1. Create a Project Folder:

- Create a folder named `syndicate-hunter`.
- Open VS Code in this folder.

2. Install Dependencies: Open your terminal in VS Code and type:

Bash
```
pip install pandas faker
```

3. The Script (`data_gen.py`): Create a new file called `data_gen.py` and paste this code.

Python
```
# data_gen.py
# ------------------------------------------------------
# GOAL: Generate banking data with a hidden money laundering ring.
# TECH: Python (Pandas) for data manipulation.
# MATH CONCEPT: Graph Theory (Nodes = People, Edges = Transactions).
# ------------------------------------------------------

import pandas as pd
import random
from faker import Faker

# Initialize the Fake Name generator
fake = Faker()

# ==========================================
# STEP 1: GENERATE ACCOUNTS (The "Nodes")
# ==========================================
print("1. Generating 100 Bank Accounts...")

accounts = []
# We create IDs from 101 to 200
for i in range(101, 201):
    accounts.append({
        "client_id": i,
        "name": fake.name(),
        "risk_score": random.randint(10, 30),  # Random low risk score
        "location": "Gauteng"
    })

# Convert to a DataFrame (Table) and save
df_accounts = pd.DataFrame(accounts)
df_accounts.to_csv("accounts.csv", index=False)
print("-> Success! Saved 'accounts.csv' (The Nodes)")


# ==========================================
# STEP 2: GENERATE NORMAL TRANSACTIONS (Noise)
# ==========================================
print("2. Generating 500 Legitimate Transactions...")

transactions = []

for _ in range(500):
    # Pick two random people
    sender = random.choice(accounts)["client_id"]
```

```
    receiver = random.choice(accounts)["client_id"]

    # Rule: You can't send money to yourself
    if sender != receiver:
        transactions.append({
            "from_id": sender,
            "to_id": receiver,
            "amount": round(random.uniform(50, 2000), 2), # Random amounts
R50 - R2000
            "type": "Legit",
            "date": fake.date_this_year()
        })

# ========================================
# STEP 3: INJECT THE FRAUD RING (The "Syndicate")
# ========================================
# We pick 5 specific IDs to be our criminals.
# They will move a large sum of money in a perfect circle.
# Path: 105 -> 110 -> 115 -> 120 -> 125 -> 105 (Loop)

print("3. Injecting the Hidden Fraud Ring...")

syndicate_members = [105, 110, 115, 120, 125]
large_amount = 50000.00  # R50,000

# Loop through the members and make them send to the next person
for i in range(len(syndicate_members)):
    sender = syndicate_members[i]
    # The '%' operator ensures the last person sends back to the first
person (Graph Cycle)
    receiver = syndicate_members[(i + 1) % len(syndicate_members)]

    # Layering: We change the amount slightly so it looks "natural" to SQL
    # e.g., R50,000 -> R49,500 -> R49,200
    cleaned_amount = large_amount - (random.uniform(0, 500))

    transactions.append({
        "from_id": sender,
        "to_id": receiver,
        "amount": round(cleaned_amount, 2),
        "type": "Structuring", # 'Structuring' is a money laundering term
        "date": "2026-02-18"   # All happens on the same day
    })

# ========================================
# STEP 4: SAVE EVERYTHING
# ========================================
df_tx = pd.DataFrame(transactions)
df_tx.to_csv("transactions.csv", index=False)
print(f"-> Success! Saved 'transactions.csv' with {len(df_tx)} rows.")
print("-> The Hidden Ring is: 105 -> 110 -> 115 -> 120 -> 125 -> 105")
```

4. Run It: Run the script in your terminal:

Bash
```
python data_gen.py
```

You will now have `accounts.csv` and `transactions.csv` in your folder.

Phase 2: The System (Neo4j Setup)

Now we act as the **Systems Engineer**. We need to load this data into a Graph Database.

1. Download Neo4j Desktop:

- Go to neo4j.com/download and download **Neo4j Desktop** (Free Personal Edition).
- Install and open it.

2. Create a Database:

- In Neo4j Desktop, click **"New Project"**.
- Click **"Add" -> "Local DBMS"**.
- Name: `SyndicateHunter`.
- Password: `password` (Keep it simple for local dev).
- Click **Create** and then **Start**.

3. Import the Files:

- Hover over your generic `SyndicateHunter` database card.
- Click the **"..."** (three dots) -> **"Open folder"** -> **"Import"**.
- **Drag and drop** your `accounts.csv` and `transactions.csv` into this folder.
- *Note: Neo4j specifically looks in this folder for security reasons.*

Phase 3: The Data Loading (Cypher Code)

Now we act as the **Database Administrator**. We use Cypher (Graph SQL) to load the data.

1. Open the Browser:

- In Neo4j Desktop, click the blue **"Open"** button (Neo4j Browser).
- You will see a command prompt `$`.

2. Load the Nodes (People): Copy and paste this code into the browser and hit **Play (►)**.

Cypher

```
// 1. LOAD ACCOUNTS
// "LOAD CSV" reads the file line by line
// "MERGE" ensures we don't create duplicates (like SQL Upsert)
LOAD CSV WITH HEADERS FROM 'file:///accounts.csv' AS row
MERGE (c:Client {id: toInteger(row.client_id)})
SET c.name = row.name,
    c.risk_score = toInteger(row.risk_score),
    c.location = row.location;
```

3. Load the Edges (Transactions): Paste this and hit **Play**.

Cypher
```
// 2. LOAD TRANSACTIONS
// We match the Sender (c1) and Receiver (c2) by ID
// Then we create a relationship [:SENT_MONEY] between them
LOAD CSV WITH HEADERS FROM 'file:///transactions.csv' AS row
MATCH (c1:Client {id: toInteger(row.from_id)})
MATCH (c2:Client {id: toInteger(row.to_id)})
CREATE (c1)-[t:SENT_MONEY {
    amount: toFloat(row.amount),
    date: row.date,
    type: row.type
}]->(c2);
```

Phase 4: The Hunt (Analysis)

Now we act as the **Investigator / Data Scientist**. We write the query that catches the criminals.

1. The "Naive" SQL Approach (Optional): If you run this, you just see a mess of data.

Cypher
```
MATCH (n)-[r]->(m) RETURN n, r, m LIMIT 25
```

2. The Syndicate Hunter Algorithm (The Winner): This is the "Math" query. It looks for a **Closed Loop** (Cycle) of exactly 5 people where the amount is large (> 40,000).

Paste this and hit **Play**:

Cypher
```
// FIND MONEY LAUNDERING RINGS
// Pattern: (a) -> (b) -> (c) -> (d) -> (e) -> (a)
// This creates a closed circle of length 5

MATCH path = (a)-[r1]->(b)-[r2]->(c)-[r3]->(d)-[r4]->(e)-[r5]->(a)

// FILTER: Only look for high-value transactions
WHERE r1.amount > 40000
  AND r2.amount > 40000
  AND r3.amount > 40000
  AND r4.amount > 40000
  AND r5.amount > 40000

// RESULT: Return the names and the full path
RETURN a.name AS Kingpin, path
```

3. The Result:

- You will see a **Visual Graph** on the screen.
- 5 dots (Nodes) will be arranged in a perfect circle/pentagon.
- These are your criminals (IDs 105, 110, 115, 120, 125).

Phase 5: How to Submit This

1. **Screenshots:** Take a screenshot of the Neo4j Graph Result (the circle). This is your "Money Shot."
2. **GitHub:** Upload `data_gen.py`, `accounts.csv`, and a text file called `queries.cypher` (containing the code from Phase 3 & 4).
3. **Readme.md:** write this:

    Syndicate Hunter: Financial Crime Detection System

    - **Tech:** Python (Pandas), Neo4j (Graph Database), Cypher Query Language.
    - **Goal:** Detect money laundering rings that traditional SQL queries miss.
    - **Method:** Used Graph Theory (Cycle Detection) to identify circular transaction flows > R40k.

You are done. You have now built a specialized banking forensics tool.