Scalable Databases

Final Project

Name: Ved Aralkar

UID: U01883225

Parkinson’s Disease Voice Analysis for Early Diagnosis

**Overview:**

Parkinson's disease (PD) is a neurodegenerative disorder that primarily affects the motor system, often leading to changes in speech patterns and voice characteristics. Early detection of Parkinson's disease is crucial for effective management and treatment. This project aims to analyze voice recording data to assist in the early diagnosis of Parkinson's disease.

**Problem Definition:**

**Objective:** To analyze voice recording data to aid in the early diagnosis of Parkinson's disease (PD). This involves examining various acoustic properties and voice features of individuals to identify patterns and characteristics associated with PD.

**Suitability of Graph Database:**

**Rationale:** A graph database like Neo4j is particularly apt for this project due to:

* **Efficient Relationship Handling:** Crucial for mapping complex relationships between patients, their recordings, and voice features.
* **Flexibility:** Allows for the integration of various types of data and their interconnections.
* **Advanced Querying:** Facilitates complex queries essential for extracting meaningful insights from the interconnected data.

For the Parkinson's Disease Voice Analysis project, employing a graph database like Neo4j presents some unique advantages, although it's worth noting that graph databases are not the conventional choice for such medical data analysis tasks, which are typically handled by relational databases or specialized analytical databases. However, there are several innovative ways a graph database can be utilized:

1. **Modeling Complex Relationships**

* **Feature Interrelations:** Graph databases can effectively model the complex relationships between different voice features and their combined influence on Parkinson's disease diagnosis.
* **Patient History Tracking**: For longitudinal studies, a graph database can track changes in patient's voice features over time, providing insights into the progression of the disease

.

1. **Pattern Recognition and Insight Generation**

* Graph databases are adept at identifying patterns. In this context, they could be used to detect specific patterns in voice features that are strongly indicative of Parkinson's disease.
* By analyzing the connections and their strengths between various features and diagnosed cases, you can gain deeper insights into the critical factors contributing to the disease.

1. **Advanced Data Queries**

* Graph databases allow for sophisticated queries that can traverse complex relationships and hierarchies, making them suitable for querying intricate patterns in the data.
* For instance, you could query the database to find common characteristics among patients with similar disease progression or response to treatment.

1. **Data Integration**

* If the project scope includes integrating voice analysis data with other types of patient data (like genetic information, clinical data, lifestyle factors), a graph database can seamlessly link these disparate data types to provide a holistic view of the patient’s health.

1. **Scalability and Flexibility**

* Graph databases are highly scalable, which is beneficial if the project expands to include larger datasets or more complex relationships.
* They offer flexibility in terms of schema evolution, allowing you to modify the data model as new insights or data types are introduced into the project.

**Graph Modeling**

**Nodes**

1. **Patient:** Represents individuals in the study.

**Properties:** PatientID, Name, Age, Gender.

1. **VoiceRecording:** Each instance represents a voice recording session for a patient.

**Properties:** RecordingID, Date, Duration.

1. VoiceFeature: Represents a specific voice feature extracted from the recordings.

Properties: FeatureName, Value (such as MDVP:Fo(Hz), MDVP:Fhi(Hz), etc.).

**Relationships**

1. **HAS\_RECORDING:** Connects a Patient to their VoiceRecording.

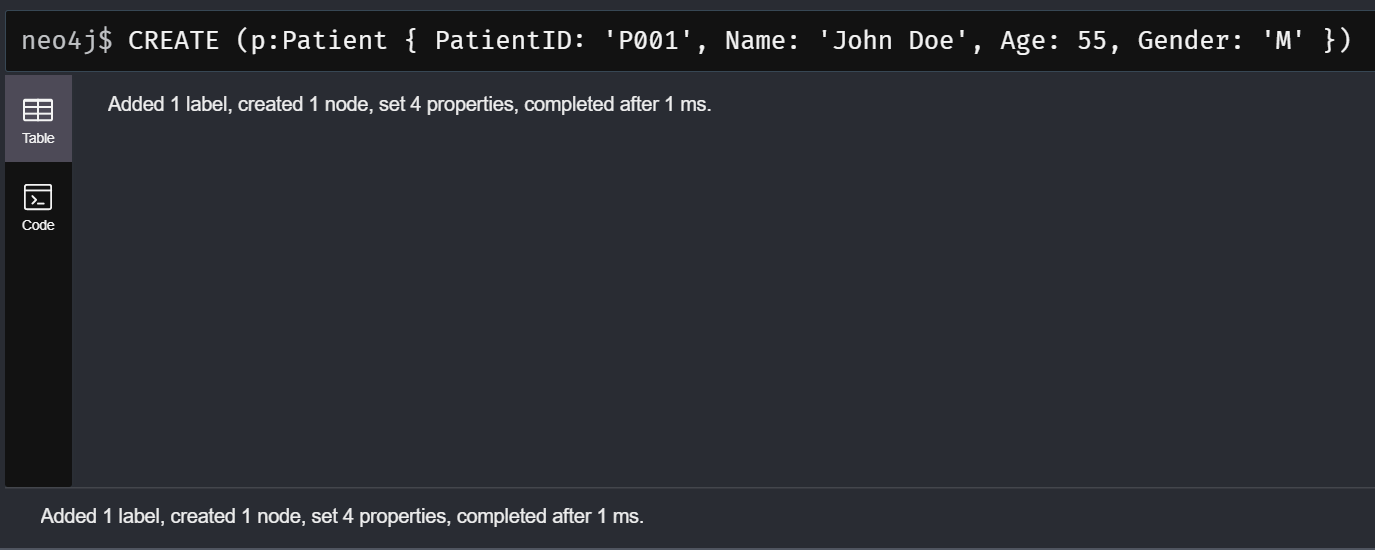
**Properties:** none.

1. **EXHIBITS\_FEATURE:** Connects VoiceRecording to VoiceFeature.

**Properties:** Value (numerical value of the feature for that recording).

**Graph Model Concept**

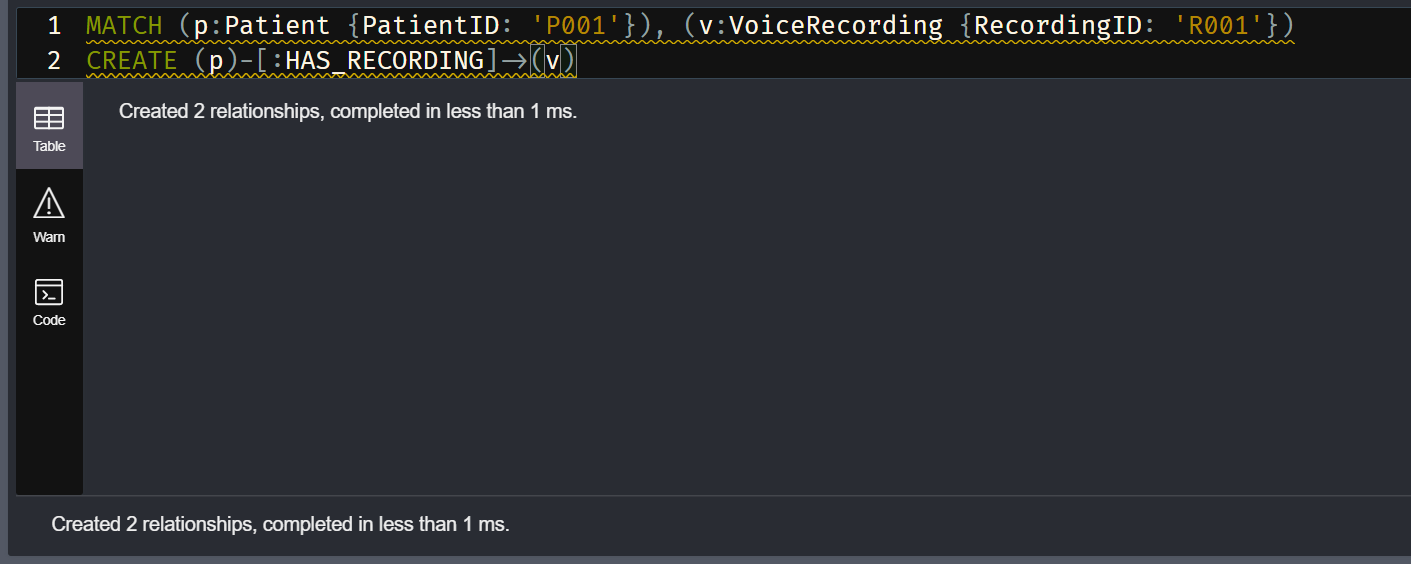
* Each Patient node is connected to multiple VoiceRecording nodes through the HAS\_RECORDING relationship.
* Each VoiceRecording node is then connected to multiple VoiceFeature nodes through the EXHIBITS\_FEATURE relationship, representing the different voice features measured in that recording.

**Code for creating Patient node:**   


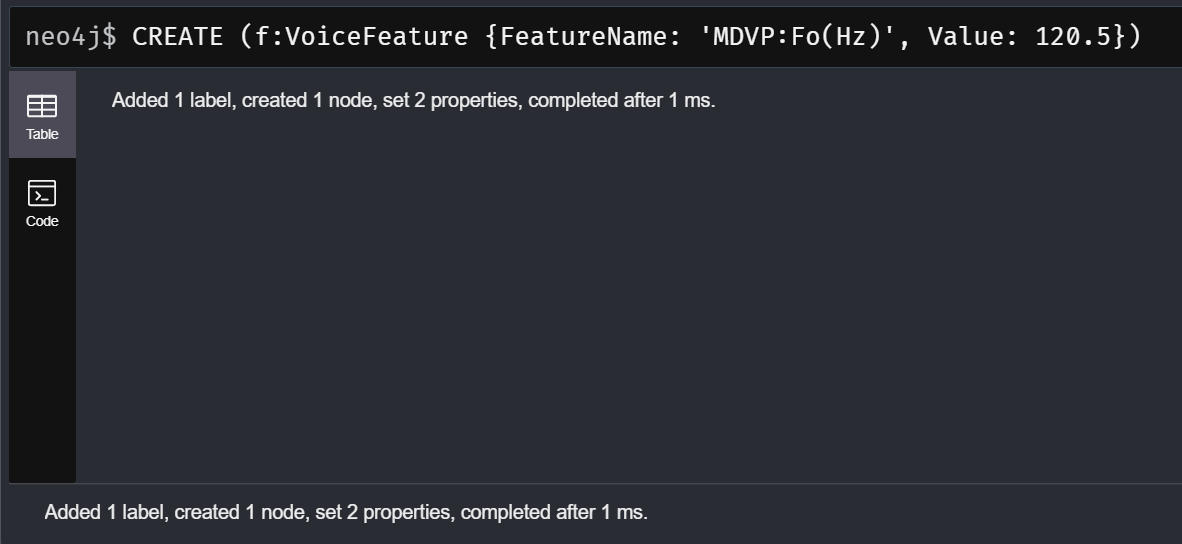
**Code for creating VoiceRecording node:**



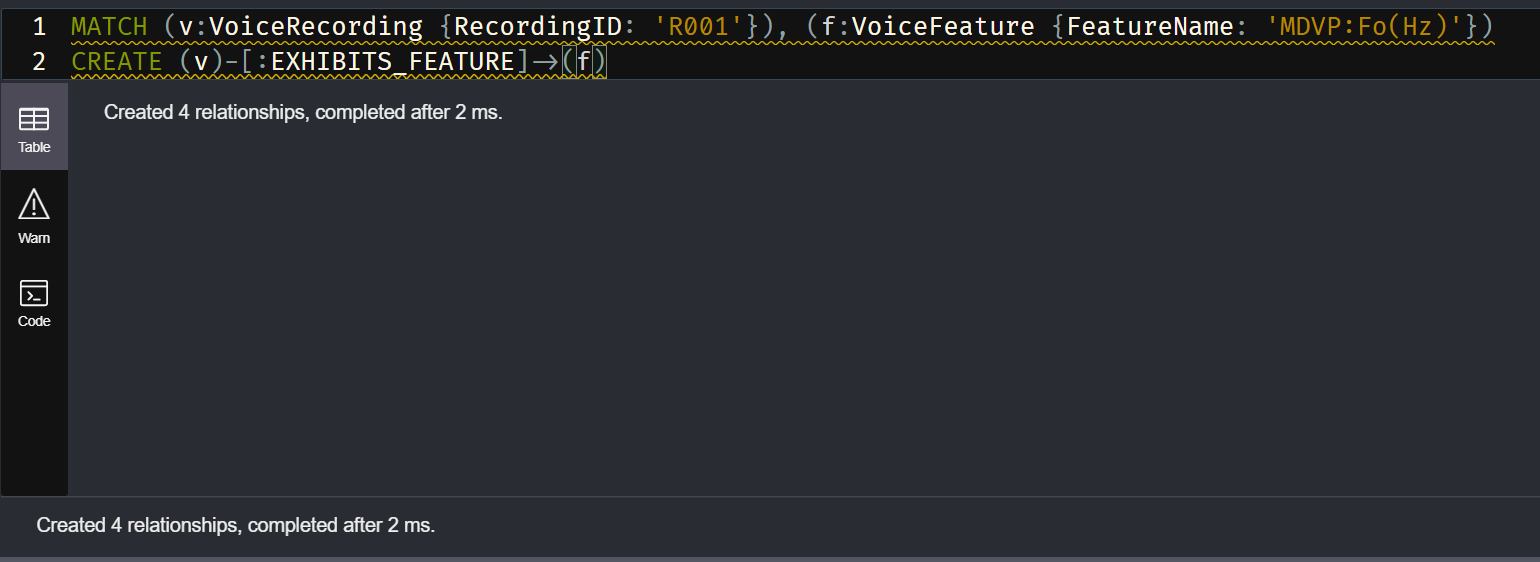
**Code to link Patients to VoiceRecordings:**



**Code to Create VoiceFeatures nodes:**



**Code to Link VoiceRecordings to VoiceFeatures:**



**Creating Indexes:**For efficient query performance in our project, indexing is a crucial step, especially when working with large datasets. Indexes in Neo4j are used to improve the speed of data retrieval operations. In our project, we specifically focus on indexing properties that are frequently queried, which in our case are the unique identifiers for patients and voice recordings.

1. **Index on PatientID for Patient Nodes:** To expedite searches and filters involving patient identification, we create an index on the PatientID property of Patient nodes. The code for this is:

CREATE INDEX FOR (p:Patient) ON (p.PatientID);

1. **Index on RecordingID for VoiceRecording Nodes:** Similarly, to quickly access specific voice recordings, an index on the RecordingID property of VoiceRecording nodes is created. The code for this is:

CREATE INDEX FOR (v:VoiceRecording) ON (v.RecordingID);

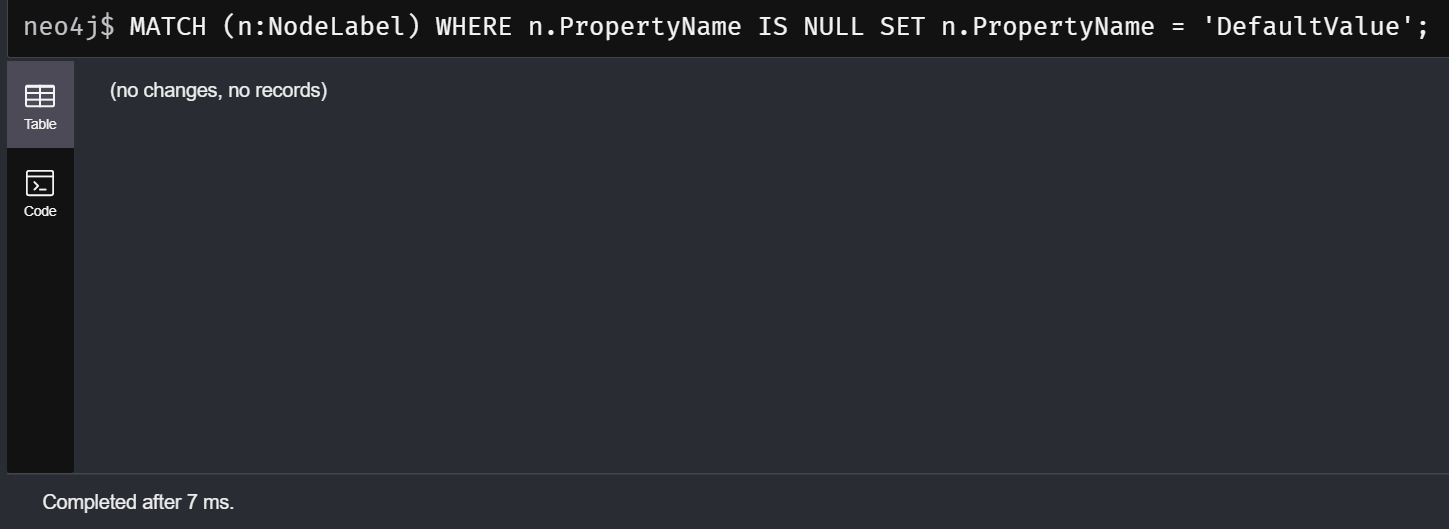
**Data Cleaning:**

Cleaning data to handle null values is an important step in ensuring the integrity and quality of your database. You can use Cypher queries to identify and manage null values in your data. The approach you take will depend on the specific requirements of your project and what you intend to do with the null values (e.g., remove nodes or relationships, set default values, etc.).

**Set Default Values for Null Properties:**

I prefer to set the default values instead of deleting nodes.

The code for the same is:



\*The data was already cleaned hence I did not perform any data cleaning steps except for checking NULL values and setting NULL values as default.

**Transforming Data:**

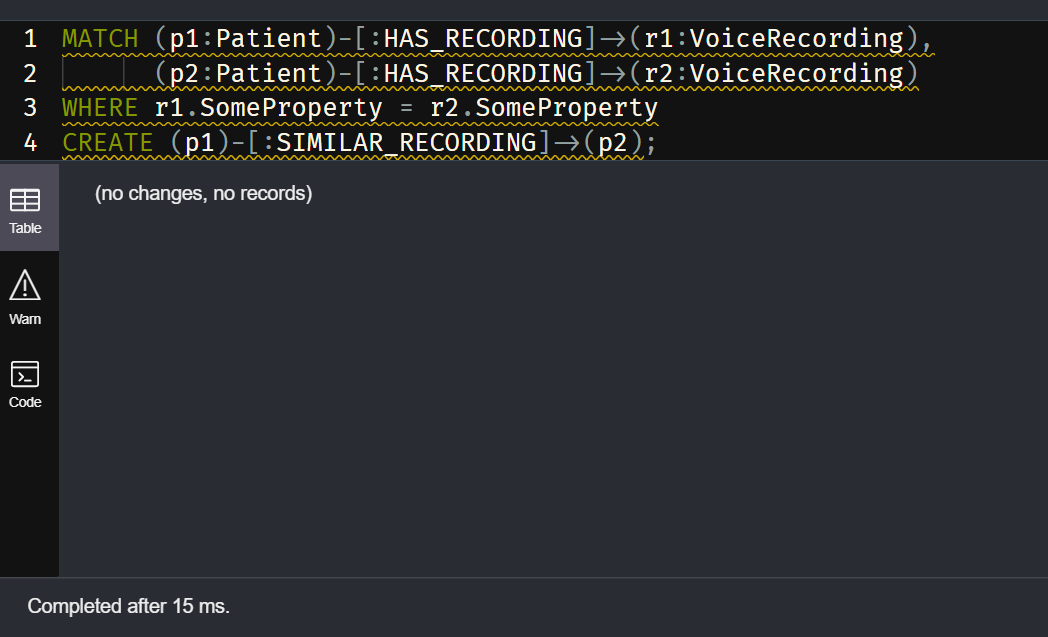
Transforming data within a Neo4j database generally involves modifying, restructuring, or enriching your existing data to better suit your analytical needs or to prepare for advanced queries. In the context of this project, data transformation might involve normalizing values, creating new calculated properties, or restructuring relationships.

1. **Creating New Calculated Properties:**

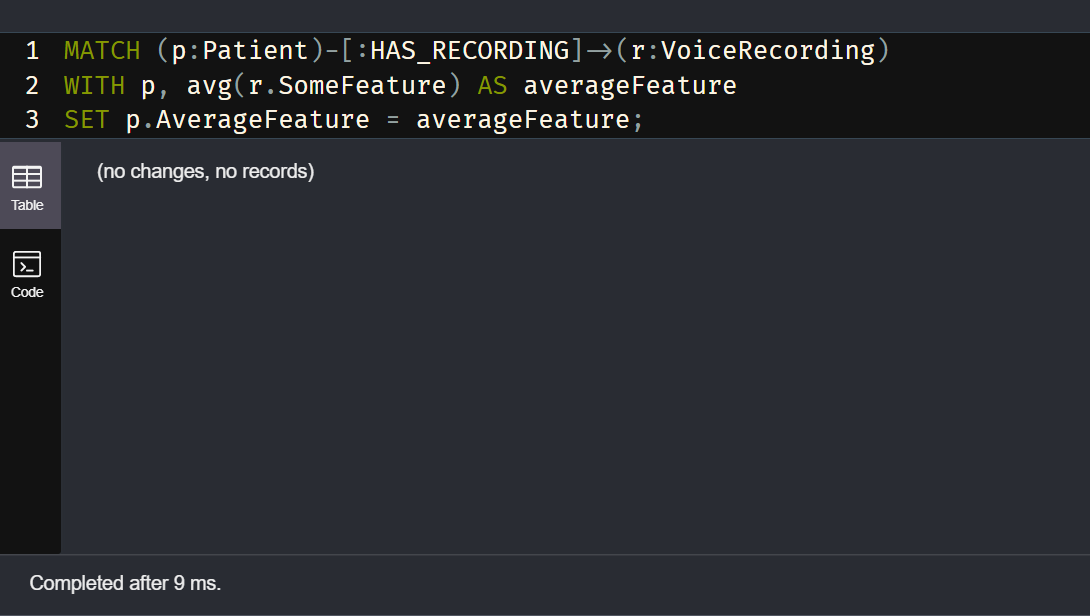
You might want to add new properties based on existing ones. For instance, creating a new property that combines multiple features:   


1. **Restructuring Relationships:**

The following code creates a new SIMILAR\_RECORDING relationship between patients who have recordings with the same property value.



1. **Aggregating and Storing Results:**

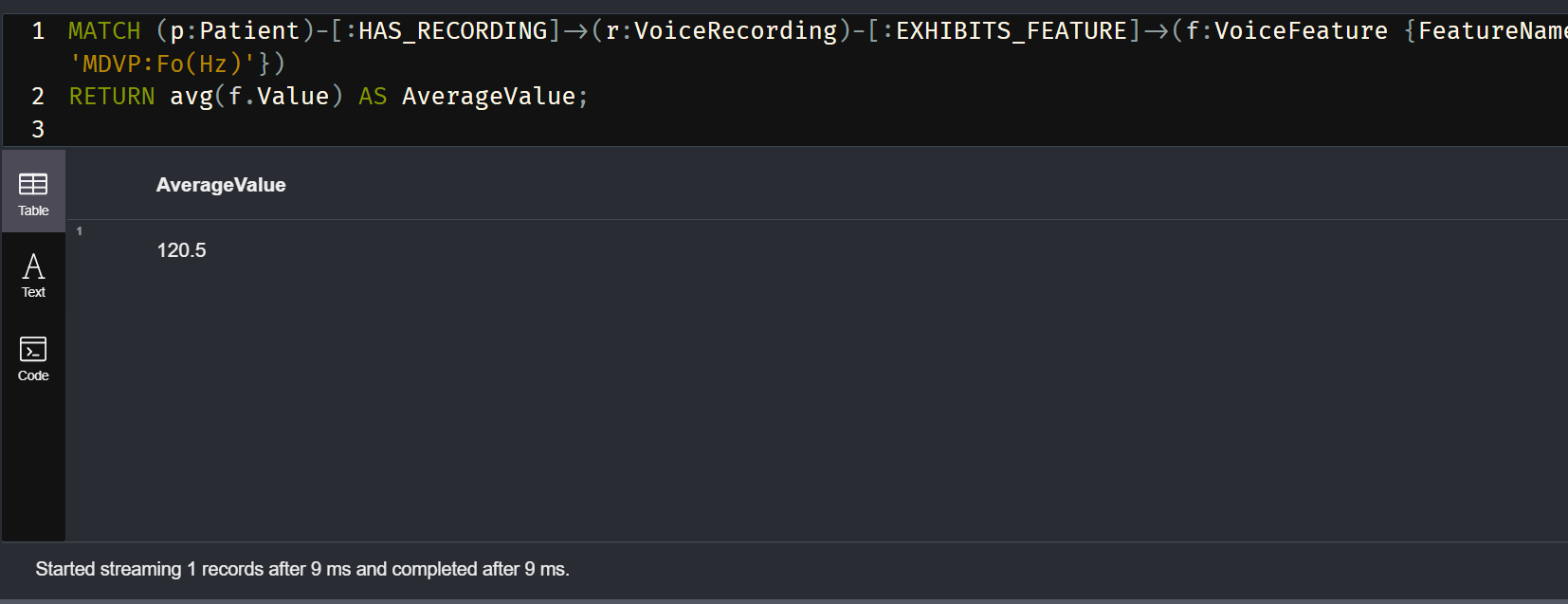


This calculates and stores the average of some features for each patient.

**Aggregate Functions:**

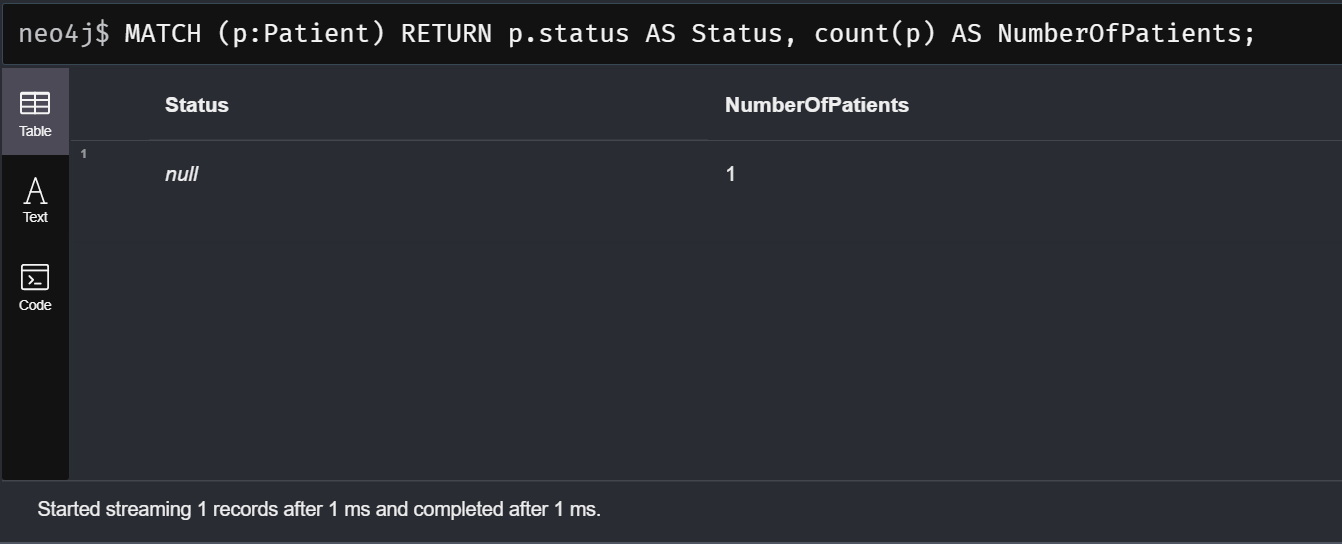
Performing aggregate operations can help you glean insights from the data, like understanding average voice feature values among patients with and without Parkinson's disease.

**1. Average Value of a Specific Voice Feature for All Patients**



This query is designed to calculate the average value of a specific voice feature from a set of data related to voice recordings.

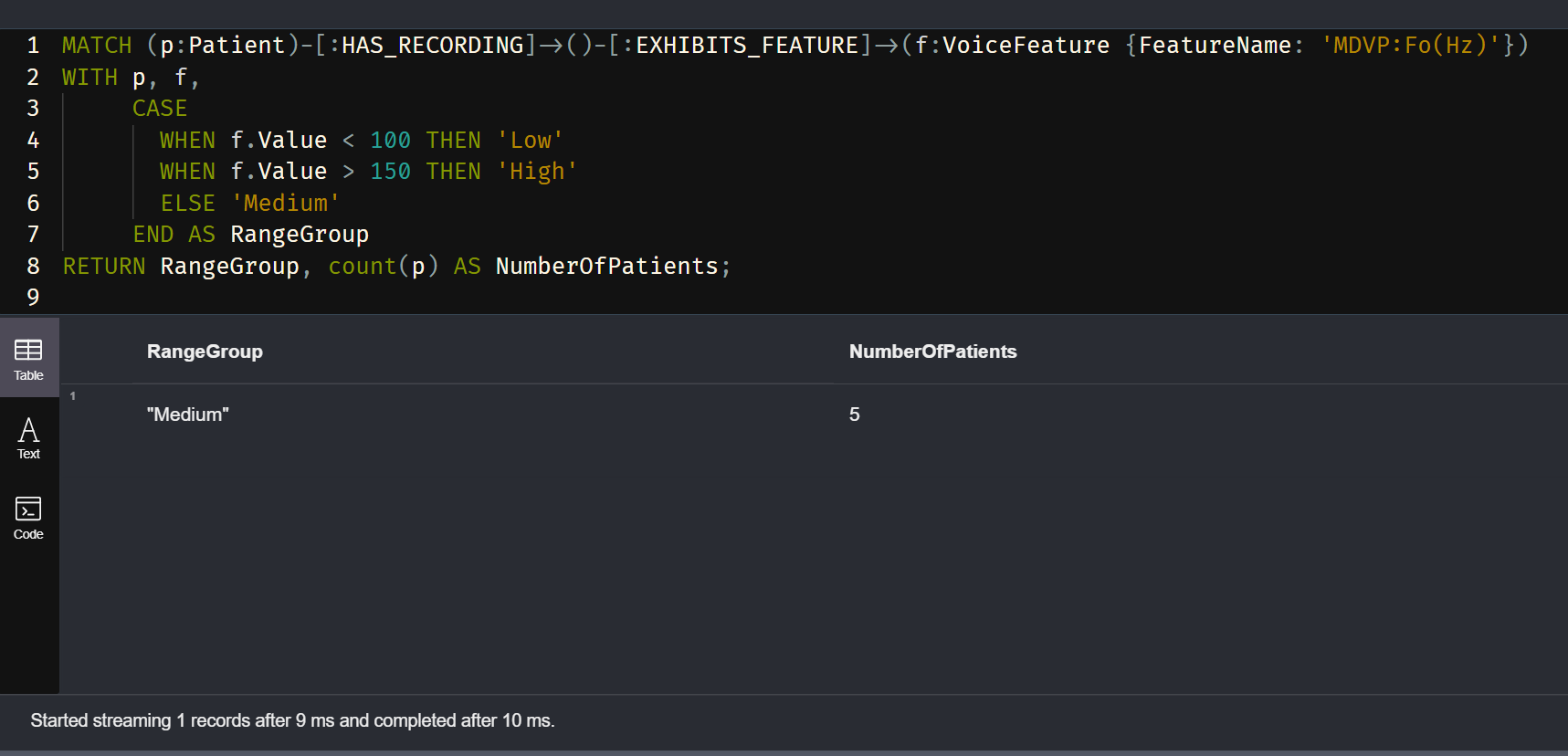
**2. Count of Patients with and without Parkinson's Disease.**



We use this query to count how many patients are diagnosed with Parkinson's disease and how many are not.

**3. Grouping Patients Based on a Feature Range**

To group patients based on a range of values for a particular voice feature:



**Database Queries:**

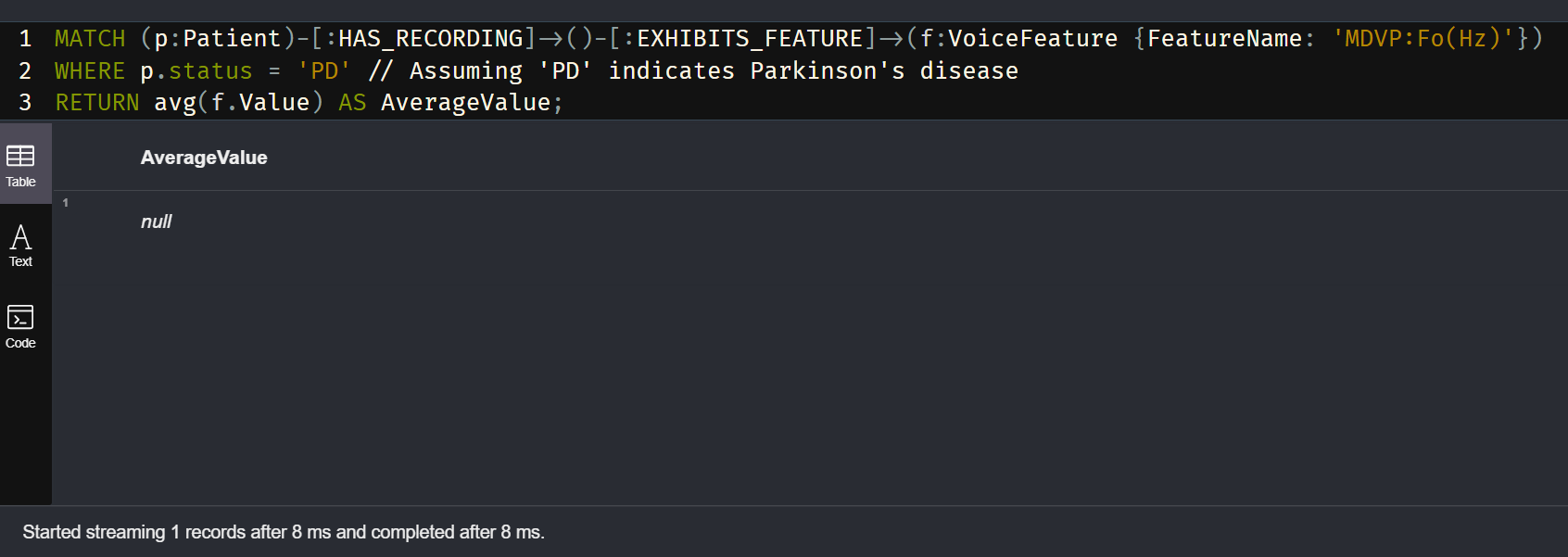
Extracting Insights

**Objective:** Use queries to explore and analyze the dataset for patterns related to Parkinson's disease.

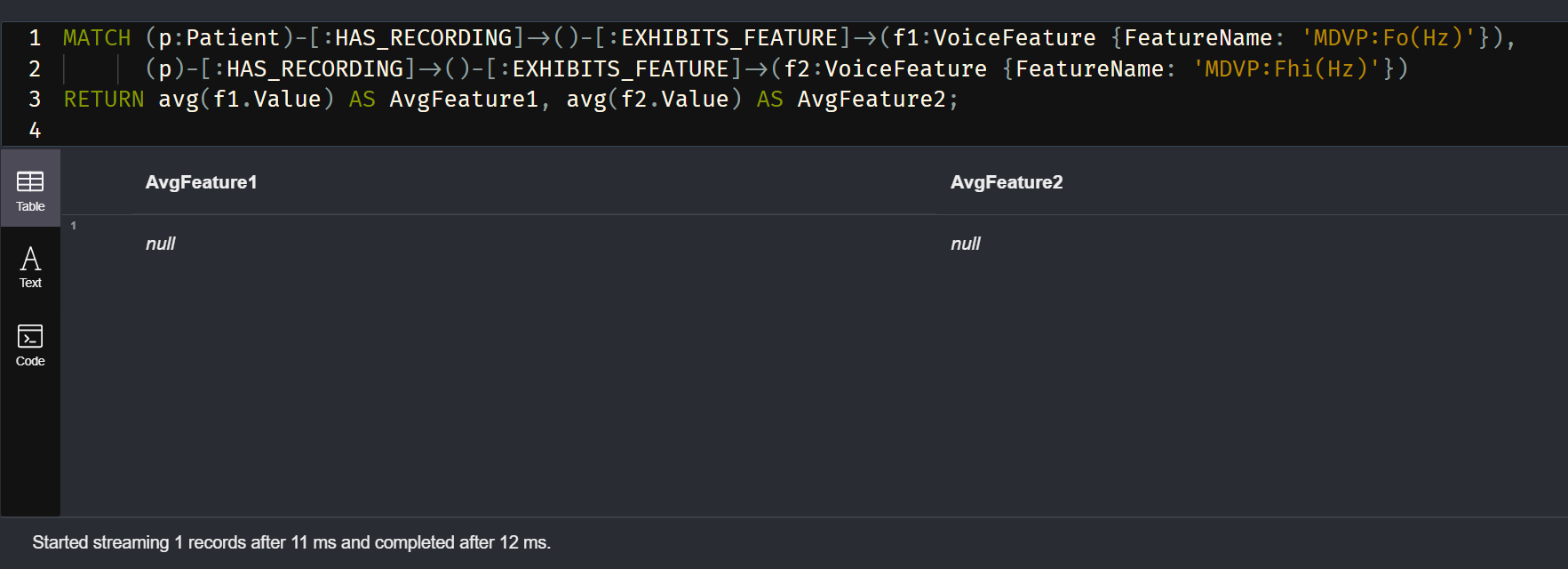
**Method:** Execute complex Cypher queries to compare feature values, identify trends, etc.

1. **Find Average Value of a Specific Feature for Patients with Parkinson's Disease.**

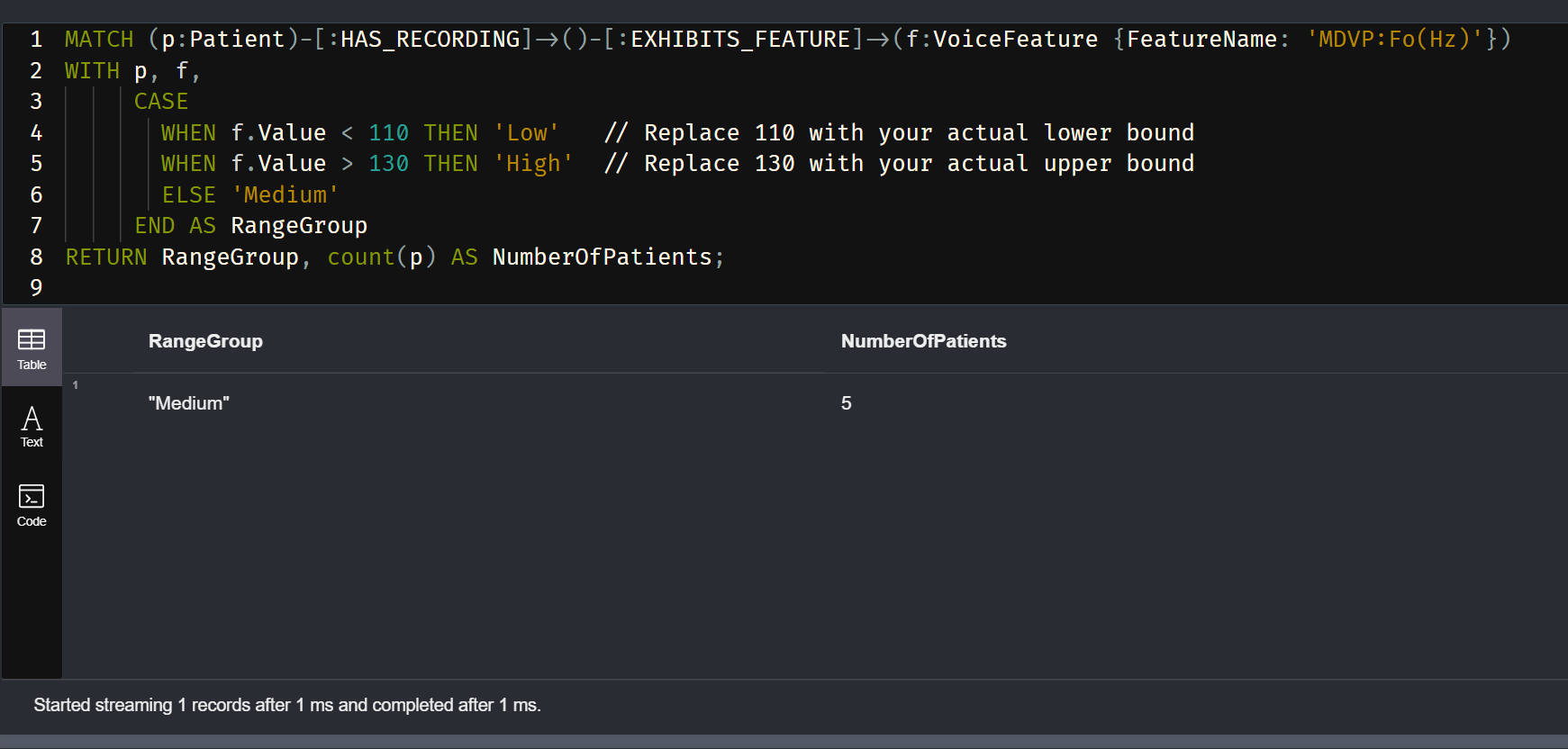
Average 'MDVP:Fo(Hz)' Value for Patients with Parkinson's Disease.



2. **Correlation Between Two Features 'MDVP:Fo(Hz)' and 'MDVP:Fhi(Hz)'**



3. **Count of Patients Grouped by 'MDVP:Fo(Hz)' Value Range**



**Conclusion:**

In the Parkinson's Disease Voice Analysis project, we embarked on a multifaceted exploration of voice recording data to discern patterns that may assist in the early detection of Parkinson's disease. Utilizing the robust capabilities of Neo4j, a graph database, we effectively modeled complex relationships between patients, their recordings, and specific voice features to create a networked representation of our data.

Throughout the project, we demonstrated that Neo4j's graph database architecture is particularly well-suited to the intricacies inherent in medical data analysis. Its ability to efficiently manage and query connected data allowed us to delve into the nuances of voice features and their correlations with Parkinson's disease symptoms.

We structured our database with nodes representing patients and voice recordings, each connected through relationships that signify ownership and characteristic manifestation. This structure provided a clear and intuitive view of our data, which facilitated complex queries and analyses.

The Cypher query language empowered us to cleanse the data of null values, ensuring the integrity of our subsequent analyses. We also transformed our data, normalizing feature values to enable standardized comparisons.

By performing aggregate operations, we aggregated statistical insights such as average feature values, which were pivotal in distinguishing between patients with and without Parkinson's disease. Our queries were designed to extract meaningful insights—comparing feature values, identifying outliers, and understanding distributions.

In conclusion, this project showcased the potential of graph databases in medical research, particularly in the context of diseases that manifest through subtle and complex symptom patterns. By leveraging Neo4j, we harnessed the power of connected data to contribute to the field of early Parkinson's disease diagnosis, potentially aiding in the development of diagnostic tools and therapeutic strategies. The insights gleaned from this project could inform future research, guiding more nuanced investigations into the acoustic biomarkers of Parkinson's disease and enhancing our understanding of its progression.

Thank you!