'template': '...',
                                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'FirstName'}},
                                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'TotalInvoices'}}}
                 }))
```

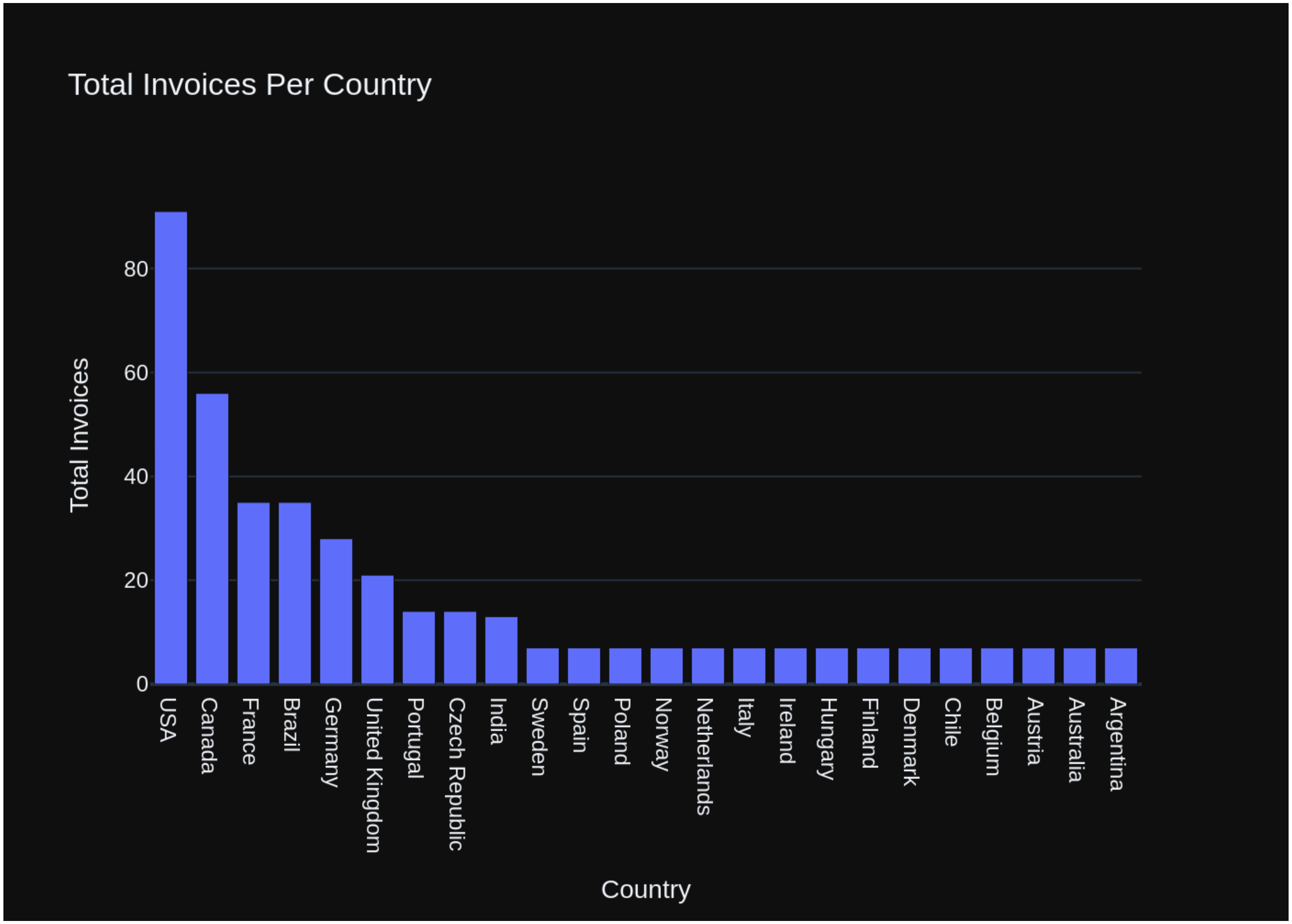
In [26]:
```
question = """
    Find the total number of invoices per country:
"""

vn.ask(question=question)
```

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

```
["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    Invoice
Date 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_InvoiceLineInvoi
ceId ON InvoiceLine (InvoiceId)\n\nCREATE INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)\n\nCREATE TAB
LE InvoiceLine\n(\n    InvoiceLineId INTEGER  NOT NULL,\n    InvoiceId INTEGER  NOT NULL,\n    TrackId INTE
GER  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 D
ELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceLineTrackId ON InvoiceLine (TrackId)\n\nC
REATE TABLE Customer\n(\n    CustomerId INTEGER  NOT NULL,\n    FirstName NVARCHAR(40)  NOT NULL,\n    Last
Name NVARCHAR(20)  NOT NULL,\n    Company NVARCHAR(80),\n    Address NVARCHAR(70),\n    City NVARCHAR(4
0),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVARCHAR(10),\n    Phone NVARCHAR(2
4),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60)  NOT NULL,\n    SupportRepId INTEGER,\n    CONSTRAINT PK
_Customer PRIMARY KEY  (CustomerId),\n    FOREIGN KEY (SupportRepId) REFERENCES Employee (EmployeeId) \n\t
\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Employee\n(\n    EmployeeId INTEGER  NOT NUL
L,\n    LastName NVARCHAR(20)  NOT NULL,\n    FirstName NVARCHAR(20)  NOT NULL,\n    Title NVARCHAR(30),\n
ReportsTo INTEGER,\n    BirthDate DATETIME,\n    HireDate DATETIME,\n    Address NVARCHAR(70),\n    City NV
ARCHAR(40),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVARCHAR(10),\n    Phone NV
ARCHAR(24),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60),\n    CONSTRAINT PK_Employee PRIMARY KEY  (Emplo
yeeId),\n    FOREIGN KEY (ReportsTo) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE N
O ACTION\n)\n\nCREATE 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 A
CTION 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 A
CTION 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 NUL
L,\n    CONSTRAINT PK_Album PRIMARY KEY  (AlbumId),\n    FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistI
d) \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 sufficien
t but requires knowledge of a specific string in a particular column, please generate an intermediate SQL q
uery 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 e
xactly as it was given before. \n", '  \n    Get the total number of invoices for each customer\n', 'SELECT
\n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName, \n    COUNT(Invoice.InvoiceI
```

~~~
d) AS TotalInvoices \nFROM \n    Customer \nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.Custom
erId \nGROUP BY \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName;', 'How many r
ecords are in table called customer', 'SELECT \n    COUNT(*) \nFROM \n    Customer;', 'How many customers a
re there', 'SELECT \n    COUNT(*) \nFROM \n    Customer;', 'Show me a list of tables in the SQLite databas
e', 'SELECT \n    Name \nFROM \n    Playlist\nJOIN \n    PlaylistTrack ON Playlist.PlaylistId = PlaylistTra
ck.PlaylistId\nWHERE \n    PlaylistTrack.TrackId IN (\n        SELECT \n            TrackId \n        FROM
\n            InvoiceLine\n        WHERE \n            UnitPrice > 0.99\n    );', '  \n    List all albums
and their corresponding artist names  \n', 'SELECT \n    Album.Title AS AlbumTitle, \n    Artist.Name AS Ar
tistName \nFROM \n    Album \nJOIN \n    Artist ON Album.ArtistId = Artist.ArtistId;', '  \n    Find all tr
acks with a name containing "What" (case-insensitive)\n', "SELECT \n    Name \nFROM \n    Track\nWHERE \n
LOWER(Name) LIKE '%what%';", '  \n    Find the total number of invoices per country:\n']
```sql
SELECT
    Invoice.BillingCountry,
    COUNT(Invoice.InvoiceId) AS TotalInvoices
FROM
    Invoice
GROUP BY
    Invoice.BillingCountry
ORDER BY
    TotalInvoices DESC;
```
SELECT
    Invoice.BillingCountry,
    COUNT(Invoice.InvoiceId) AS TotalInvoices
FROM
    Invoice
GROUP BY
    Invoice.BillingCountry
ORDER BY
    TotalInvoices DESC;
SELECT
    Invoice.BillingCountry,
    COUNT(Invoice.InvoiceId) AS TotalInvoices
FROM
    Invoice
GROUP BY
    Invoice.BillingCountry
ORDER BY
    TotalInvoices DESC;
    BillingCountry  TotalInvoices
0              USA             91
~~~

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


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

```
Out[26]: ('SELECT \n    Invoice.BillingCountry, \n    COUNT(Invoice.InvoiceId) AS TotalInvoices \nFROM \n    Invoic
         e \nGROUP BY \n    Invoice.BillingCountry \nORDER BY \n    TotalInvoices DESC;',
              BillingCountry  TotalInvoices
          0              USA             91
          1           Canada             56
          2           France             35
          3           Brazil             35
          4          Germany             28
          5   United Kingdom             21
          6         Portugal             14
          7   Czech Republic             14
          8            India             13
          9           Sweden              7
          10           Spain              7
          11          Poland              7
          12          Norway              7
          13     Netherlands              7
          14           Italy              7
          15         Ireland              7
          16         Hungary              7
          17         Finland              7
          18         Denmark              7
          19           Chile              7
          20         Belgium              7
          21         Austria              7
          22       Australia              7
          23       Argentina              7,
          Figure({
              'data': [{'name': 'Total Invoices',
                        'type': 'bar',
                        'x': array(['USA', 'Canada', 'France', 'Brazil', 'Germany', 'United Kingdom',
                                    'Portugal', 'Czech Republic', 'India', 'Sweden', 'Spain', 'Poland',
                                    'Norway', 'Netherlands', 'Italy', 'Ireland', 'Hungary', 'Finland',
                                    'Denmark', 'Chile', 'Belgium', 'Austria', 'Australia', 'Argentina'],
                                   dtype=object),
                        'y': array([91, 56, 35, 35, 28, 21, 14, 14, 13,  7,  7,  7,  7,  7,  7,  7,  7,  7,
                                     7,  7,  7,  7,  7,  7])}],
              'layout': {'template': '...',
                         'title': {'text': 'Total Invoices Per Country'},
                         'xaxis': {'title': {'text': 'Country'}},
                         'yaxis': {'title': {'text': 'Total Invoices'}}}
          }))
```

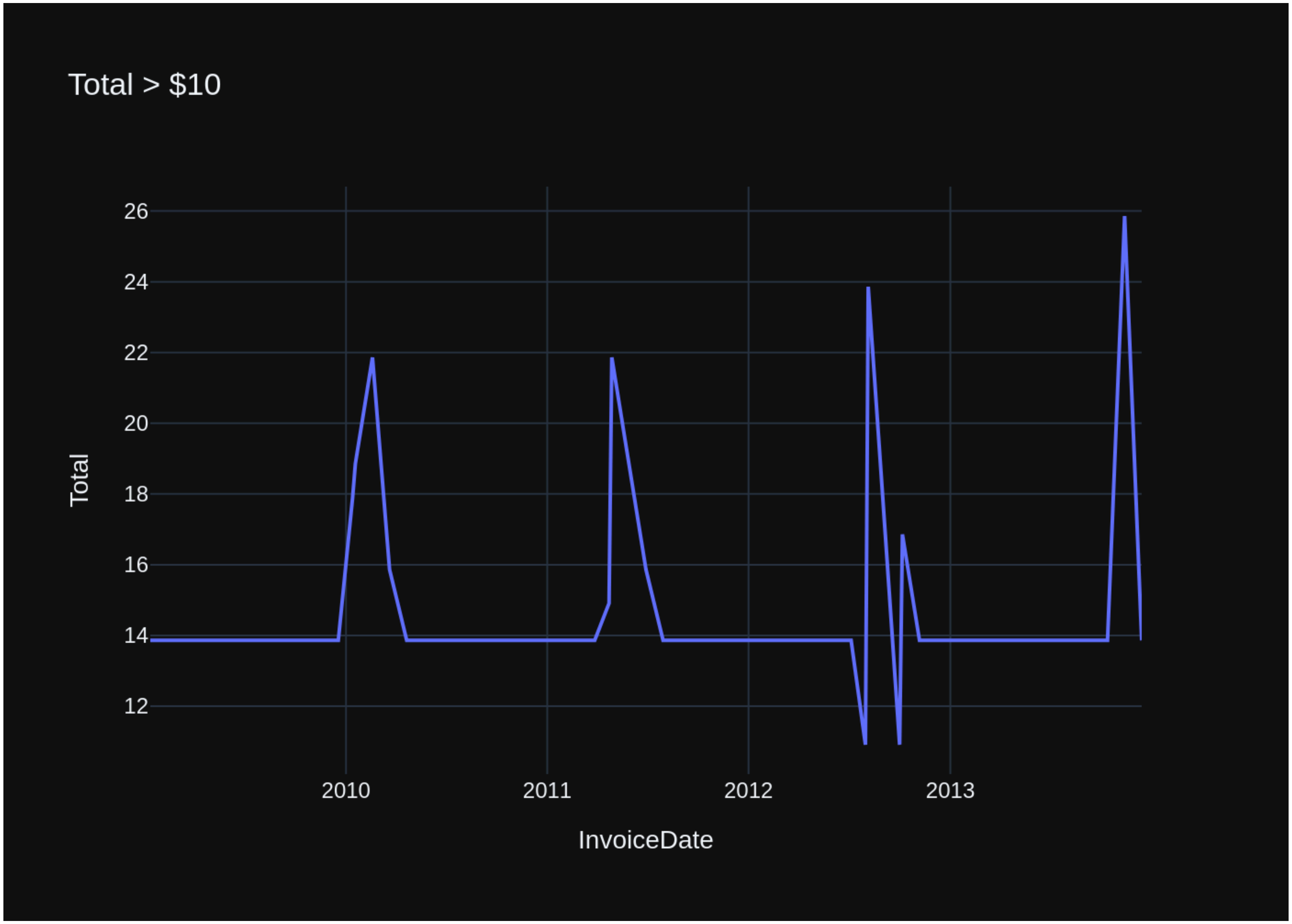
```
In [27]: question = """
             List all invoices with a total exceeding $10:
         """

         vn.ask(question=question)
```

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

```
["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    CO
NSTRAINT PK_InvoiceLine PRIMARY KEY  (InvoiceLineId),\n    FOREIGN KEY (InvoiceId) REFERENCES Invoice (Invo
iceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (TrackId) REFERENCES Track (TrackId)
\n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceLineInvoiceId ON InvoiceLine (I
nvoiceId)\n\nCREATE TABLE Invoice\n(\n    InvoiceId INTEGER  NOT NULL,\n    CustomerId INTEGER  NOT NULL,\n
InvoiceDate DATETIME  NOT NULL,\n    BillingAddress NVARCHAR(70),\n    BillingCity NVARCHAR(40),\n    Billi
ngState NVARCHAR(40),\n    BillingCountry NVARCHAR(40),\n    BillingPostalCode NVARCHAR(10),\n    Total NUM
ERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Invoice PRIMARY KEY  (InvoiceId),\n    FOREIGN KEY (CustomerId) RE
FERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_Invoice
CustomerId ON Invoice (CustomerId)\n\nCREATE INDEX IFK_InvoiceLineTrackId ON InvoiceLine (TrackId)\n\nCREAT
E 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 ACT
ION,\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    Customer
Id INTEGER  NOT NULL,\n    FirstName NVARCHAR(40)  NOT NULL,\n    LastName NVARCHAR(20)  NOT NULL,\n    Com
pany 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 NVAR
CHAR(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 DATETIM
E,\n    HireDate DATETIME,\n    Address NVARCHAR(70),\n    City NVARCHAR(40),\n    State NVARCHAR(40),\n
Country NVARCHAR(40),\n    PostalCode NVARCHAR(10),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(24),\n    Em
ail NVARCHAR(60),\n    CONSTRAINT PK_Employee PRIMARY KEY  (EmployeeId),\n    FOREIGN KEY (ReportsTo) REFER
ENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_CustomerSu
pportRepId 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 que
ry without any explanations for the question. \n2. If the provided context is almost sufficient but require
s 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 p
rovided context is insufficient, please explain why it can't be generated. \n4. Please use the most relevan
t table(s). \n5. If the question has been asked and answered before, please repeat the answer exactly as it
was given before. \n", '  \n    Find the total number of invoices per country:\n', 'SELECT \n    Invoice.Bi
llingCountry, \n    COUNT(Invoice.InvoiceId) AS TotalInvoices \nFROM \n    Invoice \nGROUP BY \n    Invoic
e.BillingCountry \nORDER BY \n    TotalInvoices DESC;', '  \n    Get the total number of invoices for each
customer\n', 'SELECT \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName, \n    CO
```

~~~
UNT(Invoice.InvoiceId) AS TotalInvoices \nFROM \n    Customer \nLEFT JOIN \n    Invoice ON Customer.Custome
rId = Invoice.CustomerId \nGROUP BY \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.Las
tName;', 'Show me a list of tables in the SQLite database', 'SELECT \n    Name \nFROM \n    Playlist\nJOIN
\n    PlaylistTrack ON Playlist.PlaylistId = PlaylistTrack.PlaylistId\nWHERE \n    PlaylistTrack.TrackId IN
(\n        SELECT \n            TrackId \n        FROM \n            InvoiceLine\n        WHERE \n
UnitPrice > 0.99\n    );', 'How many records are in table called customer', 'SELECT \n    COUNT(*) \nFROM
\n    Customer;', 'How many customers are there', 'SELECT \n    COUNT(*) \nFROM \n    Customer;', '  \n
List all albums and their corresponding artist names  \n', 'SELECT \n    Album.Title AS AlbumTitle, \n    A
rtist.Name AS ArtistName \nFROM \n    Album \nJOIN \n    Artist ON Album.ArtistId = Artist.ArtistId;', '
\n    Find all tracks with a name containing "What" (case-insensitive)\n', "SELECT \n    Name \nFROM \n
Track\nWHERE \n    LOWER(Name) LIKE '%what%';", '  \n    List all invoices with a total exceeding $10:\n']
```sql
SELECT
    Invoice.InvoiceId,
    Invoice.InvoiceDate,
    Invoice.Total
FROM
    Invoice
WHERE
    Invoice.Total > 10;
```
SELECT
    Invoice.InvoiceId,
    Invoice.InvoiceDate,
    Invoice.Total
FROM
    Invoice
WHERE
    Invoice.Total > 10;
SELECT
    Invoice.InvoiceId,
    Invoice.InvoiceDate,
    Invoice.Total
FROM
    Invoice
WHERE
    Invoice.Total > 10;
    InvoiceId          InvoiceDate  Total
0           5  2009-01-11 00:00:00  13.86
1          12  2009-02-11 00:00:00  13.86
2          19  2009-03-14 00:00:00  13.86
3          26  2009-04-14 00:00:00  13.86
~~~

```
4           33  2009-05-15 00:00:00  13.86
..         ...                  ...    ...
59         383  2013-08-12 00:00:00  13.86
60         390  2013-09-12 00:00:00  13.86
61         397  2013-10-13 00:00:00  13.86
62         404  2013-11-13 00:00:00  25.86
63         411  2013-12-14 00:00:00  13.86

[64 rows x 3 columns]
```


Total > $10
Total
26
24
22
20
18
16
14
12
2010
2011
2012
2013
InvoiceDate

```
Out[27]: ('SELECT \n    Invoice.InvoiceId, \n    Invoice.InvoiceDate, \n    Invoice.Total \nFROM \n    Invoice \nWH
         ERE \n    Invoice.Total > 10;',
              InvoiceId          InvoiceDate  Total
          0           5  2009-01-11 00:00:00  13.86
          1          12  2009-02-11 00:00:00  13.86
          2          19  2009-03-14 00:00:00  13.86
          3          26  2009-04-14 00:00:00  13.86
          4          33  2009-05-15 00:00:00  13.86
          ..        ...                  ...    ...
          59        383  2013-08-12 00:00:00  13.86
          60        390  2013-09-12 00:00:00  13.86
          61        397  2013-10-13 00:00:00  13.86
          62        404  2013-11-13 00:00:00  25.86
          63        411  2013-12-14 00:00:00  13.86

          [64 rows x 3 columns],
          Figure({
              'data': [{'hovertemplate': 'InvoiceDate=%{x}<br>Total=%{y}<extra></extra>',
                        'legendgroup': '',
                        'line': {'color': '#636efa', 'dash': 'solid'},
                        'marker': {'symbol': 'circle'},
                        'mode': 'lines',
                        'name': '',
                        'orientation': 'v',
                        'showlegend': False,
                        'type': 'scatter',
                        'x': array(['2009-01-11 00:00:00', '2009-02-11 00:00:00', '2009-03-14 00:00:00',
                                    '2009-04-14 00:00:00', '2009-05-15 00:00:00', '2009-06-15 00:00:00',
                                    '2009-07-16 00:00:00', '2009-08-16 00:00:00', '2009-09-16 00:00:00',
                                    '2009-10-17 00:00:00', '2009-11-17 00:00:00', '2009-12-18 00:00:00',
                                    '2010-01-13 00:00:00', '2010-01-18 00:00:00', '2010-02-18 00:00:00',
                                    '2010-03-21 00:00:00', '2010-04-21 00:00:00', '2010-05-22 00:00:00',
                                    '2010-06-22 00:00:00', '2010-07-23 00:00:00', '2010-08-23 00:00:00',
                                    '2010-09-23 00:00:00', '2010-10-24 00:00:00', '2010-11-24 00:00:00',
                                    '2010-12-25 00:00:00', '2011-01-25 00:00:00', '2011-02-25 00:00:00',
                                    '2011-03-28 00:00:00', '2011-04-23 00:00:00', '2011-04-28 00:00:00',
                                    '2011-05-29 00:00:00', '2011-06-29 00:00:00', '2011-07-30 00:00:00',
                                    '2011-08-30 00:00:00', '2011-09-30 00:00:00', '2011-10-31 00:00:00',
                                    '2011-12-01 00:00:00', '2012-01-01 00:00:00', '2012-02-01 00:00:00',
                                    '2012-03-03 00:00:00', '2012-04-03 00:00:00', '2012-05-04 00:00:00',
                                    '2012-06-04 00:00:00', '2012-07-05 00:00:00', '2012-07-31 00:00:00',
                                    '2012-08-05 00:00:00', '2012-09-05 00:00:00', '2012-09-28 00:00:00',
                                '2012-10-01 00:00:00', '2012-10-06 00:00:00', '2012-11-06 00:00:00',
                                '2012-12-07 00:00:00', '2013-01-07 00:00:00', '2013-02-07 00:00:00',
                                '2013-03-10 00:00:00', '2013-04-10 00:00:00', '2013-05-11 00:00:00',
                                '2013-06-11 00:00:00', '2013-07-12 00:00:00', '2013-08-12 00:00:00',
                                '2013-09-12 00:00:00', '2013-10-13 00:00:00', '2013-11-13 00:00:00',
                                '2013-12-14 00:00:00'], dtype=object),
                  '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': {'legend': {'tracegroupgap': 0},
                   'template': '...',
                   'title': {'text': 'Total > $10'},
                   'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'InvoiceDate'}},
                   'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'Total'}}}
    }))
```

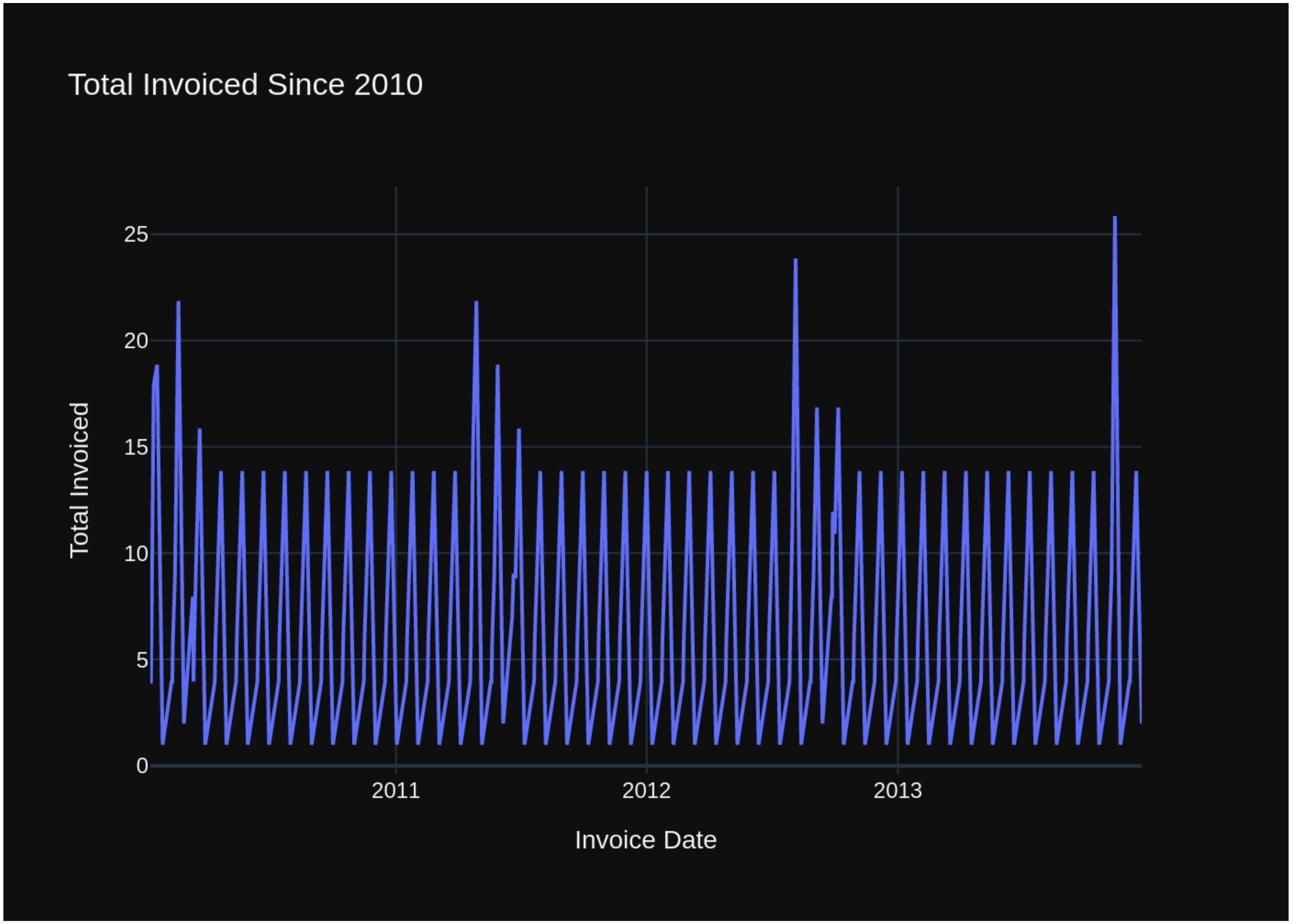
In [28]:
```python
question = """
    Find all invoices since 2010 and the total amount invoiced:
"""

vn.ask(question=question)
```

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

```
["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    Invoice
Date 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    Uni
tPrice 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 INDEX IFK_InvoiceLineInvoiceId ON InvoiceLine (InvoiceId)\n\nCREATE INDEX IFK_Invoic
eCustomerId ON Invoice (CustomerId)\n\nCREATE INDEX IFK_InvoiceLineTrackId ON InvoiceLine (TrackId)\n\nCREA
TE TABLE Customer\n(\n    CustomerId INTEGER  NOT NULL,\n    FirstName NVARCHAR(40)  NOT NULL,\n    LastNam
e 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    LastNa
me NVARCHAR(20)  NOT NULL,\n    FirstName NVARCHAR(20)  NOT NULL,\n    Title NVARCHAR(30),\n    ReportsTo I
NTEGER,\n    BirthDate DATETIME,\n    HireDate DATETIME,\n    Address NVARCHAR(70),\n    City NVARCHAR(4
0),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVARCHAR(10),\n    Phone NVARCHAR(2
4),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60),\n    CONSTRAINT PK_Employee PRIMARY KEY  (EmployeeI
d),\n    FOREIGN KEY (ReportsTo) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO AC
TION\n)\n\nCREATE TABLE Track\n(\n    TrackId INTEGER  NOT NULL,\n    Name NVARCHAR(200)  NOT NULL,\n    Al
bumId 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 A
CTION 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 A
CTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE PlaylistTrack\n(\n    PlaylistId INTEGER  NOT NULL,\n    Track
Id 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 K
EY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Al
bum\n(\n    AlbumId INTEGER  NOT NULL,\n    Title NVARCHAR(160)  NOT NULL,\n    ArtistId INTEGER  NOT NUL
L,\n    CONSTRAINT PK_Album PRIMARY KEY  (AlbumId),\n    FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistI
d) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\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 sufficien
t but requires knowledge of a specific string in a particular column, please generate an intermediate SQL q
uery 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 e
xactly as it was given before. \n", '  \n    List all invoices with a total exceeding $10:\n', 'SELECT \n
Invoice.InvoiceId, \n    Invoice.InvoiceDate, \n    Invoice.Total \nFROM \n    Invoice \nWHERE \n    Invoic
e.Total > 10;', '  \n    Find the total number of invoices per country:\n', 'SELECT \n    Invoice.BillingCo
untry, \n    COUNT(Invoice.InvoiceId) AS TotalInvoices \nFROM \n    Invoice \nGROUP BY \n    Invoice.Billin
gCountry \nORDER BY \n    TotalInvoices DESC;', '  \n    Get the total number of invoices for each customer
\n', 'SELECT \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName, \n    COUNT(Invo
ice.InvoiceId) AS TotalInvoices \nFROM \n    Customer \nLEFT JOIN \n    Invoice ON Customer.CustomerId = In
voice.CustomerId \nGROUP BY \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName;',
'Show me a list of tables in the SQLite database', 'SELECT \n    Name \nFROM \n    Playlist\nJOIN \n    Pla
ylistTrack ON Playlist.PlaylistId = PlaylistTrack.PlaylistId\nWHERE \n    PlaylistTrack.TrackId IN (\n
SELECT \n            TrackId \n        FROM \n            InvoiceLine\n        WHERE \n            UnitPric
e > 0.99\n    );', 'How many records are in table called customer', 'SELECT \n    COUNT(*) \nFROM \n    Cus
tomer;', 'How many customers are there', 'SELECT \n    COUNT(*) \nFROM \n    Customer;', '  \n    Find all
tracks with a name containing "What" (case-insensitive)\n', "SELECT \n    Name \nFROM \n    Track\nWHERE \n
LOWER(Name) LIKE '%what%';", '  \n    List all albums and their corresponding artist names  \n', 'SELECT \n
Album.Title AS AlbumTitle, \n    Artist.Name AS ArtistName \nFROM \n    Album \nJOIN \n    Artist ON Album.
ArtistId = Artist.ArtistId;', '  \n    Find all invoices since 2010 and the total amount invoiced:\n']
```sql
SELECT
    Invoice.InvoiceDate,
    SUM(Invoice.Total) AS TotalInvoiced
FROM
    Invoice
WHERE
    Invoice.InvoiceDate >= '2010-01-01'
GROUP BY
    Invoice.InvoiceDate
ORDER BY
    Invoice.InvoiceDate;
```
SELECT
    Invoice.InvoiceDate,
    SUM(Invoice.Total) AS TotalInvoiced
FROM
    Invoice
WHERE
    Invoice.InvoiceDate >= '2010-01-01'
GROUP BY
    Invoice.InvoiceDate
ORDER BY
    Invoice.InvoiceDate;
~~~

```
SELECT
    Invoice.InvoiceDate,
    SUM(Invoice.Total) AS TotalInvoiced
FROM
    Invoice
WHERE
    Invoice.InvoiceDate >= '2010-01-01'
GROUP BY
    Invoice.InvoiceDate
ORDER BY
    Invoice.InvoiceDate;
             InvoiceDate  TotalInvoiced
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]
```


Total Invoiced Since 2010
Total Invoiced
0
5
10
15
20
25
2011
2012
2013
Invoice Date

Out[28]:
```
("SELECT \n    Invoice.InvoiceDate, \n    SUM(Invoice.Total) AS TotalInvoiced \nFROM \n    Invoice \nWHERE
\n    Invoice.InvoiceDate >= '2010-01-01' \nGROUP BY \n    Invoice.InvoiceDate \nORDER BY \n    Invoice.In
voiceDate;",
     InvoiceDate  TotalInvoiced
 0    2010-01-08           3.96
 1    2010-01-09           3.96
 2    2010-01-10           6.94
 3    2010-01-13          17.91
 4    2010-01-18          18.86
 ..          ...            ...
 277  2013-12-05           3.96
 278  2013-12-06           5.94
 279  2013-12-09           8.91
 280  2013-12-14          13.86
 281  2013-12-22           1.99

 [282 rows x 2 columns],
 Figure({
     'data': [{'type': 'scatter',
               'x': array([datetime.datetime(2010, 1, 8, 0, 0),
                           datetime.datetime(2010, 1, 9, 0, 0),
                           datetime.datetime(2010, 1, 10, 0, 0), ...,
                           datetime.datetime(2013, 12, 9, 0, 0),
                           datetime.datetime(2013, 12, 14, 0, 0),
                           datetime.datetime(2013, 12, 22, 0, 0)], dtype=object),
               'y': array([ 3.96,  3.96,  6.94, ...,  8.91, 13.86,  1.99])}],
     'layout': {'template': '...',
                'title': {'text': 'Total Invoiced Since 2010'},
                'xaxis': {'title': {'text': 'Invoice Date'}},
                'yaxis': {'title': {'text': 'Total Invoiced'}}}
 }))
```

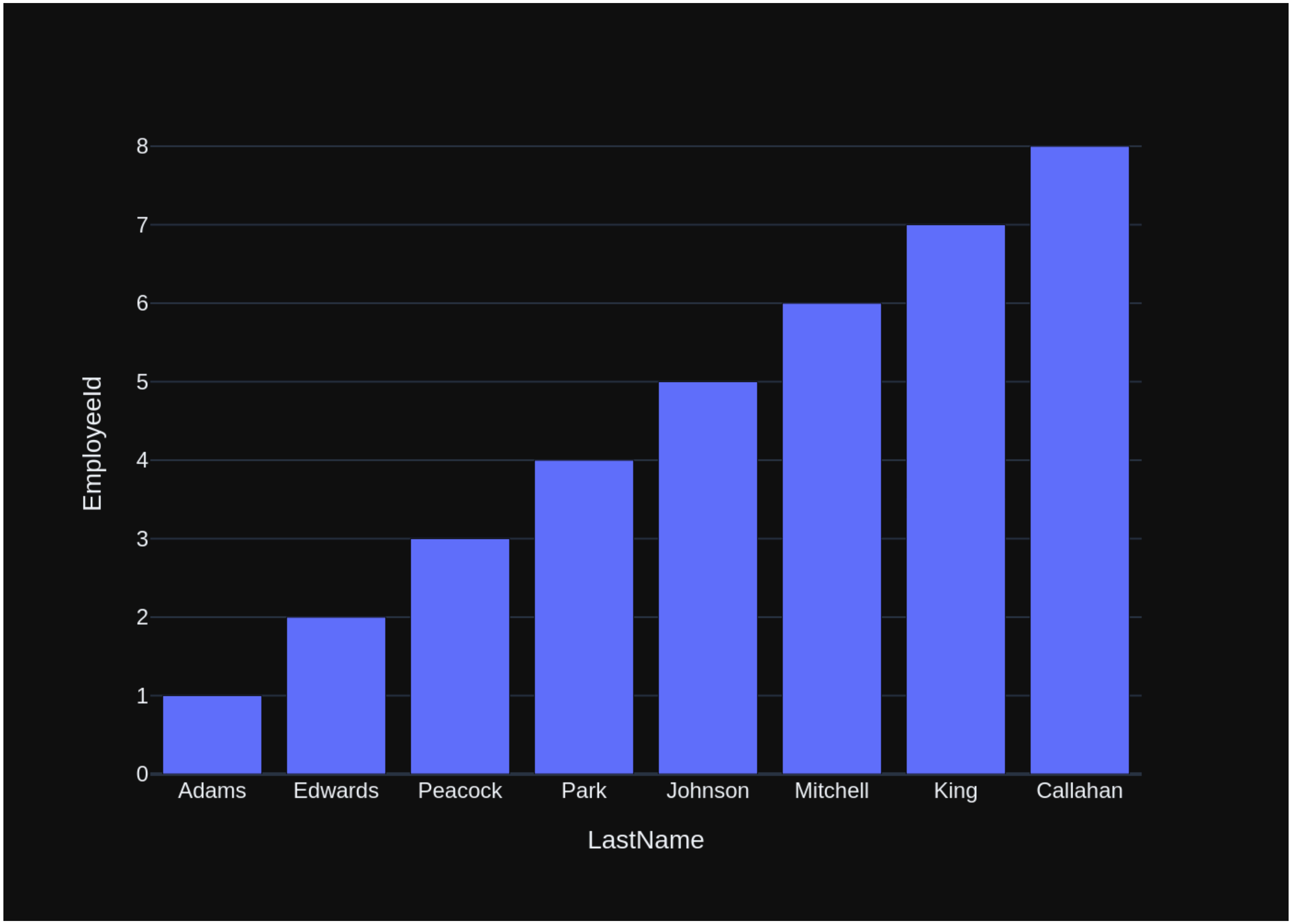
In [29]:
```
question = """
    List all employees and their reporting manager's name (if any):
"""

vn.ask(question=question)
```

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

```
["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 IN
TEGER  NOT NULL,\n    LastName NVARCHAR(20)  NOT NULL,\n    FirstName NVARCHAR(20)  NOT NULL,\n    Title NV
ARCHAR(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 NVARC
HAR(70),\n    City NVARCHAR(40),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVARCH
AR(10),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60)  NOT NULL,\n    SupportRepI
d INTEGER,\n    CONSTRAINT PK_Customer PRIMARY KEY  (CustomerId),\n    FOREIGN KEY (SupportRepId) REFERENCE
S Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_CustomerSuppor
tRepId ON Customer (SupportRepId)\n\nCREATE TABLE Invoice\n(\n    InvoiceId INTEGER  NOT NULL,\n    Custome
rId INTEGER  NOT NULL,\n    InvoiceDate DATETIME  NOT NULL,\n    BillingAddress NVARCHAR(70),\n    BillingC
ity NVARCHAR(40),\n    BillingState NVARCHAR(40),\n    BillingCountry NVARCHAR(40),\n    BillingPostalCode
NVARCHAR(10),\n    Total NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Invoice PRIMARY KEY  (InvoiceId),\n
FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)
\n\nCREATE 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    CO
NSTRAINT PK_InvoiceLine PRIMARY KEY  (InvoiceLineId),\n    FOREIGN KEY (InvoiceId) REFERENCES Invoice (Invo
iceId) \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 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_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  N
OT 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", '  \n    G
```

~~~
et the total number of invoices for each customer\n', 'SELECT \n    Customer.CustomerId, \n    Customer.Fir
stName, \n    Customer.LastName, \n    COUNT(Invoice.InvoiceId) AS TotalInvoices \nFROM \n    Customer \nLE
FT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId \nGROUP BY \n    Customer.CustomerId, \n
Customer.FirstName, \n    Customer.LastName;', '  \n    Find all invoices since 2010 and the total amount i
nvoiced:\n', "SELECT \n    Invoice.InvoiceDate, \n    SUM(Invoice.Total) AS TotalInvoiced \nFROM \n    Invo
ice \nWHERE \n    Invoice.InvoiceDate >= '2010-01-01' \nGROUP BY \n    Invoice.InvoiceDate \nORDER BY \n
Invoice.InvoiceDate;", '  \n    List all albums and their corresponding artist names  \n', 'SELECT \n    Al
bum.Title AS AlbumTitle, \n    Artist.Name AS ArtistName \nFROM \n    Album \nJOIN \n    Artist ON Album.Ar
tistId = Artist.ArtistId;', '  \n    Find the total number of invoices per country:\n', 'SELECT \n    Invoi
ce.BillingCountry, \n    COUNT(Invoice.InvoiceId) AS TotalInvoices \nFROM \n    Invoice \nGROUP BY \n    In
voice.BillingCountry \nORDER BY \n    TotalInvoices DESC;', '  \n    List all invoices with a total exceedi
ng $10:\n', 'SELECT \n    Invoice.InvoiceId, \n    Invoice.InvoiceDate, \n    Invoice.Total \nFROM \n    In
voice \nWHERE \n    Invoice.Total > 10;', 'How many customers are there', 'SELECT \n    COUNT(*) \nFROM \n
Customer;', 'How many records are in table called customer', 'SELECT \n    COUNT(*) \nFROM \n    Custome
r;', '  \n    Find all tracks with a name containing "What" (case-insensitive)\n', "SELECT \n    Name \nFRO
M \n    Track\nWHERE \n    LOWER(Name) LIKE '%what%';", 'Show me a list of tables in the SQLite database',
'SELECT \n    Name \nFROM \n    Playlist\nJOIN \n    PlaylistTrack ON Playlist.PlaylistId = PlaylistTrack.P
laylistId\nWHERE \n    PlaylistTrack.TrackId IN (\n        SELECT \n            TrackId \n        FROM \n
InvoiceLine\n        WHERE \n            UnitPrice > 0.99\n    );', "  \n    List all employees and their r
eporting manager's name (if any):\n"]
```sql
SELECT
    e.EmployeeId,
    e.LastName,
    e.FirstName,
    COALESCE(m.LastName, 'N/A') AS ManagerLastName,
    COALESCE(m.FirstName, 'N/A') AS ManagerFirstName
FROM
    Employee e
LEFT JOIN
    Employee m ON e.ReportsTo = m.EmployeeId;
```
SELECT
    e.EmployeeId,
    e.LastName,
    e.FirstName,
    COALESCE(m.LastName, 'N/A') AS ManagerLastName,
    COALESCE(m.FirstName, 'N/A') AS ManagerFirstName
FROM
    Employee e
LEFT JOIN
    Employee m ON e.ReportsTo = m.EmployeeId;
~~~

```
SELECT
    e.EmployeeId,
    e.LastName,
    e.FirstName,
    COALESCE(m.LastName, 'N/A') AS ManagerLastName,
    COALESCE(m.FirstName, 'N/A') AS ManagerFirstName
FROM
    Employee e
LEFT JOIN
    Employee m ON e.ReportsTo = m.EmployeeId;
   EmployeeId  LastName FirstName ManagerLastName ManagerFirstName
0           1     Adams    Andrew             N/A              N/A
1           2   Edwards     Nancy           Adams           Andrew
2           3   Peacock      Jane         Edwards            Nancy
3           4      Park  Margaret         Edwards            Nancy
4           5   Johnson     Steve         Edwards            Nancy
5           6  Mitchell   Michael           Adams           Andrew
6           7      King    Robert        Mitchell          Michael
7           8  Callahan     Laura        Mitchell          Michael
```


EmployeeId
0
1
2
3
4
5
6
7
8
Adams
Edwards
Peacock
Park
Johnson
Mitchell
King
Callahan
LastName

```
Out[29]: ("SELECT \n    e.EmployeeId, \n    e.LastName, \n    e.FirstName,\n    COALESCE(m.LastName, 'N/A') AS Mana
         gerLastName, \n    COALESCE(m.FirstName, 'N/A') AS ManagerFirstName \nFROM \n    Employee e \nLEFT JOIN \n
         Employee m ON e.ReportsTo = m.EmployeeId;",
             EmployeeId  LastName FirstName ManagerLastName ManagerFirstName
          0           1     Adams    Andrew             N/A              N/A
          1           2   Edwards     Nancy           Adams           Andrew
          2           3   Peacock      Jane         Edwards            Nancy
          3           4      Park  Margaret         Edwards            Nancy
          4           5   Johnson     Steve         Edwards            Nancy
          5           6  Mitchell   Michael           Adams           Andrew
          6           7      King    Robert        Mitchell          Michael
          7           8  Callahan     Laura        Mitchell          Michael,
          Figure({
              'data': [{'alignmentgroup': 'True',
                        'hovertemplate': 'LastName=%{x}<br>EmployeeId=%{y}<extra></extra>',
                        'legendgroup': '',
                        'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
                        'name': '',
                        'offsetgroup': '',
                        'orientation': 'v',
                        'showlegend': False,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Adams', 'Edwards', 'Peacock', 'Park', 'Johnson', 'Mitchell', 'King',
                                    'Callahan'], dtype=object),
                        'xaxis': 'x',
                        'y': array([1, 2, 3, 4, 5, 6, 7, 8]),
                        'yaxis': 'y'}],
              'layout': {'barmode': 'relative',
                         'legend': {'tracegroupgap': 0},
                         'margin': {'t': 60},
                         'template': '...',
                         'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'LastName'}},
                         'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'EmployeeId'}}}
          }))
```

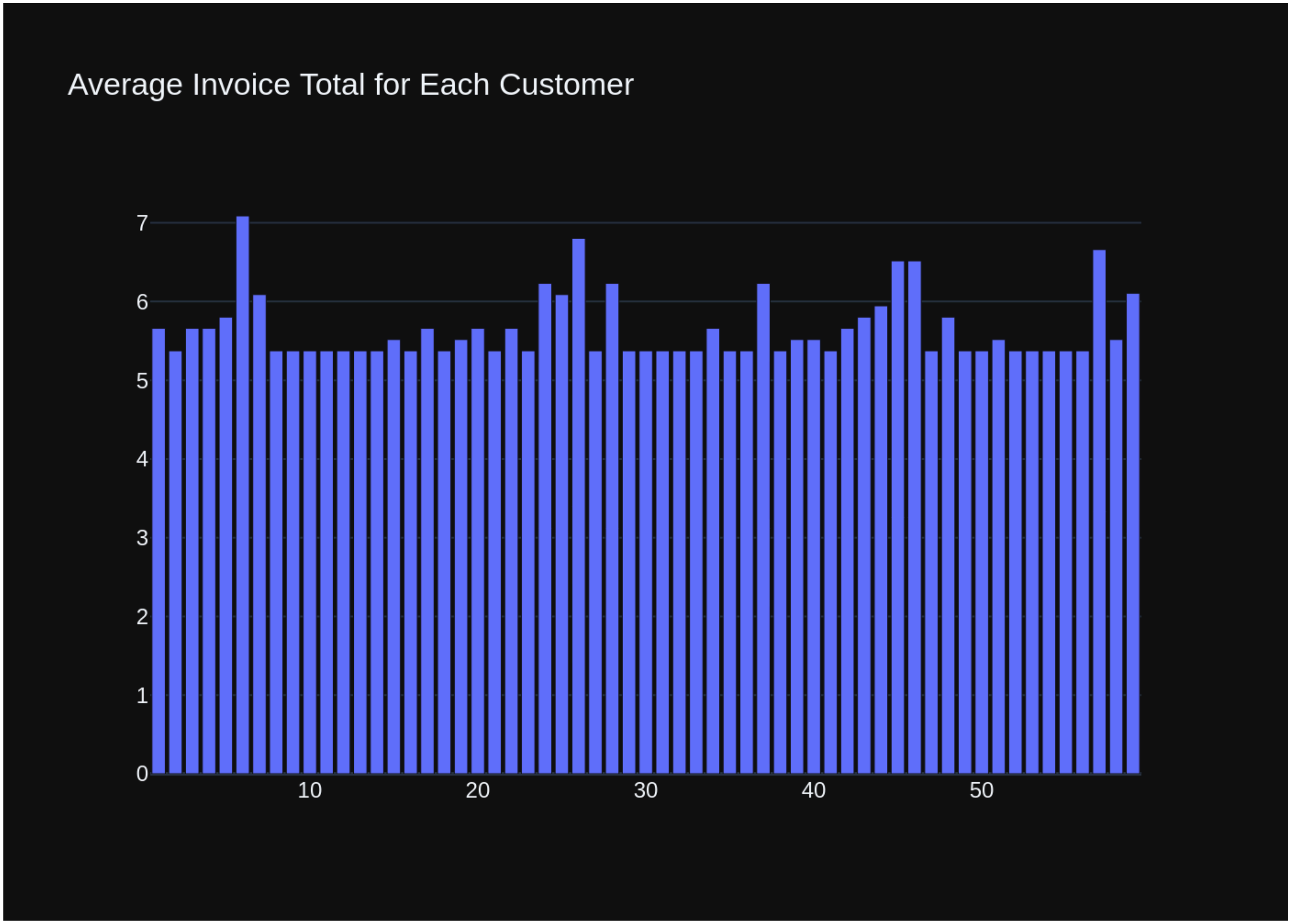
```
In [30]: question = """
             Get the average invoice total for each customer:
         """

         vn.ask(question=question)
```

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

```
["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 In
voiceLine (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(1
0),\n    Total NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Invoice PRIMARY KEY  (InvoiceId),\n    FOREIGN K
EY (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 I
NTEGER  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\n
CREATE INDEX IFK_CustomerSupportRepId ON Customer (SupportRepId)\n\nCREATE TABLE Customer\n(\n    CustomerI
d INTEGER  NOT NULL,\n    FirstName NVARCHAR(40)  NOT NULL,\n    LastName NVARCHAR(20)  NOT NULL,\n    Comp
any 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 NVAR
CHAR(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 INT
EGER  NOT NULL,\n    Name NVARCHAR(200)  NOT NULL,\n    AlbumId INTEGER,\n    MediaTypeId INTEGER  NOT NUL
L,\n    GenreId INTEGER,\n    Composer NVARCHAR(220),\n    Milliseconds INTEGER  NOT NULL,\n    Bytes INTEG
ER,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (TrackId),\n    FOREIGN K
EY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (Ge
nreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (MediaTyp
eId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Em
ployee\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 DAT
ETIME,\n    Address NVARCHAR(70),\n    City NVARCHAR(40),\n    State NVARCHAR(40),\n    Country NVARCHAR(4
0),\n    PostalCode NVARCHAR(10),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(6
0),\n    CONSTRAINT PK_Employee PRIMARY KEY  (EmployeeId),\n    FOREIGN KEY (ReportsTo) REFERENCES Employee
(EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n===Additional Context \n\nIn the chinook
database invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficient, please g
enerate a valid SQL query without any explanations for the question. \n2. If the provided context is almost
sufficient but requires knowledge of a specific string in a particular column, please generate an intermedi
ate SQL query to find the distinct strings in that column. Prepend the query with a comment saying intermed
iate_sql \n3. If the provided context is insufficient, please explain why it can't be generated. \n4. Pleas
e use the most relevant table(s). \n5. If the question has been asked and answered before, please repeat th
e answer exactly as it was given before. \n", '  \n    Get the total number of invoices for each customer
\n', 'SELECT \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName, \n    COUNT(Invo
ice.InvoiceId) AS TotalInvoices \nFROM \n    Customer \nLEFT JOIN \n    Invoice ON Customer.CustomerId = In
voice.CustomerId \nGROUP BY \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName;',
'  \n    Find all invoices since 2010 and the total amount invoiced:\n', "SELECT \n    Invoice.InvoiceDate,
\n    SUM(Invoice.Total) AS TotalInvoiced \nFROM \n    Invoice \nWHERE \n    Invoice.InvoiceDate >= '2010-0
1-01' \nGROUP BY \n    Invoice.InvoiceDate \nORDER BY \n    Invoice.InvoiceDate;", '  \n    Find the total
number of invoices per country:\n', 'SELECT \n    Invoice.BillingCountry, \n    COUNT(Invoice.InvoiceId) AS
TotalInvoices \nFROM \n    Invoice \nGROUP BY \n    Invoice.BillingCountry \nORDER BY \n    TotalInvoices D
ESC;', '  \n    List all invoices with a total exceeding $10:\n', 'SELECT \n    Invoice.InvoiceId, \n    In
voice.InvoiceDate, \n    Invoice.Total \nFROM \n    Invoice \nWHERE \n    Invoice.Total > 10;', 'How many c
ustomers are there', 'SELECT \n    COUNT(*) \nFROM \n    Customer;', 'How many records are in table called
customer', 'SELECT \n    COUNT(*) \nFROM \n    Customer;', 'Show me a list of tables in the SQLite databas
e', 'SELECT \n    Name \nFROM \n    Playlist\nJOIN \n    PlaylistTrack ON Playlist.PlaylistId = PlaylistTra
ck.PlaylistId\nWHERE \n    PlaylistTrack.TrackId IN (\n        SELECT \n            TrackId \n        FROM
\n            InvoiceLine\n        WHERE \n            UnitPrice > 0.99\n    );', "  \n    List all employe
es and their reporting manager's name (if any):\n", "SELECT \n    e.EmployeeId, \n    e.LastName, \n    e.F
irstName,\n    COALESCE(m.LastName, 'N/A') AS ManagerLastName, \n    COALESCE(m.FirstName, 'N/A') AS Manage
rFirstName \nFROM \n    Employee e \nLEFT JOIN \n    Employee m ON e.ReportsTo = m.EmployeeId;", '  \n    F
ind all tracks with a name containing "What" (case-insensitive)\n', "SELECT \n    Name \nFROM \n    Track\n
WHERE \n    LOWER(Name) LIKE '%what%';", '  \n    List all albums and their corresponding artist names    
\n', 'SELECT \n    Album.Title AS AlbumTitle, \n    Artist.Name AS ArtistName \nFROM \n    Album \nJOIN \n
Artist ON Album.ArtistId = Artist.ArtistId;', '  \n    Get the average invoice total for each customer:\n']
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    AVG(Invoice.Total) AS AverageInvoiceTotal
FROM
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
GROUP BY
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName;
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    AVG(Invoice.Total) AS AverageInvoiceTotal
FROM
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
GROUP BY
```

```
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName;
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    AVG(Invoice.Total) AS AverageInvoiceTotal
FROM
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
GROUP BY
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName;
    CustomerId  FirstName      LastName  AverageInvoiceTotal
0            1       Luís     Gonçalves             5.660000
1            2     Leonie        Köhler             5.374286
2            3   François      Tremblay             5.660000
3            4      Bjørn        Hansen             5.660000
4            5  František   Wichterlová             5.802857
5            6     Helena          Holý             7.088571
6            7     Astrid        Gruber             6.088571
7            8       Daan       Peeters             5.374286
8            9       Kara       Nielsen             5.374286
9           10    Eduardo       Martins             5.374286
10          11  Alexandre         Rocha             5.374286
11          12    Roberto       Almeida             5.374286
12          13   Fernanda         Ramos             5.374286
13          14       Mark       Philips             5.374286
14          15   Jennifer      Peterson             5.517143
15          16      Frank        Harris             5.374286
16          17       Jack         Smith             5.660000
17          18   Michelle        Brooks             5.374286
18          19        Tim         Goyer             5.517143
19          20        Dan        Miller             5.660000
20          21      Kathy         Chase             5.374286
21          22    Heather       Leacock             5.660000
22          23       John        Gordon             5.374286
23          24      Frank       Ralston             6.231429
24          25     Victor       Stevens             6.088571
```

```
25           26    Richard    Cunningham            6.802857
26           27    Patrick          Gray            5.374286
27           28      Julia       Barnett            6.231429
28           29     Robert         Brown            5.374286
29           30     Edward       Francis            5.374286
30           31     Martha          Silk            5.374286
31           32      Aaron      Mitchell            5.374286
32           33     Ellie       Sullivan            5.374286
33           34       João     Fernandes            5.660000
34           35   Madalena       Sampaio            5.374286
35           36     Hannah     Schneider            5.374286
36           37       Fynn    Zimmermann            6.231429
37           38     Niklas      Schröder            5.374286
38           39    Camille       Bernard            5.517143
39           40  Dominique      Lefebvre            5.517143
40           41       Marc        Dubois            5.374286
41           42      Wyatt        Girard            5.660000
42           43   Isabelle       Mercier            5.802857
43           44      Terhi    Hämäläinen            5.945714
44           45   Ladislav        Kovács            6.517143
45           46       Hugh      O'Reilly            6.517143
46           47      Lucas       Mancini            5.374286
47           48   Johannes  Van der Berg            5.802857
48           49  Stanisław        Wójcik            5.374286
49           50    Enrique         Muñoz            5.374286
50           51     Joakim     Johansson            5.517143
51           52       Emma         Jones            5.374286
52           53       Phil        Hughes            5.374286
53           54      Steve        Murray            5.374286
54           55       Mark        Taylor            5.374286
55           56      Diego     Gutiérrez            5.374286
56           57       Luis         Rojas            6.660000
57           58      Manoj        Pareek            5.517143
58           59       Puja    Srivastava            6.106667
```


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

```
Out[30]: ('SELECT \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName, \n    AVG(Invoice.T
         otal) AS AverageInvoiceTotal \nFROM \n    Customer \nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invo
         ice.CustomerId \nGROUP BY \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName;',
              CustomerId  FirstName      LastName  AverageInvoiceTotal
          0            1       Luís     Gonçalves             5.660000
          1            2     Leonie        Köhler             5.374286
          2            3   François      Tremblay             5.660000
          3            4      Bjørn        Hansen             5.660000
          4            5   František   Wichterlová           5.802857
          5            6     Helena          Holý             7.088571
          6            7     Astrid        Gruber             6.088571
          7            8       Daan       Peeters             5.374286
          8            9       Kara       Nielsen             5.374286
          9           10    Eduardo       Martins             5.374286
          10          11  Alexandre         Rocha             5.374286
          11          12    Roberto       Almeida             5.374286
          12          13   Fernanda         Ramos             5.374286
          13          14       Mark       Philips             5.374286
          14          15   Jennifer      Peterson             5.517143
          15          16      Frank        Harris             5.374286
          16          17       Jack         Smith             5.660000
          17          18   Michelle        Brooks             5.374286
          18          19        Tim         Goyer             5.517143
          19          20        Dan        Miller             5.660000
          20          21      Kathy         Chase             5.374286
          21          22    Heather       Leacock             5.660000
          22          23       John        Gordon             5.374286
          23          24      Frank       Ralston             6.231429
          24          25     Victor       Stevens             6.088571
          25          26    Richard    Cunningham             6.802857
          26          27    Patrick          Gray             5.374286
          27          28      Julia       Barnett             6.231429
          28          29     Robert         Brown             5.374286
          29          30     Edward       Francis             5.374286
          30          31     Martha          Silk             5.374286
          31          32      Aaron      Mitchell             5.374286
          32          33      Ellie      Sullivan             5.374286
          33          34       João     Fernandes             5.660000
          34          35   Madalena       Sampaio             5.374286
          35          36     Hannah     Schneider             5.374286
          36          37       Fynn    Zimmermann             6.231429
          37          38    Niklas      Schröder             5.374286
          38           39    Camille       Bernard             5.517143
          39           40  Dominique      Lefebvre             5.517143
          40           41       Marc        Dubois             5.374286
          41           42      Wyatt        Girard             5.660000
          42           43   Isabelle       Mercier             5.802857
          43           44      Terhi    Hämäläinen             5.945714
          44           45   Ladislav        Kovács             6.517143
          45           46       Hugh      O'Reilly             6.517143
          46           47      Lucas       Mancini             5.374286
          47           48   Johannes  Van der Berg             5.802857
          48           49  Stanisław        Wójcik             5.374286
          49           50    Enrique         Muñoz             5.374286
          50           51     Joakim     Johansson             5.517143
          51           52       Emma         Jones             5.374286
          52           53       Phil        Hughes             5.374286
          53           54      Steve        Murray             5.374286
          54           55       Mark        Taylor             5.374286
          55           56      Diego     Gutiérrez             5.374286
          56           57       Luis         Rojas             6.660000
          57           58      Manoj        Pareek             5.517143
          58           59       Puja    Srivastava             6.106667,
          Figure({
              'data': [{'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]),
                        '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])}],
              'layout': {'template': '...', 'title': {'text': 'Average Invoice Total for Each Customer'}}
          }))
```

In [31]:
```
question = """
    Find the top 5 most expensive tracks (based on unit price):
```

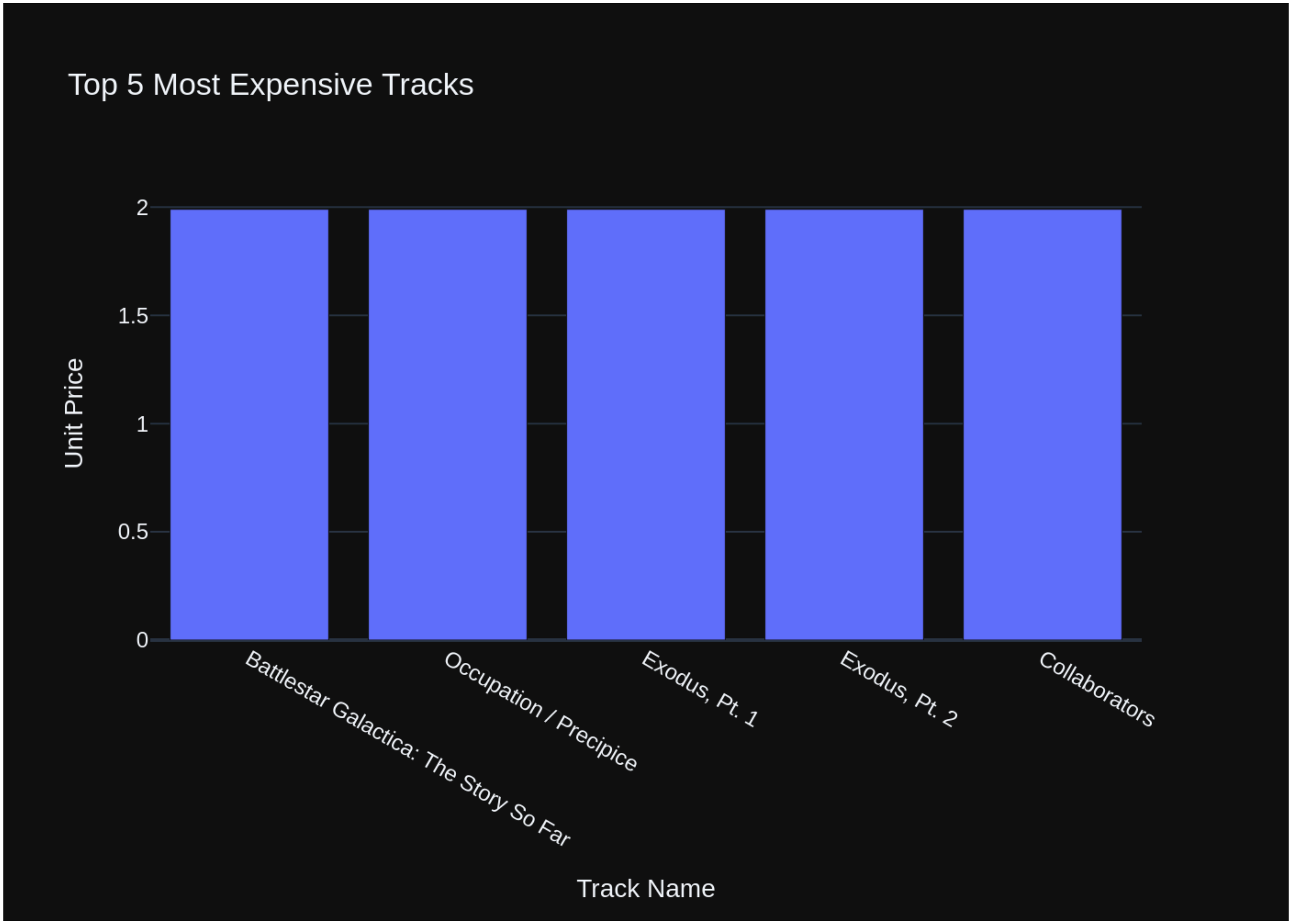
```
"""

vn.ask(question=question)
```

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

```
["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 INT
EGER,\n    MediaTypeId INTEGER  NOT NULL,\n    GenreId INTEGER,\n    Composer NVARCHAR(220),\n    Milliseco
nds INTEGER  NOT NULL,\n    Bytes INTEGER,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Track
PRIMARY KEY  (TrackId),\n    FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON
UPDATE NO ACTION,\n    FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE
NO ACTION,\n    FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON U
PDATE NO ACTION\n)\n\nCREATE 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_Inv
oiceLineTrackId ON InvoiceLine (TrackId)\n\nCREATE INDEX IFK_TrackMediaTypeId ON Track (MediaTypeId)\n\nCRE
ATE TABLE InvoiceLine\n(\n    InvoiceLineId INTEGER  NOT NULL,\n    InvoiceId INTEGER  NOT NULL,\n    Track
Id INTEGER  NOT NULL,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    Quantity INTEGER  NOT NULL,\n    CONSTRA
INT PK_InvoiceLine PRIMARY KEY  (InvoiceLineId),\n    FOREIGN KEY (InvoiceId) REFERENCES Invoice (InvoiceI
d) \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  NO
T NULL,\n    TrackId INTEGER  NOT NULL,\n    CONSTRAINT PK_PlaylistTrack PRIMARY KEY  (PlaylistId, TrackI
d),\n    FOREIGN KEY (PlaylistId) REFERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO A
CTION,\n    FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION
\n)\n\nCREATE INDEX IFK_AlbumArtistId ON Album (ArtistId)\n\nCREATE TABLE Album\n(\n    AlbumId INTEGER  NO
T NULL,\n    Title NVARCHAR(160)  NOT NULL,\n    ArtistId INTEGER  NOT NULL,\n    CONSTRAINT PK_Album PRIMA
RY KEY  (AlbumId),\n    FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON UP
DATE 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 explan
ations for the question. \n2. If the provided context is almost sufficient but requires knowledge of a spec
ific 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 in
sufficient, please explain why it can't be generated. \n4. Please use the most relevant table(s). \n5. If t
he question has been asked and answered before, please repeat the answer exactly as it was given before.
\n", 'Show me a list of tables in the SQLite database', 'SELECT \n    Name \nFROM \n    Playlist\nJOIN \n
PlaylistTrack ON Playlist.PlaylistId = PlaylistTrack.PlaylistId\nWHERE \n    PlaylistTrack.TrackId IN (\n
SELECT \n            TrackId \n        FROM \n            InvoiceLine\n        WHERE \n            UnitPric
e > 0.99\n    );', '  \n    Find all tracks with a name containing "What" (case-insensitive)\n', "SELECT \n
Name \nFROM \n    Track\nWHERE \n    LOWER(Name) LIKE '%what%';", '  \n    List all invoices with a total e
xceeding $10:\n', 'SELECT \n    Invoice.InvoiceId, \n    Invoice.InvoiceDate, \n    Invoice.Total \nFROM \n
Invoice \nWHERE \n    Invoice.Total > 10;', '  \n    List all albums and their corresponding artist names
\n', 'SELECT \n    Album.Title AS AlbumTitle, \n    Artist.Name AS ArtistName \nFROM \n    Album \nJOIN \n
Artist ON Album.ArtistId = Artist.ArtistId;', '  \n    Find all invoices since 2010 and the total amount in
voiced:\n', "SELECT \n    Invoice.InvoiceDate, \n    SUM(Invoice.Total) AS TotalInvoiced \nFROM \n    Invoi
ce \nWHERE \n    Invoice.InvoiceDate >= '2010-01-01' \nGROUP BY \n    Invoice.InvoiceDate \nORDER BY \n
Invoice.InvoiceDate;", '  \n    Get the average invoice total for each customer:\n', 'SELECT \n    Custome
r.CustomerId, \n    Customer.FirstName, \n    Customer.LastName, \n    AVG(Invoice.Total) AS AverageInvoice
```

~~~
Total \nFROM \n    Customer \nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId \nGROUP B
Y \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName;', '  \n    Find the total n
umber of invoices per country:\n', 'SELECT \n    Invoice.BillingCountry, \n    COUNT(Invoice.InvoiceId) AS 
TotalInvoices \nFROM \n    Invoice \nGROUP BY \n    Invoice.BillingCountry \nORDER BY \n    TotalInvoices D
ESC;', '  \n    Get the total number of invoices for each customer\n', 'SELECT \n    Customer.CustomerId, 
\n    Customer.FirstName, \n    Customer.LastName, \n    COUNT(Invoice.InvoiceId) AS TotalInvoices \nFROM 
\n    Customer \nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId \nGROUP BY \n    Custo
mer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName;', 'How many customers are there', 'SELEC
T \n    COUNT(*) \nFROM \n    Customer;', 'How many records are in table called customer', 'SELECT \n    CO
UNT(*) \nFROM \n    Customer;', '  \n    Find the top 5 most expensive tracks (based on unit price):\n']
```sql
SELECT
    Name,
    UnitPrice
FROM
    Track
ORDER BY
    UnitPrice DESC
LIMIT
    5;
```
SELECT
    Name,
    UnitPrice
FROM
    Track
ORDER BY
    UnitPrice DESC
LIMIT
    5;
SELECT
    Name,
    UnitPrice
FROM
    Track
ORDER BY
    UnitPrice DESC
LIMIT
    5;
                                     Name  UnitPrice
0  Battlestar Galactica: The Story So Far       1.99
1                    Occupation / Precipice       1.99
~~~

```
2                         Exodus, Pt. 1       1.99
3                         Exodus, Pt. 2       1.99
4                         Collaborators       1.99
```



```
Out[31]: ('SELECT \n    Name, \n    UnitPrice \nFROM \n    Track \nORDER BY \n    UnitPrice DESC \nLIMIT \n    5;',
                                        Name  UnitPrice
          0  Battlestar Galactica: The Story So Far       1.99
          1                 Occupation / Precipice       1.99
          2                         Exodus, Pt. 1       1.99
          3                         Exodus, Pt. 2       1.99
          4                         Collaborators       1.99,
          Figure({
              'data': [{'name': 'Unit Price',
                        'type': 'bar',
                        'x': array(['Battlestar Galactica: The Story So Far', 'Occupation / Precipice',
                                    'Exodus, Pt. 1', 'Exodus, Pt. 2', 'Collaborators'], dtype=object),
                        'y': array([1.99, 1.99, 1.99, 1.99, 1.99])}],
              'layout': {'template': '...',
                         'title': {'text': 'Top 5 Most Expensive Tracks'},
                         'xaxis': {'title': {'text': 'Track Name'}},
                         'yaxis': {'title': {'text': 'Unit Price'}}}
          }))
```

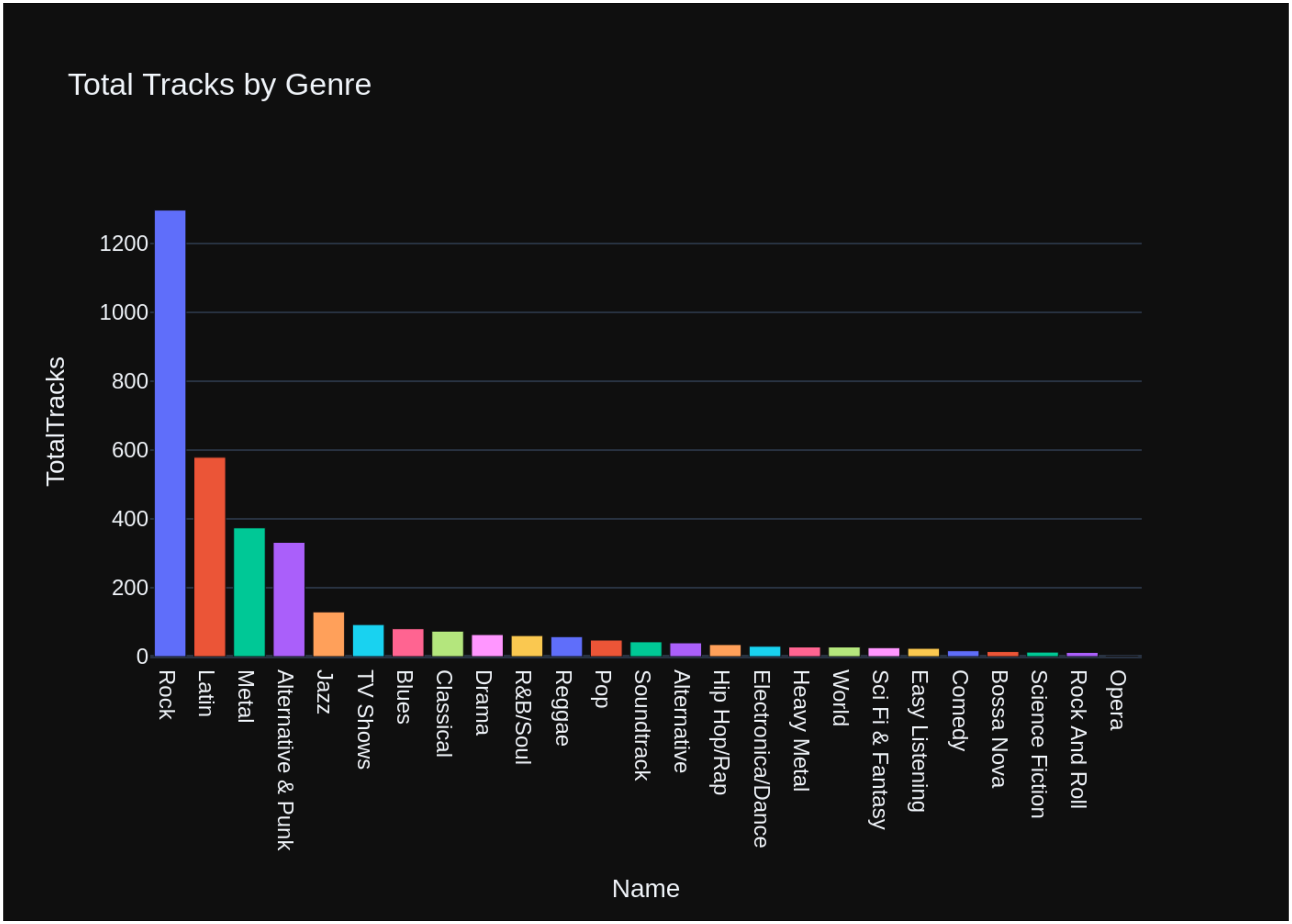
```
In [32]: question = """
             List all genres and the number of tracks in each genre:
         """

         vn.ask(question=question)
```

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

```
["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 INT
EGER,\n    MediaTypeId INTEGER  NOT NULL,\n    GenreId INTEGER,\n    Composer NVARCHAR(220),\n    Milliseco
nds INTEGER  NOT NULL,\n    Bytes INTEGER,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Track
PRIMARY KEY  (TrackId),\n    FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON
UPDATE NO ACTION,\n    FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE
NO ACTION,\n    FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON U
PDATE NO ACTION\n)\n\nCREATE INDEX IFK_TrackGenreId ON Track (GenreId)\n\nCREATE TABLE Genre\n(\n    GenreI
d 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 (AlbumI
d)\n\nCREATE INDEX IFK_TrackMediaTypeId ON Track (MediaTypeId)\n\nCREATE TABLE Album\n(\n    AlbumId INTEGE
R  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 INT
EGER  NOT NULL,\n    CONSTRAINT PK_PlaylistTrack PRIMARY KEY  (PlaylistId, TrackId),\n    FOREIGN KEY (Play
listId) REFERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (T
rackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_Alb
umArtistId ON Album (ArtistId)\n\nCREATE TABLE Playlist\n(\n    PlaylistId INTEGER  NOT NULL,\n    Name NVA
RCHAR(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 i
s 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. \n
4. Please use the most relevant table(s). \n5. If the question has been asked and answered before, please r
epeat the answer exactly as it was given before. \n", '  \n    Find the top 5 most expensive tracks (based
on unit price):\n', 'SELECT \n    Name, \n    UnitPrice \nFROM \n    Track \nORDER BY \n    UnitPrice DESC
\nLIMIT \n    5;', '  \n    List all albums and their corresponding artist names  \n', 'SELECT \n    Album.
Title AS AlbumTitle, \n    Artist.Name AS ArtistName \nFROM \n    Album \nJOIN \n    Artist ON Album.Artist
Id = Artist.ArtistId;', '  \n    Find all tracks with a name containing "What" (case-insensitive)\n', "SELE
CT \n    Name \nFROM \n    Track\nWHERE \n    LOWER(Name) LIKE '%what%';", 'Show me a list of tables in the
SQLite database', 'SELECT \n    Name \nFROM \n    Playlist\nJOIN \n    PlaylistTrack ON Playlist.PlaylistId
= PlaylistTrack.PlaylistId\nWHERE \n    PlaylistTrack.TrackId IN (\n        SELECT \n            TrackId \n
FROM \n            InvoiceLine\n        WHERE \n            UnitPrice > 0.99\n    );', '  \n    Find the to
tal number of invoices per country:\n', 'SELECT \n    Invoice.BillingCountry, \n    COUNT(Invoice.InvoiceI
d) AS TotalInvoices \nFROM \n    Invoice \nGROUP BY \n    Invoice.BillingCountry \nORDER BY \n    TotalInvo
ices DESC;', '  \n    List all invoices with a total exceeding $10:\n', 'SELECT \n    Invoice.InvoiceId, \n
Invoice.InvoiceDate, \n    Invoice.Total \nFROM \n    Invoice \nWHERE \n    Invoice.Total > 10;', '  \n
Find all invoices since 2010 and the total amount invoiced:\n', "SELECT \n    Invoice.InvoiceDate, \n    SU
M(Invoice.Total) AS TotalInvoiced \nFROM \n    Invoice \nWHERE \n    Invoice.InvoiceDate >= '2010-01-01' \n
GROUP BY \n    Invoice.InvoiceDate \nORDER BY \n    Invoice.InvoiceDate;", 'How many records are in table c
```

~~~
alled customer', 'SELECT \n    COUNT(*) \nFROM \n    Customer;', 'How many customers are there', 'SELECT \n
COUNT(*) \nFROM \n    Customer;', '  \n    Get the total number of invoices for each customer\n', 'SELECT
\n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName, \n    COUNT(Invoice.InvoiceI
d) AS TotalInvoices \nFROM \n    Customer \nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.Custom
erId \nGROUP BY \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName;', '  \n    Li
st all genres and the number of tracks in each genre:\n']
```sql
SELECT
    Genre.Name,
    COUNT(Track.TrackId) AS TotalTracks
FROM
    Genre
LEFT JOIN
    Track ON Genre.GenreId = Track.GenreId
GROUP BY
    Genre.GenreId,
    Genre.Name
ORDER BY
    TotalTracks DESC;
```
SELECT
    Genre.Name,
    COUNT(Track.TrackId) AS TotalTracks
FROM
    Genre
LEFT JOIN
    Track ON Genre.GenreId = Track.GenreId
GROUP BY
    Genre.GenreId,
    Genre.Name
ORDER BY
    TotalTracks DESC;
SELECT
    Genre.Name,
    COUNT(Track.TrackId) AS TotalTracks
FROM
    Genre
LEFT JOIN
    Track ON Genre.GenreId = Track.GenreId
GROUP BY
    Genre.GenreId,
    Genre.Name
~~~

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


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

```
Out[32]: ('SELECT \n    Genre.Name, \n    COUNT(Track.TrackId) AS TotalTracks \nFROM \n    Genre \nLEFT JOIN \n
         Track ON Genre.GenreId = Track.GenreId \nGROUP BY \n    Genre.GenreId, \n    Genre.Name \nORDER BY \n    T
         otalTracks DESC;',
                           Name  TotalTracks
          0                Rock         1297
          1               Latin          579
          2               Metal          374
          3   Alternative & Punk         332
          4                Jazz          130
          5            TV Shows           93
          6               Blues           81
          7           Classical           74
          8               Drama           64
          9            R&B/Soul           61
          10             Reggae           58
          11                Pop           48
          12         Soundtrack           43
          13        Alternative           40
          14        Hip Hop/Rap           35
          15  Electronica/Dance           30
          16        Heavy Metal           28
          17              World           28
          18    Sci Fi & Fantasy          26
          19     Easy Listening           24
          20             Comedy           17
          21         Bossa Nova           15
          22    Science Fiction           13
          23      Rock And Roll           12
          24              Opera            1,
          Figure({
              'data': [{'alignmentgroup': 'True',
                        'hovertemplate': 'Name=%{x}<br>TotalTracks=%{y}<extra></extra>',
                        'legendgroup': 'Rock',
                        'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
                        'name': 'Rock',
                        'offsetgroup': 'Rock',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Rock'], dtype=object),
                        'xaxis': 'x',
                        'y': array([1297]),
                        'yaxis': 'y'},
                       {'alignmentgroup': 'True',
                        'hovertemplate': 'Name=%{x}<br>TotalTracks=%{y}<extra></extra>',
                        'legendgroup': 'Latin',
                        'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
                        'name': 'Latin',
                        'offsetgroup': 'Latin',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Latin'], dtype=object),
                        'xaxis': 'x',
                        'y': array([579]),
                        'yaxis': 'y'},
                       {'alignmentgroup': 'True',
                        'hovertemplate': 'Name=%{x}<br>TotalTracks=%{y}<extra></extra>',
                        'legendgroup': 'Metal',
                        'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
                        'name': 'Metal',
                        'offsetgroup': 'Metal',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Metal'], dtype=object),
                        'xaxis': 'x',
                        'y': array([374]),
                        'yaxis': 'y'},
                       {'alignmentgroup': 'True',
                        'hovertemplate': 'Name=%{x}<br>TotalTracks=%{y}<extra></extra>',
                        'legendgroup': 'Alternative & Punk',
                        'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
                        'name': 'Alternative & Punk',
                        'offsetgroup': 'Alternative & Punk',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Alternative & Punk'], dtype=object),
                        'xaxis': 'x',
                        'y': array([332]),
                        'yaxis': 'y'},
                       {'alignmentgroup': 'True',
                        'hovertemplate': 'Name=%{x}<br>TotalTracks=%{y}<extra></extra>',
                        'legendgroup': 'Jazz',
                        'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
                        'name': 'Jazz',
                        'offsetgroup': 'Jazz',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Jazz'], dtype=object),
                        'xaxis': 'x',
                        'y': array([130]),
                        'yaxis': 'y'},
                       {'alignmentgroup': 'True',
                        'hovertemplate': 'Name=%{x}<br>TotalTracks=%{y}<extra></extra>',
                        'legendgroup': 'TV Shows',
                        'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
                        'name': 'TV Shows',
                        'offsetgroup': 'TV Shows',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['TV Shows'], dtype=object),
                        'xaxis': 'x',
                        'y': array([93]),
                        'yaxis': 'y'},
                       {'alignmentgroup': 'True',
                        'hovertemplate': 'Name=%{x}<br>TotalTracks=%{y}<extra></extra>',
                        'legendgroup': 'Blues',
                        'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
                        'name': 'Blues',
                        'offsetgroup': 'Blues',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Blues'], dtype=object),
                        'xaxis': 'x',
                        'y': array([81]),
                        'yaxis': 'y'},
                       {'alignmentgroup': 'True',
                        'hovertemplate': 'Name=%{x}<br>TotalTracks=%{y}<extra></extra>',
                        'legendgroup': 'Classical',
                        'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
                        'name': 'Classical',
                        'offsetgroup': 'Classical',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Classical'], dtype=object),
                        'xaxis': 'x',
                        'y': array([74]),
                        'yaxis': 'y'},
                       {'alignmentgroup': 'True',
                        'hovertemplate': 'Name=%{x}<br>TotalTracks=%{y}<extra></extra>',
                        'legendgroup': 'Drama',
                        'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
                        'name': 'Drama',
                        'offsetgroup': 'Drama',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Drama'], dtype=object),
                        'xaxis': 'x',
                        'y': array([64]),
                        'yaxis': 'y'},
                       {'alignmentgroup': 'True',
                        'hovertemplate': 'Name=%{x}<br>TotalTracks=%{y}<extra></extra>',
                        'legendgroup': 'R&B/Soul',
                        'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
                        'name': 'R&B/Soul',
                        'offsetgroup': 'R&B/Soul',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['R&B/Soul'], dtype=object),
                        'xaxis': 'x',
                        'y': array([61]),
                        'yaxis': 'y'},
                       {'alignmentgroup': 'True',
                        'hovertemplate': 'Name=%{x}<br>TotalTracks=%{y}<extra></extra>',
                        'legendgroup': 'Reggae',
                        'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
                        'name': 'Reggae',
                        'offsetgroup': 'Reggae',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Reggae'], dtype=object),
                        'xaxis': 'x',
                        'y': array([58]),
                        'yaxis': 'y'},
                       {'alignmentgroup': 'True',
                        'hovertemplate': 'Name=%{x}<br>TotalTracks=%{y}<extra></extra>',
                        'legendgroup': 'Pop',
                        'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
                        'name': 'Pop',
                        'offsetgroup': 'Pop',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Pop'], dtype=object),
                        'xaxis': 'x',
                        'y': array([48]),
                        'yaxis': 'y'},
                       {'alignmentgroup': 'True',
                        'hovertemplate': 'Name=%{x}<br>TotalTracks=%{y}<extra></extra>',
                        'legendgroup': 'Soundtrack',
                        'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
                        'name': 'Soundtrack',
                        'offsetgroup': 'Soundtrack',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Soundtrack'], dtype=object),
                        'xaxis': 'x',
                        'y': array([43]),
                        'yaxis': 'y'},
                       {'alignmentgroup': 'True',
                        'hovertemplate': 'Name=%{x}<br>TotalTracks=%{y}<extra></extra>',
                        'legendgroup': 'Alternative',
                        'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
                        'name': 'Alternative',
                        'offsetgroup': 'Alternative',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Alternative'], dtype=object),
                        'xaxis': 'x',
                        'y': array([40]),
                        'yaxis': 'y'},
                       {'alignmentgroup': 'True',
                        'hovertemplate': 'Name=%{x}<br>TotalTracks=%{y}<extra></extra>',
                        'legendgroup': 'Hip Hop/Rap',
                        'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
                        'name': 'Hip Hop/Rap',
                        'offsetgroup': 'Hip Hop/Rap',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Hip Hop/Rap'], dtype=object),
                        'xaxis': 'x',
                        'y': array([35]),
                        'yaxis': 'y'},
                       {'alignmentgroup': 'True',
                        'hovertemplate': 'Name=%{x}<br>TotalTracks=%{y}<extra></extra>',
                        'legendgroup': 'Electronica/Dance',
                        'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
                        'name': 'Electronica/Dance',
                        'offsetgroup': 'Electronica/Dance',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Electronica/Dance'], dtype=object),
                        'xaxis': 'x',
                        'y': array([30]),
                        'yaxis': 'y'},
                       {'alignmentgroup': 'True',
                        'hovertemplate': 'Name=%{x}<br>TotalTracks=%{y}<extra></extra>',
                        'legendgroup': 'Heavy Metal',
                        'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
                        'name': 'Heavy Metal',
                        'offsetgroup': 'Heavy Metal',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Heavy Metal'], dtype=object),
                        'xaxis': 'x',
                        'y': array([28]),
                        'yaxis': 'y'},
                       {'alignmentgroup': 'True',
                        'hovertemplate': 'Name=%{x}<br>TotalTracks=%{y}<extra></extra>',
                        'legendgroup': 'World',
                        'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
                        'name': 'World',
                        'offsetgroup': 'World',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['World'], dtype=object),
                        'xaxis': 'x',
                        'y': array([28]),
                        'yaxis': 'y'},
                       {'alignmentgroup': 'True',
                        'hovertemplate': 'Name=%{x}<br>TotalTracks=%{y}<extra></extra>',
                        'legendgroup': 'Sci Fi & Fantasy',
                        'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
                        'name': 'Sci Fi & Fantasy',
                        'offsetgroup': 'Sci Fi & Fantasy',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Sci Fi & Fantasy'], dtype=object),
                        'xaxis': 'x',
                        'y': array([26]),
                        'yaxis': 'y'},
                       {'alignmentgroup': 'True',
                        'hovertemplate': 'Name=%{x}<br>TotalTracks=%{y}<extra></extra>',
                        'legendgroup': 'Easy Listening',
                        'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
                        'name': 'Easy Listening',
                        'offsetgroup': 'Easy Listening',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Easy Listening'], dtype=object),
                        'xaxis': 'x',
                        'y': array([24]),
                        'yaxis': 'y'},
                       {'alignmentgroup': 'True',
                        'hovertemplate': 'Name=%{x}<br>TotalTracks=%{y}<extra></extra>',
                        'legendgroup': 'Comedy',
                        'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
                        'name': 'Comedy',
                        'offsetgroup': 'Comedy',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Comedy'], dtype=object),
                        'xaxis': 'x',
                        'y': array([17]),
                        'yaxis': 'y'},
                       {'alignmentgroup': 'True',
                        'hovertemplate': 'Name=%{x}<br>TotalTracks=%{y}<extra></extra>',
                        'legendgroup': 'Bossa Nova',
                        'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
                        'name': 'Bossa Nova',
                        'offsetgroup': 'Bossa Nova',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Bossa Nova'], dtype=object),
                        'xaxis': 'x',
                        'y': array([15]),
                        'yaxis': 'y'},
                       {'alignmentgroup': 'True',
                        'hovertemplate': 'Name=%{x}<br>TotalTracks=%{y}<extra></extra>',
                        'legendgroup': 'Science Fiction',
                        'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
                        'name': 'Science Fiction',
                        'offsetgroup': 'Science Fiction',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Science Fiction'], dtype=object),
                        'xaxis': 'x',
                        'y': array([13]),
                        'yaxis': 'y'},
                       {'alignmentgroup': 'True',
                        'hovertemplate': 'Name=%{x}<br>TotalTracks=%{y}<extra></extra>',
                        'legendgroup': 'Rock And Roll',
                        'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
                        'name': 'Rock And Roll',
                        'offsetgroup': 'Rock And Roll',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Rock And Roll'], dtype=object),
                        'xaxis': 'x',
                        'y': array([12]),
                        'yaxis': 'y'},
                       {'alignmentgroup': 'True',
                        'hovertemplate': 'Name=%{x}<br>TotalTracks=%{y}<extra></extra>',
                        'legendgroup': 'Opera',
                        'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
                        'name': 'Opera',
                        'offsetgroup': 'Opera',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Opera'], dtype=object),
                        'xaxis': 'x',
                        'y': array([1]),
                        'yaxis': 'y'}],
              'layout': {'barmode': 'relative',
                         'legend': {'title': {'text': 'Name'}, 'tracegroupgap': 0},
                         'showlegend': False,
                         'template': '...',
                         'title': {'text': 'Total Tracks by Genre'},
                         'xaxis': {'anchor': 'y',
                                   'categoryarray': [Rock, Latin, Metal, Alternative & Punk,
                                                     Jazz, TV Shows, Blues, Classical, Drama,
                                                     R&B/Soul, Reggae, Pop, Soundtrack,
                                                     Alternative, Hip Hop/Rap,
                                                     Electronica/Dance, Heavy Metal, World,
                                                     Sci Fi & Fantasy, Easy Listening,
                                                     Comedy, Bossa Nova, Science Fiction,
                                                     Rock And Roll, Opera],
                                   'categoryorder': 'array',
                                   'domain': [0.0, 1.0],
                                   'title': {'text': 'Name'}},
                         'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'TotalTracks'}}}
          }))
```

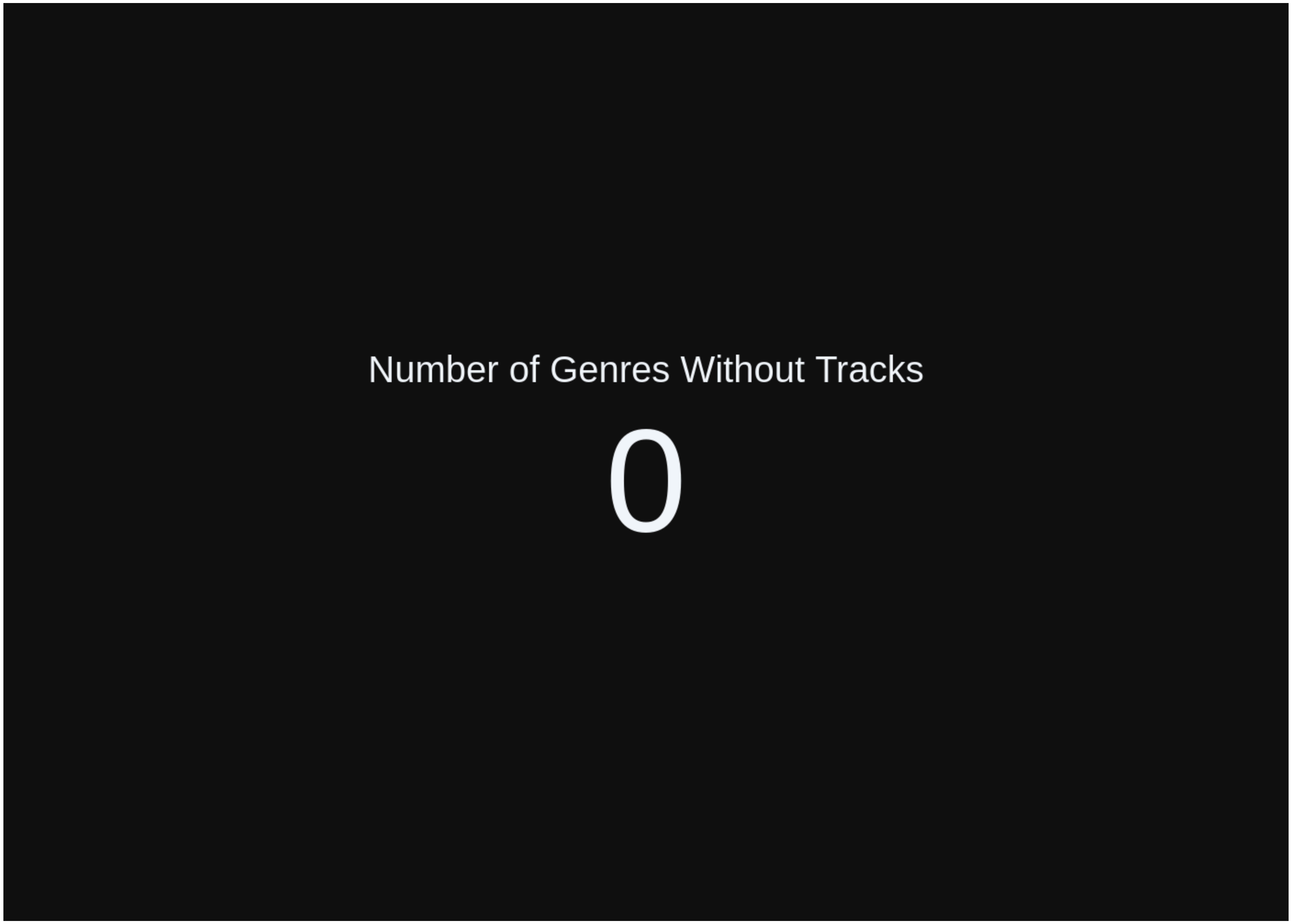
In [33]:
```
question = """
    Get all genres that do not have any tracks associated with them:
"""

vn.ask(question=question)
```

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

```
["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 NUL
L,\n    Name NVARCHAR(200)  NOT NULL,\n    AlbumId INTEGER,\n    MediaTypeId INTEGER  NOT NULL,\n    GenreI
d INTEGER,\n    Composer NVARCHAR(220),\n    Milliseconds INTEGER  NOT NULL,\n    Bytes INTEGER,\n    UnitP
rice NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (TrackId),\n    FOREIGN KEY (AlbumId) R
EFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (GenreId) REFEREN
CES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (MediaTypeId) REFERENCE
S MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_PlaylistTrac
kTrackId 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 INT
EGER  NOT NULL,\n    Title NVARCHAR(160)  NOT NULL,\n    ArtistId INTEGER  NOT NULL,\n    CONSTRAINT PK_Alb
um PRIMARY KEY  (AlbumId),\n    FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTI
ON ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_AlbumArtistId ON Album (ArtistId)\n\nCREATE TABLE PlaylistTra
ck\n(\n    PlaylistId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    CONSTRAINT PK_PlaylistTrack P
RIMARY KEY  (PlaylistId, TrackId),\n    FOREIGN KEY (PlaylistId) REFERENCES Playlist (PlaylistId) \n\t\tON
DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE
NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Artist\n(\n    ArtistId INTEGER  NOT NULL,\n    Name NVARC
HAR(120),\n    CONSTRAINT PK_Artist PRIMARY KEY  (ArtistId)\n)\n\n\n===Additional Context \n\nIn the chinoo
k database invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficient, please
generate a valid SQL query without any explanations for the question. \n2. If the provided context is almos
t sufficient but requires knowledge of a specific string in a particular column, please generate an interme
diate SQL query to find the distinct strings in that column. Prepend the query with a comment saying interm
ediate_sql \n3. If the provided context is insufficient, please explain why it can't be generated. \n4. Ple
ase use the most relevant table(s). \n5. If the question has been asked and answered before, please repeat
the answer exactly as it was given before. \n", '  \n    List all genres and the number of tracks in each g
enre:\n', 'SELECT \n    Genre.Name, \n    COUNT(Track.TrackId) AS TotalTracks \nFROM \n    Genre \nLEFT JOI
N \n    Track ON Genre.GenreId = Track.GenreId \nGROUP BY \n    Genre.GenreId, \n    Genre.Name \nORDER BY
\n    TotalTracks DESC;', '  \n    Find all tracks with a name containing "What" (case-insensitive)\n', "SE
LECT \n    Name \nFROM \n    Track\nWHERE \n    LOWER(Name) LIKE '%what%';", '  \n    List all albums and t
heir corresponding artist names  \n', 'SELECT \n    Album.Title AS AlbumTitle, \n    Artist.Name AS ArtistN
ame \nFROM \n    Album \nJOIN \n    Artist ON Album.ArtistId = Artist.ArtistId;', '  \n    Find the top 5 m
ost expensive tracks (based on unit price):\n', 'SELECT \n    Name, \n    UnitPrice \nFROM \n    Track \nOR
DER BY \n    UnitPrice DESC \nLIMIT \n    5;', 'Show me a list of tables in the SQLite database', 'SELECT
\n    Name \nFROM \n    Playlist\nJOIN \n    PlaylistTrack ON Playlist.PlaylistId = PlaylistTrack.PlaylistI
d\nWHERE \n    PlaylistTrack.TrackId IN (\n        SELECT \n            TrackId \n        FROM \n
InvoiceLine\n        WHERE \n            UnitPrice > 0.99\n    );', '  \n    Find all invoices since 2010 a
nd the total amount invoiced:\n', "SELECT \n    Invoice.InvoiceDate, \n    SUM(Invoice.Total) AS TotalInvoi
ced \nFROM \n    Invoice \nWHERE \n    Invoice.InvoiceDate >= '2010-01-01' \nGROUP BY \n    Invoice.Invoice
Date \nORDER BY \n    Invoice.InvoiceDate;", '  \n    List all invoices with a total exceeding $10:\n', 'SE
LECT \n    Invoice.InvoiceId, \n    Invoice.InvoiceDate, \n    Invoice.Total \nFROM \n    Invoice \nWHERE
```

~~~
\n    Invoice.Total > 10;', '  \n    Find the total number of invoices per country:\n', 'SELECT \n    Invoi
ce.BillingCountry, \n    COUNT(Invoice.InvoiceId) AS TotalInvoices \nFROM \n    Invoice \nGROUP BY \n    In
voice.BillingCountry \nORDER BY \n    TotalInvoices DESC;', 'How many records are in table called custome
r', 'SELECT \n    COUNT(*) \nFROM \n    Customer;', "  \n    List all employees and their reporting manage
r's name (if any):\n", "SELECT \n    e.EmployeeId, \n    e.LastName, \n    e.FirstName,\n    COALESCE(m.Las
tName, 'N/A') AS ManagerLastName, \n    COALESCE(m.FirstName, 'N/A') AS ManagerFirstName \nFROM \n    Emplo
yee e \nLEFT JOIN \n    Employee m ON e.ReportsTo = m.EmployeeId;", '  \n    Get all genres that do not hav
e any tracks associated with them:\n']
```sql
SELECT
    Name
FROM
    Genre
WHERE
    GenreId NOT IN (
        SELECT
            GenreId
        FROM
            Track
    );
```
SELECT
    Name
FROM
    Genre
WHERE
    GenreId NOT IN (
        SELECT
            GenreId
        FROM
            Track
    );
SELECT
    Name
FROM
    Genre
WHERE
    GenreId NOT IN (
        SELECT
            GenreId
        FROM
            Track
~~~

```
    );
Empty DataFrame
Columns: [Name]
Index: []
```



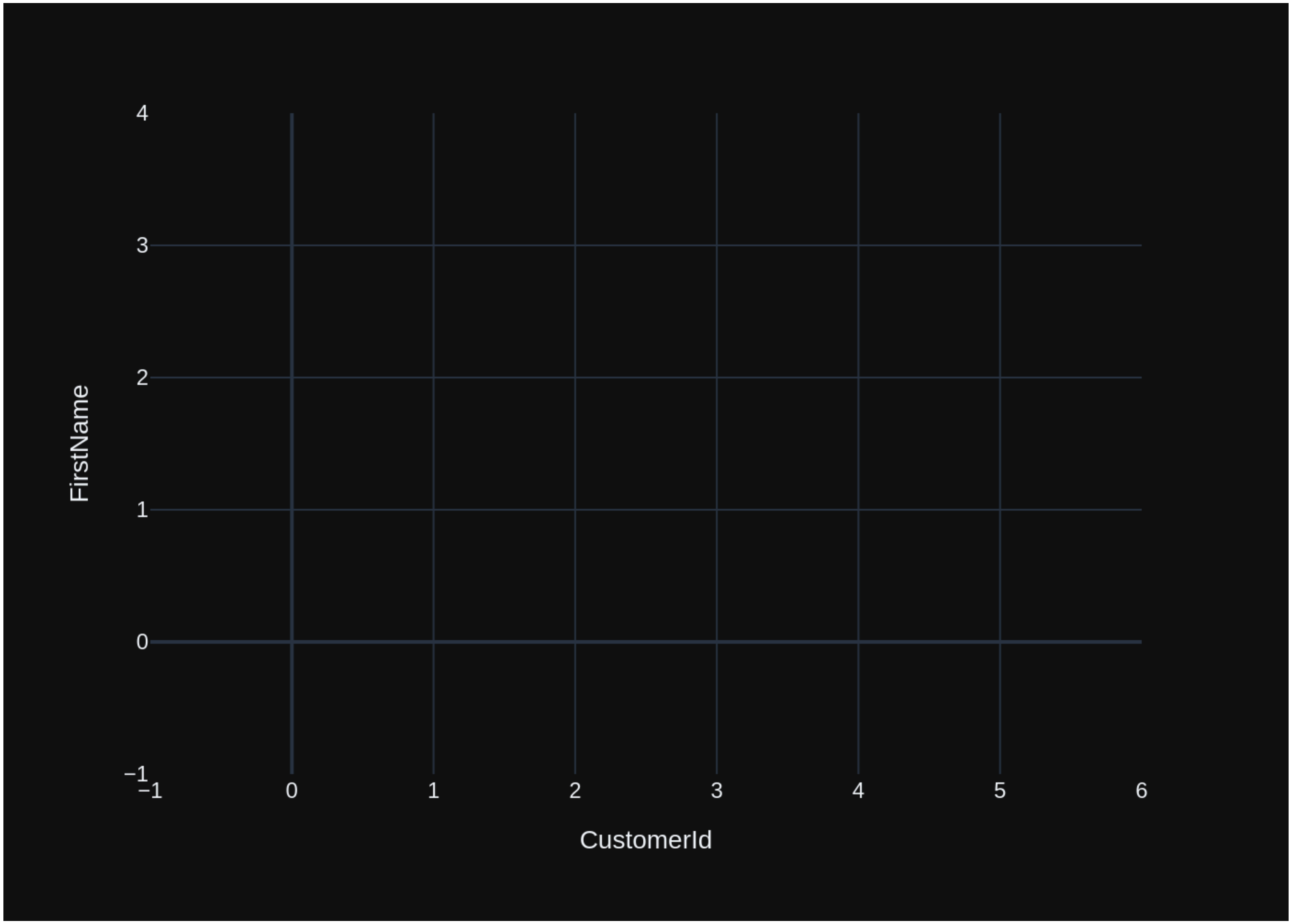
```
Out[33]: ('SELECT \n    Name \nFROM \n    Genre \nWHERE \n    GenreId NOT IN (\n        SELECT \n            GenreI
d \n        FROM \n            Track\n    );',
 Empty DataFrame
 Columns: [Name]
 Index: [],
 Figure({
     'data': [{'mode': 'number', 'title': {'text': 'Number of Genres Without Tracks'}, 'type': 'indicato
r', 'value': 0}],
     'layout': {'template': '...'}
 }))
```

```
In [34]: 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
```

```
["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    L
astName NVARCHAR(20)  NOT NULL,\n    Company NVARCHAR(80),\n    Address NVARCHAR(70),\n    City NVARCHAR(4
0),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVARCHAR(10),\n    Phone NVARCHAR(2
4),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60)  NOT NULL,\n    SupportRepId INTEGER,\n    CONSTRAINT PK
_Customer PRIMARY KEY  (CustomerId),\n    FOREIGN KEY (SupportRepId) REFERENCES Employee (EmployeeId) \n\t
\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Invoice\n(\n    InvoiceId INTEGER  NOT NULL,\n
CustomerId INTEGER  NOT NULL,\n    InvoiceDate DATETIME  NOT NULL,\n    BillingAddress NVARCHAR(70),\n    B
illingCity NVARCHAR(40),\n    BillingState NVARCHAR(40),\n    BillingCountry NVARCHAR(40),\n    BillingPost
alCode NVARCHAR(10),\n    Total NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Invoice PRIMARY KEY  (InvoiceI
d),\n    FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO A
CTION\n)\n\nCREATE TABLE InvoiceLine\n(\n    InvoiceLineId INTEGER  NOT NULL,\n    InvoiceId INTEGER  NOT N
ULL,\n    TrackId INTEGER  NOT NULL,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    Quantity INTEGER  NOT NUL
L,\n    CONSTRAINT PK_InvoiceLine PRIMARY KEY  (InvoiceLineId),\n    FOREIGN KEY (InvoiceId) REFERENCES Inv
oice (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 INTE
GER  NOT NULL,\n    LastName NVARCHAR(20)  NOT NULL,\n    FirstName NVARCHAR(20)  NOT NULL,\n    Title NVAR
CHAR(30),\n    ReportsTo INTEGER,\n    BirthDate DATETIME,\n    HireDate DATETIME,\n    Address NVARCHAR(7
0),\n    City NVARCHAR(40),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVARCHAR(1
0),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60),\n    CONSTRAINT PK_Employee PR
IMARY KEY  (EmployeeId),\n    FOREIGN KEY (ReportsTo) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO A
CTION 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 P
K_PlaylistTrack PRIMARY KEY  (PlaylistId, TrackId),\n    FOREIGN KEY (PlaylistId) REFERENCES Playlist (Play
listId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (TrackId) REFERENCES Track (TrackI
d) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Album\n(\n    AlbumId INTEGER  NOT NUL
L,\n    Title NVARCHAR(160)  NOT NULL,\n    ArtistId INTEGER  NOT NULL,\n    CONSTRAINT PK_Album PRIMARY KE
Y  (AlbumId),\n    FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON UPDATE
NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)\n\nCREATE TABLE Track\n(\n    Tr
ackId INTEGER  NOT NULL,\n    Name NVARCHAR(200)  NOT NULL,\n    AlbumId INTEGER,\n    MediaTypeId INTEGER
NOT NULL,\n    GenreId INTEGER,\n    Composer NVARCHAR(220),\n    Milliseconds INTEGER  NOT NULL,\n    Byte
s INTEGER,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (TrackId),\n    FO
REIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN
KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (M
ediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE T
ABLE Playlist\n(\n    PlaylistId INTEGER  NOT NULL,\n    Name NVARCHAR(120),\n    CONSTRAINT PK_Playlist PR
IMARY 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 conte
xt 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 b
efore. \n", '  \n    Get the total number of invoices for each customer\n', 'SELECT \n    Customer.Customer
Id, \n    Customer.FirstName, \n    Customer.LastName, \n    COUNT(Invoice.InvoiceId) AS TotalInvoices \nFR
OM \n    Customer \nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId \nGROUP BY \n    Cu
stomer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName;', 'How many customers are there', 'SE
LECT \n    COUNT(*) \nFROM \n    Customer;', 'How many records are in table called customer', 'SELECT \n
COUNT(*) \nFROM \n    Customer;', '  \n    Get the average invoice total for each customer:\n', 'SELECT \n
Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName, \n    AVG(Invoice.Total) AS Average
InvoiceTotal \nFROM \n    Customer \nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId \n
GROUP BY \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName;', '  \n    List all
invoices with a total exceeding $10:\n', 'SELECT \n    Invoice.InvoiceId, \n    Invoice.InvoiceDate, \n
Invoice.Total \nFROM \n    Invoice \nWHERE \n    Invoice.Total > 10;', '  \n    Find all invoices since 201
0 and the total amount invoiced:\n', "SELECT \n    Invoice.InvoiceDate, \n    SUM(Invoice.Total) AS TotalIn
voiced \nFROM \n    Invoice \nWHERE \n    Invoice.InvoiceDate >= '2010-01-01' \nGROUP BY \n    Invoice.Invo
iceDate \nORDER BY \n    Invoice.InvoiceDate;", '  \n    Find the total number of invoices per country:\n',
'SELECT \n    Invoice.BillingCountry, \n    COUNT(Invoice.InvoiceId) AS TotalInvoices \nFROM \n    Invoice
\nGROUP BY \n    Invoice.BillingCountry \nORDER BY \n    TotalInvoices DESC;', "  \n    List all employees
and their reporting manager's name (if any):\n", "SELECT \n    e.EmployeeId, \n    e.LastName, \n    e.Firs
tName,\n    COALESCE(m.LastName, 'N/A') AS ManagerLastName, \n    COALESCE(m.FirstName, 'N/A') AS ManagerFi
rstName \nFROM \n    Employee e \nLEFT JOIN \n    Employee m ON e.ReportsTo = m.EmployeeId;", 'Show me a li
st of tables in the SQLite database', 'SELECT \n    Name \nFROM \n    Playlist\nJOIN \n    PlaylistTrack ON
Playlist.PlaylistId = PlaylistTrack.PlaylistId\nWHERE \n    PlaylistTrack.TrackId IN (\n        SELECT \n
TrackId \n        FROM \n            InvoiceLine\n        WHERE \n            UnitPrice > 0.99\n    );', '
\n    List all albums and their corresponding artist names  \n', 'SELECT \n    Album.Title AS AlbumTitle,
\n    Artist.Name AS ArtistName \nFROM \n    Album \nJOIN \n    Artist ON Album.ArtistId = Artist.ArtistI
d;', '  \n    List all customers who have not placed any orders:\n']
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName
FROM
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
WHERE
    Invoice.InvoiceId IS NULL;
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName
FROM
```

```
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
WHERE
    Invoice.InvoiceId IS NULL;
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName
FROM
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
WHERE
    Invoice.InvoiceId IS NULL;
Empty DataFrame
Columns: [CustomerId, FirstName, LastName]
Index: []
```


4
3
2
1
0
−1
FirstName
−1
0
1
2
3
4
5
6
CustomerId

Out[34]:
```
('SELECT \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName \nFROM \n    Custome
r \nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId \nWHERE \n    Invoice.InvoiceId IS
NULL;',
 Empty DataFrame
 Columns: [CustomerId, FirstName, LastName]
 Index: [],
 Figure({
     'data': [],
     'layout': {'barmode': 'relative',
                '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': 'FirstName'}}}
 }))
```

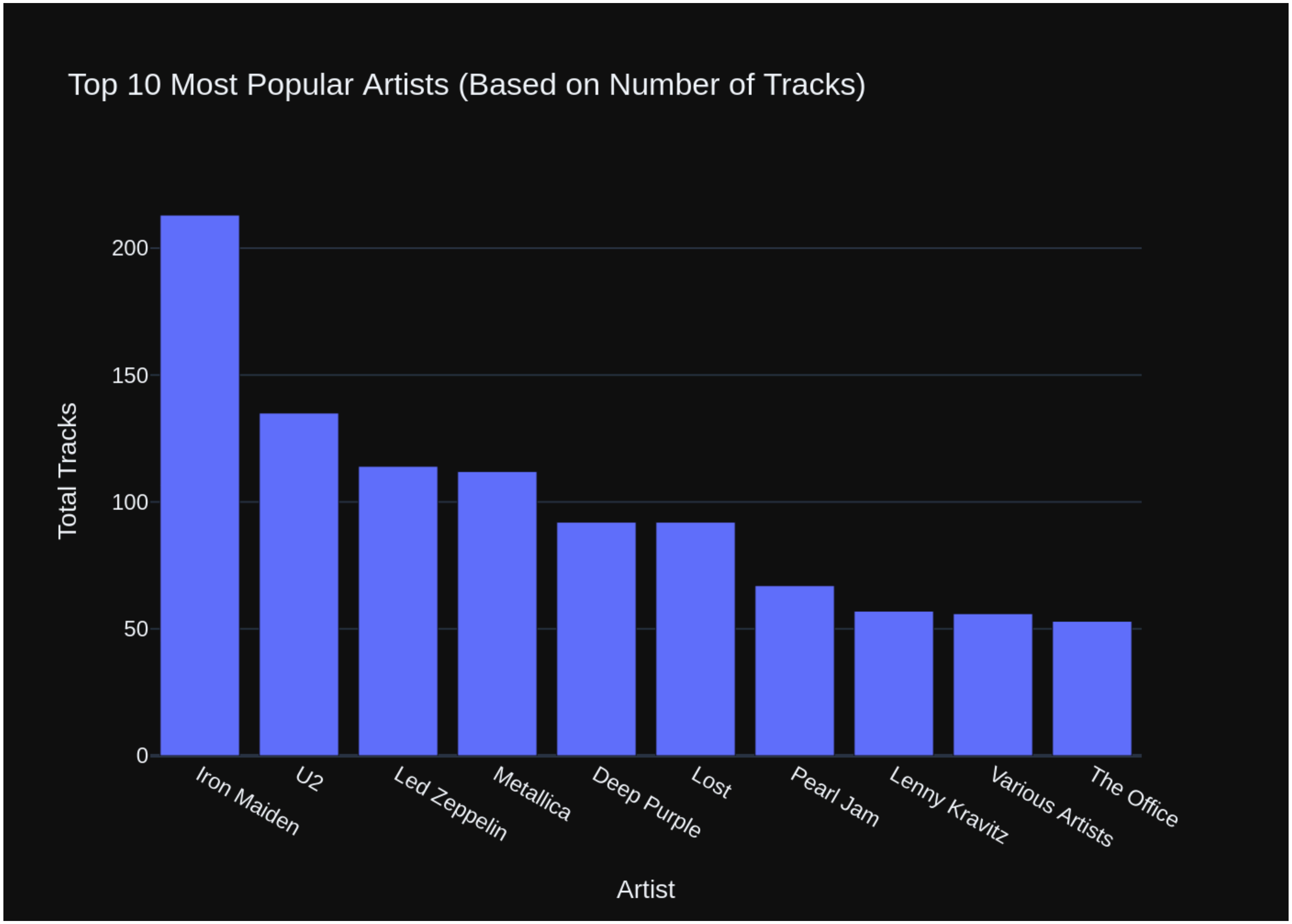
In [35]:
```
question = """
    Get the top 10 most popular artists (based on the number of tracks):
"""

vn.ask(question=question)
```

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

```
["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 INT
EGER,\n    MediaTypeId INTEGER  NOT NULL,\n    GenreId INTEGER,\n    Composer NVARCHAR(220),\n    Milliseco
nds INTEGER  NOT NULL,\n    Bytes INTEGER,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Track
PRIMARY KEY  (TrackId),\n    FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON
UPDATE NO ACTION,\n    FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE
NO ACTION,\n    FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON U
PDATE NO ACTION\n)\n\nCREATE INDEX IFK_AlbumArtistId ON Album (ArtistId)\n\nCREATE INDEX IFK_TrackAlbumId O
N 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\nCREA
TE 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 (A
rtistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_PlaylistTrackTrackId ON Playl
istTrack (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  (Pla
ylistId)\n)\n\nCREATE TABLE PlaylistTrack\n(\n    PlaylistId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT N
ULL,\n    CONSTRAINT PK_PlaylistTrack PRIMARY KEY  (PlaylistId, TrackId),\n    FOREIGN KEY (PlaylistId) REF
ERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (TrackId) REF
ERENCES 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 sufficie
nt, please generate a valid SQL query without any explanations for the question. \n2. If the provided conte
xt is almost sufficient but requires knowledge of a specific string in a particular column, please generate
an intermediate SQL query to find the distinct strings in that column. Prepend the query with a comment say
ing intermediate_sql \n3. If the provided context is insufficient, please explain why it can't be generate
d. \n4. Please use the most relevant table(s). \n5. If the question has been asked and answered before, ple
ase repeat the answer exactly as it was given before. \n", '  \n    Find the top 5 most expensive tracks (b
ased on unit price):\n', 'SELECT \n    Name, \n    UnitPrice \nFROM \n    Track \nORDER BY \n    UnitPrice
DESC \nLIMIT \n    5;', '  \n    List all genres and the number of tracks in each genre:\n', 'SELECT \n
Genre.Name, \n    COUNT(Track.TrackId) AS TotalTracks \nFROM \n    Genre \nLEFT JOIN \n    Track ON Genre.G
enreId = Track.GenreId \nGROUP BY \n    Genre.GenreId, \n    Genre.Name \nORDER BY \n    TotalTracks DES
C;', '  \n    List all albums and their corresponding artist names  \n', 'SELECT \n    Album.Title AS Album
Title, \n    Artist.Name AS ArtistName \nFROM \n    Album \nJOIN \n    Artist ON Album.ArtistId = Artist.Ar
tistId;', '  \n    Find all tracks with a name containing "What" (case-insensitive)\n', "SELECT \n    Name
\nFROM \n    Track\nWHERE \n    LOWER(Name) LIKE '%what%';", 'Show me a list of tables in the SQLite databa
se', 'SELECT \n    Name \nFROM \n    Playlist\nJOIN \n    PlaylistTrack ON Playlist.PlaylistId = PlaylistTr
ack.PlaylistId\nWHERE \n    PlaylistTrack.TrackId IN (\n        SELECT \n            TrackId \n        FROM
\n            InvoiceLine\n        WHERE \n            UnitPrice > 0.99\n    );', 'How many customers are t
here', 'SELECT \n    COUNT(*) \nFROM \n    Customer;', '  \n    List all invoices with a total exceeding $1
0:\n', 'SELECT \n    Invoice.InvoiceId, \n    Invoice.InvoiceDate, \n    Invoice.Total \nFROM \n    Invoice
\nWHERE \n    Invoice.Total > 10;', 'How many records are in table called customer', 'SELECT \n    COUNT(*)
\nFROM \n    Customer;', '  \n    Get the average invoice total for each customer:\n', 'SELECT \n    Custom
```

~~~
er.CustomerId, \n    Customer.FirstName, \n    Customer.LastName, \n    AVG(Invoice.Total) AS AverageInvoic
eTotal \nFROM \n    Customer \nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId \nGROUP
BY \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName;', '  \n    Find the total
number of invoices per country:\n', 'SELECT \n    Invoice.BillingCountry, \n    COUNT(Invoice.InvoiceId) AS
TotalInvoices \nFROM \n    Invoice \nGROUP BY \n    Invoice.BillingCountry \nORDER BY \n    TotalInvoices D
ESC;', '  \n    Get the top 10 most popular artists (based on the number of tracks):\n']
```sql
SELECT
    Artist.Name,
    COUNT(Track.TrackId) AS TotalTracks
FROM
    Artist
LEFT JOIN
    Album ON Artist.ArtistId = Album.ArtistId
LEFT JOIN
    Track ON Album.AlbumId = Track.AlbumId
GROUP BY
    Artist.ArtistId,
    Artist.Name
ORDER BY
    TotalTracks DESC
LIMIT
    10;
~~~
SELECT
    Artist.Name,
    COUNT(Track.TrackId) AS TotalTracks
FROM
    Artist
LEFT JOIN
    Album ON Artist.ArtistId = Album.ArtistId
LEFT JOIN
    Track ON Album.AlbumId = Track.AlbumId
GROUP BY
    Artist.ArtistId,
    Artist.Name
ORDER BY
    TotalTracks DESC
LIMIT
    10;
SELECT
    Artist.Name,
````
````

```
    COUNT(Track.TrackId) AS TotalTracks
FROM
    Artist
LEFT JOIN
    Album ON Artist.ArtistId = Album.ArtistId
LEFT JOIN
    Track ON Album.AlbumId = Track.AlbumId
GROUP BY
    Artist.ArtistId,
    Artist.Name
ORDER BY
    TotalTracks DESC
LIMIT
    10;
              Name  TotalTracks
0      Iron Maiden          213
1               U2          135
2     Led Zeppelin          114
3        Metallica          112
4      Deep Purple           92
5             Lost           92
6        Pearl Jam           67
7    Lenny Kravitz           57
8  Various Artists           56
9       The Office           53
```


Top 10 Most Popular Artists (Based on Number of Tracks)
Total Tracks
0
50
100
150
200
Iron Maiden
U2
Led Zeppelin
Metallica
Deep Purple
Lost
Pearl Jam
Lenny Kravitz
Various Artists
The Office
Artist

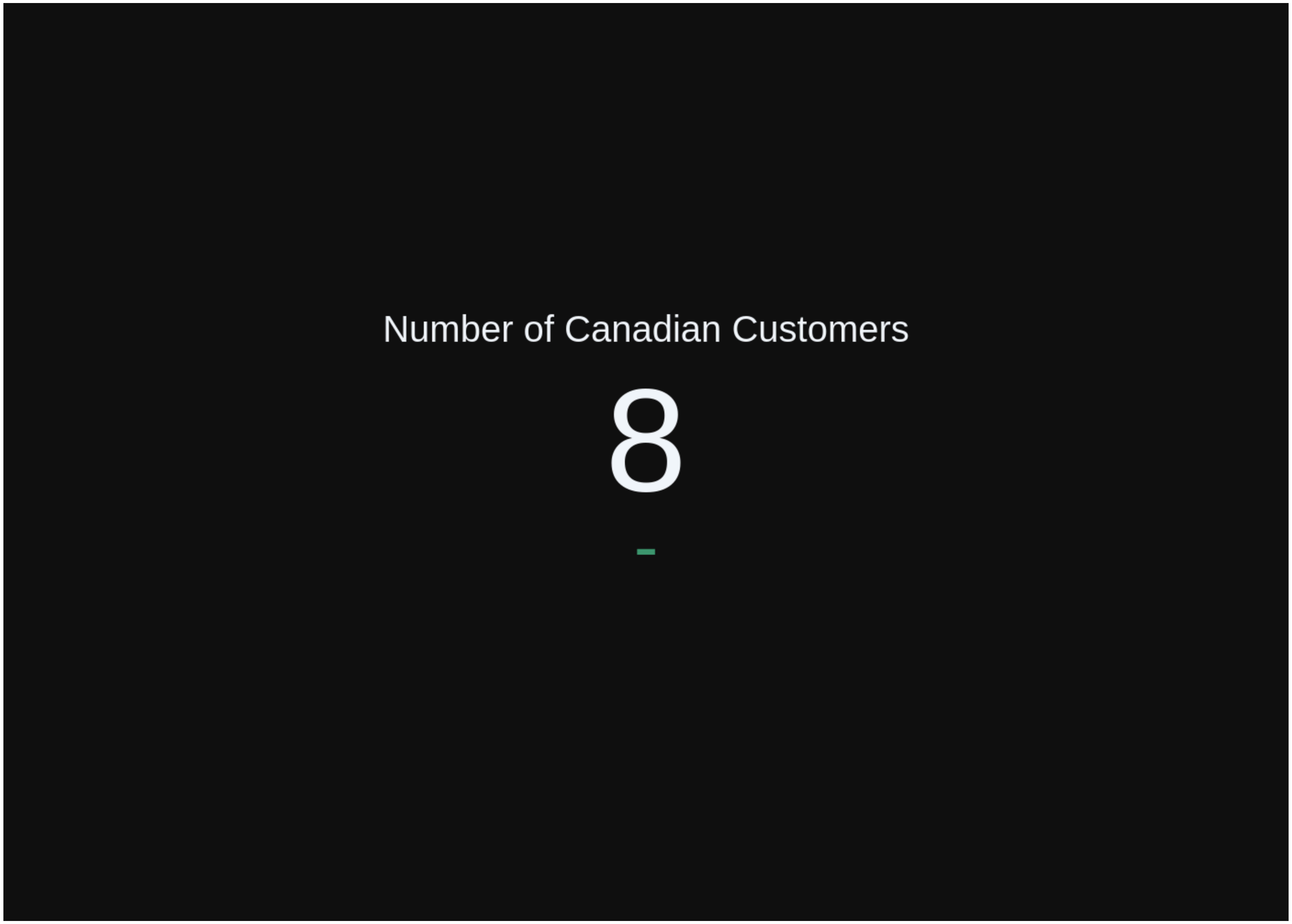
```
Out[35]: ('SELECT \n    Artist.Name, \n    COUNT(Track.TrackId) AS TotalTracks \nFROM \n    Artist \nLEFT JOIN \n
         Album ON Artist.ArtistId = Album.ArtistId \nLEFT JOIN \n    Track ON Album.AlbumId = Track.AlbumId \nGROUP
         BY \n    Artist.ArtistId, \n    Artist.Name \nORDER BY \n    TotalTracks DESC \nLIMIT \n    10;',
                       Name  TotalTracks
          0      Iron Maiden          213
          1               U2          135
          2     Led Zeppelin          114
          3        Metallica          112
          4      Deep Purple           92
          5             Lost           92
          6        Pearl Jam           67
          7    Lenny Kravitz           57
          8  Various Artists           56
          9       The Office           53,
          Figure({
              'data': [{'type': 'bar',
                        'x': array(['Iron Maiden', 'U2', 'Led Zeppelin', 'Metallica', 'Deep Purple', 'Lost',
                                    'Pearl Jam', 'Lenny Kravitz', 'Various Artists', 'The Office'],
                                   dtype=object),
                        'y': array([213, 135, 114, 112,  92,  92,  67,  57,  56,  53])}],
              'layout': {'template': '...',
                         'title': {'text': 'Top 10 Most Popular Artists (Based on Number of Tracks)'},
                         'xaxis': {'title': {'text': 'Artist'}},
                         'yaxis': {'title': {'text': 'Total Tracks'}}}
          }))
```

```
In [36]: 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
```

```
["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    L
astName NVARCHAR(20)  NOT NULL,\n    Company NVARCHAR(80),\n    Address NVARCHAR(70),\n    City NVARCHAR(4
0),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVARCHAR(10),\n    Phone NVARCHAR(2
4),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60)  NOT NULL,\n    SupportRepId INTEGER,\n    CONSTRAINT PK
_Customer PRIMARY KEY  (CustomerId),\n    FOREIGN KEY (SupportRepId) REFERENCES Employee (EmployeeId) \n\t
\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_CustomerSupportRepId ON Customer (SupportR
epId)\n\nCREATE TABLE Invoice\n(\n    InvoiceId INTEGER  NOT NULL,\n    CustomerId INTEGER  NOT NULL,\n
InvoiceDate DATETIME  NOT NULL,\n    BillingAddress NVARCHAR(70),\n    BillingCity NVARCHAR(40),\n    Billi
ngState NVARCHAR(40),\n    BillingCountry NVARCHAR(40),\n    BillingPostalCode NVARCHAR(10),\n    Total NUM
ERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Invoice PRIMARY KEY  (InvoiceId),\n    FOREIGN KEY (CustomerId) RE
FERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_Invoice
CustomerId ON Invoice (CustomerId)\n\nCREATE TABLE Employee\n(\n    EmployeeId INTEGER  NOT NULL,\n    Last
Name 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(4
0),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVARCHAR(10),\n    Phone NVARCHAR(2
4),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60),\n    CONSTRAINT PK_Employee PRIMARY KEY  (EmployeeI
d),\n    FOREIGN KEY (ReportsTo) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO AC
TION\n)\n\nCREATE TABLE InvoiceLine\n(\n    InvoiceLineId INTEGER  NOT NULL,\n    InvoiceId INTEGER  NOT NU
LL,\n    TrackId INTEGER  NOT NULL,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    Quantity INTEGER  NOT NUL
L,\n    CONSTRAINT PK_InvoiceLine PRIMARY KEY  (InvoiceLineId),\n    FOREIGN KEY (InvoiceId) REFERENCES Inv
oice (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 Invoic
eLine (TrackId)\n\nCREATE INDEX IFK_InvoiceLineInvoiceId ON InvoiceLine (InvoiceId)\n\nCREATE INDEX IFK_Emp
loyeeReportsTo ON Employee (ReportsTo)\n\nCREATE TABLE PlaylistTrack\n(\n    PlaylistId INTEGER  NOT NUL
L,\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===A
dditional Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the pro
vided 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 particul
ar column, please generate an intermediate SQL query to find the distinct strings in that column. Prepend t
he query with a comment saying intermediate_sql \n3. If the provided context is insufficient, please explai
n why it can't be generated. \n4. Please use the most relevant table(s). \n5. If the question has been aske
d and answered before, please repeat the answer exactly as it was given before. \n", '  \n    Get the total
number of invoices for each customer\n', 'SELECT \n    Customer.CustomerId, \n    Customer.FirstName, \n
Customer.LastName, \n    COUNT(Invoice.InvoiceId) AS TotalInvoices \nFROM \n    Customer \nLEFT JOIN \n
Invoice ON Customer.CustomerId = Invoice.CustomerId \nGROUP BY \n    Customer.CustomerId, \n    Customer.Fi
rstName, \n    Customer.LastName;', '  \n    Find the total number of invoices per country:\n', 'SELECT \n
Invoice.BillingCountry, \n    COUNT(Invoice.InvoiceId) AS TotalInvoices \nFROM \n    Invoice \nGROUP BY \n
Invoice.BillingCountry \nORDER BY \n    TotalInvoices DESC;', 'How many customers are there', 'SELECT \n
COUNT(*) \nFROM \n    Customer;', 'How many records are in table called customer', 'SELECT \n    COUNT(*)
\nFROM \n    Customer;', '  \n    Get the average invoice total for each customer:\n', 'SELECT \n    Custom
er.CustomerId, \n    Customer.FirstName, \n    Customer.LastName, \n    AVG(Invoice.Total) AS AverageInvoic
eTotal \nFROM \n    Customer \nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId \nGROUP
BY \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName;', "  \n    List all employ
ees and their reporting manager's name (if any):\n", "SELECT \n    e.EmployeeId, \n    e.LastName, \n    e.
FirstName,\n    COALESCE(m.LastName, 'N/A') AS ManagerLastName, \n    COALESCE(m.FirstName, 'N/A') AS Manag
erFirstName \nFROM \n    Employee e \nLEFT JOIN \n    Employee m ON e.ReportsTo = m.EmployeeId;", '  \n
Find all invoices since 2010 and the total amount invoiced:\n', "SELECT \n    Invoice.InvoiceDate, \n    SU
M(Invoice.Total) AS TotalInvoiced \nFROM \n    Invoice \nWHERE \n    Invoice.InvoiceDate >= '2010-01-01' \n
GROUP BY \n    Invoice.InvoiceDate \nORDER BY \n    Invoice.InvoiceDate;", 'Show me a list of tables in the
SQLite database', 'SELECT \n    Name \nFROM \n    Playlist\nJOIN \n    PlaylistTrack ON Playlist.PlaylistId
= PlaylistTrack.PlaylistId\nWHERE \n    PlaylistTrack.TrackId IN (\n        SELECT \n            TrackId \n
FROM \n            InvoiceLine\n        WHERE \n            UnitPrice > 0.99\n    );', '  \n    List all in
voices with a total exceeding $10:\n', 'SELECT \n    Invoice.InvoiceId, \n    Invoice.InvoiceDate, \n    In
voice.Total \nFROM \n    Invoice \nWHERE \n    Invoice.Total > 10;', '  \n    Find the top 5 most expensive
tracks (based on unit price):\n', 'SELECT \n    Name, \n    UnitPrice \nFROM \n    Track \nORDER BY \n    U
nitPrice DESC \nLIMIT \n    5;', '  \n     List all customers from Canada and their email addresses:\n']
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    Customer.Email
FROM
    Customer
WHERE
    Customer.Country = 'Canada';
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    Customer.Email
FROM
    Customer
WHERE
    Customer.Country = 'Canada';
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    Customer.Email
FROM
```

```
    Customer
WHERE
    Customer.Country = 'Canada';
   CustomerId FirstName  LastName                   Email
0           3  François  Tremblay     ftremblay@gmail.com
1          14      Mark   Philips      mphilips12@shaw.ca
2          15  Jennifer  Peterson     jenniferp@rogers.ca
3          29    Robert     Brown        robbrown@shaw.ca
4          30    Edward   Francis     edfrancis@yachoo.ca
5          31    Martha      Silk    marthasilk@gmail.com
6          32     Aaron  Mitchell  aaronmitchell@yahoo.ca
7          33    Ellie  Sullivan  ellie.sullivan@shaw.ca
```


Number of Canadian Customers
8
-

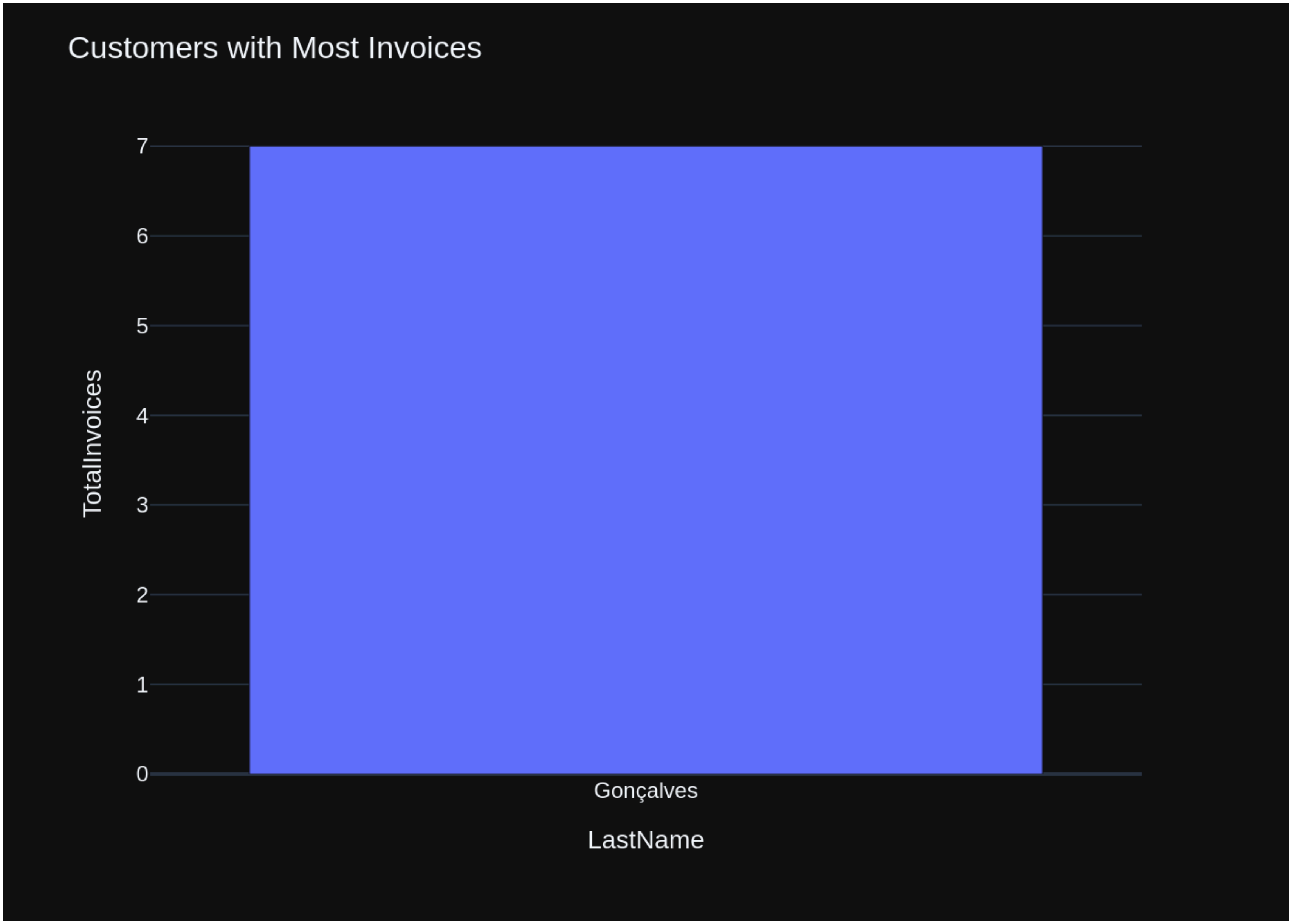
Out[36]:
```
("SELECT \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName, \n    Customer.Emai
l \nFROM \n    Customer \nWHERE \n    Customer.Country = 'Canada';",
    CustomerId FirstName  LastName                   Email
 0           3  François  Tremblay     ftremblay@gmail.com
 1          14      Mark   Philips      mphilips12@shaw.ca
 2          15  Jennifer  Peterson     jenniferp@rogers.ca
 3          29    Robert     Brown        robbrown@shaw.ca
 4          30    Edward   Francis     edfrancis@yachoo.ca
 5          31    Martha      Silk    marthasilk@gmail.com
 6          32     Aaron  Mitchell  aaronmitchell@yahoo.ca
 7          33     Ellie  Sullivan  ellie.sullivan@shaw.ca,
 Figure({
     'data': [{'delta': {'reference': 8},
               'mode': 'number+delta',
               'title': {'text': 'Number of Canadian Customers'},
               'type': 'indicator',
               'value': 8}],
     'layout': {'template': '...'}
 }))
```

In [37]:
```python
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
```

```
["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 INTE
GER  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(4
0),\n    BillingPostalCode NVARCHAR(10),\n    Total NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Invoice PRI
MARY KEY  (InvoiceId),\n    FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO AC
TION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceLineInvoiceId ON InvoiceLine (InvoiceId)\n\nCREATE T
ABLE InvoiceLine\n(\n    InvoiceLineId INTEGER  NOT NULL,\n    InvoiceId INTEGER  NOT NULL,\n    TrackId IN
TEGER  NOT NULL,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    Quantity INTEGER  NOT NULL,\n    CONSTRAINT P
K_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\n
CREATE TABLE Customer\n(\n    CustomerId INTEGER  NOT NULL,\n    FirstName NVARCHAR(40)  NOT NULL,\n    Las
tName NVARCHAR(20)  NOT NULL,\n    Company NVARCHAR(80),\n    Address NVARCHAR(70),\n    City NVARCHAR(4
0),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVARCHAR(10),\n    Phone NVARCHAR(2
4),\n    Fax NVARCHAR(24),\n    Email NVARCHAR(60)  NOT NULL,\n    SupportRepId INTEGER,\n    CONSTRAINT PK
_Customer PRIMARY KEY  (CustomerId),\n    FOREIGN KEY (SupportRepId) REFERENCES Employee (EmployeeId) \n\t
\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_CustomerSupportRepId ON Customer (SupportR
epId)\n\nCREATE TABLE Employee\n(\n    EmployeeId INTEGER  NOT NULL,\n    LastName NVARCHAR(20)  NOT NUL
L,\n    FirstName NVARCHAR(20)  NOT NULL,\n    Title NVARCHAR(30),\n    ReportsTo INTEGER,\n    BirthDate D
ATETIME,\n    HireDate DATETIME,\n    Address NVARCHAR(70),\n    City NVARCHAR(40),\n    State NVARCHAR(4
0),\n    Country NVARCHAR(40),\n    PostalCode NVARCHAR(10),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(2
4),\n    Email NVARCHAR(60),\n    CONSTRAINT PK_Employee PRIMARY KEY  (EmployeeId),\n    FOREIGN KEY (Repor
tsTo) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK
_EmployeeReportsTo ON Employee (ReportsTo)\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\n\n===Additional Context \n\nIn the ch
inook database invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficient, pl
ease 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 in
termediate SQL query to find the distinct strings in that column. Prepend the query with a comment saying i
ntermediate_sql \n3. If the provided context is insufficient, please explain why it can't be generated. \n
4. Please use the most relevant table(s). \n5. If the question has been asked and answered before, please r
epeat the answer exactly as it was given before. \n", '  \n    Get the total number of invoices for each cu
stomer\n', 'SELECT \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName, \n    COUN
T(Invoice.InvoiceId) AS TotalInvoices \nFROM \n    Customer \nLEFT JOIN \n    Invoice ON Customer.CustomerI
d = Invoice.CustomerId \nGROUP BY \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastN
ame;', '  \n    Get the average invoice total for each customer:\n', 'SELECT \n    Customer.CustomerId, \n
Customer.FirstName, \n    Customer.LastName, \n    AVG(Invoice.Total) AS AverageInvoiceTotal \nFROM \n    C
ustomer \nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId \nGROUP BY \n    Customer.Cus
tomerId, \n    Customer.FirstName, \n    Customer.LastName;', '  \n    List all invoices with a total excee
ding $10:\n', 'SELECT \n    Invoice.InvoiceId, \n    Invoice.InvoiceDate, \n    Invoice.Total \nFROM \n    
Invoice \nWHERE \n    Invoice.Total > 10;', '  \n    Find the total number of invoices per country:\n', 'SE
LECT \n    Invoice.BillingCountry, \n    COUNT(Invoice.InvoiceId) AS TotalInvoices \nFROM \n    Invoice \nG
ROUP BY \n    Invoice.BillingCountry \nORDER BY \n    TotalInvoices DESC;', '  \n    Find all invoices sinc
e 2010 and the total amount invoiced:\n', "SELECT \n    Invoice.InvoiceDate, \n    SUM(Invoice.Total) AS To
talInvoiced \nFROM \n    Invoice \nWHERE \n    Invoice.InvoiceDate >= '2010-01-01' \nGROUP BY \n    Invoic
e.InvoiceDate \nORDER BY \n    Invoice.InvoiceDate;", '  \n    Find the top 5 most expensive tracks (based 
on unit price):\n', 'SELECT \n    Name, \n    UnitPrice \nFROM \n    Track \nORDER BY \n    UnitPrice DESC 
\nLIMIT \n    5;', 'How many customers are there', 'SELECT \n    COUNT(*) \nFROM \n    Customer;', 'How man
y records are in table called customer', 'SELECT \n    COUNT(*) \nFROM \n    Customer;', '  \n     List all
customers from Canada and their email addresses:\n', "SELECT \n    Customer.CustomerId, \n    Customer.Firs
tName, \n    Customer.LastName, \n    Customer.Email \nFROM \n    Customer \nWHERE \n    Customer.Country =
'Canada';", 'Show me a list of tables in the SQLite database', 'SELECT \n    Name \nFROM \n    Playlist\nJO
IN \n    PlaylistTrack ON Playlist.PlaylistId = PlaylistTrack.PlaylistId\nWHERE \n    PlaylistTrack.TrackId
IN (\n        SELECT \n            TrackId \n        FROM \n            InvoiceLine\n        WHERE \n
UnitPrice > 0.99\n    );', '  \n     Find the customer with the most invoices \n']
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    COUNT(Invoice.InvoiceId) AS TotalInvoices
FROM
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
GROUP BY
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName
ORDER BY
    TotalInvoices DESC
LIMIT
    1;
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    COUNT(Invoice.InvoiceId) AS TotalInvoices
```

```
FROM
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
GROUP BY
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName
ORDER BY
    TotalInvoices DESC
LIMIT
    1;
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    COUNT(Invoice.InvoiceId) AS TotalInvoices
FROM
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
GROUP BY
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName
ORDER BY
    TotalInvoices DESC
LIMIT
    1;
   CustomerId FirstName   LastName  TotalInvoices
0           1      Luís  Gonçalves              7
```


Customers with Most Invoices
TotalInvoices
0
1
2
3
4
5
6
7
Gonçalves
LastName

```
Out[37]: ('SELECT \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName, \n    COUNT(Invoic
         e.InvoiceId) AS TotalInvoices \nFROM \n    Customer \nLEFT JOIN \n    Invoice ON Customer.CustomerId = Inv
         oice.CustomerId \nGROUP BY \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName \n
         ORDER BY \n    TotalInvoices DESC \nLIMIT \n    1;',
             CustomerId FirstName   LastName  TotalInvoices
          0           1      Luís  Gonçalves              7,
          Figure({
              'data': [{'alignmentgroup': 'True',
                        'hovertemplate': 'LastName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
                        'legendgroup': '',
                        'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
                        'name': '',
                        'offsetgroup': '',
                        'orientation': 'v',
                        'showlegend': False,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Gonçalves'], dtype=object),
                        'xaxis': 'x',
                        'y': array([7]),
                        'yaxis': 'y'}],
              'layout': {'barmode': 'relative',
                         'legend': {'tracegroupgap': 0},
                         'margin': {'t': 60},
                         'template': '...',
                         'title': {'text': 'Customers with Most Invoices'},
                         'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'LastName'}},
                         'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'TotalInvoices'}}}
          }))
```

```
In [ ]:
```

Advanced SQL questions

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

         vn.ask(question=question)
```

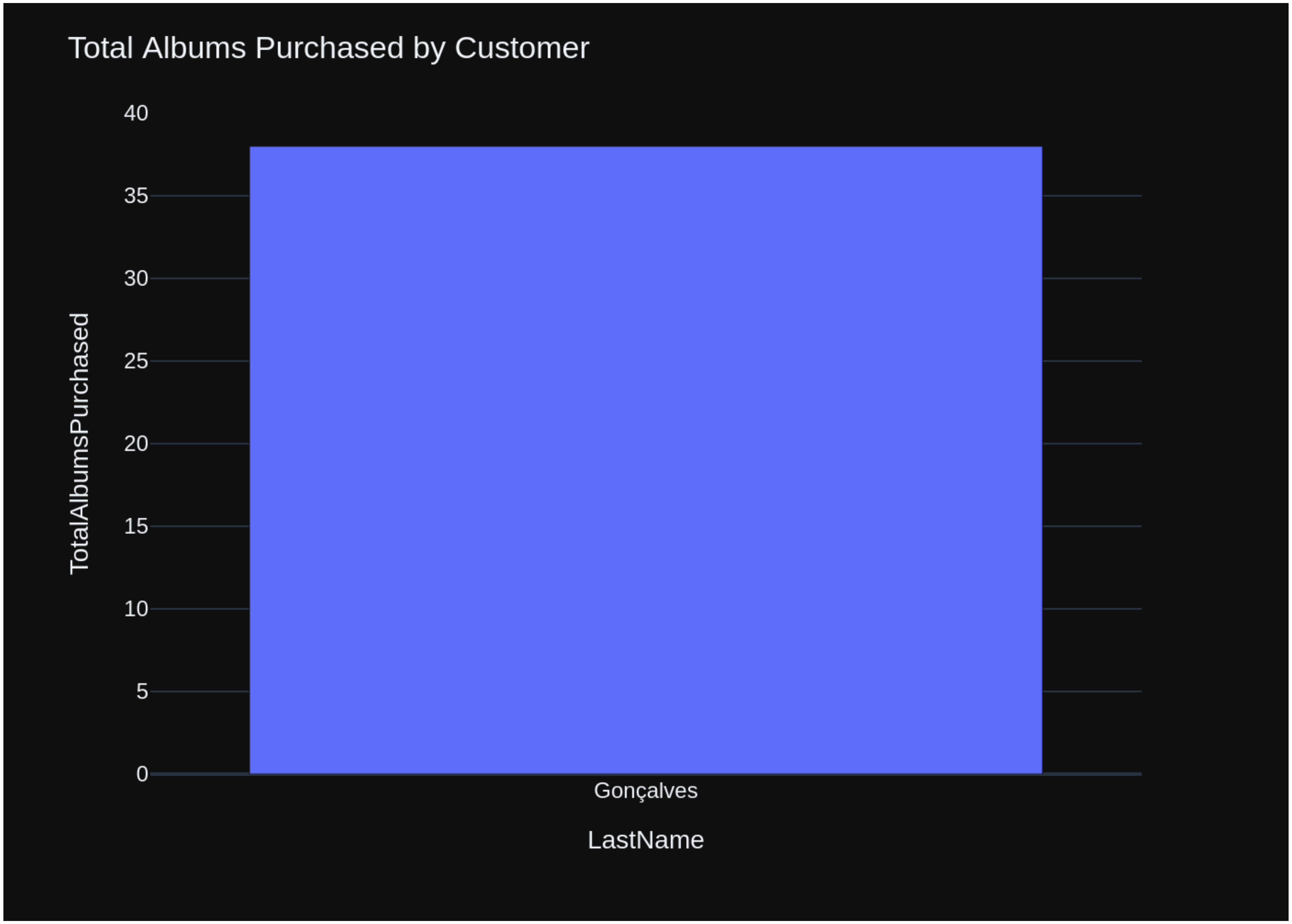
```
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
["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 INT
EGER,\n    MediaTypeId INTEGER  NOT NULL,\n    GenreId INTEGER,\n    Composer NVARCHAR(220),\n    Milliseco
nds INTEGER  NOT NULL,\n    Bytes INTEGER,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Track
PRIMARY KEY  (TrackId),\n    FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON
UPDATE NO ACTION,\n    FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE
NO ACTION,\n    FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON U
PDATE NO ACTION\n)\n\nCREATE TABLE Album\n(\n    AlbumId INTEGER  NOT NULL,\n    Title NVARCHAR(160)  NOT N
ULL,\n    ArtistId INTEGER  NOT NULL,\n    CONSTRAINT PK_Album PRIMARY KEY  (AlbumId),\n    FOREIGN KEY (Ar
tistId) REFERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_A
lbumArtistId 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    Q
uantity INTEGER  NOT NULL,\n    CONSTRAINT PK_InvoiceLine PRIMARY KEY  (InvoiceLineId),\n    FOREIGN KEY (I
nvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (T
rackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_Inv
oiceCustomerId ON Invoice (CustomerId)\n\nCREATE TABLE Invoice\n(\n    InvoiceId INTEGER  NOT NULL,\n    Cu
stomerId INTEGER  NOT NULL,\n    InvoiceDate DATETIME  NOT NULL,\n    BillingAddress NVARCHAR(70),\n    Bil
lingCity NVARCHAR(40),\n    BillingState NVARCHAR(40),\n    BillingCountry NVARCHAR(40),\n    BillingPostal
Code NVARCHAR(10),\n    Total NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Invoice PRIMARY KEY  (InvoiceI
d),\n    FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO A
CTION\n)\n\nCREATE INDEX IFK_TrackAlbumId ON Track (AlbumId)\n\nCREATE INDEX IFK_InvoiceLineInvoiceId ON In
voiceLine (InvoiceId)\n\nCREATE INDEX IFK_InvoiceLineTrackId ON InvoiceLine (TrackId)\n\nCREATE TABLE Playl
istTrack\n(\n    PlaylistId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    CONSTRAINT PK_PlaylistT
rack 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 mean
s order\n\n===Response Guidelines \n1. If the provided context is sufficient, please generate a valid SQL q
uery without any explanations for the question. \n2. If the provided context is almost sufficient but requi
res knowledge of a specific string in a particular column, please generate an intermediate SQL query to fin
d 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 releva
nt table(s). \n5. If the question has been asked and answered before, please repeat the answer exactly as i
t was given before. \n", '  \n     Find the customer with the most invoices \n', 'SELECT \n    Customer.Cus
tomerId, \n    Customer.FirstName, \n    Customer.LastName, \n    COUNT(Invoice.InvoiceId) AS TotalInvoices
\nFROM \n    Customer \nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId \nGROUP BY \n
Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName \nORDER BY \n    TotalInvoices DESC
\nLIMIT \n    1;', '  \n    Get the top 10 most popular artists (based on the number of tracks):\n', 'SELEC
T \n    Artist.Name, \n    COUNT(Track.TrackId) AS TotalTracks \nFROM \n    Artist \nLEFT JOIN \n    Album
ON Artist.ArtistId = Album.ArtistId \nLEFT JOIN \n    Track ON Album.AlbumId = Track.AlbumId \nGROUP BY \n
Artist.ArtistId, \n    Artist.Name \nORDER BY \n    TotalTracks DESC \nLIMIT \n    10;', '  \n    Get the t
otal number of invoices for each customer\n', 'SELECT \n    Customer.CustomerId, \n    Customer.FirstName,
```

~~~
\n    Customer.LastName, \n    COUNT(Invoice.InvoiceId) AS TotalInvoices \nFROM \n    Customer \nLEFT JOIN
\n    Invoice ON Customer.CustomerId = Invoice.CustomerId \nGROUP BY \n    Customer.CustomerId, \n    Custo
mer.FirstName, \n    Customer.LastName;', '  \n    Find all invoices since 2010 and the total amount invoic
ed:\n', "SELECT \n    Invoice.InvoiceDate, \n    SUM(Invoice.Total) AS TotalInvoiced \nFROM \n    Invoice
\nWHERE \n    Invoice.InvoiceDate >= '2010-01-01' \nGROUP BY \n    Invoice.InvoiceDate \nORDER BY \n    Inv
oice.InvoiceDate;", '  \n    List all invoices with a total exceeding $10:\n', 'SELECT \n    Invoice.Invoic
eId, \n    Invoice.InvoiceDate, \n    Invoice.Total \nFROM \n    Invoice \nWHERE \n    Invoice.Total > 1
0;', '  \n    Find the top 5 most expensive tracks (based on unit price):\n', 'SELECT \n    Name, \n    Uni
tPrice \nFROM \n    Track \nORDER BY \n    UnitPrice DESC \nLIMIT \n    5;', '  \n    Find the total number
of invoices per country:\n', 'SELECT \n    Invoice.BillingCountry, \n    COUNT(Invoice.InvoiceId) AS TotalI
nvoices \nFROM \n    Invoice \nGROUP BY \n    Invoice.BillingCountry \nORDER BY \n    TotalInvoices DESC;',
'  \n    Get the average invoice total for each customer:\n', 'SELECT \n    Customer.CustomerId, \n    Cust
omer.FirstName, \n    Customer.LastName, \n    AVG(Invoice.Total) AS AverageInvoiceTotal \nFROM \n    Custo
mer \nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId \nGROUP BY \n    Customer.Custome
rId, \n    Customer.FirstName, \n    Customer.LastName;', '  \n    List all albums and their corresponding
artist names  \n', 'SELECT \n    Album.Title AS AlbumTitle, \n    Artist.Name AS ArtistName \nFROM \n    Al
bum \nJOIN \n    Artist ON Album.ArtistId = Artist.ArtistId;', '  \n    List all genres and the number of t
racks in each genre:\n', 'SELECT \n    Genre.Name, \n    COUNT(Track.TrackId) AS TotalTracks \nFROM \n    G
enre \nLEFT JOIN \n    Track ON Genre.GenreId = Track.GenreId \nGROUP BY \n    Genre.GenreId, \n    Genre.N
ame \nORDER BY \n    TotalTracks DESC;', '  \n     Find the customer who bought the most albums in total qu
antity (across all invoices): \n']
```sql
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    SUM(InvoiceLine.Quantity) AS TotalAlbumsPurchased
FROM
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
LEFT JOIN
    InvoiceLine ON Invoice.InvoiceId = InvoiceLine.InvoiceId
WHERE
    InvoiceLine.TrackId IN (
        SELECT
            TrackId
        FROM
            Track
        WHERE
            AlbumId IS NOT NULL
    )
~~~

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

```
FROM
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
LEFT JOIN
    InvoiceLine ON Invoice.InvoiceId = InvoiceLine.InvoiceId
WHERE
    InvoiceLine.TrackId IN (
        SELECT
            TrackId
        FROM
            Track
        WHERE
            AlbumId IS NOT NULL
    )
GROUP BY
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName
ORDER BY
    TotalAlbumsPurchased DESC
LIMIT
    1;
   CustomerId FirstName   LastName  TotalAlbumsPurchased
0           1      Luís  Gonçalves                    38
```


Total Albums Purchased by Customer
TotalAlbumsPurchased
40
35
30
25
20
15
10
5
0
Gonçalves
LastName

```
Out[38]: ('SELECT \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName, \n    SUM(InvoiceLi
         ne.Quantity) AS TotalAlbumsPurchased \nFROM \n    Customer \nLEFT JOIN \n    Invoice ON Customer.CustomerI
         d = Invoice.CustomerId \nLEFT JOIN \n    InvoiceLine ON Invoice.InvoiceId = InvoiceLine.InvoiceId \nWHERE
         \n    InvoiceLine.TrackId IN (\n        SELECT \n            TrackId \n        FROM \n            Track \n
         WHERE \n            AlbumId IS NOT NULL\n    ) \nGROUP BY \n    Customer.CustomerId, \n    Customer.FirstN
         ame, \n    Customer.LastName \nORDER BY \n    TotalAlbumsPurchased DESC \nLIMIT \n    1;',
             CustomerId FirstName   LastName  TotalAlbumsPurchased
          0           1      Luís  Gonçalves                    38,
          Figure({
              'data': [{'alignmentgroup': 'True',
                        'hovertemplate': 'LastName=%{x}<br>TotalAlbumsPurchased=%{y}<extra></extra>',
                        'legendgroup': '',
                        'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
                        'name': '',
                        'offsetgroup': '',
                        'orientation': 'v',
                        'showlegend': False,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Gonçalves'], dtype=object),
                        'xaxis': 'x',
                        'y': array([38]),
                        'yaxis': 'y'}],
              'layout': {'barmode': 'relative',
                         'legend': {'tracegroupgap': 0},
                         'margin': {'t': 60},
                         'template': '...',
                         'title': {'text': 'Total Albums Purchased by Customer'},
                         'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'LastName'}},
                         'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'TotalAlbumsPurchased'}}}
          }))
```

```
In [39]: question = """
              Find the top 5 customer who bought the most albums in total quantity (across all invoices):
         """

         vn.ask(question=question)
```

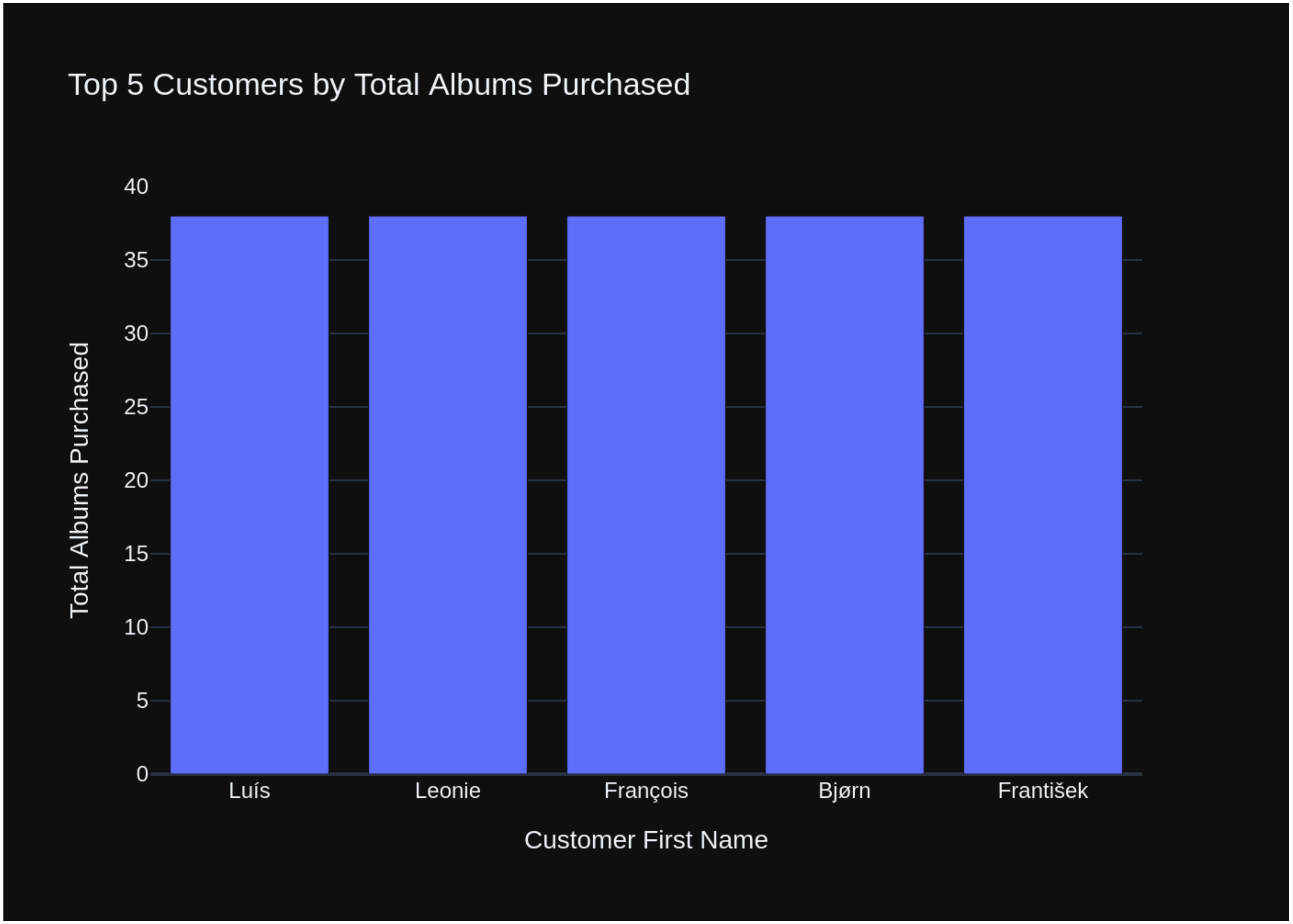
```
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
["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 INT
EGER,\n    MediaTypeId INTEGER  NOT NULL,\n    GenreId INTEGER,\n    Composer NVARCHAR(220),\n    Milliseco
nds INTEGER  NOT NULL,\n    Bytes INTEGER,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Track
PRIMARY KEY  (TrackId),\n    FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON
UPDATE NO ACTION,\n    FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE
NO ACTION,\n    FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON U
PDATE NO ACTION\n)\n\nCREATE TABLE Album\n(\n    AlbumId INTEGER  NOT NULL,\n    Title NVARCHAR(160)  NOT N
ULL,\n    ArtistId INTEGER  NOT NULL,\n    CONSTRAINT PK_Album PRIMARY KEY  (AlbumId),\n    FOREIGN KEY (Ar
tistId) REFERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_A
lbumArtistId 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    Q
uantity INTEGER  NOT NULL,\n    CONSTRAINT PK_InvoiceLine PRIMARY KEY  (InvoiceLineId),\n    FOREIGN KEY (I
nvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (T
rackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Invoice
\n(\n    InvoiceId INTEGER  NOT NULL,\n    CustomerId INTEGER  NOT NULL,\n    InvoiceDate DATETIME  NOT NUL
L,\n    BillingAddress NVARCHAR(70),\n    BillingCity NVARCHAR(40),\n    BillingState NVARCHAR(40),\n    Bi
llingCountry NVARCHAR(40),\n    BillingPostalCode NVARCHAR(10),\n    Total NUMERIC(10,2)  NOT NULL,\n    CO
NSTRAINT PK_Invoice PRIMARY KEY  (InvoiceId),\n    FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerI
d) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceCustomerId ON Invoice (Custo
merId)\n\nCREATE INDEX IFK_TrackAlbumId ON Track (AlbumId)\n\nCREATE INDEX IFK_InvoiceLineTrackId ON Invoic
eLine (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  (Artis
tId)\n)\n\n\n===Additional Context \n\nIn the chinook database invoice means order\n\n===Response Guideline
s \n1. If the provided context is sufficient, please generate a valid SQL query without any explanations fo
r the question. \n2. If the provided context is almost sufficient but requires knowledge of a specific stri
ng in a particular column, please generate an intermediate SQL query to find the distinct strings in that c
olumn. Prepend the query with a comment saying intermediate_sql \n3. If the provided context is insufficien
t, please explain why it can't be generated. \n4. Please use the most relevant table(s). \n5. If the questi
on has been asked and answered before, please repeat the answer exactly as it was given before. \n", '  \n
Find the customer who bought the most albums in total quantity (across all invoices): \n', 'SELECT \n    Cu
stomer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName, \n    SUM(InvoiceLine.Quantity) AS To
talAlbumsPurchased \nFROM \n    Customer \nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.Custome
rId \nLEFT JOIN \n    InvoiceLine ON Invoice.InvoiceId = InvoiceLine.InvoiceId \nWHERE \n    InvoiceLine.Tr
ackId IN (\n        SELECT \n            TrackId \n        FROM \n            Track \n        WHERE \n
AlbumId IS NOT NULL\n    ) \nGROUP BY \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.L
astName \nORDER BY \n    TotalAlbumsPurchased DESC \nLIMIT \n    1;', '  \n    Get the top 10 most popular
artists (based on the number of tracks):\n', 'SELECT \n    Artist.Name, \n    COUNT(Track.TrackId) AS Total
Tracks \nFROM \n    Artist \nLEFT JOIN \n    Album ON Artist.ArtistId = Album.ArtistId \nLEFT JOIN \n    Tr
ack ON Album.AlbumId = Track.AlbumId \nGROUP BY \n    Artist.ArtistId, \n    Artist.Name \nORDER BY \n    T
otalTracks DESC \nLIMIT \n    10;', '  \n     Find the customer with the most invoices \n', 'SELECT \n    C
ustomer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName, \n    COUNT(Invoice.InvoiceId) AS To
talInvoices \nFROM \n    Customer \nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId \nG
ROUP BY \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName \nORDER BY \n    Total
Invoices DESC \nLIMIT \n    1;', '  \n    Find the top 5 most expensive tracks (based on unit price):\n',
'SELECT \n    Name, \n    UnitPrice \nFROM \n    Track \nORDER BY \n    UnitPrice DESC \nLIMIT \n    5;', '
\n    Get the total number of invoices for each customer\n', 'SELECT \n    Customer.CustomerId, \n    Custo
mer.FirstName, \n    Customer.LastName, \n    COUNT(Invoice.InvoiceId) AS TotalInvoices \nFROM \n    Custom
er \nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId \nGROUP BY \n    Customer.Customer
Id, \n    Customer.FirstName, \n    Customer.LastName;', '  \n    List all invoices with a total exceeding
$10:\n', 'SELECT \n    Invoice.InvoiceId, \n    Invoice.InvoiceDate, \n    Invoice.Total \nFROM \n    Invoi
ce \nWHERE \n    Invoice.Total > 10;', '  \n    Get the average invoice total for each customer:\n', 'SELEC
T \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName, \n    AVG(Invoice.Total) AS
AverageInvoiceTotal \nFROM \n    Customer \nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.Custom
erId \nGROUP BY \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName;', '  \n    Fi
nd all invoices since 2010 and the total amount invoiced:\n', "SELECT \n    Invoice.InvoiceDate, \n    SUM
(Invoice.Total) AS TotalInvoiced \nFROM \n    Invoice \nWHERE \n    Invoice.InvoiceDate >= '2010-01-01' \nG
ROUP BY \n    Invoice.InvoiceDate \nORDER BY \n    Invoice.InvoiceDate;", '  \n    List all albums and thei
r corresponding artist names  \n', 'SELECT \n    Album.Title AS AlbumTitle, \n    Artist.Name AS ArtistName
\nFROM \n    Album \nJOIN \n    Artist ON Album.ArtistId = Artist.ArtistId;', '  \n    Find the total numbe
r of invoices per country:\n', 'SELECT \n    Invoice.BillingCountry, \n    COUNT(Invoice.InvoiceId) AS Tota
lInvoices \nFROM \n    Invoice \nGROUP BY \n    Invoice.BillingCountry \nORDER BY \n    TotalInvoices DES
C;', '  \n     Find the top 5 customer who bought the most albums in total quantity (across all invoice
s):\n']
```

~~~
```sql
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    SUM(InvoiceLine.Quantity) AS TotalAlbumsPurchased
FROM
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
LEFT JOIN
    InvoiceLine ON Invoice.InvoiceId = InvoiceLine.InvoiceId
WHERE
    InvoiceLine.TrackId IN (
        SELECT
            TrackId
        FROM
            Track
        WHERE
~~~

~~~
            AlbumId IS NOT NULL
    )
GROUP BY
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName
ORDER BY
    TotalAlbumsPurchased DESC
LIMIT
    5;
```
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    SUM(InvoiceLine.Quantity) AS TotalAlbumsPurchased
FROM
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
LEFT JOIN
    InvoiceLine ON Invoice.InvoiceId = InvoiceLine.InvoiceId
WHERE
    InvoiceLine.TrackId IN (
        SELECT
            TrackId
        FROM
            Track
        WHERE
            AlbumId IS NOT NULL
    )
GROUP BY
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName
ORDER BY
    TotalAlbumsPurchased DESC
LIMIT
    5;
SELECT
    Customer.CustomerId,
    Customer.FirstName,
~~~

```
    Customer.LastName,
    SUM(InvoiceLine.Quantity) AS TotalAlbumsPurchased
FROM
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
LEFT JOIN
    InvoiceLine ON Invoice.InvoiceId = InvoiceLine.InvoiceId
WHERE
    InvoiceLine.TrackId IN (
        SELECT
            TrackId
        FROM
            Track
        WHERE
            AlbumId IS NOT NULL
    )
GROUP BY
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName
ORDER BY
    TotalAlbumsPurchased DESC
LIMIT
    5;
   CustomerId  FirstName     LastName  TotalAlbumsPurchased
0           1       Luís    Gonçalves                    38
1           2     Leonie       Köhler                    38
2           3   François     Tremblay                    38
3           4      Bjørn       Hansen                    38
4           5  František  Wichterlová                    38
```


Top 5 Customers by Total Albums Purchased
40
35
30
25
20
15
10
5
0
Total Albums Purchased
Luís
Leonie
François
Bjørn
František
Customer First Name

Out[39]:
```
('SELECT \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName, \n    SUM(InvoiceLi
ne.Quantity) AS TotalAlbumsPurchased \nFROM \n    Customer \nLEFT JOIN \n    Invoice ON Customer.CustomerI
d = Invoice.CustomerId \nLEFT JOIN \n    InvoiceLine ON Invoice.InvoiceId = InvoiceLine.InvoiceId \nWHERE
\n    InvoiceLine.TrackId IN (\n        SELECT \n            TrackId \n        FROM \n            Track \n
WHERE \n            AlbumId IS NOT NULL\n    ) \nGROUP BY \n    Customer.CustomerId, \n    Customer.FirstN
ame, \n    Customer.LastName \nORDER BY \n    TotalAlbumsPurchased DESC \nLIMIT \n    5;',
    CustomerId  FirstName     LastName  TotalAlbumsPurchased
 0           1       Luís    Gonçalves                    38
 1           2     Leonie       Köhler                    38
 2           3   François     Tremblay                    38
 3           4      Bjørn       Hansen                    38
 4           5  František  Wichterlová                    38,
 Figure({
     'data': [{'name': 'Total Albums Purchased',
               'type': 'bar',
               'x': array(['Luís', 'Leonie', 'François', 'Bjørn', 'František'], dtype=object),
               'y': array([38, 38, 38, 38, 38])}],
     'layout': {'template': '...',
                'title': {'text': 'Top 5 Customers by Total Albums Purchased'},
                'xaxis': {'title': {'text': 'Customer First Name'}},
                'yaxis': {'title': {'text': 'Total Albums Purchased'}}}
 }))
```

In [40]:
```python
question = """
     Find the top 3 customers who spent the most money overall:
"""

vn.ask(question=question)
```

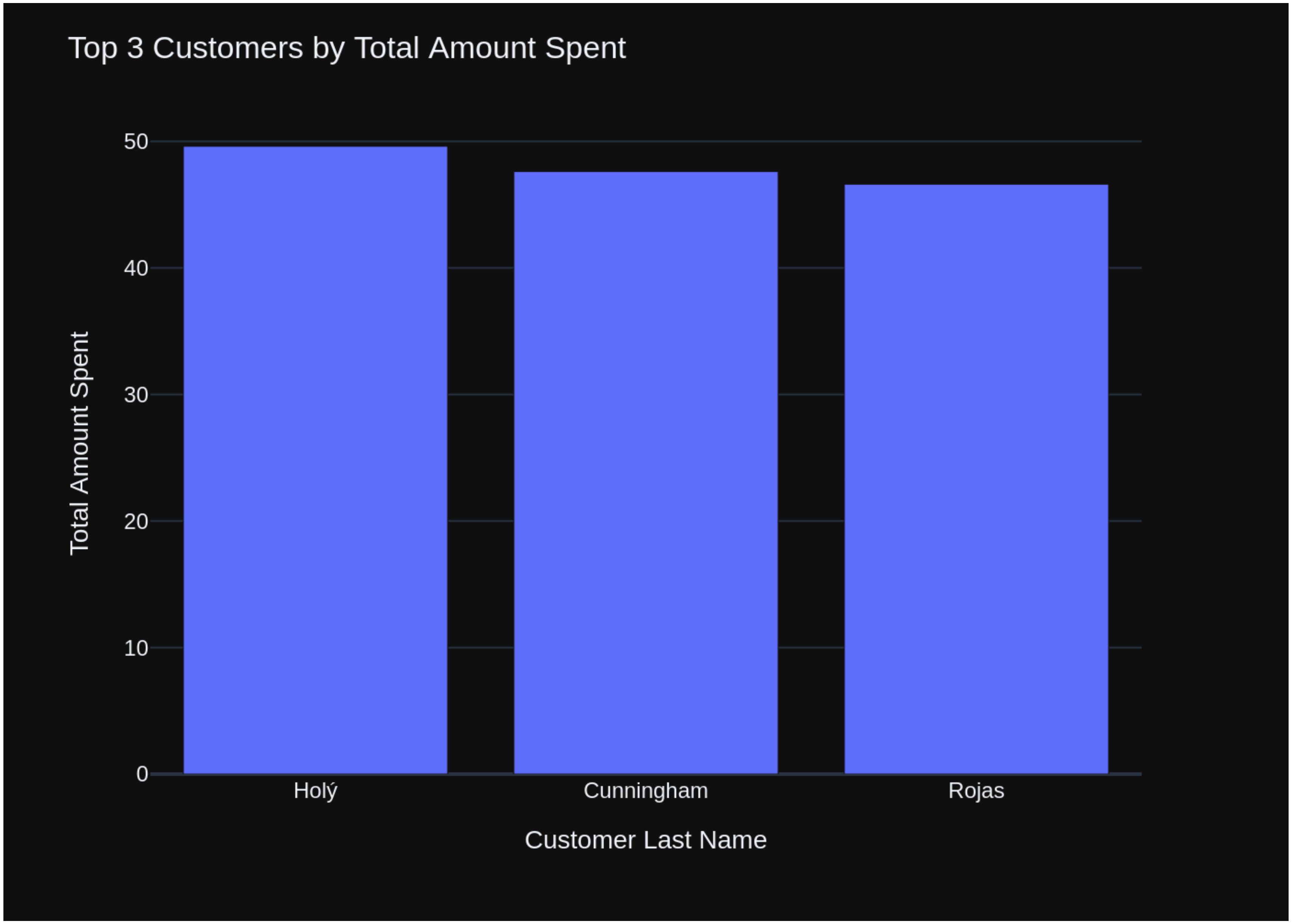
```
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
["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    Invoice
Date 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    Uni
tPrice 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 Customer\n(\n    CustomerId INTEGER  NOT NULL,\n    FirstName NVARCHAR(40)  NO
T NULL,\n    LastName NVARCHAR(20)  NOT NULL,\n    Company NVARCHAR(80),\n    Address NVARCHAR(70),\n    Ci
ty NVARCHAR(40),\n    State NVARCHAR(40),\n    Country NVARCHAR(40),\n    PostalCode NVARCHAR(10),\n    Pho
ne 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 (Empl
oyeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_CustomerSupportRepId ON Custom
er (SupportRepId)\n\nCREATE TABLE Track\n(\n    TrackId INTEGER  NOT NULL,\n    Name NVARCHAR(200)  NOT NUL
L,\n    AlbumId INTEGER,\n    MediaTypeId INTEGER  NOT NULL,\n    GenreId INTEGER,\n    Composer NVARCHAR(2
20),\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 D
ELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)\n\nCRE
ATE INDEX IFK_EmployeeReportsTo ON Employee (ReportsTo)\n\nCREATE TABLE PlaylistTrack\n(\n    PlaylistId IN
TEGER  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 UPDAT
E NO ACTION,\n    FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO A
CTION\n)\n\nCREATE TABLE Employee\n(\n    EmployeeId INTEGER  NOT NULL,\n    LastName NVARCHAR(20)  NOT NUL
L,\n    FirstName NVARCHAR(20)  NOT NULL,\n    Title NVARCHAR(30),\n    ReportsTo INTEGER,\n    BirthDate D
ATETIME,\n    HireDate DATETIME,\n    Address NVARCHAR(70),\n    City NVARCHAR(40),\n    State NVARCHAR(4
0),\n    Country NVARCHAR(40),\n    PostalCode NVARCHAR(10),\n    Phone NVARCHAR(24),\n    Fax NVARCHAR(2
4),\n    Email NVARCHAR(60),\n    CONSTRAINT PK_Employee PRIMARY KEY  (EmployeeId),\n    FOREIGN KEY (Repor
tsTo) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE 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 S
QL query without any explanations for the question. \n2. If the provided context is almost sufficient but r
equires knowledge of a specific string in a particular column, please generate an intermediate SQL query to
find the distinct strings in that column. Prepend the query with a comment saying intermediate_sql \n3. If
the provided context is insufficient, please explain why it can't be generated. \n4. Please use the most re
levant table(s). \n5. If the question has been asked and answered before, please repeat the answer exactly
as it was given before. \n", '  \n     Find the top 5 customer who bought the most albums in total quantity
```

~~~
(across all invoices):\n', 'SELECT \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.Last
Name, \n    SUM(InvoiceLine.Quantity) AS TotalAlbumsPurchased \nFROM \n    Customer \nLEFT JOIN \n    Invoi
ce ON Customer.CustomerId = Invoice.CustomerId \nLEFT JOIN \n    InvoiceLine ON Invoice.InvoiceId = Invoice
Line.InvoiceId \nWHERE \n    InvoiceLine.TrackId IN (\n        SELECT \n            TrackId \n        FROM 
\n            Track \n        WHERE \n            AlbumId IS NOT NULL\n    ) \nGROUP BY \n    Customer.Cust
omerId, \n    Customer.FirstName, \n    Customer.LastName \nORDER BY \n    TotalAlbumsPurchased DESC \nLIMI
T \n    5;', '  \n    Find the top 5 most expensive tracks (based on unit price):\n', 'SELECT \n    Name, 
\n    UnitPrice \nFROM \n    Track \nORDER BY \n    UnitPrice DESC \nLIMIT \n    5;', '  \n     Find the cu
stomer who bought the most albums in total quantity (across all invoices): \n', 'SELECT \n    Customer.Cust
omerId, \n    Customer.FirstName, \n    Customer.LastName, \n    SUM(InvoiceLine.Quantity) AS TotalAlbumsPu
rchased \nFROM \n    Customer \nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId \nLEFT 
JOIN \n    InvoiceLine ON Invoice.InvoiceId = InvoiceLine.InvoiceId \nWHERE \n    InvoiceLine.TrackId IN 
(\n        SELECT \n            TrackId \n        FROM \n            Track \n        WHERE \n            Al
bumId IS NOT NULL\n    ) \nGROUP BY \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.Las
tName \nORDER BY \n    TotalAlbumsPurchased DESC \nLIMIT \n    1;', '  \n     Find the customer with the mo
st invoices \n', 'SELECT \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName, \n 
COUNT(Invoice.InvoiceId) AS TotalInvoices \nFROM \n    Customer \nLEFT JOIN \n    Invoice ON Customer.Custo
merId = Invoice.CustomerId \nGROUP BY \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.L
astName \nORDER BY \n    TotalInvoices DESC \nLIMIT \n    1;', '  \n    Get the average invoice total for e
ach customer:\n', 'SELECT \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName, \n 
AVG(Invoice.Total) AS AverageInvoiceTotal \nFROM \n    Customer \nLEFT JOIN \n    Invoice ON Customer.Custo
merId = Invoice.CustomerId \nGROUP BY \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.L
astName;', '  \n    Get the top 10 most popular artists (based on the number of tracks):\n', 'SELECT \n 
Artist.Name, \n    COUNT(Track.TrackId) AS TotalTracks \nFROM \n    Artist \nLEFT JOIN \n    Album ON Artis
t.ArtistId = Album.ArtistId \nLEFT JOIN \n    Track ON Album.AlbumId = Track.AlbumId \nGROUP BY \n    Artis
t.ArtistId, \n    Artist.Name \nORDER BY \n    TotalTracks DESC \nLIMIT \n    10;', '  \n    Get the total 
number of invoices for each customer\n', 'SELECT \n    Customer.CustomerId, \n    Customer.FirstName, \n 
Customer.LastName, \n    COUNT(Invoice.InvoiceId) AS TotalInvoices \nFROM \n    Customer \nLEFT JOIN \n 
Invoice ON Customer.CustomerId = Invoice.CustomerId \nGROUP BY \n    Customer.CustomerId, \n    Customer.Fi
rstName, \n    Customer.LastName;', '  \n    List all invoices with a total exceeding $10:\n', 'SELECT \n 
Invoice.InvoiceId, \n    Invoice.InvoiceDate, \n    Invoice.Total \nFROM \n    Invoice \nWHERE \n    Invoic
e.Total > 10;', '  \n    Find all invoices since 2010 and the total amount invoiced:\n', "SELECT \n    Invo
ice.InvoiceDate, \n    SUM(Invoice.Total) AS TotalInvoiced \nFROM \n    Invoice \nWHERE \n    Invoice.Invoi
ceDate >= '2010-01-01' \nGROUP BY \n    Invoice.InvoiceDate \nORDER BY \n    Invoice.InvoiceDate;", 'How ma
ny customers are there', 'SELECT \n    COUNT(*) \nFROM \n    Customer;', '  \n     Find the top 3 customers
who spent the most money overall:\n']
```sql
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    SUM(Invoice.Total) AS TotalAmountSpent
~~~

~~~
FROM
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
GROUP BY
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName
ORDER BY
    TotalAmountSpent DESC
LIMIT
    3;
```
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    SUM(Invoice.Total) AS TotalAmountSpent
FROM
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
GROUP BY
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName
ORDER BY
    TotalAmountSpent DESC
LIMIT
    3;
SELECT
    Customer.CustomerId,
    Customer.FirstName,
    Customer.LastName,
    SUM(Invoice.Total) AS TotalAmountSpent
FROM
    Customer
LEFT JOIN
    Invoice ON Customer.CustomerId = Invoice.CustomerId
GROUP BY
    Customer.CustomerId,
    Customer.FirstName,
~~~

```
    Customer.LastName
ORDER BY
    TotalAmountSpent DESC
LIMIT
    3;
   CustomerId FirstName    LastName  TotalAmountSpent
0           6    Helena        Holý             49.62
1          26   Richard  Cunningham             47.62
2          57      Luis       Rojas             46.62
```


Top 3 Customers by Total Amount Spent
Total Amount Spent
50
40
30
20
10
0
Holý
Cunningham
Rojas
Customer Last Name

Out[40]:
```
('SELECT \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName, \n    SUM(Invoice.T
otal) AS TotalAmountSpent \nFROM \n    Customer \nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoic
e.CustomerId \nGROUP BY \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName \nORD
ER BY \n    TotalAmountSpent DESC \nLIMIT \n    3;',
    CustomerId FirstName    LastName  TotalAmountSpent
 0           6    Helena        Holý             49.62
 1          26   Richard  Cunningham             47.62
 2          57      Luis       Rojas             46.62,
 Figure({
     'data': [{'alignmentgroup': 'True',
               'hovertemplate': 'LastName=%{x}<br>TotalAmountSpent=%{y}<extra></extra>',
               'legendgroup': '',
               'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
               'name': '',
               'offsetgroup': '',
               'orientation': 'v',
               'showlegend': False,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Holý', 'Cunningham', 'Rojas'], dtype=object),
               'xaxis': 'x',
               'y': array([49.62, 47.62, 46.62]),
               'yaxis': 'y'}],
     'layout': {'barmode': 'relative',
                'legend': {'tracegroupgap': 0},
                'margin': {'t': 60},
                'template': '...',
                'title': {'text': 'Top 3 Customers by Total Amount Spent'},
                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'Customer Last Name'}},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'Total Amount Spent'}}}
 }))
```

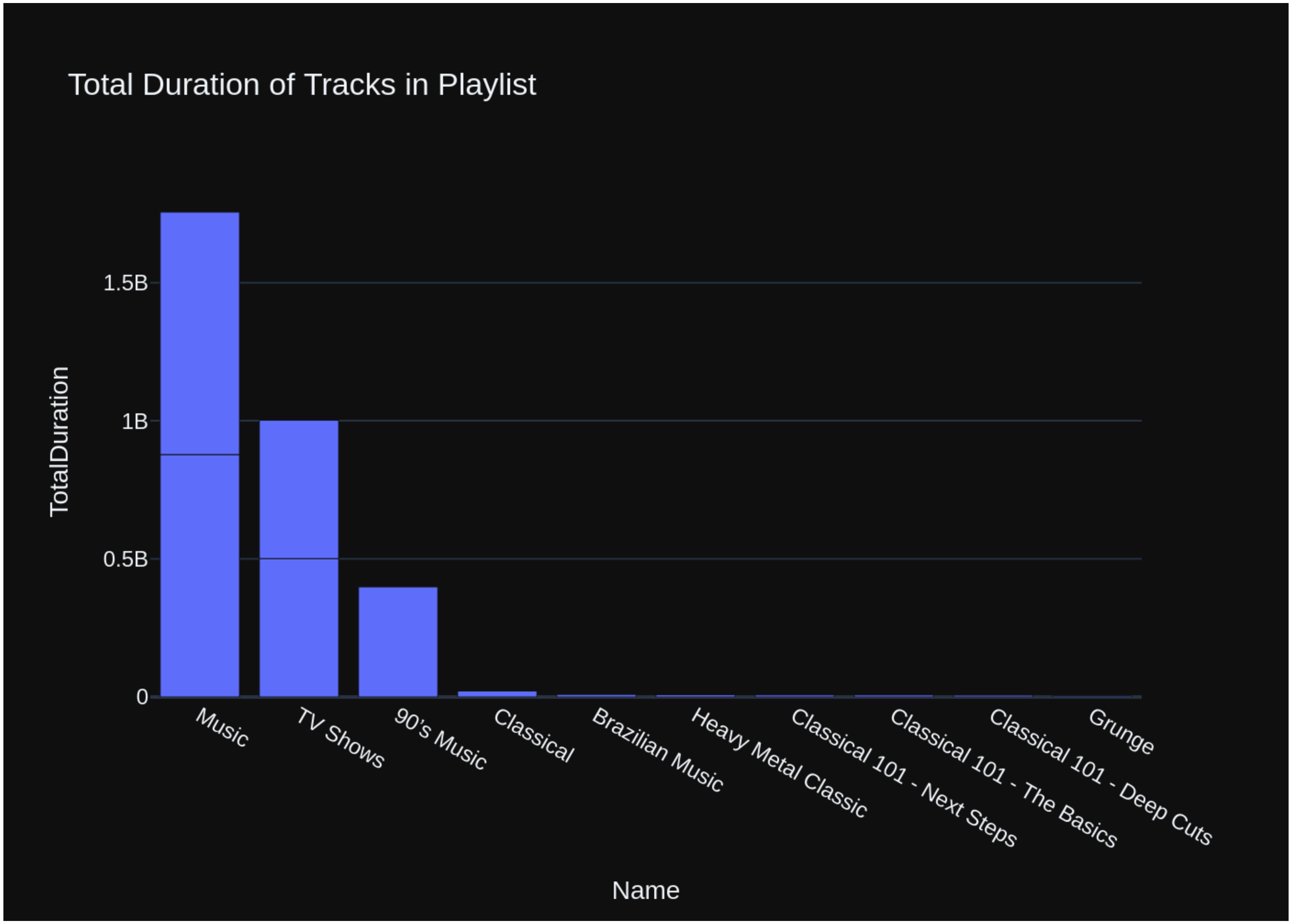
In [41]:
```python
question = """
     Get all playlists containing at least 10 tracks and the total duration of those tracks:
"""

vn.ask(question=question)
```

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

```
["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    Playlis
tId 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 INT
EGER,\n    MediaTypeId INTEGER  NOT NULL,\n    GenreId INTEGER,\n    Composer NVARCHAR(220),\n    Milliseco
nds INTEGER  NOT NULL,\n    Bytes INTEGER,\n    UnitPrice NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Track
PRIMARY KEY  (TrackId),\n    FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON
UPDATE NO ACTION,\n    FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE
NO ACTION,\n    FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON U
PDATE NO ACTION\n)\n\nCREATE TABLE PlaylistTrack\n(\n    PlaylistId INTEGER  NOT NULL,\n    TrackId INTEGER
NOT NULL,\n    CONSTRAINT PK_PlaylistTrack PRIMARY KEY  (PlaylistId, TrackId),\n    FOREIGN KEY (PlaylistI
d) REFERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (TrackI
d) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_TrackGen
reId ON Track (GenreId)\n\nCREATE INDEX IFK_TrackAlbumId ON Track (AlbumId)\n\nCREATE INDEX IFK_TrackMediaT
ypeId 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\t
ON 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 q
uestion. \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. P
repend the query with a comment saying intermediate_sql \n3. If the provided context is insufficient, pleas
e explain why it can't be generated. \n4. Please use the most relevant table(s). \n5. If the question has b
een asked and answered before, please repeat the answer exactly as it was given before. \n", '  \n    List
all genres and the number of tracks in each genre:\n', 'SELECT \n    Genre.Name, \n    COUNT(Track.TrackId)
AS TotalTracks \nFROM \n    Genre \nLEFT JOIN \n    Track ON Genre.GenreId = Track.GenreId \nGROUP BY \n
Genre.GenreId, \n    Genre.Name \nORDER BY \n    TotalTracks DESC;', '  \n    Get the top 10 most popular a
rtists (based on the number of tracks):\n', 'SELECT \n    Artist.Name, \n    COUNT(Track.TrackId) AS TotalT
racks \nFROM \n    Artist \nLEFT JOIN \n    Album ON Artist.ArtistId = Album.ArtistId \nLEFT JOIN \n    Tra
ck ON Album.AlbumId = Track.AlbumId \nGROUP BY \n    Artist.ArtistId, \n    Artist.Name \nORDER BY \n    To
talTracks DESC \nLIMIT \n    10;', '  \n     Find the top 5 customer who bought the most albums in total qu
antity (across all invoices):\n', 'SELECT \n    Customer.CustomerId, \n    Customer.FirstName, \n    Custom
er.LastName, \n    SUM(InvoiceLine.Quantity) AS TotalAlbumsPurchased \nFROM \n    Customer \nLEFT JOIN \n
Invoice ON Customer.CustomerId = Invoice.CustomerId \nLEFT JOIN \n    InvoiceLine ON Invoice.InvoiceId = In
voiceLine.InvoiceId \nWHERE \n    InvoiceLine.TrackId IN (\n        SELECT \n            TrackId \n
FROM \n            Track \n        WHERE \n            AlbumId IS NOT NULL\n    ) \nGROUP BY \n    Custome
r.CustomerId, \n    Customer.FirstName, \n    Customer.LastName \nORDER BY \n    TotalAlbumsPurchased DESC
\nLIMIT \n    5;', '  \n    List all albums and their corresponding artist names  \n', 'SELECT \n    Album.
Title AS AlbumTitle, \n    Artist.Name AS ArtistName \nFROM \n    Album \nJOIN \n    Artist ON Album.Artist
Id = Artist.ArtistId;', '  \n    Find all tracks with a name containing "What" (case-insensitive)\n', "SELE
```

~~~
CT \n    Name \nFROM \n    Track\nWHERE \n    LOWER(Name) LIKE '%what%';", '  \n     Find the customer who
bought the most albums in total quantity (across all invoices): \n', 'SELECT \n    Customer.CustomerId, \n
Customer.FirstName, \n    Customer.LastName, \n    SUM(InvoiceLine.Quantity) AS TotalAlbumsPurchased \nFROM
\n    Customer \nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId \nLEFT JOIN \n    Invo
iceLine ON Invoice.InvoiceId = InvoiceLine.InvoiceId \nWHERE \n    InvoiceLine.TrackId IN (\n        SELECT
\n            TrackId \n        FROM \n            Track \n        WHERE \n            AlbumId IS NOT NULL
\n    ) \nGROUP BY \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName \nORDER BY
\n    TotalAlbumsPurchased DESC \nLIMIT \n    1;', '  \n    Find the top 5 most expensive tracks (based on
unit price):\n', 'SELECT \n    Name, \n    UnitPrice \nFROM \n    Track \nORDER BY \n    UnitPrice DESC \nL
IMIT \n    5;', 'Show me a list of tables in the SQLite database', 'SELECT \n    Name \nFROM \n    Playlist
\nJOIN \n    PlaylistTrack ON Playlist.PlaylistId = PlaylistTrack.PlaylistId\nWHERE \n    PlaylistTrack.Tra
ckId IN (\n        SELECT \n            TrackId \n        FROM \n            InvoiceLine\n        WHERE \n
UnitPrice > 0.99\n    );', '  \n    Find all invoices since 2010 and the total amount invoiced:\n', "SELECT
\n    Invoice.InvoiceDate, \n    SUM(Invoice.Total) AS TotalInvoiced \nFROM \n    Invoice \nWHERE \n    Inv
oice.InvoiceDate >= '2010-01-01' \nGROUP BY \n    Invoice.InvoiceDate \nORDER BY \n    Invoice.InvoiceDat
e;", '  \n    List all invoices with a total exceeding $10:\n', 'SELECT \n    Invoice.InvoiceId, \n    Invo
ice.InvoiceDate, \n    Invoice.Total \nFROM \n    Invoice \nWHERE \n    Invoice.Total > 10;', '  \n     Get
all playlists containing at least 10 tracks and the total duration of those tracks:\n']
```sql
SELECT
    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
ORDER BY
    TotalDuration DESC;
```
SELECT
    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
ORDER BY
    TotalDuration DESC;
SELECT
    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
ORDER BY
    TotalDuration DESC;
                         Name  TotalDuration
0                       Music      877683083
1                       Music      877683083
2                    TV Shows      501094957
3                    TV Shows      501094957
4                  90’s Music      398705153
5                   Classical       21770592
6             Brazilian Music        9486559
7         Heavy Metal Classic        8206312
8   Classical 101 - Next Steps        7575051
9   Classical 101 - The Basics        7439811
10   Classical 101 - Deep Cuts        6755730
11                     Grunge        4122018
```


Total Duration of Tracks in Playlist
TotalDuration
1.5B
1B
0.5B
0
Music
TV Shows
90's Music
Classical
Brazilian Music
Heavy Metal Classic
Classical 101 - Next Steps
Classical 101 - The Basics
Classical 101 - Deep Cuts
Grunge
Name

```
Out[41]: ('SELECT \n    Playlist.Name, \n    SUM(Track.Milliseconds) AS TotalDuration \nFROM \n    Playlist \nJOIN
         \n    PlaylistTrack ON Playlist.PlaylistId = PlaylistTrack.PlaylistId \nJOIN \n    Track ON PlaylistTrack.
         TrackId = Track.TrackId \nGROUP BY \n    Playlist.PlaylistId, \n    Playlist.Name \nHAVING \n    COUNT(Tra
         ck.TrackId) >= 10 \nORDER BY \n    TotalDuration DESC;',
                                 Name  TotalDuration
          0                     Music      877683083
          1                     Music      877683083
          2                  TV Shows      501094957
          3                  TV Shows      501094957
          4                 90’s Music      398705153
          5                 Classical       21770592
          6           Brazilian Music        9486559
          7       Heavy Metal Classic        8206312
          8   Classical 101 - Next Steps        7575051
          9   Classical 101 - The Basics        7439811
          10   Classical 101 - Deep Cuts        6755730
          11                    Grunge        4122018,
          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', 'Music', 'TV Shows', 'TV Shows', '90’s Music', 'Classical',
                                    'Brazilian Music', 'Heavy Metal Classic', 'Classical 101 - Next Steps',
                                    'Classical 101 - The Basics', 'Classical 101 - Deep Cuts', 'Grunge'],
                                   dtype=object),
                        'xaxis': 'x',
                        'y': array([877683083, 877683083, 501094957, 501094957, 398705153,  21770592,
                                      9486559,   8206312,   7575051,   7439811,   6755730,   4122018]),
                        'yaxis': 'y'}],
              'layout': {'barmode': 'relative',
                         'legend': {'tracegroupgap': 0},
                         'template': '...',
                         'title': {'text': 'Total Duration of Tracks in Playlist'},
                         'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'Name'}},
                         'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'TotalDuration'}}}
          }))
```

In [42]:

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

vn.ask(question=question)
```

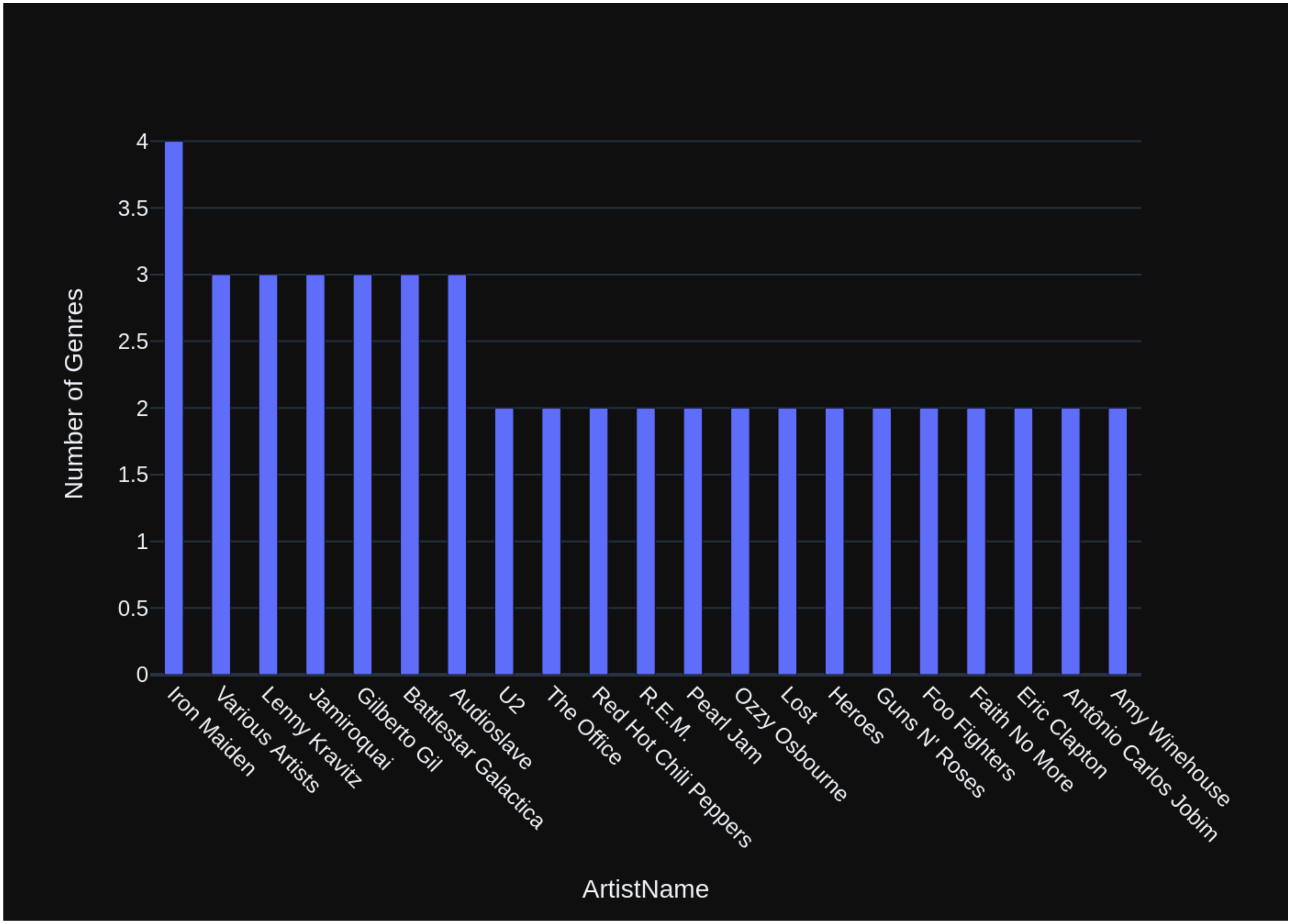
```
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

```
["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 NUL
L,\n    Name NVARCHAR(200)  NOT NULL,\n    AlbumId INTEGER,\n    MediaTypeId INTEGER  NOT NULL,\n    GenreI
d INTEGER,\n    Composer NVARCHAR(220),\n    Milliseconds INTEGER  NOT NULL,\n    Bytes INTEGER,\n    UnitP
rice NUMERIC(10,2)  NOT NULL,\n    CONSTRAINT PK_Track PRIMARY KEY  (TrackId),\n    FOREIGN KEY (AlbumId) R
EFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (GenreId) REFEREN
CES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (MediaTypeId) REFERENCE
S 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    AlbumI
d INTEGER  NOT NULL,\n    Title NVARCHAR(160)  NOT NULL,\n    ArtistId INTEGER  NOT NULL,\n    CONSTRAINT P
K_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 I
FK_PlaylistTrackTrackId ON PlaylistTrack (TrackId)\n\nCREATE TABLE Artist\n(\n    ArtistId INTEGER  NOT NUL
L,\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\nC
REATE TABLE PlaylistTrack\n(\n    PlaylistId INTEGER  NOT NULL,\n    TrackId INTEGER  NOT NULL,\n    CONSTR
AINT PK_PlaylistTrack PRIMARY KEY  (PlaylistId, TrackId),\n    FOREIGN KEY (PlaylistId) REFERENCES Playlist
(PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n    FOREIGN KEY (TrackId) REFERENCES Track (Tr
ackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n===Additional Context \n\nIn the chinook datab
ase invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficient, please genera
te a valid SQL query without any explanations for the question. \n2. If the provided context is almost suff
icient 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 us
e the most relevant table(s). \n5. If the question has been asked and answered before, please repeat the an
swer exactly as it was given before. \n", '  \n    Get the top 10 most popular artists (based on the number
of tracks):\n', 'SELECT \n    Artist.Name, \n    COUNT(Track.TrackId) AS TotalTracks \nFROM \n    Artist \n
LEFT JOIN \n    Album ON Artist.ArtistId = Album.ArtistId \nLEFT JOIN \n    Track ON Album.AlbumId = Track.
AlbumId \nGROUP BY \n    Artist.ArtistId, \n    Artist.Name \nORDER BY \n    TotalTracks DESC \nLIMIT \n
10;', '  \n    List all genres and the number of tracks in each genre:\n', 'SELECT \n    Genre.Name, \n
COUNT(Track.TrackId) AS TotalTracks \nFROM \n    Genre \nLEFT JOIN \n    Track ON Genre.GenreId = Track.Gen
reId \nGROUP BY \n    Genre.GenreId, \n    Genre.Name \nORDER BY \n    TotalTracks DESC;', '  \n    List al
l albums and their corresponding artist names  \n', 'SELECT \n    Album.Title AS AlbumTitle, \n    Artist.N
ame AS ArtistName \nFROM \n    Album \nJOIN \n    Artist ON Album.ArtistId = Artist.ArtistId;', '  \n     G
et all playlists containing at least 10 tracks and the total duration of those tracks:\n', 'SELECT \n    Pl
aylist.Name, \n    SUM(Track.Milliseconds) AS TotalDuration \nFROM \n    Playlist \nJOIN \n    PlaylistTrac
k ON Playlist.PlaylistId = PlaylistTrack.PlaylistId \nJOIN \n    Track ON PlaylistTrack.TrackId = Track.Tra
ckId \nGROUP BY \n    Playlist.PlaylistId, \n    Playlist.Name \nHAVING \n    COUNT(Track.TrackId) >= 10 \n
ORDER BY \n    TotalDuration DESC;', '  \n     Find the top 5 customer who bought the most albums in total
quantity (across all invoices):\n', 'SELECT \n    Customer.CustomerId, \n    Customer.FirstName, \n    Cust
omer.LastName, \n    SUM(InvoiceLine.Quantity) AS TotalAlbumsPurchased \nFROM \n    Customer \nLEFT JOIN \n
```

~~~
Invoice ON Customer.CustomerId = Invoice.CustomerId \nLEFT JOIN \n    InvoiceLine ON Invoice.InvoiceId = In
voiceLine.InvoiceId \nWHERE \n    InvoiceLine.TrackId IN (\n        SELECT \n            TrackId \n        
FROM \n            Track \n        WHERE \n            AlbumId IS NOT NULL\n    ) \nGROUP BY \n    Custome
r.CustomerId, \n    Customer.FirstName, \n    Customer.LastName \nORDER BY \n    TotalAlbumsPurchased DESC 
\nLIMIT \n    5;', '  \n     Find the customer who bought the most albums in total quantity (across all inv
oices): \n', 'SELECT \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName, \n    SU
M(InvoiceLine.Quantity) AS TotalAlbumsPurchased \nFROM \n    Customer \nLEFT JOIN \n    Invoice ON Custome
r.CustomerId = Invoice.CustomerId \nLEFT JOIN \n    InvoiceLine ON Invoice.InvoiceId = InvoiceLine.InvoiceI
d \nWHERE \n    InvoiceLine.TrackId IN (\n        SELECT \n            TrackId \n        FROM \n            
Track \n        WHERE \n            AlbumId IS NOT NULL\n    ) \nGROUP BY \n    Customer.CustomerId, \n    
Customer.FirstName, \n    Customer.LastName \nORDER BY \n    TotalAlbumsPurchased DESC \nLIMIT \n    1;', '
\n    Find the top 5 most expensive tracks (based on unit price):\n', 'SELECT \n    Name, \n    UnitPrice 
\nFROM \n    Track \nORDER BY \n    UnitPrice DESC \nLIMIT \n    5;', '  \n    Find all tracks with a name 
containing "What" (case-insensitive)\n', "SELECT \n    Name \nFROM \n    Track\nWHERE \n    LOWER(Name) LIK
E '%what%';", 'Show me a list of tables in the SQLite database', 'SELECT \n    Name \nFROM \n    Playlist\n
JOIN \n    PlaylistTrack ON Playlist.PlaylistId = PlaylistTrack.PlaylistId\nWHERE \n    PlaylistTrack.Track
Id IN (\n        SELECT \n            TrackId \n        FROM \n            InvoiceLine\n        WHERE \n    
UnitPrice > 0.99\n    );', '  \n     Find the top 3 customers who spent the most money overall:\n', 'SELECT
\n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName, \n    SUM(Invoice.Total) AS T
otalAmountSpent \nFROM \n    Customer \nLEFT JOIN \n    Invoice ON Customer.CustomerId = Invoice.CustomerId
\nGROUP BY \n    Customer.CustomerId, \n    Customer.FirstName, \n    Customer.LastName \nORDER BY \n    To
talAmountSpent DESC \nLIMIT \n    3;', '  \n     Identify artists who have albums with tracks appearing in 
multiple genres:\n']
```sql
SELECT
    Artist.Name AS ArtistName,
    COUNT(DISTINCT Genre.Name) AS NumberOfGenres
FROM
    Artist
LEFT JOIN
    Album ON Artist.ArtistId = Album.ArtistId
LEFT JOIN
    Track ON Album.AlbumId = Track.AlbumId
LEFT JOIN
    Genre ON Track.GenreId = Genre.GenreId
GROUP BY
    ArtistName
HAVING
    NumberOfGenres > 1
ORDER BY
    NumberOfGenres DESC;
~~~
````
````

```
SELECT
    Artist.Name AS ArtistName,
    COUNT(DISTINCT Genre.Name) AS NumberOfGenres
FROM
    Artist
LEFT JOIN
    Album ON Artist.ArtistId = Album.ArtistId
LEFT JOIN
    Track ON Album.AlbumId = Track.AlbumId
LEFT JOIN
    Genre ON Track.GenreId = Genre.GenreId
GROUP BY
    ArtistName
HAVING
    NumberOfGenres > 1
ORDER BY
    NumberOfGenres DESC;
SELECT
    Artist.Name AS ArtistName,
    COUNT(DISTINCT Genre.Name) AS NumberOfGenres
FROM
    Artist
LEFT JOIN
    Album ON Artist.ArtistId = Album.ArtistId
LEFT JOIN
    Track ON Album.AlbumId = Track.AlbumId
LEFT JOIN
    Genre ON Track.GenreId = Genre.GenreId
GROUP BY
    ArtistName
HAVING
    NumberOfGenres > 1
ORDER BY
    NumberOfGenres DESC;
             ArtistName  NumberOfGenres
0           Iron Maiden               4
1       Various Artists               3
2         Lenny Kravitz               3
3            Jamiroquai               3
4          Gilberto Gil               3
5  Battlestar Galactica               3
6            Audioslave               3
```

```
7                       U2                2
8               The Office                2
9   Red Hot Chili Peppers                 2
10                  R.E.M.                2
11               Pearl Jam                2
12           Ozzy Osbourne                2
13                    Lost                2
14                  Heroes                2
15           Guns N' Roses                2
16            Foo Fighters                2
17           Faith No More                2
18            Eric Clapton                2
19    Antônio Carlos Jobim                2
20           Amy Winehouse                2
```


Number of Genres
4
3.5
3
2.5
2
1.5
1
0.5
0
Iron Maiden
Various Artists
Lenny Kravitz
Jamiroquai
Gilberto Gil
Battlestar Galactica
Audioslave
U2
The Office
Red Hot Chili Peppers
R.E.M.
Pearl Jam
Ozzy Osbourne
Lost
Heroes
Guns N' Roses
Foo Fighters
Faith No More
Eric Clapton
Antônio Carlos Jobim
Amy Winehouse
ArtistName

Out[42]:
```
('SELECT \n    Artist.Name AS ArtistName, \n    COUNT(DISTINCT Genre.Name) AS NumberOfGenres \nFROM \n
Artist \nLEFT JOIN \n    Album ON Artist.ArtistId = Album.ArtistId \nLEFT JOIN \n    Track ON Album.AlbumI
d = Track.AlbumId \nLEFT JOIN \n    Genre ON Track.GenreId = Genre.GenreId \nGROUP BY \n    ArtistName \nH
AVING \n    NumberOfGenres > 1 \nORDER BY \n    NumberOfGenres DESC;',
                ArtistName  NumberOfGenres
 0             Iron Maiden               4
 1         Various Artists               3
 2           Lenny Kravitz               3
 3              Jamiroquai               3
 4             Gilberto Gil              3
 5    Battlestar Galactica               3
 6              Audioslave               3
 7                      U2               2
 8              The Office               2
 9   Red Hot Chili Peppers               2
 10                 R.E.M.               2
 11              Pearl Jam               2
 12          Ozzy Osbourne               2
 13                   Lost               2
 14                 Heroes               2
 15          Guns N' Roses               2
 16           Foo Fighters               2
 17          Faith No More               2
 18           Eric Clapton               2
 19   Antônio Carlos Jobim               2
 20          Amy Winehouse               2,
 Figure({
     'data': [{'alignmentgroup': 'True',
               'hovertemplate': 'ArtistName=%{x}<br>NumberOfGenres=%{y}<extra></extra>',
               'legendgroup': '',
               'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
               'name': '',
               'offsetgroup': '',
               'orientation': 'v',
               'showlegend': False,
               'textposition': 'auto',
               'type': 'bar',
               'width': 0.4,
               'x': array(['Iron Maiden', 'Various Artists', 'Lenny Kravitz', 'Jamiroquai',
                           'Gilberto Gil', 'Battlestar Galactica', 'Audioslave', 'U2',
                           'The Office', 'Red Hot Chili Peppers', 'R.E.M.', 'Pearl Jam',
                           'Ozzy Osbourne', 'Lost', 'Heroes', "Guns N' Roses", 'Foo Fighters',
                           'Faith No More', 'Eric Clapton', 'Antônio Carlos Jobim',
                           'Amy Winehouse'], dtype=object),
               'xaxis': 'x',
               'y': array([4, 3, 3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]),
               'yaxis': 'y'}],
     'layout': {'barmode': 'relative',
                'legend': {'tracegroupgap': 0},
                'margin': {'t': 60},
                'showlegend': False,
                'template': '...',
                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'tickangle': 45, 'title': {'text': 'ArtistN
ame'}},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'Number of Genres'}}}
 }))
```

Check completion time

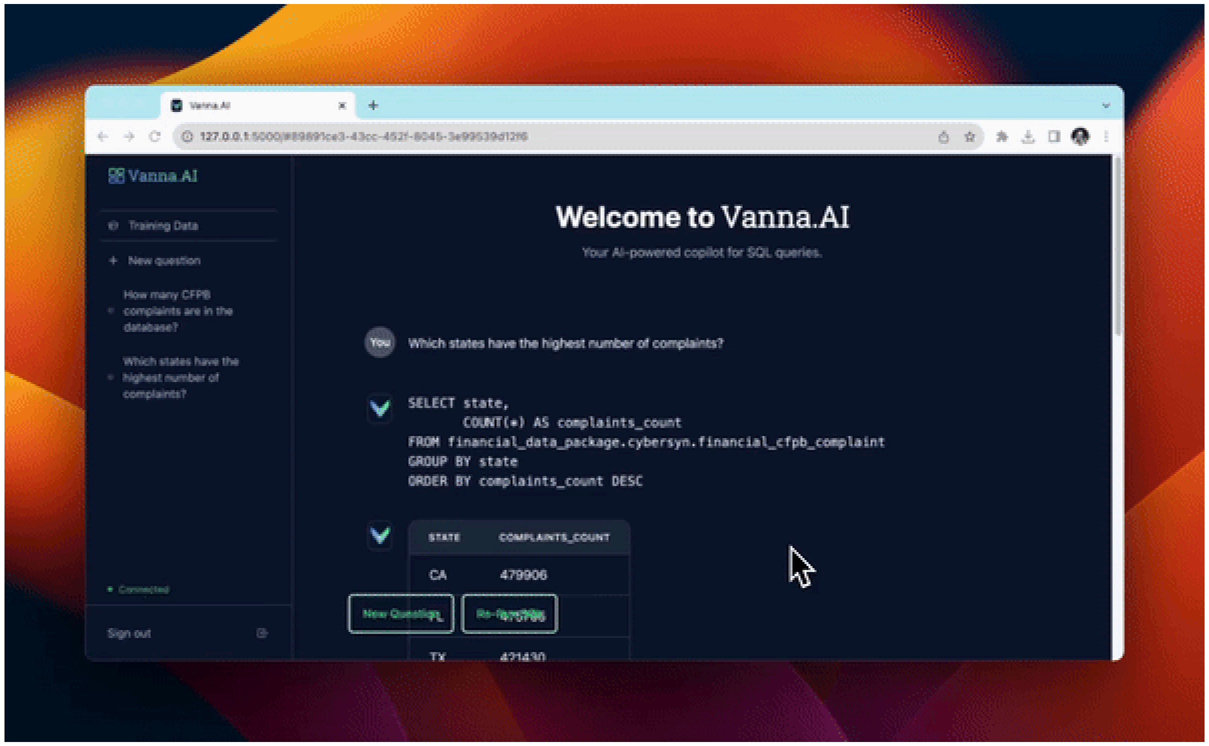
In [43]:
```
ts_stop = time()

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 'gemini-1.5-flash' LLM took : 110.03 sec
```

In [44]:
```
from datetime import datetime
print(datetime.now())
```

```
2024-06-21 00:12:39.370028
```

Launch the User Interface


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

Welcome to Vanna.AI

Your AI-powered copilot for SQL queries.

You Which states have the highest number of complaints?

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

STATE	COMPLAINTS_COUNT
CA	479906
FL	[illegible]
TX	421430

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

Next Steps

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

- Streamlit app
- Flask app
- Slackbot