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'
model_name = 'gemini-1.5-pro'
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]:
```python
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]:
```python
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]:
```python
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")
```

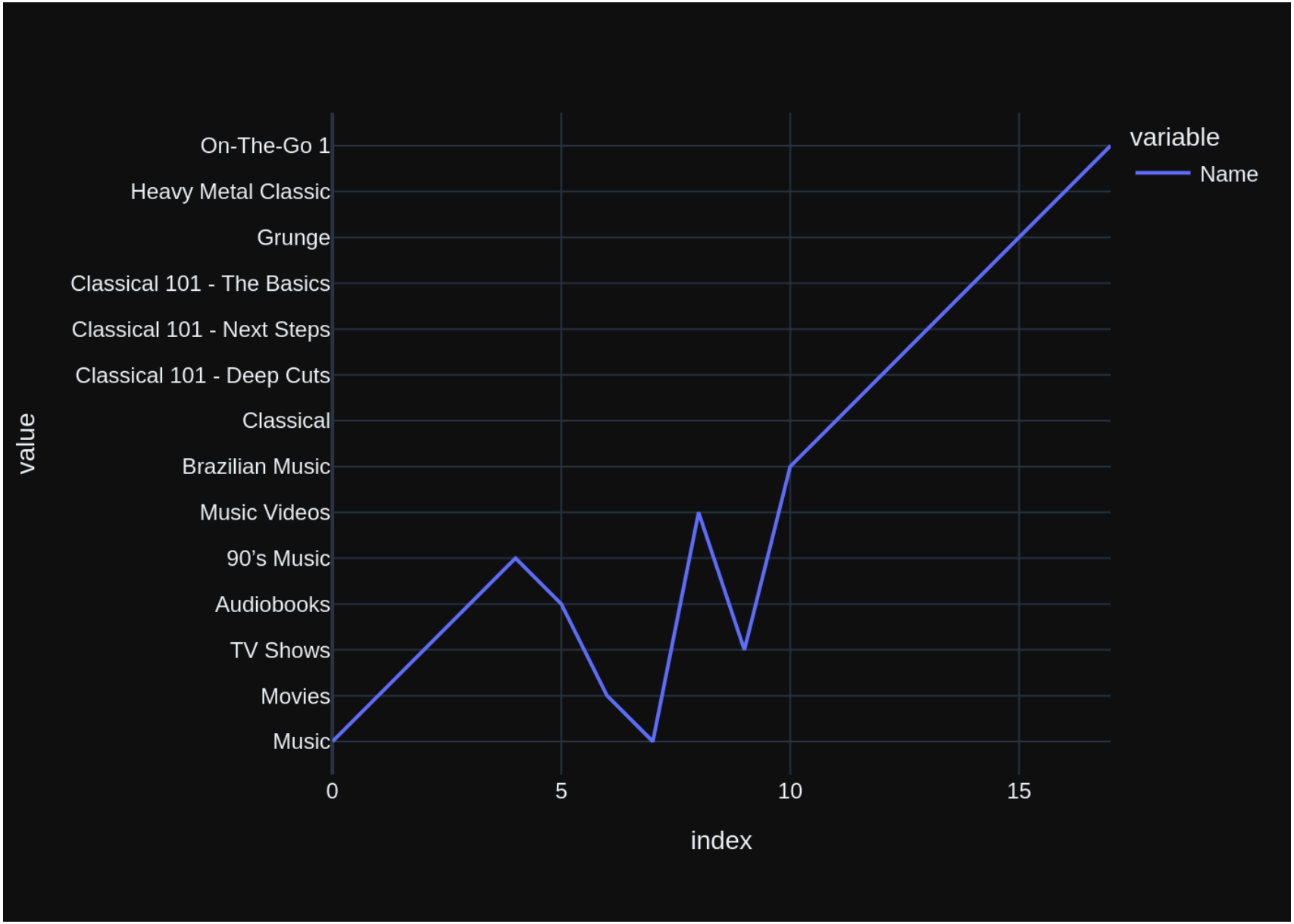
```
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

["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_Playlist 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   Quantity INTEGER NOT NULL,\n   CONSTRAINT PK_InvoiceLine PRIMARY KEY (InvoiceLineId),\n   FOREIGN KEY (InvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE PlaylistTrack\n(\n   PlaylistId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   CONSTRAINT PK_PlaylistTrack PRIMARY KEY (PlaylistId, TrackId),\n   FOREIGN KEY (PlaylistId) REFERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Track\n(\n   TrackId INTEGER NOT NULL,\n   Name NVARCHAR(200) NOT NULL,\n   AlbumId INTEGER,\n   MediaTypeId INTEGER NOT NULL,\n   GenreId INTEGER,\n   Composer NVARCHAR(220),\n   Milliseconds INTEGER NOT NULL,\n   Bytes INTEGER,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Track PRIMARY KEY (TrackId),\n   FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE MediaType\n(\n   MediaTypeId INTEGER NOT NULL,\n   Name NVARCHAR(120),\n   CONSTRAINT PK_MediaType PRIMARY KEY (MediaTypeId)\n)\n\nCREATE TABLE Artist\n(\n   ArtistId INTEGER NOT NULL,\n   Name NVARCHAR(120),\n   CONSTRAINT PK_Artist PRIMARY KEY (ArtistId)\n)\n\nCREATE TABLE Album\n(\n   AlbumId INTEGER NOT NULL,\n   Title NVARCHAR(160) NOT NULL,\n   ArtistId INTEGER NOT NULL,\n   CONSTRAINT PK_Album PRIMARY KEY (AlbumId),\n   FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Genre\n(\n   GenreId INTEGER NOT NULL,\n   Name NVARCHAR(120),\n   CONSTRAINT PK_Genre PRIMARY KEY (GenreId)\n)\n\nCREATE TABLE Invoice\n(\n   InvoiceId INTEGER NOT NULL,\n   CustomerId INTEGER NOT NULL,\n   InvoiceDate DATETIME NOT NULL,\n   BillingAddress NVARCHAR(70),\n   BillingCity NVARCHAR(40),\n   BillingState NVARCHAR(40),\n   BillingCountry NVARCHAR(40),\n   BillingPostalCode NVARCHAR(10),\n   Total NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Invoice PRIMARY KEY (InvoiceId),\n   FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_EmployeeReportsTo ON Employee (ReportsTo)\n\n\n===Additional Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficient, please generate a valid SQL query without any explanations for the question. \n2. If the provided context is almost sufficient but requires knowledge of a specific string in a particular column, please generate an intermediate SQL query to find the distinct strings in that column. Prepend the query with a comment saying intermediate_sql \n3. If the provided context is insufficient, please explain why it can't be generated. \n4. Please use the most relevant table(s). \n5. If the question has been asked and answered before, please repeat the answer exactly as it was given before. \n", 'Show me a list of tables in the SQLite database']

```sql
SELECT
    T.Name
FROM
    Playlist AS T;
```

```
SELECT
    T.Name
FROM
    Playlist AS T;
SELECT
    T.Name
FROM
    Playlist AS T;
```

	Name
0	Music
1	Movies
2	TV Shows
3	Audiobooks
4	90’s Music
5	Audiobooks
6	Movies
7	Music
8	Music Videos
9	TV Shows
10	Brazilian Music
11	Classical
12	Classical 101 - Deep Cuts
13	Classical 101 - Next Steps
14	Classical 101 - The Basics
15	Grunge
16	Heavy Metal Classic
17	On-The-Go 1


On-The-Go 1
Heavy Metal Classic
Grunge
Classical 101 - The Basics
Classical 101 - Next Steps
Classical 101 - Deep Cuts
Classical
Brazilian Music
Music Videos
90’s Music
Audiobooks
TV Shows
Movies
Music
value
0
5
10
15
index
variable
Name

Out[19]: ('SELECT \n   T.Name\nFROM\n   Playlist AS T;',

	Name
0	Music
1	Movies
2	TV Shows
3	Audiobooks
4	90’s Music
5	Audiobooks
6	Movies
7	Music
8	Music Videos
9	TV Shows
10	Brazilian Music
11	Classical
12	Classical 101 - Deep Cuts
13	Classical 101 - Next Steps
14	Classical 101 - The Basics
15	Grunge
16	Heavy Metal Classic
17	On-The-Go 1,

```
Figure({
    'data': [{'hovertemplate': 'variable=Name<br>index=%{x}<br>value=%{y}<extra></extra>',
              'legendgroup': 'Name',
              'line': {'color': '#636efa', 'dash': 'solid'},
              'marker': {'symbol': 'circle'},
              'mode': 'lines',
              'name': 'Name',
              'orientation': 'v',
              'showlegend': True,
              'type': 'scatter',
              'x': array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17]),
              'xaxis': 'x',
              'y': array(['Music', 'Movies', 'TV Shows', 'Audiobooks', '90’s Music', 'Audiobooks',
                          'Movies', 'Music', 'Music Videos', 'TV Shows', 'Brazilian Music',
                          'Classical', 'Classical 101 - Deep Cuts', 'Classical 101 - Next Steps',
                          'Classical 101 - The Basics', 'Grunge', 'Heavy Metal Classic',
                          'On-The-Go 1'], dtype=object),
              'yaxis': 'y'}],
    'layout': {'legend': {'title': {'text': 'variable'}, 'tracegroupgap': 0},
               'margin': {'t': 60},
               'template': '...',
               'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'index'}},
```

```
                    'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'value'}}}
    }))
```

In [20]:
```
vn.ask(question="How many 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   LastName NVARCHAR(20) NOT NULL,\n   Company NVARCHAR(80),\n   Address NVARCHAR(70),\n   City NVARCHAR(40),\n   State NVARCHAR(40),\n   Country NVARCHAR(40),\n   PostalCode NVARCHAR(10),\n   Phone NVARCHAR(24),\n   Fax NVARCHAR(24),\n   Email NVARCHAR(60) NOT NULL,\n   SupportRepId INTEGER,\n   CONSTRAINT PK_Customer PRIMARY KEY (CustomerId),\n   FOREIGN KEY (SupportRepId) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Invoice\n(\n   InvoiceId INTEGER NOT NULL,\n   CustomerId INTEGER NOT NULL,\n   InvoiceDate DATETIME NOT NULL,\n   BillingAddress NVARCHAR(70),\n   BillingCity NVARCHAR(40),\n   BillingState NVARCHAR(40),\n   BillingCountry NVARCHAR(40),\n   BillingPostalCode NVARCHAR(10),\n   Total NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Invoice PRIMARY KEY (InvoiceId),\n   FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE InvoiceLine\n(\n   InvoiceLineId INTEGER NOT NULL,\n   InvoiceId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   Quantity INTEGER NOT NULL,\n   CONSTRAINT PK_InvoiceLine PRIMARY KEY (InvoiceLineId),\n   FOREIGN KEY (InvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Album\n(\n   AlbumId INTEGER NOT NULL,\n   Title NVARCHAR(160) NOT NULL,\n   ArtistId INTEGER NOT NULL,\n   CONSTRAINT PK_Album PRIMARY KEY (AlbumId),\n   FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)\n\nCREATE TABLE Employee\n(\n   EmployeeId INTEGER NOT NULL,\n   LastName NVARCHAR(20) NOT NULL,\n   FirstName NVARCHAR(20) NOT NULL,\n   Title NVARCHAR(30),\n   ReportsTo INTEGER,\n   BirthDate DATETIME,\n   HireDate DATETIME,\n   Address NVARCHAR(70),\n   City NVARCHAR(40),\n   State NVARCHAR(40),\n   Country NVARCHAR(40),\n   PostalCode NVARCHAR(10),\n   Phone NVARCHAR(24),\n   Fax NVARCHAR(24),\n   Email NVARCHAR(60),\n   CONSTRAINT PK_Employee PRIMARY KEY (EmployeeId),\n   FOREIGN KEY (ReportsTo) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Track\n(\n   TrackId INTEGER NOT NULL,\n   Name NVARCHAR(200) NOT NULL,\n   AlbumId INTEGER,\n   MediaTypeId INTEGER NOT NULL,\n   GenreId INTEGER,\n   Composer NVARCHAR(220),\n   Milliseconds INTEGER NOT NULL,\n   Bytes INTEGER,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Track PRIMARY KEY (TrackId),\n   FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_CustomerSupportRepId ON Customer (SupportRepId)\n\nCREATE TABLE Playlist\n(\n   PlaylistId INTEGER NOT NULL,\n   Name NVARCHAR(120),\n   CONSTRAINT PK_Playlist PRIMARY KEY (PlaylistId)\n)\n\nCREATE TABLE PlaylistTrack\n(\n   PlaylistId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   CONSTRAINT PK_PlaylistTrack PRIMARY KEY (PlaylistId, TrackId),\n   FOREIGN KEY (PlaylistId) REFERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n===Additional Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficient, please generate a valid SQL query without any explanations for the question. \n2. If the provided context is almost sufficient but requires knowledge of a specific string in a particular column, please generate an intermediate SQL query to find the distinct strings in that column. Prepend the query with a comment saying 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    T.Name\nFROM\n    Playlist AS T;', 'H
ow many records are in table called customer']
```

```sql
SELECT
    COUNT(*)
FROM
    Customer;
```

```
SELECT
    COUNT(*)
FROM
    Customer;
SELECT
    COUNT(*)
FROM
    Customer;
   COUNT(*)
0        59
```

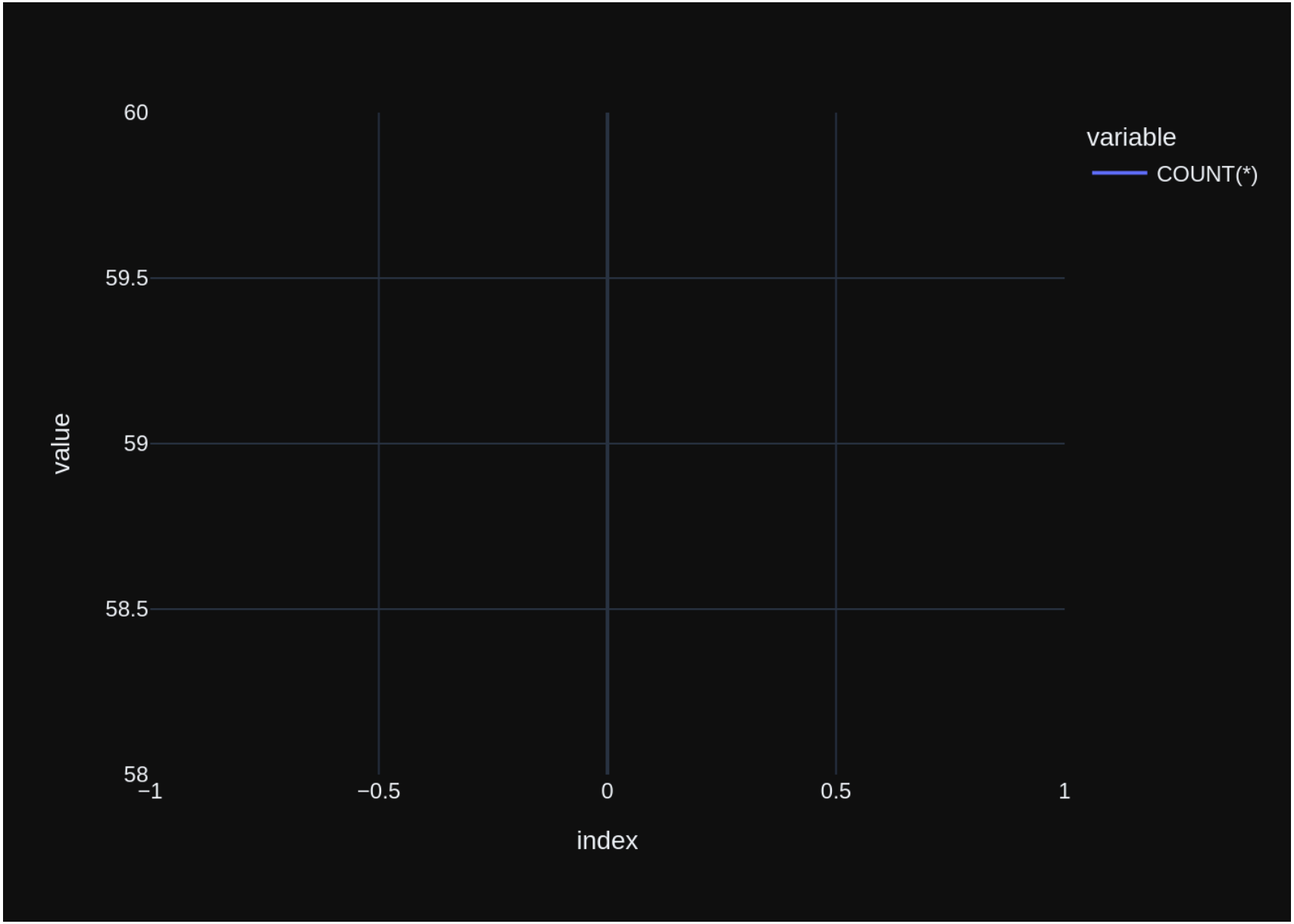

Number of Records in Customer Table
59

Out[20]:
```
('SELECT \n    COUNT(*)\nFROM \n    Customer;',
   COUNT(*)
0        59,
Figure({
    'data': [{'mode': 'number+delta',
              'title': {'text': 'Number of Records in Customer Table'},
              'type': 'indicator',
              'value': 59}],
    'layout': {'template': '...'}
}))
```

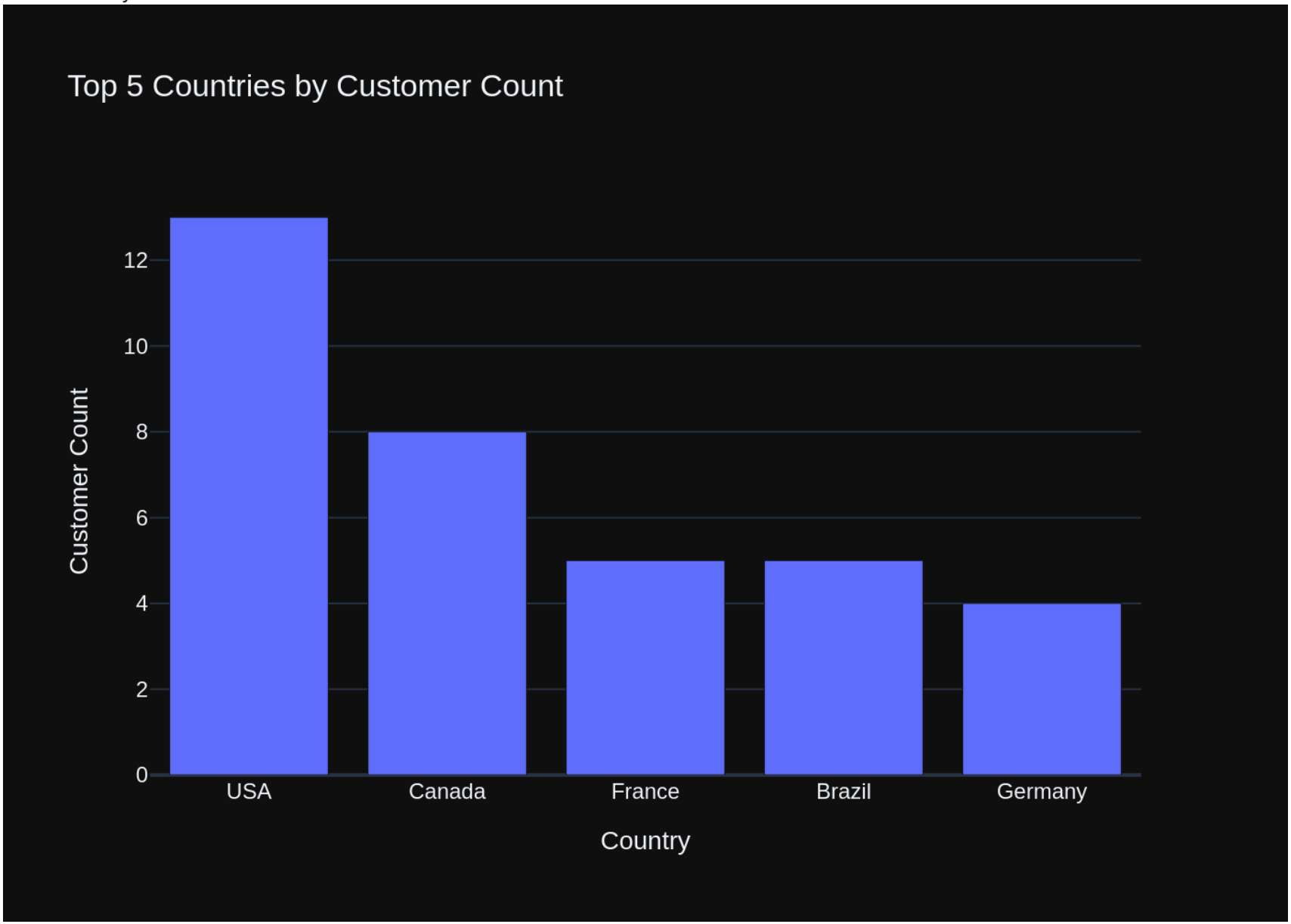
In [21]:
```
vn.ask(question="How many customers are there")
```

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

```
ROM\n    Playlist AS T;', '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   LastName NVARCHAR(20) NOT NULL,\n   Company NVARCHAR(80),\n   Address NVARCHAR(70),\n   City NVARCHAR(40),\n   State NVARCHAR(40),\n   Country NVARCHAR(40),\n   PostalCode NVARCHAR(10),\n   Phone NVARCHAR(24),\n   Fax NVARCHAR(24),\n   Email NVARCHAR(60) NOT NULL,\n   SupportRepId INTEGER,\n   CONSTRAINT PK_Customer PRIMARY KEY (CustomerId),\n   FOREIGN KEY (SupportRepId) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Invoice\n(\n   InvoiceId INTEGER NOT NULL,\n   CustomerId INTEGER NOT NULL,\n   InvoiceDate DATETIME NOT NULL,\n   BillingAddress NVARCHAR(70),\n   BillingCity NVARCHAR(40),\n   BillingState NVARCHAR(40),\n   BillingCountry NVARCHAR(40),\n   BillingPostalCode NVARCHAR(10),\n   Total NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Invoice PRIMARY KEY (InvoiceId),\n   FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE InvoiceLine\n(\n   InvoiceLineId INTEGER NOT NULL,\n   InvoiceId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   Quantity INTEGER NOT NULL,\n   CONSTRAINT PK_InvoiceLine PRIMARY KEY (InvoiceLineId),\n   FOREIGN KEY (InvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Employee\n(\n   EmployeeId INTEGER NOT NULL,\n   LastName NVARCHAR(20) NOT NULL,\n   FirstName NVARCHAR(20) NOT NULL,\n   Title NVARCHAR(30),\n   ReportsTo INTEGER,\n   BirthDate DATETIME,\n   HireDate DATETIME,\n   Address NVARCHAR(70),\n   City NVARCHAR(40),\n   State NVARCHAR(40),\n   Country NVARCHAR(40),\n   PostalCode NVARCHAR(10),\n   Phone NVARCHAR(24),\n   Fax NVARCHAR(24),\n   Email NVARCHAR(60),\n   CONSTRAINT PK_Employee PRIMARY KEY (EmployeeId),\n   FOREIGN KEY (ReportsTo) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE MediaType\n(\n   MediaTypeId INTEGER NOT NULL,\n   Name NVARCHAR(120),\n   CONSTRAINT PK_MediaType PRIMARY KEY (MediaTypeId)\n)\n\nCREATE TABLE Playlist\n(\n   PlaylistId INTEGER NOT NULL,\n   Name NVARCHAR(120),\n   CONSTRAINT PK_Playlist PRIMARY KEY (PlaylistId)\n)\n\nCREATE INDEX IFK_CustomerSupportRepId ON Customer (SupportRepId)\n\nCREATE TABLE PlaylistTrack\n(\n   PlaylistId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   CONSTRAINT PK_PlaylistTrack PRIMARY KEY (PlaylistId, TrackId),\n   FOREIGN KEY (PlaylistId) REFERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Track\n(\n   TrackId INTEGER NOT NULL,\n   Name NVARCHAR(200) NOT NULL,\n   AlbumId INTEGER,\n   MediaTypeId INTEGER NOT NULL,\n   GenreId INTEGER,\n   Composer NVARCHAR(220),\n   Milliseconds INTEGER NOT NULL,\n   Bytes INTEGER,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Track PRIMARY KEY (TrackId),\n   FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Album\n(\n   AlbumId INTEGER NOT NULL,\n   Title NVARCHAR(160) NOT NULL,\n   ArtistId INTEGER NOT NULL,\n   CONSTRAINT PK_Album PRIMARY KEY (AlbumId),\n   FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n===Additional Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficient, please generate a valid SQL query without any explanations for the question. \n2. If the provided context is almost sufficient but requires knowledge of a specific string in a particular column, please generate an intermediate SQL query to find 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    T.Name\nFROM\n    Playlist AS T;', 'what are the top 5 countries that customers come fro
m?']
```sql
SELECT
    C.Country,
    COUNT(*) AS CustomerCount
FROM
    Customer AS C
GROUP BY
    C.Country
ORDER BY
    CustomerCount DESC
LIMIT
    5;
```
SELECT
    C.Country,
    COUNT(*) AS CustomerCount
FROM
    Customer AS C
GROUP BY
    C.Country
ORDER BY
    CustomerCount DESC
LIMIT
    5;
SELECT
    C.Country,
    COUNT(*) AS CustomerCount
FROM
    Customer AS C
GROUP BY
    C.Country
ORDER BY
    CustomerCount DESC
LIMIT
    5;
```

	Country	CustomerCount
0	USA	13
1	Canada	8
2	France	5
3	Brazil	5
4	Germany	4



Out[22]:
```
('SELECT \n    C.Country,\n    COUNT(*) AS CustomerCount\nFROM \n    Customer AS C\nGROUP BY \n    C.Count
ry\nORDER BY \n    CustomerCount DESC\nLIMIT \n    5;',
    Country  CustomerCount
 0      USA             13
 1   Canada              8
 2   France              5
 3   Brazil              5
 4  Germany              4,
 Figure({
     'data': [{'type': 'bar',
               'x': array(['USA', 'Canada', 'France', 'Brazil', 'Germany'], dtype=object),
               'y': array([13,  8,  5,  5,  4])}],
     'layout': {'template': '...',
                'title': {'text': 'Top 5 Countries by Customer Count'},
                'xaxis': {'title': {'text': 'Country'}},
                'yaxis': {'title': {'text': 'Customer Count'}}}
 }))
```

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 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_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\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   T.Name\nFROM\n   Playlist AS T;', 'what are the top 5 countries that customers come fro m?', 'SELECT \n   C.Country,\n   COUNT(*) AS CustomerCount\nFROM \n   Customer AS C\nGROUP BY \n   C.Co untry\nORDER BY \n   CustomerCount DESC\nLIMIT \n   5;', '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']

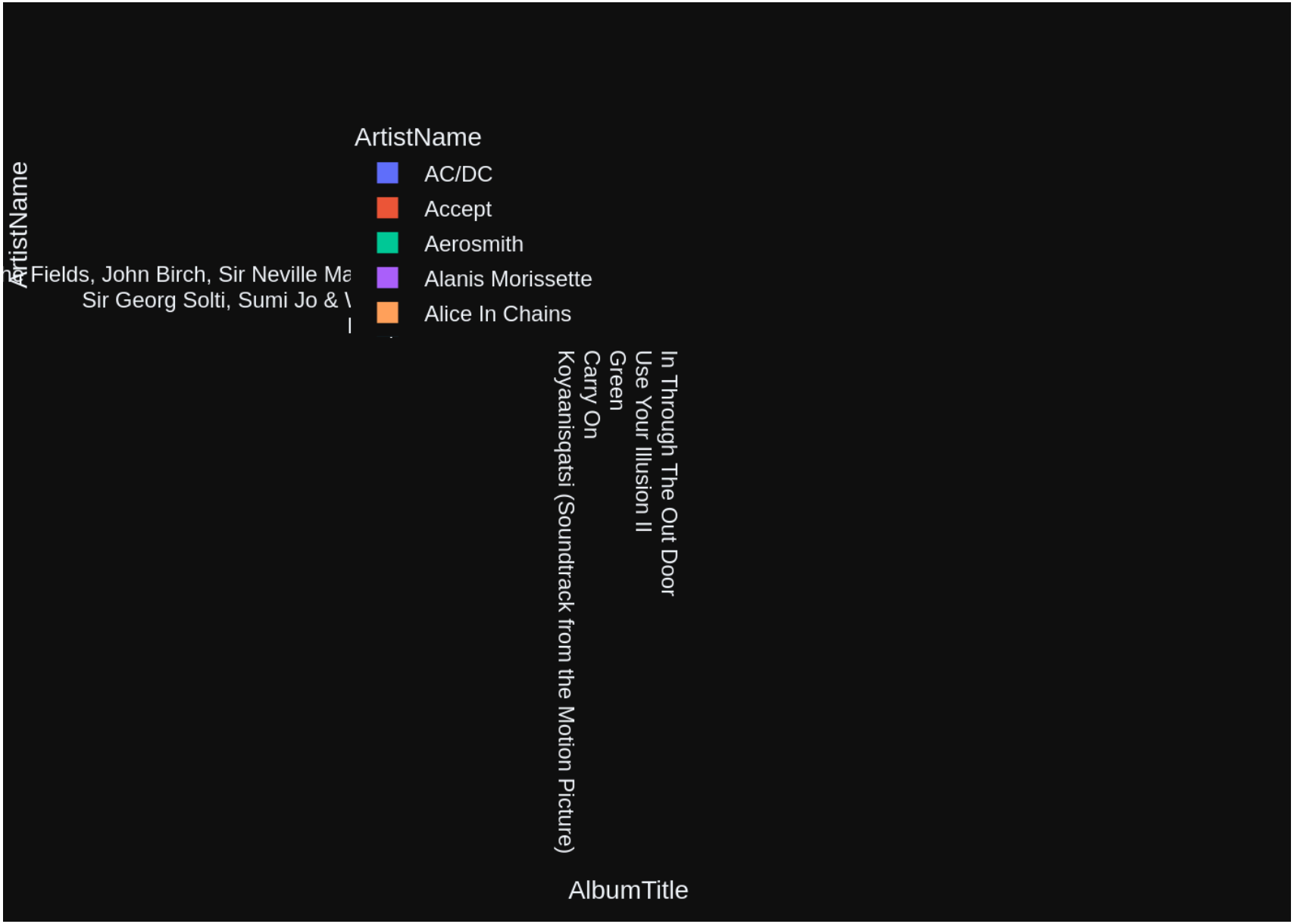
```sql
SELECT
    A.Title AS AlbumTitle,
    Ar.Name AS ArtistName
FROM
    Album AS A
JOIN
    Artist AS Ar ON A.ArtistId = Ar.ArtistId;
```

SELECT

```
    A.Title AS AlbumTitle,
    Ar.Name AS ArtistName
FROM
    Album AS A
JOIN
    Artist AS Ar ON A.ArtistId = Ar.ArtistId;
SELECT
    A.Title AS AlbumTitle,
    Ar.Name AS ArtistName
FROM
    Album AS A
JOIN
    Artist AS Ar ON A.ArtistId = Ar.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]
```


ArtistName
AC/DC
Accept
Aerosmith
Alanis Morissette
Alice In Chains
ArtistName
Fields, John Birch, Sir Neville Ma
Sir Georg Solti, Sumi Jo & \
In Through The Out Door
Use Your Illusion II
Green
Carry On
Koyaanisqatsi (Soundtrack from the Motion Picture)
AlbumTitle

```
Out[23]: ('SELECT \n    A.Title AS AlbumTitle,\n    Ar.Name AS ArtistName\nFROM \n    Album AS A\nJOIN \n    Artist
         AS Ar ON A.ArtistId = Ar.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': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
                        'legendgroup': 'AC/DC',
                        'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
                        'name': 'AC/DC',
                        'offsetgroup': 'AC/DC',
                        'orientation': 'v',
                        'showlegend': True,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['For Those About To Rock We Salute You', 'Let There Be Rock'],
                                   dtype=object),
```

```
 'xaxis': 'x',
 'y': array(['AC/DC', 'AC/DC'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Accept',
 'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
 'name': 'Accept',
 'offsetgroup': 'Accept',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Balls to the Wall', 'Restless and Wild'], dtype=object),
 'xaxis': 'x',
 'y': array(['Accept', 'Accept'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Aerosmith',
 'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
 'name': 'Aerosmith',
 'offsetgroup': 'Aerosmith',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Big Ones'], dtype=object),
 'xaxis': 'x',
 'y': array(['Aerosmith'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Alanis Morissette',
 'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
 'name': 'Alanis Morissette',
 'offsetgroup': 'Alanis Morissette',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Jagged Little Pill'], dtype=object),
```

```
 'xaxis': 'x',
 'y': array(['Alanis Morissette'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Alice In Chains',
 'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
 'name': 'Alice In Chains',
 'offsetgroup': 'Alice In Chains',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Facelift'], dtype=object),
 'xaxis': 'x',
 'y': array(['Alice In Chains'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Antônio Carlos Jobim',
 'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
 'name': 'Antônio Carlos Jobim',
 'offsetgroup': 'Antônio Carlos Jobim',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Warner 25 Anos', 'Chill: Brazil (Disc 2)'], dtype=object),
 'xaxis': 'x',
 'y': array(['Antônio Carlos Jobim', 'Antônio Carlos Jobim'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Apocalyptica',
 'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
 'name': 'Apocalyptica',
 'offsetgroup': 'Apocalyptica',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Plays Metallica By Four Cellos'], dtype=object),
```

```
 'xaxis': 'x',
 'y': array(['Apocalyptica'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Audioslave',
 'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
 'name': 'Audioslave',
 'offsetgroup': 'Audioslave',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Audioslave', 'Out Of Exile', 'Revelations'], dtype=object),
 'xaxis': 'x',
 'y': array(['Audioslave', 'Audioslave', 'Audioslave'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'BackBeat',
 'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
 'name': 'BackBeat',
 'offsetgroup': 'BackBeat',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['BackBeat Soundtrack'], dtype=object),
 'xaxis': 'x',
 'y': array(['BackBeat'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Billy Cobham',
 'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
 'name': 'Billy Cobham',
 'offsetgroup': 'Billy Cobham',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['The Best Of Billy Cobham'], dtype=object),
```

```
 'xaxis': 'x',
 'y': array(['Billy Cobham'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Black Label Society',
 'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
 'name': 'Black Label Society',
 'offsetgroup': 'Black Label Society',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Alcohol Fueled Brewtality Live! [Disc 1]',
             'Alcohol Fueled Brewtality Live! [Disc 2]'], dtype=object),
 'xaxis': 'x',
 'y': array(['Black Label Society', 'Black Label Society'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Black Sabbath',
 'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
 'name': 'Black Sabbath',
 'offsetgroup': 'Black Sabbath',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Black Sabbath', 'Black Sabbath Vol. 4 (Remaster)'], dtype=object),
 'xaxis': 'x',
 'y': array(['Black Sabbath', 'Black Sabbath'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Body Count',
 'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
 'name': 'Body Count',
 'offsetgroup': 'Body Count',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
```

```
 'x': array(['Body Count'], dtype=object),
 'xaxis': 'x',
 'y': array(['Body Count'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Bruce Dickinson',
 'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
 'name': 'Bruce Dickinson',
 'offsetgroup': 'Bruce Dickinson',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Chemical Wedding'], dtype=object),
 'xaxis': 'x',
 'y': array(['Bruce Dickinson'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Buddy Guy',
 'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
 'name': 'Buddy Guy',
 'offsetgroup': 'Buddy Guy',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['The Best Of Buddy Guy - The Millenium Collection'], dtype=object),
 'xaxis': 'x',
 'y': array(['Buddy Guy'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Caetano Veloso',
 'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
 'name': 'Caetano Veloso',
 'offsetgroup': 'Caetano Veloso',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
```

```
 'x': array(['Prenda Minha', 'Sozinho Remix Ao Vivo'], dtype=object),
 'xaxis': 'x',
 'y': array(['Caetano Veloso', 'Caetano Veloso'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Chico Buarque',
 'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
 'name': 'Chico Buarque',
 'offsetgroup': 'Chico Buarque',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Minha Historia'], dtype=object),
 'xaxis': 'x',
 'y': array(['Chico Buarque'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Chico Science & Nação Zumbi',
 'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
 'name': 'Chico Science & Nação Zumbi',
 'offsetgroup': 'Chico Science & Nação Zumbi',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Afrociberdelia', 'Da Lama Ao Caos'], dtype=object),
 'xaxis': 'x',
 'y': array(['Chico Science & Nação Zumbi', 'Chico Science & Nação Zumbi'],
           dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Cidade Negra',
 'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
 'name': 'Cidade Negra',
 'offsetgroup': 'Cidade Negra',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
```

```
 'type': 'bar',
 'x': array(['Acústico MTV [Live]', 'Cidade Negra - Hits'], dtype=object),
 'xaxis': 'x',
 'y': array(['Cidade Negra', 'Cidade Negra'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Cláudio Zoli',
 'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
 'name': 'Cláudio Zoli',
 'offsetgroup': 'Cláudio Zoli',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Na Pista'], dtype=object),
 'xaxis': 'x',
 'y': array(['Cláudio Zoli'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Various Artists',
 'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
 'name': 'Various Artists',
 'offsetgroup': 'Various Artists',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Axé Bahia 2001', 'Carnaval 2001', 'Sambas De Enredo 2001',
             'Vozes do MPB'], dtype=object),
 'xaxis': 'x',
 'y': array(['Various Artists', 'Various Artists', 'Various Artists',
             'Various Artists'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Led Zeppelin',
 'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
 'name': 'Led Zeppelin',
 'offsetgroup': 'Led Zeppelin',
 'orientation': 'v',
```

```
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['BBC Sessions [Disc 1] [Live]', 'Physical Graffiti [Disc 1]',
             'BBC Sessions [Disc 2] [Live]', 'Coda', 'Houses Of The Holy',
             'In Through The Out Door', 'IV', 'Led Zeppelin I', 'Led Zeppelin II',
             'Led Zeppelin III', 'Physical Graffiti [Disc 2]', 'Presence',
             'The Song Remains The Same (Disc 1)',
             'The Song Remains The Same (Disc 2)'], dtype=object),
 'xaxis': 'x',
 'y': array(['Led Zeppelin', 'Led Zeppelin', 'Led Zeppelin', 'Led Zeppelin',
             'Led Zeppelin', 'Led Zeppelin', 'Led Zeppelin', 'Led Zeppelin',
             'Led Zeppelin', 'Led Zeppelin', 'Led Zeppelin', 'Led Zeppelin',
             'Led Zeppelin', 'Led Zeppelin'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Frank Zappa & Captain Beefheart',
 'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
 'name': 'Frank Zappa & Captain Beefheart',
 'offsetgroup': 'Frank Zappa & Captain Beefheart',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Bongo Fury'], dtype=object),
 'xaxis': 'x',
 'y': array(['Frank Zappa & Captain Beefheart'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Marcos Valle',
 'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
 'name': 'Marcos Valle',
 'offsetgroup': 'Marcos Valle',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Chill: Brazil (Disc 1)'], dtype=object),
 'xaxis': 'x',
 'y': array(['Marcos Valle'], dtype=object),
```

```
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Metallica',
 'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
 'name': 'Metallica',
 'offsetgroup': 'Metallica',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Garage Inc. (Disc 1)', 'Black Album', 'Garage Inc. (Disc 2)',
             "Kill 'Em All", 'Load', 'Master Of Puppets', 'ReLoad',
             'Ride The Lightning', 'St. Anger', '...And Justice For All'],
            dtype=object),
 'xaxis': 'x',
 'y': array(['Metallica', 'Metallica', 'Metallica', 'Metallica', 'Metallica',
             'Metallica', 'Metallica', 'Metallica', 'Metallica', 'Metallica'],
            dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Queen',
 'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
 'name': 'Queen',
 'offsetgroup': 'Queen',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Greatest Hits II', 'Greatest Hits I', 'News Of The World'],
            dtype=object),
 'xaxis': 'x',
 'y': array(['Queen', 'Queen', 'Queen'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Kiss',
 'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
 'name': 'Kiss',
 'offsetgroup': 'Kiss',
 'orientation': 'v',
```

```
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Greatest Kiss', 'Unplugged [Live]'], dtype=object),
 'xaxis': 'x',
 'y': array(['Kiss', 'Kiss'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Spyro Gyra',
 'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
 'name': 'Spyro Gyra',
 'offsetgroup': 'Spyro Gyra',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Heart of the Night', 'Morning Dance'], dtype=object),
 'xaxis': 'x',
 'y': array(['Spyro Gyra', 'Spyro Gyra'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Green Day',
 'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
 'name': 'Green Day',
 'offsetgroup': 'Green Day',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['International Superhits', 'American Idiot'], dtype=object),
 'xaxis': 'x',
 'y': array(['Green Day', 'Green Day'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'David Coverdale',
 'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
 'name': 'David Coverdale',
 'offsetgroup': 'David Coverdale',
 'orientation': 'v',
```

```
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Into The Light'], dtype=object),
 'xaxis': 'x',
 'y': array(['David Coverdale'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Gonzaguinha',
 'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
 'name': 'Gonzaguinha',
 'offsetgroup': 'Gonzaguinha',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Meus Momentos'], dtype=object),
 'xaxis': 'x',
 'y': array(['Gonzaguinha'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Os Mutantes',
 'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
 'name': 'Os Mutantes',
 'offsetgroup': 'Os Mutantes',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Minha História'], dtype=object),
 'xaxis': 'x',
 'y': array(['Os Mutantes'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Deep Purple',
 'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
 'name': 'Deep Purple',
 'offsetgroup': 'Deep Purple',
 'orientation': 'v',
```

```
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['MK III The Final Concerts [Disc 1]', 'The Final Concerts (Disc 2)',
             'Come Taste The Band', 'Deep Purple In Rock', 'Fireball',
             "Knocking at Your Back Door: The Best Of Deep Purple in the 80's",
             'Machine Head', 'Purpendicular', 'Slaves And Masters', 'Stormbringer',
             'The Battle Rages On'], dtype=object),
 'xaxis': 'x',
 'y': array(['Deep Purple', 'Deep Purple', 'Deep Purple', 'Deep Purple',
             'Deep Purple', 'Deep Purple', 'Deep Purple', 'Deep Purple',
             'Deep Purple', 'Deep Purple', 'Deep Purple'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Santana',
 'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
 'name': 'Santana',
 'offsetgroup': 'Santana',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Supernatural', 'Santana - As Years Go By', 'Santana Live'],
           dtype=object),
 'xaxis': 'x',
 'y': array(['Santana', 'Santana', 'Santana'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Ed Motta',
 'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
 'name': 'Ed Motta',
 'offsetgroup': 'Ed Motta',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['The Best of Ed Motta'], dtype=object),
 'xaxis': 'x',
 'y': array(['Ed Motta'], dtype=object),
 'yaxis': 'y'},
```

```
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Miles Davis',
 'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
 'name': 'Miles Davis',
 'offsetgroup': 'Miles Davis',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['The Essential Miles Davis [Disc 1]',
             'The Essential Miles Davis [Disc 2]', 'Miles Ahead'], dtype=object),
 'xaxis': 'x',
 'y': array(['Miles Davis', 'Miles Davis', 'Miles Davis'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Gene Krupa',
 'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
 'name': 'Gene Krupa',
 'offsetgroup': 'Gene Krupa',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(["Up An' Atom"], dtype=object),
 'xaxis': 'x',
 'y': array(['Gene Krupa'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Toquinho & Vinícius',
 'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
 'name': 'Toquinho & Vinícius',
 'offsetgroup': 'Toquinho & Vinícius',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Vinícius De Moraes - Sem Limite'], dtype=object),
 'xaxis': 'x',
 'y': array(['Toquinho & Vinícius'], dtype=object),
```

```
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Creedence Clearwater Revival',
 'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
 'name': 'Creedence Clearwater Revival',
 'offsetgroup': 'Creedence Clearwater Revival',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Chronicle, Vol. 1', 'Chronicle, Vol. 2'], dtype=object),
 'xaxis': 'x',
 'y': array(['Creedence Clearwater Revival', 'Creedence Clearwater Revival'],
           dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Cássia Eller',
 'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
 'name': 'Cássia Eller',
 'offsetgroup': 'Cássia Eller',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Cássia Eller - Coleção Sem Limite [Disc 2]',
             'Cássia Eller - Sem Limite [Disc 1]'], dtype=object),
 'xaxis': 'x',
 'y': array(['Cássia Eller', 'Cássia Eller'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Def Leppard',
 'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
 'name': 'Def Leppard',
 'offsetgroup': 'Def Leppard',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(["Vault: Def Leppard's Greatest Hits"], dtype=object),
```

```
 'xaxis': 'x',
 'y': array(['Def Leppard'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Dennis Chambers',
 'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
 'name': 'Dennis Chambers',
 'offsetgroup': 'Dennis Chambers',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Outbreak'], dtype=object),
 'xaxis': 'x',
 'y': array(['Dennis Chambers'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Djavan',
 'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
 'name': 'Djavan',
 'offsetgroup': 'Djavan',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Djavan Ao Vivo - Vol. 02', 'Djavan Ao Vivo - Vol. 1'], dtype=object),
 'xaxis': 'x',
 'y': array(['Djavan', 'Djavan'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Elis Regina',
 'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
 'name': 'Elis Regina',
 'offsetgroup': 'Elis Regina',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Elis Regina-Minha História'], dtype=object),
```

```
 'xaxis': 'x',
 'y': array(['Elis Regina'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Eric Clapton',
 'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
 'name': 'Eric Clapton',
 'offsetgroup': 'Eric Clapton',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['The Cream Of Clapton', 'Unplugged'], dtype=object),
 'xaxis': 'x',
 'y': array(['Eric Clapton', 'Eric Clapton'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Faith No More',
 'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
 'name': 'Faith No More',
 'offsetgroup': 'Faith No More',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Album Of The Year', 'Angel Dust', 'King For A Day Fool For A Lifetime',
             'The Real Thing'], dtype=object),
 'xaxis': 'x',
 'y': array(['Faith No More', 'Faith No More', 'Faith No More', 'Faith No More'],
           dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Falamansa',
 'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
 'name': 'Falamansa',
 'offsetgroup': 'Falamansa',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
```

```
 'type': 'bar',
 'x': array(['Deixa Entrar'], dtype=object),
 'xaxis': 'x',
 'y': array(['Falamansa'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Foo Fighters',
 'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
 'name': 'Foo Fighters',
 'offsetgroup': 'Foo Fighters',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['In Your Honor [Disc 1]', 'In Your Honor [Disc 2]', 'One By One',
             'The Colour And The Shape'], dtype=object),
 'xaxis': 'x',
 'y': array(['Foo Fighters', 'Foo Fighters', 'Foo Fighters', 'Foo Fighters'],
            dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Frank Sinatra',
 'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
 'name': 'Frank Sinatra',
 'offsetgroup': 'Frank Sinatra',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['My Way: The Best Of Frank Sinatra [Disc 1]'], dtype=object),
 'xaxis': 'x',
 'y': array(['Frank Sinatra'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Funk Como Le Gusta',
 'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
 'name': 'Funk Como Le Gusta',
 'offsetgroup': 'Funk Como Le Gusta',
 'orientation': 'v',
```

```
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Roda De Funk'], dtype=object),
 'xaxis': 'x',
 'y': array(['Funk Como Le Gusta'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Gilberto Gil',
 'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
 'name': 'Gilberto Gil',
 'offsetgroup': 'Gilberto Gil',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['As Canções de Eu Tu Eles', 'Quanta Gente Veio Ver (Live)',
             'Quanta Gente Veio ver--Bônus De Carnaval'], dtype=object),
 'xaxis': 'x',
 'y': array(['Gilberto Gil', 'Gilberto Gil', 'Gilberto Gil'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Godsmack',
 'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
 'name': 'Godsmack',
 'offsetgroup': 'Godsmack',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Faceless'], dtype=object),
 'xaxis': 'x',
 'y': array(['Godsmack'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': "Guns N' Roses",
 'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
 'name': "Guns N' Roses",
 'offsetgroup': "Guns N' Roses",
```

```
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Appetite for Destruction', 'Use Your Illusion I',
             'Use Your Illusion II'], dtype=object),
 'xaxis': 'x',
 'y': array(["Guns N' Roses", "Guns N' Roses", "Guns N' Roses"], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Incognito',
 'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
 'name': 'Incognito',
 'offsetgroup': 'Incognito',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Blue Moods'], dtype=object),
 'xaxis': 'x',
 'y': array(['Incognito'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Iron Maiden',
 'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
 'name': 'Iron Maiden',
 'offsetgroup': 'Iron Maiden',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['A Matter of Life and Death', 'A Real Dead One', 'A Real Live One',
             'Brave New World', 'Dance Of Death', 'Fear Of The Dark', 'Iron Maiden',
             'Killers', 'Live After Death', 'Live At Donington 1992 (Disc 1)',
             'Live At Donington 1992 (Disc 2)', 'No Prayer For The Dying',
             'Piece Of Mind', 'Powerslave', 'Rock In Rio [CD1]', 'Rock In Rio [CD2]',
             'Seventh Son of a Seventh Son', 'Somewhere in Time',
             'The Number of The Beast', 'The X Factor', 'Virtual XI'], dtype=object),
 'xaxis': 'x',
 'y': array(['Iron Maiden', 'Iron Maiden', 'Iron Maiden', 'Iron Maiden',
```

```
                'Iron Maiden', 'Iron Maiden', 'Iron Maiden', 'Iron Maiden',
                'Iron Maiden', 'Iron Maiden', 'Iron Maiden', 'Iron Maiden',
                'Iron Maiden', 'Iron Maiden', 'Iron Maiden', 'Iron Maiden',
                'Iron Maiden', 'Iron Maiden', 'Iron Maiden', 'Iron Maiden',
                'Iron Maiden'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'James Brown',
 'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
 'name': 'James Brown',
 'offsetgroup': 'James Brown',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Sex Machine'], dtype=object),
 'xaxis': 'x',
 'y': array(['James Brown'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Jamiroquai',
 'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
 'name': 'Jamiroquai',
 'offsetgroup': 'Jamiroquai',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Emergency On Planet Earth', 'Synkronized',
             'The Return Of The Space Cowboy'], dtype=object),
 'xaxis': 'x',
 'y': array(['Jamiroquai', 'Jamiroquai', 'Jamiroquai'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'JET',
 'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
 'name': 'JET',
 'offsetgroup': 'JET',
 'orientation': 'v',
```

```
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Get Born'], dtype=object),
 'xaxis': 'x',
 'y': array(['JET'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Jimi Hendrix',
 'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
 'name': 'Jimi Hendrix',
 'offsetgroup': 'Jimi Hendrix',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Are You Experienced?'], dtype=object),
 'xaxis': 'x',
 'y': array(['Jimi Hendrix'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Joe Satriani',
 'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
 'name': 'Joe Satriani',
 'offsetgroup': 'Joe Satriani',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Surfing with the Alien (Remastered)'], dtype=object),
 'xaxis': 'x',
 'y': array(['Joe Satriani'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Jorge Ben',
 'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
 'name': 'Jorge Ben',
 'offsetgroup': 'Jorge Ben',
 'orientation': 'v',
```

```
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Jorge Ben Jor 25 Anos'], dtype=object),
 'xaxis': 'x',
 'y': array(['Jorge Ben'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Jota Quest',
 'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
 'name': 'Jota Quest',
 'offsetgroup': 'Jota Quest',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Jota Quest-1995'], dtype=object),
 'xaxis': 'x',
 'y': array(['Jota Quest'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'João Suplicy',
 'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
 'name': 'João Suplicy',
 'offsetgroup': 'João Suplicy',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Cafezinho'], dtype=object),
 'xaxis': 'x',
 'y': array(['João Suplicy'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Judas Priest',
 'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
 'name': 'Judas Priest',
 'offsetgroup': 'Judas Priest',
 'orientation': 'v',
```

```
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Living After Midnight'], dtype=object),
 'xaxis': 'x',
 'y': array(['Judas Priest'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Legião Urbana',
 'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
 'name': 'Legião Urbana',
 'offsetgroup': 'Legião Urbana',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['A TempestadeTempestade Ou O Livro Dos Dias', 'Mais Do Mesmo'],
           dtype=object),
 'xaxis': 'x',
 'y': array(['Legião Urbana', 'Legião Urbana'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Lenny Kravitz',
 'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
 'name': 'Lenny Kravitz',
 'offsetgroup': 'Lenny Kravitz',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Greatest Hits'], dtype=object),
 'xaxis': 'x',
 'y': array(['Lenny Kravitz'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Lulu Santos',
 'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
 'name': 'Lulu Santos',
 'offsetgroup': 'Lulu Santos',
```

```
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Lulu Santos - RCA 100 Anos De Música - Álbum 01',
             'Lulu Santos - RCA 100 Anos De Música - Álbum 02'], dtype=object),
 'xaxis': 'x',
 'y': array(['Lulu Santos', 'Lulu Santos'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Marillion',
 'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
 'name': 'Marillion',
 'offsetgroup': 'Marillion',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Misplaced Childhood'], dtype=object),
 'xaxis': 'x',
 'y': array(['Marillion'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Marisa Monte',
 'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
 'name': 'Marisa Monte',
 'offsetgroup': 'Marisa Monte',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Barulhinho Bom'], dtype=object),
 'xaxis': 'x',
 'y': array(['Marisa Monte'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Marvin Gaye',
 'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
 'name': 'Marvin Gaye',
```

```
 'offsetgroup': 'Marvin Gaye',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Seek And Shall Find: More Of The Best (1963-1981)'], dtype=object),
 'xaxis': 'x',
 'y': array(['Marvin Gaye'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Men At Work',
 'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
 'name': 'Men At Work',
 'offsetgroup': 'Men At Work',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['The Best Of Men At Work'], dtype=object),
 'xaxis': 'x',
 'y': array(['Men At Work'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Milton Nascimento',
 'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
 'name': 'Milton Nascimento',
 'offsetgroup': 'Milton Nascimento',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Milton Nascimento Ao Vivo', 'Minas'], dtype=object),
 'xaxis': 'x',
 'y': array(['Milton Nascimento', 'Milton Nascimento'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Motörhead',
 'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
 'name': 'Motörhead',
```

```
 'offsetgroup': 'Motörhead',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Ace Of Spades'], dtype=object),
 'xaxis': 'x',
 'y': array(['Motörhead'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Mônica Marianno',
 'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
 'name': 'Mônica Marianno',
 'offsetgroup': 'Mônica Marianno',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Demorou...'], dtype=object),
 'xaxis': 'x',
 'y': array(['Mônica Marianno'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Mötley Crüe',
 'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
 'name': 'Mötley Crüe',
 'offsetgroup': 'Mötley Crüe',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Motley Crue Greatest Hits'], dtype=object),
 'xaxis': 'x',
 'y': array(['Mötley Crüe'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Nirvana',
 'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
 'name': 'Nirvana',
```

```
 'offsetgroup': 'Nirvana',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['From The Muddy Banks Of The Wishkah [Live]', 'Nevermind'], dtype=object),
 'xaxis': 'x',
 'y': array(['Nirvana', 'Nirvana'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'O Terço',
 'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
 'name': 'O Terço',
 'offsetgroup': 'O Terço',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Compositores'], dtype=object),
 'xaxis': 'x',
 'y': array(['O Terço'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Olodum',
 'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
 'name': 'Olodum',
 'offsetgroup': 'Olodum',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Olodum'], dtype=object),
 'xaxis': 'x',
 'y': array(['Olodum'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Os Paralamas Do Sucesso',
 'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
 'name': 'Os Paralamas Do Sucesso',
```

```
 'offsetgroup': 'Os Paralamas Do Sucesso',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Acústico MTV', 'Arquivo II', 'Arquivo Os Paralamas Do Sucesso'],
           dtype=object),
 'xaxis': 'x',
 'y': array(['Os Paralamas Do Sucesso', 'Os Paralamas Do Sucesso',
             'Os Paralamas Do Sucesso'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Ozzy Osbourne',
 'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
 'name': 'Ozzy Osbourne',
 'offsetgroup': 'Ozzy Osbourne',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Bark at the Moon (Remastered)', 'Blizzard of Ozz',
             'Diary of a Madman (Remastered)', 'No More Tears (Remastered)',
             'Tribute', 'Speak of the Devil'], dtype=object),
 'xaxis': 'x',
 'y': array(['Ozzy Osbourne', 'Ozzy Osbourne', 'Ozzy Osbourne', 'Ozzy Osbourne',
             'Ozzy Osbourne', 'Ozzy Osbourne'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Page & Plant',
 'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
 'name': 'Page & Plant',
 'offsetgroup': 'Page & Plant',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Walking Into Clarksdale'], dtype=object),
 'xaxis': 'x',
 'y': array(['Page & Plant'], dtype=object),
 'yaxis': 'y'},
```

```
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Passengers',
 'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
 'name': 'Passengers',
 'offsetgroup': 'Passengers',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Original Soundtracks 1'], dtype=object),
 'xaxis': 'x',
 'y': array(['Passengers'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': "Paul D'Ianno",
 'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
 'name': "Paul D'Ianno",
 'offsetgroup': "Paul D'Ianno",
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['The Beast Live'], dtype=object),
 'xaxis': 'x',
 'y': array(["Paul D'Ianno"], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Pearl Jam',
 'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
 'name': 'Pearl Jam',
 'offsetgroup': 'Pearl Jam',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Live On Two Legs [Live]', 'Pearl Jam', 'Riot Act', 'Ten', 'Vs.'],
            dtype=object),
 'xaxis': 'x',
 'y': array(['Pearl Jam', 'Pearl Jam', 'Pearl Jam', 'Pearl Jam', 'Pearl Jam'],
```

```
           dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Pink Floyd',
 'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
 'name': 'Pink Floyd',
 'offsetgroup': 'Pink Floyd',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Dark Side Of The Moon'], dtype=object),
 'xaxis': 'x',
 'y': array(['Pink Floyd'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Planet Hemp',
 'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
 'name': 'Planet Hemp',
 'offsetgroup': 'Planet Hemp',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Os Cães Ladram Mas A Caravana Não Pára'], dtype=object),
 'xaxis': 'x',
 'y': array(['Planet Hemp'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'R.E.M. Feat. Kate Pearson',
 'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
 'name': 'R.E.M. Feat. Kate Pearson',
 'offsetgroup': 'R.E.M. Feat. Kate Pearson',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Out Of Time'], dtype=object),
 'xaxis': 'x',
```

```
 'y': array(['R.E.M. Feat. Kate Pearson'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'R.E.M.',
 'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
 'name': 'R.E.M.',
 'offsetgroup': 'R.E.M.',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Green', 'New Adventures In Hi-Fi', 'The Best Of R.E.M.: The IRS Years'],
           dtype=object),
 'xaxis': 'x',
 'y': array(['R.E.M.', 'R.E.M.', 'R.E.M.'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Raimundos',
 'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
 'name': 'Raimundos',
 'offsetgroup': 'Raimundos',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Cesta Básica'], dtype=object),
 'xaxis': 'x',
 'y': array(['Raimundos'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Raul Seixas',
 'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
 'name': 'Raul Seixas',
 'offsetgroup': 'Raul Seixas',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Raul Seixas'], dtype=object),
```

```
 'xaxis': 'x',
 'y': array(['Raul Seixas'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Red Hot Chili Peppers',
 'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
 'name': 'Red Hot Chili Peppers',
 'offsetgroup': 'Red Hot Chili Peppers',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Blood Sugar Sex Magik', 'By The Way', 'Californication'], dtype=object),
 'xaxis': 'x',
 'y': array(['Red Hot Chili Peppers', 'Red Hot Chili Peppers',
             'Red Hot Chili Peppers'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Rush',
 'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
 'name': 'Rush',
 'offsetgroup': 'Rush',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Retrospective I (1974-1980)'], dtype=object),
 'xaxis': 'x',
 'y': array(['Rush'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Skank',
 'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
 'name': 'Skank',
 'offsetgroup': 'Skank',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
```

```
 'x': array(['Maquinarama', 'O Samba Poconé'], dtype=object),
 'xaxis': 'x',
 'y': array(['Skank', 'Skank'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Smashing Pumpkins',
 'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
 'name': 'Smashing Pumpkins',
 'offsetgroup': 'Smashing Pumpkins',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Judas 0: B-Sides and Rarities', 'Rotten Apples: Greatest Hits'],
           dtype=object),
 'xaxis': 'x',
 'y': array(['Smashing Pumpkins', 'Smashing Pumpkins'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Soundgarden',
 'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
 'name': 'Soundgarden',
 'offsetgroup': 'Soundgarden',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['A-Sides'], dtype=object),
 'xaxis': 'x',
 'y': array(['Soundgarden'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Stevie Ray Vaughan & Double Trouble',
 'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
 'name': 'Stevie Ray Vaughan & Double Trouble',
 'offsetgroup': 'Stevie Ray Vaughan & Double Trouble',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
```

```
 'type': 'bar',
 'x': array(['In Step'], dtype=object),
 'xaxis': 'x',
 'y': array(['Stevie Ray Vaughan & Double Trouble'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Stone Temple Pilots',
 'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
 'name': 'Stone Temple Pilots',
 'offsetgroup': 'Stone Temple Pilots',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Core'], dtype=object),
 'xaxis': 'x',
 'y': array(['Stone Temple Pilots'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'System Of A Down',
 'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
 'name': 'System Of A Down',
 'offsetgroup': 'System Of A Down',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Mezmerize'], dtype=object),
 'xaxis': 'x',
 'y': array(['System Of A Down'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Terry Bozzio, Tony Levin & Steve Stevens',
 'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
 'name': 'Terry Bozzio, Tony Levin & Steve Stevens',
 'offsetgroup': 'Terry Bozzio, Tony Levin & Steve Stevens',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
```

```
 'type': 'bar',
 'x': array(['[1997] Black Light Syndrome'], dtype=object),
 'xaxis': 'x',
 'y': array(['Terry Bozzio, Tony Levin & Steve Stevens'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'The Black Crowes',
 'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
 'name': 'The Black Crowes',
 'offsetgroup': 'The Black Crowes',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Live [Disc 1]', 'Live [Disc 2]'], dtype=object),
 'xaxis': 'x',
 'y': array(['The Black Crowes', 'The Black Crowes'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'The Clash',
 'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
 'name': 'The Clash',
 'offsetgroup': 'The Clash',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['The Singles'], dtype=object),
 'xaxis': 'x',
 'y': array(['The Clash'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'The Cult',
 'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
 'name': 'The Cult',
 'offsetgroup': 'The Cult',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
```

```
                     'type': 'bar',
                     'x': array(['Beyond Good And Evil',
                                 'Pure Cult: The Best Of The Cult (For Rockers, Ravers, Lovers & Sinners) [U
K]'],
                                dtype=object),
                     'xaxis': 'x',
                     'y': array(['The Cult', 'The Cult'], dtype=object),
                     'yaxis': 'y'},
                    {'alignmentgroup': 'True',
                     'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
                     'legendgroup': 'The Doors',
                     'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
                     'name': 'The Doors',
                     'offsetgroup': 'The Doors',
                     'orientation': 'v',
                     'showlegend': True,
                     'textposition': 'auto',
                     'type': 'bar',
                     'x': array(['The Doors'], dtype=object),
                     'xaxis': 'x',
                     'y': array(['The Doors'], dtype=object),
                     'yaxis': 'y'},
                    {'alignmentgroup': 'True',
                     'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
                     'legendgroup': 'The Police',
                     'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
                     'name': 'The Police',
                     'offsetgroup': 'The Police',
                     'orientation': 'v',
                     'showlegend': True,
                     'textposition': 'auto',
                     'type': 'bar',
                     'x': array(['The Police Greatest Hits'], dtype=object),
                     'xaxis': 'x',
                     'y': array(['The Police'], dtype=object),
                     'yaxis': 'y'},
                    {'alignmentgroup': 'True',
                     'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
                     'legendgroup': 'The Rolling Stones',
                     'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
                     'name': 'The Rolling Stones',
                     'offsetgroup': 'The Rolling Stones',
```

```
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Hot Rocks, 1964-1971 (Disc 1)', 'No Security', 'Voodoo Lounge'],
           dtype=object),
 'xaxis': 'x',
 'y': array(['The Rolling Stones', 'The Rolling Stones', 'The Rolling Stones'],
           dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'The Tea Party',
 'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
 'name': 'The Tea Party',
 'offsetgroup': 'The Tea Party',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Tangents', 'Transmission'], dtype=object),
 'xaxis': 'x',
 'y': array(['The Tea Party', 'The Tea Party'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'The Who',
 'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
 'name': 'The Who',
 'offsetgroup': 'The Who',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['My Generation - The Very Best Of The Who'], dtype=object),
 'xaxis': 'x',
 'y': array(['The Who'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Tim Maia',
 'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
```

```
 'name': 'Tim Maia',
 'offsetgroup': 'Tim Maia',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Serie Sem Limite (Disc 1)', 'Serie Sem Limite (Disc 2)'], dtype=object),
 'xaxis': 'x',
 'y': array(['Tim Maia', 'Tim Maia'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Titãs',
 'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
 'name': 'Titãs',
 'offsetgroup': 'Titãs',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Acústico', 'Volume Dois'], dtype=object),
 'xaxis': 'x',
 'y': array(['Titãs', 'Titãs'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Battlestar Galactica',
 'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
 'name': 'Battlestar Galactica',
 'offsetgroup': 'Battlestar Galactica',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Battlestar Galactica: The Story So Far',
             'Battlestar Galactica, Season 3'], dtype=object),
 'xaxis': 'x',
 'y': array(['Battlestar Galactica', 'Battlestar Galactica'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Heroes',
```

```
 'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
 'name': 'Heroes',
 'offsetgroup': 'Heroes',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Heroes, Season 1'], dtype=object),
 'xaxis': 'x',
 'y': array(['Heroes'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Lost',
 'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
 'name': 'Lost',
 'offsetgroup': 'Lost',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Lost, Season 3', 'Lost, Season 1', 'Lost, Season 2', 'LOST, Season 4'],
           dtype=object),
 'xaxis': 'x',
 'y': array(['Lost', 'Lost', 'Lost', 'Lost'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'U2',
 'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
 'name': 'U2',
 'offsetgroup': 'U2',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Achtung Baby', "All That You Can't Leave Behind", 'B-Sides 1980-1990',
             'How To Dismantle An Atomic Bomb', 'Pop', 'Rattle And Hum',
             'The Best Of 1980-1990', 'War', 'Zooropa',
             'Instant Karma: The Amnesty International Campaign to Save Darfur'],
            dtype=object),
 'xaxis': 'x',
```

```
 'y': array(['U2', 'U2', 'U2', 'U2', 'U2', 'U2', 'U2', 'U2', 'U2', 'U2'],
           dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'UB40',
 'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
 'name': 'UB40',
 'offsetgroup': 'UB40',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['UB40 The Best Of - Volume Two [UK]'], dtype=object),
 'xaxis': 'x',
 'y': array(['UB40'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Van Halen',
 'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
 'name': 'Van Halen',
 'offsetgroup': 'Van Halen',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Diver Down', 'The Best Of Van Halen, Vol. I', 'Van Halen',
             'Van Halen III'], dtype=object),
 'xaxis': 'x',
 'y': array(['Van Halen', 'Van Halen', 'Van Halen', 'Van Halen'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Velvet Revolver',
 'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
 'name': 'Velvet Revolver',
 'offsetgroup': 'Velvet Revolver',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
```

```
 'x': array(['Contraband'], dtype=object),
 'xaxis': 'x',
 'y': array(['Velvet Revolver'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Vinícius De Moraes',
 'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
 'name': 'Vinícius De Moraes',
 'offsetgroup': 'Vinícius De Moraes',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Vinicius De Moraes'], dtype=object),
 'xaxis': 'x',
 'y': array(['Vinícius De Moraes'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Zeca Pagodinho',
 'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
 'name': 'Zeca Pagodinho',
 'offsetgroup': 'Zeca Pagodinho',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Ao Vivo [IMPORT]'], dtype=object),
 'xaxis': 'x',
 'y': array(['Zeca Pagodinho'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'The Office',
 'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
 'name': 'The Office',
 'offsetgroup': 'The Office',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
```

```
 'x': array(['The Office, Season 1', 'The Office, Season 2', 'The Office, Season 3'],
           dtype=object),
 'xaxis': 'x',
 'y': array(['The Office', 'The Office', 'The Office'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Dread Zeppelin',
 'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
 'name': 'Dread Zeppelin',
 'offsetgroup': 'Dread Zeppelin',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Un-Led-Ed'], dtype=object),
 'xaxis': 'x',
 'y': array(['Dread Zeppelin'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Battlestar Galactica (Classic)',
 'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
 'name': 'Battlestar Galactica (Classic)',
 'offsetgroup': 'Battlestar Galactica (Classic)',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Battlestar Galactica (Classic), Season 1'], dtype=object),
 'xaxis': 'x',
 'y': array(['Battlestar Galactica (Classic)'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Aquaman',
 'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
 'name': 'Aquaman',
 'offsetgroup': 'Aquaman',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
```

```
 'type': 'bar',
 'x': array(['Aquaman'], dtype=object),
 'xaxis': 'x',
 'y': array(['Aquaman'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Scorpions',
 'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
 'name': 'Scorpions',
 'offsetgroup': 'Scorpions',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['20th Century Masters - The Millennium Collection: The Best of Scorpions'],
           dtype=object),
 'xaxis': 'x',
 'y': array(['Scorpions'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'House Of Pain',
 'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
 'name': 'House Of Pain',
 'offsetgroup': 'House Of Pain',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['House of Pain'], dtype=object),
 'xaxis': 'x',
 'y': array(['House Of Pain'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'O Rappa',
 'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
 'name': 'O Rappa',
 'offsetgroup': 'O Rappa',
 'orientation': 'v',
 'showlegend': True,
```

```
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Radio Brasil (O Som da Jovem Vanguarda) - Seleccao de Henrique Amaro'],
           dtype=object),
 'xaxis': 'x',
 'y': array(['O Rappa'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Cake',
 'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
 'name': 'Cake',
 'offsetgroup': 'Cake',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Cake: B-Sides and Rarities'], dtype=object),
 'xaxis': 'x',
 'y': array(['Cake'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Aisha Duo',
 'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
 'name': 'Aisha Duo',
 'offsetgroup': 'Aisha Duo',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Quiet Songs'], dtype=object),
 'xaxis': 'x',
 'y': array(['Aisha Duo'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Habib Koité and Bamada',
 'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
 'name': 'Habib Koité and Bamada',
 'offsetgroup': 'Habib Koité and Bamada',
 'orientation': 'v',
```

```
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Muso Ko'], dtype=object),
 'xaxis': 'x',
 'y': array(['Habib Koité and Bamada'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Karsh Kale',
 'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
 'name': 'Karsh Kale',
 'offsetgroup': 'Karsh Kale',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Realize'], dtype=object),
 'xaxis': 'x',
 'y': array(['Karsh Kale'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'The Posies',
 'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
 'name': 'The Posies',
 'offsetgroup': 'The Posies',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Every Kind of Light'], dtype=object),
 'xaxis': 'x',
 'y': array(['The Posies'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Luciana Souza/Romero Lubambo',
 'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
 'name': 'Luciana Souza/Romero Lubambo',
 'offsetgroup': 'Luciana Souza/Romero Lubambo',
 'orientation': 'v',
```

```
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Duos II'], dtype=object),
 'xaxis': 'x',
 'y': array(['Luciana Souza/Romero Lubambo'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Aaron Goldberg',
 'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
 'name': 'Aaron Goldberg',
 'offsetgroup': 'Aaron Goldberg',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Worlds'], dtype=object),
 'xaxis': 'x',
 'y': array(['Aaron Goldberg'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Nicolaus Esterhazy Sinfonia',
 'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
 'name': 'Nicolaus Esterhazy Sinfonia',
 'offsetgroup': 'Nicolaus Esterhazy Sinfonia',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['The Best of Beethoven'], dtype=object),
 'xaxis': 'x',
 'y': array(['Nicolaus Esterhazy Sinfonia'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Temple of the Dog',
 'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
 'name': 'Temple of the Dog',
 'offsetgroup': 'Temple of the Dog',
 'orientation': 'v',
```

```
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Temple of the Dog'], dtype=object),
 'xaxis': 'x',
 'y': array(['Temple of the Dog'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Chris Cornell',
 'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
 'name': 'Chris Cornell',
 'offsetgroup': 'Chris Cornell',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Carry On'], dtype=object),
 'xaxis': 'x',
 'y': array(['Chris Cornell'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Alberto Turco & Nova Schola Gregoriana',
 'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
 'name': 'Alberto Turco & Nova Schola Gregoriana',
 'offsetgroup': 'Alberto Turco & Nova Schola Gregoriana',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Adorate Deum: Gregorian Chant from the Proper of the Mass'],
            dtype=object),
 'xaxis': 'x',
 'y': array(['Alberto Turco & Nova Schola Gregoriana'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Richard Marlow & The Choir of Trinity College, Cambridge',
 'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
 'name': 'Richard Marlow & The Choir of Trinity College, Cambridge',
 'offsetgroup': 'Richard Marlow & The Choir of Trinity College, Cambridge',
```

```
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Allegri: Miserere'], dtype=object),
 'xaxis': 'x',
 'y': array(['Richard Marlow & The Choir of Trinity College, Cambridge'],
           dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'English Concert & Trevor Pinnock',
 'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
 'name': 'English Concert & Trevor Pinnock',
 'offsetgroup': 'English Concert & Trevor Pinnock',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Pachelbel: Canon & Gigue',
             'Handel: Music for the Royal Fireworks (Original Version 1749)'],
           dtype=object),
 'xaxis': 'x',
 'y': array(['English Concert & Trevor Pinnock', 'English Concert & Trevor Pinnock'],
           dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Anne-Sophie Mutter, Herbert Von Karajan & Wiener Philharmoniker',
 'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
 'name': 'Anne-Sophie Mutter, Herbert Von Karajan & Wiener Philharmoniker',
 'offsetgroup': 'Anne-Sophie Mutter, Herbert Von Karajan & Wiener Philharmoniker',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Vivaldi: The Four Seasons'], dtype=object),
 'xaxis': 'x',
 'y': array(['Anne-Sophie Mutter, Herbert Von Karajan & Wiener Philharmoniker'],
           dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
```

```
                  'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
                  'legendgroup': 'Hilary Hahn, Jeffrey Kahane, Los Angeles Chamber Orchestra & Margaret Batje
r',
                  'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
                  'name': 'Hilary Hahn, Jeffrey Kahane, Los Angeles Chamber Orchestra & Margaret Batjer',
                  'offsetgroup': 'Hilary Hahn, Jeffrey Kahane, Los Angeles Chamber Orchestra & Margaret Batje
r',
                  'orientation': 'v',
                  'showlegend': True,
                  'textposition': 'auto',
                  'type': 'bar',
                  'x': array(['Bach: Violin Concertos'], dtype=object),
                  'xaxis': 'x',
                  'y': array(['Hilary Hahn, Jeffrey Kahane, Los Angeles Chamber Orchestra & Margaret Batje
r'],
                             dtype=object),
                  'yaxis': 'y'},
                 {'alignmentgroup': 'True',
                  'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
                  'legendgroup': 'Wilhelm Kempff',
                  'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
                  'name': 'Wilhelm Kempff',
                  'offsetgroup': 'Wilhelm Kempff',
                  'orientation': 'v',
                  'showlegend': True,
                  'textposition': 'auto',
                  'type': 'bar',
                  'x': array(['Bach: Goldberg Variations'], dtype=object),
                  'xaxis': 'x',
                  'y': array(['Wilhelm Kempff'], dtype=object),
                  'yaxis': 'y'},
                 {'alignmentgroup': 'True',
                  'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
                  'legendgroup': 'Yo-Yo Ma',
                  'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
                  'name': 'Yo-Yo Ma',
                  'offsetgroup': 'Yo-Yo Ma',
                  'orientation': 'v',
                  'showlegend': True,
                  'textposition': 'auto',
                  'type': 'bar',
                  'x': array(['Bach: The Cello Suites'], dtype=object),
```

```
                    'xaxis': 'x',
                    'y': array(['Yo-Yo Ma'], dtype=object),
                    'yaxis': 'y'},
                   {'alignmentgroup': 'True',
                    'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
                    'legendgroup': 'Scholars Baroque Ensemble',
                    'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
                    'name': 'Scholars Baroque Ensemble',
                    'offsetgroup': 'Scholars Baroque Ensemble',
                    'orientation': 'v',
                    'showlegend': True,
                    'textposition': 'auto',
                    'type': 'bar',
                    'x': array(['Handel: The Messiah (Highlights)'], dtype=object),
                    'xaxis': 'x',
                    'y': array(['Scholars Baroque Ensemble'], dtype=object),
                    'yaxis': 'y'},
                   {'alignmentgroup': 'True',
                    'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
                    'legendgroup': 'Academy of St. Martin in the Fields & Sir Neville Marriner',
                    'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
                    'name': 'Academy of St. Martin in the Fields & Sir Neville Marriner',
                    'offsetgroup': 'Academy of St. Martin in the Fields & Sir Neville Marriner',
                    'orientation': 'v',
                    'showlegend': True,
                    'textposition': 'auto',
                    'type': 'bar',
                    'x': array(['The World of Classical Favourites'], dtype=object),
                    'xaxis': 'x',
                    'y': array(['Academy of St. Martin in the Fields & Sir Neville Marriner'],
                               dtype=object),
                    'yaxis': 'y'},
                   {'alignmentgroup': 'True',
                    'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
                    'legendgroup': 'Academy of St. Martin in the Fields Chamber Ensemble & Sir Neville Marrine
r',
                    'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
                    'name': 'Academy of St. Martin in the Fields Chamber Ensemble & Sir Neville Marriner',
                    'offsetgroup': 'Academy of St. Martin in the Fields Chamber Ensemble & Sir Neville Marrine
r',
                    'orientation': 'v',
                    'showlegend': True,
```

```
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Sir Neville Marriner: A Celebration'], dtype=object),
 'xaxis': 'x',
 'y': array(['Academy of St. Martin in the Fields Chamber Ensemble & Sir Neville Marriner'],
           dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Berliner Philharmoniker, Claudio Abbado & Sabine Meyer',
 'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
 'name': 'Berliner Philharmoniker, Claudio Abbado & Sabine Meyer',
 'offsetgroup': 'Berliner Philharmoniker, Claudio Abbado & Sabine Meyer',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Mozart: Wind Concertos'], dtype=object),
 'xaxis': 'x',
 'y': array(['Berliner Philharmoniker, Claudio Abbado & Sabine Meyer'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Royal Philharmonic Orchestra & Sir Thomas Beecham',
 'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
 'name': 'Royal Philharmonic Orchestra & Sir Thomas Beecham',
 'offsetgroup': 'Royal Philharmonic Orchestra & Sir Thomas Beecham',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Haydn: Symphonies 99 - 104'], dtype=object),
 'xaxis': 'x',
 'y': array(['Royal Philharmonic Orchestra & Sir Thomas Beecham'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Orchestre Révolutionnaire et Romantique & John Eliot Gardiner',
 'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
 'name': 'Orchestre Révolutionnaire et Romantique & John Eliot Gardiner',
 'offsetgroup': 'Orchestre Révolutionnaire et Romantique & John Eliot Gardiner',
 'orientation': 'v',
```

```
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Beethoven: Symhonies Nos. 5 & 6'], dtype=object),
 'xaxis': 'x',
 'y': array(['Orchestre Révolutionnaire et Romantique & John Eliot Gardiner'],
           dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Britten Sinfonia, Ivor Bolton & Lesley Garrett',
 'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
 'name': 'Britten Sinfonia, Ivor Bolton & Lesley Garrett',
 'offsetgroup': 'Britten Sinfonia, Ivor Bolton & Lesley Garrett',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['A Soprano Inspired'], dtype=object),
 'xaxis': 'x',
 'y': array(['Britten Sinfonia, Ivor Bolton & Lesley Garrett'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Chicago Symphony Chorus, Chicago Symphony Orchestra & Sir Georg Solti',
 'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
 'name': 'Chicago Symphony Chorus, Chicago Symphony Orchestra & Sir Georg Solti',
 'offsetgroup': 'Chicago Symphony Chorus, Chicago Symphony Orchestra & Sir Georg Solti',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Great Opera Choruses'], dtype=object),
 'xaxis': 'x',
 'y': array(['Chicago Symphony Chorus, Chicago Symphony Orchestra & Sir Georg Solti'],
           dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Sir Georg Solti & Wiener Philharmoniker',
 'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
 'name': 'Sir Georg Solti & Wiener Philharmoniker',
```

```
                      'offsetgroup': 'Sir Georg Solti & Wiener Philharmoniker',
                      'orientation': 'v',
                      'showlegend': True,
                      'textposition': 'auto',
                      'type': 'bar',
                      'x': array(['Wagner: Favourite Overtures'], dtype=object),
                      'xaxis': 'x',
                      'y': array(['Sir Georg Solti & Wiener Philharmoniker'], dtype=object),
                      'yaxis': 'y'},
                     {'alignmentgroup': 'True',
                      'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
                      'legendgroup': ('Academy of St. Martin in the F' ... 'ville Marriner & Sylvia McNair'),
                      'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
                      'name': ('Academy of St. Martin in the F' ... 'ville Marriner & Sylvia McNair'),
                      'offsetgroup': ('Academy of St. Martin in the F' ... 'ville Marriner & Sylvia McNair'),
                      'orientation': 'v',
                      'showlegend': True,
                      'textposition': 'auto',
                      'type': 'bar',
                      'x': array(['Fauré: Requiem, Ravel: Pavane & Others'], dtype=object),
                      'xaxis': 'x',
                      'y': array(['Academy of St. Martin in the Fields, John Birch, Sir Neville Marriner & Sylvia
McNair'],
                                 dtype=object),
                      'yaxis': 'y'},
                     {'alignmentgroup': 'True',
                      'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
                      'legendgroup': 'London Symphony Orchestra & Sir Charles Mackerras',
                      'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
                      'name': 'London Symphony Orchestra & Sir Charles Mackerras',
                      'offsetgroup': 'London Symphony Orchestra & Sir Charles Mackerras',
                      'orientation': 'v',
                      'showlegend': True,
                      'textposition': 'auto',
                      'type': 'bar',
                      'x': array(['Tchaikovsky: The Nutcracker'], dtype=object),
                      'xaxis': 'x',
                      'y': array(['London Symphony Orchestra & Sir Charles Mackerras'], dtype=object),
                      'yaxis': 'y'},
                     {'alignmentgroup': 'True',
                      'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
                      'legendgroup': 'Barry Wordsworth & BBC Concert Orchestra',
```

```
 'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
 'name': 'Barry Wordsworth & BBC Concert Orchestra',
 'offsetgroup': 'Barry Wordsworth & BBC Concert Orchestra',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['The Last Night of the Proms'], dtype=object),
 'xaxis': 'x',
 'y': array(['Barry Wordsworth & BBC Concert Orchestra'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Herbert Von Karajan, Mirella Freni & Wiener Philharmoniker',
 'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
 'name': 'Herbert Von Karajan, Mirella Freni & Wiener Philharmoniker',
 'offsetgroup': 'Herbert Von Karajan, Mirella Freni & Wiener Philharmoniker',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Puccini: Madama Butterfly - Highlights'], dtype=object),
 'xaxis': 'x',
 'y': array(['Herbert Von Karajan, Mirella Freni & Wiener Philharmoniker'],
           dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Eugene Ormandy',
 'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
 'name': 'Eugene Ormandy',
 'offsetgroup': 'Eugene Ormandy',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Holst: The Planets, Op. 32 & Vaughan Williams: Fantasies',
             'Strauss: Waltzes', 'Respighi:Pines of Rome'], dtype=object),
 'xaxis': 'x',
 'y': array(['Eugene Ormandy', 'Eugene Ormandy', 'Eugene Ormandy'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
```

```
                    'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
                    'legendgroup': 'Luciano Pavarotti',
                    'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
                    'name': 'Luciano Pavarotti',
                    'offsetgroup': 'Luciano Pavarotti',
                    'orientation': 'v',
                    'showlegend': True,
                    'textposition': 'auto',
                    'type': 'bar',
                    'x': array(["Pavarotti's Opera Made Easy"], dtype=object),
                    'xaxis': 'x',
                    'y': array(['Luciano Pavarotti'], dtype=object),
                    'yaxis': 'y'},
                   {'alignmentgroup': 'True',
                    'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
                    'legendgroup': 'Leonard Bernstein & New York Philharmonic',
                    'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
                    'name': 'Leonard Bernstein & New York Philharmonic',
                    'offsetgroup': 'Leonard Bernstein & New York Philharmonic',
                    'orientation': 'v',
                    'showlegend': True,
                    'textposition': 'auto',
                    'type': 'bar',
                    'x': array(["Great Performances - Barber's Adagio and Other Romantic Favorites for String
s"],
                               dtype=object),
                    'xaxis': 'x',
                    'y': array(['Leonard Bernstein & New York Philharmonic'], dtype=object),
                    'yaxis': 'y'},
                   {'alignmentgroup': 'True',
                    'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
                    'legendgroup': 'Boston Symphony Orchestra & Seiji Ozawa',
                    'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
                    'name': 'Boston Symphony Orchestra & Seiji Ozawa',
                    'offsetgroup': 'Boston Symphony Orchestra & Seiji Ozawa',
                    'orientation': 'v',
                    'showlegend': True,
                    'textposition': 'auto',
                    'type': 'bar',
                    'x': array(['Carmina Burana'], dtype=object),
                    'xaxis': 'x',
                    'y': array(['Boston Symphony Orchestra & Seiji Ozawa'], dtype=object),
```

```
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Aaron Copland & London Symphony Orchestra',
 'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
 'name': 'Aaron Copland & London Symphony Orchestra',
 'offsetgroup': 'Aaron Copland & London Symphony Orchestra',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['A Copland Celebration, Vol. I'], dtype=object),
 'xaxis': 'x',
 'y': array(['Aaron Copland & London Symphony Orchestra'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Ton Koopman',
 'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
 'name': 'Ton Koopman',
 'offsetgroup': 'Ton Koopman',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Bach: Toccata & Fugue in D Minor'], dtype=object),
 'xaxis': 'x',
 'y': array(['Ton Koopman'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Sergei Prokofiev & Yuri Temirkanov',
 'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
 'name': 'Sergei Prokofiev & Yuri Temirkanov',
 'offsetgroup': 'Sergei Prokofiev & Yuri Temirkanov',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Prokofiev: Symphony No.1'], dtype=object),
 'xaxis': 'x',
 'y': array(['Sergei Prokofiev & Yuri Temirkanov'], dtype=object),
```

```
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Chicago Symphony Orchestra & Fritz Reiner',
 'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
 'name': 'Chicago Symphony Orchestra & Fritz Reiner',
 'offsetgroup': 'Chicago Symphony Orchestra & Fritz Reiner',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Scheherazade'], dtype=object),
 'xaxis': 'x',
 'y': array(['Chicago Symphony Orchestra & Fritz Reiner'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Orchestra of The Age of Enlightenment',
 'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
 'name': 'Orchestra of The Age of Enlightenment',
 'offsetgroup': 'Orchestra of The Age of Enlightenment',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Bach: The Brandenburg Concertos'], dtype=object),
 'xaxis': 'x',
 'y': array(['Orchestra of The Age of Enlightenment'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Emanuel Ax, Eugene Ormandy & Philadelphia Orchestra',
 'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
 'name': 'Emanuel Ax, Eugene Ormandy & Philadelphia Orchestra',
 'offsetgroup': 'Emanuel Ax, Eugene Ormandy & Philadelphia Orchestra',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Chopin: Piano Concertos Nos. 1 & 2'], dtype=object),
 'xaxis': 'x',
 'y': array(['Emanuel Ax, Eugene Ormandy & Philadelphia Orchestra'], dtype=object),
```

```
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'James Levine',
 'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
 'name': 'James Levine',
 'offsetgroup': 'James Levine',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Mascagni: Cavalleria Rusticana'], dtype=object),
 'xaxis': 'x',
 'y': array(['James Levine'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Berliner Philharmoniker & Hans Rosbaud',
 'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
 'name': 'Berliner Philharmoniker & Hans Rosbaud',
 'offsetgroup': 'Berliner Philharmoniker & Hans Rosbaud',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Sibelius: Finlandia'], dtype=object),
 'xaxis': 'x',
 'y': array(['Berliner Philharmoniker & Hans Rosbaud'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Maurizio Pollini',
 'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
 'name': 'Maurizio Pollini',
 'offsetgroup': 'Maurizio Pollini',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Beethoven Piano Sonatas: Moonlight & Pastorale'], dtype=object),
 'xaxis': 'x',
 'y': array(['Maurizio Pollini'], dtype=object),
```

```
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Gustav Mahler',
 'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
 'name': 'Gustav Mahler',
 'offsetgroup': 'Gustav Mahler',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Great Recordings of the Century - Mahler: Das Lied von der Erde'],
           dtype=object),
 'xaxis': 'x',
 'y': array(['Gustav Mahler'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Felix Schmidt, London Symphony Orchestra & Rafael Frühbeck de Burgos',
 'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
 'name': 'Felix Schmidt, London Symphony Orchestra & Rafael Frühbeck de Burgos',
 'offsetgroup': 'Felix Schmidt, London Symphony Orchestra & Rafael Frühbeck de Burgos',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Elgar: Cello Concerto & Vaughan Williams: Fantasias'], dtype=object),
 'xaxis': 'x',
 'y': array(['Felix Schmidt, London Symphony Orchestra & Rafael Frühbeck de Burgos'],
           dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Edo de Waart & San Francisco Symphony',
 'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
 'name': 'Edo de Waart & San Francisco Symphony',
 'offsetgroup': 'Edo de Waart & San Francisco Symphony',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Adams, John: The Chairman Dances'], dtype=object),
```

```
                      'xaxis': 'x',
                      'y': array(['Edo de Waart & San Francisco Symphony'], dtype=object),
                      'yaxis': 'y'},
                     {'alignmentgroup': 'True',
                      'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
                      'legendgroup': 'Antal Doráti & London Symphony Orchestra',
                      'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
                      'name': 'Antal Doráti & London Symphony Orchestra',
                      'offsetgroup': 'Antal Doráti & London Symphony Orchestra',
                      'orientation': 'v',
                      'showlegend': True,
                      'textposition': 'auto',
                      'type': 'bar',
                      'x': array(["Tchaikovsky: 1812 Festival Overture, Op.49, Capriccio Italien & Beethoven: Wel
lington's Victory"],
                                 dtype=object),
                      'xaxis': 'x',
                      'y': array(['Antal Doráti & London Symphony Orchestra'], dtype=object),
                      'yaxis': 'y'},
                     {'alignmentgroup': 'True',
                      'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
                      'legendgroup': 'Choir Of Westminster Abbey & Simon Preston',
                      'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
                      'name': 'Choir Of Westminster Abbey & Simon Preston',
                      'offsetgroup': 'Choir Of Westminster Abbey & Simon Preston',
                      'orientation': 'v',
                      'showlegend': True,
                      'textposition': 'auto',
                      'type': 'bar',
                      'x': array(['Palestrina: Missa Papae Marcelli & Allegri: Miserere'], dtype=object),
                      'xaxis': 'x',
                      'y': array(['Choir Of Westminster Abbey & Simon Preston'], dtype=object),
                      'yaxis': 'y'},
                     {'alignmentgroup': 'True',
                      'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
                      'legendgroup': 'Michael Tilson Thomas & San Francisco Symphony',
                      'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
                      'name': 'Michael Tilson Thomas & San Francisco Symphony',
                      'offsetgroup': 'Michael Tilson Thomas & San Francisco Symphony',
                      'orientation': 'v',
                      'showlegend': True,
                      'textposition': 'auto',
```

```
 'type': 'bar',
 'x': array(['Prokofiev: Romeo & Juliet', 'Berlioz: Symphonie Fantastique'],
           dtype=object),
 'xaxis': 'x',
 'y': array(['Michael Tilson Thomas & San Francisco Symphony',
             'Michael Tilson Thomas & San Francisco Symphony'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Chor der Wiener Staatsoper, Herbert Von Karajan & Wiener Philharmoniker',
 'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
 'name': 'Chor der Wiener Staatsoper, Herbert Von Karajan & Wiener Philharmoniker',
 'offsetgroup': 'Chor der Wiener Staatsoper, Herbert Von Karajan & Wiener Philharmoniker',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Bizet: Carmen Highlights'], dtype=object),
 'xaxis': 'x',
 'y': array(['Chor der Wiener Staatsoper, Herbert Von Karajan & Wiener Philharmoniker'],
           dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': "The King's Singers",
 'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
 'name': "The King's Singers",
 'offsetgroup': "The King's Singers",
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['English Renaissance'], dtype=object),
 'xaxis': 'x',
 'y': array(["The King's Singers"], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Berliner Philharmoniker & Herbert Von Karajan',
 'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
 'name': 'Berliner Philharmoniker & Herbert Von Karajan',
 'offsetgroup': 'Berliner Philharmoniker & Herbert Von Karajan',
```

```
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Grieg: Peer Gynt Suites & Sibelius: Pelléas et Mélisande',
             'Mozart: Symphonies Nos. 40 & 41',
             'Prokofiev: Symphony No.5 & Stravinksy: Le Sacre Du Printemps'],
            dtype=object),
 'xaxis': 'x',
 'y': array(['Berliner Philharmoniker & Herbert Von Karajan',
             'Berliner Philharmoniker & Herbert Von Karajan',
             'Berliner Philharmoniker & Herbert Von Karajan'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Sir Georg Solti, Sumi Jo & Wiener Philharmoniker',
 'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
 'name': 'Sir Georg Solti, Sumi Jo & Wiener Philharmoniker',
 'offsetgroup': 'Sir Georg Solti, Sumi Jo & Wiener Philharmoniker',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Mozart Gala: Famous Arias'], dtype=object),
 'xaxis': 'x',
 'y': array(['Sir Georg Solti, Sumi Jo & Wiener Philharmoniker'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': "Christopher O'Riley",
 'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
 'name': "Christopher O'Riley",
 'offsetgroup': "Christopher O'Riley",
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['SCRIABIN: Vers la flamme'], dtype=object),
 'xaxis': 'x',
 'y': array(["Christopher O'Riley"], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
```

```
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Fretwork',
 'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
 'name': 'Fretwork',
 'offsetgroup': 'Fretwork',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Armada: Music from the Courts of England and Spain'], dtype=object),
 'xaxis': 'x',
 'y': array(['Fretwork'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Amy Winehouse',
 'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
 'name': 'Amy Winehouse',
 'offsetgroup': 'Amy Winehouse',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Back to Black', 'Frank'], dtype=object),
 'xaxis': 'x',
 'y': array(['Amy Winehouse', 'Amy Winehouse'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Calexico',
 'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
 'name': 'Calexico',
 'offsetgroup': 'Calexico',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Carried to Dust (Bonus Track Version)'], dtype=object),
 'xaxis': 'x',
 'y': array(['Calexico'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
```

```
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Otto Klemperer & Philharmonia Orchestra',
 'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
 'name': 'Otto Klemperer & Philharmonia Orchestra',
 'offsetgroup': 'Otto Klemperer & Philharmonia Orchestra',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(["Beethoven: Symphony No. 6 'Pastoral' Etc."], dtype=object),
 'xaxis': 'x',
 'y': array(['Otto Klemperer & Philharmonia Orchestra'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Yehudi Menuhin',
 'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
 'name': 'Yehudi Menuhin',
 'offsetgroup': 'Yehudi Menuhin',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Bartok: Violin & Viola Concertos'], dtype=object),
 'xaxis': 'x',
 'y': array(['Yehudi Menuhin'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Philharmonia Orchestra & Sir Neville Marriner',
 'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
 'name': 'Philharmonia Orchestra & Sir Neville Marriner',
 'offsetgroup': 'Philharmonia Orchestra & Sir Neville Marriner',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(["Mendelssohn: A Midsummer Night's Dream"], dtype=object),
 'xaxis': 'x',
 'y': array(['Philharmonia Orchestra & Sir Neville Marriner'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
```

```
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Academy of St. Martin in the Fields, Sir Neville Marriner & Thurston Dart',
 'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
 'name': 'Academy of St. Martin in the Fields, Sir Neville Marriner & Thurston Dart',
 'offsetgroup': 'Academy of St. Martin in the Fields, Sir Neville Marriner & Thurston Dart',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Bach: Orchestral Suites Nos. 1 - 4'], dtype=object),
 'xaxis': 'x',
 'y': array(['Academy of St. Martin in the Fields, Sir Neville Marriner & Thurston Dart'],
           dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Les Arts Florissants & William Christie',
 'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
 'name': 'Les Arts Florissants & William Christie',
 'offsetgroup': 'Les Arts Florissants & William Christie',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Charpentier: Divertissements, Airs & Concerts'], dtype=object),
 'xaxis': 'x',
 'y': array(['Les Arts Florissants & William Christie'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'The 12 Cellists of The Berlin Philharmonic',
 'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
 'name': 'The 12 Cellists of The Berlin Philharmonic',
 'offsetgroup': 'The 12 Cellists of The Berlin Philharmonic',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['South American Getaway'], dtype=object),
 'xaxis': 'x',
 'y': array(['The 12 Cellists of The Berlin Philharmonic'], dtype=object),
 'yaxis': 'y'},
```

```
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Adrian Leaper & Doreen de Feis',
 'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
 'name': 'Adrian Leaper & Doreen de Feis',
 'offsetgroup': 'Adrian Leaper & Doreen de Feis',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Górecki: Symphony No. 3'], dtype=object),
 'xaxis': 'x',
 'y': array(['Adrian Leaper & Doreen de Feis'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Roger Norrington, London Classical Players',
 'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
 'name': 'Roger Norrington, London Classical Players',
 'offsetgroup': 'Roger Norrington, London Classical Players',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Purcell: The Fairy Queen'], dtype=object),
 'xaxis': 'x',
 'y': array(['Roger Norrington, London Classical Players'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': "Charles Dutoit & L'Orchestre Symphonique de Montréal",
 'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
 'name': "Charles Dutoit & L'Orchestre Symphonique de Montréal",
 'offsetgroup': "Charles Dutoit & L'Orchestre Symphonique de Montréal",
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['The Ultimate Relexation Album'], dtype=object),
 'xaxis': 'x',
 'y': array(["Charles Dutoit & L'Orchestre Symphonique de Montréal"], dtype=object),
 'yaxis': 'y'},
```

```
                    {'alignmentgroup': 'True',
                     'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
                     'legendgroup': ('Equale Brass Ensemble, John El' ... 'Monteverdi Orchestra and Choir'),
                     'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
                     'name': ('Equale Brass Ensemble, John El' ... 'Monteverdi Orchestra and Choir'),
                     'offsetgroup': ('Equale Brass Ensemble, John El' ... 'Monteverdi Orchestra and Choir'),
                     'orientation': 'v',
                     'showlegend': True,
                     'textposition': 'auto',
                     'type': 'bar',
                     'x': array(['Purcell: Music for the Queen Mary'], dtype=object),
                     'xaxis': 'x',
                     'y': array(['Equale Brass Ensemble, John Eliot Gardiner & Munich Monteverdi Orchestra and C
hoir'],
                                dtype=object),
                     'yaxis': 'y'},
                    {'alignmentgroup': 'True',
                     'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
                     'legendgroup': "Kent Nagano and Orchestre de l'Opéra de Lyon",
                     'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
                     'name': "Kent Nagano and Orchestre de l'Opéra de Lyon",
                     'offsetgroup': "Kent Nagano and Orchestre de l'Opéra de Lyon",
                     'orientation': 'v',
                     'showlegend': True,
                     'textposition': 'auto',
                     'type': 'bar',
                     'x': array(['Weill: The Seven Deadly Sins'], dtype=object),
                     'xaxis': 'x',
                     'y': array(["Kent Nagano and Orchestre de l'Opéra de Lyon"], dtype=object),
                     'yaxis': 'y'},
                    {'alignmentgroup': 'True',
                     'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
                     'legendgroup': 'Julian Bream',
                     'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
                     'name': 'Julian Bream',
                     'offsetgroup': 'Julian Bream',
                     'orientation': 'v',
                     'showlegend': True,
                     'textposition': 'auto',
                     'type': 'bar',
                     'x': array(['J.S. Bach: Chaconne, Suite in E Minor, Partita in E Major & Prelude, Fugue and
Allegro'],
```

```
           dtype=object),
  'xaxis': 'x',
  'y': array(['Julian Bream'], dtype=object),
  'yaxis': 'y'},
 {'alignmentgroup': 'True',
  'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
  'legendgroup': 'Martin Roscoe',
  'marker': {'color': '#FFA15A', 'pattern': {'shape': ''}},
  'name': 'Martin Roscoe',
  'offsetgroup': 'Martin Roscoe',
  'orientation': 'v',
  'showlegend': True,
  'textposition': 'auto',
  'type': 'bar',
  'x': array(['Szymanowski: Piano Works, Vol. 1'], dtype=object),
  'xaxis': 'x',
  'y': array(['Martin Roscoe'], dtype=object),
  'yaxis': 'y'},
 {'alignmentgroup': 'True',
  'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
  'legendgroup': 'Göteborgs Symfoniker & Neeme Järvi',
  'marker': {'color': '#19d3f3', 'pattern': {'shape': ''}},
  'name': 'Göteborgs Symfoniker & Neeme Järvi',
  'offsetgroup': 'Göteborgs Symfoniker & Neeme Järvi',
  'orientation': 'v',
  'showlegend': True,
  'textposition': 'auto',
  'type': 'bar',
  'x': array(['Nielsen: The Six Symphonies'], dtype=object),
  'xaxis': 'x',
  'y': array(['Göteborgs Symfoniker & Neeme Järvi'], dtype=object),
  'yaxis': 'y'},
 {'alignmentgroup': 'True',
  'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
  'legendgroup': 'Itzhak Perlman',
  'marker': {'color': '#FF6692', 'pattern': {'shape': ''}},
  'name': 'Itzhak Perlman',
  'offsetgroup': 'Itzhak Perlman',
  'orientation': 'v',
  'showlegend': True,
  'textposition': 'auto',
  'type': 'bar',
```

```
 'x': array(["Great Recordings of the Century: Paganini's 24 Caprices"], dtype=object),
 'xaxis': 'x',
 'y': array(['Itzhak Perlman'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Michele Campanella',
 'marker': {'color': '#B6E880', 'pattern': {'shape': ''}},
 'name': 'Michele Campanella',
 'offsetgroup': 'Michele Campanella',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(["Liszt - 12 Études D'Execution Transcendante"], dtype=object),
 'xaxis': 'x',
 'y': array(['Michele Campanella'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Gerald Moore',
 'marker': {'color': '#FF97FF', 'pattern': {'shape': ''}},
 'name': 'Gerald Moore',
 'offsetgroup': 'Gerald Moore',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
 'type': 'bar',
 'x': array(['Great Recordings of the Century - Shubert: Schwanengesang, 4 Lieder'],
           dtype=object),
 'xaxis': 'x',
 'y': array(['Gerald Moore'], dtype=object),
 'yaxis': 'y'},
{'alignmentgroup': 'True',
 'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
 'legendgroup': 'Mela Tenenbaum, Pro Musica Prague & Richard Kapp',
 'marker': {'color': '#FECB52', 'pattern': {'shape': ''}},
 'name': 'Mela Tenenbaum, Pro Musica Prague & Richard Kapp',
 'offsetgroup': 'Mela Tenenbaum, Pro Musica Prague & Richard Kapp',
 'orientation': 'v',
 'showlegend': True,
 'textposition': 'auto',
```

```
                     'type': 'bar',
                     'x': array(['Locatelli: Concertos for Violin, Strings and Continuo, Vol. 3'],
                                dtype=object),
                     'xaxis': 'x',
                     'y': array(['Mela Tenenbaum, Pro Musica Prague & Richard Kapp'], dtype=object),
                     'yaxis': 'y'},
                    {'alignmentgroup': 'True',
                     'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
                     'legendgroup': 'Emerson String Quartet',
                     'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
                     'name': 'Emerson String Quartet',
                     'offsetgroup': 'Emerson String Quartet',
                     'orientation': 'v',
                     'showlegend': True,
                     'textposition': 'auto',
                     'type': 'bar',
                     'x': array(["Schubert: The Late String Quartets & String Quintet (3 CD's)"],
                                dtype=object),
                     'xaxis': 'x',
                     'y': array(['Emerson String Quartet'], dtype=object),
                     'yaxis': 'y'},
                    {'alignmentgroup': 'True',
                     'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
                     'legendgroup': ('C. Monteverdi, Nigel Rogers - ' ... 'roque; London Cornett & Sackbu'),
                     'marker': {'color': '#EF553B', 'pattern': {'shape': ''}},
                     'name': ('C. Monteverdi, Nigel Rogers - ' ... 'roque; London Cornett & Sackbu'),
                     'offsetgroup': ('C. Monteverdi, Nigel Rogers - ' ... 'roque; London Cornett & Sackbu'),
                     'orientation': 'v',
                     'showlegend': True,
                     'textposition': 'auto',
                     'type': 'bar',
                     'x': array(["Monteverdi: L'Orfeo"], dtype=object),
                     'xaxis': 'x',
                     'y': array(['C. Monteverdi, Nigel Rogers - Chiaroscuro; London Baroque; London Cornett & Sa
ckbu'],
                                dtype=object),
                     'yaxis': 'y'},
                    {'alignmentgroup': 'True',
                     'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
                     'legendgroup': 'Nash Ensemble',
                     'marker': {'color': '#00cc96', 'pattern': {'shape': ''}},
                     'name': 'Nash Ensemble',
```

```
                    'offsetgroup': 'Nash Ensemble',
                    'orientation': 'v',
                    'showlegend': True,
                    'textposition': 'auto',
                    'type': 'bar',
                    'x': array(['Mozart: Chamber Music'], dtype=object),
                    'xaxis': 'x',
                    'y': array(['Nash Ensemble'], dtype=object),
                    'yaxis': 'y'},
                   {'alignmentgroup': 'True',
                    'hovertemplate': 'ArtistName=%{y}<br>AlbumTitle=%{x}<extra></extra>',
                    'legendgroup': 'Philip Glass Ensemble',
                    'marker': {'color': '#ab63fa', 'pattern': {'shape': ''}},
                    'name': 'Philip Glass Ensemble',
                    'offsetgroup': 'Philip Glass Ensemble',
                    'orientation': 'v',
                    'showlegend': True,
                    'textposition': 'auto',
                    'type': 'bar',
                    'x': array(['Koyaanisqatsi (Soundtrack from the Motion Picture)'], dtype=object),
                    'xaxis': 'x',
                    'y': array(['Philip Glass Ensemble'], dtype=object),
                    'yaxis': 'y'}],
        'layout': {'barmode': 'relative',
                   'legend': {'title': {'text': 'ArtistName'}, 'tracegroupgap': 0},
                   'margin': {'t': 60},
                   'template': '...',
                   'xaxis': {'anchor': 'y',
                             'categoryorder': 'total ascending',
                             'domain': [0.0, 1.0],
                             'title': {'text': 'AlbumTitle'}},
                   'yaxis': {'anchor': 'x',
                             'categoryarray': [Philip Glass Ensemble, Nash Ensemble,
                                               C. Monteverdi, Nigel Rogers -
                                               Chiaroscuro; London Baroque; London
                                               Cornett & Sackbu, ..., Aerosmith,
                                               Accept, AC/DC],
                             'categoryorder': 'array',
                             'domain': [0.0, 1.0],
                             'title': {'text': 'ArtistName'}}}
})
```

In [24]:
```
question = """
    Find all tracks with a name containing "What" (case-insensitive)
"""

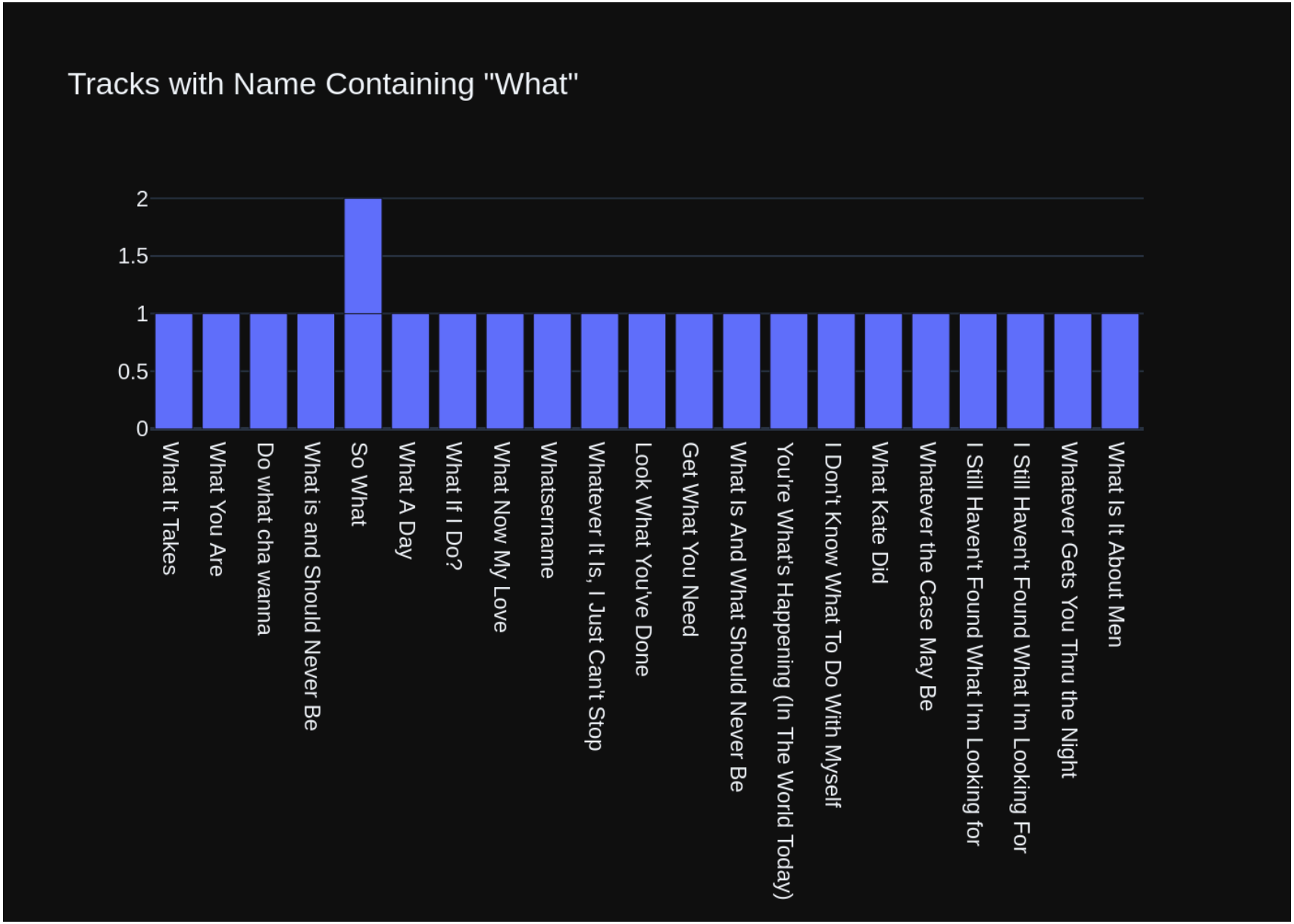
vn.ask(question=question)
```

```
Number of requested results 10 is greater than number of elements in index 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_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    A.Title AS AlbumTitle,\n    Ar.Name
AS ArtistName\nFROM \n    Album AS A\nJOIN \n    Artist AS Ar ON A.ArtistId = Ar.ArtistId;', 'Show me a lis
t of tables in the SQLite database', 'SELECT \n    T.Name\nFROM\n    Playlist AS T;', 'what are the top 5 c
ountries that customers come from?', 'SELECT \n    C.Country,\n    COUNT(*) AS CustomerCount\nFROM \n    Cu
stomer AS C\nGROUP BY \n    C.Country\nORDER BY \n    CustomerCount DESC\nLIMIT \n    5;', 'How many custom
ers are there', 'SELECT \n    COUNT(*)\nFROM \n    Customer;', 'How many records are in table called custom
er', 'SELECT \n    COUNT(*)\nFROM \n    Customer;', '  \n    Find all tracks with a name containing "What"
(case-insensitive)\n']
```sql
SELECT
    T.Name AS TrackName
FROM
    Track AS T
WHERE
    UPPER(T.Name) LIKE '%WHAT%';
```
```

```
SELECT
    T.Name AS TrackName
FROM
    Track AS T
WHERE
    UPPER(T.Name) LIKE '%WHAT%';
SELECT
    T.Name AS TrackName
FROM
    Track AS T
WHERE
    UPPER(T.Name) LIKE '%WHAT%';
```

	TrackName
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 Name Containing "What"
2
1.5
1
0.5
0
What It Takes
What You Are
Do what cha wanna
What is and Should Never Be
So What
What A Day
What If I Do?
What Now My Love
Whatsername
Whatever It Is, I Just Can't Stop
Look What You've Done
Get What You Need
What Is And What Should Never Be
You're What's Happening (In The World Today)
I Don't Know What To Do With Myself
What Kate Did
Whatever the Case May Be
I Still Haven't Found What I'm Looking for
I Still Haven't Found What I'm Looking For
Whatever Gets You Thru the Night
What Is It About Men

```
Out[24]: ("SELECT \n    T.Name AS TrackName\nFROM \n    Track AS T\nWHERE \n    UPPER(T.Name) LIKE '%WHAT%';",
                                          TrackName
          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': [{'name': 'Number of Tracks',
                        '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),
                        'y': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                              1]}],
```

```
    'layout': {'template': '...', 'title': {'text': 'Tracks with Name Containing "What"'}}
})
```

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 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 INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)\n\nCREATE TABLE Invoice\n(\n   InvoiceId INTEGER NOT NULL,\n   CustomerId INTEGER NOT NULL,\n   InvoiceDate DATETIME NOT NULL,\n   BillingAddress NVARCHAR(70),\n   BillingCity NVARCHAR(40),\n   BillingState NVARCHAR(40),\n   BillingCountry NVARCHAR(40),\n   BillingPostalCode NVARCHAR(10),\n   Total NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Invoice PRIMARY KEY (InvoiceId),\n   FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceLineInvoiceId ON InvoiceLine (InvoiceId)\n\nCREATE TABLE InvoiceLine\n(\n   InvoiceLineId INTEGER NOT NULL,\n   InvoiceId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   Quantity INTEGER NOT NULL,\n   CONSTRAINT PK_InvoiceLine PRIMARY KEY (InvoiceLineId),\n   FOREIGN KEY (InvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceLineTrackId ON InvoiceLine (TrackId)\n\nCREATE TABLE Customer\n(\n   CustomerId INTEGER NOT NULL,\n   FirstName NVARCHAR(40) NOT NULL,\n   LastName NVARCHAR(20) NOT NULL,\n   Company NVARCHAR(80),\n   Address NVARCHAR(70),\n   City NVARCHAR(40),\n   State NVARCHAR(40),\n   Country NVARCHAR(40),\n   PostalCode NVARCHAR(10),\n   Phone NVARCHAR(24),\n   Fax NVARCHAR(24),\n   Email NVARCHAR(60) NOT NULL,\n   SupportRepId INTEGER,\n   CONSTRAINT PK_Customer PRIMARY KEY (CustomerId),\n   FOREIGN KEY (SupportRepId) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_CustomerSupportRepId ON Customer (SupportRepId)\n\nCREATE INDEX IFK_EmployeeReportsTo ON Employee (ReportsTo)\n\nCREATE TABLE Employee\n(\n   EmployeeId INTEGER NOT NULL,\n   LastName NVARCHAR(20) NOT NULL,\n   FirstName NVARCHAR(20) NOT NULL,\n   Title NVARCHAR(30),\n   ReportsTo INTEGER,\n   BirthDate DATETIME,\n   HireDate DATETIME,\n   Address NVARCHAR(70),\n   City NVARCHAR(40),\n   State NVARCHAR(40),\n   Country NVARCHAR(40),\n   PostalCode NVARCHAR(10),\n   Phone NVARCHAR(24),\n   Fax NVARCHAR(24),\n   Email NVARCHAR(60),\n   CONSTRAINT PK_Employee PRIMARY KEY (EmployeeId),\n   FOREIGN KEY (ReportsTo) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Track\n(\n   TrackId INTEGER NOT NULL,\n   Name NVARCHAR(200) NOT NULL,\n   AlbumId INTEGER,\n   MediaTypeId INTEGER NOT NULL,\n   GenreId INTEGER,\n   Composer NVARCHAR(220),\n   Milliseconds INTEGER NOT NULL,\n   Bytes INTEGER,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Track PRIMARY KEY (TrackId),\n   FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n===Additional Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficient, please generate a valid SQL query without any explanations for the question. \n2. If the provided context is almost sufficient but requires knowledge of a specific string in a particular column, please generate an intermediate SQL query to find the distinct strings in that column. Prepend the query with a comment saying intermediate_sql \n3. If the provided context is insufficient, please explain why it can't be generated. \n4. Please use the most relevant table(s). \n5. If the question has been asked and answered before, please repeat the answer exactly as it was given before. \n", '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;', 'what are the top 5 countries that customers come from?', 'SELECT \n   C.Country,\n   COUNT(*) AS CustomerCount\nFROM \n   Customer AS C\nGROUP BY \n   C.Country\nORDER BY \n   CustomerCount DESC\nLIMIT \n

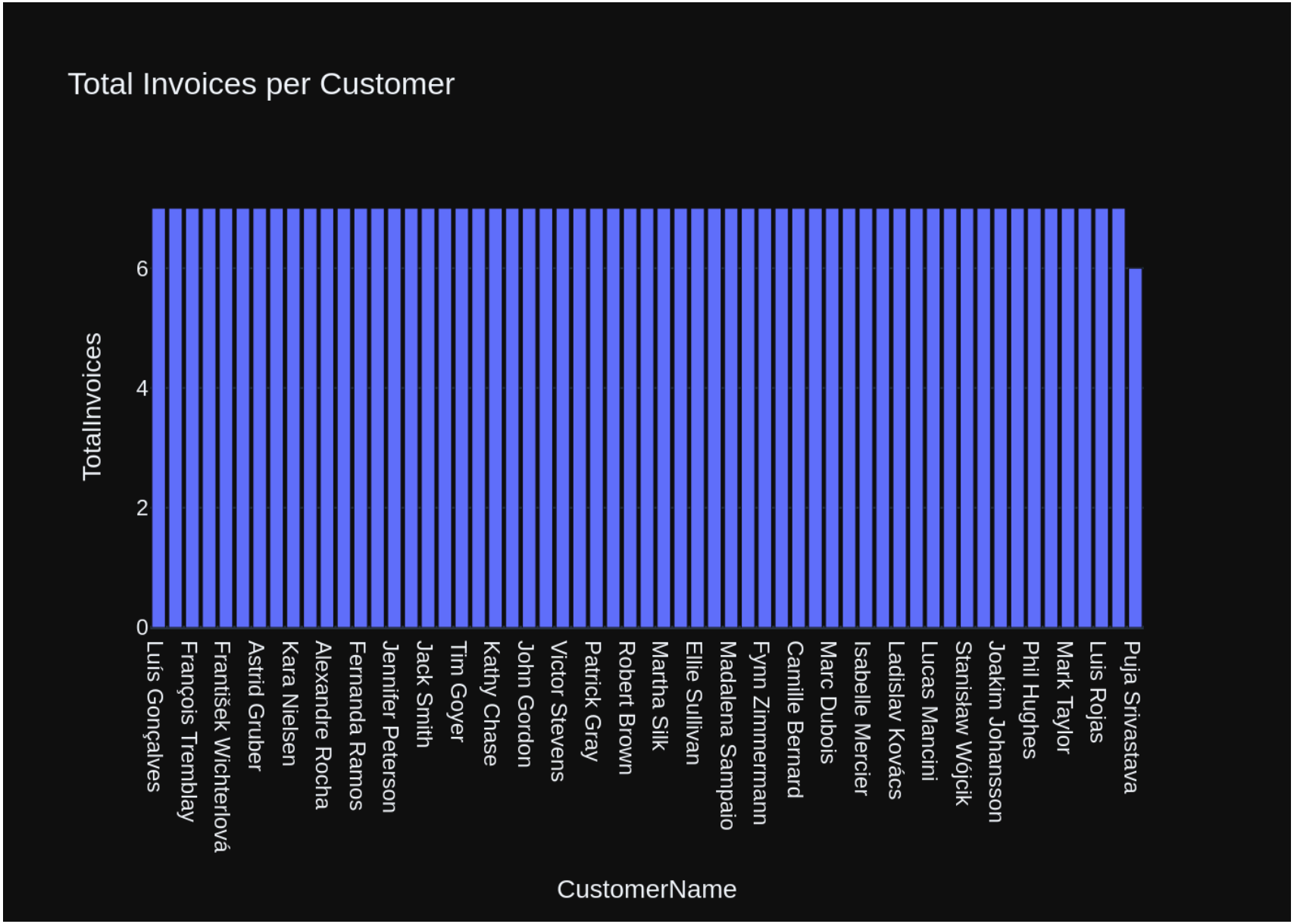
```
5;', '  \n    List all albums and their corresponding artist names  \n', 'SELECT \n    A.Title AS AlbumTitl
e,\n    Ar.Name AS ArtistName\nFROM \n    Album AS A\nJOIN \n    Artist AS Ar ON A.ArtistId = Ar.ArtistI
d;', '  \n    Find all tracks with a name containing "What" (case-insensitive)\n', "SELECT \n    T.Name AS
TrackName\nFROM \n    Track AS T\nWHERE \n    UPPER(T.Name) LIKE '%WHAT%';", 'Show me a list of tables in t
he SQLite database', 'SELECT \n    T.Name\nFROM\n    Playlist AS T;', '  \n    Get the total number of invo
ices for each customer\n']
```

```sql
SELECT
    C.FirstName || ' ' || C.LastName AS CustomerName,
    COUNT(I.InvoiceId) AS TotalInvoices
FROM
    Customer AS C
LEFT JOIN
    Invoice AS I ON C.CustomerId = I.CustomerId
GROUP BY
    C.CustomerId;
```

```
SELECT
    C.FirstName || ' ' || C.LastName AS CustomerName,
    COUNT(I.InvoiceId) AS TotalInvoices
FROM
    Customer AS C
LEFT JOIN
    Invoice AS I ON C.CustomerId = I.CustomerId
GROUP BY
    C.CustomerId;
SELECT
    C.FirstName || ' ' || C.LastName AS CustomerName,
    COUNT(I.InvoiceId) AS TotalInvoices
FROM
    Customer AS C
LEFT JOIN
    Invoice AS I ON C.CustomerId = I.CustomerId
GROUP BY
    C.CustomerId;
             CustomerName  TotalInvoices
0         Luís Gonçalves              7
1          Leonie Köhler              7
2       François Tremblay             7
3            Bjørn Hansen             7
4   František Wichterlová              7
5             Helena Holý              7
```

6	Astrid Gruber	7
7	Daan Peeters	7
8	Kara Nielsen	7
9	Eduardo Martins	7
10	Alexandre Rocha	7
11	Roberto Almeida	7
12	Fernanda Ramos	7
13	Mark Philips	7
14	Jennifer Peterson	7
15	Frank Harris	7
16	Jack Smith	7
17	Michelle Brooks	7
18	Tim Goyer	7
19	Dan Miller	7
20	Kathy Chase	7
21	Heather Leacock	7
22	John Gordon	7
23	Frank Ralston	7
24	Victor Stevens	7
25	Richard Cunningham	7
26	Patrick Gray	7
27	Julia Barnett	7
28	Robert Brown	7
29	Edward Francis	7
30	Martha Silk	7
31	Aaron Mitchell	7
32	Ellie Sullivan	7
33	João Fernandes	7
34	Madalena Sampaio	7
35	Hannah Schneider	7
36	Fynn Zimmermann	7
37	Niklas Schröder	7
38	Camille Bernard	7
39	Dominique Lefebvre	7
40	Marc Dubois	7
41	Wyatt Girard	7
42	Isabelle Mercier	7
43	Terhi Hämäläinen	7
44	Ladislav Kovács	7
45	Hugh O'Reilly	7
46	Lucas Mancini	7
47	Johannes Van der Berg	7

48	Stanisław Wójcik	7
49	Enrique Muñoz	7
50	Joakim Johansson	7
51	Emma Jones	7
52	Phil Hughes	7
53	Steve Murray	7
54	Mark Taylor	7
55	Diego Gutiérrez	7
56	Luis Rojas	7
57	Manoj Pareek	7
58	Puja Srivastava	6


Total Invoices per Customer
TotalInvoices
0
2
4
6
CustomerName
Luís Gonçalves
François Tremblay
František Wichterlová
Astrid Gruber
Kara Nielsen
Alexandre Rocha
Fernanda Ramos
Jennifer Peterson
Jack Smith
Tim Goyer
Kathy Chase
John Gordon
Victor Stevens
Patrick Gray
Robert Brown
Martha Silk
Ellie Sullivan
Madalena Sampaio
Fynn Zimmermann
Camille Bernard
Marc Dubois
Isabelle Mercier
Ladislav Kovács
Lucas Mancini
Stanisław Wójcik
Joakim Johansson
Phil Hughes
Mark Taylor
Luis Rojas
Puja Srivastava

Out[25]: ("SELECT \n   C.FirstName || ' ' || C.LastName AS CustomerName,\n   COUNT(I.InvoiceId) AS TotalInvoices \nFROM \n   Customer AS C\nLEFT JOIN \n   Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n   C.CustomerId;",

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

```
38        Camille Bernard             7
39     Dominique Lefebvre             7
40            Marc Dubois             7
41           Wyatt Girard             7
42       Isabelle Mercier             7
43       Terhi Hämäläinen             7
44        Ladislav Kovács             7
45          Hugh O'Reilly             7
46          Lucas Mancini             7
47  Johannes Van der Berg             7
48       Stanisław Wójcik             7
49          Enrique Muñoz             7
50       Joakim Johansson             7
51             Emma Jones             7
52            Phil Hughes             7
53           Steve Murray             7
54            Mark Taylor             7
55        Diego Gutiérrez             7
56             Luis Rojas             7
57           Manoj Pareek             7
58        Puja Srivastava             6,
Figure({
    'data': [{'alignmentgroup': 'True',
              'hovertemplate': 'CustomerName=%{x}<br>TotalInvoices=%{y}<extra></extra>',
              'legendgroup': '',
              'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
              'name': '',
              'offsetgroup': '',
              'orientation': 'v',
              'showlegend': False,
              'textposition': 'auto',
              'type': 'bar',
              'x': array(['Luís Gonçalves', 'Leonie Köhler', 'François Tremblay', 'Bjørn Hansen',
                          'František Wichterlová', 'Helena Holý', 'Astrid Gruber', 'Daan Peeters',
                          'Kara Nielsen', 'Eduardo Martins', 'Alexandre Rocha', 'Roberto Almeida',
                          'Fernanda Ramos', 'Mark Philips', 'Jennifer Peterson', 'Frank Harris',
                          'Jack Smith', 'Michelle Brooks', 'Tim Goyer', 'Dan Miller',
                          'Kathy Chase', 'Heather Leacock', 'John Gordon', 'Frank Ralston',
                          'Victor Stevens', 'Richard Cunningham', 'Patrick Gray', 'Julia Barnett',
                          'Robert Brown', 'Edward Francis', 'Martha Silk', 'Aaron Mitchell',
                          'Ellie Sullivan', 'João Fernandes', 'Madalena Sampaio',
                          'Hannah Schneider', 'Fynn Zimmermann', 'Niklas Schröder',
```

```
                                'Camille Bernard', 'Dominique Lefebvre', 'Marc Dubois', 'Wyatt Girard',
                                'Isabelle Mercier', 'Terhi Hämäläinen', 'Ladislav Kovács',
                                "Hugh O'Reilly", 'Lucas Mancini', 'Johannes Van der Berg',
                                'Stanisław Wójcik', 'Enrique Muñoz', 'Joakim Johansson', 'Emma Jones',
                                'Phil Hughes', 'Steve Murray', 'Mark Taylor', 'Diego Gutiérrez',
                                'Luis Rojas', 'Manoj Pareek', 'Puja Srivastava'], dtype=object),
                  'xaxis': 'x',
                  'y': array([7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
                              7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
                              7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 6]),
                  'yaxis': 'y'}],
       'layout': {'barmode': 'relative',
                  'legend': {'tracegroupgap': 0},
                  'template': '...',
                  'title': {'text': 'Total Invoices per Customer'},
                  'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'CustomerName'}},
                  '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 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 Invoice\n(\n   InvoiceId INTEGER NOT NULL,\n   CustomerId INTEGER NOT NULL,\n   InvoiceDate DATETIME NOT NULL,\n   BillingAddress NVARCHAR(70),\n   BillingCity NVARCHAR(40),\n   BillingState NVARCHAR(40),\n   BillingCountry NVARCHAR(40),\n   BillingPostalCode NVARCHAR(10),\n   Total NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Invoice PRIMARY KEY (InvoiceId),\n   FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceLineInvoiceId ON InvoiceLine (InvoiceId)\n\nCREATE INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)\n\nCREATE TABLE InvoiceLine\n(\n   InvoiceLineId INTEGER NOT NULL,\n   InvoiceId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   Quantity INTEGER NOT NULL,\n   CONSTRAINT PK_InvoiceLine PRIMARY KEY (InvoiceLineId),\n   FOREIGN KEY (InvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceLineTrackId ON InvoiceLine (TrackId)\n\nCREATE TABLE Customer\n(\n   CustomerId INTEGER NOT NULL,\n   FirstName NVARCHAR(40) NOT NULL,\n   LastName NVARCHAR(20) NOT NULL,\n   Company NVARCHAR(80),\n   Address NVARCHAR(70),\n   City NVARCHAR(40),\n   State NVARCHAR(40),\n   Country NVARCHAR(40),\n   PostalCode NVARCHAR(10),\n   Phone NVARCHAR(24),\n   Fax NVARCHAR(24),\n   Email NVARCHAR(60) NOT NULL,\n   SupportRepId INTEGER,\n   CONSTRAINT PK_Customer PRIMARY KEY (CustomerId),\n   FOREIGN KEY (SupportRepId) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Employee\n(\n   EmployeeId INTEGER NOT NULL,\n   LastName NVARCHAR(20) NOT NULL,\n   FirstName NVARCHAR(20) NOT NULL,\n   Title NVARCHAR(30),\n   ReportsTo INTEGER,\n   BirthDate DATETIME,\n   HireDate DATETIME,\n   Address NVARCHAR(70),\n   City NVARCHAR(40),\n   State NVARCHAR(40),\n   Country NVARCHAR(40),\n   PostalCode NVARCHAR(10),\n   Phone NVARCHAR(24),\n   Fax NVARCHAR(24),\n   Email NVARCHAR(60),\n   CONSTRAINT PK_Employee PRIMARY KEY (EmployeeId),\n   FOREIGN KEY (ReportsTo) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Track\n(\n   TrackId INTEGER NOT NULL,\n   Name NVARCHAR(200) NOT NULL,\n   AlbumId INTEGER,\n   MediaTypeId INTEGER NOT NULL,\n   GenreId INTEGER,\n   Composer NVARCHAR(220),\n   Milliseconds INTEGER NOT NULL,\n   Bytes INTEGER,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Track PRIMARY KEY (TrackId),\n   FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_EmployeeReportsTo ON Employee (ReportsTo)\n\nCREATE TABLE Album\n(\n   AlbumId INTEGER NOT NULL,\n   Title NVARCHAR(160) NOT NULL,\n   ArtistId INTEGER NOT NULL,\n   CONSTRAINT PK_Album PRIMARY KEY (AlbumId),\n   FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n===Additional Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficient, please generate a valid SQL query without any explanations for the question. \n2. If the provided context is almost sufficient but requires knowledge of a specific string in a particular column, please generate an intermediate SQL query to find the distinct strings in that column. Prepend the query with a comment saying intermediate_sql \n3. If the provided context is insufficient, please explain why it can't be generated. \n4. Please use the most relevant table(s). \n5. If the question has been asked and answered before, please repeat the answer exactly as it was given before. \n", ' \n   Get the total number of invoices for each customer\n', "SELECT \n   C.FirstName || ' ' || C.LastName AS CustomerName,\n   COUNT(I.InvoiceId) AS TotalInvoices\nFROM \n

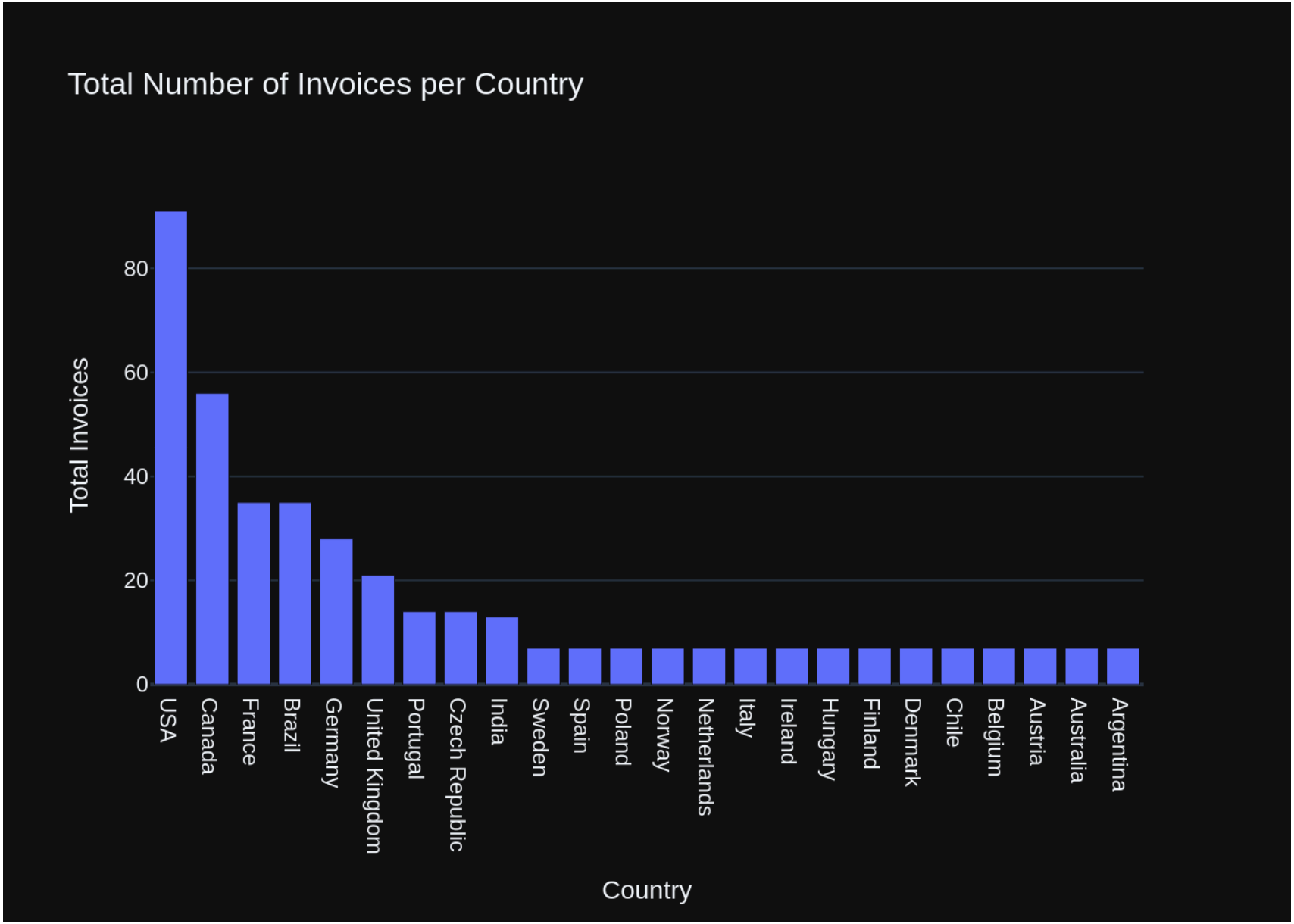
Customer AS C\nLEFT JOIN \n   Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n   C.CustomerId;",
'what are the top 5 countries that customers come from?', 'SELECT \n   C.Country,\n   COUNT(*) AS Custome
rCount\nFROM \n   Customer AS C\nGROUP BY \n   C.Country\nORDER BY \n   CustomerCount DESC\nLIMIT \n
5;', 'How many records are in table called customer', 'SELECT \n   COUNT(*)\nFROM \n   Customer;', 'How m
any customers are there', 'SELECT \n   COUNT(*)\nFROM \n   Customer;', ' \n   List all albums and their
corresponding artist names \n', 'SELECT \n   A.Title AS AlbumTitle,\n   Ar.Name AS ArtistName\nFROM \n
Album AS A\nJOIN \n   Artist AS Ar ON A.ArtistId = Ar.ArtistId;', ' \n   Find all tracks with a name con
taining "What" (case-insensitive)\n', "SELECT \n   T.Name AS TrackName\nFROM \n   Track AS T\nWHERE \n
UPPER(T.Name) LIKE '%WHAT%';", 'Show me a list of tables in the SQLite database', 'SELECT \n   T.Name\nFRO
M\n   Playlist AS T;', ' \n   Find the total number of invoices per country:\n']

```sql
SELECT
    C.Country,
    COUNT(I.InvoiceId) AS TotalInvoices
FROM
    Customer AS C
LEFT JOIN
    Invoice AS I ON C.CustomerId = I.CustomerId
GROUP BY
    C.Country
ORDER BY
    TotalInvoices DESC;
```

SELECT
    C.Country,
    COUNT(I.InvoiceId) AS TotalInvoices
FROM
    Customer AS C
LEFT JOIN
    Invoice AS I ON C.CustomerId = I.CustomerId
GROUP BY
    C.Country
ORDER BY
    TotalInvoices DESC;
SELECT
    C.Country,
    COUNT(I.InvoiceId) AS TotalInvoices
FROM
    Customer AS C
LEFT JOIN
    Invoice AS I ON C.CustomerId = I.CustomerId
GROUP BY

```
    C.Country
ORDER BY
    TotalInvoices DESC;
```

	Country	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 Number of 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    C.Country,\n    COUNT(I.InvoiceId) AS TotalInvoices\nFROM \n    Customer AS C\nLEFT JOIN \n
Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n    C.Country\nORDER BY \n    TotalInvoices DES
C;',
            Country  TotalInvoices
 0             USA            91
 1          Canada            56
 2          France            35
 3          Brazil            35
 4         Germany            28
 5   United Kingdom            21
 6        Portugal            14
 7   Czech Republic            14
 8           India            13
 9          Sweden             7
 10          Spain             7
 11         Poland             7
 12         Norway             7
 13    Netherlands             7
 14          Italy             7
 15        Ireland             7
 16        Hungary             7
 17        Finland             7
 18        Denmark             7
 19          Chile             7
 20        Belgium             7
 21        Austria             7
 22      Australia             7
 23      Argentina             7,
 Figure({
     'data': [{'alignmentgroup': 'True',
               'hovertemplate': 'Country=%{x}<br>TotalInvoices=%{y}<extra></extra>',
               'legendgroup': '',
               'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
               'name': '',
               'offsetgroup': '',
               'orientation': 'v',
               'showlegend': False,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['USA', 'Canada', 'France', 'Brazil', 'Germany', 'United Kingdom',
                           'Portugal', 'Czech Republic', 'India', 'Sweden', 'Spain', 'Poland',
                           'Norway', 'Netherlands', 'Italy', 'Ireland', 'Hungary', 'Finland',
```

```
                                'Denmark', 'Chile', 'Belgium', 'Austria', 'Australia', 'Argentina'],
                               dtype=object),
                  'xaxis': 'x',
                  'y': array([91, 56, 35, 35, 28, 21, 14, 14, 13,  7,  7,  7,  7,  7,  7,  7,  7,  7,
                               7,  7,  7,  7,  7,  7]),
                  'yaxis': 'y'}],
      'layout': {'barmode': 'relative',
                 'legend': {'tracegroupgap': 0},
                 'template': '...',
                 'title': {'text': 'Total Number of Invoices per Country'},
                 'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'Country'}},
                 'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], '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 8, updating n_results = 8
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

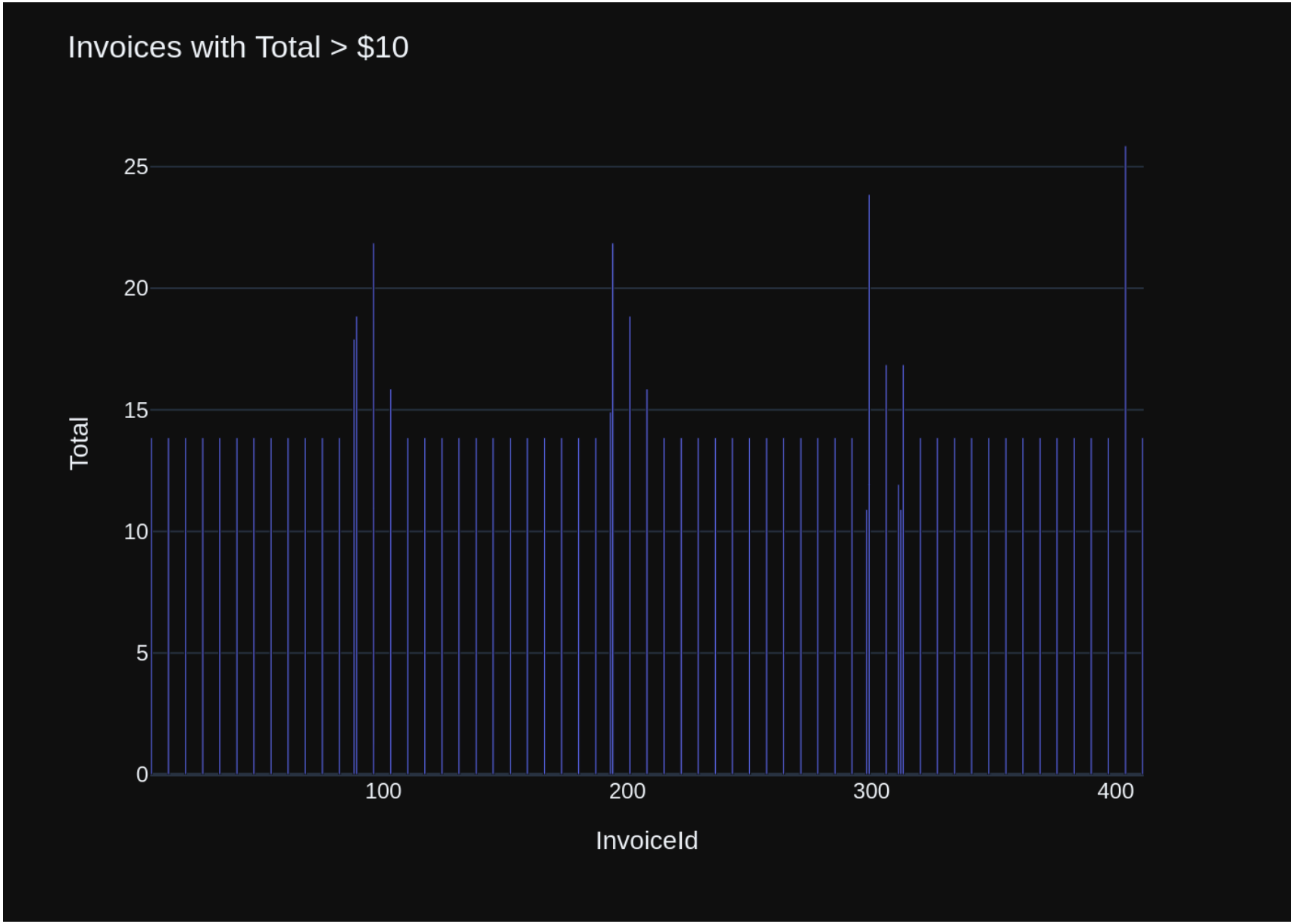
["You are a SQLite expert. Please help to generate a SQL query to answer the question. Your response should ONLY be based on the given context and follow the response guidelines and format instructions. \n===Tables \nCREATE TABLE InvoiceLine\n(\n   InvoiceLineId INTEGER NOT NULL,\n   InvoiceId INTEGER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

```
Invoices\nFROM \n    Customer AS C\nLEFT JOIN \n    Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY
\n    C.Country\nORDER BY \n    TotalInvoices DESC;', 'How many records are in table called customer', 'SEL
ECT \n    COUNT(*)\nFROM \n    Customer;', 'How many customers are there', 'SELECT \n    COUNT(*)\nFROM \n
Customer;', 'what are the top 5 countries that customers come from?', 'SELECT \n    C.Country,\n    COUNT
(*) AS CustomerCount\nFROM \n    Customer AS C\nGROUP BY \n    C.Country\nORDER BY \n    CustomerCount DESC
\nLIMIT \n    5;', '  \n    List all albums and their corresponding artist names  \n', 'SELECT \n    A.Titl
e AS AlbumTitle,\n    Ar.Name AS ArtistName\nFROM \n    Album AS A\nJOIN \n    Artist AS Ar ON A.ArtistId =
Ar.ArtistId;', '  \n    Find all tracks with a name containing "What" (case-insensitive)\n', "SELECT \n
T.Name AS TrackName\nFROM \n    Track AS T\nWHERE \n    UPPER(T.Name) LIKE '%WHAT%';", 'Show me a list of t
ables in the SQLite database', 'SELECT \n    T.Name\nFROM\n    Playlist AS T;', '  \n    List all invoices
with a total exceeding $10:\n']
```

```sql
SELECT
    *
FROM
    Invoice
WHERE
    Total > 10;
```

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

```
63        411          44  2013-12-14 00:00:00            Porthaninkatu 9

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

[64 rows x 9 columns]
```


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

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

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

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

```
                           285, 292, 298, 299, 306, 311, 312, 313, 320, 327, 334, 341, 348, 355,
                           362, 369, 376, 383, 390, 397, 404, 411]),
                'xaxis': 'x',
                'y': array([13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86,
                           13.86, 13.86, 17.91, 18.86, 21.86, 15.86, 13.86, 13.86, 13.86, 13.86,
                           13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 14.91, 21.86,
                           18.86, 15.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86,
                           13.86, 13.86, 13.86, 13.86, 10.91, 23.86, 16.86, 11.94, 10.91, 16.86,
                           13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86, 13.86,
                           13.86, 13.86, 25.86, 13.86]),
                'yaxis': 'y'}],
     'layout': {'barmode': 'relative',
                'legend': {'tracegroupgap': 0},
                'margin': {'t': 60},
                'template': '...',
                'title': {'text': 'Invoices with Total > $10'},
                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'InvoiceId'}},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'Total'}}}
 }))
```

In [28]:

```
question = """
    Find all invoices since 2010 and the total amount invoiced:
"""

vn.ask(question=question)
```

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

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

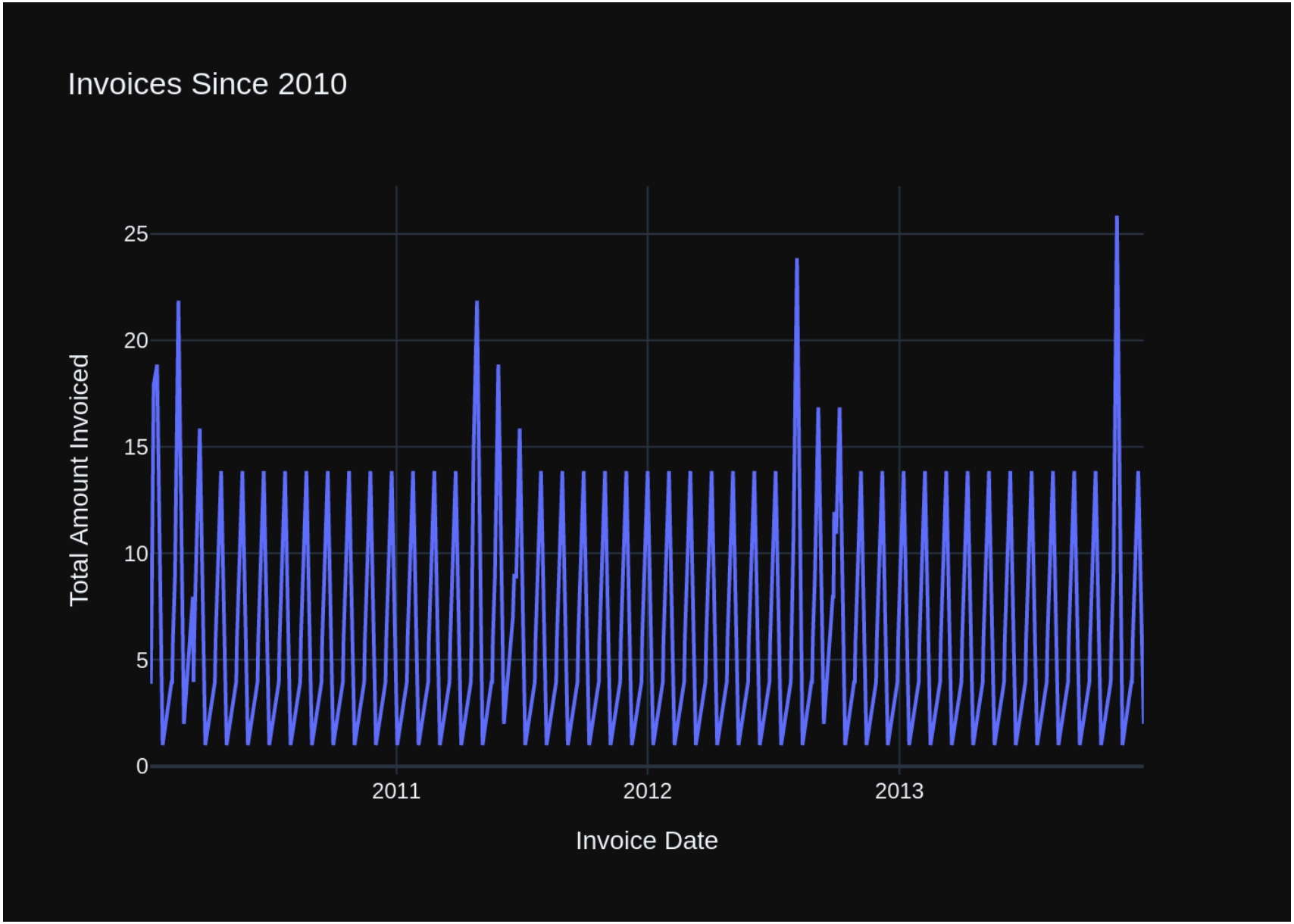
most relevant table(s). \n5. If the question has been asked and answered before, please repeat the answer e
xactly as it was given before. \n", ' \n   List all invoices with a total exceeding $10:\n', 'SELECT \n
*\nFROM \n   Invoice\nWHERE \n   Total > 10;', ' \n   Find the total number of invoices per countr
y:\n', 'SELECT \n   C.Country,\n   COUNT(I.InvoiceId) AS TotalInvoices\nFROM \n   Customer AS C\nLEFT JO
IN \n   Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n   C.Country\nORDER BY \n   TotalInvoice
s DESC;', ' \n   Get the total number of invoices for each customer\n', "SELECT \n   C.FirstName || ' '
|| C.LastName AS CustomerName,\n   COUNT(I.InvoiceId) AS TotalInvoices\nFROM \n   Customer AS C\nLEFT JOI
N \n   Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n   C.CustomerId;", 'How many records are i
n table called customer', 'SELECT \n   COUNT(*)\nFROM \n   Customer;', 'How many customers are there', 'S
ELECT \n   COUNT(*)\nFROM \n   Customer;', 'what are the top 5 countries that customers come from?', 'SEL
ECT \n   C.Country,\n   COUNT(*) AS CustomerCount\nFROM \n   Customer AS C\nGROUP BY \n   C.Country\nOR
DER BY \n   CustomerCount DESC\nLIMIT \n   5;', ' \n   Find all tracks with a name containing "What" (c
ase-insensitive)\n', "SELECT \n   T.Name AS TrackName\nFROM \n   Track AS T\nWHERE \n   UPPER(T.Name) LI
KE '%WHAT%';", ' \n   List all albums and their corresponding artist names \n', 'SELECT \n   A.Title AS
AlbumTitle,\n   Ar.Name AS ArtistName\nFROM \n   Album AS A\nJOIN \n   Artist AS Ar ON A.ArtistId = Ar.A
rtistId;', 'Show me a list of tables in the SQLite database', 'SELECT \n   T.Name\nFROM\n   Playlist AS
T;', ' \n   Find all invoices since 2010 and the total amount invoiced:\n']

```sql
SELECT
    I.InvoiceDate,
    SUM(IL.UnitPrice * IL.Quantity) AS TotalAmount
FROM
    Invoice AS I
JOIN
    InvoiceLine AS IL ON I.InvoiceId = IL.InvoiceId
WHERE
    I.InvoiceDate >= '2010-01-01'
GROUP BY
    I.InvoiceDate
ORDER BY
    I.InvoiceDate;
```

```
SELECT
    I.InvoiceDate,
    SUM(IL.UnitPrice * IL.Quantity) AS TotalAmount
FROM
    Invoice AS I
JOIN
    InvoiceLine AS IL ON I.InvoiceId = IL.InvoiceId
WHERE
    I.InvoiceDate >= '2010-01-01'
GROUP BY
```

```
    I.InvoiceDate
ORDER BY
    I.InvoiceDate;
SELECT
    I.InvoiceDate,
    SUM(IL.UnitPrice * IL.Quantity) AS TotalAmount
FROM
    Invoice AS I
JOIN
    InvoiceLine AS IL ON I.InvoiceId = IL.InvoiceId
WHERE
    I.InvoiceDate >= '2010-01-01'
GROUP BY
    I.InvoiceDate
ORDER BY
    I.InvoiceDate;
             InvoiceDate  TotalAmount
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]
```


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

Out[28]:
```
("SELECT \n    I.InvoiceDate,\n    SUM(IL.UnitPrice * IL.Quantity) AS TotalAmount\nFROM \n    Invoice AS I
\nJOIN \n    InvoiceLine AS IL ON I.InvoiceId = IL.InvoiceId\nWHERE \n    I.InvoiceDate >= '2010-01-01'\nG
ROUP BY \n    I.InvoiceDate\nORDER BY \n    I.InvoiceDate;",
              InvoiceDate  TotalAmount
 0    2010-01-08 00:00:00         3.96
 1    2010-01-09 00:00:00         3.96
 2    2010-01-10 00:00:00         6.94
 3    2010-01-13 00:00:00        17.91
 4    2010-01-18 00:00:00        18.86
 ..                   ...          ...
 277  2013-12-05 00:00:00         3.96
 278  2013-12-06 00:00:00         5.94
 279  2013-12-09 00:00:00         8.91
 280  2013-12-14 00:00:00        13.86
 281  2013-12-22 00:00:00         1.99

 [282 rows x 2 columns],
 Figure({
     'data': [{'mode': 'lines',
               'name': 'Total Amount Invoiced',
               'type': 'scatter',
               'x': array(['2010-01-08 00:00:00', '2010-01-09 00:00:00', '2010-01-10 00:00:00',
                           ..., '2013-12-09 00:00:00', '2013-12-14 00:00:00',
                           '2013-12-22 00:00:00'], dtype=object),
               'y': array([ 3.96,  3.96,  6.94, ...,  8.91, 13.86,  1.99])}],
     'layout': {'template': '...',
                'title': {'text': 'Invoices Since 2010'},
                'xaxis': {'title': {'text': 'Invoice Date'}},
                'yaxis': {'title': {'text': 'Total Amount Invoiced'}}}
 }))
```

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

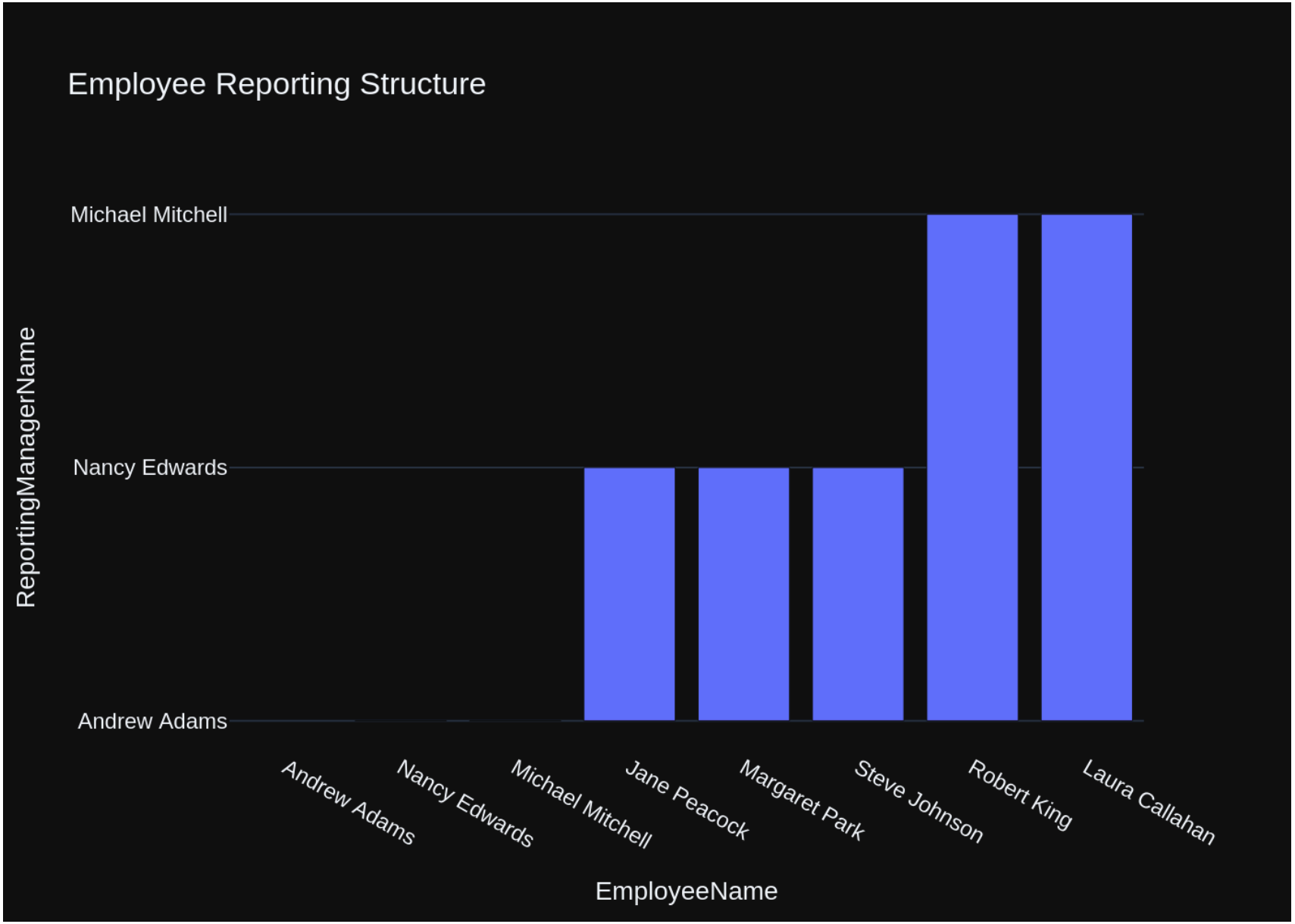
vn.ask(question=question)
```

```
Number of requested results 10 is greater than number of elements in index 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 INTEGER NOT NULL,\n   LastName NVARCHAR(20) NOT NULL,\n   FirstName NVARCHAR(20) NOT NULL,\n   Title NVARCHAR(30),\n   ReportsTo INTEGER,\n   BirthDate DATETIME,\n   HireDate DATETIME,\n   Address NVARCHAR(70),\n   City NVARCHAR(40),\n   State NVARCHAR(40),\n   Country NVARCHAR(40),\n   PostalCode NVARCHAR(10),\n   Phone NVARCHAR(24),\n   Fax NVARCHAR(24),\n   Email NVARCHAR(60),\n   CONSTRAINT PK_Employee PRIMARY KEY (EmployeeId),\n   FOREIGN KEY (ReportsTo) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Customer\n(\n   CustomerId INTEGER NOT NULL,\n   FirstName NVARCHAR(40) NOT NULL,\n   LastName NVARCHAR(20) NOT NULL,\n   Company NVARCHAR(80),\n   Address NVARCHAR(70),\n   City NVARCHAR(40),\n   State NVARCHAR(40),\n   Country NVARCHAR(40),\n   PostalCode NVARCHAR(10),\n   Phone NVARCHAR(24),\n   Fax NVARCHAR(24),\n   Email NVARCHAR(60) NOT NULL,\n   SupportRepId INTEGER,\n   CONSTRAINT PK_Customer PRIMARY KEY (CustomerId),\n   FOREIGN KEY (SupportRepId) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_CustomerSupportRepId ON Customer (SupportRepId)\n\nCREATE TABLE Invoice\n(\n   InvoiceId INTEGER NOT NULL,\n   CustomerId INTEGER NOT NULL,\n   InvoiceDate DATETIME NOT NULL,\n   BillingAddress NVARCHAR(70),\n   BillingCity NVARCHAR(40),\n   BillingState NVARCHAR(40),\n   BillingCountry NVARCHAR(40),\n   BillingPostalCode NVARCHAR(10),\n   Total NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Invoice PRIMARY KEY (InvoiceId),\n   FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE InvoiceLine\n(\n   InvoiceLineId INTEGER NOT NULL,\n   InvoiceId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   Quantity INTEGER NOT NULL,\n   CONSTRAINT PK_InvoiceLine PRIMARY KEY (InvoiceLineId),\n   FOREIGN KEY (InvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Track\n(\n   TrackId INTEGER NOT NULL,\n   Name NVARCHAR(200) NOT NULL,\n   AlbumId INTEGER,\n   MediaTypeId INTEGER NOT NULL,\n   GenreId INTEGER,\n   Composer NVARCHAR(220),\n   Milliseconds INTEGER NOT NULL,\n   Bytes INTEGER,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Track PRIMARY KEY (TrackId),\n   FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)\n\nCREATE TABLE Artist\n(\n   ArtistId INTEGER NOT NULL,\n   Name NVARCHAR(120),\n   CONSTRAINT PK_Artist PRIMARY KEY (ArtistId)\n)\n\nCREATE TABLE PlaylistTrack\n(\n   PlaylistId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   CONSTRAINT PK_PlaylistTrack PRIMARY KEY (PlaylistId, TrackId),\n   FOREIGN KEY (PlaylistId) REFERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n===Additional Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficient, please generate a valid SQL query without any explanations for the question. \n2. If the provided context is almost sufficient but requires knowledge of a specific string in a particular column, please generate an intermediate SQL query to find the distinct strings in that column. Prepend the query with a comment saying intermediate_sql \n3. If the provided context is insufficient, please explain why it can't be generated. \n4. Please use the most relevant table(s). \n5. If the question has been asked and answered before, please repeat the answer exactly as it was given before. \n", ' \n   G

```
et the total number of invoices for each customer\n', "SELECT \n    C.FirstName || ' ' || C.LastName AS Cus
tomerName,\n    COUNT(I.InvoiceId) AS TotalInvoices\nFROM \n    Customer AS C\nLEFT JOIN \n    Invoice AS I
ON C.CustomerId = I.CustomerId\nGROUP BY \n    C.CustomerId;", 'what are the top 5 countries that customers
come from?', 'SELECT \n    C.Country,\n    COUNT(*) AS CustomerCount\nFROM \n    Customer AS C\nGROUP BY \n
C.Country\nORDER BY \n    CustomerCount DESC\nLIMIT \n    5;', '  \n    Find all invoices since 2010 and th
e total amount invoiced:\n', "SELECT \n    I.InvoiceDate,\n    SUM(IL.UnitPrice * IL.Quantity) AS TotalAmou
nt\nFROM \n    Invoice AS I\nJOIN \n    InvoiceLine AS IL ON I.InvoiceId = IL.InvoiceId\nWHERE \n    I.Invo
iceDate >= '2010-01-01'\nGROUP BY \n    I.InvoiceDate\nORDER BY \n    I.InvoiceDate;", '  \n    Find the to
tal number of invoices per country:\n', 'SELECT \n    C.Country,\n    COUNT(I.InvoiceId) AS TotalInvoices\n
FROM \n    Customer AS C\nLEFT JOIN \n    Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n    C.Cou
ntry\nORDER BY \n    TotalInvoices DESC;', '  \n    List all invoices with a total exceeding $10:\n', 'SELE
CT \n    *\nFROM \n    Invoice\nWHERE \n    Total > 10;', '  \n    List all albums and their corresponding
artist names  \n', 'SELECT \n    A.Title AS AlbumTitle,\n    Ar.Name AS ArtistName\nFROM \n    Album AS A\n
JOIN \n    Artist AS Ar ON A.ArtistId = Ar.ArtistId;', '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    Find all tracks with a name containing "What" (case-insensitive)\n', "SELECT \n
T.Name AS TrackName\nFROM \n    Track AS T\nWHERE \n    UPPER(T.Name) LIKE '%WHAT%';", 'Show me a list of t
ables in the SQLite database', 'SELECT \n    T.Name\nFROM\n    Playlist AS T;', "  \n    List all employees
and their reporting manager's name (if any):\n"]
```sql
SELECT
    E.FirstName || ' ' || E.LastName AS EmployeeName,
    R.FirstName || ' ' || R.LastName AS ReportingManagerName
FROM
    Employee AS E
LEFT JOIN
    Employee AS R ON E.ReportsTo = R.EmployeeId;
```
SELECT
    E.FirstName || ' ' || E.LastName AS EmployeeName,
    R.FirstName || ' ' || R.LastName AS ReportingManagerName
FROM
    Employee AS E
LEFT JOIN
    Employee AS R ON E.ReportsTo = R.EmployeeId;
SELECT
    E.FirstName || ' ' || E.LastName AS EmployeeName,
    R.FirstName || ' ' || R.LastName AS ReportingManagerName
FROM
    Employee AS E
LEFT JOIN
    Employee AS R ON E.ReportsTo = R.EmployeeId;
```

	EmployeeName	ReportingManagerName
0	Andrew Adams	None
1	Nancy Edwards	Andrew Adams
2	Jane Peacock	Nancy Edwards
3	Margaret Park	Nancy Edwards
4	Steve Johnson	Nancy Edwards
5	Michael Mitchell	Andrew Adams
6	Robert King	Michael Mitchell
7	Laura Callahan	Michael Mitchell


Employee Reporting Structure
Michael Mitchell
Nancy Edwards
Andrew Adams
ReportingManagerName
Andrew Adams
Nancy Edwards
Michael Mitchell
Jane Peacock
Margaret Park
Steve Johnson
Robert King
Laura Callahan
EmployeeName

```
Out[29]: ("SELECT \n    E.FirstName || ' ' || E.LastName AS EmployeeName,\n    R.FirstName || ' ' || R.LastName AS
         ReportingManagerName\nFROM \n    Employee AS E\nLEFT JOIN \n    Employee AS R ON E.ReportsTo = R.EmployeeI
         d;",
                 EmployeeName ReportingManagerName
          0      Andrew Adams                 None
          1     Nancy Edwards         Andrew Adams
          2      Jane Peacock        Nancy Edwards
          3     Margaret Park        Nancy Edwards
          4     Steve Johnson        Nancy Edwards
          5  Michael Mitchell         Andrew Adams
          6       Robert King     Michael Mitchell
          7    Laura Callahan     Michael Mitchell,
          Figure({
              'data': [{'alignmentgroup': 'True',
                        'hoverinfo': 'text',
                        'hovertemplate': 'EmployeeName=%{x}<br>ReportingManagerName=%{y}<extra></extra>',
                        'hovertext': [Employee: Andrew Adams<br>Reporting Manager: None,
                                      Employee: Nancy Edwards<br>Reporting Manager: Andrew
                                      Adams, Employee: Jane Peacock<br>Reporting Manager:
                                      Nancy Edwards, Employee: Margaret Park<br>Reporting
                                      Manager: Nancy Edwards, Employee: Steve
                                      Johnson<br>Reporting Manager: Nancy Edwards, Employee:
                                      Michael Mitchell<br>Reporting Manager: Andrew Adams,
                                      Employee: Robert King<br>Reporting Manager: Michael
                                      Mitchell, Employee: Laura Callahan<br>Reporting
                                      Manager: Michael Mitchell],
                        'legendgroup': '',
                        'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
                        'name': '',
                        'offsetgroup': '',
                        'orientation': 'v',
                        'showlegend': False,
                        'textposition': 'auto',
                        'type': 'bar',
                        'x': array(['Andrew Adams', 'Nancy Edwards', 'Jane Peacock', 'Margaret Park',
                                    'Steve Johnson', 'Michael Mitchell', 'Robert King', 'Laura Callahan'],
                                   dtype=object),
                        'xaxis': 'x',
                        'y': array([None, 'Andrew Adams', 'Nancy Edwards', 'Nancy Edwards', 'Nancy Edwards',
                                    'Andrew Adams', 'Michael Mitchell', 'Michael Mitchell'], dtype=object),
                        'yaxis': 'y'}],
              'layout': {'barmode': 'relative',
```

```
                    'legend': {'tracegroupgap': 0},
                    'template': '...',
                    'title': {'text': 'Employee Reporting Structure'},
                    'xaxis': {'anchor': 'y',
                              'categoryorder': 'total ascending',
                              'domain': [0.0, 1.0],
                              'title': {'text': 'EmployeeName'}},
                    'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'ReportingManagerName'}}}
      }))
```

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
```

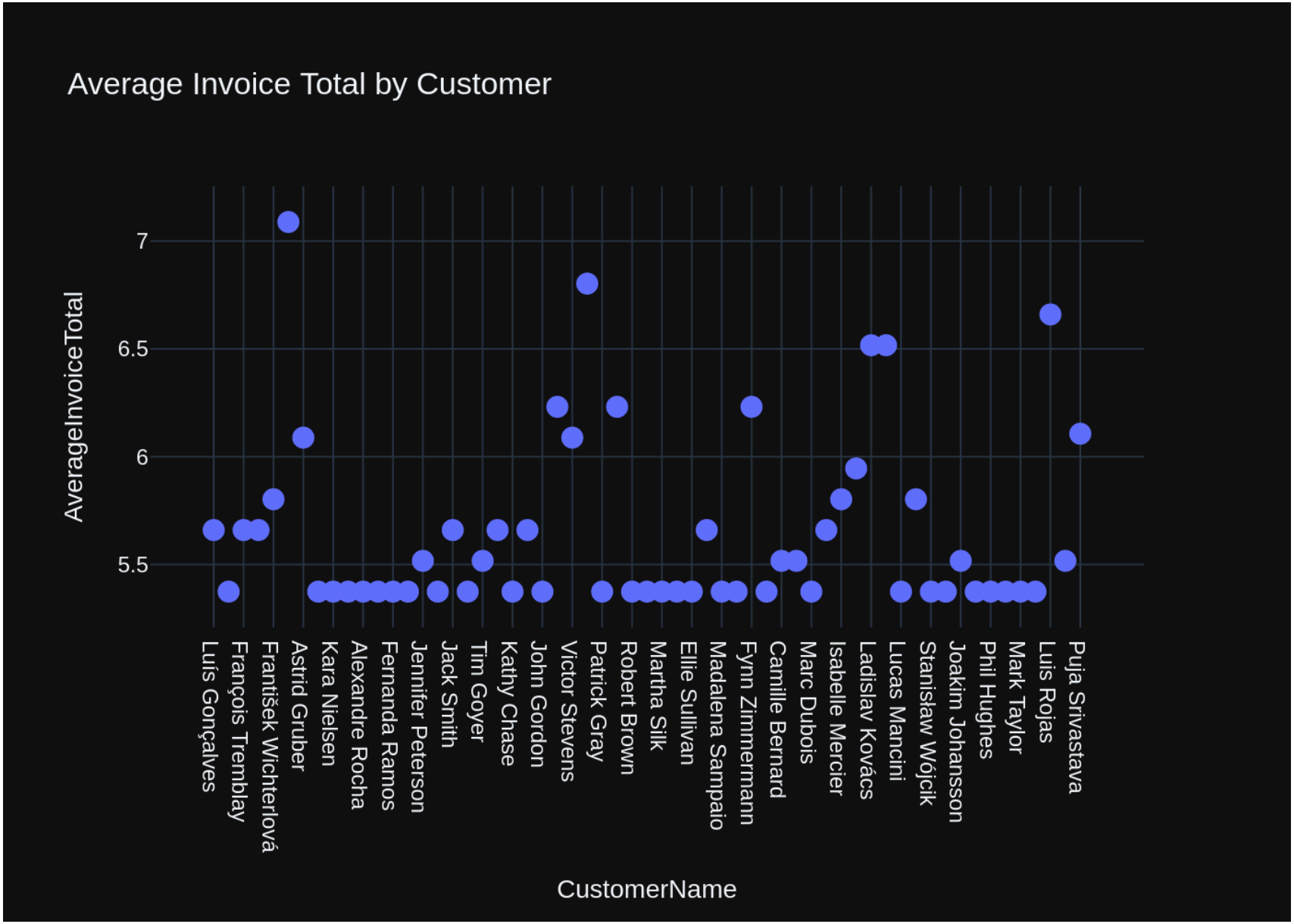
["You are a SQLite expert. Please help to generate a SQL query to answer the question. Your response should ONLY be based on the given context and follow the response guidelines and format instructions. \n===Tables \nCREATE INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)\n\nCREATE INDEX IFK_InvoiceLineInvoiceId ON InvoiceLine (InvoiceId)\n\nCREATE TABLE Invoice\n(\n   InvoiceId INTEGER NOT NULL,\n   CustomerId INTEGER NOT NULL,\n   InvoiceDate DATETIME NOT NULL,\n   BillingAddress NVARCHAR(70),\n   BillingCity NVARCHAR(40),\n   BillingState NVARCHAR(40),\n   BillingCountry NVARCHAR(40),\n   BillingPostalCode NVARCHAR(10),\n   Total NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Invoice PRIMARY KEY (InvoiceId),\n   FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE InvoiceLine\n(\n   InvoiceLineId INTEGER NOT NULL,\n   InvoiceId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   Quantity INTEGER NOT NULL,\n   CONSTRAINT PK_InvoiceLine PRIMARY KEY (InvoiceLineId),\n   FOREIGN KEY (InvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceLineTrackId ON InvoiceLine (TrackId)\n\nCREATE INDEX IFK_CustomerSupportRepId ON Customer (SupportRepId)\n\nCREATE TABLE Customer\n(\n   CustomerId INTEGER NOT NULL,\n   FirstName NVARCHAR(40) NOT NULL,\n   LastName NVARCHAR(20) NOT NULL,\n   Company NVARCHAR(80),\n   Address NVARCHAR(70),\n   City NVARCHAR(40),\n   State NVARCHAR(40),\n   Country NVARCHAR(40),\n   PostalCode NVARCHAR(10),\n   Phone NVARCHAR(24),\n   Fax NVARCHAR(24),\n   Email NVARCHAR(60) NOT NULL,\n   SupportRepId INTEGER,\n   CONSTRAINT PK_Customer PRIMARY KEY (CustomerId),\n   FOREIGN KEY (SupportRepId) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_EmployeeReportsTo ON Employee (ReportsTo)\n\nCREATE TABLE Track\n(\n   TrackId INTEGER NOT NULL,\n   Name NVARCHAR(200) NOT NULL,\n   AlbumId INTEGER,\n   MediaTypeId INTEGER NOT NULL,\n   GenreId INTEGER,\n   Composer NVARCHAR(220),\n   Milliseconds INTEGER NOT NULL,\n   Bytes INTEGER,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Track PRIMARY KEY (TrackId),\n   FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Employee\n(\n   EmployeeId INTEGER NOT NULL,\n   LastName NVARCHAR(20) NOT NULL,\n   FirstName NVARCHAR(20) NOT NULL,\n   Title NVARCHAR(30),\n   ReportsTo INTEGER,\n   BirthDate DATETIME,\n   HireDate DATETIME,\n   Address NVARCHAR(70),\n   City NVARCHAR(40),\n   State NVARCHAR(40),\n   Country NVARCHAR(40),\n   PostalCode NVARCHAR(10),\n   Phone NVARCHAR(24),\n   Fax NVARCHAR(24),\n   Email NVARCHAR(60),\n   CONSTRAINT PK_Employee PRIMARY KEY (EmployeeId),\n   FOREIGN KEY (ReportsTo) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n===Additional Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficient, please generate a valid SQL query without any explanations for the question. \n2. If the provided context is almost sufficient but requires knowledge of a specific string in a particular column, please generate an intermediate SQL query to find the distinct strings in that column. Prepend the query with a comment saying intermediate_sql \n3. If the provided context is insufficient, please explain why it can't be generated. \n4. Please use the most relevant table(s). \n5. If the question has been asked and answered before, please repeat the answer exactly as it was given before. \n", ' \n   Get the total number of invoices for each customer \n', "SELECT \n   C.FirstName || ' ' || C.LastName AS CustomerName,\n   COUNT(I.InvoiceId) AS TotalInvoices\nFROM \n   Customer AS C\nLEFT JOIN \n   Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n   C.CustomerId;", ' \n   Find all invoices since 2010 and the total amount invoiced:\n', "SELECT \n   I.In

```
voiceDate,\n    SUM(IL.UnitPrice * IL.Quantity) AS TotalAmount\nFROM \n    Invoice AS I\nJOIN \n    Invoice
Line AS IL ON I.InvoiceId = IL.InvoiceId\nWHERE \n    I.InvoiceDate >= '2010-01-01'\nGROUP BY \n    I.Invoi
ceDate\nORDER BY \n    I.InvoiceDate;", '  \n    Find the total number of invoices per country:\n', 'SELECT
\n    C.Country,\n    COUNT(I.InvoiceId) AS TotalInvoices\nFROM \n    Customer AS C\nLEFT JOIN \n    Invoic
e AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n    C.Country\nORDER BY \n    TotalInvoices DESC;', '  \n
List all invoices with a total exceeding $10:\n', 'SELECT \n    *\nFROM \n    Invoice\nWHERE \n    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    Customer;', 'what are the top 5 countries tha
t customers come from?', 'SELECT \n    C.Country,\n    COUNT(*) AS CustomerCount\nFROM \n    Customer AS C
\nGROUP BY \n    C.Country\nORDER BY \n    CustomerCount DESC\nLIMIT \n    5;', "  \n    List all employees
and their reporting manager's name (if any):\n", "SELECT \n    E.FirstName || ' ' || E.LastName AS Employee
Name,\n    R.FirstName || ' ' || R.LastName AS ReportingManagerName\nFROM \n    Employee AS E\nLEFT JOIN \n
Employee AS R ON E.ReportsTo = R.EmployeeId;", '  \n    Find all tracks with a name containing "What" (case
-insensitive)\n', "SELECT \n    T.Name AS TrackName\nFROM \n    Track AS T\nWHERE \n    UPPER(T.Name) LIKE
'%WHAT%';", '  \n    List all albums and their corresponding artist names  \n', 'SELECT \n    A.Title AS Al
bumTitle,\n    Ar.Name AS ArtistName\nFROM \n    Album AS A\nJOIN \n    Artist AS Ar ON A.ArtistId = Ar.Art
istId;', '  \n    Get the average invoice total for each customer:\n']
SELECT
    C.FirstName || ' ' || C.LastName AS CustomerName,
    AVG(I.Total) AS AverageInvoiceTotal
FROM
    Customer AS C
LEFT JOIN
    Invoice AS I ON C.CustomerId = I.CustomerId
GROUP BY
    C.CustomerId;
SELECT
    C.FirstName || ' ' || C.LastName AS CustomerName,
    AVG(I.Total) AS AverageInvoiceTotal
FROM
    Customer AS C
LEFT JOIN
    Invoice AS I ON C.CustomerId = I.CustomerId
GROUP BY
    C.CustomerId;
SELECT
    C.FirstName || ' ' || C.LastName AS CustomerName,
    AVG(I.Total) AS AverageInvoiceTotal
FROM
    Customer AS C
LEFT JOIN
    Invoice AS I ON C.CustomerId = I.CustomerId
```

```
GROUP BY
    C.CustomerId;
```

	CustomerName	AverageInvoiceTotal
0	Luís Gonçalves	5.660000
1	Leonie Köhler	5.374286
2	François Tremblay	5.660000
3	Bjørn Hansen	5.660000
4	František Wichterlová	5.802857
5	Helena Holý	7.088571
6	Astrid Gruber	6.088571
7	Daan Peeters	5.374286
8	Kara Nielsen	5.374286
9	Eduardo Martins	5.374286
10	Alexandre Rocha	5.374286
11	Roberto Almeida	5.374286
12	Fernanda Ramos	5.374286
13	Mark Philips	5.374286
14	Jennifer Peterson	5.517143
15	Frank Harris	5.374286
16	Jack Smith	5.660000
17	Michelle Brooks	5.374286
18	Tim Goyer	5.517143
19	Dan Miller	5.660000
20	Kathy Chase	5.374286
21	Heather Leacock	5.660000
22	John Gordon	5.374286
23	Frank Ralston	6.231429
24	Victor Stevens	6.088571
25	Richard Cunningham	6.802857
26	Patrick Gray	5.374286
27	Julia Barnett	6.231429
28	Robert Brown	5.374286
29	Edward Francis	5.374286
30	Martha Silk	5.374286
31	Aaron Mitchell	5.374286
32	Ellie Sullivan	5.374286
33	João Fernandes	5.660000
34	Madalena Sampaio	5.374286
35	Hannah Schneider	5.374286
36	Fynn Zimmermann	6.231429
37	Niklas Schröder	5.374286
38	Camille Bernard	5.517143

39	Dominique Lefebvre	5.517143
40	Marc Dubois	5.374286
41	Wyatt Girard	5.660000
42	Isabelle Mercier	5.802857
43	Terhi Hämäläinen	5.945714
44	Ladislav Kovács	6.517143
45	Hugh O'Reilly	6.517143
46	Lucas Mancini	5.374286
47	Johannes Van der Berg	5.802857
48	Stanisław Wójcik	5.374286
49	Enrique Muñoz	5.374286
50	Joakim Johansson	5.517143
51	Emma Jones	5.374286
52	Phil Hughes	5.374286
53	Steve Murray	5.374286
54	Mark Taylor	5.374286
55	Diego Gutiérrez	5.374286
56	Luis Rojas	6.660000
57	Manoj Pareek	5.517143
58	Puja Srivastava	6.106667


Average Invoice Total by Customer
AverageInvoiceTotal
7
6.5
6
5.5
CustomerName
Luís Gonçalves
François Tremblay
František Wichterlová
Astrid Gruber
Kara Nielsen
Alexandre Rocha
Fernanda Ramos
Jennifer Peterson
Jack Smith
Tim Goyer
Kathy Chase
John Gordon
Victor Stevens
Patrick Gray
Robert Brown
Martha Silk
Ellie Sullivan
Madalena Sampaio
Fynn Zimmermann
Camille Bernard
Marc Dubois
Isabelle Mercier
Ladislav Kovács
Lucas Mancini
Stanisław Wójcik
Joakim Johansson
Phil Hughes
Mark Taylor
Luis Rojas
Puja Srivastava

Out[30]: ("SELECT \n   C.FirstName || ' ' || C.LastName AS CustomerName,\n   AVG(I.Total) AS AverageInvoiceTotal \nFROM \n   Customer AS C\nLEFT JOIN \n   Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n   C.CustomerId;",

	CustomerName	AverageInvoiceTotal
0	Luís Gonçalves	5.660000
1	Leonie Köhler	5.374286
2	François Tremblay	5.660000
3	Bjørn Hansen	5.660000
4	František Wichterlová	5.802857
5	Helena Holý	7.088571
6	Astrid Gruber	6.088571
7	Daan Peeters	5.374286
8	Kara Nielsen	5.374286
9	Eduardo Martins	5.374286
10	Alexandre Rocha	5.374286
11	Roberto Almeida	5.374286
12	Fernanda Ramos	5.374286
13	Mark Philips	5.374286
14	Jennifer Peterson	5.517143
15	Frank Harris	5.374286
16	Jack Smith	5.660000
17	Michelle Brooks	5.374286
18	Tim Goyer	5.517143
19	Dan Miller	5.660000
20	Kathy Chase	5.374286
21	Heather Leacock	5.660000
22	John Gordon	5.374286
23	Frank Ralston	6.231429
24	Victor Stevens	6.088571
25	Richard Cunningham	6.802857
26	Patrick Gray	5.374286
27	Julia Barnett	6.231429
28	Robert Brown	5.374286
29	Edward Francis	5.374286
30	Martha Silk	5.374286
31	Aaron Mitchell	5.374286
32	Ellie Sullivan	5.374286
33	João Fernandes	5.660000
34	Madalena Sampaio	5.374286
35	Hannah Schneider	5.374286
36	Fynn Zimmermann	6.231429
37	Niklas Schröder	5.374286

```
38        Camille Bernard            5.517143
39     Dominique Lefebvre            5.517143
40            Marc Dubois            5.374286
41           Wyatt Girard            5.660000
42       Isabelle Mercier            5.802857
43       Terhi Hämäläinen            5.945714
44        Ladislav Kovács            6.517143
45          Hugh O'Reilly            6.517143
46          Lucas Mancini            5.374286
47  Johannes Van der Berg            5.802857
48       Stanisław Wójcik            5.374286
49          Enrique Muñoz            5.374286
50       Joakim Johansson            5.517143
51             Emma Jones            5.374286
52            Phil Hughes            5.374286
53           Steve Murray            5.374286
54            Mark Taylor            5.374286
55        Diego Gutiérrez            5.374286
56             Luis Rojas            6.660000
57           Manoj Pareek            5.517143
58        Puja Srivastava            6.106667,
Figure({
    'data': [{'hovertemplate': 'CustomerName=%{x}<br>AverageInvoiceTotal=%{y}<extra></extra>',
              'legendgroup': '',
              'marker': {'color': '#636efa', 'size': 12, 'symbol': 'circle'},
              'mode': 'markers',
              'name': '',
              'orientation': 'v',
              'showlegend': False,
              'type': 'scatter',
              'x': array(['Luís Gonçalves', 'Leonie Köhler', 'François Tremblay', 'Bjørn Hansen',
                          'František Wichterlová', 'Helena Holý', 'Astrid Gruber', 'Daan Peeters',
                          'Kara Nielsen', 'Eduardo Martins', 'Alexandre Rocha', 'Roberto Almeida',
                          'Fernanda Ramos', 'Mark Philips', 'Jennifer Peterson', 'Frank Harris',
                          'Jack Smith', 'Michelle Brooks', 'Tim Goyer', 'Dan Miller',
                          'Kathy Chase', 'Heather Leacock', 'John Gordon', 'Frank Ralston',
                          'Victor Stevens', 'Richard Cunningham', 'Patrick Gray', 'Julia Barnett',
                          'Robert Brown', 'Edward Francis', 'Martha Silk', 'Aaron Mitchell',
                          'Ellie Sullivan', 'João Fernandes', 'Madalena Sampaio',
                          'Hannah Schneider', 'Fynn Zimmermann', 'Niklas Schröder',
                          'Camille Bernard', 'Dominique Lefebvre', 'Marc Dubois', 'Wyatt Girard',
                          'Isabelle Mercier', 'Terhi Hämäläinen', 'Ladislav Kovács',
```

```
                                "Hugh O'Reilly", 'Lucas Mancini', 'Johannes Van der Berg',
                                'Stanisław Wójcik', 'Enrique Muñoz', 'Joakim Johansson', 'Emma Jones',
                                'Phil Hughes', 'Steve Murray', 'Mark Taylor', 'Diego Gutiérrez',
                                'Luis Rojas', 'Manoj Pareek', 'Puja Srivastava'], dtype=object),
                   'xaxis': 'x',
                   'y': array([5.66      , 5.37428571, 5.66      , 5.66      , 5.80285714, 7.08857143,
                               6.08857143, 5.37428571, 5.37428571, 5.37428571, 5.37428571, 5.37428571,
                               5.37428571, 5.37428571, 5.51714286, 5.37428571, 5.66      , 5.37428571,
                               5.51714286, 5.66      , 5.37428571, 5.66      , 5.37428571, 6.23142857,
                               6.08857143, 6.80285714, 5.37428571, 6.23142857, 5.37428571, 5.37428571,
                               5.37428571, 5.37428571, 5.37428571, 5.66      , 5.37428571, 5.37428571,
                               6.23142857, 5.37428571, 5.51714286, 5.51714286, 5.37428571, 5.66      ,
                               5.80285714, 5.94571429, 6.51714286, 6.51714286, 5.37428571, 5.80285714,
                               5.37428571, 5.37428571, 5.51714286, 5.37428571, 5.37428571, 5.37428571,
                               5.37428571, 5.37428571, 6.66      , 5.51714286, 6.10666667]),
                   'yaxis': 'y'}],
        'layout': {'legend': {'tracegroupgap': 0},
                   'template': '...',
                   'title': {'text': 'Average Invoice Total by Customer'},
                   'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'CustomerName'}},
                   'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'AverageInvoiceTotal'}}}
    }))
```

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
```

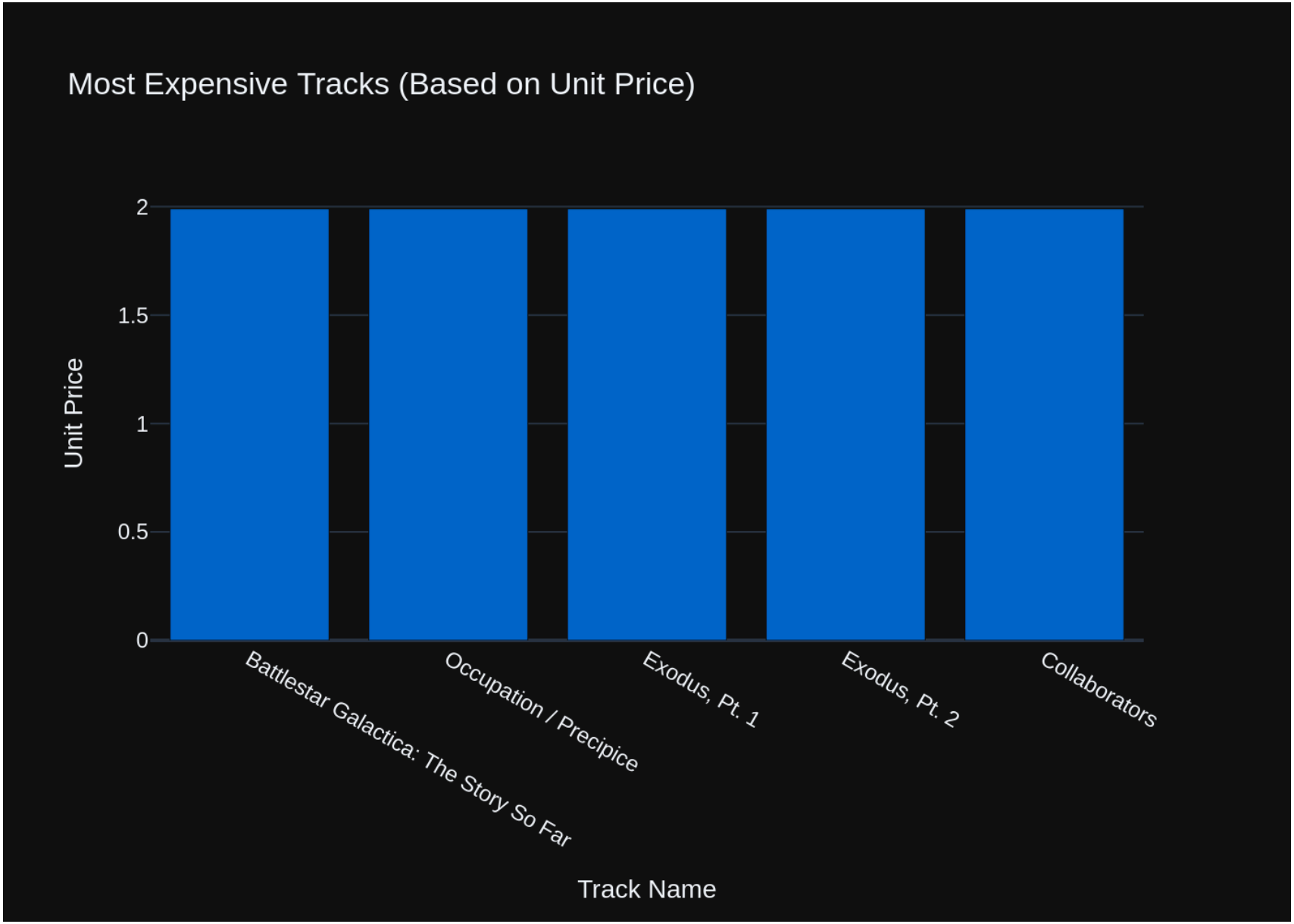
["You are a SQLite expert. Please help to generate a SQL query to answer the question. Your response should ONLY be based on the given context and follow the response guidelines and format instructions. \n===Tables \nCREATE TABLE Track\n(\n   TrackId INTEGER NOT NULL,\n   Name NVARCHAR(200) NOT NULL,\n   AlbumId INTEGER,\n   MediaTypeId INTEGER NOT NULL,\n   GenreId INTEGER,\n   Composer NVARCHAR(220),\n   Milliseconds INTEGER NOT NULL,\n   Bytes INTEGER,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Track PRIMARY KEY (TrackId),\n   FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_TrackAlbumId ON Track (AlbumId)\n\nCREATE INDEX IFK_TrackGenreId ON Track (GenreId)\n\nCREATE INDEX IFK_PlaylistTrackTrackId ON PlaylistTrack (TrackId)\n\nCREATE INDEX IFK_InvoiceLineTrackId ON InvoiceLine (TrackId)\n\nCREATE INDEX IFK_TrackMediaTypeId ON Track (MediaTypeId)\n\nCREATE TABLE InvoiceLine\n(\n   InvoiceLineId INTEGER NOT NULL,\n   InvoiceId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   Quantity INTEGER NOT NULL,\n   CONSTRAINT PK_InvoiceLine PRIMARY KEY (InvoiceLineId),\n   FOREIGN KEY (InvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE PlaylistTrack\n(\n   PlaylistId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   CONSTRAINT PK_PlaylistTrack PRIMARY KEY (PlaylistId, TrackId),\n   FOREIGN KEY (PlaylistId) REFERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_AlbumArtistId ON Album (ArtistId)\n\nCREATE TABLE Album\n(\n   AlbumId INTEGER NOT NULL,\n   Title NVARCHAR(160) NOT NULL,\n   ArtistId INTEGER NOT NULL,\n   CONSTRAINT PK_Album PRIMARY KEY (AlbumId),\n   FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n===Additional Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficient, please generate a valid SQL query without any explanations for the question. \n2. If the provided context is almost sufficient but requires knowledge of a specific string in a particular column, please generate an intermediate SQL query to find the distinct strings in that column. Prepend the query with a comment saying intermediate_sql \n3. If the provided context is insufficient, please explain why it can't be generated. \n4. Please use the most relevant table(s). \n5. If the question has been asked and answered before, please repeat the answer exactly as it was given before. \n", ' \n   Find all tracks with a name containing "What" (case-insensitive)\n', "SELECT \n   T.Name AS TrackName\nFROM \n   Track AS T\nWHERE \n   UPPER(T.Name) LIKE '%WHAT%';", ' \n   List all invoices with a total exceeding $10:\n', 'SELECT \n   *\nFROM \n   Invoice\nWHERE \n   Total > 10;', ' \n   Find all invoices since 2010 and the total amount invoiced:\n', "SELECT \n   I.InvoiceDate,\n   SUM(IL.UnitPrice * IL.Quantity) AS TotalAmount\nFROM \n   Invoice AS I\nJOIN \n   InvoiceLine AS IL ON I.InvoiceId = IL.InvoiceId\nWHERE \n   I.InvoiceDate >= '2010-01-01'\nGROUP BY \n   I.InvoiceDate\nORDER BY \n   I.InvoiceDate;", ' \n   List all albums and their corresponding artist names \n', 'SELECT \n   A.Title AS AlbumTitle,\n   Ar.Name AS ArtistName\nFROM \n   Album AS A\nJOIN \n   Artist AS Ar ON A.ArtistId = Ar.ArtistId;', ' \n   Get the average invoice total for each customer:\n', "SELECT \n   C.FirstName || ' ' || C.LastName AS CustomerName,\n   AVG(I.Total) AS AverageInvoiceTotal\nFROM \n   Customer AS C\nLEFT JOIN \n   Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n   C.CustomerId;", 'what are the top 5 countries that customers come from?', 'SELECT \n   C.Country,\n   COUNT(*) AS CustomerCount\nFROM \n   Customer AS C\nGROUP BY \n   C.Country\nORDER BY \n   CustomerCount DESC\nLIMIT \n   5;', ' \n   Find the total nu

```
mber of invoices per country:\n', 'SELECT \n    C.Country,\n    COUNT(I.InvoiceId) AS TotalInvoices\nFROM
\n    Customer AS C\nLEFT JOIN \n    Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n    C.Country
\nORDER BY \n    TotalInvoices DESC;', '  \n    Get the total number of invoices for each customer\n', "SEL
ECT \n    C.FirstName || ' ' || C.LastName AS CustomerName,\n    COUNT(I.InvoiceId) AS TotalInvoices\nFROM
\n    Customer AS C\nLEFT JOIN \n    Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n    C.Customer
Id;", 'Show me a list of tables in the SQLite database', 'SELECT \n    T.Name\nFROM\n    Playlist AS T;',
'How many customers are there', 'SELECT \n    COUNT(*)\nFROM \n    Customer;', '  \n    Find the top 5 most
expensive tracks (based on unit price):\n']
```

```sql
SELECT
    T.Name AS TrackName,
    T.UnitPrice AS UnitPrice
FROM
    Track AS T
ORDER BY
    T.UnitPrice DESC
LIMIT
    5;
```

```
SELECT
    T.Name AS TrackName,
    T.UnitPrice AS UnitPrice
FROM
    Track AS T
ORDER BY
    T.UnitPrice DESC
LIMIT
    5;
SELECT
    T.Name AS TrackName,
    T.UnitPrice AS UnitPrice
FROM
    Track AS T
ORDER BY
    T.UnitPrice DESC
LIMIT
    5;
                               TrackName  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    T.Name AS TrackName,\n    T.UnitPrice AS UnitPrice\nFROM \n    Track AS T\nORDER BY \n
T.UnitPrice DESC\nLIMIT \n    5;',
                                TrackName  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': [{'alignmentgroup': 'True',
               'hovertemplate': 'TrackName=%{x}<br>UnitPrice=%{y}<extra></extra>',
               'legendgroup': '',
               'marker': {'color': 'rgb(0,102,204)', 'pattern': {'shape': ''}},
               'name': '',
               'offsetgroup': '',
               'orientation': 'v',
               'showlegend': False,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Battlestar Galactica: The Story So Far', 'Occupation / Precipice',
                           'Exodus, Pt. 1', 'Exodus, Pt. 2', 'Collaborators'], dtype=object),
               'xaxis': 'x',
               'y': array([1.99, 1.99, 1.99, 1.99, 1.99]),
               'yaxis': 'y'}],
     'layout': {'barmode': 'relative',
                'legend': {'tracegroupgap': 0},
                'template': '...',
                'title': {'text': 'Most Expensive Tracks (Based on Unit Price)'},
                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'Track Name'}},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'Unit Price'}}}
 }))
```

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

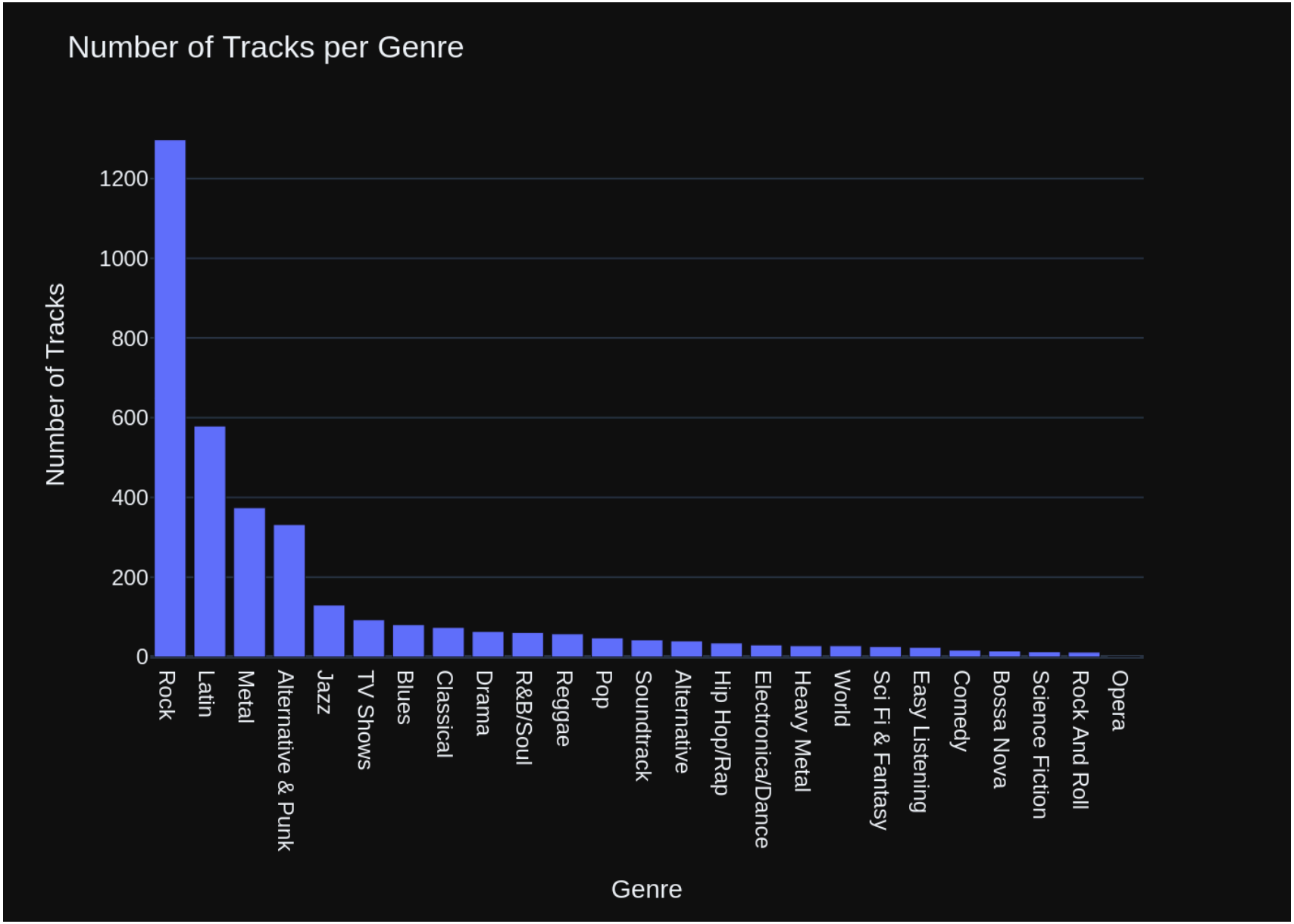
vn.ask(question=question)
```

```
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 INTEGER,\n   MediaTypeId INTEGER NOT NULL,\n   GenreId INTEGER,\n   Composer NVARCHAR(220),\n   Milliseconds INTEGER NOT NULL,\n   Bytes INTEGER,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Track PRIMARY KEY (TrackId),\n   FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_TrackGenreId ON Track (GenreId)\n\nCREATE TABLE Genre\n(\n   GenreId INTEGER NOT NULL,\n   Name NVARCHAR(120),\n   CONSTRAINT PK_Genre PRIMARY KEY (GenreId)\n)\n\nCREATE INDEX IFK_PlaylistTrackTrackId ON PlaylistTrack (TrackId)\n\nCREATE INDEX IFK_TrackAlbumId ON Track (AlbumId)\n\nCREATE INDEX IFK_TrackMediaTypeId ON Track (MediaTypeId)\n\nCREATE TABLE Album\n(\n   AlbumId INTEGER NOT NULL,\n   Title NVARCHAR(160) NOT NULL,\n   ArtistId INTEGER NOT NULL,\n   CONSTRAINT PK_Album PRIMARY KEY (AlbumId),\n   FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE PlaylistTrack\n(\n   PlaylistId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   CONSTRAINT PK_PlaylistTrack PRIMARY KEY (PlaylistId, TrackId),\n   FOREIGN KEY (PlaylistId) REFERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_AlbumArtistId ON Album (ArtistId)\n\nCREATE TABLE Playlist\n(\n   PlaylistId INTEGER NOT NULL,\n   Name NVARCHAR(120),\n   CONSTRAINT PK_Playlist PRIMARY KEY (PlaylistId)\n)\n\n\n\n===Additional Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficient, please generate a valid SQL query without any explanations for the question. \n2. If the provided context is almost sufficient but requires knowledge of a specific string in a particular column, please generate an intermediate SQL query to find the distinct strings in that column. Prepend the query with a comment saying intermediate_sql \n3. If the provided context is insufficient, please explain why it can't be generated. \n4. Please use the most relevant table(s). \n5. If the question has been asked and answered before, please repeat the answer exactly as it was given before. \n", ' \n   Find the top 5 most expensive tracks (based on unit price):\n', 'SELECT \n   T.Name AS TrackName,\n   T.UnitPrice AS UnitPrice\nFROM \n   Track AS T\nORDER BY \n   T.UnitPrice DESC\nLIMIT \n   5;', ' \n   List all albums and their corresponding artist names \n', 'SELECT \n   A.Title AS AlbumTitle,\n   Ar.Name AS ArtistName\nFROM \n   Album AS A\nJOIN \n   Artist AS Ar ON A.ArtistId = Ar.ArtistId;', ' \n   Find all tracks with a name containing "What" (case-insensitive)\n', "SELECT \n   T.Name AS TrackName\nFROM \n   Track AS T\nWHERE \n   UPPER(T.Name) LIKE '%WHAT%';", 'Show me a list of tables in the SQLite database', 'SELECT \n   T.Name\nFROM\n   Playlist AS T;', 'what are the top 5 countries that customers come from?', 'SELECT \n   C.Country,\n   COUNT(*) AS CustomerCount\nFROM \n   Customer AS C\nGROUP BY \n   C.Country\nORDER BY \n   CustomerCount DESC\nLIMIT \n   5;', ' \n   Find all invoices since 2010 and the total amount invoiced:\n', "SELECT \n   I.InvoiceDate,\n   SUM(IL.UnitPrice * IL.Quantity) AS TotalAmount\nFROM \n   Invoice AS I\nJOIN \n   InvoiceLine AS IL ON I.InvoiceId = IL.InvoiceId\nWHERE \n   I.InvoiceDate >= '2010-01-01'\nGROUP BY \n   I.InvoiceDate\nORDER BY \n   I.InvoiceDate;", ' \n   Find the total number of invoices per country:\n', 'SELECT \n   C.Country,\n   COUNT(I.InvoiceId) AS TotalInvoices\nFROM \n   Customer AS C\nLEFT JOIN \n   Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n   C.Country\nORDER BY \n   TotalInvoices DESC;', 'How many records are in table called customer', 'SELECT \n   COUNT(*)\nFROM \n   Customer;', ' \n   Get the total

```
number of invoices for each customer\n', "SELECT \n    C.FirstName || ' ' || C.LastName AS CustomerName,\n
COUNT(I.InvoiceId) AS TotalInvoices\nFROM \n    Customer AS C\nLEFT JOIN \n    Invoice AS I ON C.CustomerId
= I.CustomerId\nGROUP BY \n    C.CustomerId;", '  \n    List all invoices with a total exceeding $10:\n',
'SELECT \n    *\nFROM \n    Invoice\nWHERE \n    Total > 10;', '  \n    List all genres and the number of t
racks in each genre:\n']
SELECT
    G.Name AS GenreName,
    COUNT(T.TrackId) AS TrackCount
FROM
    Genre AS G
LEFT JOIN
    Track AS T ON G.GenreId = T.GenreId
GROUP BY
    G.GenreId
ORDER BY
    TrackCount DESC;
SELECT
    G.Name AS GenreName,
    COUNT(T.TrackId) AS TrackCount
FROM
    Genre AS G
LEFT JOIN
    Track AS T ON G.GenreId = T.GenreId
GROUP BY
    G.GenreId
ORDER BY
    TrackCount DESC;
SELECT
    G.Name AS GenreName,
    COUNT(T.TrackId) AS TrackCount
FROM
    Genre AS G
LEFT JOIN
    Track AS T ON G.GenreId = T.GenreId
GROUP BY
    G.GenreId
ORDER BY
    TrackCount DESC;
             GenreName  TrackCount
0                 Rock        1297
1                Latin         579
2                Metal         374
```

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


Number of Tracks per Genre
Number of Tracks
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
Genre

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

```
                            'Heavy Metal', 'World', 'Sci Fi & Fantasy', 'Easy Listening', 'Comedy',
                            'Bossa Nova', 'Science Fiction', 'Rock And Roll', 'Opera'], dtype=object),
                'xaxis': 'x',
                'y': array([1297,  579,  374,  332,  130,   93,   81,   74,   64,   61,   58,   48,
                             43,   40,   35,   30,   28,   28,   26,   24,   17,   15,   13,   12,
                              1]),
                'yaxis': 'y'}],
     'layout': {'barmode': 'relative',
                'legend': {'tracegroupgap': 0},
                'margin': {'t': 60},
                'template': '...',
                'title': {'text': 'Number of Tracks per Genre'},
                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'Genre'}},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'Number of Tracks'}}}
 }))
```

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

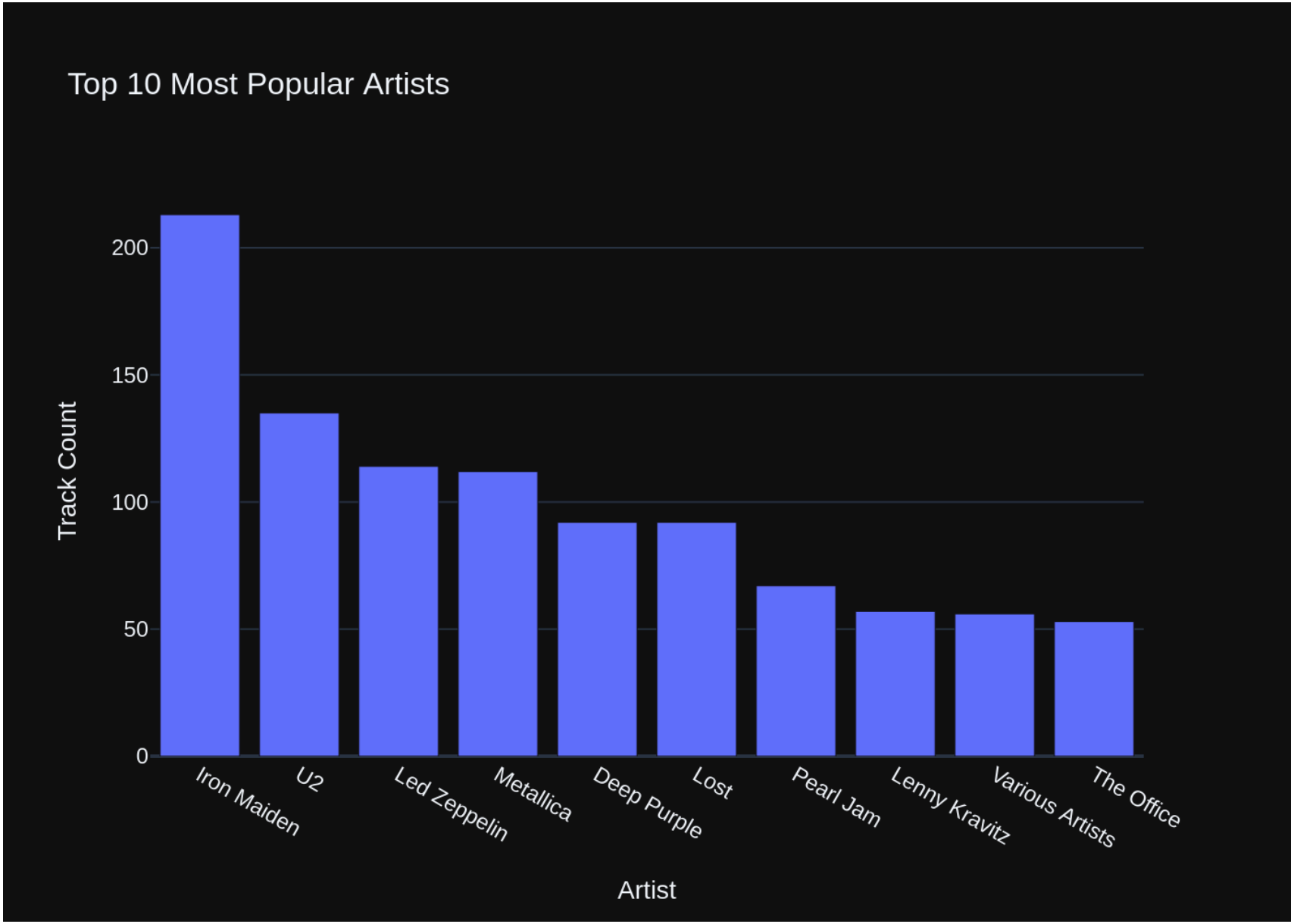
vn.ask(question=question)
```

```
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   G.Name AS GenreName,\n   COUNT(T.TrackId) AS TrackCount\nFROM \n   Genre AS G\nLE FT JOIN \n   Track AS T ON G.GenreId = T.GenreId\nGROUP BY \n   G.GenreId\nORDER BY \n   TrackCount DES C;', ' \n   Find all tracks with a name containing "What" (case-insensitive)\n', "SELECT \n   T.Name AS TrackName\nFROM \n   Track AS T\nWHERE \n   UPPER(T.Name) LIKE '%WHAT%';", ' \n   List all albums and t heir corresponding artist names \n', 'SELECT \n   A.Title AS AlbumTitle,\n   Ar.Name AS ArtistName\nFROM \n   Album AS A\nJOIN \n   Artist AS Ar ON A.ArtistId = Ar.ArtistId;', ' \n   Find the top 5 most expen sive tracks (based on unit price):\n', 'SELECT \n   T.Name AS TrackName,\n   T.UnitPrice AS UnitPrice\nFR OM \n   Track AS T\nORDER BY \n   T.UnitPrice DESC\nLIMIT \n   5;', 'Show me a list of tables in the SQL ite database', 'SELECT \n   T.Name\nFROM\n   Playlist AS T;', ' \n   Find all invoices since 2010 and t he total amount invoiced:\n', "SELECT \n   I.InvoiceDate,\n   SUM(IL.UnitPrice * IL.Quantity) AS TotalAmo unt\nFROM \n   Invoice AS I\nJOIN \n   InvoiceLine AS IL ON I.InvoiceId = IL.InvoiceId\nWHERE \n   I.Inv oiceDate >= '2010-01-01'\nGROUP BY \n   I.InvoiceDate\nORDER BY \n   I.InvoiceDate;", ' \n   List all i nvoices with a total exceeding $10:\n', 'SELECT \n   *\nFROM \n   Invoice\nWHERE \n   Total > 10;', 'wha t are the top 5 countries that customers come from?', 'SELECT \n   C.Country,\n   COUNT(*) AS CustomerCou nt\nFROM \n   Customer AS C\nGROUP BY \n   C.Country\nORDER BY \n   CustomerCount DESC\nLIMIT \n   5;',

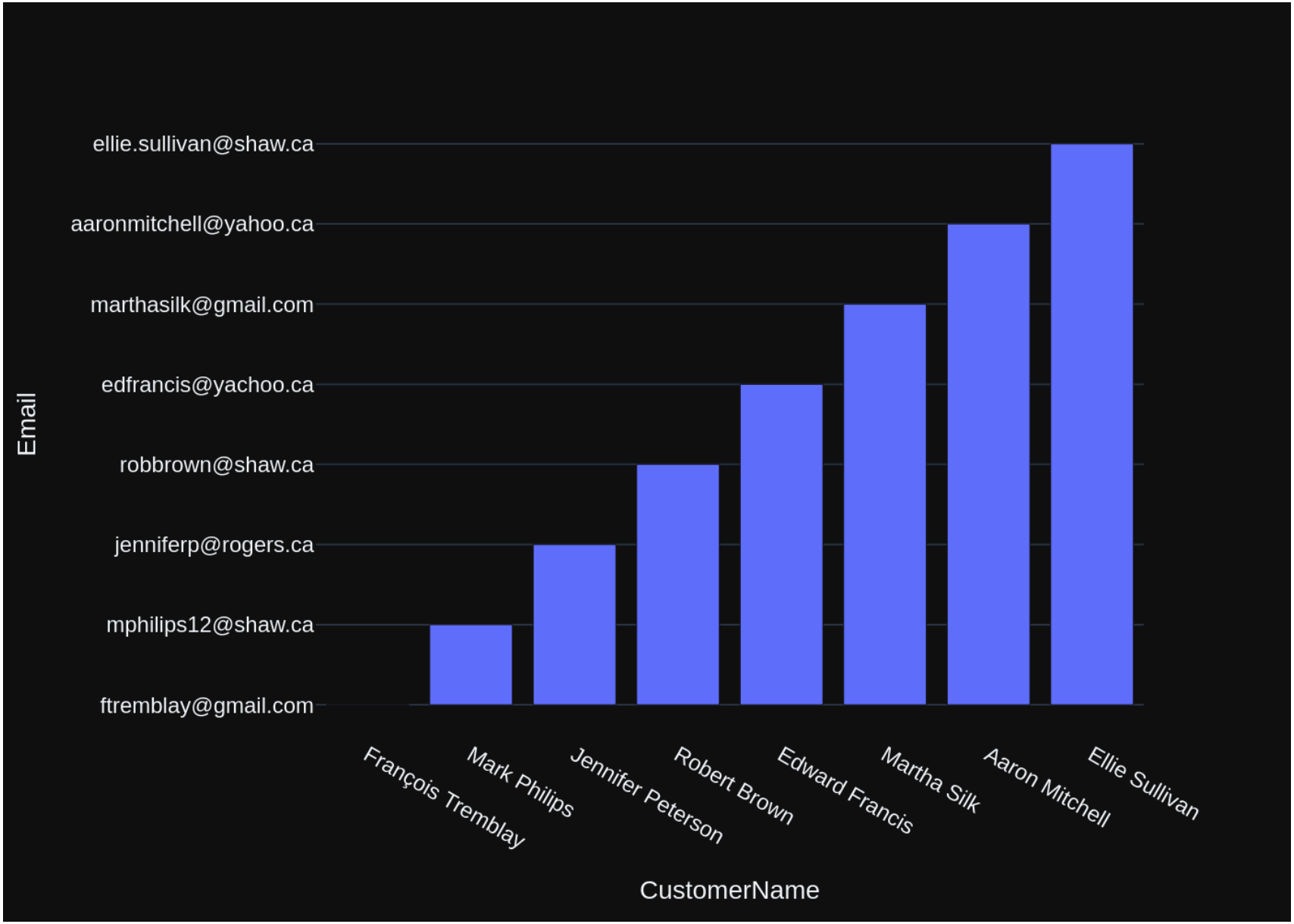
```
"  \n    List all employees and their reporting manager's name (if any):\n", "SELECT \n    E.FirstName || '
' || E.LastName AS EmployeeName,\n    R.FirstName || ' ' || R.LastName AS ReportingManagerName\nFROM \n
Employee AS E\nLEFT JOIN \n    Employee AS R ON E.ReportsTo = R.EmployeeId;", '  \n    Get the average invo
ice total for each customer:\n', "SELECT \n    C.FirstName || ' ' || C.LastName AS CustomerName,\n    AVG
(I.Total) AS AverageInvoiceTotal\nFROM \n    Customer AS C\nLEFT JOIN \n    Invoice AS I ON C.CustomerId =
I.CustomerId\nGROUP BY \n    C.CustomerId;", '  \n    Get all genres that do not have any tracks associated
with them:\n']
```sql
SELECT
    G.Name AS GenreName
FROM
    Genre AS G
LEFT JOIN
    Track AS T ON G.GenreId = T.GenreId
WHERE
    T.TrackId IS NULL;
```
SELECT
    G.Name AS GenreName
FROM
    Genre AS G
LEFT JOIN
    Track AS T ON G.GenreId = T.GenreId
WHERE
    T.TrackId IS NULL;
SELECT
    G.Name AS GenreName
FROM
    Genre AS G
LEFT JOIN
    Track AS T ON G.GenreId = T.GenreId
WHERE
    T.TrackId IS NULL;
Empty DataFrame
Columns: [GenreName]
Index: []
```

Out[33]:
```
('SELECT \n    G.Name AS GenreName\nFROM \n    Genre AS G\nLEFT JOIN \n    Track AS T ON G.GenreId = T.Gen
reId\nWHERE \n    T.TrackId IS NULL;',
 Empty DataFrame
 Columns: [GenreName]
 Index: [],
 Figure({
     'data': [{'domain': {'x': [0.0, 1.0], 'y': [0.0, 1.0]},
               'hovertemplate': 'GenreName=%{label}<extra></extra>',
               'labels': array([], dtype=object),
               'legendgroup': '',
               'name': '',
               'showlegend': True,
               'type': 'pie'}],
     'layout': {'legend': {'tracegroupgap': 0}, 'margin': {'t': 60}, '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   LastName NVARCHAR(20) NOT NULL,\n   Company NVARCHAR(80),\n   Address NVARCHAR(70),\n   City NVARCHAR(40),\n   State NVARCHAR(40),\n   Country NVARCHAR(40),\n   PostalCode NVARCHAR(10),\n   Phone NVARCHAR(24),\n   Fax NVARCHAR(24),\n   Email NVARCHAR(60) NOT NULL,\n   SupportRepId INTEGER,\n   CONSTRAINT PK_Customer PRIMARY KEY (CustomerId),\n   FOREIGN KEY (SupportRepId) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Invoice\n(\n   InvoiceId INTEGER NOT NULL,\n   CustomerId INTEGER NOT NULL,\n   InvoiceDate DATETIME NOT NULL,\n   BillingAddress NVARCHAR(70),\n   BillingCity NVARCHAR(40),\n   BillingState NVARCHAR(40),\n   BillingCountry NVARCHAR(40),\n   BillingPostalCode NVARCHAR(10),\n   Total NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Invoice PRIMARY KEY (InvoiceId),\n   FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE InvoiceLine\n(\n   InvoiceLineId INTEGER NOT NULL,\n   InvoiceId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   Quantity INTEGER NOT NULL,\n   CONSTRAINT PK_InvoiceLine PRIMARY KEY (InvoiceLineId),\n   FOREIGN KEY (InvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Employee\n(\n   EmployeeId INTEGER NOT NULL,\n   LastName NVARCHAR(20) NOT NULL,\n   FirstName NVARCHAR(20) NOT NULL,\n   Title NVARCHAR(30),\n   ReportsTo INTEGER,\n   BirthDate DATETIME,\n   HireDate DATETIME,\n   Address NVARCHAR(70),\n   City NVARCHAR(40),\n   State NVARCHAR(40),\n   Country NVARCHAR(40),\n   PostalCode NVARCHAR(10),\n   Phone NVARCHAR(24),\n   Fax NVARCHAR(24),\n   Email NVARCHAR(60),\n   CONSTRAINT PK_Employee PRIMARY KEY (EmployeeId),\n   FOREIGN KEY (ReportsTo) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_CustomerSupportRepId ON Customer (SupportRepId)\n\nCREATE TABLE PlaylistTrack\n(\n   PlaylistId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   CONSTRAINT PK_PlaylistTrack PRIMARY KEY (PlaylistId, TrackId),\n   FOREIGN KEY (PlaylistId) REFERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Album\n(\n   AlbumId INTEGER NOT NULL,\n   Title NVARCHAR(160) NOT NULL,\n   ArtistId INTEGER NOT NULL,\n   CONSTRAINT PK_Album PRIMARY KEY (AlbumId),\n   FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)\n\nCREATE TABLE Track\n(\n   TrackId INTEGER NOT NULL,\n   Name NVARCHAR(200) NOT NULL,\n   AlbumId INTEGER,\n   MediaTypeId INTEGER NOT NULL,\n   GenreId INTEGER,\n   Composer NVARCHAR(220),\n   Milliseconds INTEGER NOT NULL,\n   Bytes INTEGER,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Track PRIMARY KEY (TrackId),\n   FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Playlist\n(\n   PlaylistId INTEGER NOT NULL,\n   Name NVARCHAR(120),\n   CONSTRAINT PK_Playlist PRIMARY KEY (PlaylistId)\n)\n\n\n===Additional Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficient, please generate a valid SQL query without any explanations for the question. \n2. If the provided context is almost sufficient but requires knowledge of a specific string in a particular column, please generate an intermediate SQL query to find the distinct strings in that column. Prepend the query with a comment saying 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    C.FirstName || '
' || C.LastName AS CustomerName,\n    COUNT(I.InvoiceId) AS TotalInvoices\nFROM \n    Customer AS C\nLEFT J
OIN \n    Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n    C.CustomerId;", 'How many customers a
re there', 'SELECT \n    COUNT(*)\nFROM \n    Customer;', 'what are the top 5 countries that customers come
from?', 'SELECT \n    C.Country,\n    COUNT(*) AS CustomerCount\nFROM \n    Customer AS C\nGROUP BY \n
C.Country\nORDER BY \n    CustomerCount DESC\nLIMIT \n    5;', 'How many records are in table called custom
er', 'SELECT \n    COUNT(*)\nFROM \n    Customer;', '  \n    Find the total number of invoices per countr
y:\n', 'SELECT \n    C.Country,\n    COUNT(I.InvoiceId) AS TotalInvoices\nFROM \n    Customer AS C\nLEFT JO
IN \n    Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n    C.Country\nORDER BY \n    TotalInvoice
s DESC;', '  \n    Get the average invoice total for each customer:\n', "SELECT \n    C.FirstName || ' ' ||
C.LastName AS CustomerName,\n    AVG(I.Total) AS AverageInvoiceTotal\nFROM \n    Customer AS C\nLEFT JOIN
\n    Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n    C.CustomerId;", '  \n    List all invoice
s with a total exceeding $10:\n', 'SELECT \n    *\nFROM \n    Invoice\nWHERE \n    Total > 10;', '  \n    F
ind all invoices since 2010 and the total amount invoiced:\n', "SELECT \n    I.InvoiceDate,\n    SUM(IL.Uni
tPrice * IL.Quantity) AS TotalAmount\nFROM \n    Invoice AS I\nJOIN \n    InvoiceLine AS IL ON I.InvoiceId
= IL.InvoiceId\nWHERE \n    I.InvoiceDate >= '2010-01-01'\nGROUP BY \n    I.InvoiceDate\nORDER BY \n    I.I
nvoiceDate;", "  \n    List all employees and their reporting manager's name (if any):\n", "SELECT \n    E.
FirstName || ' ' || E.LastName AS EmployeeName,\n    R.FirstName || ' ' || R.LastName AS ReportingManagerNa
me\nFROM \n    Employee AS E\nLEFT JOIN \n    Employee AS R ON E.ReportsTo = R.EmployeeId;", '  \n    List
all albums and their corresponding artist names  \n', 'SELECT \n    A.Title AS AlbumTitle,\n    Ar.Name AS
ArtistName\nFROM \n    Album AS A\nJOIN \n    Artist AS Ar ON A.ArtistId = Ar.ArtistId;', '  \n    List all
customers who have not placed any orders:\n']
```sql
SELECT
    *
FROM
    Customer
WHERE
    CustomerId NOT IN (SELECT DISTINCT CustomerId FROM Invoice);
```
SELECT
    *
FROM
    Customer
WHERE
    CustomerId NOT IN (SELECT DISTINCT CustomerId FROM Invoice);
SELECT
    *
FROM
    Customer
```

```
WHERE
    CustomerId NOT IN (SELECT DISTINCT CustomerId FROM Invoice);
Empty DataFrame
Columns: [CustomerId, FirstName, LastName, Company, Address, City, State, Country, PostalCode, Phone, Fax,
Email, SupportRepId]
Index: []
```

Out[34]:
```
('SELECT \n    *\nFROM \n    Customer\nWHERE \n    CustomerId NOT IN (SELECT DISTINCT CustomerId FROM Invo
ice);',
 Empty DataFrame
 Columns: [CustomerId, FirstName, LastName, Company, Address, City, State, Country, PostalCode, Phone, Fa
x, Email, SupportRepId]
 Index: [],
 Figure({
     'data': [{'mode': 'number', 'title': {'text': 'Customers Without Orders'}, 'type': 'indicator', 'valu
e': 0}],
     'layout': {'font': {'size': 12}, 'template': '...', 'title': {'text': 'Customers Without Orders'}}
 }))
```

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 INTEGER,\n   MediaTypeId INTEGER NOT NULL,\n   GenreId INTEGER,\n   Composer NVARCHAR(220),\n   Milliseconds INTEGER NOT NULL,\n   Bytes INTEGER,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Track PRIMARY KEY (TrackId),\n   FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_AlbumArtistId ON Album (ArtistId)\n\nCREATE INDEX IFK_TrackAlbumId ON Track (AlbumId)\n\nCREATE TABLE Artist\n(\n   ArtistId INTEGER NOT NULL,\n   Name NVARCHAR(120),\n   CONSTRAINT PK_Artist PRIMARY KEY (ArtistId)\n)\n\nCREATE INDEX IFK_TrackGenreId ON Track (GenreId)\n\nCREATE TABLE Album\n(\n   AlbumId INTEGER NOT NULL,\n   Title NVARCHAR(160) NOT NULL,\n   ArtistId INTEGER NOT NULL,\n   CONSTRAINT PK_Album PRIMARY KEY (AlbumId),\n   FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_PlaylistTrackTrackId ON PlaylistTrack (TrackId)\n\nCREATE INDEX IFK_TrackMediaTypeId ON Track (MediaTypeId)\n\nCREATE TABLE Playlist\n(\n   PlaylistId INTEGER NOT NULL,\n   Name NVARCHAR(120),\n   CONSTRAINT PK_Playlist PRIMARY KEY (PlaylistId)\n)\n\nCREATE TABLE PlaylistTrack\n(\n   PlaylistId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   CONSTRAINT PK_PlaylistTrack PRIMARY KEY (PlaylistId, TrackId),\n   FOREIGN KEY (PlaylistId) REFERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n===Additional Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficient, please generate a valid SQL query without any explanations for the question. \n2. If the provided context is almost sufficient but requires knowledge of a specific string in a particular column, please generate an intermediate SQL query to find the distinct strings in that column. Prepend the query with a comment saying intermediate_sql \n3. If the provided context is insufficient, please explain why it can't be generated. \n4. Please use the most relevant table(s). \n5. If the question has been asked and answered before, please repeat the answer exactly as it was given before. \n", ' \n   Find the top 5 most expensive tracks (based on unit price):\n', 'SELECT \n   T.Name AS TrackName,\n   T.UnitPrice AS UnitPrice\nFROM \n   Track AS T\nORDER BY \n   T.UnitPrice DESC\nLIMIT \n   5;', ' \n   List all genres and the number of tracks in each genre:\n', 'SELECT \n   G.Name AS GenreName,\n   COUNT(T.TrackId) AS TrackCount\nFROM \n   Genre AS G\nLEFT JOIN \n   Track AS T ON G.GenreId = T.GenreId\nGROUP BY \n   G.GenreId\nORDER BY \n   TrackCount DESC;', ' \n   List all albums and their corresponding artist names \n', 'SELECT \n   A.Title AS AlbumTitle,\n   Ar.Name AS ArtistName\nFROM \n   Album AS A\nJOIN \n   Artist AS Ar ON A.ArtistId = Ar.ArtistId;', 'what are the top 5 countries that customers come from?', 'SELECT \n   C.Country,\n   COUNT(*) AS CustomerCount\nFROM \n   Customer AS C\nGROUP BY \n   C.Country\nORDER BY \n   CustomerCount DESC\nLIMIT \n   5;', ' \n   Find all tracks with a name containing "What" (case-insensitive)\n', "SELECT \n   T.Name AS TrackName\nFROM \n   Track AS T\nWHERE \n   UPPER(T.Name) LIKE '%WHAT%';", ' \n   List all invoices with a total exceeding $10:\n', 'SELECT \n   *\nFROM \n   Invoice\nWHERE \n   Total > 10;', ' \n   Get the average invoice total for each customer:\n', "SELECT \n   C.FirstName || ' ' || C.LastName AS CustomerName,\n   AVG(I.Total) AS AverageInvoiceTotal\nFROM \n   Customer AS C\nLEFT JOIN \n   Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n   C.CustomerId;", 'How many customers are there', 'SELECT \n   COUNT(*)\nFROM \n   Customer;', ' \n   Find the total number of invoices per country:\n', 'SELECT \n

```
C.Country,\n    COUNT(I.InvoiceId) AS TotalInvoices\nFROM \n    Customer AS C\nLEFT JOIN \n    Invoice AS I
ON C.CustomerId = I.CustomerId\nGROUP BY \n    C.Country\nORDER BY \n    TotalInvoices DESC;', 'How many re
cords are in table called customer', 'SELECT \n    COUNT(*)\nFROM \n    Customer;', '  \n    Get the top 10
most popular artists (based on the number of tracks):\n']
```sql
SELECT
    A.Name AS ArtistName,
    COUNT(T.TrackId) AS TrackCount
FROM
    Artist AS A
LEFT JOIN
    Album AS Al ON A.ArtistId = Al.ArtistId
LEFT JOIN
    Track AS T ON Al.AlbumId = T.AlbumId
GROUP BY
    A.ArtistId
ORDER BY
    TrackCount DESC
LIMIT
    10;
```
SELECT
    A.Name AS ArtistName,
    COUNT(T.TrackId) AS TrackCount
FROM
    Artist AS A
LEFT JOIN
    Album AS Al ON A.ArtistId = Al.ArtistId
LEFT JOIN
    Track AS T ON Al.AlbumId = T.AlbumId
GROUP BY
    A.ArtistId
ORDER BY
    TrackCount DESC
LIMIT
    10;
SELECT
    A.Name AS ArtistName,
    COUNT(T.TrackId) AS TrackCount
FROM
    Artist AS A
LEFT JOIN
```

```
    Album AS Al ON A.ArtistId = Al.ArtistId
LEFT JOIN
    Track AS T ON Al.AlbumId = T.AlbumId
GROUP BY
    A.ArtistId
ORDER BY
    TrackCount DESC
LIMIT
    10;
```

	ArtistName	TrackCount
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
Track Count
200
150
100
50
0
Iron Maiden
U2
Led Zeppelin
Metallica
Deep Purple
Lost
Pearl Jam
Lenny Kravitz
Various Artists
The Office
Artist

Out[35]:
```
('SELECT \n    A.Name AS ArtistName,\n    COUNT(T.TrackId) AS TrackCount\nFROM \n    Artist AS A\nLEFT JOI
N \n    Album AS Al ON A.ArtistId = Al.ArtistId\nLEFT JOIN \n    Track AS T ON Al.AlbumId = T.AlbumId\nGRO
UP BY \n    A.ArtistId\nORDER BY \n    TrackCount DESC\nLIMIT \n    10;',
         ArtistName  TrackCount
 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'},
                'xaxis': {'title': {'text': 'Artist'}},
                'yaxis': {'title': {'text': 'Track Count'}}}
 }))
```

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   LastName NVARCHAR(20) NOT NULL,\n   Company NVARCHAR(80),\n   Address NVARCHAR(70),\n   City NVARCHAR(40),\n   State NVARCHAR(40),\n   Country NVARCHAR(40),\n   PostalCode NVARCHAR(10),\n   Phone NVARCHAR(24),\n   Fax NVARCHAR(24),\n   Email NVARCHAR(60) NOT NULL,\n   SupportRepId INTEGER,\n   CONSTRAINT PK_Customer PRIMARY KEY (CustomerId),\n   FOREIGN KEY (SupportRepId) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_CustomerSupportRepId ON Customer (SupportRepId)\n\nCREATE TABLE Invoice\n(\n   InvoiceId INTEGER NOT NULL,\n   CustomerId INTEGER NOT NULL,\n   InvoiceDate DATETIME NOT NULL,\n   BillingAddress NVARCHAR(70),\n   BillingCity NVARCHAR(40),\n   BillingState NVARCHAR(40),\n   BillingCountry NVARCHAR(40),\n   BillingPostalCode NVARCHAR(10),\n   Total NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Invoice PRIMARY KEY (InvoiceId),\n   FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)\n\nCREATE TABLE Employee\n(\n   EmployeeId INTEGER NOT NULL,\n   LastName NVARCHAR(20) NOT NULL,\n   FirstName NVARCHAR(20) NOT NULL,\n   Title NVARCHAR(30),\n   ReportsTo INTEGER,\n   BirthDate DATETIME,\n   HireDate DATETIME,\n   Address NVARCHAR(70),\n   City NVARCHAR(40),\n   State NVARCHAR(40),\n   Country NVARCHAR(40),\n   PostalCode NVARCHAR(10),\n   Phone NVARCHAR(24),\n   Fax NVARCHAR(24),\n   Email NVARCHAR(60),\n   CONSTRAINT PK_Employee PRIMARY KEY (EmployeeId),\n   FOREIGN KEY (ReportsTo) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE InvoiceLine\n(\n   InvoiceLineId INTEGER NOT NULL,\n   InvoiceId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   Quantity INTEGER NOT NULL,\n   CONSTRAINT PK_InvoiceLine PRIMARY KEY (InvoiceLineId),\n   FOREIGN KEY (InvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceLineTrackId ON InvoiceLine (TrackId)\n\nCREATE INDEX IFK_InvoiceLineInvoiceId ON InvoiceLine (InvoiceId)\n\nCREATE INDEX IFK_EmployeeReportsTo ON Employee (ReportsTo)\n\nCREATE TABLE PlaylistTrack\n(\n   PlaylistId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   CONSTRAINT PK_PlaylistTrack PRIMARY KEY (PlaylistId, TrackId),\n   FOREIGN KEY (PlaylistId) REFERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n===Additional Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficient, please generate a valid SQL query without any explanations for the question. \n2. If the provided context is almost sufficient but requires knowledge of a specific string in a particular column, please generate an intermediate SQL query to find the distinct strings in that column. Prepend the query with a comment saying intermediate_sql \n3. If the provided context is insufficient, please explain why it can't be generated. \n4. Please use the most relevant table(s). \n5. If the question has been asked and answered before, please repeat the answer exactly as it was given before. \n", 'what are the top 5 countries that customers come from?', 'SELECT \n   C.Country,\n   COUNT(*) AS CustomerCount\nFROM \n   Customer AS C\nGROUP BY \n   C.Country\nORDER BY \n   CustomerCount DESC\nLIMIT \n   5;', ' \n   Find the total number of invoices per country:\n', 'SELECT \n   C.Country,\n   COUNT(I.InvoiceId) AS TotalInvoices\nFROM \n   Customer AS C\nLEFT JOIN \n   Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n   C.Country\nORDER BY \n   TotalInvoices DESC;', ' \n   Get the total number of invoices for each customer \n', "SELECT \n   C.FirstName || ' ' || C.LastName AS CustomerName,\n   COUNT(I.InvoiceId) AS TotalInvoic

```
es\nFROM \n    Customer AS C\nLEFT JOIN \n    Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n
C.CustomerId;", 'How many customers are there', 'SELECT \n    COUNT(*)\nFROM \n    Customer;', '  \n    Get
the average invoice total for each customer:\n', "SELECT \n    C.FirstName || ' ' || C.LastName AS Customer
Name,\n    AVG(I.Total) AS AverageInvoiceTotal\nFROM \n    Customer AS C\nLEFT JOIN \n    Invoice AS I ON
C.CustomerId = I.CustomerId\nGROUP BY \n    C.CustomerId;", '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.FirstName || ' ' || E.LastName AS EmployeeName,\n    R.FirstName ||
' ' || R.LastName AS ReportingManagerName\nFROM \n    Employee AS E\nLEFT JOIN \n    Employee AS R ON E.Rep
ortsTo = R.EmployeeId;", '  \n    List all invoices with a total exceeding $10:\n', 'SELECT \n    *\nFROM
\n    Invoice\nWHERE \n    Total > 10;', '  \n    Find all invoices since 2010 and the total amount invoice
d:\n', "SELECT \n    I.InvoiceDate,\n    SUM(IL.UnitPrice * IL.Quantity) AS TotalAmount\nFROM \n    Invoice
AS I\nJOIN \n    InvoiceLine AS IL ON I.InvoiceId = IL.InvoiceId\nWHERE \n    I.InvoiceDate >= '2010-01-0
1'\nGROUP BY \n    I.InvoiceDate\nORDER BY \n    I.InvoiceDate;", '  \n    List all albums and their corres
ponding artist names  \n', 'SELECT \n    A.Title AS AlbumTitle,\n    Ar.Name AS ArtistName\nFROM \n    Albu
m AS A\nJOIN \n    Artist AS Ar ON A.ArtistId = Ar.ArtistId;', '  \n     List all customers from Canada and
their email addresses:\n']
```sql
SELECT
    C.FirstName || ' ' || C.LastName AS CustomerName,
    C.Email
FROM
    Customer AS C
WHERE
    C.Country = 'Canada';
```
SELECT
    C.FirstName || ' ' || C.LastName AS CustomerName,
    C.Email
FROM
    Customer AS C
WHERE
    C.Country = 'Canada';
SELECT
    C.FirstName || ' ' || C.LastName AS CustomerName,
    C.Email
FROM
    Customer AS C
WHERE
    C.Country = 'Canada';
        CustomerName   Email
0 François Tremblay   ftremblay@gmail.com
1   Mark Philips   mphilips12@shaw.ca
```

```
2  Jennifer Peterson     jenniferp@rogers.ca
3       Robert Brown        robbrown@shaw.ca
4     Edward Francis     edfrancis@yachoo.ca
5        Martha Silk    marthasilk@gmail.com
6     Aaron Mitchell  aaronmitchell@yahoo.ca
7     Ellie Sullivan  ellie.sullivan@shaw.ca
```



Out[36]:
```
("SELECT \n    C.FirstName || ' ' || C.LastName AS CustomerName,\n    C.Email\nFROM \n    Customer AS C\nW
HERE \n    C.Country = 'Canada';",
         CustomerName                   Email
 0  François Tremblay     ftremblay@gmail.com
 1       Mark Philips      mphilips12@shaw.ca
 2  Jennifer Peterson     jenniferp@rogers.ca
 3       Robert Brown        robbrown@shaw.ca
 4     Edward Francis     edfrancis@yachoo.ca
 5        Martha Silk    marthasilk@gmail.com
 6     Aaron Mitchell  aaronmitchell@yahoo.ca
 7     Ellie Sullivan  ellie.sullivan@shaw.ca,
 Figure({
     'data': [{'alignmentgroup': 'True',
               'hovertemplate': 'CustomerName=%{x}<br>Email=%{y}<extra></extra>',
               'legendgroup': '',
               'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
               'name': '',
               'offsetgroup': '',
               'orientation': 'v',
               'showlegend': False,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['François Tremblay', 'Mark Philips', 'Jennifer Peterson',
                           'Robert Brown', 'Edward Francis', 'Martha Silk', 'Aaron Mitchell',
                           'Ellie Sullivan'], dtype=object),
               'xaxis': 'x',
               'y': array(['ftremblay@gmail.com', 'mphilips12@shaw.ca', 'jenniferp@rogers.ca',
                           'robbrown@shaw.ca', 'edfrancis@yachoo.ca', 'marthasilk@gmail.com',
                           'aaronmitchell@yahoo.ca', 'ellie.sullivan@shaw.ca'], dtype=object),
               'yaxis': 'y'}],
     'layout': {'barmode': 'relative',
                'legend': {'tracegroupgap': 0},
                'margin': {'t': 60},
                'template': '...',
                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'CustomerName'}},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'Email'}}}
 }))
```

In [37]:
```
question = """
     Find the customer with the most invoices
"""
```

```
vn.ask(question=question)
```

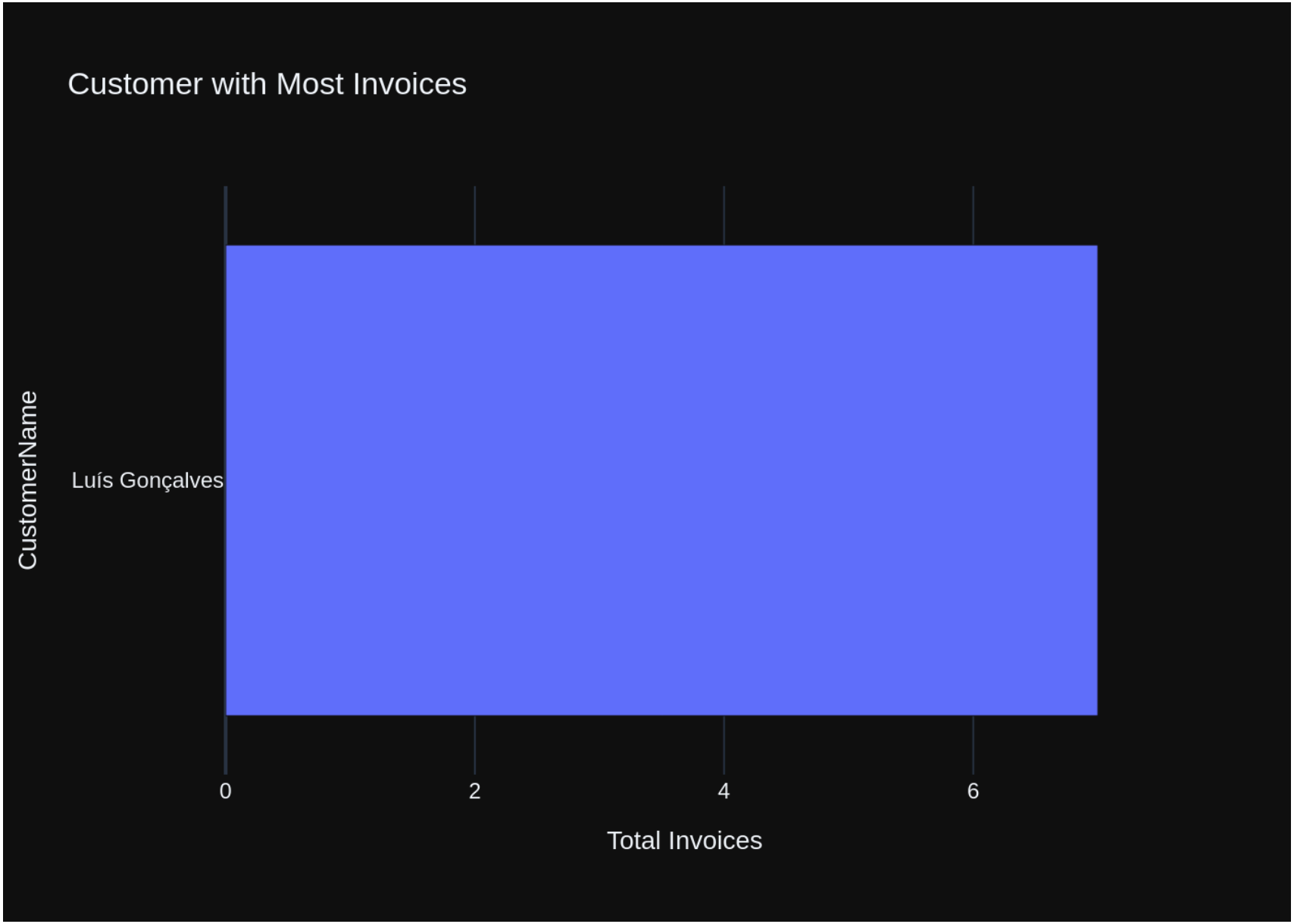
```
Number of requested results 10 is greater than number of elements in index 1, updating n_results = 1
```

["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 INTEGER NOT NULL,\n   CustomerId INTEGER NOT NULL,\n   InvoiceDate DATETIME NOT NULL,\n   BillingAddress NVARCHAR(70),\n   BillingCity NVARCHAR(40),\n   BillingState NVARCHAR(40),\n   BillingCountry NVARCHAR(40),\n   BillingPostalCode NVARCHAR(10),\n   Total NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Invoice PRIMARY KEY (InvoiceId),\n   FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceLineInvoiceId ON InvoiceLine (InvoiceId)\n\nCREATE TABLE InvoiceLine\n(\n   InvoiceLineId INTEGER NOT NULL,\n   InvoiceId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   Quantity INTEGER NOT NULL,\n   CONSTRAINT PK_InvoiceLine PRIMARY KEY (InvoiceLineId),\n   FOREIGN KEY (InvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceLineTrackId ON InvoiceLine (TrackId)\n\nCREATE TABLE Customer\n(\n   CustomerId INTEGER NOT NULL,\n   FirstName NVARCHAR(40) NOT NULL,\n   LastName NVARCHAR(20) NOT NULL,\n   Company NVARCHAR(80),\n   Address NVARCHAR(70),\n   City NVARCHAR(40),\n   State NVARCHAR(40),\n   Country NVARCHAR(40),\n   PostalCode NVARCHAR(10),\n   Phone NVARCHAR(24),\n   Fax NVARCHAR(24),\n   Email NVARCHAR(60) NOT NULL,\n   SupportRepId INTEGER,\n   CONSTRAINT PK_Customer PRIMARY KEY (CustomerId),\n   FOREIGN KEY (SupportRepId) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_CustomerSupportRepId ON Customer (SupportRepId)\n\nCREATE TABLE Employee\n(\n   EmployeeId INTEGER NOT NULL,\n   LastName NVARCHAR(20) NOT NULL,\n   FirstName NVARCHAR(20) NOT NULL,\n   Title NVARCHAR(30),\n   ReportsTo INTEGER,\n   BirthDate DATETIME,\n   HireDate DATETIME,\n   Address NVARCHAR(70),\n   City NVARCHAR(40),\n   State NVARCHAR(40),\n   Country NVARCHAR(40),\n   PostalCode NVARCHAR(10),\n   Phone NVARCHAR(24),\n   Fax NVARCHAR(24),\n   Email NVARCHAR(60),\n   CONSTRAINT PK_Employee PRIMARY KEY (EmployeeId),\n   FOREIGN KEY (ReportsTo) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_EmployeeReportsTo ON Employee (ReportsTo)\n\nCREATE TABLE Track\n(\n   TrackId INTEGER NOT NULL,\n   Name NVARCHAR(200) NOT NULL,\n   AlbumId INTEGER,\n   MediaTypeId INTEGER NOT NULL,\n   GenreId INTEGER,\n   Composer NVARCHAR(220),\n   Milliseconds INTEGER NOT NULL,\n   Bytes INTEGER,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Track PRIMARY KEY (TrackId),\n   FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n===Additional Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficient, please generate a valid SQL query without any explanations for the question. \n2. If the provided context is almost sufficient but requires knowledge of a specific string in a particular column, please generate an intermediate SQL query to find the distinct strings in that column. Prepend the query with a comment saying intermediate_sql \n3. If the provided context is insufficient, please explain why it can't be generated. \n4. Please use the most relevant table(s). \n5. If the question has been asked and answered before, please repeat the answer exactly as it was given before. \n", ' \n   Get the total number of invoices for each customer\n', "SELECT \n   C.FirstName || ' ' || C.LastName AS CustomerName,\n   COUNT(I.InvoiceId) AS TotalInvoices\nFROM \n   Customer AS C\nLEFT JOIN \n   Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n   C.CustomerId;", ' \n   Find the total number of invoices per country:\n', 'SELECT \n   C.Countr

y,\n   COUNT(I.InvoiceId) AS TotalInvoices\nFROM \n   Customer AS C\nLEFT JOIN \n   Invoice AS I ON C.Cu
stomerId = I.CustomerId\nGROUP BY \n   C.Country\nORDER BY \n   TotalInvoices DESC;', ' \n   List all i
nvoices with a total exceeding $10:\n', 'SELECT \n   *\nFROM \n   Invoice\nWHERE \n   Total > 10;', '
\n   Find all invoices since 2010 and the total amount invoiced:\n', "SELECT \n   I.InvoiceDate,\n   SUM
(IL.UnitPrice * IL.Quantity) AS TotalAmount\nFROM \n   Invoice AS I\nJOIN \n   InvoiceLine AS IL ON I.Inv
oiceId = IL.InvoiceId\nWHERE \n   I.InvoiceDate >= '2010-01-01'\nGROUP BY \n   I.InvoiceDate\nORDER BY \n
I.InvoiceDate;", ' \n   Get the average invoice total for each customer:\n', "SELECT \n   C.FirstName ||
' ' || C.LastName AS CustomerName,\n   AVG(I.Total) AS AverageInvoiceTotal\nFROM \n   Customer AS C\nLEFT
JOIN \n   Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n   C.CustomerId;", 'what are the top 5
countries that customers come from?', 'SELECT \n   C.Country,\n   COUNT(*) AS CustomerCount\nFROM \n   C
ustomer AS C\nGROUP BY \n   C.Country\nORDER BY \n   CustomerCount DESC\nLIMIT \n   5;', ' \n   Find t
he top 5 most expensive tracks (based on unit price):\n', 'SELECT \n   T.Name AS TrackName,\n   T.UnitPri
ce AS UnitPrice\nFROM \n   Track AS T\nORDER BY \n   T.UnitPrice DESC\nLIMIT \n   5;', 'How many custome
rs are there', 'SELECT \n   COUNT(*)\nFROM \n   Customer;', 'How many records are in table called custome
r', 'SELECT \n   COUNT(*)\nFROM \n   Customer;', ' \n   List all customers from Canada and their email
addresses:\n', "SELECT \n   C.FirstName || ' ' || C.LastName AS CustomerName,\n   C.Email\nFROM \n   Cus
tomer AS C\nWHERE \n   C.Country = 'Canada';", ' \n   Find the customer with the most invoices \n']

```
SELECT
    C.FirstName || ' ' || C.LastName AS CustomerName,
    COUNT(I.InvoiceId) AS TotalInvoices
FROM
    Customer AS C
LEFT JOIN
    Invoice AS I ON C.CustomerId = I.CustomerId
GROUP BY
    C.CustomerId
ORDER BY
    TotalInvoices DESC
LIMIT
    1;
SELECT
    C.FirstName || ' ' || C.LastName AS CustomerName,
    COUNT(I.InvoiceId) AS TotalInvoices
FROM
    Customer AS C
LEFT JOIN
    Invoice AS I ON C.CustomerId = I.CustomerId
GROUP BY
    C.CustomerId
ORDER BY
    TotalInvoices DESC
LIMIT
```

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


Customer with Most Invoices
CustomerName
Luís Gonçalves
0
2
4
6
Total Invoices

Out[37]:
```
("SELECT \n    C.FirstName || ' ' || C.LastName AS CustomerName,\n    COUNT(I.InvoiceId) AS TotalInvoices
\nFROM \n    Customer AS C\nLEFT JOIN \n    Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n    C.
CustomerId\nORDER BY \n    TotalInvoices DESC\nLIMIT \n    1;",
      CustomerName  TotalInvoices
 0  Luís Gonçalves             7,
 Figure({
     'data': [{'alignmentgroup': 'True',
               'hovertemplate': 'TotalInvoices=%{x}<br>CustomerName=%{y}<extra></extra>',
               'legendgroup': '',
               'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
               'name': '',
               'offsetgroup': '',
               'orientation': 'h',
               'showlegend': False,
               'textposition': 'auto',
               'type': 'bar',
               'x': array([7]),
               'xaxis': 'x',
               'y': array(['Luís Gonçalves'], dtype=object),
               'yaxis': 'y'}],
     'layout': {'barmode': 'relative',
                'legend': {'tracegroupgap': 0},
                'template': '...',
                'title': {'text': 'Customer with Most Invoices'},
                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'Total Invoices'}},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'CustomerName'}}}
 }))
```

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
```

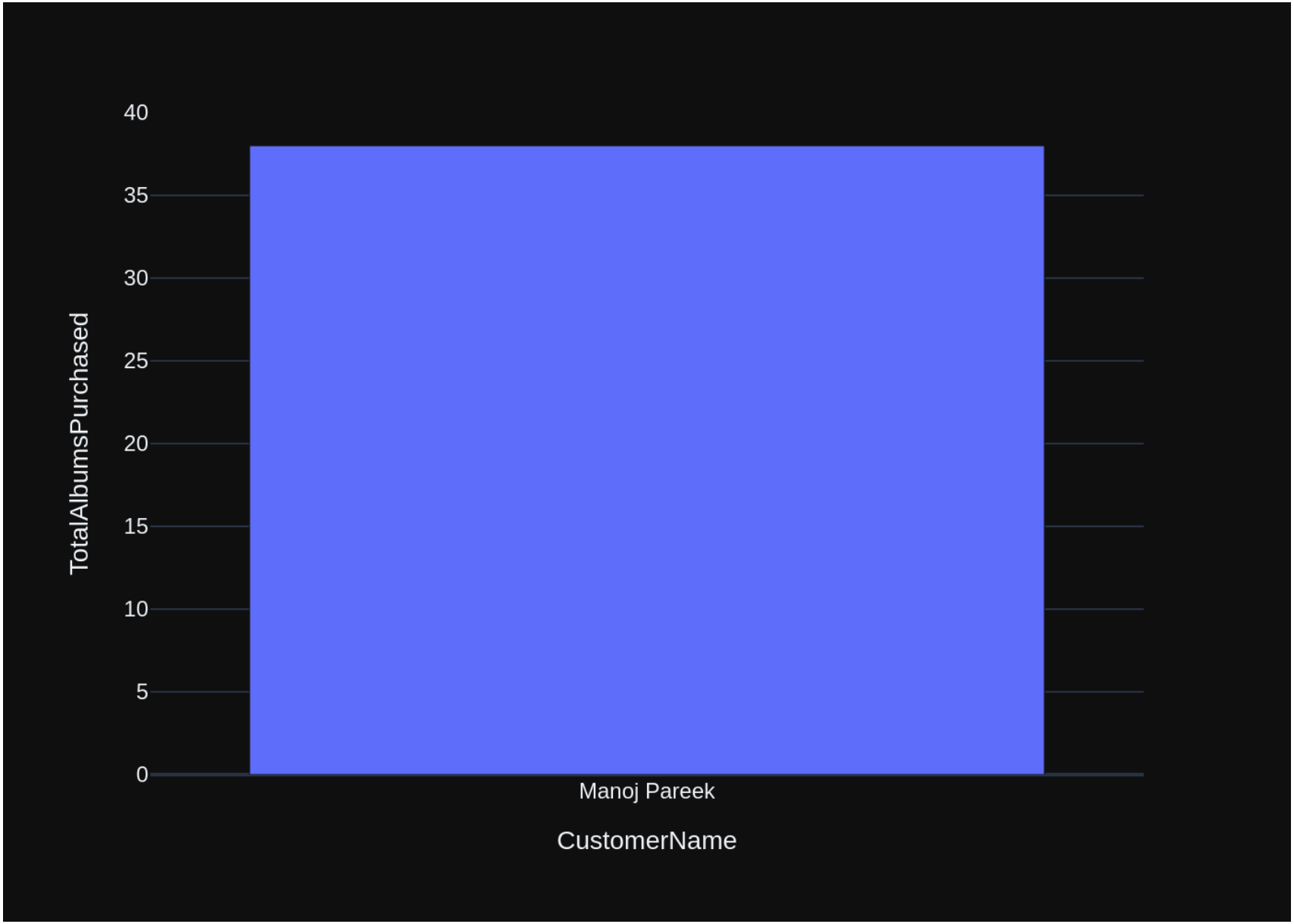
["You are a SQLite expert. Please help to generate a SQL query to answer the question. Your response should ONLY be based on the given context and follow the response guidelines and format instructions. \n===Tables \nCREATE TABLE Track\n(\n   TrackId INTEGER NOT NULL,\n   Name NVARCHAR(200) NOT NULL,\n   AlbumId INTEGER,\n   MediaTypeId INTEGER NOT NULL,\n   GenreId INTEGER,\n   Composer NVARCHAR(220),\n   Milliseconds INTEGER NOT NULL,\n   Bytes INTEGER,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Track PRIMARY KEY (TrackId),\n   FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Album\n(\n   AlbumId INTEGER NOT NULL,\n   Title NVARCHAR(160) NOT NULL,\n   ArtistId INTEGER NOT NULL,\n   CONSTRAINT PK_Album PRIMARY KEY (AlbumId),\n   FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_AlbumArtistId ON Album (ArtistId)\n\nCREATE TABLE InvoiceLine\n(\n   InvoiceLineId INTEGER NOT NULL,\n   InvoiceId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   Quantity INTEGER NOT NULL,\n   CONSTRAINT PK_InvoiceLine PRIMARY KEY (InvoiceLineId),\n   FOREIGN KEY (InvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)\n\nCREATE TABLE Invoice\n(\n   InvoiceId INTEGER NOT NULL,\n   CustomerId INTEGER NOT NULL,\n   InvoiceDate DATETIME NOT NULL,\n   BillingAddress NVARCHAR(70),\n   BillingCity NVARCHAR(40),\n   BillingState NVARCHAR(40),\n   BillingCountry NVARCHAR(40),\n   BillingPostalCode NVARCHAR(10),\n   Total NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Invoice PRIMARY KEY (InvoiceId),\n   FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_TrackAlbumId ON Track (AlbumId)\n\nCREATE INDEX IFK_InvoiceLineInvoiceId ON InvoiceLine (InvoiceId)\n\nCREATE INDEX IFK_InvoiceLineTrackId ON InvoiceLine (TrackId)\n\nCREATE TABLE PlaylistTrack\n(\n   PlaylistId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   CONSTRAINT PK_PlaylistTrack PRIMARY KEY (PlaylistId, TrackId),\n   FOREIGN KEY (PlaylistId) REFERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\n\n===Additional Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficient, please generate a valid SQL query without any explanations for the question. \n2. If the provided context is almost sufficient but requires knowledge of a specific string in a particular column, please generate an intermediate SQL query to find the distinct strings in that column. Prepend the query with a comment saying intermediate_sql \n3. If the provided context is insufficient, please explain why it can't be generated. \n4. Please use the most relevant table(s). \n5. If the question has been asked and answered before, please repeat the answer exactly as it was given before. \n", ' \n   Find the customer with the most invoices \n', "SELECT \n   C.FirstName || ' ' || C.LastName AS CustomerName,\n   COUNT(I.InvoiceId) AS TotalInvoices\nFROM \n   Customer AS C\nLEFT JOIN \n   Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n   C.CustomerId\nORDER BY \n   TotalInvoices DESC\nLIMIT \n   1;", ' \n   Get the total number of invoices for each customer\n', "SELECT \n   C.FirstName || ' ' || C.LastName AS CustomerName,\n   COUNT(I.InvoiceId) AS TotalInvoices\nFROM \n   Customer AS C\nLEFT JOIN \n   Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n   C.CustomerId;", ' \n   Get the top 10 most popular artists (based on the number of tracks):\n', 'SELECT \n   A.Name AS ArtistName,\n   COUNT(T.TrackId) AS TrackCount\nFROM \n   Artist AS A\nLEFT JOIN \n   Album AS Al ON A.ArtistId = Al.ArtistId\nLEFT JOIN \n   Track AS T ON Al.AlbumId = T.AlbumId\nGROUP BY \n   A.ArtistId\nORDE

R BY \n   TrackCount DESC\nLIMIT \n   10;', ' \n   Find all invoices since 2010 and the total amount in
voiced:\n', "SELECT \n   I.InvoiceDate,\n   SUM(IL.UnitPrice * IL.Quantity) AS TotalAmount\nFROM \n   In
voice AS I\nJOIN \n   InvoiceLine AS IL ON I.InvoiceId = IL.InvoiceId\nWHERE \n   I.InvoiceDate >= '2010-
01-01'\nGROUP BY \n   I.InvoiceDate\nORDER BY \n   I.InvoiceDate;", ' \n   Find the total number of inv
oices per country:\n', 'SELECT \n   C.Country,\n   COUNT(I.InvoiceId) AS TotalInvoices\nFROM \n   Custom
er AS C\nLEFT JOIN \n   Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n   C.Country\nORDER BY \n
TotalInvoices DESC;', ' \n   Find the top 5 most expensive tracks (based on unit price):\n', 'SELECT \n
T.Name AS TrackName,\n   T.UnitPrice AS UnitPrice\nFROM \n   Track AS T\nORDER BY \n   T.UnitPrice DESC
\nLIMIT \n   5;', ' \n   Get the average invoice total for each customer:\n', "SELECT \n   C.FirstName
|| ' ' || C.LastName AS CustomerName,\n   AVG(I.Total) AS AverageInvoiceTotal\nFROM \n   Customer AS C\nL
EFT JOIN \n   Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n   C.CustomerId;", ' \n   List al
l invoices with a total exceeding $10:\n', 'SELECT \n   *\nFROM \n   Invoice\nWHERE \n   Total > 10;', '
\n   List all albums and their corresponding artist names \n', 'SELECT \n   A.Title AS AlbumTitle,\n
Ar.Name AS ArtistName\nFROM \n   Album AS A\nJOIN \n   Artist AS Ar ON A.ArtistId = Ar.ArtistId;', ' \n
List all genres and the number of tracks in each genre:\n', 'SELECT \n   G.Name AS GenreName,\n   COUNT
(T.TrackId) AS TrackCount\nFROM \n   Genre AS G\nLEFT JOIN \n   Track AS T ON G.GenreId = T.GenreId\nGROU
P BY \n   G.GenreId\nORDER BY \n   TrackCount DESC;', ' \n   Find the customer who bought the most alb
ums in total quantity (across all invoices): \n']

```sql
SELECT
    C.FirstName || ' ' || C.LastName AS CustomerName,
    SUM(IL.Quantity) AS TotalAlbumsPurchased
FROM
    Customer AS C
JOIN
    Invoice AS I ON C.CustomerId = I.CustomerId
JOIN
    InvoiceLine AS IL ON I.InvoiceId = IL.InvoiceId
WHERE
    IL.TrackId IN (
        SELECT
            TrackId
        FROM
            Track
        WHERE
            AlbumId IS NOT NULL
    )
GROUP BY
    C.CustomerId
ORDER BY
    TotalAlbumsPurchased DESC
LIMIT
```

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

```
            AlbumId IS NOT NULL
    )
GROUP BY
    C.CustomerId
ORDER BY
    TotalAlbumsPurchased DESC
LIMIT
    1;
   CustomerName  TotalAlbumsPurchased
0  Manoj Pareek                   38
```


40
35
30
25
20
15
10
5
0
TotalAlbumsPurchased
Manoj Pareek
CustomerName

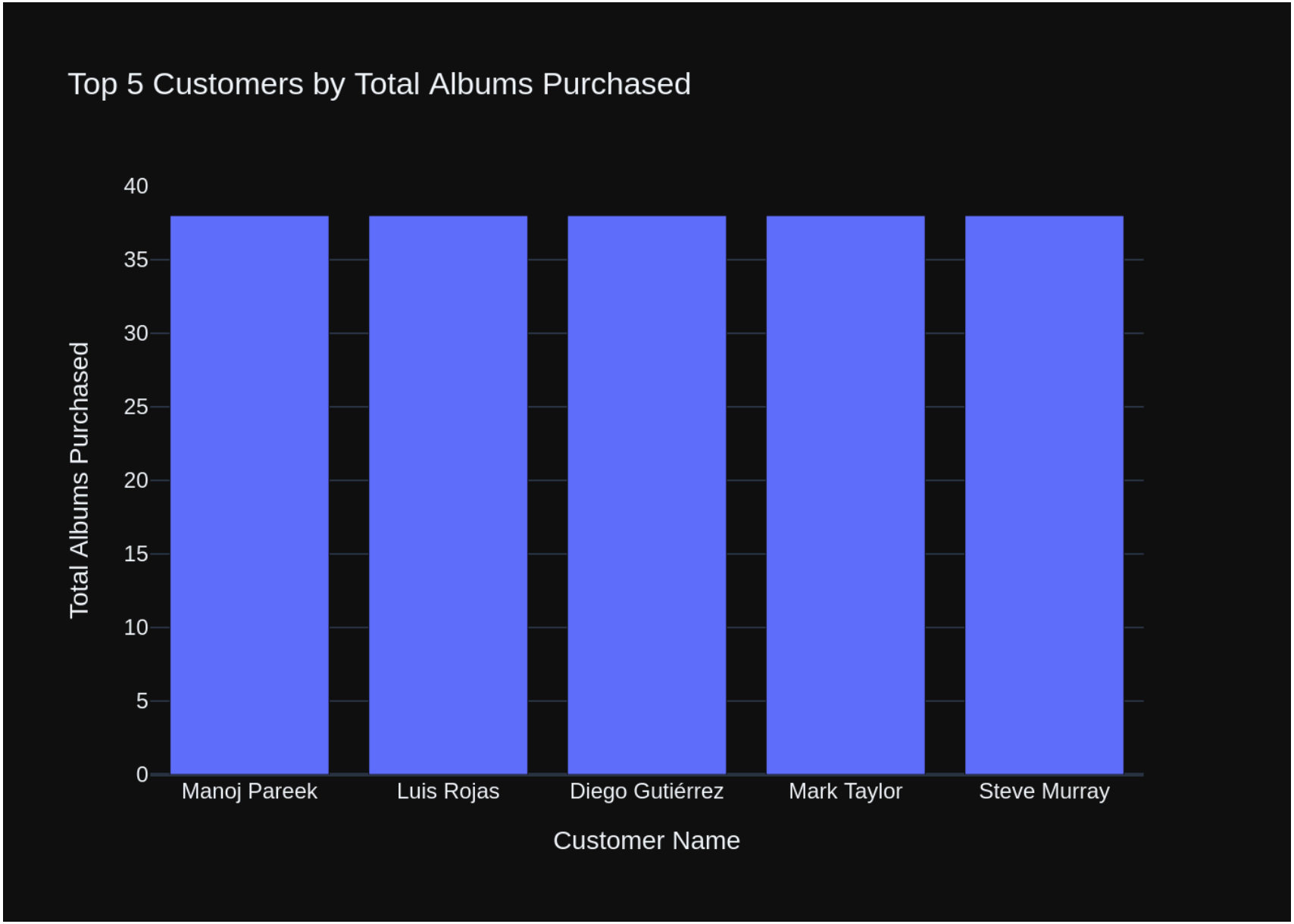
Out[38]:
```
("SELECT \n    C.FirstName || ' ' || C.LastName AS CustomerName,\n    SUM(IL.Quantity) AS TotalAlbumsPurch
ased\nFROM \n    Customer AS C\nJOIN \n    Invoice AS I ON C.CustomerId = I.CustomerId\nJOIN \n    Invoice
Line AS IL ON I.InvoiceId = IL.InvoiceId\nWHERE \n    IL.TrackId IN (\n        SELECT \n            TrackI
d\n        FROM \n            Track\n        WHERE \n            AlbumId IS NOT NULL\n    )\nGROUP BY \n
C.CustomerId\nORDER BY \n    TotalAlbumsPurchased DESC\nLIMIT \n    1;",
    CustomerName  TotalAlbumsPurchased
 0  Manoj Pareek                    38,
 Figure({
     'data': [{'alignmentgroup': 'True',
               'hovertemplate': 'CustomerName=%{x}<br>TotalAlbumsPurchased=%{y}<extra></extra>',
               'legendgroup': '',
               'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
               'name': '',
               'offsetgroup': '',
               'orientation': 'v',
               'showlegend': False,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Manoj Pareek'], dtype=object),
               'xaxis': 'x',
               'y': array([38]),
               'yaxis': 'y'}],
     'layout': {'barmode': 'relative',
                'legend': {'tracegroupgap': 0},
                'margin': {'t': 60},
                'template': '...',
                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'CustomerName'}},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'TotalAlbumsPurchased'}}}
 }))
```

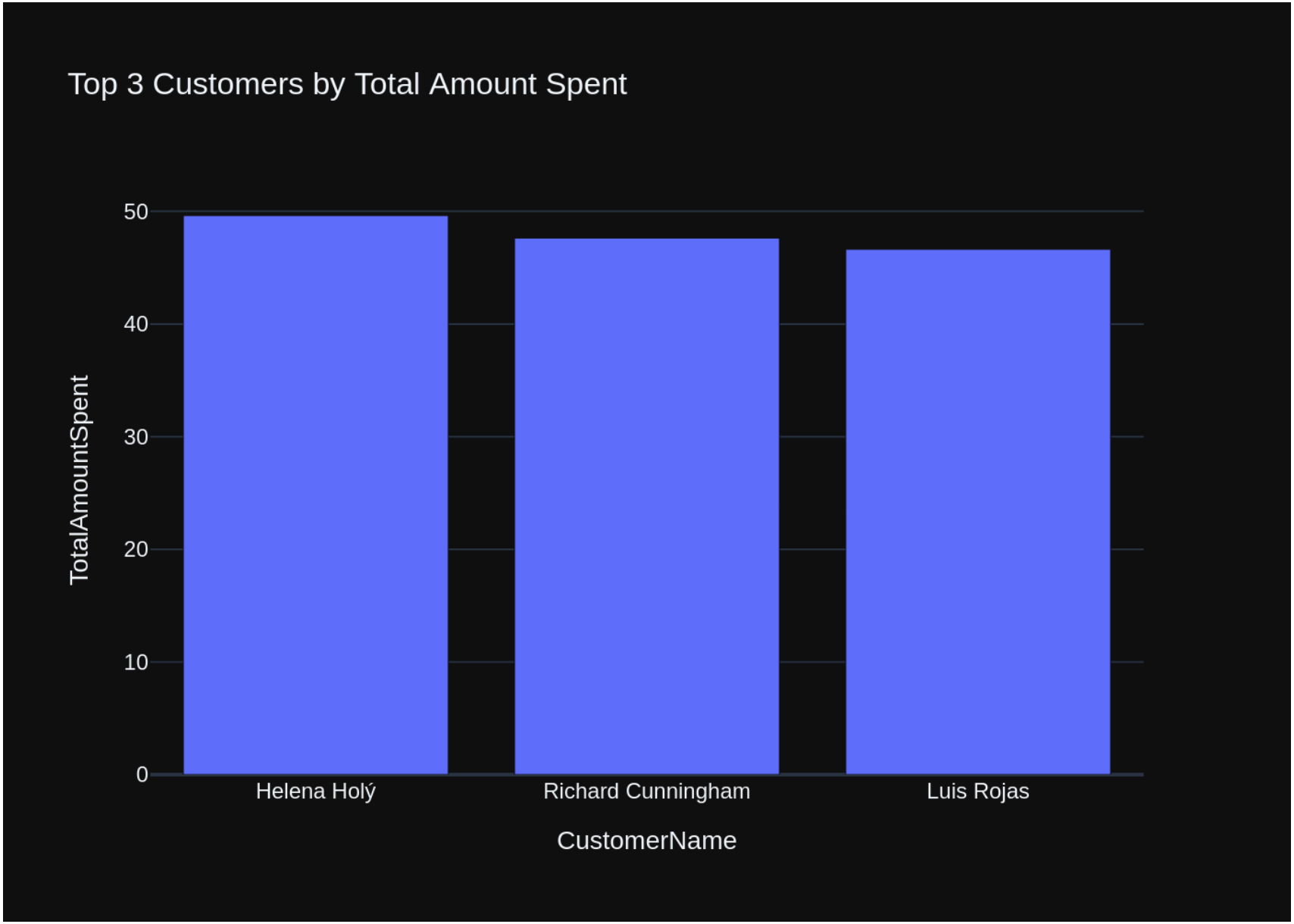
In [39]:
```
question = """
     Find the top 5 customer who bought the most albums in total quantity (across all invoices):
"""

vn.ask(question=question)
```

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

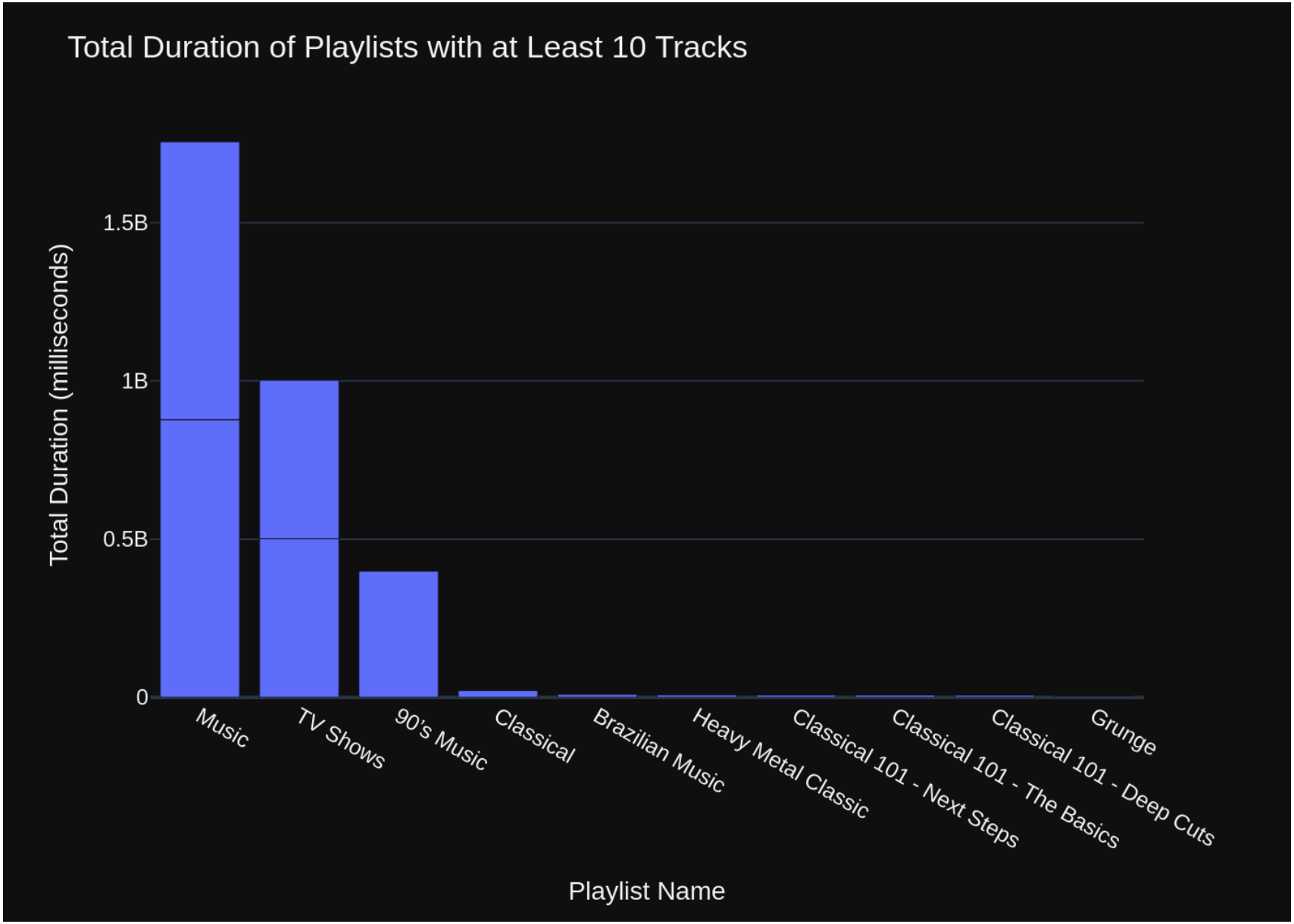
["You are a SQLite expert. Please help to generate a SQL query to answer the question. Your response should ONLY be based on the given context and follow the response guidelines and format instructions. \n===Tables \nCREATE TABLE Track\n(\n   TrackId INTEGER NOT NULL,\n   Name NVARCHAR(200) NOT NULL,\n   AlbumId INTEGER,\n   MediaTypeId INTEGER NOT NULL,\n   GenreId INTEGER,\n   Composer NVARCHAR(220),\n   Milliseconds INTEGER NOT NULL,\n   Bytes INTEGER,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Track PRIMARY KEY (TrackId),\n   FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Album\n(\n   AlbumId INTEGER NOT NULL,\n   Title NVARCHAR(160) NOT NULL,\n   ArtistId INTEGER NOT NULL,\n   CONSTRAINT PK_Album PRIMARY KEY (AlbumId),\n   FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_AlbumArtistId ON Album (ArtistId)\n\nCREATE TABLE InvoiceLine\n(\n   InvoiceLineId INTEGER NOT NULL,\n   InvoiceId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   Quantity INTEGER NOT NULL,\n   CONSTRAINT PK_InvoiceLine PRIMARY KEY (InvoiceLineId),\n   FOREIGN KEY (InvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Invoice\n(\n   InvoiceId INTEGER NOT NULL,\n   CustomerId INTEGER NOT NULL,\n   InvoiceDate DATETIME NOT NULL,\n   BillingAddress NVARCHAR(70),\n   BillingCity NVARCHAR(40),\n   BillingState NVARCHAR(40),\n   BillingCountry NVARCHAR(40),\n   BillingPostalCode NVARCHAR(10),\n   Total NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Invoice PRIMARY KEY (InvoiceId),\n   FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)\n\nCREATE INDEX IFK_TrackAlbumId ON Track (AlbumId)\n\nCREATE INDEX IFK_InvoiceLineTrackId ON InvoiceLine (TrackId)\n\nCREATE INDEX IFK_InvoiceLineInvoiceId ON InvoiceLine (InvoiceId)\n\nCREATE TABLE Artist\n(\n   ArtistId INTEGER NOT NULL,\n   Name NVARCHAR(120),\n   CONSTRAINT PK_Artist PRIMARY KEY (ArtistId)\n)\n\n\n===Additional Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficient, please generate a valid SQL query without any explanations for the question. \n2. If the provided context is almost sufficient but requires knowledge of a specific string in a particular column, please generate an intermediate SQL query to find the distinct strings in that column. Prepend the query with a comment saying intermediate_sql \n3. If the provided context is insufficient, please explain why it can't be generated. \n4. Please use the most relevant table(s). \n5. If the question has been asked and answered before, please repeat the answer exactly as it was given before. \n", ' \n   Find the customer who bought the most albums in total quantity (across all invoices): \n', "SELECT \n   C.FirstName || ' ' || C.LastName AS CustomerName,\n   SUM(IL.Quantity) AS TotalAlbumsPurchased\nFROM \n   Customer AS C\nJOIN \n   Invoice AS I ON C.CustomerId = I.CustomerId\nJOIN \n   InvoiceLine AS IL ON I.InvoiceId = IL.InvoiceId\nWHERE \n   IL.TrackId IN (\n   SELECT \n   TrackId\n   FROM \n   Track\n   WHERE \n   AlbumId IS NOT NULL\n   )\nGROUP BY \n   C.CustomerId\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   A.Name AS ArtistName,\n   COUNT(T.TrackId) AS TrackCount\nFROM \n   Artist AS A\nLEFT JOIN \n   Album AS Al ON A.ArtistId = Al.ArtistId\nLEFT JOIN \n   Track AS T ON Al.AlbumId = T.AlbumId\nGROUP BY \n   A.ArtistId\nORDER BY \n   TrackCount DESC\nLIMIT \n   10;', ' \n   Find the customer with the most invoices \n', "SELECT \n   C.FirstName || ' ' || C.LastName AS CustomerName,\n   COUNT(I.InvoiceId) AS TotalInvoices\nFROM \n   Customer AS C\nLEFT JOIN \n   Invoice AS I ON C.CustomerId

```
= I.CustomerId\nGROUP BY \n    C.CustomerId\nORDER BY \n    TotalInvoices DESC\nLIMIT \n    1;", '  \n    F
ind the top 5 most expensive tracks (based on unit price):\n', 'SELECT \n    T.Name AS TrackName,\n    T.Un
itPrice AS UnitPrice\nFROM \n    Track AS T\nORDER BY \n    T.UnitPrice DESC\nLIMIT \n    5;', '  \n    Get
the total number of invoices for each customer\n', "SELECT \n    C.FirstName || ' ' || C.LastName AS Custom
erName,\n    COUNT(I.InvoiceId) AS TotalInvoices\nFROM \n    Customer AS C\nLEFT JOIN \n    Invoice AS I ON
C.CustomerId = I.CustomerId\nGROUP BY \n    C.CustomerId;", '  \n    Get the average invoice total for each
customer:\n', "SELECT \n    C.FirstName || ' ' || C.LastName AS CustomerName,\n    AVG(I.Total) AS AverageI
nvoiceTotal\nFROM \n    Customer AS C\nLEFT JOIN \n    Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP B
Y \n    C.CustomerId;", '  \n    List all invoices with a total exceeding $10:\n', 'SELECT \n    *\nFROM \n
Invoice\nWHERE \n    Total > 10;', 'what are the top 5 countries that customers come from?', 'SELECT \n
C.Country,\n    COUNT(*) AS CustomerCount\nFROM \n    Customer AS C\nGROUP BY \n    C.Country\nORDER BY \n
CustomerCount DESC\nLIMIT \n    5;', '  \n    Find the total number of invoices per country:\n', 'SELECT \n
C.Country,\n    COUNT(I.InvoiceId) AS TotalInvoices\nFROM \n    Customer AS C\nLEFT JOIN \n    Invoice AS I
ON C.CustomerId = I.CustomerId\nGROUP BY \n    C.Country\nORDER BY \n    TotalInvoices DESC;', '  \n    Fin
d all invoices since 2010 and the total amount invoiced:\n', "SELECT \n    I.InvoiceDate,\n    SUM(IL.UnitP
rice * IL.Quantity) AS TotalAmount\nFROM \n    Invoice AS I\nJOIN \n    InvoiceLine AS IL ON I.InvoiceId =
IL.InvoiceId\nWHERE \n    I.InvoiceDate >= '2010-01-01'\nGROUP BY \n    I.InvoiceDate\nORDER BY \n    I.Inv
oiceDate;", '  \n     Find the top 5 customer who bought the most albums in total quantity (across all invo
ices):\n']
```sql
SELECT
    C.FirstName || ' ' || C.LastName AS CustomerName,
    SUM(IL.Quantity) AS TotalAlbumsPurchased
FROM
    Customer AS C
JOIN
    Invoice AS I ON C.CustomerId = I.CustomerId
JOIN
    InvoiceLine AS IL ON I.InvoiceId = IL.InvoiceId
WHERE
    IL.TrackId IN (
        SELECT
            TrackId
        FROM
            Track
        WHERE
            AlbumId IS NOT NULL
    )
GROUP BY
    C.CustomerId
ORDER BY
    TotalAlbumsPurchased DESC
```
```

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

```
        WHERE
            AlbumId IS NOT NULL
    )
GROUP BY
    C.CustomerId
ORDER BY
    TotalAlbumsPurchased DESC
LIMIT
    5;
      CustomerName  TotalAlbumsPurchased
0     Manoj Pareek                    38
1       Luis Rojas                    38
2  Diego Gutiérrez                    38
3      Mark Taylor                    38
4     Steve Murray                    38
```


Top 5 Customers by Total Albums Purchased
Total Albums Purchased
40
35
30
25
20
15
10
5
0
Manoj Pareek
Luis Rojas
Diego Gutiérrez
Mark Taylor
Steve Murray
Customer Name

Out[39]:
```
("SELECT \n    C.FirstName || ' ' || C.LastName AS CustomerName,\n    SUM(IL.Quantity) AS TotalAlbumsPurch
ased\nFROM \n    Customer AS C\nJOIN \n    Invoice AS I ON C.CustomerId = I.CustomerId\nJOIN \n    Invoice
Line AS IL ON I.InvoiceId = IL.InvoiceId\nWHERE \n    IL.TrackId IN (\n        SELECT \n            TrackI
d\n        FROM \n            Track\n        WHERE \n            AlbumId IS NOT NULL\n    )\nGROUP BY \n
C.CustomerId\nORDER BY \n    TotalAlbumsPurchased DESC\nLIMIT \n    5;",
       CustomerName  TotalAlbumsPurchased
 0     Manoj Pareek                   38
 1       Luis Rojas                   38
 2  Diego Gutiérrez                   38
 3      Mark Taylor                   38
 4     Steve Murray                   38,
 Figure({
     'data': [{'type': 'bar',
               'x': array(['Manoj Pareek', 'Luis Rojas', 'Diego Gutiérrez', 'Mark Taylor',
                           'Steve Murray'], dtype=object),
               'y': array([38, 38, 38, 38, 38])}],
     'layout': {'template': '...',
                'title': {'text': 'Top 5 Customers by Total Albums Purchased'},
                'xaxis': {'title': {'text': 'Customer Name'}},
                'yaxis': {'title': {'text': 'Total Albums Purchased'}}}
 }))
```

In [40]:
```
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   InvoiceDate DATETIME NOT NULL,\n   BillingAddress NVARCHAR(70),\n   BillingCity NVARCHAR(40),\n   BillingState NVARCHAR(40),\n   BillingCountry NVARCHAR(40),\n   BillingPostalCode NVARCHAR(10),\n   Total NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Invoice PRIMARY KEY (InvoiceId),\n   FOREIGN KEY (CustomerId) REFERENCES Customer (CustomerId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE InvoiceLine\n(\n   InvoiceLineId INTEGER NOT NULL,\n   InvoiceId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   Quantity INTEGER NOT NULL,\n   CONSTRAINT PK_InvoiceLine PRIMARY KEY (InvoiceLineId),\n   FOREIGN KEY (InvoiceId) REFERENCES Invoice (InvoiceId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Customer\n(\n   CustomerId INTEGER NOT NULL,\n   FirstName NVARCHAR(40) NOT NULL,\n   LastName NVARCHAR(20) NOT NULL,\n   Company NVARCHAR(80),\n   Address NVARCHAR(70),\n   City NVARCHAR(40),\n   State NVARCHAR(40),\n   Country NVARCHAR(40),\n   PostalCode NVARCHAR(10),\n   Phone NVARCHAR(24),\n   Fax NVARCHAR(24),\n   Email NVARCHAR(60) NOT NULL,\n   SupportRepId INTEGER,\n   CONSTRAINT PK_Customer PRIMARY KEY (CustomerId),\n   FOREIGN KEY (SupportRepId) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_CustomerSupportRepId ON Customer (SupportRepId)\n\nCREATE TABLE Track\n(\n   TrackId INTEGER NOT NULL,\n   Name NVARCHAR(200) NOT NULL,\n   AlbumId INTEGER,\n   MediaTypeId INTEGER NOT NULL,\n   GenreId INTEGER,\n   Composer NVARCHAR(220),\n   Milliseconds INTEGER NOT NULL,\n   Bytes INTEGER,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Track PRIMARY KEY (TrackId),\n   FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceCustomerId ON Invoice (CustomerId)\n\nCREATE INDEX IFK_EmployeeReportsTo ON Employee (ReportsTo)\n\nCREATE TABLE PlaylistTrack\n(\n   PlaylistId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   CONSTRAINT PK_PlaylistTrack PRIMARY KEY (PlaylistId, TrackId),\n   FOREIGN KEY (PlaylistId) REFERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE Employee\n(\n   EmployeeId INTEGER NOT NULL,\n   LastName NVARCHAR(20) NOT NULL,\n   FirstName NVARCHAR(20) NOT NULL,\n   Title NVARCHAR(30),\n   ReportsTo INTEGER,\n   BirthDate DATETIME,\n   HireDate DATETIME,\n   Address NVARCHAR(70),\n   City NVARCHAR(40),\n   State NVARCHAR(40),\n   Country NVARCHAR(40),\n   PostalCode NVARCHAR(10),\n   Phone NVARCHAR(24),\n   Fax NVARCHAR(24),\n   Email NVARCHAR(60),\n   CONSTRAINT PK_Employee PRIMARY KEY (EmployeeId),\n   FOREIGN KEY (ReportsTo) REFERENCES Employee (EmployeeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceLineTrackId ON InvoiceLine (TrackId)\n\n\n===Additional Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficient, please generate a valid SQL query without any explanations for the question. \n2. If the provided context is almost sufficient but requires knowledge of a specific string in a particular column, please generate an intermediate SQL query to find the distinct strings in that column. Prepend the query with a comment saying intermediate_sql \n3. If the provided context is insufficient, please explain why it can't be generated. \n4. Please use the most relevant table(s). \n5. If the question has been asked and answered before, please repeat the answer exactly as it was given before. \n", ' \n   Find the top 5 customer who bought the most albums in total quantity

```
(across all invoices):\n', "SELECT \n    C.FirstName || ' ' || C.LastName AS CustomerName,\n    SUM(IL.Quan
tity) AS TotalAlbumsPurchased\nFROM \n    Customer AS C\nJOIN \n    Invoice AS I ON C.CustomerId = I.Custom
erId\nJOIN \n    InvoiceLine AS IL ON I.InvoiceId = IL.InvoiceId\nWHERE \n    IL.TrackId IN (\n        SELE
CT \n            TrackId\n        FROM \n            Track\n        WHERE \n            AlbumId IS NOT NULL
\n    )\nGROUP BY \n    C.CustomerId\nORDER BY \n    TotalAlbumsPurchased DESC\nLIMIT \n    5;", '  \n    F
ind the top 5 most expensive tracks (based on unit price):\n', 'SELECT \n    T.Name AS TrackName,\n    T.Un
itPrice AS UnitPrice\nFROM \n    Track AS T\nORDER BY \n    T.UnitPrice DESC\nLIMIT \n    5;', '  \n     Fi
nd the customer who bought the most albums in total quantity (across all invoices): \n', "SELECT \n    C.Fi
rstName || ' ' || C.LastName AS CustomerName,\n    SUM(IL.Quantity) AS TotalAlbumsPurchased\nFROM \n    Cus
tomer AS C\nJOIN \n    Invoice AS I ON C.CustomerId = I.CustomerId\nJOIN \n    InvoiceLine AS IL ON I.Invoi
ceId = IL.InvoiceId\nWHERE \n    IL.TrackId IN (\n        SELECT \n            TrackId\n        FROM \n
Track\n        WHERE \n            AlbumId IS NOT NULL\n    )\nGROUP BY \n    C.CustomerId\nORDER BY \n
TotalAlbumsPurchased DESC\nLIMIT \n    1;", '  \n     Find the customer with the most invoices \n', "SELECT
\n    C.FirstName || ' ' || C.LastName AS CustomerName,\n    COUNT(I.InvoiceId) AS TotalInvoices\nFROM \n
Customer AS C\nLEFT JOIN \n    Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n    C.CustomerId\nOR
DER BY \n    TotalInvoices DESC\nLIMIT \n    1;", 'what are the top 5 countries that customers come from?',
'SELECT \n    C.Country,\n    COUNT(*) AS CustomerCount\nFROM \n    Customer AS C\nGROUP BY \n    C.Country
\nORDER BY \n    CustomerCount DESC\nLIMIT \n    5;', '  \n    Get the average invoice total for each custo
mer:\n', "SELECT \n    C.FirstName || ' ' || C.LastName AS CustomerName,\n    AVG(I.Total) AS AverageInvoic
eTotal\nFROM \n    Customer AS C\nLEFT JOIN \n    Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n
C.CustomerId;", '  \n    Get the top 10 most popular artists (based on the number of tracks):\n', 'SELECT
\n    A.Name AS ArtistName,\n    COUNT(T.TrackId) AS TrackCount\nFROM \n    Artist AS A\nLEFT JOIN \n    Al
bum AS Al ON A.ArtistId = Al.ArtistId\nLEFT JOIN \n    Track AS T ON Al.AlbumId = T.AlbumId\nGROUP BY \n
A.ArtistId\nORDER BY \n    TrackCount DESC\nLIMIT \n    10;', '  \n    Get the total number of invoices for
each customer\n', "SELECT \n    C.FirstName || ' ' || C.LastName AS CustomerName,\n    COUNT(I.InvoiceId) A
S TotalInvoices\nFROM \n    Customer AS C\nLEFT JOIN \n    Invoice AS I ON C.CustomerId = I.CustomerId\nGRO
UP BY \n    C.CustomerId;", '  \n    Find the total number of invoices per country:\n', 'SELECT \n    C.Cou
ntry,\n    COUNT(I.InvoiceId) AS TotalInvoices\nFROM \n    Customer AS C\nLEFT JOIN \n    Invoice AS I ON
C.CustomerId = I.CustomerId\nGROUP BY \n    C.Country\nORDER BY \n    TotalInvoices DESC;', '  \n    List a
ll invoices with a total exceeding $10:\n', 'SELECT \n    *\nFROM \n    Invoice\nWHERE \n    Total > 10;',
'  \n     Find the top 3 customers who spent the most money overall:\n']
```sql
SELECT
    C.FirstName || ' ' || C.LastName AS CustomerName,
    SUM(I.Total) AS TotalAmountSpent
FROM
    Customer AS C
JOIN
    Invoice AS I ON C.CustomerId = I.CustomerId
GROUP BY
    C.CustomerId
ORDER BY
```
```

```
    TotalAmountSpent DESC
LIMIT
    3;
```
```
SELECT
    C.FirstName || ' ' || C.LastName AS CustomerName,
    SUM(I.Total) AS TotalAmountSpent
FROM
    Customer AS C
JOIN
    Invoice AS I ON C.CustomerId = I.CustomerId
GROUP BY
    C.CustomerId
ORDER BY
    TotalAmountSpent DESC
LIMIT
    3;
SELECT
    C.FirstName || ' ' || C.LastName AS CustomerName,
    SUM(I.Total) AS TotalAmountSpent
FROM
    Customer AS C
JOIN
    Invoice AS I ON C.CustomerId = I.CustomerId
GROUP BY
    C.CustomerId
ORDER BY
    TotalAmountSpent DESC
LIMIT
    3;
```
```
         CustomerName  TotalAmountSpent
0         Helena Holý             49.62
1  Richard Cunningham             47.62
2          Luis Rojas             46.62
```


Top 3 Customers by Total Amount Spent
TotalAmountSpent
50
40
30
20
10
0
Helena Holý
Richard Cunningham
Luis Rojas
CustomerName

Out[40]:
```
("SELECT \n    C.FirstName || ' ' || C.LastName AS CustomerName,\n    SUM(I.Total) AS TotalAmountSpent\nFR
OM \n    Customer AS C\nJOIN \n    Invoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n    C.CustomerI
d\nORDER BY \n    TotalAmountSpent DESC\nLIMIT \n    3;",
          CustomerName  TotalAmountSpent
 0         Helena Holý            49.62
 1  Richard Cunningham            47.62
 2          Luis Rojas            46.62,
 Figure({
     'data': [{'alignmentgroup': 'True',
               'hovertemplate': 'CustomerName=%{x}<br>TotalAmountSpent=%{y}<extra></extra>',
               'legendgroup': '',
               'marker': {'color': '#636efa', 'pattern': {'shape': ''}},
               'name': '',
               'offsetgroup': '',
               'orientation': 'v',
               'showlegend': False,
               'textposition': 'auto',
               'type': 'bar',
               'x': array(['Helena Holý', 'Richard Cunningham', 'Luis Rojas'], dtype=object),
               'xaxis': 'x',
               'y': array([49.62, 47.62, 46.62]),
               'yaxis': 'y'}],
     'layout': {'barmode': 'relative',
                'legend': {'tracegroupgap': 0},
                'template': '...',
                'title': {'text': 'Top 3 Customers by Total Amount Spent'},
                'xaxis': {'anchor': 'y',
                          'categoryorder': 'total descending',
                          'domain': [0.0, 1.0],
                          'title': {'text': 'CustomerName'}},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'TotalAmountSpent'}}}
 }))
```

In [41]:
```
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   PlaylistId INTEGER NOT NULL,\n   Name NVARCHAR(120),\n   CONSTRAINT PK_Playlist PRIMARY KEY (PlaylistId)\n)\n\nCREATE TABLE Track\n(\n   TrackId INTEGER NOT NULL,\n   Name NVARCHAR(200) NOT NULL,\n   AlbumId INTEGER,\n   MediaTypeId INTEGER NOT NULL,\n   GenreId INTEGER,\n   Composer NVARCHAR(220),\n   Milliseconds INTEGER NOT NULL,\n   Bytes INTEGER,\n   UnitPrice NUMERIC(10,2) NOT NULL,\n   CONSTRAINT PK_Track PRIMARY KEY (TrackId),\n   FOREIGN KEY (AlbumId) REFERENCES Album (AlbumId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (GenreId) REFERENCES Genre (GenreId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (MediaTypeId) REFERENCES MediaType (MediaTypeId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE TABLE PlaylistTrack\n(\n   PlaylistId INTEGER NOT NULL,\n   TrackId INTEGER NOT NULL,\n   CONSTRAINT PK_PlaylistTrack PRIMARY KEY (PlaylistId, TrackId),\n   FOREIGN KEY (PlaylistId) REFERENCES Playlist (PlaylistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION,\n   FOREIGN KEY (TrackId) REFERENCES Track (TrackId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_TrackGenreId ON Track (GenreId)\n\nCREATE INDEX IFK_TrackAlbumId ON Track (AlbumId)\n\nCREATE INDEX IFK_TrackMediaTypeId ON Track (MediaTypeId)\n\nCREATE INDEX IFK_AlbumArtistId ON Album (ArtistId)\n\nCREATE TABLE Album\n(\n   AlbumId INTEGER NOT NULL,\n   Title NVARCHAR(160) NOT NULL,\n   ArtistId INTEGER NOT NULL,\n   CONSTRAINT PK_Album PRIMARY KEY (AlbumId),\n   FOREIGN KEY (ArtistId) REFERENCES Artist (ArtistId) \n\t\tON DELETE NO ACTION ON UPDATE NO ACTION\n)\n\nCREATE INDEX IFK_InvoiceLineTrackId ON InvoiceLine (TrackId) \n\n\n===Additional Context \n\nIn the chinook database invoice means order\n\n===Response Guidelines \n1. If the provided context is sufficient, please generate a valid SQL query without any explanations for the question. \n2. If the provided context is almost sufficient but requires knowledge of a specific string in a particular column, please generate an intermediate SQL query to find the distinct strings in that column. Prepend the query with a comment saying intermediate_sql \n3. If the provided context is insufficient, please explain why it can't be generated. \n4. Please use the most relevant table(s). \n5. If the question has been asked and answered before, please repeat the answer exactly as it was given before. \n", ' \n   List all genres and the number of tracks in each genre:\n', 'SELECT \n   G.Name AS GenreName,\n   COUNT(T.TrackId) AS TrackCount\nFROM \n   Genre AS G\nLEFT JOIN \n   Track AS T ON G.GenreId = T.GenreId\nGROUP BY \n   G.GenreId\nORDER BY \n   TrackCount DESC;', ' \n   Get the top 10 most popular artists (based on the number of tracks):\n', 'SELECT \n   A.Name AS ArtistName,\n   COUNT(T.TrackId) AS TrackCount\nFROM \n   Artist AS A\nLEFT JOIN \n   Album AS Al ON A.ArtistId = Al.ArtistId\nLEFT JOIN \n   Track AS T ON Al.AlbumId = T.AlbumId\nGROUP BY \n   A.ArtistId\nORDER BY \n   TrackCount DESC\nLIMIT \n   10;', ' \n   Find the top 5 customer who bought the most albums in total quantity (across all invoices):\n', "SELECT \n   C.FirstName || ' ' || C.LastName AS CustomerName,\n   SUM(IL.Quantity) AS TotalAlbumsPurchased\nFROM \n   Customer AS C\nJOIN \n   Invoice AS I ON C.CustomerId = I.CustomerId\nJOIN \n   InvoiceLine AS IL ON I.InvoiceId = IL.InvoiceId\nWHERE \n   IL.TrackId IN (\n   SELECT \n   TrackId\n   FROM \n   Track\n   WHERE \n   AlbumId IS NOT NULL\n   )\nGROUP BY \n   C.CustomerId\nORDER BY \n   TotalAlbumsPurchased DESC\nLIMIT \n   5;", 'Show me a list of tables in the SQLite database', 'SELECT \n   T.Name\nFROM\n   Playlist AS T;', ' \n   Find the customer who bought the most albums in total quantity (across all invoices): \n', "SELECT \n   C.FirstName || ' ' || C.LastName AS CustomerName,\n   SUM(IL.Quantity) AS TotalAlbumsPurchased\nFROM \n   Customer AS C\nJOIN \n   Invoice AS I ON C.CustomerId = I.CustomerId\nJOIN \n   InvoiceLine AS IL ON I.InvoiceId = IL.InvoiceId\nWHERE \n   IL.TrackId IN (\n   SEL

```
ECT \n            TrackId\n        FROM \n            Track\n        WHERE \n            AlbumId IS NOT NUL
L\n    )\nGROUP BY \n    C.CustomerId\nORDER BY \n    TotalAlbumsPurchased DESC\nLIMIT \n    1;", '  \n
Find all tracks with a name containing "What" (case-insensitive)\n', "SELECT \n    T.Name AS TrackName\nFRO
M \n    Track AS T\nWHERE \n    UPPER(T.Name) LIKE '%WHAT%';", '  \n    List all albums and their correspon
ding artist names  \n', 'SELECT \n    A.Title AS AlbumTitle,\n    Ar.Name AS ArtistName\nFROM \n    Album A
S A\nJOIN \n    Artist AS Ar ON A.ArtistId = Ar.ArtistId;', '  \n    Find the top 5 most expensive tracks
(based on unit price):\n', 'SELECT \n    T.Name AS TrackName,\n    T.UnitPrice AS UnitPrice\nFROM \n    Tra
ck AS T\nORDER BY \n    T.UnitPrice DESC\nLIMIT \n    5;', '  \n    Find all invoices since 2010 and the to
tal amount invoiced:\n', "SELECT \n    I.InvoiceDate,\n    SUM(IL.UnitPrice * IL.Quantity) AS TotalAmount\n
FROM \n    Invoice AS I\nJOIN \n    InvoiceLine AS IL ON I.InvoiceId = IL.InvoiceId\nWHERE \n    I.InvoiceD
ate >= '2010-01-01'\nGROUP BY \n    I.InvoiceDate\nORDER BY \n    I.InvoiceDate;", '  \n     Find the top 3
customers who spent the most money overall:\n', "SELECT \n    C.FirstName || ' ' || C.LastName AS CustomerN
ame,\n    SUM(I.Total) AS TotalAmountSpent\nFROM \n    Customer AS C\nJOIN \n    Invoice AS I ON C.Customer
Id = I.CustomerId\nGROUP BY \n    C.CustomerId\nORDER BY \n    TotalAmountSpent DESC\nLIMIT \n    3;", '
\n     Get all playlists containing at least 10 tracks and the total duration of those tracks:\n']
```sql
SELECT
    P.Name AS PlaylistName,
    SUM(T.Milliseconds) AS TotalDuration
FROM
    Playlist AS P
JOIN
    PlaylistTrack AS PT ON P.PlaylistId = PT.PlaylistId
JOIN
    Track AS T ON PT.TrackId = T.TrackId
GROUP BY
    P.PlaylistId
HAVING
    COUNT(T.TrackId) >= 10
ORDER BY
    TotalDuration DESC;
```
SELECT
    P.Name AS PlaylistName,
    SUM(T.Milliseconds) AS TotalDuration
FROM
    Playlist AS P
JOIN
    PlaylistTrack AS PT ON P.PlaylistId = PT.PlaylistId
JOIN
    Track AS T ON PT.TrackId = T.TrackId
GROUP BY
```

```
    P.PlaylistId
HAVING
    COUNT(T.TrackId) >= 10
ORDER BY
    TotalDuration DESC;
SELECT
    P.Name AS PlaylistName,
    SUM(T.Milliseconds) AS TotalDuration
FROM
    Playlist AS P
JOIN
    PlaylistTrack AS PT ON P.PlaylistId = PT.PlaylistId
JOIN
    Track AS T ON PT.TrackId = T.TrackId
GROUP BY
    P.PlaylistId
HAVING
    COUNT(T.TrackId) >= 10
ORDER BY
    TotalDuration DESC;
                PlaylistName  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 Playlists with at Least 10 Tracks
Total Duration (milliseconds)
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
Playlist Name

Out[41]:
```
('SELECT \n    P.Name AS PlaylistName,\n    SUM(T.Milliseconds) AS TotalDuration\nFROM \n    Playlist AS P
\nJOIN \n    PlaylistTrack AS PT ON P.PlaylistId = PT.PlaylistId\nJOIN \n    Track AS T ON PT.TrackId = T.
TrackId\nGROUP BY \n    P.PlaylistId\nHAVING \n    COUNT(T.TrackId) >= 10\nORDER BY \n    TotalDuration DE
SC;',
                    PlaylistName  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': 'PlaylistName=%{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},
                'margin': {'t': 60},
                'template': '...',
                'title': {'text': 'Total Duration of Playlists with at Least 10 Tracks'},
                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0], 'title': {'text': 'Playlist Name'}},
```

```
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0], 'title': {'text': 'Total Duration (millisec
onds)'}}}
 }))
```

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   A.Name AS ArtistName,\n   COUNT(T.TrackId) AS TrackCount\nFROM \n   Artist AS A\nLEFT JOIN \n   Album AS Al ON A.ArtistId = Al.ArtistId\nLEFT JOIN \n   Track AS T ON Al.AlbumId = T.AlbumId\nGROUP BY \n   A.ArtistId\nORDER BY \n   TrackCount DESC\nLIMIT \n   10;', ' \n   List all a lbums and their corresponding artist names \n', 'SELECT \n   A.Title AS AlbumTitle,\n   Ar.Name AS Artis tName\nFROM \n   Album AS A\nJOIN \n   Artist AS Ar ON A.ArtistId = Ar.ArtistId;', ' \n   List all genr es and the number of tracks in each genre:\n', 'SELECT \n   G.Name AS GenreName,\n   COUNT(T.TrackId) AS TrackCount\nFROM \n   Genre AS G\nLEFT JOIN \n   Track AS T ON G.GenreId = T.GenreId\nGROUP BY \n   G.Ge nreId\nORDER BY \n   TrackCount DESC;', ' \n   Find the top 5 customer who bought the most albums in to tal quantity (across all invoices):\n', "SELECT \n   C.FirstName || ' ' || C.LastName AS CustomerName,\n SUM(IL.Quantity) AS TotalAlbumsPurchased\nFROM \n   Customer AS C\nJOIN \n   Invoice AS I ON C.CustomerId = I.CustomerId\nJOIN \n   InvoiceLine AS IL ON I.InvoiceId = IL.InvoiceId\nWHERE \n   IL.TrackId IN (\n SELECT \n   TrackId\n   FROM \n   Track\n   WHERE \n   AlbumId IS NOT NULL\n   )\nGROUP BY \n   C.CustomerId\nORDER BY \n   TotalAlbumsPurchased DESC\nLIMIT \n   5;", ' \n Find the customer who bought the most albums in total quantity (across all invoices): \n', "SELECT \n   C. FirstName || ' ' || C.LastName AS CustomerName,\n   SUM(IL.Quantity) AS TotalAlbumsPurchased\nFROM \n   C

ustomer AS C\nJOIN \n   Invoice AS I ON C.CustomerId = I.CustomerId\nJOIN \n   InvoiceLine AS IL ON I.Inv
oiceId = IL.InvoiceId\nWHERE \n   IL.TrackId IN (\n   SELECT \n   TrackId\n   FROM \n
Track\n   WHERE \n   AlbumId IS NOT NULL\n   )\nGROUP BY \n   C.CustomerId\nORDER BY \n
TotalAlbumsPurchased DESC\nLIMIT \n   1;", ' \n   Get all playlists containing at least 10 tracks and t
he total duration of those tracks:\n', 'SELECT \n   P.Name AS PlaylistName,\n   SUM(T.Milliseconds) AS To
talDuration\nFROM \n   Playlist AS P\nJOIN \n   PlaylistTrack AS PT ON P.PlaylistId = PT.PlaylistId\nJOIN
\n   Track AS T ON PT.TrackId = T.TrackId\nGROUP BY \n   P.PlaylistId\nHAVING \n   COUNT(T.TrackId) >= 1
0\nORDER BY \n   TotalDuration DESC;', ' \n   Find the top 5 most expensive tracks (based on unit pric
e):\n', 'SELECT \n   T.Name AS TrackName,\n   T.UnitPrice AS UnitPrice\nFROM \n   Track AS T\nORDER BY
\n   T.UnitPrice DESC\nLIMIT \n   5;', ' \n   Find all tracks with a name containing "What" (case-insen
sitive)\n', "SELECT \n   T.Name AS TrackName\nFROM \n   Track AS T\nWHERE \n   UPPER(T.Name) LIKE '%WHA
T%';", 'Show me a list of tables in the SQLite database', 'SELECT \n   T.Name\nFROM\n   Playlist AS T;',
' \n   Find the top 3 customers who spent the most money overall:\n', "SELECT \n   C.FirstName || ' ' |
| C.LastName AS CustomerName,\n   SUM(I.Total) AS TotalAmountSpent\nFROM \n   Customer AS C\nJOIN \n   I
nvoice AS I ON C.CustomerId = I.CustomerId\nGROUP BY \n   C.CustomerId\nORDER BY \n   TotalAmountSpent DE
SC\nLIMIT \n   3;", ' \n   Identify artists who have albums with tracks appearing in multiple genre
s:\n']

```sql
SELECT DISTINCT
    A.ArtistId,
    A.Name AS ArtistName
FROM
    Artist AS A
JOIN
    Album AS Al ON A.ArtistId = Al.ArtistId
JOIN
    Track AS T ON Al.AlbumId = T.AlbumId
GROUP BY
    A.ArtistId, T.GenreId
HAVING
    COUNT(T.GenreId) > 1;
```

```
SELECT DISTINCT
    A.ArtistId,
    A.Name AS ArtistName
FROM
    Artist AS A
JOIN
    Album AS Al ON A.ArtistId = Al.ArtistId
JOIN
    Track AS T ON Al.AlbumId = T.AlbumId
GROUP BY
```

```
    A.ArtistId, T.GenreId
HAVING
    COUNT(T.GenreId) > 1;
SELECT DISTINCT
    A.ArtistId,
    A.Name AS ArtistName
FROM
    Artist AS A
JOIN
    Album AS Al ON A.ArtistId = Al.ArtistId
JOIN
    Track AS T ON Al.AlbumId = T.AlbumId
GROUP BY
    A.ArtistId, T.GenreId
HAVING
    COUNT(T.GenreId) > 1;
```

	ArtistId	ArtistName
0	1	AC/DC
1	2	Accept
2	3	Aerosmith
3	4	Alanis Morissette
4	5	Alice In Chains
..	...	...
131	226	Eugene Ormandy
132	245	Michael Tilson Thomas & San Francisco Symphony
133	247	The King's Singers
134	248	Berliner Philharmoniker & Herbert Von Karajan
135	252	Amy Winehouse

[136 rows x 2 columns]

Out[42]:
```
('SELECT DISTINCT\n    A.ArtistId,\n    A.Name AS ArtistName\nFROM \n    Artist AS A\nJOIN \n    Album AS
Al ON A.ArtistId = Al.ArtistId\nJOIN \n    Track AS T ON Al.AlbumId = T.AlbumId\nGROUP BY \n    A.ArtistI
d, T.GenreId\nHAVING \n    COUNT(T.GenreId) > 1;',
      ArtistId                                 ArtistName
 0           1                                      AC/DC
 1           2                                     Accept
 2           3                                   Aerosmith
 3           4                             Alanis Morissette
 4           5                               Alice In Chains
 ..        ...                                        ...
 131       226                               Eugene Ormandy
 132       245  Michael Tilson Thomas & San Francisco Symphony
 133       247                             The King's Singers
 134       248   Berliner Philharmoniker & Herbert Von Karajan
 135       252                                Amy Winehouse

 [136 rows x 2 columns],
 Figure({
     'data': [{'delta': {'reference': 1},
               'mode': 'number+delta',
               'title': {'text': 'Artists with Multigenre Albums'},
               'type': 'indicator',
               'value': 136}],
     'layout': {'height': 100,
                'template': '...',
                'xaxis': {'anchor': 'y', 'domain': [0.0, 1.0]},
                'yaxis': {'anchor': 'x', 'domain': [0.0, 1.0]}}
 }))
```

Check completion time

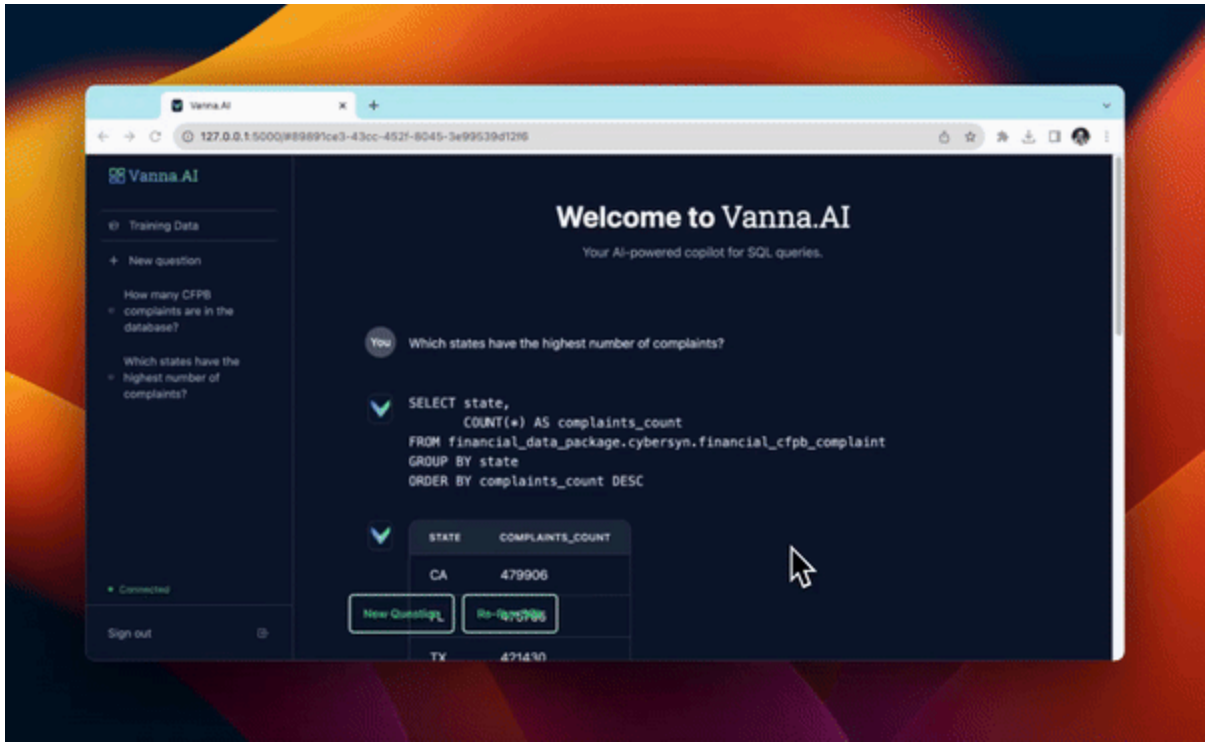
In [43]:
```
ts_stop = time()

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-pro' LLM took : 110.20 sec
```

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

```
2024-06-21 00:26:10.531788
```

Launch the User Interface



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

Next Steps

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

- Streamlit app
- Flask app
- Slackbot