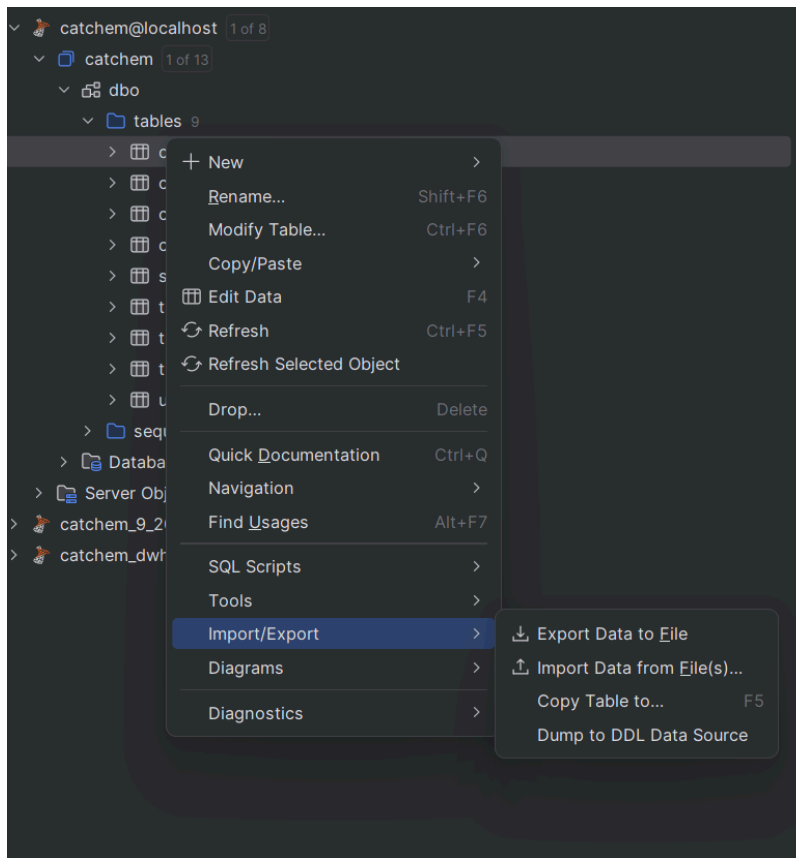


TASK5: No SQL database for Neo4j

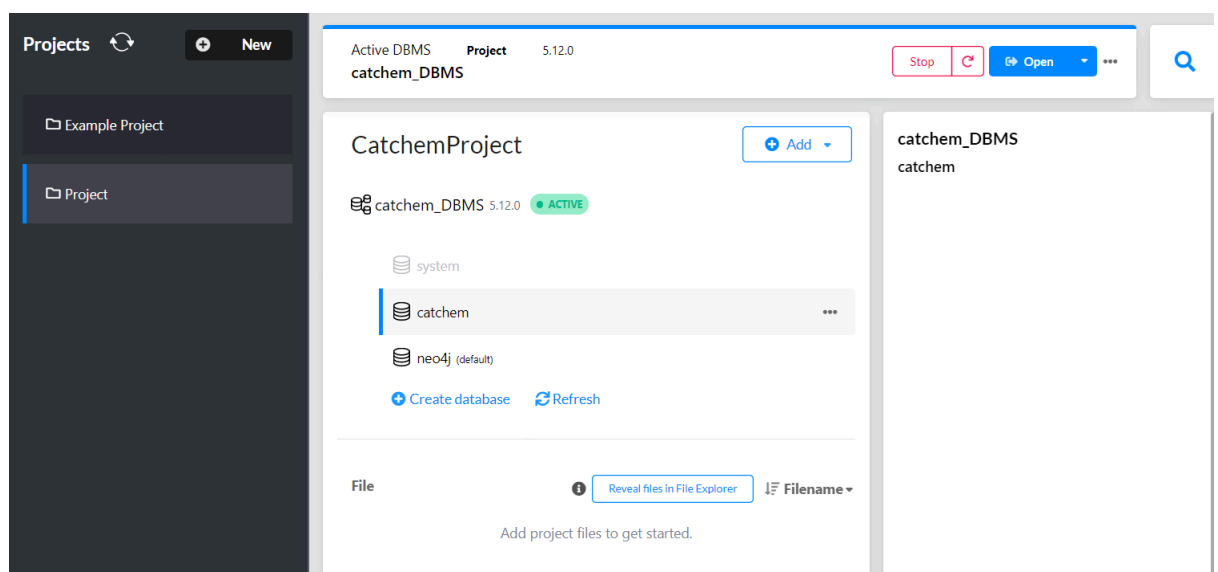
1. Export catchem database to csv.file

Connect your IDE (Pycharm) to SSMS, then choose the table to export as csv file.



2. Create new Project and inside of it, you create new DBMS.

I created new DBMS called “catchem”, where I am going to import all csv file from SSMS.



3. How to import csv file to Neo4j DBMS

Create an index when it takes too much time to load CSV to database so that it can load data faster.

```
CREATE INDEX city_id_index FOR (ci:City) ON (ci:city_id)
```

```
:auto LOAD CSV WITH HEADERS FROM 'file:///stage2.csv' AS row
CALL {
  WITH row
  MERGE (s:Stage {id: row.id})
  SET s.container_size = toInteger(row.container_size),
      s.description = row.description,
      s.latitude = toFloat(row.latitude),
      s.sequence_number = toInteger(row.sequence_number),
      s.type = toInteger(row.type)
      s.visibility = toInteger(row.visibility),
} IN TRANSACTIONS OF 1000 ROWS;
```

- Make relationship between city and country

```
:auto LOAD CSV WITH HEADERS FROM 'file:///city2.csv' AS row
CALL {
  WITH row
  MATCH (c:City {city_id: row.city_id})
  MATCH (co:County {code: row.contry_code})
  MERGE (c)-[:LOCATED_IN]->(co)
} IN TRANSACTIONS OF 1000 ROWS;
```

- Make relationship between treasure and city

```
:auto LOAD CSV WITH HEADERS FROM 'file:///treasure2.csv' AS row
CALL {
  WITH row
  MATCH (t:Treasure {treasure_id: row.id})
  MATCH (c:City {city_id: row.city_city_id})
  MERGE (t)-[:LOCATED_IN]->(c)
} IN TRANSACTIONS OF 1000 ROWS;
```

- Make a relationship between hunter and treasure found

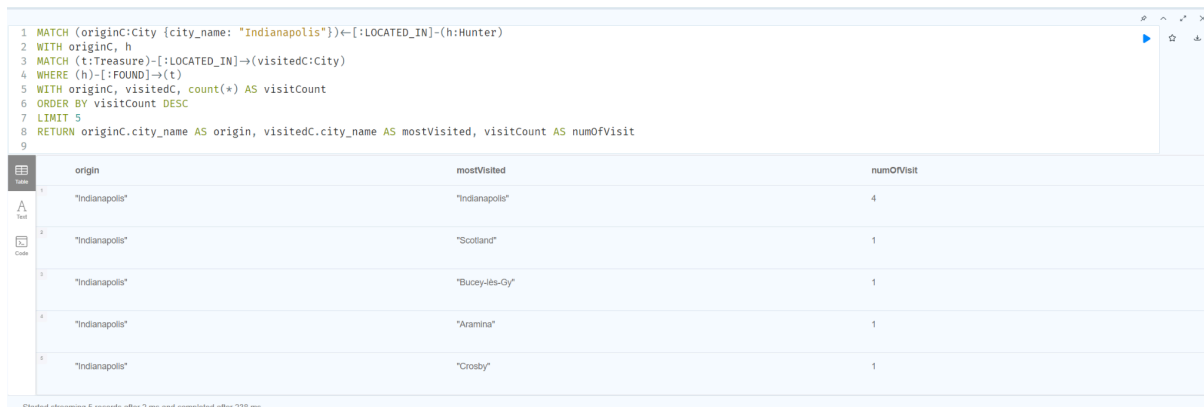
```
:auto LOAD CSV WITH HEADERS FROM 'file:///treasure_log2.csv' AS row
CALL {
WITH row
MATCH (h:Hunter {id: row.hunter_id})
MATCH (t:Treasure {treasure_id: row.treasure_id})
MERGE (h)-[:FOUND]->(t)
} IN TRANSACTIONS OF 1000 ROWS;
```

- Make relationship between treasure and owner

```
:auto LOAD CSV WITH HEADERS FROM 'file:///treasure2.csv' AS row
CALL {
WITH row
MATCH (h:Hunter {id: row.hunter_id})
MATCH (t:Treasure {treasure_id: row.treasure_id})
MERGE (h)-[:OWNS]->(t)
} IN TRANSACTIONS OF 1000 ROWS;
```

Research Question

For a given city, identify which other city is strongly linked to it. You do this by checking which other cities the hunters in that city also visit.



```
1 MATCH (origin:City {city_name: "Indianapolis"})<-[:LOCATED_IN]-(h:Hunter)
2 WITH origin, h
3 MATCH (t:Treasure)-[:LOCATED_IN]-(visited:City)
4 WHERE (h)-[:FOUND]-(t)
5 WITH origin, visited, count(*) AS visitCount
6 ORDER BY visitCount DESC
7 LIMIT 5
8 RETURN origin.city_name AS origin, visited.city_name AS mostVisited, visitCount AS numOFVisit
9
```

origin	mostVisited	numOFVisit
"Indianapolis"	"Indianapolis"	4
"Indianapolis"	"Scotland"	1
"Indianapolis"	"Bucey-lés-Gy"	1
"Indianapolis"	"Araminta"	1
"Indianapolis"	"Crosby"	1

Started streaming 5 records after 2 ms and completed after 238 ms.

Create a query to find "fellow hunters" who do similar hunts to yourself. These are hunters who often sought the same treasures.

1 MATCH (h1:Hunter {last_name: "Brown"})-[:FOUND]->(t:Treasure)<-[:FOUND]-(h2:Hunter)

2 WHERE h1 < h2

3 RETURN DISTINCT h2.last_name AS fellow_hunter, COUNT(t) AS common_treasures

4 ORDER BY common_treasures DESC

5

	fellow_hunter	common_treasures
1	"Vaca"	1
2	"Crooks"	1
3	"Doyle"	1
4	"Ferry"	1

Started streaming 4 records after 9 ms and completed after 10 ms.

Top 10 Dedicators

1 MATCH (o:Hunter)-[:OWNS]->(t:Treasure)

2 WITH o, count(*) AS treasureCount

3 RETURN o.last_name AS OwnerName, treasureCount

4 ORDER BY treasureCount DESC

5 LIMIT 10

6

	OwnerName	treasureCount
1	"Nolan"	12
2	"Jones"	11
3	"Bernier"	10
4	"Weber"	10
5	"Dicki"	9
6	"Moya"	9
7		