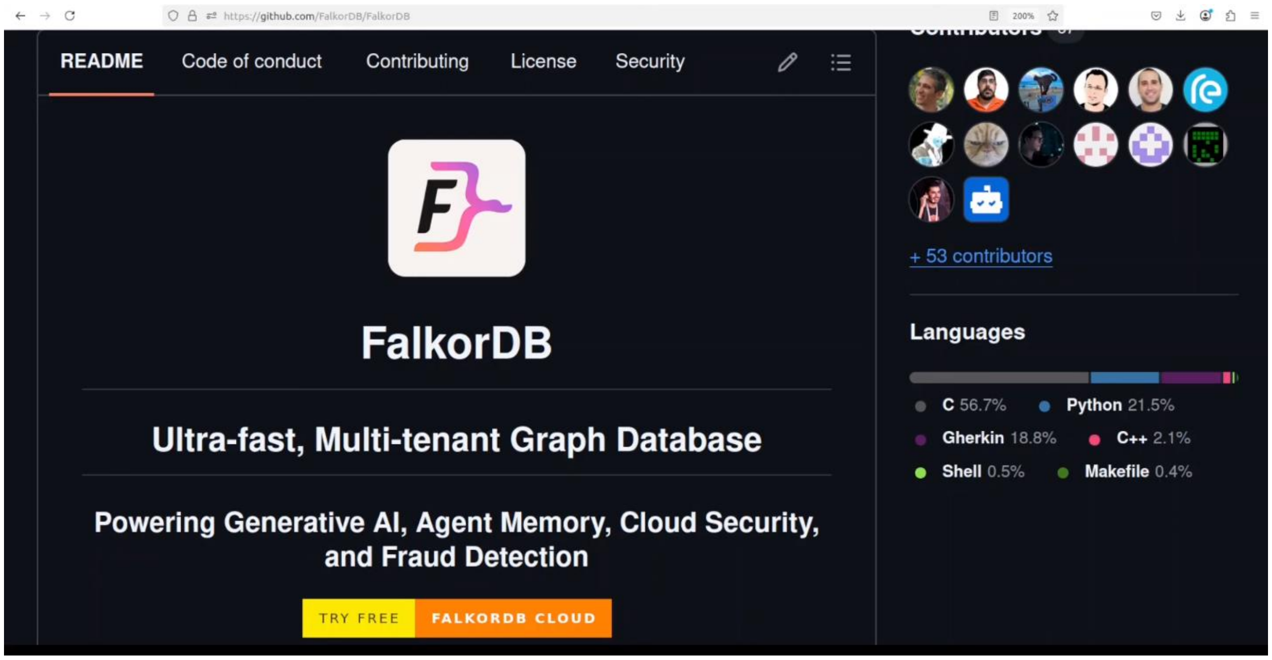
FalkorDB - GraphRAG for GenAI Apps - Install and Test Locally

Fahd Mirza
45.5K subscribers

Join · Subscribe · 53

1,005 views Sep 2, 2025 #falkordb
This video is a step-by-step Tutorial to install and use FalkorDB, which is a Knowledge Graph for LLM (GraphRAG).



FalkorDB

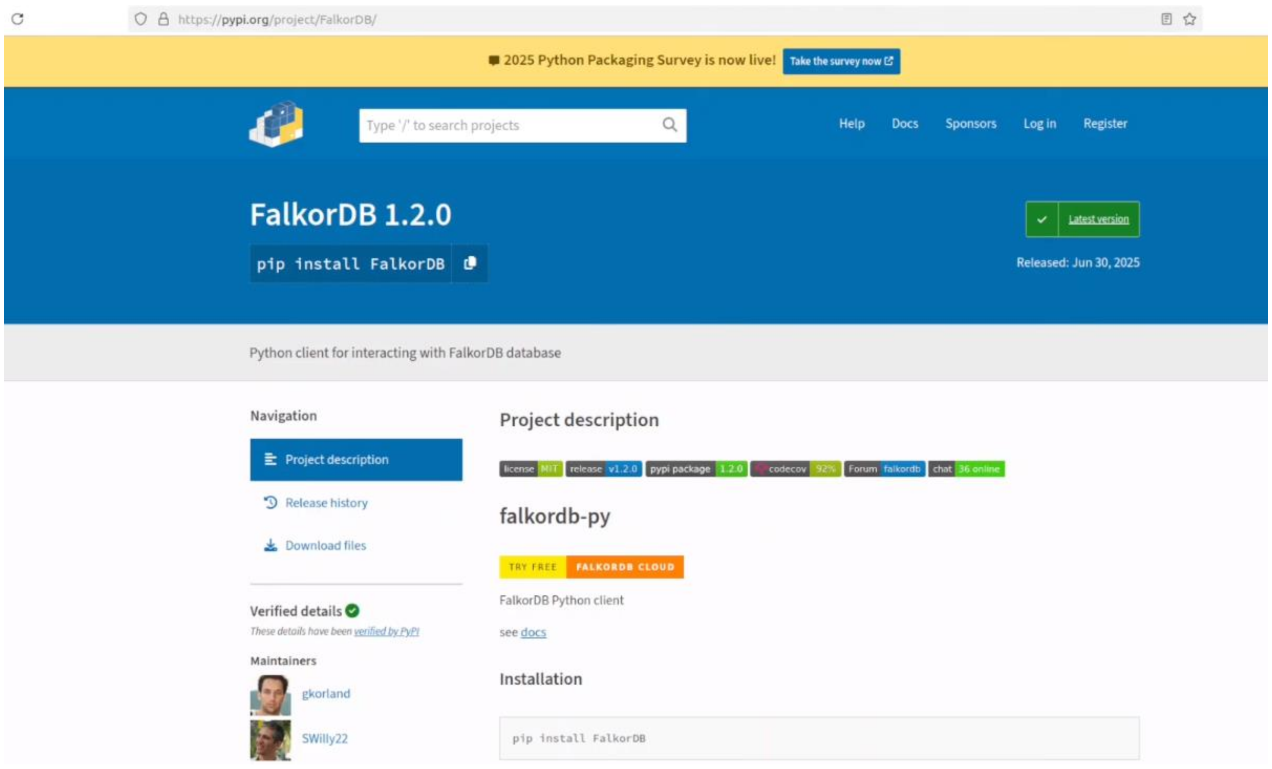
Ultra-fast, Multi-tenant Graph Database

Powering Generative AI, Agent Memory, Cloud Security, and Fraud Detection

TRY FREE FALKORDB CLOUD

Languages

● C 56.7%	● Python 21.5%
● Gherkin 18.8%	● C++ 2.1%
● Shell 0.5%	● Makefile 0.4%



2025 Python Packaging Survey is now live!

FalkorDB 1.2.0

```
pip install FalkorDB
```

Released: Jun 30, 2025

Python client for interacting with FalkorDB database

Navigation

Project description

falkordb-py

FalkorDB Python client

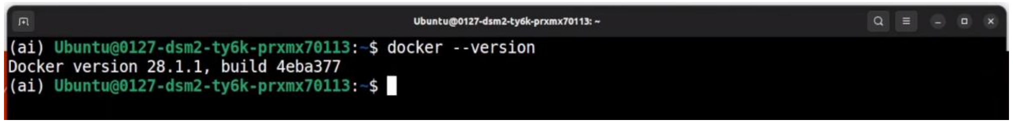
see docs

Installation

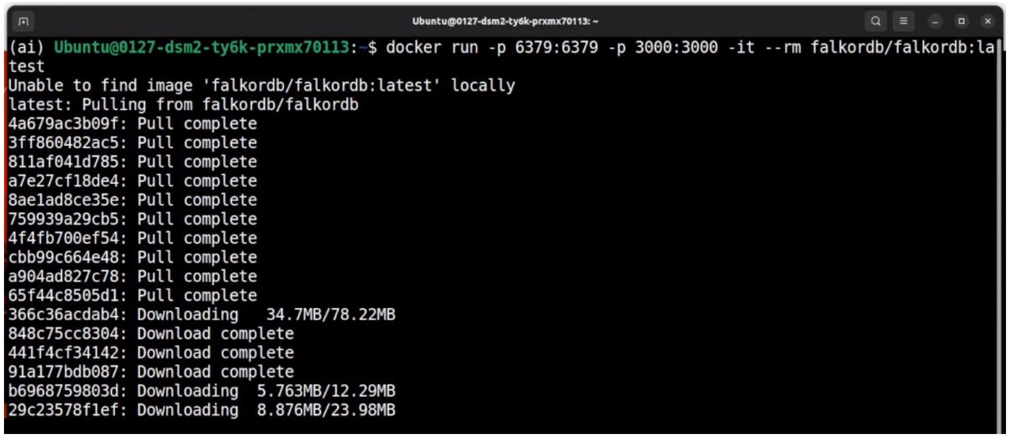
```
pip install FalkorDB
```

Verified details
These details have been verified by PyPI

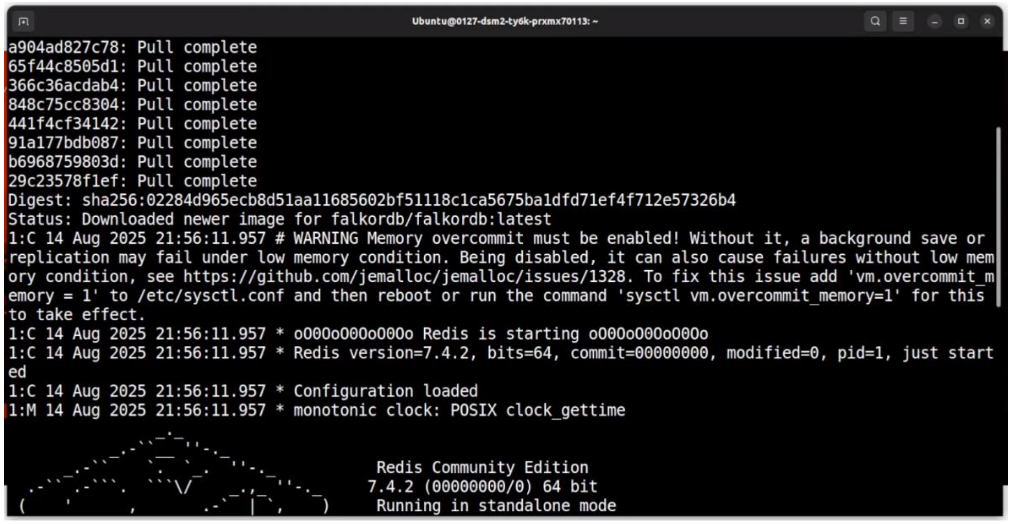
Maintainers
gkorland
SWilly22



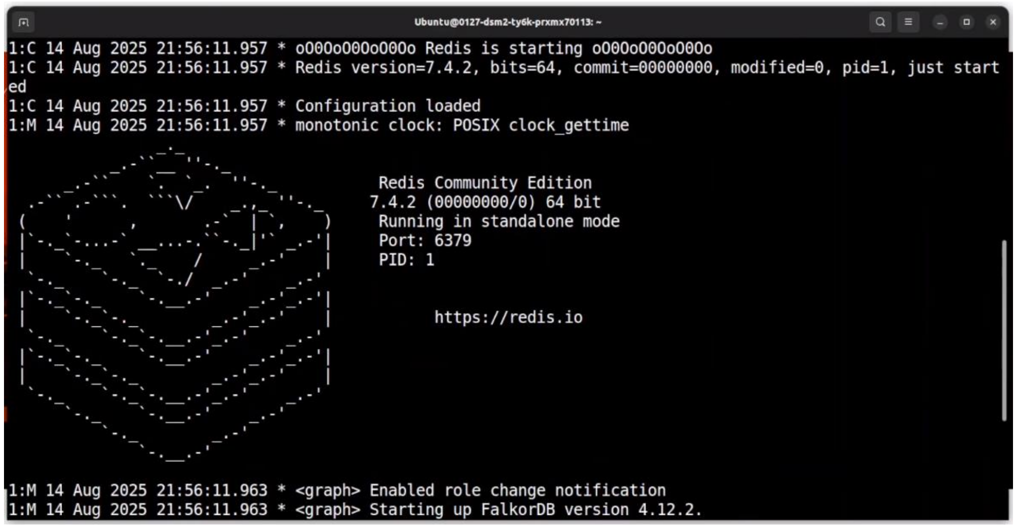
```
(ai) Ubuntu@0127-dsm2-ty6k-prxmx70113:~$ docker --version
Docker version 28.1.1, build 4eba377
(ai) Ubuntu@0127-dsm2-ty6k-prxmx70113:~$ 
```



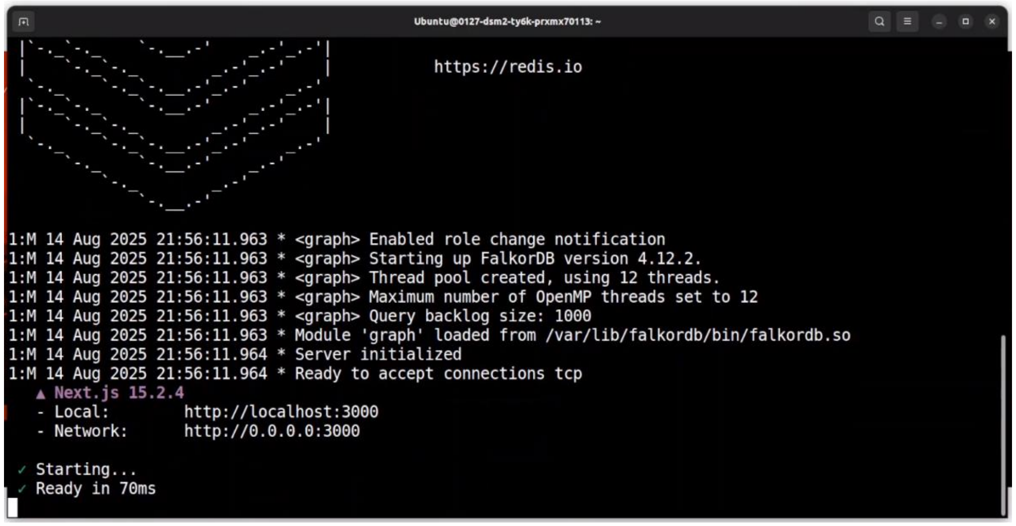
```
(ai) Ubuntu@0127-dsm2-ty6k-prxmx70113:~$ docker run -p 6379:6379 -p 3000:3000 -it --rm falkordb/falkordb:la
test
Unable to find image 'falkordb/falkordb:latest' locally
latest: Pulling from falkordb/falkordb
4a679ac3b09f: Pull complete
3ff860482ac5: Pull complete
811af041d785: Pull complete
a7e27cf18de4: Pull complete
8ae1ad8ce35e: Pull complete
759939a29cb5: Pull complete
4f4fb700ef54: Pull complete
cbb99c664e48: Pull complete
a904ad827c78: Pull complete
65f44c8505d1: Pull complete
366c36acdab4: Downloading   34.7MB/78.22MB
848c75cc8304: Download complete
441f4cf34142: Download complete
91a177bdb087: Download complete
b6968759803d: Downloading  5.763MB/12.29MB
29c23578f1ef: Downloading  8.876MB/23.98MB
```



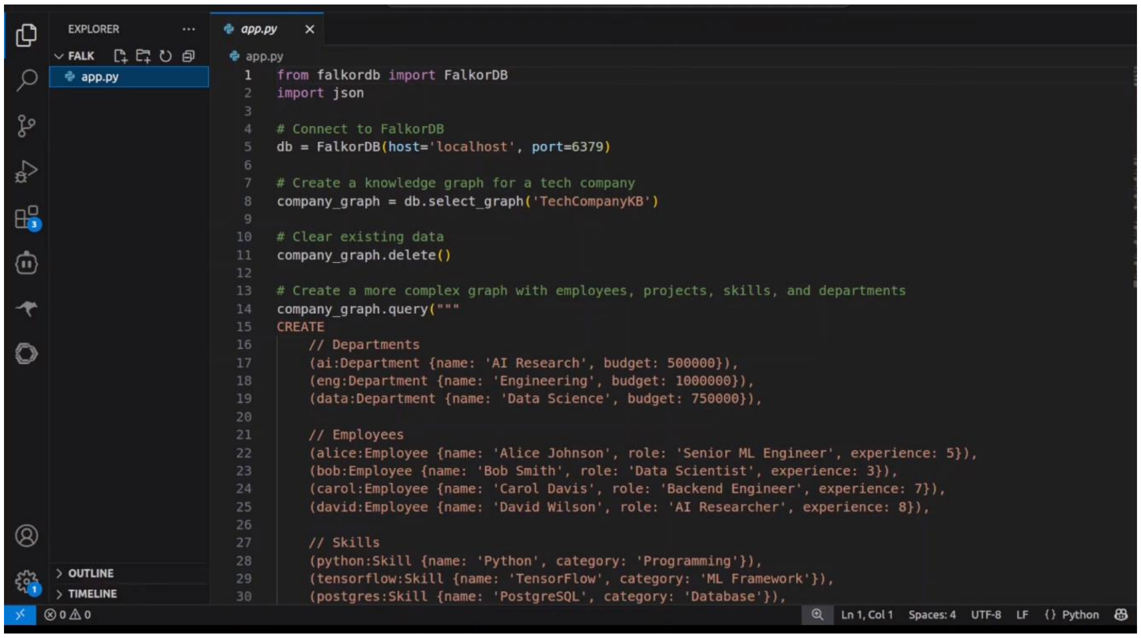
```
a904ad827c78: Pull complete
65f44c8505d1: Pull complete
366c36acdab4: Pull complete
848c75cc8304: Pull complete
441f4cf34142: Pull complete
91a177bdb087: Pull complete
b6968759803d: Pull complete
29c23578f1ef: Pull complete
Digest: sha256:02284d965ecb8d51aa11685602bf51118c1ca5675ba1dfd71ef4f712e57326b4
Status: Downloaded newer image for falkordb/falkordb:latest
1:C 14 Aug 2025 21:56:11.957 # WARNING Memory overcommit must be enabled! Without it, a background save or
replication may fail under low memory condition. Being disabled, it can also cause failures without low mem
ory condition, see https://github.com/jemalloc/jemalloc/issues/1328. To fix this issue add 'vm.overcommit_m
emory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this
to take effect.
1:C 14 Aug 2025 21:56:11.957 * oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 14 Aug 2025 21:56:11.957 * Redis version=7.4.2, bits=64, commit=00000000, modified=0, pid=1, just start
ed
1:C 14 Aug 2025 21:56:11.957 * Configuration loaded
1:M 14 Aug 2025 21:56:11.957 * monotonic clock: POSIX clock_gettime
                _._
           _.-``__ ''-._
      _.-``    `.  `_.  ''-._           Redis Community Edition
  .-`` .-```.  ```\/    _.,_ ''-._     7.4.2 (00000000/0) 64 bit
 (    '      ,       .-`  | `,    )     Running in standalone mode
```



```
1:C 14 Aug 2025 21:56:11.957 * oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 14 Aug 2025 21:56:11.957 * Redis version=7.4.2, bits=64, commit=00000000, modified=0, pid=1, just start
ed
1:C 14 Aug 2025 21:56:11.957 * Configuration loaded
1:M 14 Aug 2025 21:56:11.957 * monotonic clock: POSIX clock_gettime
                _._
           _.-``__ ''-._
      _.-``    `.  `_.  ''-._           Redis Community Edition
  .-`` .-```.  ```\/    _.,_ ''-._     7.4.2 (00000000/0) 64 bit
 (    '      ,       .-`  | `,    )     Running in standalone mode
 |`-._`-...-` __...-.``-._|'` _.-'|     Port: 6379
 |    `-._   `._    /     _.-'    |     PID: 1
  `-._    `-._  `-./  _.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |           https://redis.io
  `-._    `-._`-.__.-'_.-'    _.-'
 |`-._`-._    `-.__.-'    _.-'_.-'|
 |    `-._`-._        _.-'_.-'    |
  `-._    `-._`-.__.-'_.-'    _.-'
      `-._    `-.__.-'    _.-'
          `-._        _.-'
              `-.__.-'

1:M 14 Aug 2025 21:56:11.963 * <graph> Enabled role change notification
1:M 14 Aug 2025 21:56:11.963 * <graph> Starting up FalkorDB version 4.12.2.
```

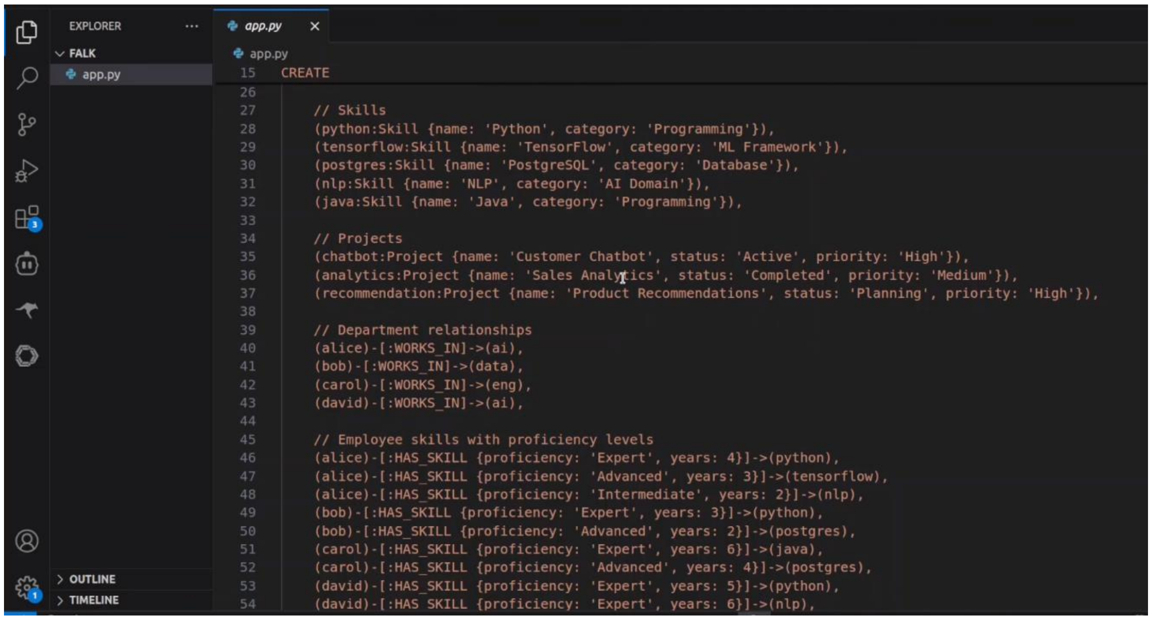


```
                                          https://redis.io

1:M 14 Aug 2025 21:56:11.963 * <graph> Enabled role change notification
1:M 14 Aug 2025 21:56:11.963 * <graph> Starting up FalkorDB version 4.12.2.
1:M 14 Aug 2025 21:56:11.963 * <graph> Thread pool created, using 12 threads.
1:M 14 Aug 2025 21:56:11.963 * <graph> Maximum number of OpenMP threads set to 12
1:M 14 Aug 2025 21:56:11.963 * <graph> Query backlog size: 1000
1:M 14 Aug 2025 21:56:11.963 * Module 'graph' loaded from /var/lib/falkordb/bin/falkordb.so
1:M 14 Aug 2025 21:56:11.964 * Server initialized
1:M 14 Aug 2025 21:56:11.964 * Ready to accept connections tcp
  ▲ Next.js 15.2.4
   - Local:        http://localhost:3000
   - Network:      http://0.0.0.0:3000

 ✓ Starting...
 ✓ Ready in 70ms
```

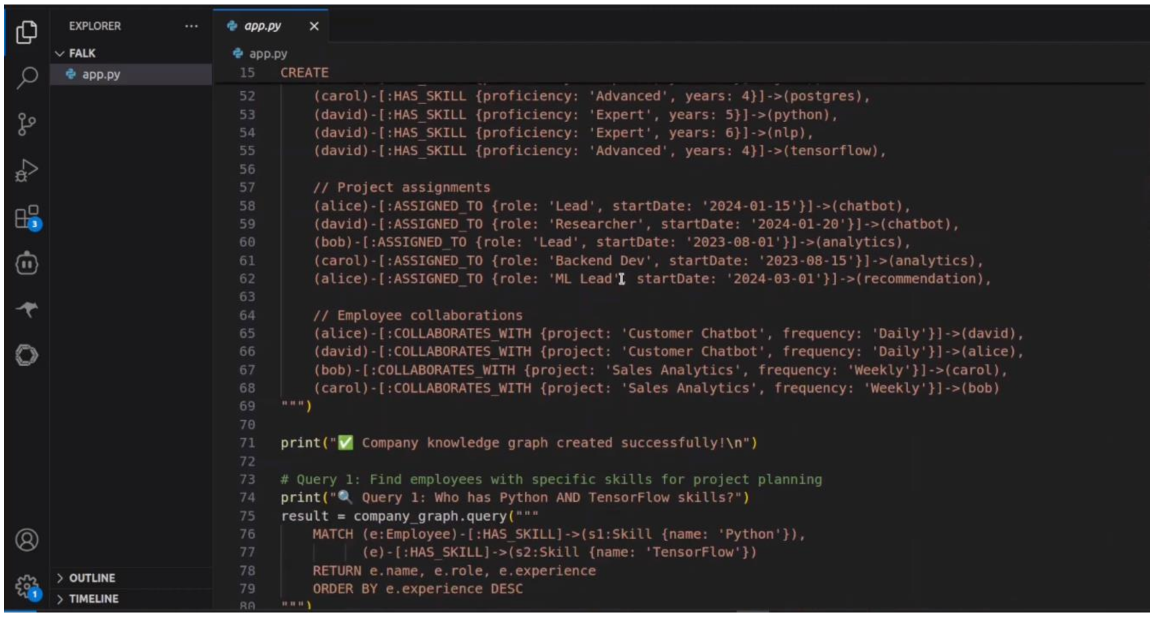


```python
from falkordb import FalkorDB
import json

# Connect to FalkorDB
db = FalkorDB(host='localhost', port=6379)

# Create a knowledge graph for a tech company
company_graph = db.select_graph('TechCompanyKB')

# Clear existing data
company_graph.delete()

# Create a more complex graph with employees, projects, skills, and departments
company_graph.query("""
CREATE
    // Departments
    (ai:Department {name: 'AI Research', budget: 500000}),
    (eng:Department {name: 'Engineering', budget: 1000000}),
    (data:Department {name: 'Data Science', budget: 750000}),

    // Employees
    (alice:Employee {name: 'Alice Johnson', role: 'Senior ML Engineer', experience: 5}),
    (bob:Employee {name: 'Bob Smith', role: 'Data Scientist', experience: 3}),
    (carol:Employee {name: 'Carol Davis', role: 'Backend Engineer', experience: 7}),
    (david:Employee {name: 'David Wilson', role: 'AI Researcher', experience: 8}),

    // Skills
    (python:Skill {name: 'Python', category: 'Programming'}),
    (tensorflow:Skill {name: 'TensorFlow', category: 'ML Framework'}),
    (postgres:Skill {name: 'PostgreSQL', category: 'Database'}),
```

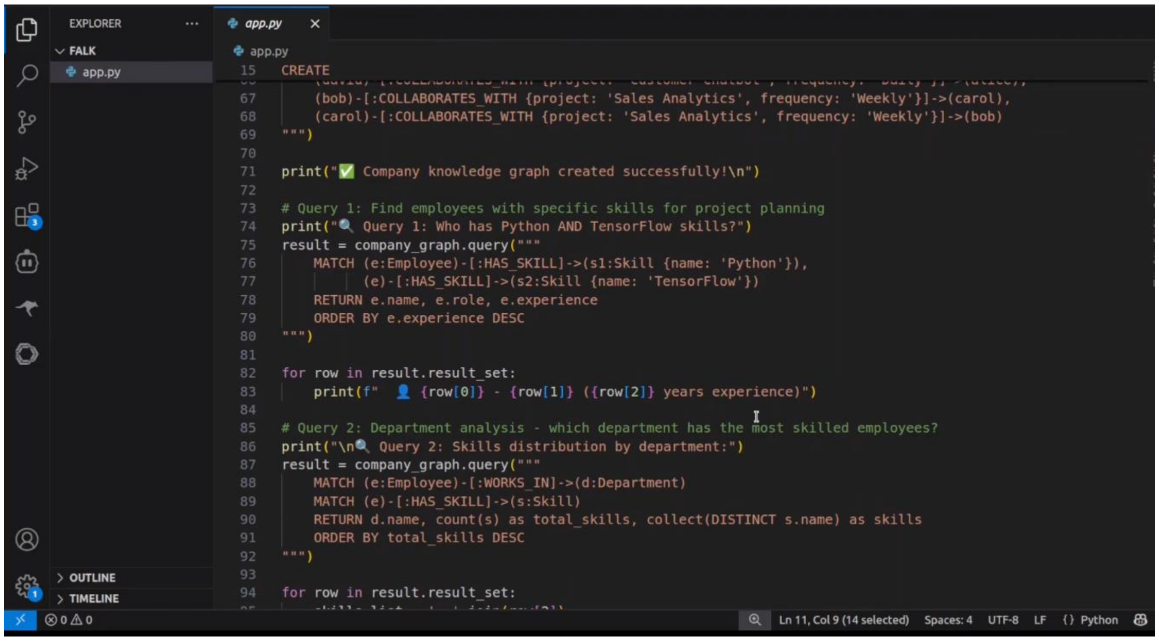
This code demonstrates the use of **FalkorDB** on a real-world use case, this is a complete knowledge graph that models workplace relationships.



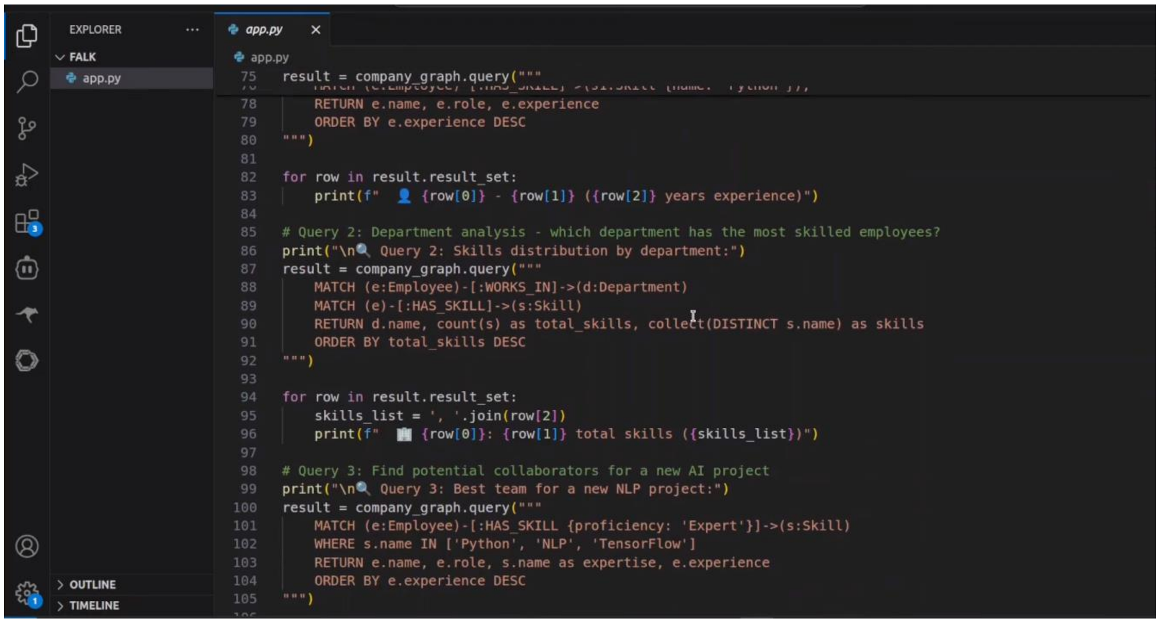
```python
CREATE

    // Skills
    (python:Skill {name: 'Python', category: 'Programming'}),
    (tensorflow:Skill {name: 'TensorFlow', category: 'ML Framework'}),
    (postgres:Skill {name: 'PostgreSQL', category: 'Database'}),
    (nlp:Skill {name: 'NLP', category: 'AI Domain'}),
    (java:Skill {name: 'Java', category: 'Programming'}),

    // Projects
    (chatbot:Project {name: 'Customer Chatbot', status: 'Active', priority: 'High'}),
    (analytics:Project {name: 'Sales Analytics', status: 'Completed', priority: 'Medium'}),
    (recommendation:Project {name: 'Product Recommendations', status: 'Planning', priority: 'High'}),

    // Department relationships
    (alice)-[:WORKS_IN]->(ai),
    (bob)-[:WORKS_IN]->(data),
    (carol)-[:WORKS_IN]->(eng),
    (david)-[:WORKS_IN]->(ai),

    // Employee skills with proficiency levels
    (alice)-[:HAS_SKILL {proficiency: 'Expert', years: 4}]->(python),
    (alice)-[:HAS_SKILL {proficiency: 'Advanced', years: 3}]->(tensorflow),
    (alice)-[:HAS_SKILL {proficiency: 'Intermediate', years: 2}]->(nlp),
    (bob)-[:HAS_SKILL {proficiency: 'Expert', years: 3}]->(python),
    (bob)-[:HAS_SKILL {proficiency: 'Advanced', years: 2}]->(postgres),
    (carol)-[:HAS_SKILL {proficiency: 'Expert', years: 6}]->(java),
    (carol)-[:HAS_SKILL {proficiency: 'Advanced', years: 4}]->(postgres),
    (david)-[:HAS_SKILL {proficiency: 'Expert', years: 5}]->(python),
    (david)-[:HAS_SKILL {proficiency: 'Expert', years: 6}]->(nlp),
```



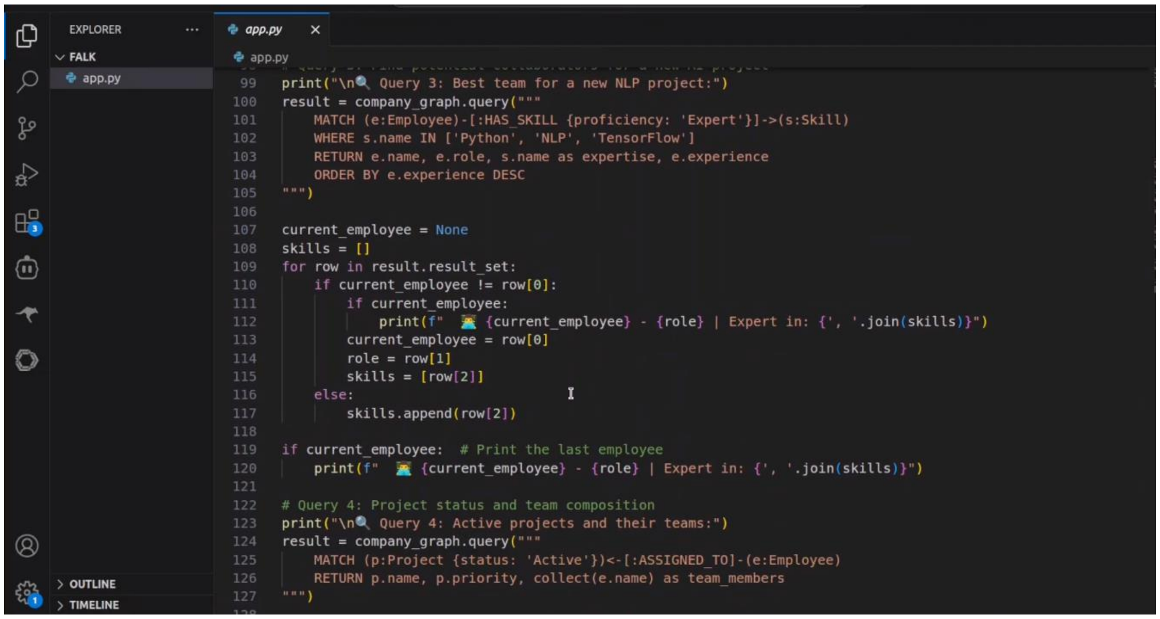
```
CREATE
    (carol)-[:HAS_SKILL {proficiency: 'Advanced', years: 4}]->(postgres),
    (david)-[:HAS_SKILL {proficiency: 'Expert', years: 5}]->(python),
    (david)-[:HAS_SKILL {proficiency: 'Expert', years: 6}]->(nlp),
    (david)-[:HAS_SKILL {proficiency: 'Advanced', years: 4}]->(tensorflow),

    // Project assignments
    (alice)-[:ASSIGNED_TO {role: 'Lead', startDate: '2024-01-15'}]->(chatbot),
    (david)-[:ASSIGNED_TO {role: 'Researcher', startDate: '2024-01-20'}]->(chatbot),
    (bob)-[:ASSIGNED_TO {role: 'Lead', startDate: '2023-08-01'}]->(analytics),
    (carol)-[:ASSIGNED_TO {role: 'Backend Dev', startDate: '2023-08-15'}]->(analytics),
    (alice)-[:ASSIGNED_TO {role: 'ML Lead', startDate: '2024-03-01'}]->(recommendation),

    // Employee collaborations
    (alice)-[:COLLABORATES_WITH {project: 'Customer Chatbot', frequency: 'Daily'}]->(david),
    (david)-[:COLLABORATES_WITH {project: 'Customer Chatbot', frequency: 'Daily'}]->(alice),
    (bob)-[:COLLABORATES_WITH {project: 'Sales Analytics', frequency: 'Weekly'}]->(carol),
    (carol)-[:COLLABORATES_WITH {project: 'Sales Analytics', frequency: 'Weekly'}]->(bob)
""")

print("✅ Company knowledge graph created successfully!\n")

# Query 1: Find employees with specific skills for project planning
print("🔍 Query 1: Who has Python AND TensorFlow skills?")
result = company_graph.query("""
    MATCH (e:Employee)-[:HAS_SKILL]->(s1:Skill {name: 'Python'}),
          (e)-[:HAS_SKILL]->(s2:Skill {name: 'TensorFlow'})
    RETURN e.name, e.role, e.experience
    ORDER BY e.experience DESC
```



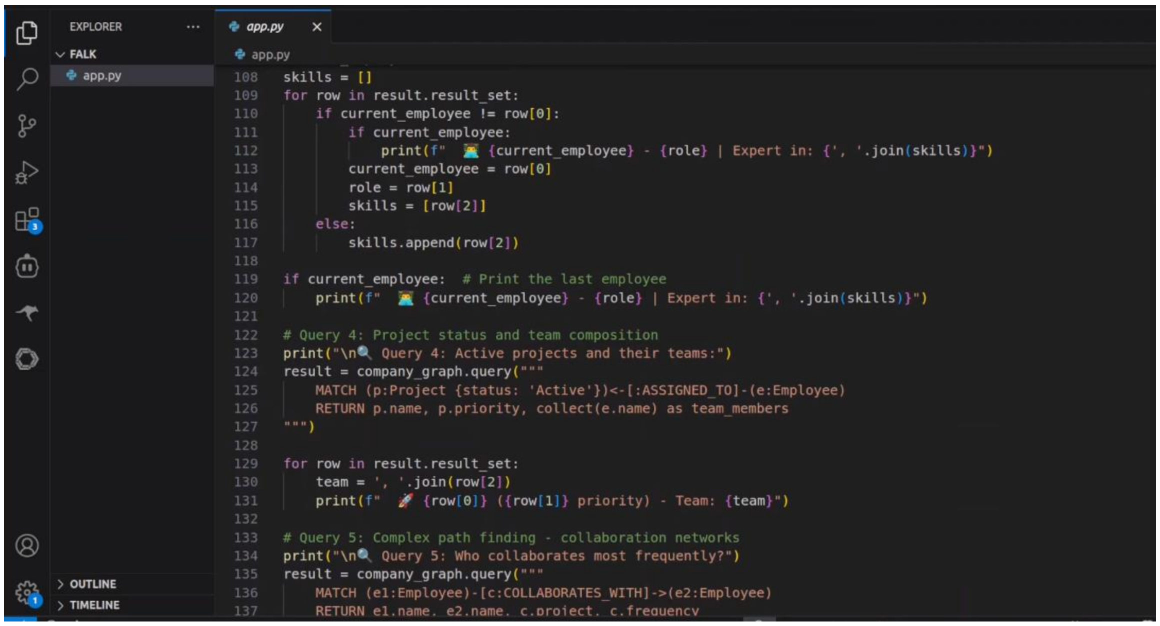
```
CREATE
    (bob)-[:COLLABORATES_WITH {project: 'Sales Analytics', frequency: 'Weekly'}]->(carol),
    (carol)-[:COLLABORATES_WITH {project: 'Sales Analytics', frequency: 'Weekly'}]->(bob)
""")

print("✅ Company knowledge graph created successfully!\n")

# Query 1: Find employees with specific skills for project planning
print("🔍 Query 1: Who has Python AND TensorFlow skills?")
result = company_graph.query("""
    MATCH (e:Employee)-[:HAS_SKILL]->(s1:Skill {name: 'Python'}),
          (e)-[:HAS_SKILL]->(s2:Skill {name: 'TensorFlow'})
    RETURN e.name, e.role, e.experience
    ORDER BY e.experience DESC
""")

for row in result.result_set:
    print(f"  👤 {row[0]} - {row[1]} ({row[2]} years experience)")

# Query 2: Department analysis - which department has the most skilled employees?
print("\n🔍 Query 2: Skills distribution by department:")
result = company_graph.query("""
    MATCH (e:Employee)-[:WORKS_IN]->(d:Department)
    MATCH (e)-[:HAS_SKILL]->(s:Skill)
    RETURN d.name, count(s) as total_skills, collect(DISTINCT s.name) as skills
    ORDER BY total_skills DESC
""")

for row in result.result_set:
```



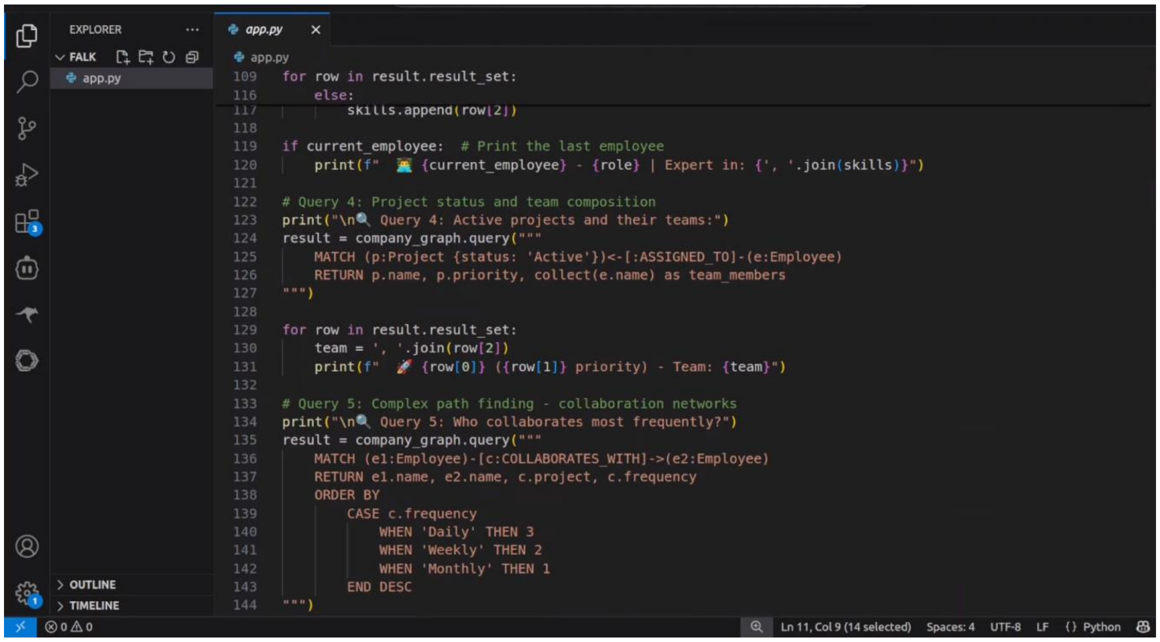
```
result = company_graph.query("""
    RETURN e.name, e.role, e.experience
    ORDER BY e.experience DESC
""")

for row in result.result_set:
    print(f"  👤 {row[0]} - {row[1]} ({row[2]} years experience)")

# Query 2: Department analysis - which department has the most skilled employees?
print("\n🔍 Query 2: Skills distribution by department:")
result = company_graph.query("""
    MATCH (e:Employee)-[:WORKS_IN]->(d:Department)
    MATCH (e)-[:HAS_SKILL]->(s:Skill)
    RETURN d.name, count(s) as total_skills, collect(DISTINCT s.name) as skills
    ORDER BY total_skills DESC
""")

for row in result.result_set:
    skills_list = ', '.join(row[2])
    print(f"  🏢 {row[0]}: {row[1]} total skills ({skills_list})")

# Query 3: Find potential collaborators for a new AI project
print("\n🔍 Query 3: Best team for a new NLP project:")
result = company_graph.query("""
    MATCH (e:Employee)-[:HAS_SKILL {proficiency: 'Expert'}]->(s:Skill)
    WHERE s.name IN ['Python', 'NLP', 'TensorFlow']
    RETURN e.name, e.role, s.name as expertise, e.experience
    ORDER BY e.experience DESC
""")
```



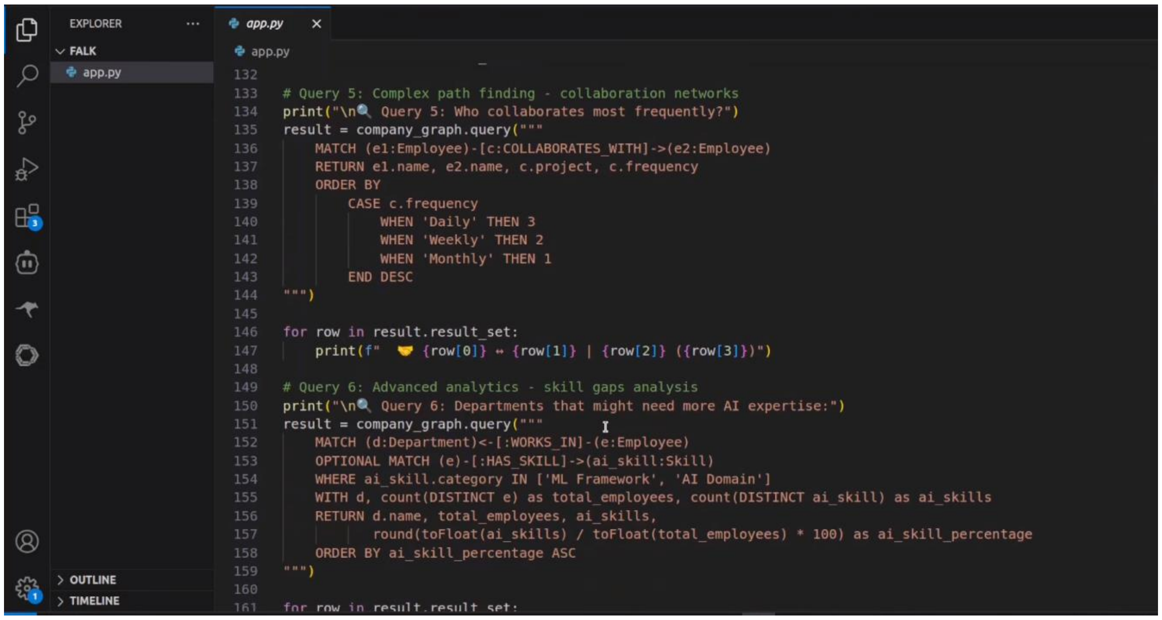
```python
print("\n🔍 Query 3: Best team for a new NLP project:")
result = company_graph.query("""
    MATCH (e:Employee)-[:HAS_SKILL {proficiency: 'Expert'}]->(s:Skill)
    WHERE s.name IN ['Python', 'NLP', 'TensorFlow']
    RETURN e.name, e.role, s.name as expertise, e.experience
    ORDER BY e.experience DESC
""")

current_employee = None
skills = []
for row in result.result_set:
    if current_employee != row[0]:
        if current_employee:
            print(f"  👨‍💻 {current_employee} - {role} | Expert in: {', '.join(skills)}")
        current_employee = row[0]
        role = row[1]
        skills = [row[2]]
    else:
        skills.append(row[2])

if current_employee:  # Print the last employee
    print(f"  👨‍💻 {current_employee} - {role} | Expert in: {', '.join(skills)}")

# Query 4: Project status and team composition
print("\n🔍 Query 4: Active projects and their teams:")
result = company_graph.query("""
    MATCH (p:Project {status: 'Active'})<-[:ASSIGNED_TO]-(e:Employee)
    RETURN p.name, p.priority, collect(e.name) as team_members
""")
```



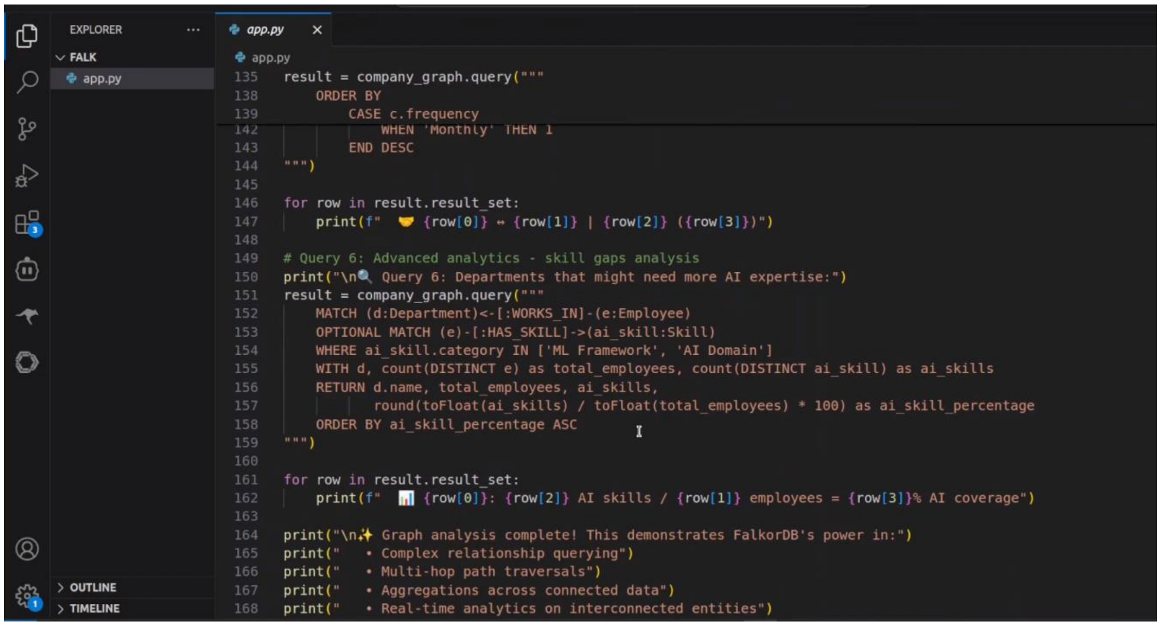
```python
skills = []
for row in result.result_set:
    if current_employee != row[0]:
        if current_employee:
            print(f"  👨‍💻 {current_employee} - {role} | Expert in: {', '.join(skills)}")
        current_employee = row[0]
        role = row[1]
        skills = [row[2]]
    else:
        skills.append(row[2])

if current_employee:  # Print the last employee
    print(f"  👨‍💻 {current_employee} - {role} | Expert in: {', '.join(skills)}")

# Query 4: Project status and team composition
print("\n🔍 Query 4: Active projects and their teams:")
result = company_graph.query("""
    MATCH (p:Project {status: 'Active'})<-[:ASSIGNED_TO]-(e:Employee)
    RETURN p.name, p.priority, collect(e.name) as team_members
""")

for row in result.result_set:
    team = ', '.join(row[2])
    print(f"  🚀 {row[0]} ({row[1]} priority) - Team: {team}")

# Query 5: Complex path finding - collaboration networks
print("\n🔍 Query 5: Who collaborates most frequently?")
result = company_graph.query("""
    MATCH (e1:Employee)-[c:COLLABORATES_WITH]->(e2:Employee)
    RETURN e1.name, e2.name, c.project, c.frequency
```



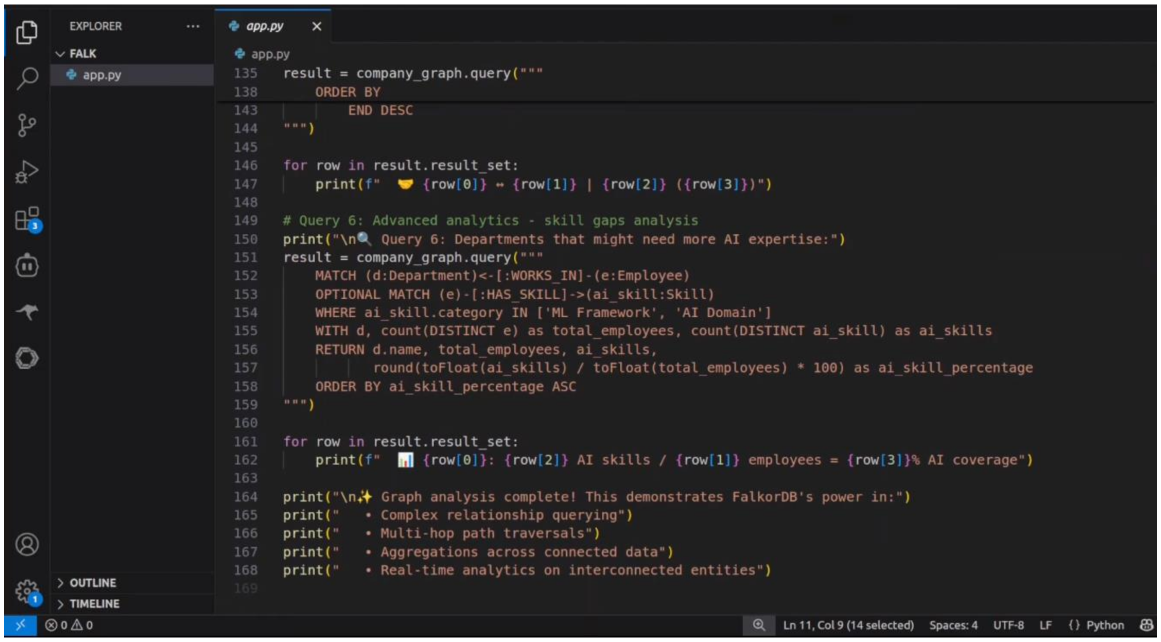
```python
for row in result.result_set:
    else:
        skills.append(row[2])

if current_employee:  # Print the last employee
    print(f"  👨‍💻 {current_employee} - {role} | Expert in: {', '.join(skills)}")

# Query 4: Project status and team composition
print("\n🔍 Query 4: Active projects and their teams:")
result = company_graph.query("""
    MATCH (p:Project {status: 'Active'})<-[:ASSIGNED_TO]-(e:Employee)
    RETURN p.name, p.priority, collect(e.name) as team_members
""")

for row in result.result_set:
    team = ', '.join(row[2])
    print(f"  🚀 {row[0]} ({row[1]} priority) - Team: {team}")

# Query 5: Complex path finding - collaboration networks
print("\n🔍 Query 5: Who collaborates most frequently?")
result = company_graph.query("""
    MATCH (e1:Employee)-[c:COLLABORATES_WITH]->(e2:Employee)
    RETURN e1.name, e2.name, c.project, c.frequency
    ORDER BY
        CASE c.frequency
            WHEN 'Daily' THEN 3
            WHEN 'Weekly' THEN 2
            WHEN 'Monthly' THEN 1
        END DESC
""")
```



```python

# Query 5: Complex path finding - collaboration networks
print("\n🔍 Query 5: Who collaborates most frequently?")
result = company_graph.query("""
    MATCH (e1:Employee)-[c:COLLABORATES_WITH]->(e2:Employee)
    RETURN e1.name, e2.name, c.project, c.frequency
    ORDER BY
        CASE c.frequency
            WHEN 'Daily' THEN 3
            WHEN 'Weekly' THEN 2
            WHEN 'Monthly' THEN 1
        END DESC
""")

for row in result.result_set:
    print(f"  🤝 {row[0]} ↔ {row[1]} | {row[2]} ({row[3]})")

# Query 6: Advanced analytics - skill gaps analysis
print("\n🔍 Query 6: Departments that might need more AI expertise:")
result = company_graph.query("""
    MATCH (d:Department)<-[:WORKS_IN]-(e:Employee)
    OPTIONAL MATCH (e)-[:HAS_SKILL]->(ai_skill:Skill)
    WHERE ai_skill.category IN ['ML Framework', 'AI Domain']
    WITH d, count(DISTINCT e) as total_employees, count(DISTINCT ai_skill) as ai_skills
    RETURN d.name, total_employees, ai_skills,
            round(toFloat(ai_skills) / toFloat(total_employees) * 100) as ai_skill_percentage
    ORDER BY ai_skill_percentage ASC
""")

for row in result.result_set:
```

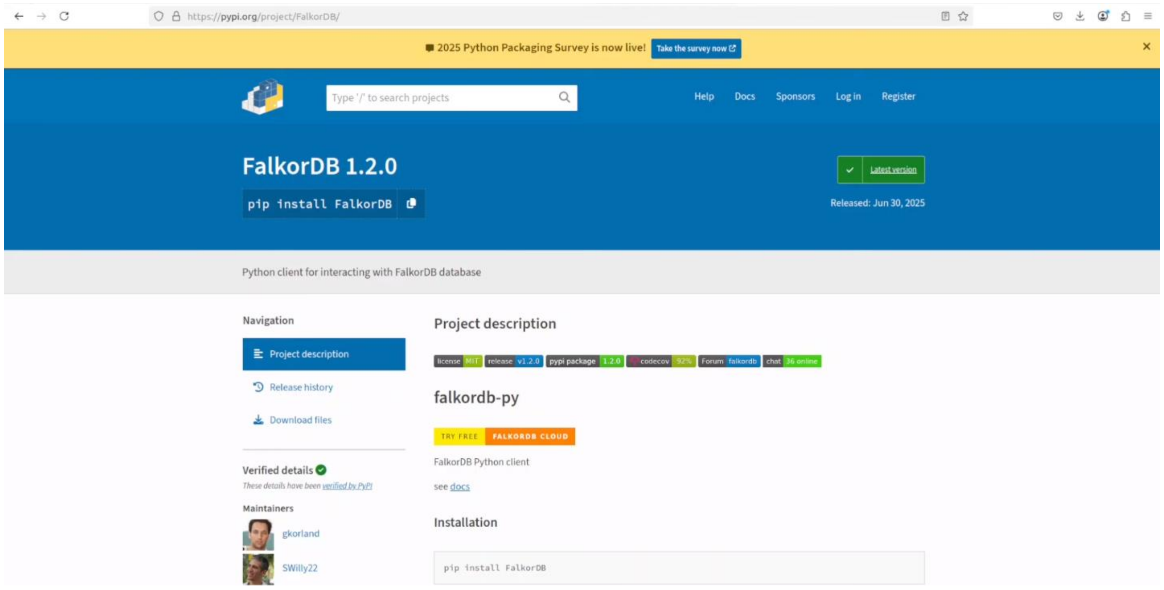


```python
result = company_graph.query("""
    ORDER BY
        CASE c.frequency
            WHEN 'Monthly' THEN 1
        END DESC
""")

for row in result.result_set:
    print(f"  🤝 {row[0]} ↔ {row[1]} | {row[2]} ({row[3]})")

# Query 6: Advanced analytics - skill gaps analysis
print("\n🔍 Query 6: Departments that might need more AI expertise:")
result = company_graph.query("""
    MATCH (d:Department)<-[:WORKS_IN]-(e:Employee)
    OPTIONAL MATCH (e)-[:HAS_SKILL]->(ai_skill:Skill)
    WHERE ai_skill.category IN ['ML Framework', 'AI Domain']
    WITH d, count(DISTINCT e) as total_employees, count(DISTINCT ai_skill) as ai_skills
    RETURN d.name, total_employees, ai_skills,
            round(toFloat(ai_skills) / toFloat(total_employees) * 100) as ai_skill_percentage
    ORDER BY ai_skill_percentage ASC
""")

for row in result.result_set:
    print(f"  📊 {row[0]}: {row[2]} AI skills / {row[1]} employees = {row[3]}% AI coverage")

print("\n✨ Graph analysis complete! This demonstrates FalkorDB's power in:")
print("   • Complex relationship querying")
print("   • Multi-hop path traversals")
print("   • Aggregations across connected data")
print("   • Real-time analytics on interconnected entities")
```



```python
result = company_graph.query("""
    ORDER BY
        END DESC
""")

for row in result.result_set:
    print(f"  🤝 {row[0]} ↔ {row[1]} | {row[2]} ({row[3]})")

# Query 6: Advanced analytics - skill gaps analysis
print("\n🔍 Query 6: Departments that might need more AI expertise:")
result = company_graph.query("""
    MATCH (d:Department)<-[:WORKS_IN]-(e:Employee)
    OPTIONAL MATCH (e)-[:HAS_SKILL]->(ai_skill:Skill)
    WHERE ai_skill.category IN ['ML Framework', 'AI Domain']
    WITH d, count(DISTINCT e) as total_employees, count(DISTINCT ai_skill) as ai_skills
    RETURN d.name, total_employees, ai_skills,
            round(toFloat(ai_skills) / toFloat(total_employees) * 100) as ai_skill_percentage
    ORDER BY ai_skill_percentage ASC
""")

for row in result.result_set:
    print(f"  📊 {row[0]}: {row[2]} AI skills / {row[1]} employees = {row[3]}% AI coverage")

print("\n✨ Graph analysis complete! This demonstrates FalkorDB's power in:")
print("   • Complex relationship querying")
print("   • Multi-hop path traversals")
print("   • Aggregations across connected data")
print("   • Real-time analytics on interconnected entities")

```

```
⊗ 0 ⚠ 0    Ln 11, Col 9 (14 selected)   Spaces: 4   UTF-8   LF   {} Python
```



https://pypi.org/project/FalkorDB/

2025 Python Packaging Survey is now live! Take the survey now

Help Docs Sponsors Log in Register

FalkorDB 1.2.0

```
pip install FalkorDB
```

Latest version

Released: Jun 30, 2025

Python client for interacting with FalkorDB database

Navigation

Project description

Release history

Download files

Verified details

These details have been verified by PyPI

Maintainers

gkorland

SWilly22

Project description

license MIT | release v1.2.0 | pypi package 1.2.0 | codecov 92% | Forum falkordb | chat 36 online

falkordb-py

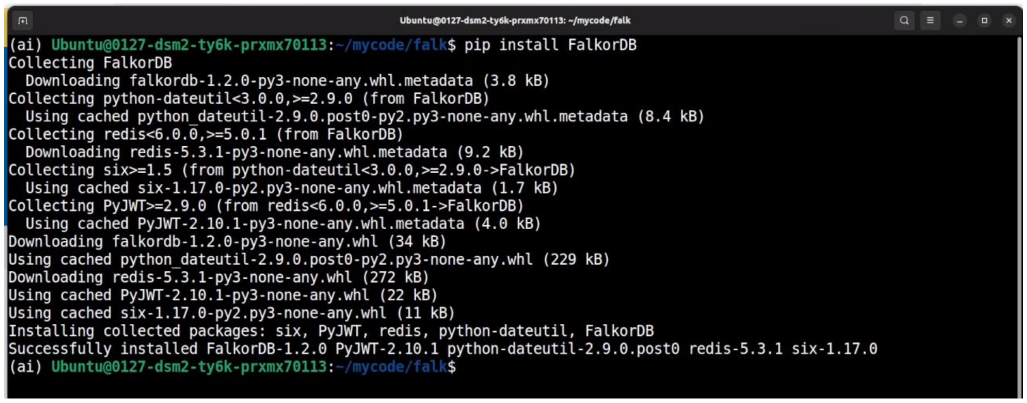
TRY FREE FALKORDB CLOUD

FalkorDB Python client

see docs

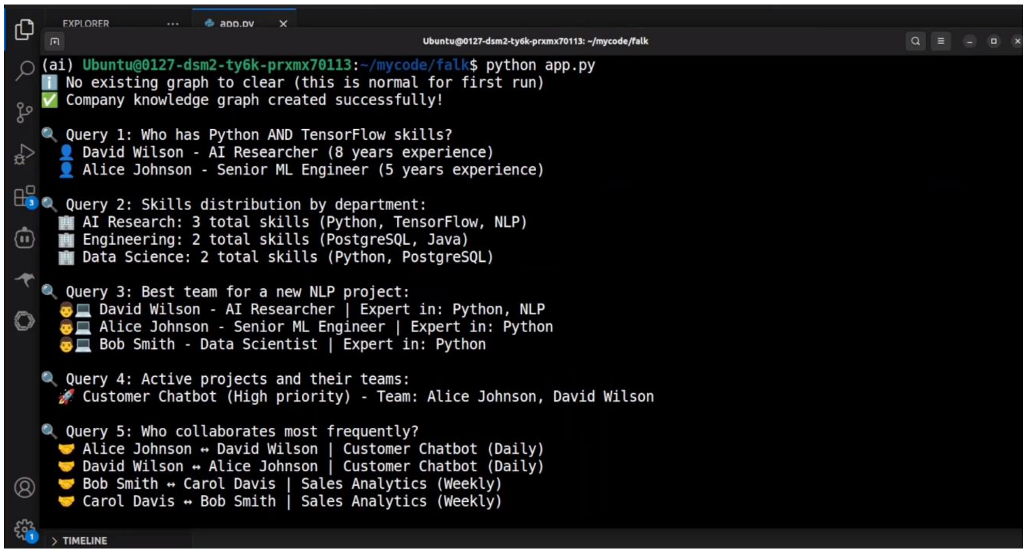
Installation

```
pip install FalkorDB
```



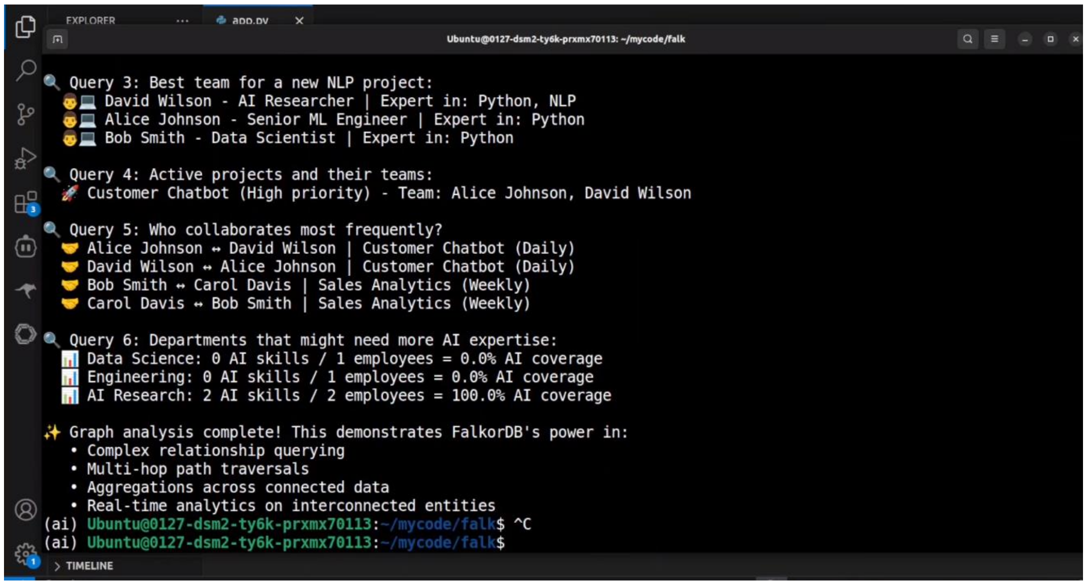
Ubuntu@0127-dsm2-ty6k-prxmx70113: ~/mycode/falk

```
(ai) Ubuntu@0127-dsm2-ty6k-prxmx70113:~/mycode/falk$ pip install FalkorDB
Collecting FalkorDB
  Downloading falkordb-1.2.0-py3-none-any.whl.metadata (3.8 kB)
Collecting python-dateutil<3.0.0,>=2.9.0 (from FalkorDB)
  Using cached python_dateutil-2.9.0.post0-py2.py3-none-any.whl.metadata (8.4 kB)
Collecting redis<6.0.0,>=5.0.1 (from FalkorDB)
  Downloading redis-5.3.1-py3-none-any.whl.metadata (9.2 kB)
Collecting six>=1.5 (from python-dateutil<3.0.0,>=2.9.0->FalkorDB)
  Using cached six-1.17.0-py2.py3-none-any.whl.metadata (1.7 kB)
Collecting PyJWT>=2.9.0 (from redis<6.0.0,>=5.0.1->FalkorDB)
  Using cached PyJWT-2.10.1-py3-none-any.whl.metadata (4.0 kB)
Downloading falkordb-1.2.0-py3-none-any.whl (34 kB)
Using cached python_dateutil-2.9.0.post0-py2.py3-none-any.whl (229 kB)
Downloading redis-5.3.1-py3-none-any.whl (272 kB)
Using cached PyJWT-2.10.1-py3-none-any.whl (22 kB)
Using cached six-1.17.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: six, PyJWT, redis, python-dateutil, FalkorDB
Successfully installed FalkorDB-1.2.0 PyJWT-2.10.1 python-dateutil-2.9.0.post0 redis-5.3.1 six-1.17.0
(ai) Ubuntu@0127-dsm2-ty6k-prxmx70113:~/mycode/falk$
```

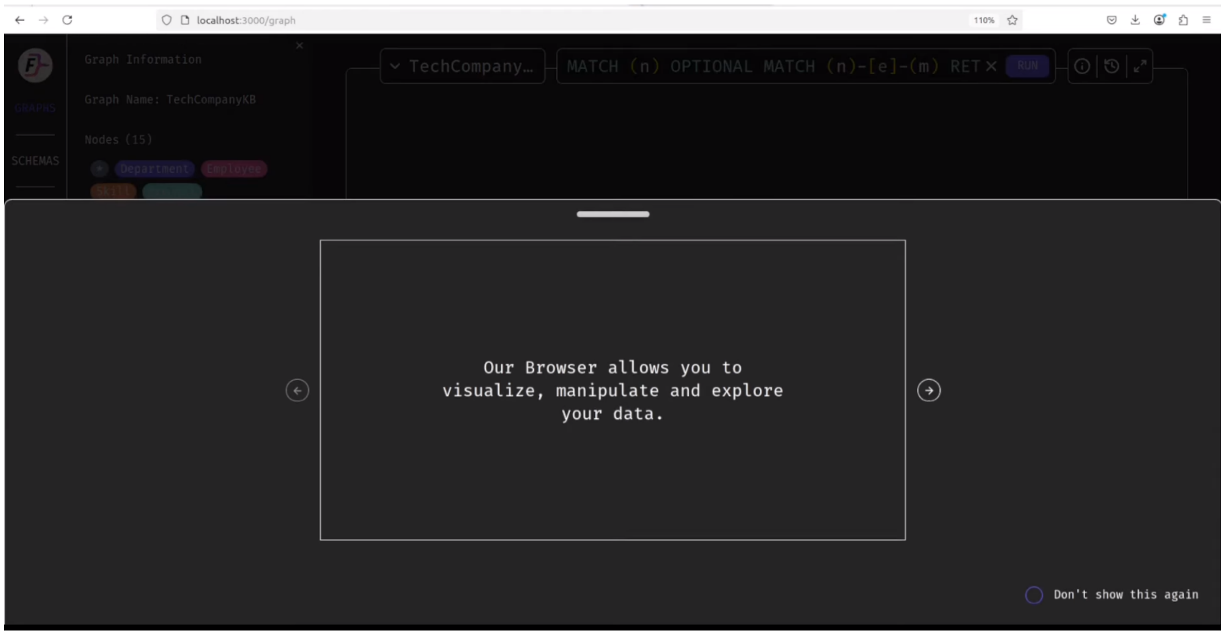


Ubuntu@0127-dsm2-ty6k-prxmx70113: ~/mycode/falk

```
(ai) Ubuntu@0127-dsm2-ty6k-prxmx70113:~/mycode/falk$ python app.py
ℹ️ No existing graph to clear (this is normal for first run)
✅ Company knowledge graph created successfully!

🔍 Query 1: Who has Python AND TensorFlow skills?
  👤 David Wilson - AI Researcher (8 years experience)
  👤 Alice Johnson - Senior ML Engineer (5 years experience)

🔍 Query 2: Skills distribution by department:
  🏢 AI Research: 3 total skills (Python, TensorFlow, NLP)
  🏢 Engineering: 2 total skills (PostgreSQL, Java)
  🏢 Data Science: 2 total skills (Python, PostgreSQL)

🔍 Query 3: Best team for a new NLP project:
  👨‍💻 David Wilson - AI Researcher | Expert in: Python, NLP
  👨‍💻 Alice Johnson - Senior ML Engineer | Expert in: Python
  👨‍💻 Bob Smith - Data Scientist | Expert in: Python

🔍 Query 4: Active projects and their teams:
  🚀 Customer Chatbot (High priority) - Team: Alice Johnson, David Wilson

🔍 Query 5: Who collaborates most frequently?
  🤝 Alice Johnson ↔ David Wilson | Customer Chatbot (Daily)
  🤝 David Wilson ↔ Alice Johnson | Customer Chatbot (Daily)
  🤝 Bob Smith ↔ Carol Davis | Sales Analytics (Weekly)
  🤝 Carol Davis ↔ Bob Smith | Sales Analytics (Weekly)
```

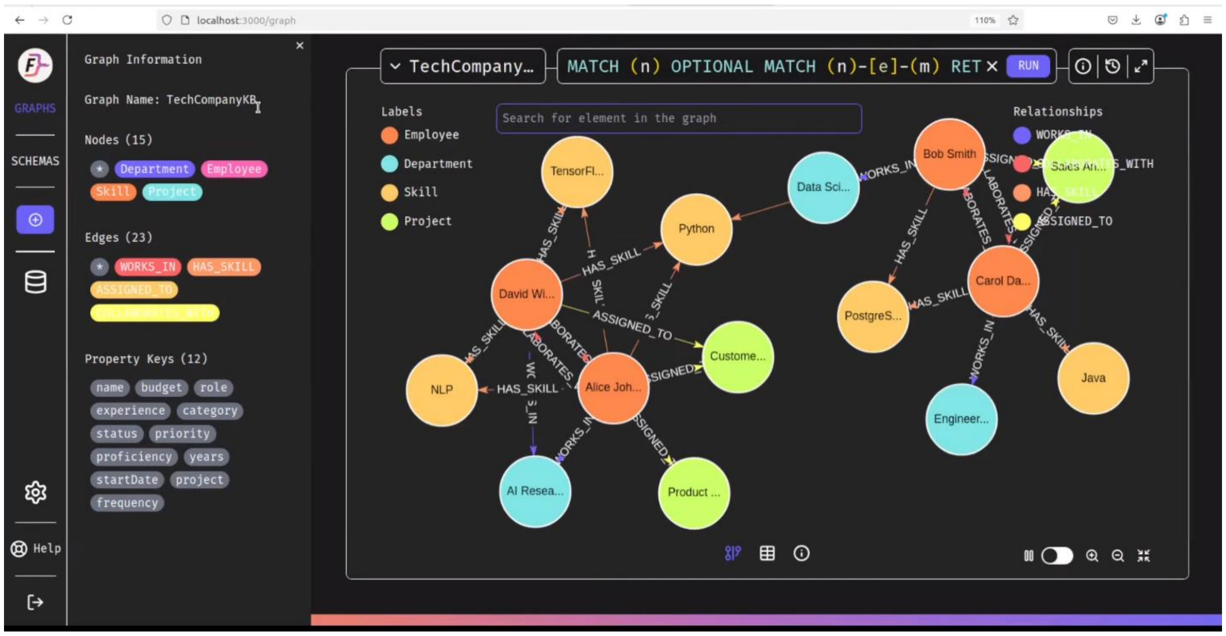


```
🔍 Query 3: Best team for a new NLP project:
  👨‍💻 David Wilson - AI Researcher | Expert in: Python, NLP
  👨‍💻 Alice Johnson - Senior ML Engineer | Expert in: Python
  👨‍💻 Bob Smith - Data Scientist | Expert in: Python

🔍 Query 4: Active projects and their teams:
  🚀 Customer Chatbot (High priority) - Team: Alice Johnson, David Wilson

🔍 Query 5: Who collaborates most frequently?
  🤝 Alice Johnson ↔ David Wilson | Customer Chatbot (Daily)
  🤝 David Wilson ↔ Alice Johnson | Customer Chatbot (Daily)
  🤝 Bob Smith ↔ Carol Davis | Sales Analytics (Weekly)
  🤝 Carol Davis ↔ Bob Smith | Sales Analytics (Weekly)

🔍 Query 6: Departments that might need more AI expertise:
  📊 Data Science: 0 AI skills / 1 employees = 0.0% AI coverage
  📊 Engineering: 0 AI skills / 1 employees = 0.0% AI coverage
  📊 AI Research: 2 AI skills / 2 employees = 100.0% AI coverage

✨ Graph analysis complete! This demonstrates FalkorDB's power in:
  • Complex relationship querying
  • Multi-hop path traversals
  • Aggregations across connected data
  • Real-time analytics on interconnected entities
(ai) Ubuntu@0127-dsm2-ty6k-prxmx70113:~/mycode/falk$ ^C
(ai) Ubuntu@0127-dsm2-ty6k-prxmx70113:~/mycode/falk$
```



Our Browser allows you to
visualize, manipulate and explore
your data.

Don't show this again



Graph Information

Graph Name: TechCompanyKB

Nodes (15)

Department Employee Skill Project

Edges (23)

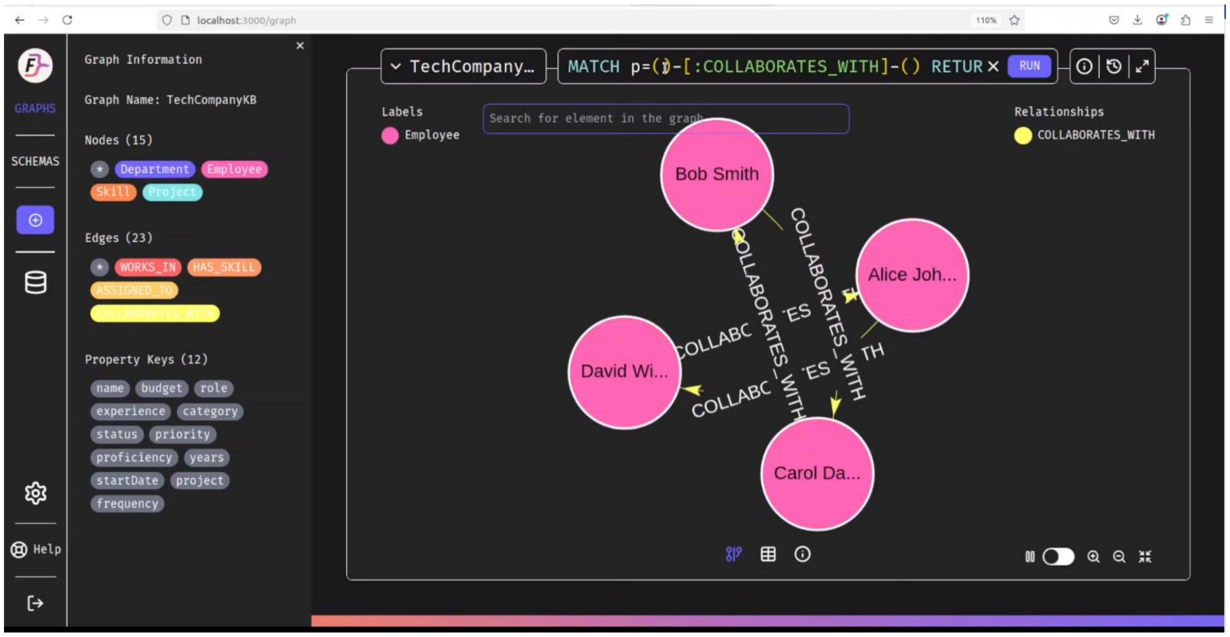
WORKS_IN HAS_SKILL ASSIGNED_TO COLLABORATES_WITH

Property Keys (12)

name budget role experience category status priority proficiency years startDate project frequency

Labels
- ● Employee
- ● Department
- ● Skill
- ● Project

Relationships
- ● WORKS_IN
- ● COLLABORATES_WITH
- ● HAS_SKILL
- ● ASSIGNED_TO


localhost:3000/graph

Graph Information

Graph Name: TechCompanyKB

Nodes (15)

* Department Employee Skill Project

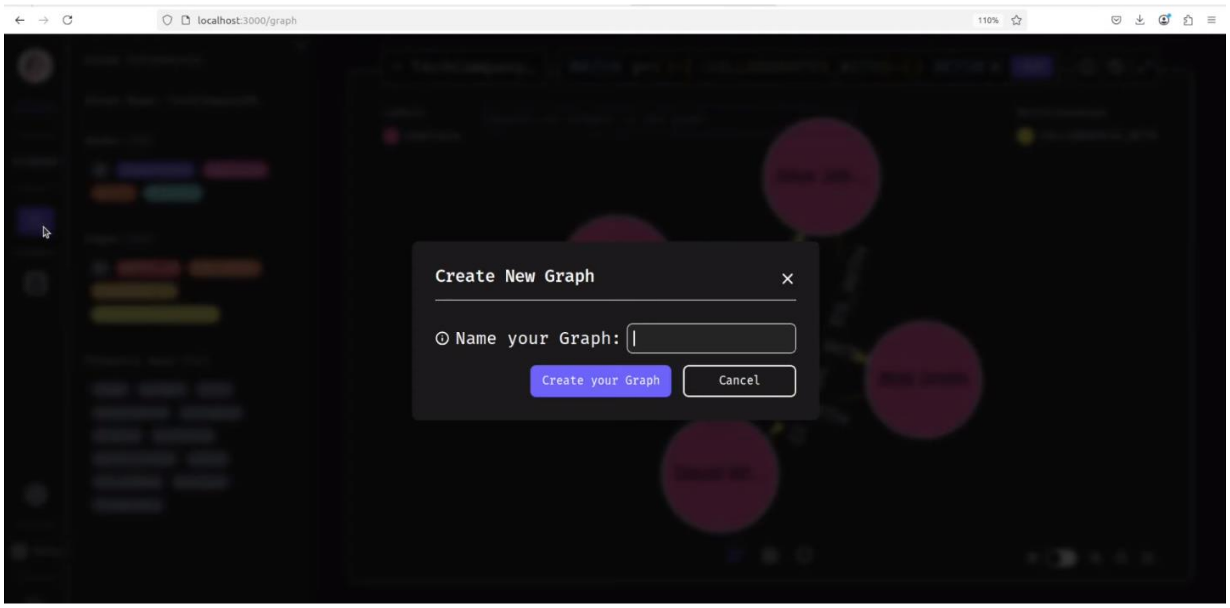
Edges (23)

* WORKS_IN HAS_SKILL ASSIGNED_TO COLLABORATES_WITH

Property Keys (12)

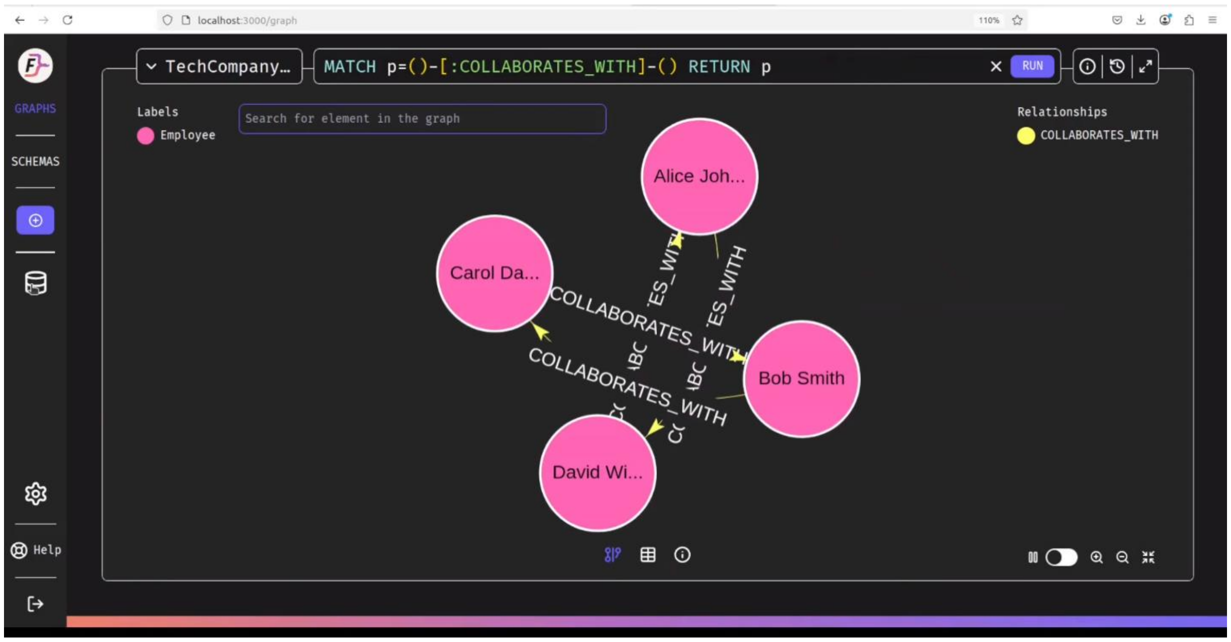
name budget role experience category status priority proficiency years startDate project frequency

```
MATCH p=()-[:COLLABORATES_WITH]-() RETUR
```


localhost:3000/graph

Create New Graph

Name your Graph:


localhost:3000/graph

```
MATCH p=()-[:COLLABORATES_WITH]-() RETURN p
```