databricks2-weaviate-ml-linkedin-job-postings

(https://databricks.com)

`Manan Kalra` - `Machine Learning Engineer, Genesys`

`manankalra29@gmail.com` - `https://linkedin.com/in/manankalra`

- ## Dataset
  https://www.kaggle.com/datasets/arshkon/linkedin-job-postings
- ## Description
  The following pipeline allows you to ask exploratory questions related to thousands of job postings that were advertised on LinkedIn in the past two years and returns a succinct response in relation to the same.
- ## How it works?
  - After preprocessing the data, this workflow uses **Weaviate** to create vector embeddings for the job postings.
    - Embeddings are stored in **Weaviate Cloud** for quick access at any time.
    - **Cohere**'s multilingual model is used to create embeddings. For generative tasks, Anthropic's Claude-**Sonnet** is used.
    - Both are being accessed via **Bedrock**.
    - A collection named **JobPostings** was populated with the embeddings. The vector indices are created using **HNSW**.
    - Utility functiona are included to perform **hybrid search** either with a single query or **generative search** along with a prompt.
  - It allows the user to answer questions like:
    - **Which location has the most AI/ML jobs?**

      Example response:

      Based on the data provided, the location that has the most AI/ML job postings is the United States. Several job postings list the location as "United States" or specific cities/regions within the United States such as New York City Metropolitan Area, Sunnyvale CA, Greater Chicago Area, Waltham MA, Bentonville AR, San Jose CA, and Los Angeles CA.

    - **What is the average salary for the above included jobs? Also, which is the most frequent title for jobs in this domain?**

      Example response:

      Based on the data provided, here are the answers to your questions:

      1. The average salary range for the included jobs is difficult to calculate precisely since many of the salary listings are incomplete or show wide ranges. However, based on the available data, a rough estimate of the average salary range would be around $200,000 - $300,000 per year.
      2. The most frequent job title in this domain, based on the data provided, is "Family Practice-Without OB Physician". This title appears 9 times across the different job listings.

      Please note that these answers are based solely on the data you provided, and may not represent the complete picture of salaries and job titles in this domain.

    - **Are there any healthcare jobs in California? If yes, which city are they based in? Are any of them based in Irvine?**

      Example response:

      Based on the data provided, there are several healthcare jobs located in California, specifically:
      1. Telemedicine Physician - CA 100% Remote (Irvine, CA)
      2. Internal Medicine Physician - $125/hourly - $135/hourly (San Mateo, CA)
      3. Family Practice-Without OB Physician - $125/hourly - $135/hourly (San Mateo, CA)

4. Family Practice-Without OB Physician - $133/yearly (Sacramento, CA)
5. Family Practice-Without OB Physician - $275,000/yearly - $325,000/yearly (San Jose, CA)
6. Medical Doctor (Silicon Valley, CA)

So yes, there are healthcare jobs listed in the data that are based in Irvine, California specifically for the role of "Telemedicine Physician - CA 100% Remote".

- **Limitations**
  - Doesn't use any quantization techniques which could have been useful to compress the high-dimenstional embeddings.
  - Options to improve embeddgins aweren't explored. No named vectors were used as well.
  - HNSW uses the default params.
  - No multi-tenancy options are enabled for scalability.

```
!pip3 install weaviate-client
```

```
a1e1/protobuf-5.27.1-cp38-abi3-manylinux2014_x86_64.whl (309 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 309.2/309.2 kB 74.1 MB/s eta 0:00:00
Requirement already satisfied: setuptools in /local_disk0/.ephemeral_nfs/cluster_libraries/python/lib/python3.1
0/site-packages (from grpcio-tools<2.0.0,>=1.57.0->weaviate-client) (65.5.1)
Requirement already satisfied: certifi in /databricks/python3/lib/python3.10/site-packages (from httpx<=0.27.0,
>=0.25.0->weaviate-client) (2022.12.7)
Collecting httpcore==1.*
  Downloading https://purecloud.jfrog.io/purecloud/api/pypi/pypi/packages/packages/78/d4/e5d7e4f2174f8a4d63c889
7d79eb8fe2503f7ecc03282fee1fa2719c2704/httpcore-1.0.5-py3-none-any.whl (https://purecloud.jfrog.io/purecloud/ap
i/pypi/pypi/packages/packages/78/d4/e5d7e4f2174f8a4d63c8897d79eb8fe2503f7ecc03282fee1fa2719c2704/httpcore-1.0.5
-py3-none-any.whl) (77 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 77.9/77.9 kB 41.2 MB/s eta 0:00:00
Requirement already satisfied: sniffio in /databricks/python3/lib/python3.10/site-packages (from httpx<=0.27.0,
>=0.25.0->weaviate-client) (1.2.0)
Requirement already satisfied: idna in /databricks/python3/lib/python3.10/site-packages (from httpx<=0.27.0,>=
0.25.0->weaviate-client) (3.4)
Requirement already satisfied: anyio in /databricks/python3/lib/python3.10/site-packages (from httpx<=0.27.0,>=
0.25.0->weaviate-client) (3.5.0)
Collecting h11<0.15,>=0.13
  Downloading https://purecloud.jfrog.io/purecloud/api/pypi/pypi/packages/packages/95/04/ff642e65ad6b90db43e668
d70ffb6736436c7ce41fcc549f4e9472234127/h11-0.14.0-py3-none-any.whl (https://purecloud.jfrog.io/purecloud/api/py
```

```
# imports

import os
import json
import pandas as pd

import weaviate
from weaviate.auth import AuthApiKey
from weaviate.config import AdditionalConfig, Timeout
from weaviate.classes.config import Configure, DataType, Property
from weaviate.classes.query import Metrics

from pyspark.sql.types import ArrayType, FloatType
from pyspark.sql.functions import col, concat_ws, count, date_format, from_unixtime, isnan, isnull, round, udf, whe
```

```
# load data

# using databricks here, if you need to re-run in Colab - please use a Google Drive mount instead for quick ingesti
postings_csv = "/dbfs/FileStore/tables/manan_dfs/wml/postings.csv"

# directly reading with spark gave me schema issues, this is juts a quick hack instead of creating the expected sch
postings = spark.createDataFrame(pd.read_csv(postings_csv))

display(postings)
```

```
/databricks/spark/python/pyspark/sql/pandas/utils.py:43: DeprecationWarning: distutils Version classes are deprecat
ed. Use packaging.version instead.
  if LooseVersion(pandas.__version__) < LooseVersion(minimum_pandas_version):
```

```
/databricks/spark/python/pyspark/sql/pandas/utils.py:77: DeprecationWarning: distutils Version classes are deprecat
ed. Use packaging.version instead.
  if LooseVersion(pyarrow.__version__) < LooseVersion(minimum_pyarrow_version):
<frozen importlib._bootstrap>:914: ImportWarning: ImportHookFinder.find_spec() not found; falling back to find_modu
le()
```

Table | New result table: ON

	job_id	company_name	title
1	921716	Corcoran Sawyer Smith	Marketing Coordinator
2	1829192	null	Mental Health Therapist/Counselor
3	10998357	The National Exemplar	Assitant Restaurant Manager
4	23221523	Abrams Fensterman, LLP	Senior Elder Law / Trusts and Estate
5	35982263	null	Service Technician
6	91700727	Downtown Raleigh Alliance	Economic Development and Plannin
7	103254301	Raw Cereal	Producer
8	112576855	null	Building Engineer
9	1218575	Children's Nebraska	Respiratory Therapist
10	2264355	Bay West Church	Worship Leader
11	9615617	Glastender, Inc.	Inside Customer Service Associate
12	11009123	PGAV Destinations	Project Architect
13	56482768	null	Appalachian Highlands Women's Bu
14	56924323	null	Structural Engineer

658+ rows | Truncated data due to byte limit

```
# looking here for potential missing values in job titles, descriptions, etc. to replace them with empty strings

# this is only to get some insights

# for example: >75% of the rows are missing salary related information
# so generative queries like  "What is the verage salary for a Machine Learning Engineer in Ney York?"" on a respon
# however, I've encountered some hallucinations

postings_rows = postings.count()

postings_missing = postings.select([
    round((count(when(isnull(c), c)) / postings_rows * 100), 2).alias(c) for c in postings.columns
])

postings_missing = postings_missing.toPandas().transpose().reset_index()
postings_missing.columns = ["feature", "missing_pct"]
postings_missing = spark.createDataFrame(postings_missing)

display(postings_missing)
```

```
/usr/lib/python3.10/socket.py:776: ResourceWarning: unclosed <socket.socket fd=68, family=AddressFamily.AF_INET, ty
pe=SocketKind.SOCK_STREAM, proto=6, laddr=('127.0.0.1', 47842), raddr=('127.0.0.1', 40813)>
  self._sock = None
ResourceWarning: Enable tracemalloc to get the object allocation traceback
/databricks/spark/python/pyspark/sql/pandas/conversion.py:161: DeprecationWarning: distutils Version classes are de
precated. Use packaging.version instead.
  if LooseVersion(pa.__version__) >= LooseVersion("13.0.0"):
/databricks/spark/python/pyspark/sql/pandas/utils.py:43: DeprecationWarning: distutils Version classes are deprecat
ed. Use packaging.version instead.
  if LooseVersion(pandas.__version__) < LooseVersion(minimum_pandas_version):
/databricks/spark/python/pyspark/sql/pandas/utils.py:77: DeprecationWarning: distutils Version classes are deprecat
ed. Use packaging.version instead.
  if LooseVersion(pyarrow.__version__) < LooseVersion(minimum_pyarrow_version):
```

Table | missing | New result table: ON

	feature	missing_pct
1	job_id	0
2	company_name	1.39
3	title	0
4	description	0.01

5	max_salary	75.94
6	pay_period	70.87
7	location	0
8	company_id	1.39
9	views	1.36
10	med_salary	94.93
11	min_salary	75.94
12	formatted_work_type	0
13	applies	81.17
14	original_listed_time	0
15	remote_allowed	87.69

28 rows

```
# filling missing values

fill_values = {
    "company_name": " ",
    "description": " ",
    "description": " ",
    "formatted_experience_level": " ",
    "max_salary": " ",
    "min_salary": " ",
    "pay_period": " "
}

postings_filled = postings.fillna(fill_values)
```

```
# formatting date, might be helpful for embeddings

postings_filled = postings_filled.withColumn("formatted_date", from_unixtime(col("original_listed_time") / 1000).ca
postings_filled = postings_filled.withColumn("formatted_date", date_format(col("formatted_date"), "MMMM d, y - EEEE

display(postings_filled.groupBy("formatted_date").count())
```

Table | Visualization 1 — New result table: ON

	formatted_date	count
1	February 19, 2024 - Monday	2
2	April 16, 2024 - Tuesday	4328
3	March 13, 2024 - Wednesday	5
4	December 21, 2023 - Thursday	1
5	December 8, 2023 - Friday	1
6	January 24, 2024 - Wednesday	1
7	April 18, 2024 - Thursday	33603
8	March 22, 2024 - Friday	40
9	March 7, 2024 - Thursday	1
10	April 8, 2024 - Monday	1243
11	March 31, 2024 - Sunday	2
12	April 3, 2024 - Wednesday	51
13	March 23, 2024 - Saturday	265
14	April 15, 2024 - Monday	7666
15	February 14, 2024 - Wednesday	1

61 rows

```
# using sentence-transformers, was trying to create my own vectors for a single column created by joining all featu
# but later came across the built-in way to do it with Weaviate

# postings_filled = postings_filled.withColumn(
#   "posting_to_embed",
#   concat_ws(
#     " ",
#     col("title"),
#     col("company_name"),
#     col("description"),
#     col("location"),
#     col("formatted_experience_level"),
#     col("max_salary"),
#     col("min_salary"),
#     col("pay_period"),
#     col("formatted_work_type"),
#     col("formatted_date"),
#     col("job_posting_url"),
#   )
# )

# display(postings_filled.select("posting_to_embed"))
```

```
# converting to json for ingesting objects in batches
# there might be a way to do it with the dataframe too, haven't searched more in detail as I was familiar with this

postings_json = postings_filled.select(
  "title",
  "company_name",
  "description",
  "location",
  "formatted_experience_level",
  "max_salary",
  "min_salary",
  "pay_period",
  "formatted_work_type",
  "formatted_date",
  "job_posting_url",
).toJSON().collect()

postings_json = [json.loads(job_posting) for job_posting in postings_json]
```

```
/usr/lib/python3.10/socket.py:776: ResourceWarning: unclosed <socket.socket fd=69, family=AddressFamily.AF_INET, ty
pe=SocketKind.SOCK_STREAM, proto=6, laddr=('127.0.0.1', 58302), raddr=('127.0.0.1', 34187)>
  self._sock = None
ResourceWarning: Enable tracemalloc to get the object allocation traceback
```

```
# using weaviate cloud here, using the 14 day free-trial for now though
# prevents me creating a local backup or in S3

WCS_URL = "<use-your-own>"
WEAVIATE_API_KEY = "<use-your-own>"

# resorted to using Bedrock, as I've reached spending limits here
# OPENAI_API_KEY = "<use-your-own>"
# COHERE_API_KEY = "<use-your-own>"

headers = {
    "X-AWS-Access-Key": "<use-your-own>",
    "X-AWS-Secret-Key": "<use-your-own>",
    "X-AWS-Session-Token": "<use-your-own>"  # using this as I've temp creds, ref: https://github.com/weaviate/weav
}

# client.close()
client = weaviate.connect_to_wcs(
    cluster_url=WCS_URL,
    auth_credentials=AuthApiKey(WEAVIATE_API_KEY),
    headers=headers,
    additional_config=AdditionalConfig(
        timeout=Timeout(init=2, query=120, insert=120)  # embedded the entire dataset, sometimes it lead to timeout
    )
)

print(client.is_ready(), json.dumps(client.get_meta()))
```


```
True {"hostname": "http://[::]:8080 (http://[::]:8080)", "modules": {"generative-anyscale": {"documentationHre
f": "https://docs.anyscale.com/endpoints/overview (https://docs.anyscale.com/endpoints/overview)", "name": "Gen
erative Search - Anyscale"}, "generative-aws": {"documentationHref": "https://docs.aws.amazon.com/bedrock/lates
t/APIReference/welcome.html (https://docs.aws.amazon.com/bedrock/latest/APIReference/welcome.html)", "name": "G
enerative Search - AWS"}, "generative-cohere": {"documentationHref": "https://docs.cohere.com/reference/chat (h
ttps://docs.cohere.com/reference/chat)", "name": "Generative Search - Cohere"}, "generative-mistral": {"documen
tationHref": "https://docs.mistral.ai/api/ (https://docs.mistral.ai/api/)", "name": "Generative Search - Mistra
l"}, "generative-openai": {"documentationHref": "https://platform.openai.com/docs/api-reference/completions (ht
tps://platform.openai.com/docs/api-reference/completions)", "name": "Generative Search - OpenAI"}, "generative-
palm": {"documentationHref": "https://cloud.google.com/vertex-ai/docs/generative-ai/chat/test-chat-prompts (htt
ps://cloud.google.com/vertex-ai/docs/generative-ai/chat/test-chat-prompts)", "name": "Generative Search - Googl
e PaLM"}, "multi2vec-palm": {"documentationHref": "https://cloud.google.com/vertex-ai/generative-ai/docs/embedd
ings/get-multimodal-embeddings (https://cloud.google.com/vertex-ai/generative-ai/docs/embeddings/get-multimodal
-embeddings)", "name": "Google PaLM Multimodal Module"}, "qna-openai": {"documentationHref": "https://platform.
openai.com/docs/api-reference/completions (https://platform.openai.com/docs/api-reference/completions)", "nam
e": "OpenAI Question & Answering Module"}, "ref2vec-centroid": {}, "reranker-cohere": {"documentationHref": "ht
tps://txt.cohere.com/rerank/ (https://txt.cohere.com/rerank/)", "name": "Reranker - Cohere"}, "reranker-voyagea
i": {"documentationHref": "https://docs.voyageai.com/reference/reranker-api (https://docs.voyageai.com/referenc
e/reranker-api)", "name": "Reranker - VoyageAI"}, "text2vec-aws": {"documentationHref": "https://docs.aws.amazo
n.com/bedrock/latest/userguide/titan-embedding-models.html (https://docs.aws.amazon.com/bedrock/latest/userguid
e/titan-embedding-models.html)", "name": "AWS Module"}, "text2vec-cohere": {"documentationHref": "https://docs.
```

```
# deleting the exisitng collection if it exists already, to create a new one later
# if client.collections.exists("JobPostings"):
#   client.collections.delete("JobPostings")

client.collections.create(
  "JobPostings",
  vectorizer_config=Configure.Vectorizer.text2vec_aws(
    region="us-east-1",
    service="bedrock",
    model="cohere.embed-multilingual-v3",  # not sure if all the jobs are in English, so using the multilingual mod
  ),
  # [
  #   # using named vectors was exponentially increasing the number of dimensions and was taking a lot of time too
  #   # I also noticed that named vectors do not seem to work with vector_index_config
  #
  #   Configure.NamedVectors.text2vec_aws(
  #     name="title",
  #     region="us-east-1",
  #     source_properties=["title"],
  #     service="bedrock",
  #     model="cohere.embed-multilingual-v3",
  #   ),
  #   Configure.NamedVectors.text2vec_aws(
  #     name="description",
  #     region="us-east-1",
  #     source_properties=["description"],
  #     service="bedrock",
  #     model="cohere.embed-multilingual-v3",
  #   ),
  # ],
  vector_index_config=Configure.VectorIndex.hnsw(),  # recently learned about this in the deeplearning.ai + weaviat
  generative_config=Configure.Generative.aws(  # to allow the user to ask questions based on the search responses r
    region="us-east-1",
    model="anthropic.claude-3-sonnet-20240229-v1:0"
  )
)
```

```
<weaviate.collections.collection.Collection at 0x7ff5a3e33b20>
```

```
# ingestion

job_postings_collection = client.collections.get("JobPostings")

# no need to re-run with weavaite cloud, took 1-2 hours at first

with job_postings_collection.batch.dynamic() as batch:
  for job_posting in postings_json:
    batch.add_object(
      properties=job_posting,
    )
```

```
/databricks/python_shell/dbruntime/safe_oinspect.py:133: DeprecationWarning: `getargspec` function is deprecated as
of IPython 7.10and will be removed in future versions.
  argspec = oinspect.getargspec(obj)
/databricks/python_shell/dbruntime/safe_oinspect.py:133: DeprecationWarning: `getargspec` function is deprecated as
of IPython 7.10and will be removed in future versions.
  argspec = oinspect.getargspec(obj)
/databricks/python_shell/dbruntime/safe_oinspect.py:133: DeprecationWarning: `getargspec` function is deprecated as
of IPython 7.10and will be removed in future versions.
  argspec = oinspect.getargspec(obj)
/databricks/python_shell/dbruntime/safe_oinspect.py:133: DeprecationWarning: `getargspec` function is deprecated as
of IPython 7.10and will be removed in future versions.
  argspec = oinspect.getargspec(obj)
```

Utils

```
count = job_postings_collection.aggregate.over_all(total_count=True)
print("Total items: ", count)

top_job_title = job_postings_collection.aggregate.over_all(
  return_metrics=Metrics("title").text(
    top_occurrences_count=True,
    top_occurrences_value=True,
    min_occurrences=2,
  )
)
print("Top job title: ", top_job_title)
```

```
Total items:  AggregateReturn(properties={}, total_count=23481)
Top job title:  AggregateReturn(properties={'title': AggregateText(count=None, top_occurrences=[TopOccurrence(count
=109, value='Administrative Assistant'), TopOccurrence(count=105, value='Project Manager')])}, total_count=23481)
```

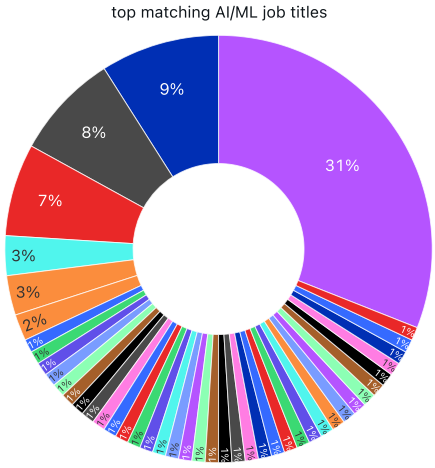
```
def query_job_postings(collection_name: str, query: str, prompt: str=None, limit: int=50) -> str:
  """
  Performs either hybrid or generative search based on user request.

  Parameters
  ----------
  collection name : str
    Name of the collection

  query : str
    User query keywords.

  prompt : str
    Additonal question that the user can ask on top of the search responses.

  limit : int
    Number of search responses to return.

  Returns
  -------
  str
    Either the search results or the generated response by the LLM.

  """
  collection = client.collections.get(collection_name)

  if prompt:  # if the user included a prompt in the request, do a generative search to answer the question based o
    response = collection.generate.hybrid(
      query=query,
      query_properties=["title", "location"],  # fields to use for keyword search
      grouped_task=prompt + " Only answer this based on the data provided above.",  # generation is performed on th
      # grouped_properties=["title"],
      limit=limit,
      alpha=0.75,  # want to favour the vector search more here
    )
  else:
    response = collection.query.hybrid(
      query=query,
      query_properties=["title", "location"],
      limit=limit,
      alpha=0.75,
    )

  return response
```

Results

```
# Example 1 - Semantic Search

response = query_job_postings(collection_name="JobPostings", query="Machine Learning Engineer, Software Engineer, D

res = []
for res_job in response.objects:
  res.append([res_job.properties["title"], res_job.properties["company_name"], res_job.properties["formatted_date"]

display(pd.DataFrame(res, columns=["title", "company_name", "date"]))
```

```
/databricks/spark/python/pyspark/sql/pandas/utils.py:43: DeprecationWarning: distutils Version classes are deprecat
ed. Use packaging.version instead.
  if LooseVersion(pandas.__version__) < LooseVersion(minimum_pandas_version):
/databricks/spark/python/pyspark/sql/pandas/utils.py:77: DeprecationWarning: distutils Version classes are deprecat
ed. Use packaging.version instead.
  if LooseVersion(pyarrow.__version__) < LooseVersion(minimum_pyarrow_version):
```

Table | Visualization 1

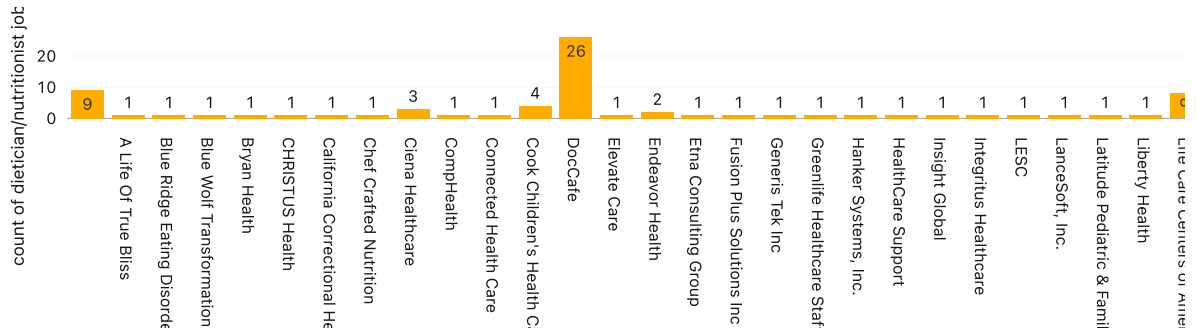


100 rows

```
# Example 2 - Semantic Search

response = query_job_postings(collection_name="JobPostings", query="Dietician, Nutritionist", limit=100)

res = []
for res_job in response.objects:
  res.append([res_job.properties["title"], res_job.properties["company_name"], res_job.properties["location"]])

display(pd.DataFrame(res, columns=["title", "company_name", "location"]))
```

```
/databricks/spark/python/pyspark/sql/pandas/utils.py:43: DeprecationWarning: distutils Version classes are deprecat
ed. Use packaging.version instead.
  if LooseVersion(pandas.__version__) < LooseVersion(minimum_pandas_version):
/databricks/spark/python/pyspark/sql/pandas/utils.py:77: DeprecationWarning: distutils Version classes are deprecat
ed. Use packaging.version instead.
  if LooseVersion(pyarrow.__version__) < LooseVersion(minimum_pyarrow_version):
```

Table | Visualization 2 | Visualization 3



company_name
Family Clinic
fing
are System
alth Care Services
er Care Center

100 rows

```
# Example 3 - Generative Search

response = query_job_postings(
  collection_name="JobPostings",
  query="Machine Learning Engineer, Data Scientist, Artificial Intelligence, Applied Scientist",
  prompt="Which location has the most AI/ML jobs?",
  limit=25
)

print(response.generated)
```

```
Based on the data provided, the location that has the most AI/ML job postings is the United States. Several job pos
tings list the location as "United States" or specific cities/regions within the United States such as New York Cit
y Metropolitan Area, Sunnyvale CA, Greater Chicago Area, Waltham MA, Bentonville AR, San Jose CA, and Los Angeles C
A.
```

```
# Example 4 - Generative Search

response = query_job_postings(
  collection_name="JobPostings",
  query="Healthcare, Medicine",
  prompt="What is the average salary for the above included jobs? Also, which is the most frequent title for jobs i
  limit=50
)

print(response.generated)
```

```
Based on the data provided, here are the answers to your questions:

1. The average salary range for the included jobs is difficult to calculate precisely since many of the salary list
ings are incomplete or show wide ranges. However, based on the available data, a rough estimate of the average sala
ry range would be around $200,000 - $300,000 per year.

2. The most frequent job title in this domain, based on the data provided, is "Family Practice-Without OB Physicia
n". This title appears 9 times across the different job listings.

Please note that these answers are based solely on the data you provided, and may not represent the complete pictur
e of salaries and job titles in this domain.
```

```
# Example 5 - Generative Search

response = query_job_postings(
  collection_name="JobPostings",
  query="Healthcare, Medicine",
  prompt="Are there any healthcare jobs in California? If yes, which city are they based in? Are any of them based
  limit=50
)

print(response.generated)
```

```
/databricks/python_shell/dbruntime/safe_oinspect.py:133: DeprecationWarning: `getargspec` function is deprecated as
of IPython 7.10and will be removed in future versions.
  argspec = oinspect.getargspec(obj)
Based on the data provided, there are several healthcare jobs located in California, specifically:

1. Telemedicine Physician - CA 100% Remote (Irvine, CA)
2. Internal Medicine Physician - $125/hourly - $135/hourly (San Mateo, CA)
3. Family Practice-Without OB Physician - $125/hourly - $135/hourly (San Mateo, CA)
4. Family Practice-Without OB Physician - $133/yearly (Sacramento, CA)
5. Family Practice-Without OB Physician - $275,000/yearly - $325,000/yearly (San Jose, CA)
```

```
6. Medical Doctor (Silicon Valley, CA)

So yes, there are healthcare jobs listed in the data that are based in Irvine, California specifically for the role
of "Telemedicine Physician - CA 100% Remote".
```

END