**Neo4j Work:** Importing a CSV file into Neo4j

**Prepared by:** Mert OLÇAMAN

**Week:** 4

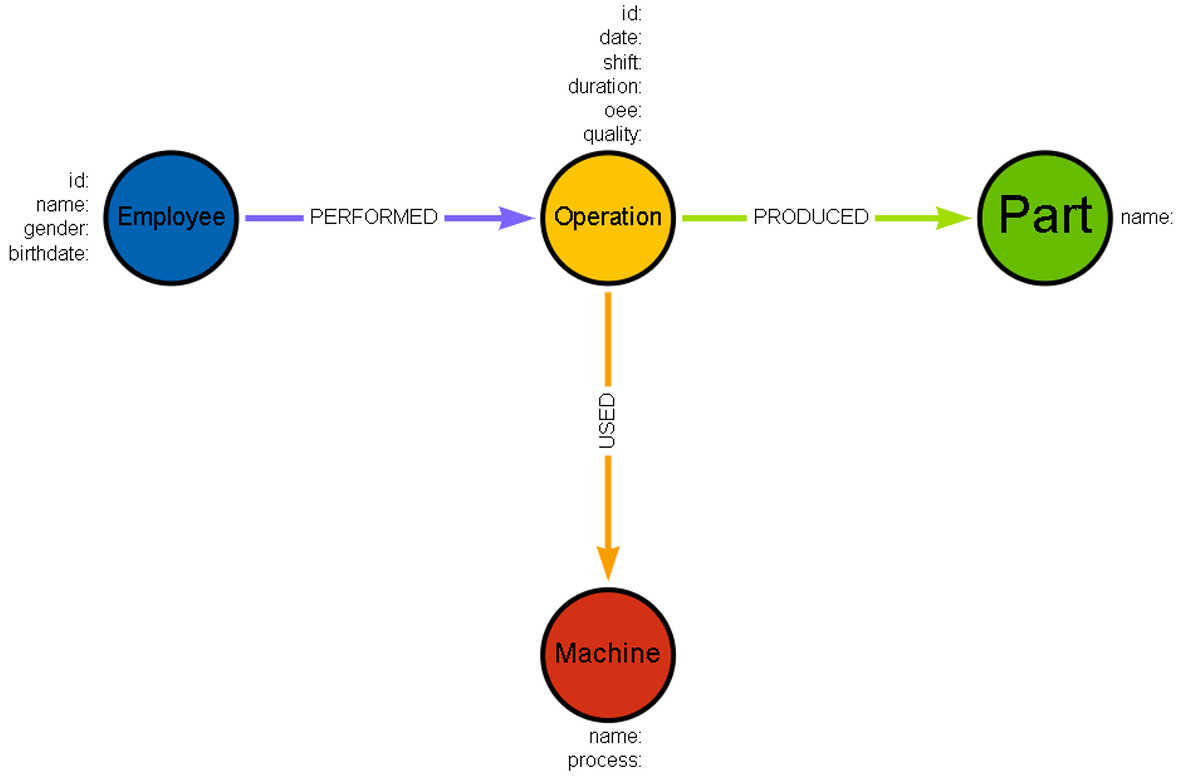
**patika.dev**

**NewMind AI Bootcamp**

**INTRODUCTION**

In this work, how to import a csv file into Neo4j Aura (cloud version) was shown in 2 ways. The dataset and the graph database design can be seen below respectively.





1. **through the link**

At first, the generated csv file was uploaded to github directory. Secondly, it was directly imported by using function.

**Step1 - Adding Constraints:** Before starting everything, constraints should be specified. Otherwise, there might be some improper nodes, which causes wrong results.

**// --- ADDING CONSTRAINTS --- \\**

**// Employee constraint**

CREATE CONSTRAINT employee\_id IF NOT EXISTS

FOR (e:Employee)

REQUIRE e.employee\_id IS UNIQUE;

**// Operation constraint**

CREATE CONSTRAINT operation\_id IF NOT EXISTS

FOR (o:Operation)

REQUIRE o.operation\_id IS UNIQUE;

**// Part constraint**

CREATE CONSTRAINT part\_name IF NOT EXISTS

FOR (p:Part)

REQUIRE p.part\_name IS UNIQUE;

**// Machine constraint**

CREATE CONSTRAINT machine\_name IF NOT EXISTS

FOR (m:Machine)

REQUIRE m.machine\_name IS UNIQUE;

**Step2 – Loading CSV Files & Creating Nodes with Properties & Adding Relationships:** While loading data from the link, an alias should be specified to use in the other parts of the cypher query. It was specifed as **data** in this work.

For each node created, the unique property has to be specified in the merge line as a property. As for Employee node, only employee\_id was set as unique property, so it was the only property which was provided in the merge, as it can be seen in the code cell below. If there had been multiple unique properties, they would have been specifed as well. Additionally, data alias was used to call the relevant data from the imported dataset by using the same column name as the imported dataset had. What it is meant here that EmployeeID was the name of the column in the dataset. It is valid for the other variables which start with the alias of data such as data.Name, data.Gender, and data.Shift.

On the other hand, the numerical values like duration must be converted into float by toFloat() function while calling the data from the csv file. The column of date was converted to date, as the format was only data (time not included).

**// --- LOADING CSV FILE --- \\**

LOAD CSV WITH HEADERS FROM 'https://media.githubusercontent.com/media/mertolcaman/patika\_newmind\_ai\_bootcamp/refs/heads/main/4.week/mydata.csv' as **data**

**// --- CREATING NODES WITH PROPERTIES--- \\**

**// Creating Employee Node**

MERGE (e:Employee {employee\_id: data.EmployeeID})

SET e.name = data.Name,

e.gender = data.Gender,

e.birthdate = data.DateOfBirth

**// Creating Operation Node**

MERGE (o:Operation {operation\_id: data.OperationID})

SET o.date = date(data.Date),

o.shift = data.Shift,

o.duration = toFloat(data.Duration),

o.oee = toFloat(data.OEE),

o.quality = data.Quality

**// Creating Machine Node**

MERGE (m:Machine {machine\_name: data.Machine})

SET m.process = data.Process

**// Creating Part Node**

MERGE (p:Part {part\_name: data.Part})

**// --- CREATING RELATIONSHIPS --- \\**

**// Creating PERFORMED Relationship: Employee -> Operation**

MERGE (e)-[:PERFORMED]->(o)

**// Creating USED Relationship: Operation -> Machine**

MERGE (o)-[:USED]->(m)

**// Creating PRODUCED Relationship: Operation -> Person**

MERGE (o)-[:PRODUCED]->(p);

**GRAPH DESIGN**

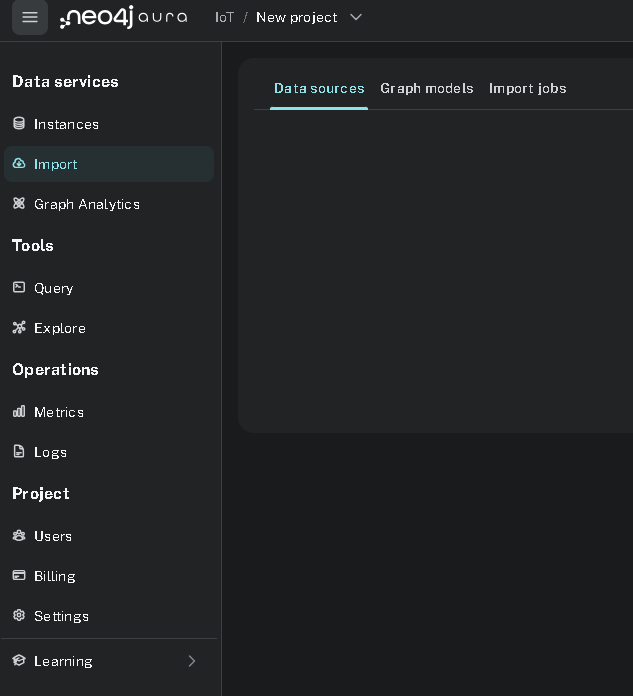
**TEST**

**QUESTIONS**

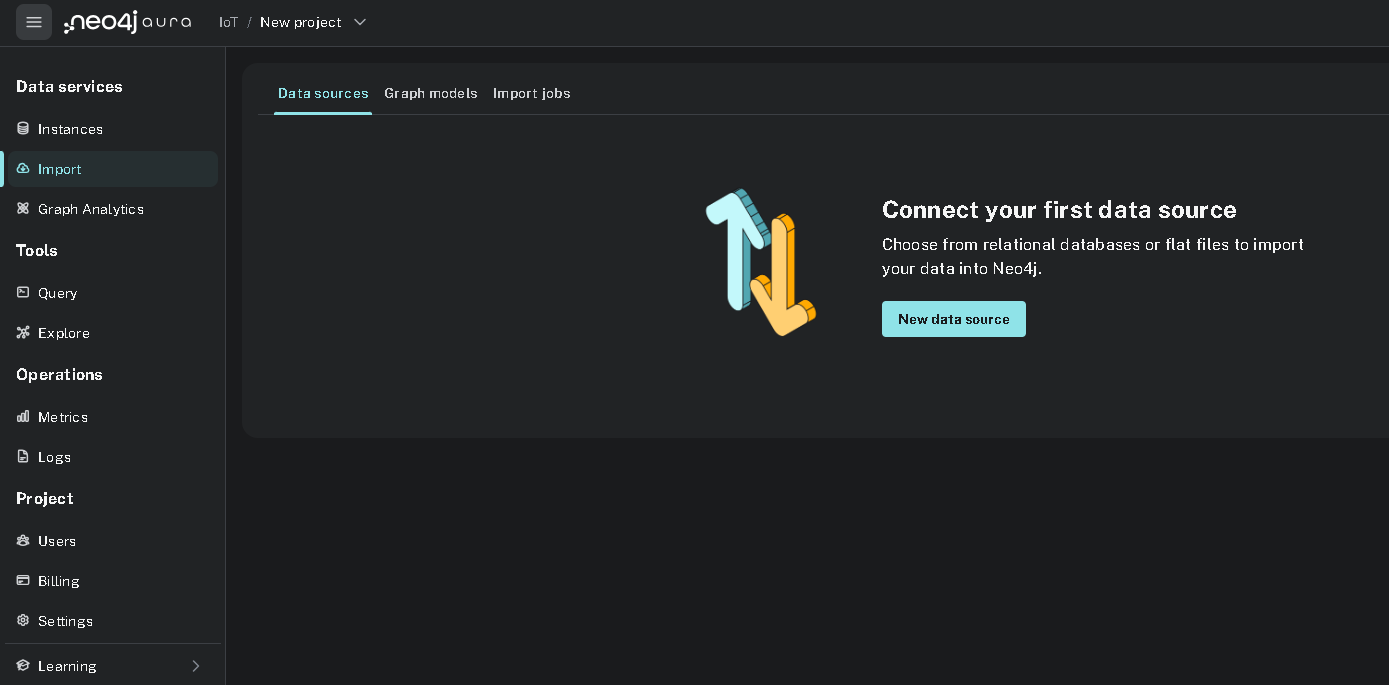
***Before starting, each question related to the query was specified on top of the query.***

1. **through the import section in Neo4j Aura**

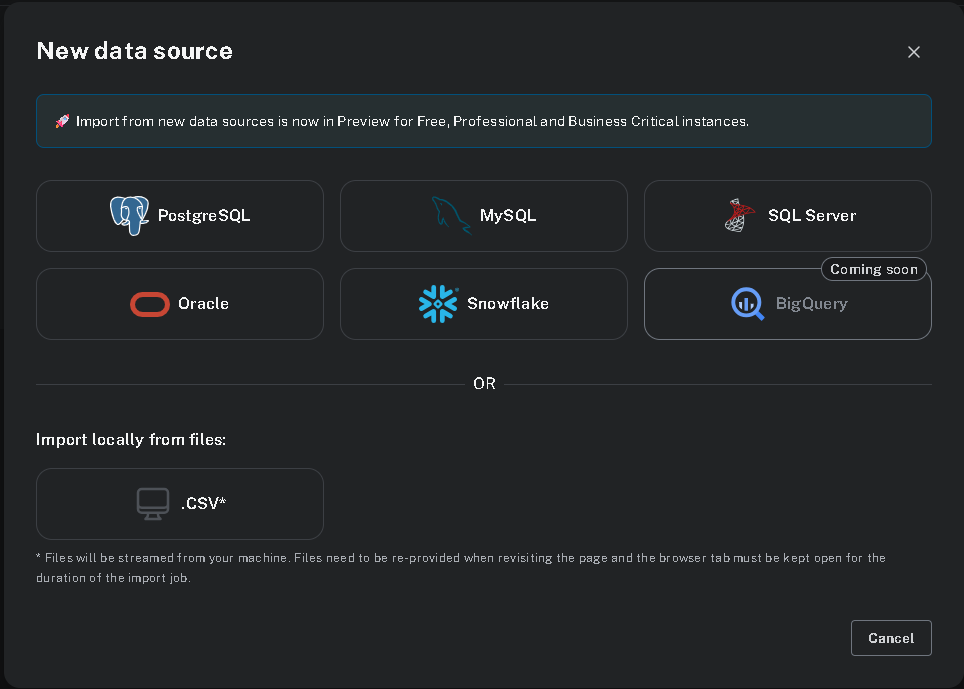
The second way to import the data into Neo4j is to use the import section in the menu left-hand side as it is shown below.



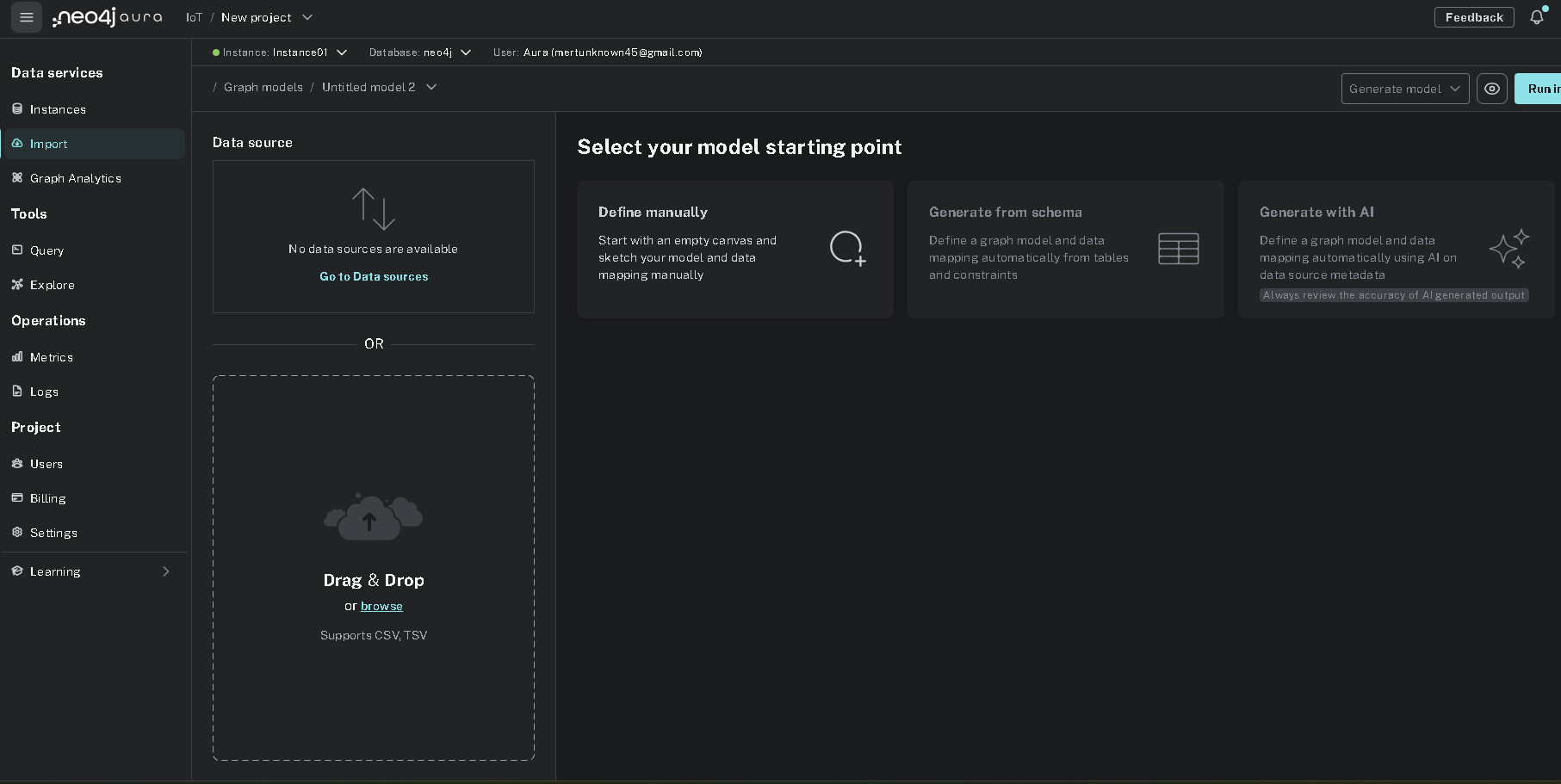
**Step-1:** After choosing import section, new data source is chosen.



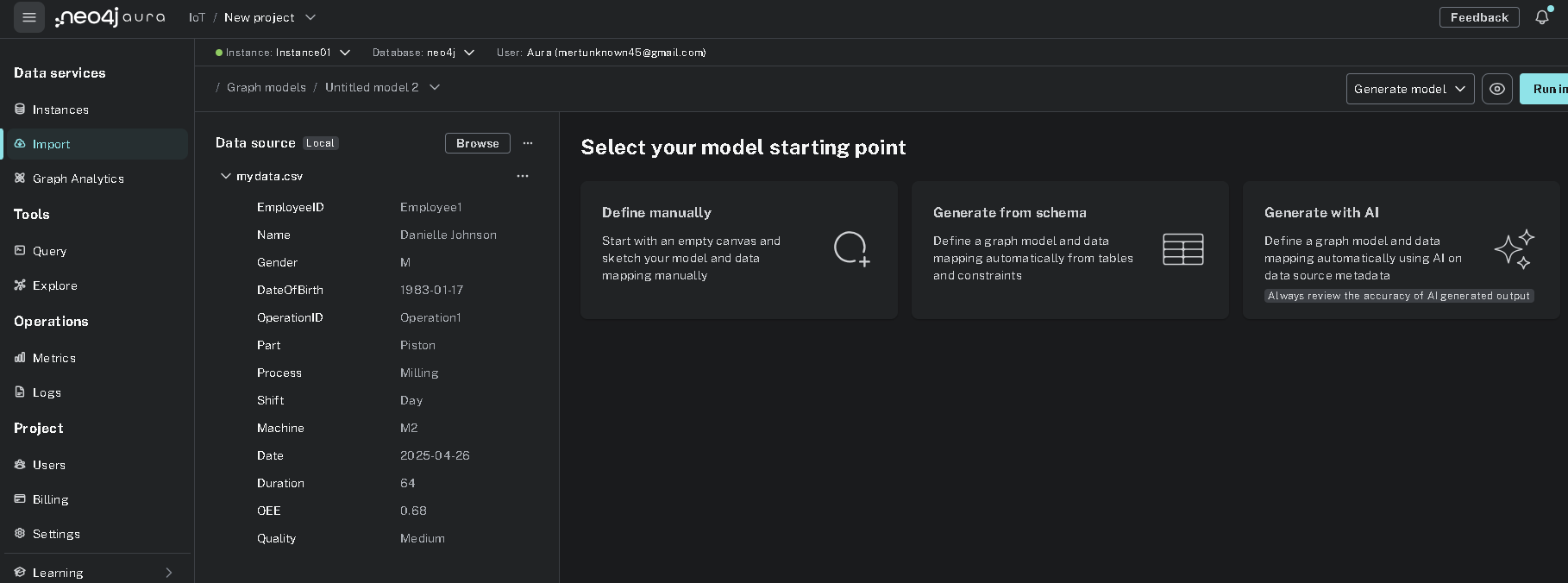
**Step-2:** Since the csv file would be uploaded, .csv\* option is chosen.



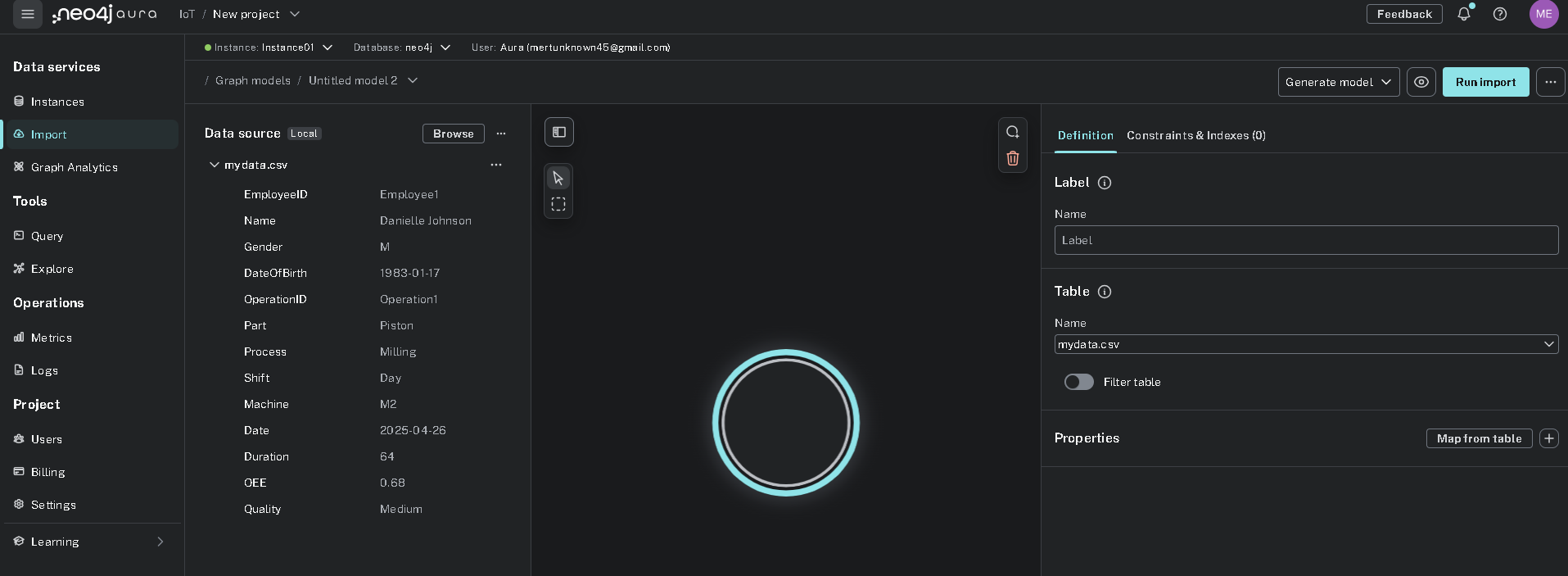
**Step-3:** Csv file which would like to be uploaded is found by either browsing or dragging into the window.



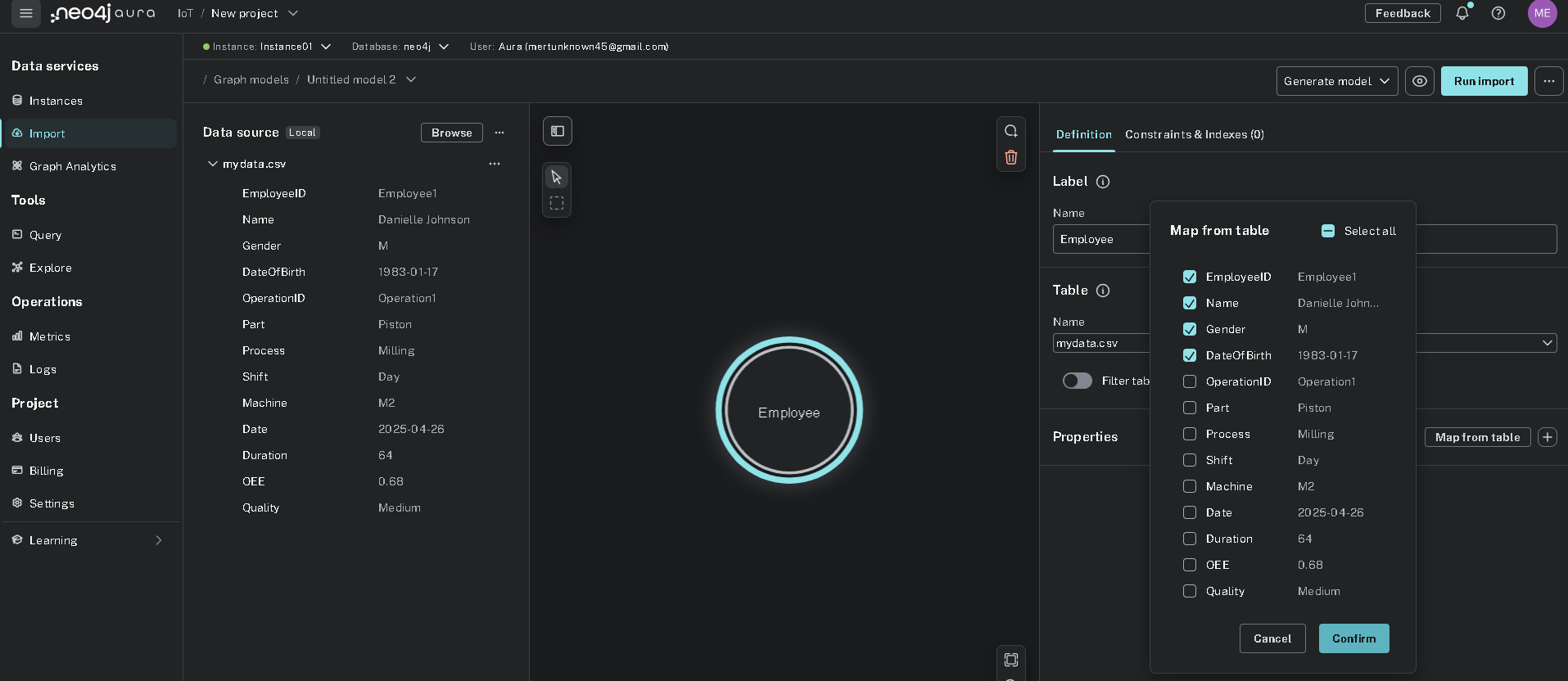
**Step-4:** Since all of the nodes, properties, and relationship would be specified manually, define manually option is clicked.



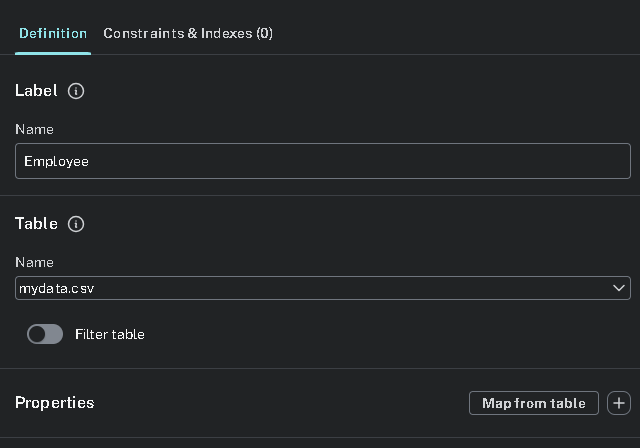
**Step-5:** A screen like below appears. A sample row is shown in the left-hand side. Label name is provided through the window positioned on the right. Because of having only 1 csv file, there is no need of changing the table name by drop-down menu.



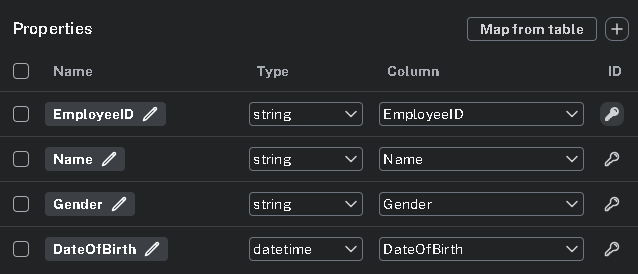
**Step-6:** When it comes to the properties of the node, they can be chosen from pop-up window, after clicking Map from table button. All of the column names related to Employee are checked.



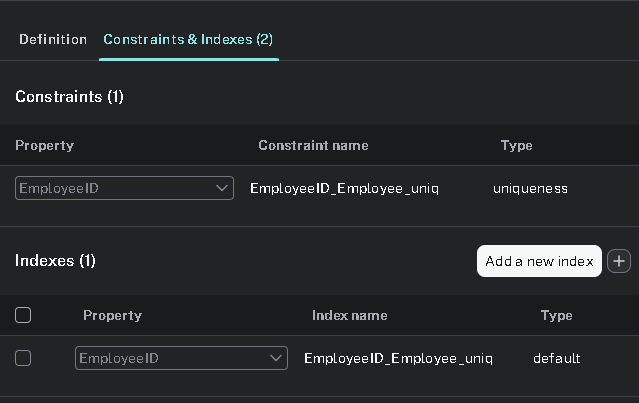
**Step-7:** Label name is chosen as relevant keyword, in other words, Employee here.



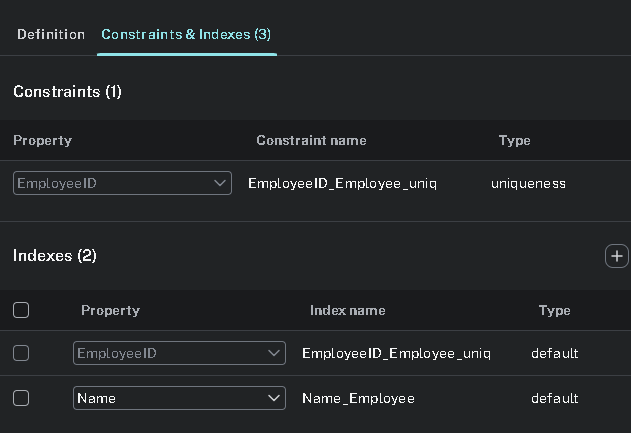
**Step-8:** After that, each property type is checked. If there is any wrong ones, it is corrected. Also, ID should be unique, so EmployeeID is selected by clicking the key button on the right.



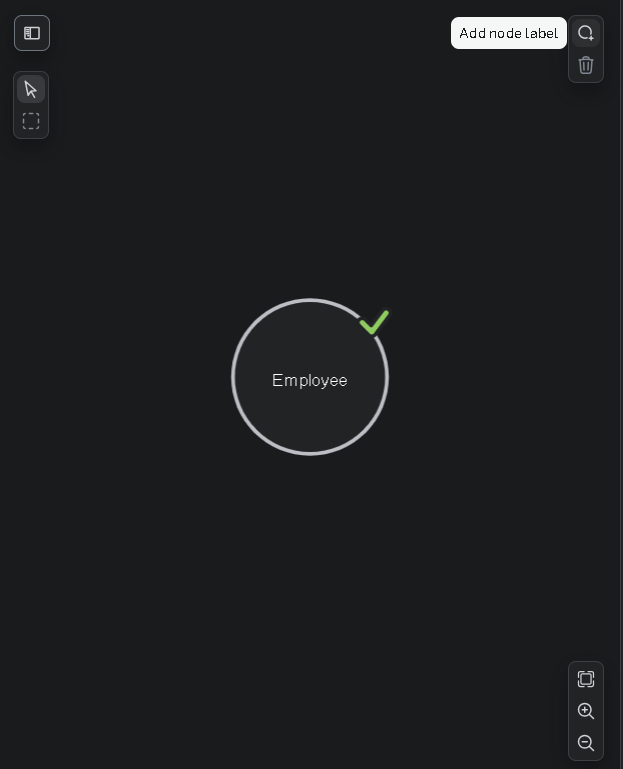
**Step-9:** In Constraints & Indexes tab, plus button (Add a new index) is clicked to add another index as name for employees.



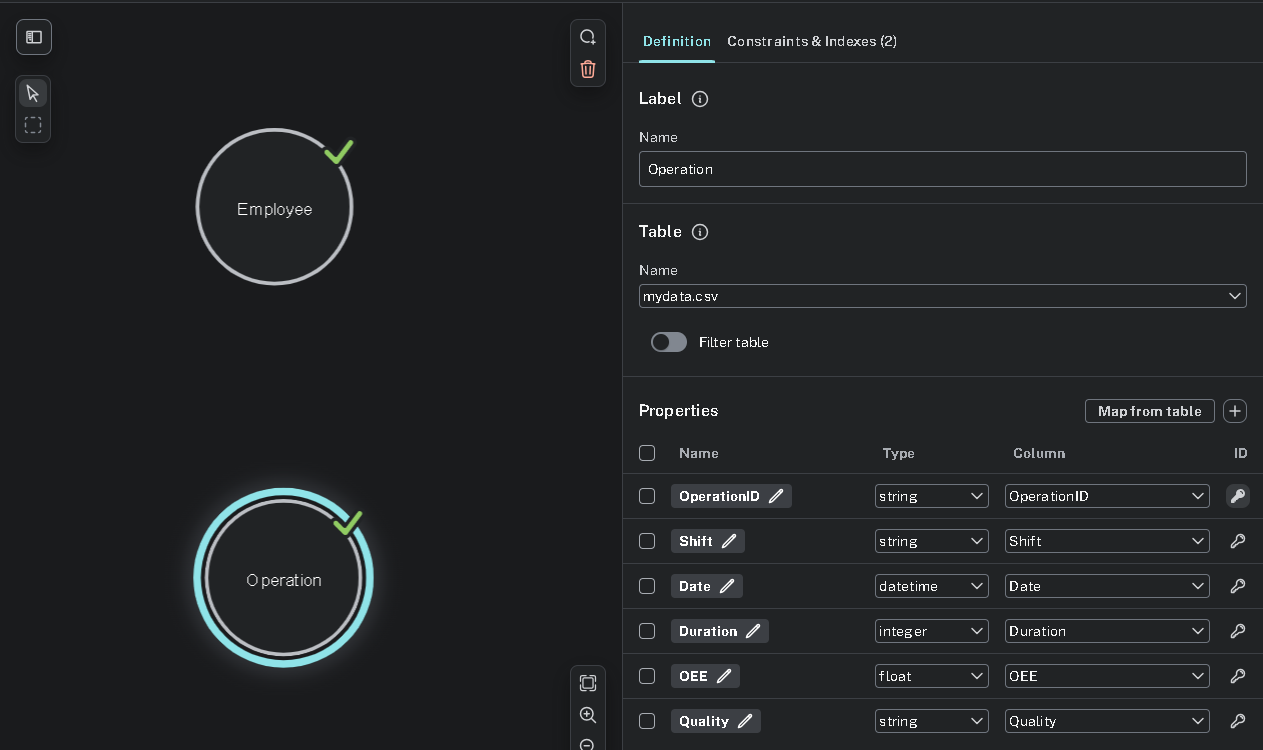
**Step-10:** Relevant column name is chosen, name here.



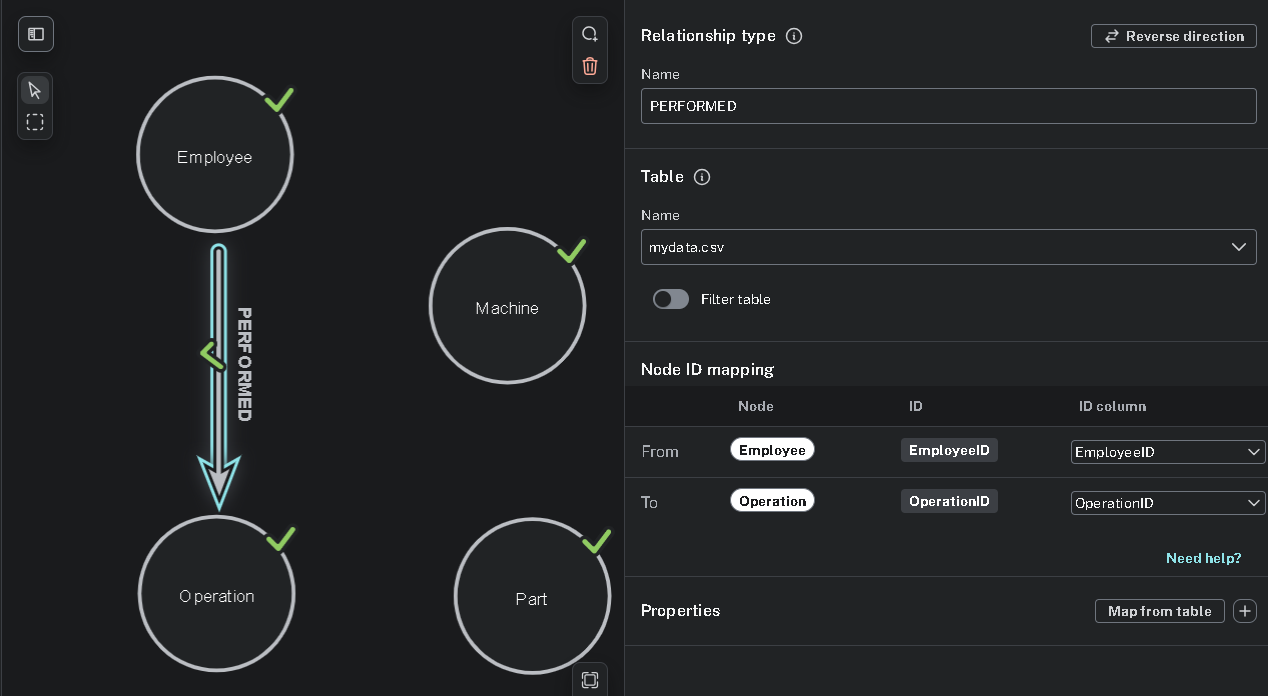
**Step-11:** After all, Employee node is created.



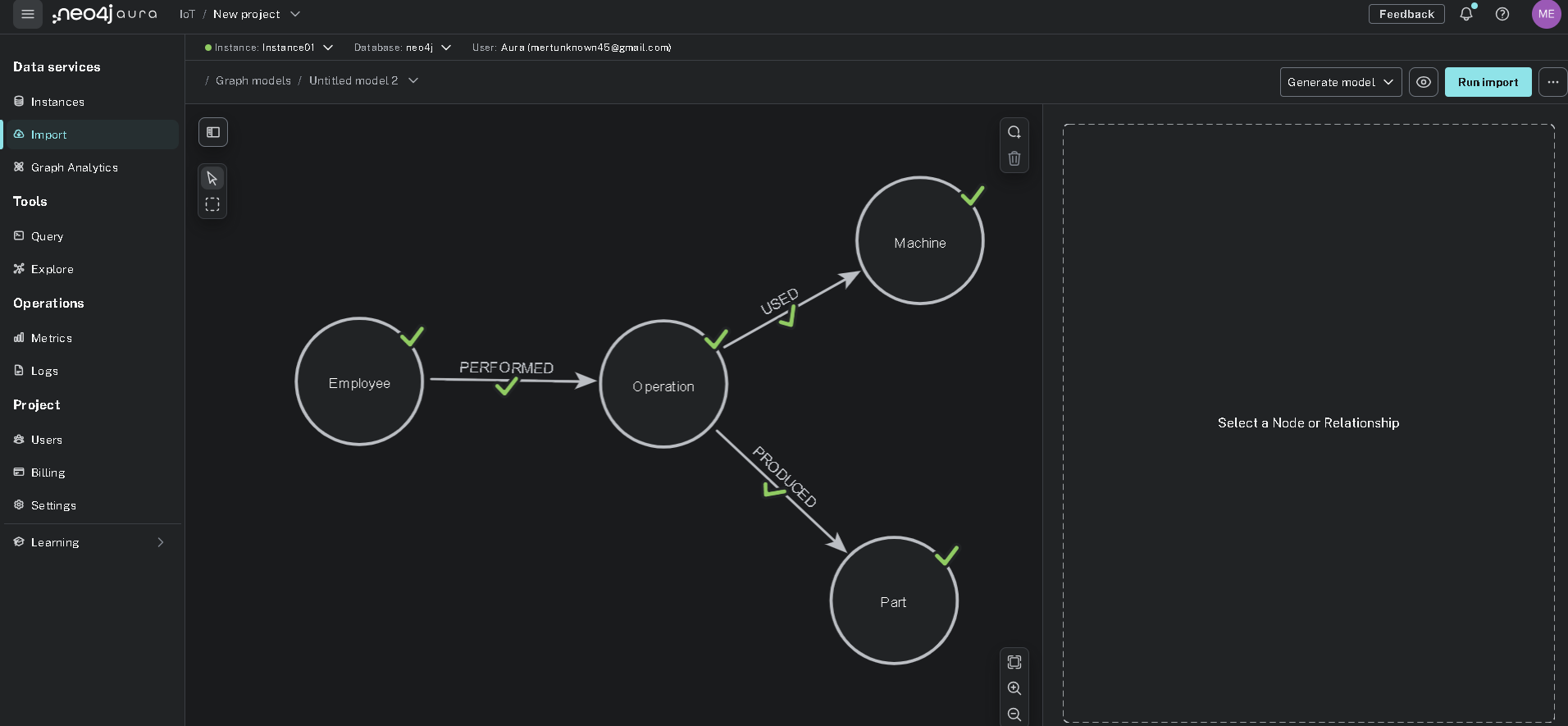
**Step-8:** Similar steps are applied for Operation node.



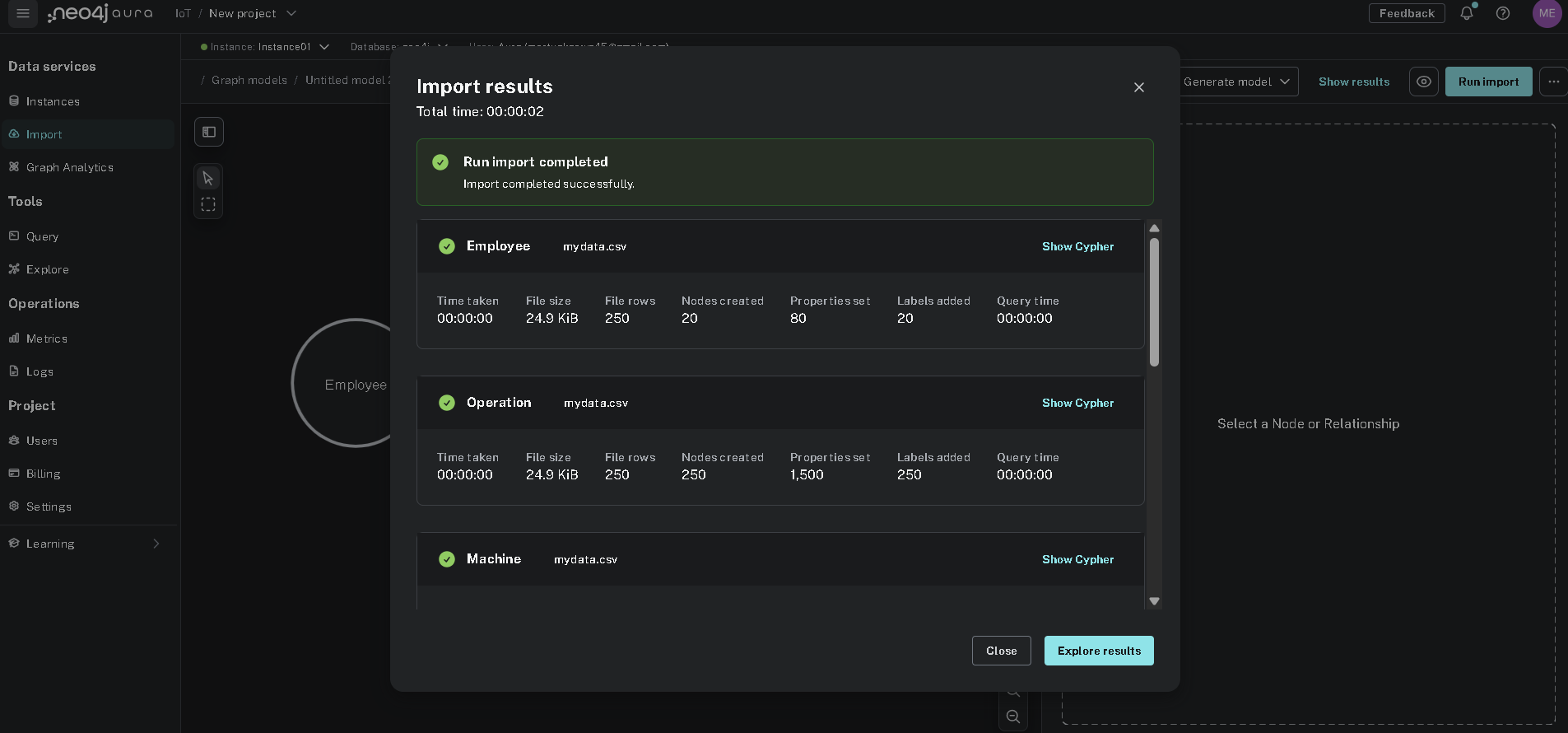
**Step-9:** After creating all of the nodes, the relationships are created by clicking and dragging from one node edge to another. Relationship name is specified as name. If there is any property which should be attached to relationship, it can be added through the plus button near map from table. However, there is no property to add for this dataset.



**Step-12:** All of them are created as below. At the end, Run Import button is clicked to perform all of the settings.

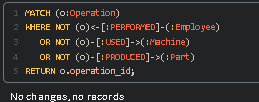


**Step-13:** The result is seen in the windown below. Here, how many nodes, properties, and labels were created can be seen for each node type. Also, the total time is visible.



**TEST**

After loading a dataset, it should be checked if it has any errors. According to the query, there is no record found, which is related to not linked nodes.



**SUMMARY**

In this task, a CSV dataset was imported into Neo4j Aura through both the Cypher LOAD CSV method and the manual interface provided in the platform. Throughout the process, data integrity was maintained by defining appropriate constraints, assigning correct data types, and establishing meaningful relationships between nodes.

The work demonstrated how structured operational data can be effectively modeled in a graph format to allow more flexible and insightful querying. Emphasis was placed on the importance of schema design, accurate property mapping, and validation during the import process.

As a result, a well-structured graph model was created, enabling rich analysis of operations, employee activities, machine utilization, and part production, which provides a solid foundation for further analytical tasks in future assignments.