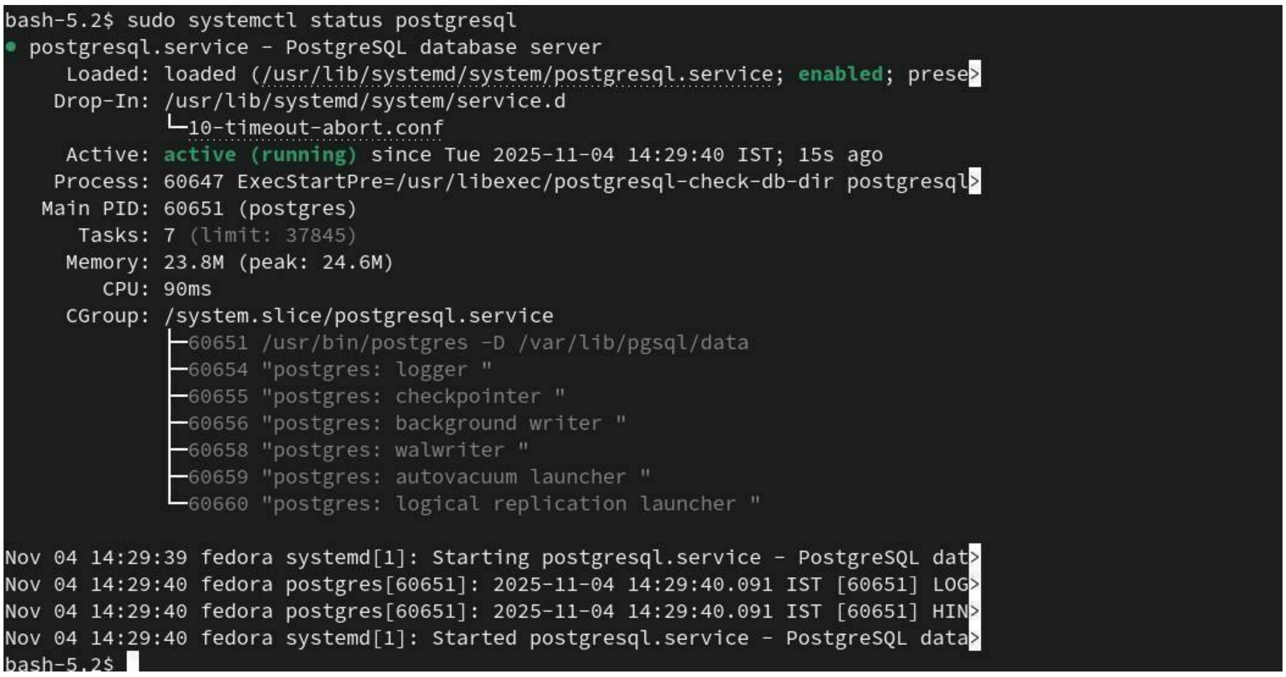
DBMS Lab 11

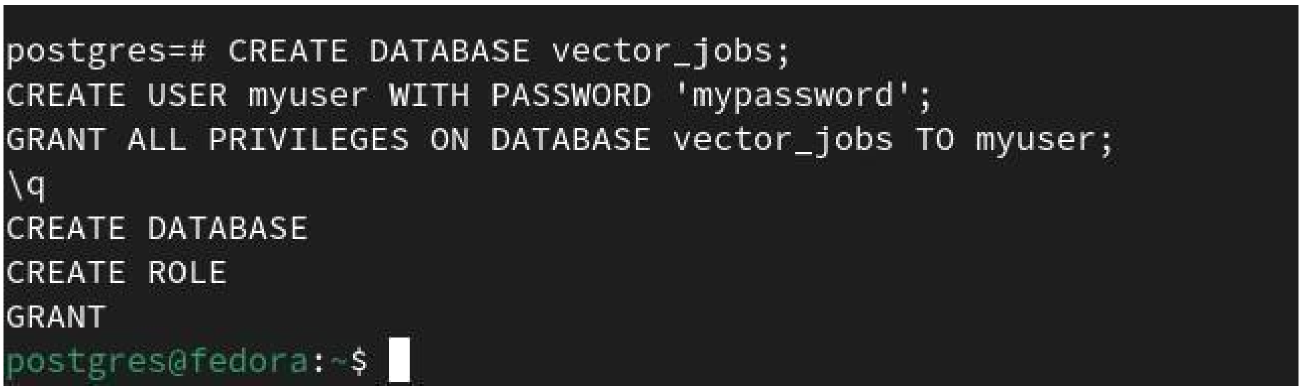
Name: Rithvik Rajesh Matta

SRN: PES2UG23CS485

Section: H

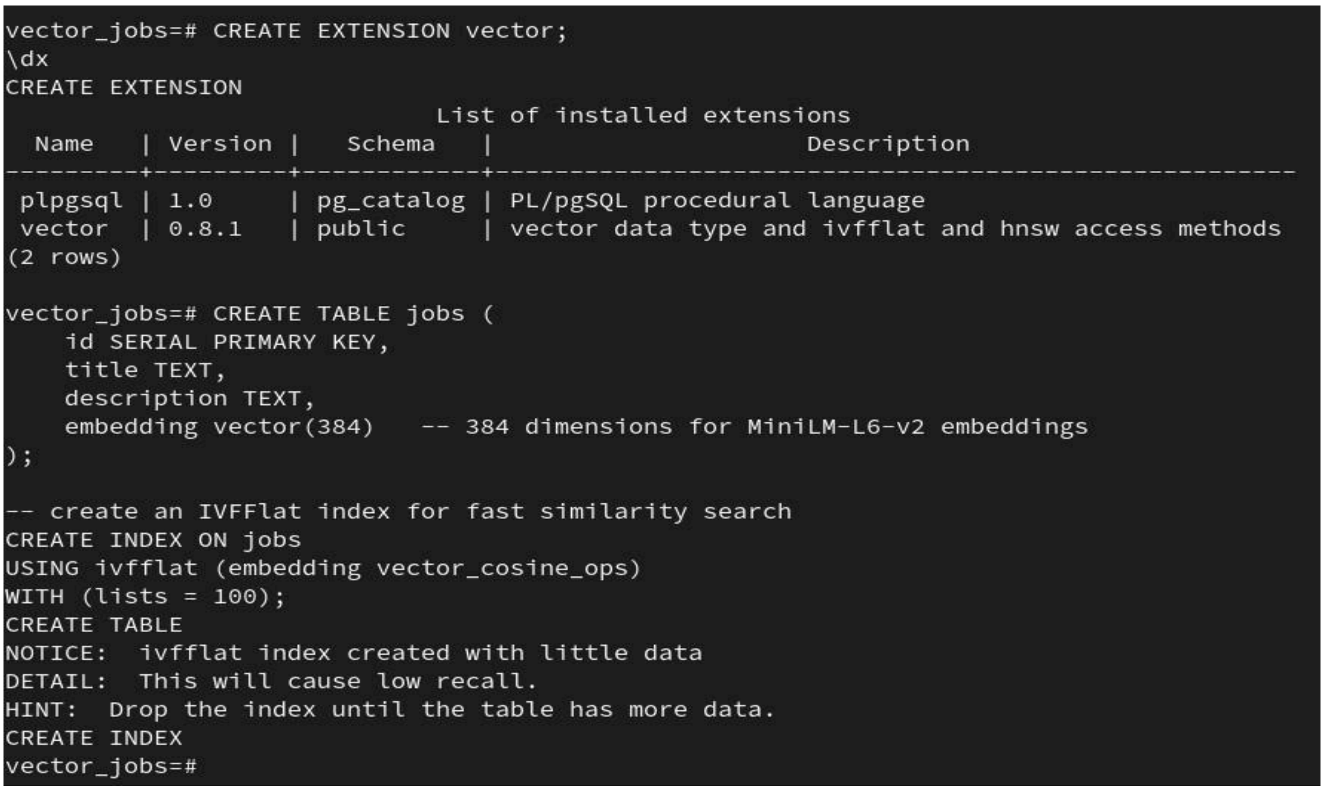


```
bash-5.2$ sudo systemctl status postgresql
● postgresql.service - PostgreSQL database server
     Loaded: loaded (/usr/lib/systemd/system/postgresql.service; enabled; prese>
    Drop-In: /usr/lib/systemd/system/service.d
             └─10-timeout-abort.conf
     Active: active (running) since Tue 2025-11-04 14:29:40 IST; 15s ago
    Process: 60647 ExecStartPre=/usr/libexec/postgresql-check-db-dir postgresql>
   Main PID: 60651 (postgres)
      Tasks: 7 (limit: 37845)
     Memory: 23.8M (peak: 24.6M)
        CPU: 90ms
     CGroup: /system.slice/postgresql.service
             ├─60651 /usr/bin/postgres -D /var/lib/pgsql/data
             ├─60654 "postgres: logger "
             ├─60655 "postgres: checkpointer "
             ├─60656 "postgres: background writer "
             ├─60658 "postgres: walwriter "
             ├─60659 "postgres: autovacuum launcher "
             └─60660 "postgres: logical replication launcher "

Nov 04 14:29:39 fedora systemd[1]: Starting postgresql.service - PostgreSQL dat>
Nov 04 14:29:40 fedora postgres[60651]: 2025-11-04 14:29:40.091 IST [60651] LOG>
Nov 04 14:29:40 fedora postgres[60651]: 2025-11-04 14:29:40.091 IST [60651] HIN>
Nov 04 14:29:40 fedora systemd[1]: Started postgresql.service - PostgreSQL data>
bash-5.2$
```



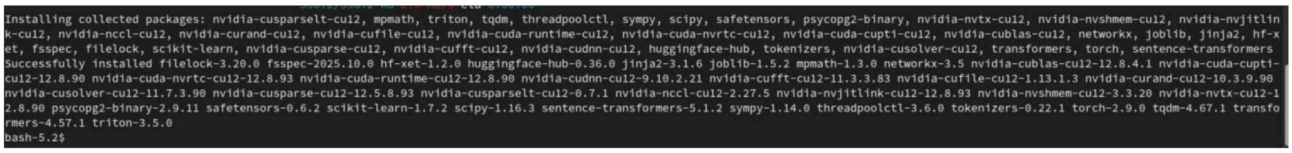
```
postgres=# CREATE DATABASE vector_jobs;
CREATE USER myuser WITH PASSWORD 'mypassword';
GRANT ALL PRIVILEGES ON DATABASE vector_jobs TO myuser;
\q
CREATE DATABASE
CREATE ROLE
GRANT
postgres@fedora:~$
```

Task 1:



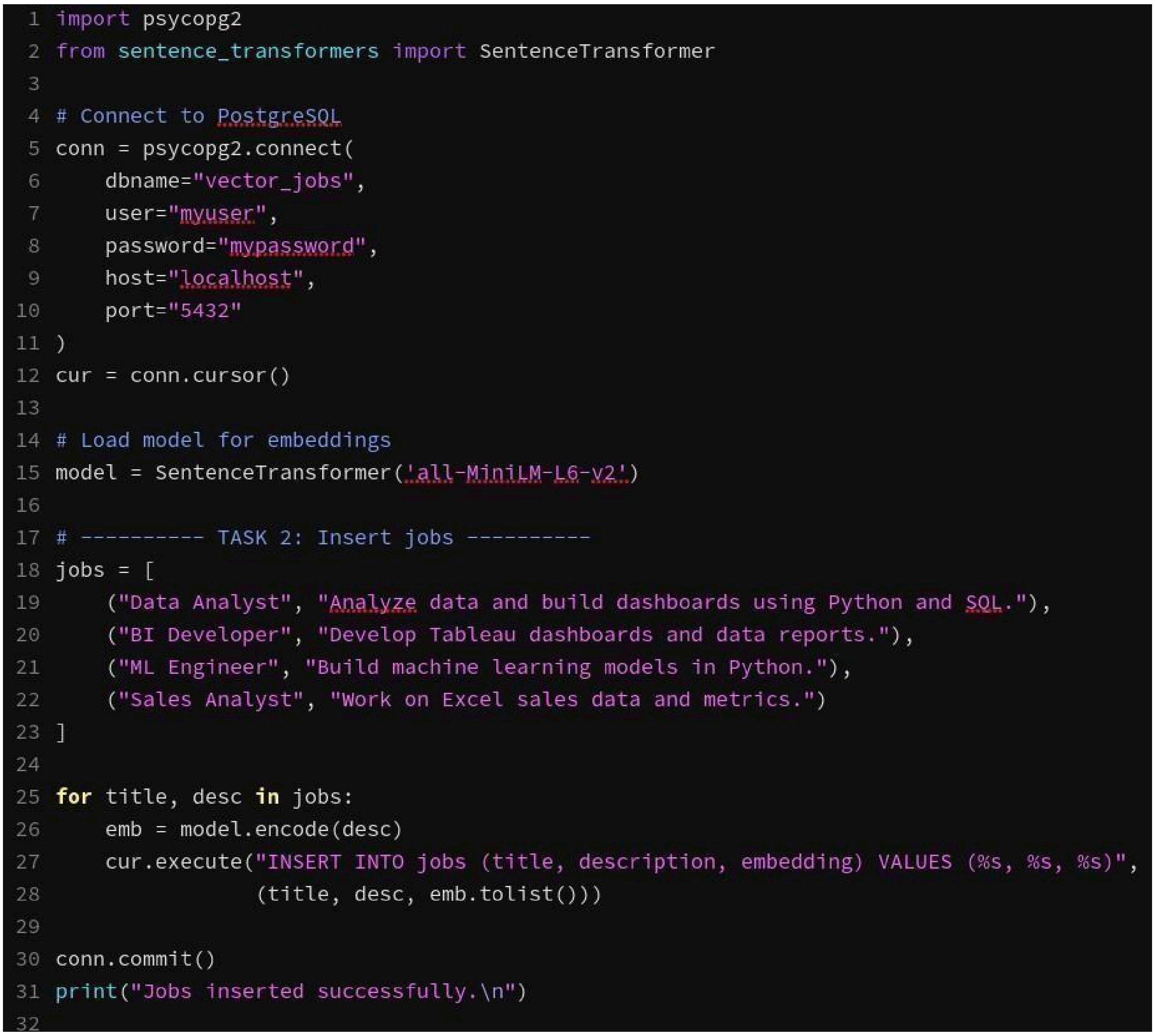
```
vector_jobs=# CREATE EXTENSION vector;
\dx
CREATE EXTENSION
                             List of installed extensions
  Name   | Version |   Schema   |                     Description
---------+---------+------------+------------------------------------------------------
 plpgsql | 1.0     | pg_catalog | PL/pgSQL procedural language
 vector  | 0.8.1   | public     | vector data type and ivfflat and hnsw access methods
(2 rows)

vector_jobs=# CREATE TABLE jobs (
    id SERIAL PRIMARY KEY,
    title TEXT,
    description TEXT,
    embedding vector(384)   -- 384 dimensions for MiniLM-L6-v2 embeddings
);

-- create an IVFFlat index for fast similarity search
CREATE INDEX ON jobs
USING ivfflat (embedding vector_cosine_ops)
WITH (lists = 100);
CREATE TABLE
NOTICE:  ivfflat index created with little data
DETAIL:  This will cause low recall.
HINT:  Drop the index until the table has more data.
CREATE INDEX
vector_jobs=#
```

Task 2 installation:



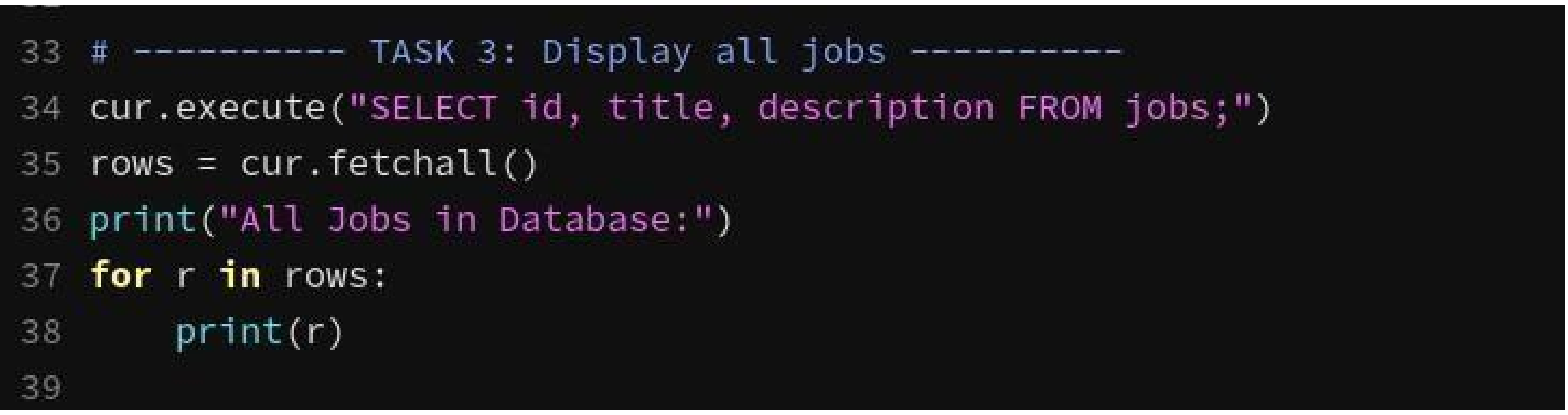
```
Installing collected packages: nvidia-cusparselt-cu12, mpmath, triton, tqdm, threadpoolctl, sympy, scipy, safetensors, psycopg2-binary, nvidia-nvtx-cu12, nvidia-nvshmem-cu12, nvidia-nvjitlin
k-cu12, nvidia-nccl-cu12, nvidia-curand-cu12, nvidia-cufile-cu12, nvidia-cuda-runtime-cu12, nvidia-cuda-nvrtc-cu12, nvidia-cuda-cupti-cu12, nvidia-cublas-cu12, networkx, joblib, jinja2, hf-x
et, fsspec, filelock, scikit-learn, nvidia-cusparse-cu12, nvidia-cufft-cu12, nvidia-cudnn-cu12, huggingface-hub, tokenizers, nvidia-cusolver-cu12, transformers, torch, sentence-transformers
Successfully installed filelock-3.20.0 fsspec-2025.10.0 hf-xet-1.2.0 huggingface-hub-0.36.0 jinja2-3.1.6 joblib-1.5.2 mpmath-1.3.0 networkx-3.5 nvidia-cublas-cu12-12.8.4.1 nvidia-cuda-cupti-
cu12-12.8.90 nvidia-cuda-nvrtc-cu12-12.8.93 nvidia-cuda-runtime-cu12-12.8.90 nvidia-cudnn-cu12-9.10.2.21 nvidia-cufft-cu12-11.3.3.83 nvidia-cufile-cu12-1.13.1.3 nvidia-curand-cu12-10.3.9.90
nvidia-cusolver-cu12-11.7.3.90 nvidia-cusparse-cu12-12.5.8.93 nvidia-cusparselt-cu12-0.7.1 nvidia-nccl-cu12-2.27.5 nvidia-nvjitlink-cu12-12.8.93 nvidia-nvshmem-cu12-3.3.20 nvidia-nvtx-cu12-1
2.8.90 psycopg2-binary-2.9.11 safetensors-0.6.2 scikit-learn-1.7.2 scipy-1.16.3 sentence-transformers-5.1.2 sympy-1.14.0 threadpoolctl-3.6.0 tokenizers-0.22.1 torch-2.9.0 tqdm-4.67.1 transfo
rmers-4.57.1 triton-3.5.0
bash-5.2$
```

Task 2 code:



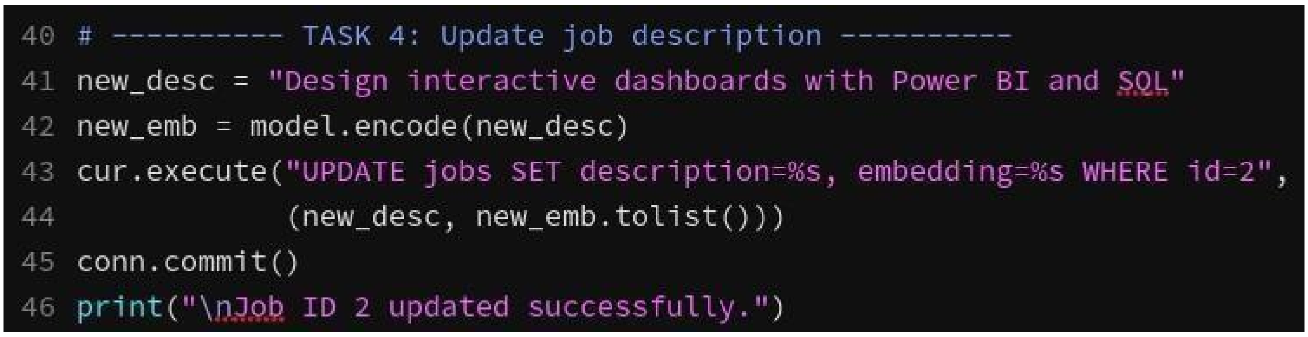
```python
import psycopg2
from sentence_transformers import SentenceTransformer

# Connect to PostgreSQL
conn = psycopg2.connect(
    dbname="vector_jobs",
    user="myuser",
    password="mypassword",
    host="localhost",
    port="5432"
)
cur = conn.cursor()

# Load model for embeddings
model = SentenceTransformer('all-MiniLM-L6-v2')

# ---------- TASK 2: Insert jobs ----------
jobs = [
    ("Data Analyst", "Analyze data and build dashboards using Python and SQL."),
    ("BI Developer", "Develop Tableau dashboards and data reports."),
    ("ML Engineer", "Build machine learning models in Python."),
    ("Sales Analyst", "Work on Excel sales data and metrics.")
]

for title, desc in jobs:
    emb = model.encode(desc)
    cur.execute("INSERT INTO jobs (title, description, embedding) VALUES (%s, %s, %s)",
                (title, desc, emb.tolist()))

conn.commit()
print("Jobs inserted successfully.\n")

```

Task 3 code:



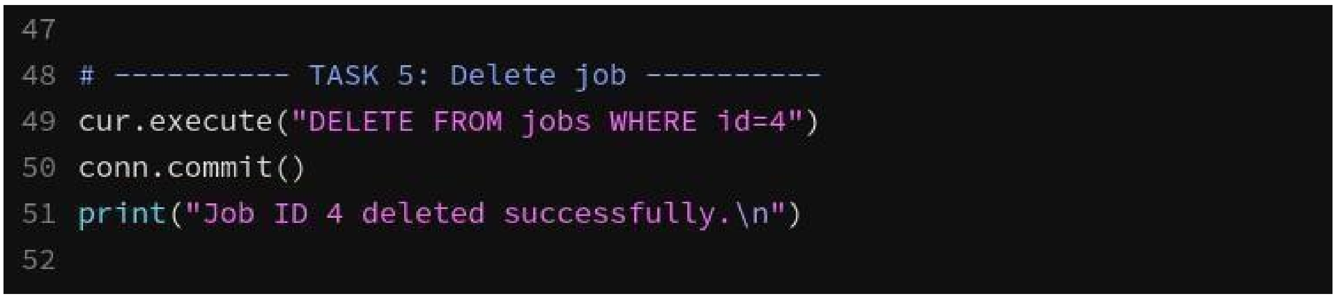
```python
# ---------- TASK 3: Display all jobs ----------
cur.execute("SELECT id, title, description FROM jobs;")
rows = cur.fetchall()
print("All Jobs in Database:")
for r in rows:
    print(r)

```

Task 4 code:



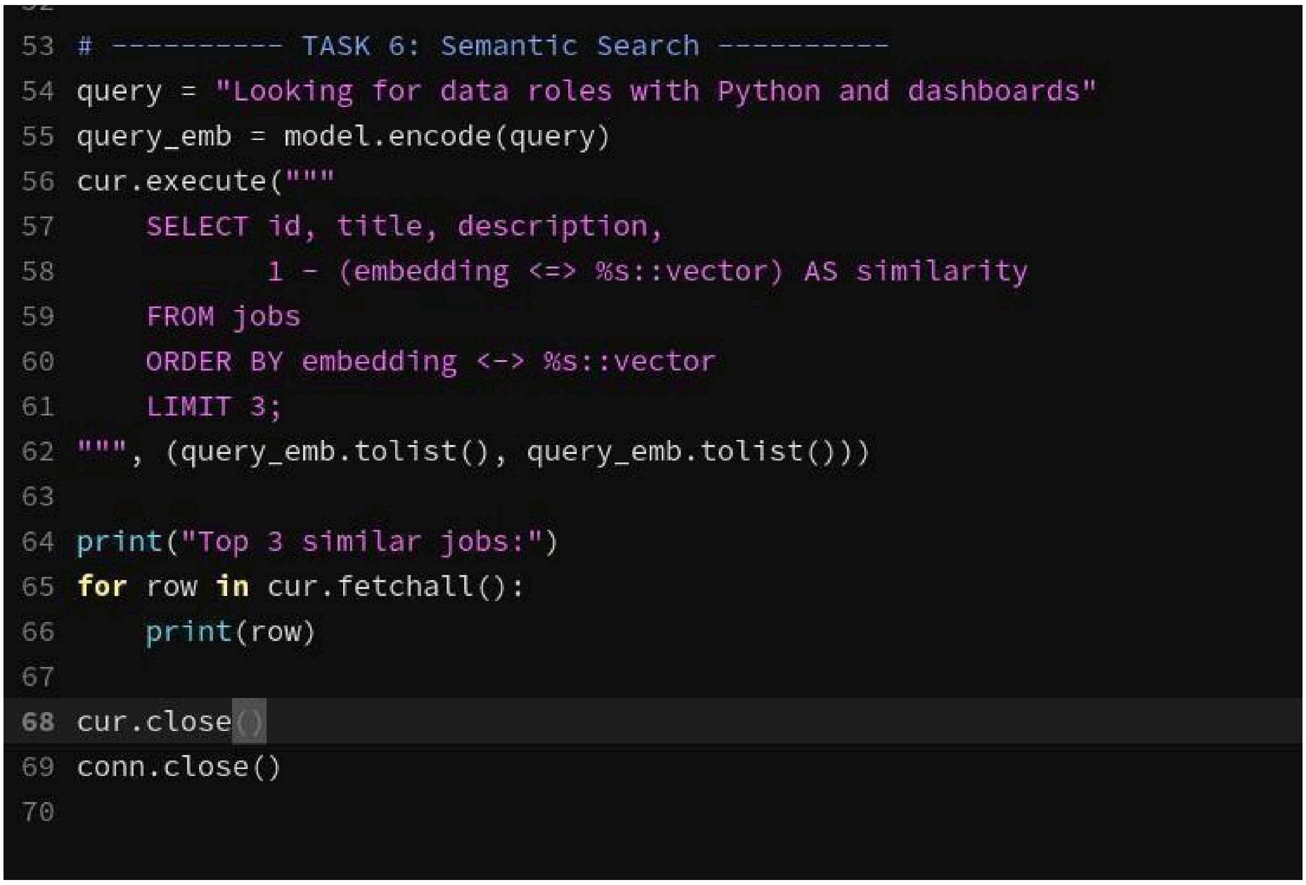
```python
# ---------- TASK 4: Update job description ----------
new_desc = "Design interactive dashboards with Power BI and SQL"
new_emb = model.encode(new_desc)
cur.execute("UPDATE jobs SET description=%s, embedding=%s WHERE id=2",
            (new_desc, new_emb.tolist()))
conn.commit()
print("\nJob ID 2 updated successfully.")
```

Task 5 code:



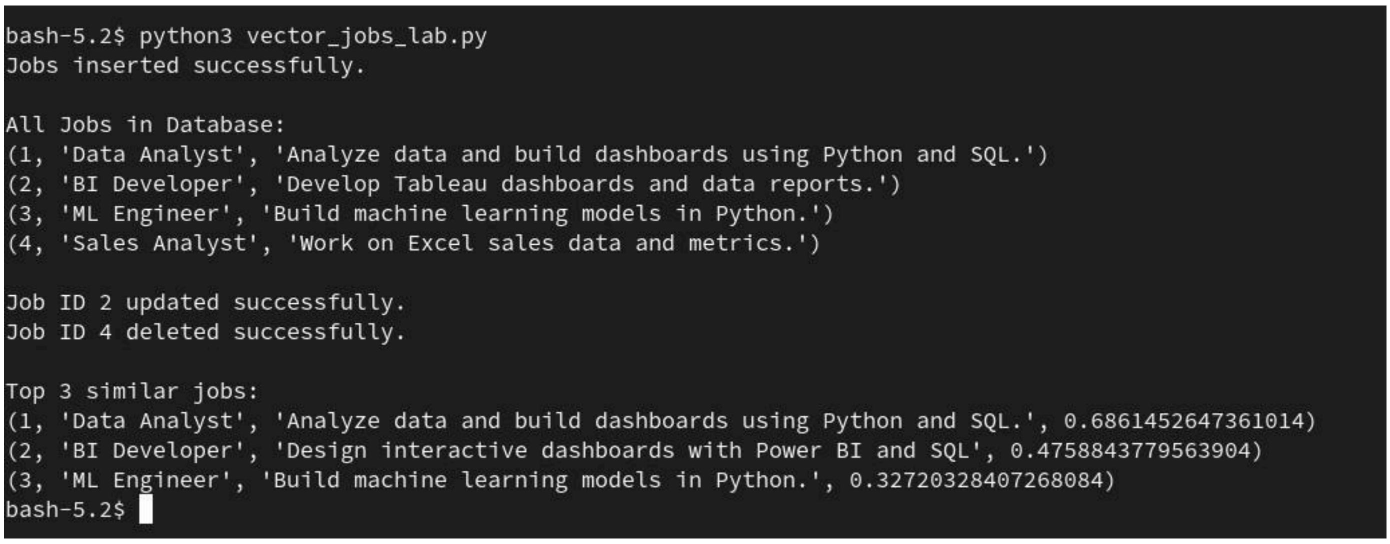
```python

# ---------- TASK 5: Delete job ----------
cur.execute("DELETE FROM jobs WHERE id=4")
conn.commit()
print("Job ID 4 deleted successfully.\n")

```

Task 6 code:



```python
# ---------- TASK 6: Semantic Search ----------
query = "Looking for data roles with Python and dashboards"
query_emb = model.encode(query)
cur.execute("""
    SELECT id, title, description,
           1 - (embedding <=> %s::vector) AS similarity
    FROM jobs
    ORDER BY embedding <-> %s::vector
    LIMIT 3;
""", (query_emb.tolist(), query_emb.tolist()))

print("Top 3 similar jobs:")
for row in cur.fetchall():
    print(row)

cur.close()
conn.close()

```

Task 2,3,4,5,6 output:



```
bash-5.2$ python3 vector_jobs_lab.py
Jobs inserted successfully.

All Jobs in Database:
(1, 'Data Analyst', 'Analyze data and build dashboards using Python and SQL.')
(2, 'BI Developer', 'Develop Tableau dashboards and data reports.')
(3, 'ML Engineer', 'Build machine learning models in Python.')
(4, 'Sales Analyst', 'Work on Excel sales data and metrics.')

Job ID 2 updated successfully.
Job ID 4 deleted successfully.

Top 3 similar jobs:
(1, 'Data Analyst', 'Analyze data and build dashboards using Python and SQL.', 0.6861452647361014)
(2, 'BI Developer', 'Design interactive dashboards with Power BI and SQL', 0.4758843779563904)
(3, 'ML Engineer', 'Build machine learning models in Python.', 0.32720328407268084)
bash-5.2$
```

CODE: ***# Import required libraries***

```
import psycopg2                          # For connecting to PostgreSQL
from sentence_transformers import SentenceTransformer  # For text
embeddings

# Connect to the PostgreSQL database
conn = psycopg2.connect(
   dbname="vector_jobs",       # Name of your database
   user="myuser",             # Database username
   password="RakeshRK@1829",    # Database password
   host="localhost",           # Host address
   port="5432"                # Default PostgreSQL port
)
cur = conn.cursor()  # Create a cursor to interact with the database

# Load the sentence transformer model for embeddings
model = SentenceTransformer('all-MiniLM-L6-v2')

# Sample job data: (title, description)
jobs = [
   ("Data Analyst", "Analyze data and build dashboards using Python and
SQL."),
   ("BI Developer", "Develop Tableau dashboards and data reports."),
   ("ML Engineer", "Build machine learning models in Python."),
   ("Sales Analyst", "Work on Excel sales data and metrics.")
]

# Insert job data into the database with vector embeddings
for title, desc in jobs:
   emb = model.encode(desc)  # Encode description into a vector
   cur.execute(
        "INSERT INTO jobs (title, description, embedding) VALUES (%s, %s, %s)",
        (title, desc, emb.tolist())  # Convert NumPy array to Python list before
storing
    )
    conn.commit()  # Commit after each insertion

print("Jobs inserted successfully.\n")

# Retrieve and display all jobs from the database
cur.execute("SELECT id, title, description FROM jobs;")
rows = cur.fetchall()

print("All Jobs in Database:")
for r in rows:
    print(r)

# --- Update a job entry (e.g., job with ID 2) ---
new_desc = "Design interactive dashboards with Power BI and SQL"
new_emb = model.encode(new_desc)

cur.execute(
    "UPDATE jobs SET description=%s, embedding=%s WHERE id=2",
    (new_desc, new_emb.tolist())
)
conn.commit()
print("\nJob ID 2 updated successfully.")

# --- Delete a job entry (e.g., job with ID 4) ---
cur.execute("DELETE FROM jobs WHERE id=4")
conn.commit()
print("Job ID 4 deleted successfully.\n")

# --- Perform a semantic search using vector similarity ---
query = "Looking for data roles with Python and dashboards"
query_emb = model.encode(query)

# Find the top 3 most similar jobs using cosine similarity
cur.execute("""
    SELECT
        id, title, description,
        1 - (embedding <=> %s::vector) AS similarity   -- Compute similarity score
    FROM jobs
    ORDER BY embedding <-> %s::vector              -- Order by distance (lower =
more similar)
   LIMIT 3;
""", (query_emb.tolist(), query_emb.tolist()))

print("Top 3 similar jobs:")
for row in cur.fetchall():
   print(row)

# Close connections
cur.close()
conn.close()
```