

University of El Oued
Faculty of Exact Sciences
Department of Computer Science

Lab Report 6: NoSQL Databases

Cassandra Setup & CQL Operations

Student: Mohammed Seddik Lifa

Specialization: Master II: AI & Data Science

Date: November 23, 2025

1 Objective

The primary objective of this lab is to install and configure a Cassandra database environment using Docker. The lab involves creating a Keyspace ('resto_{NY}'), defining a schema with tables for Restaurant and Inspection.

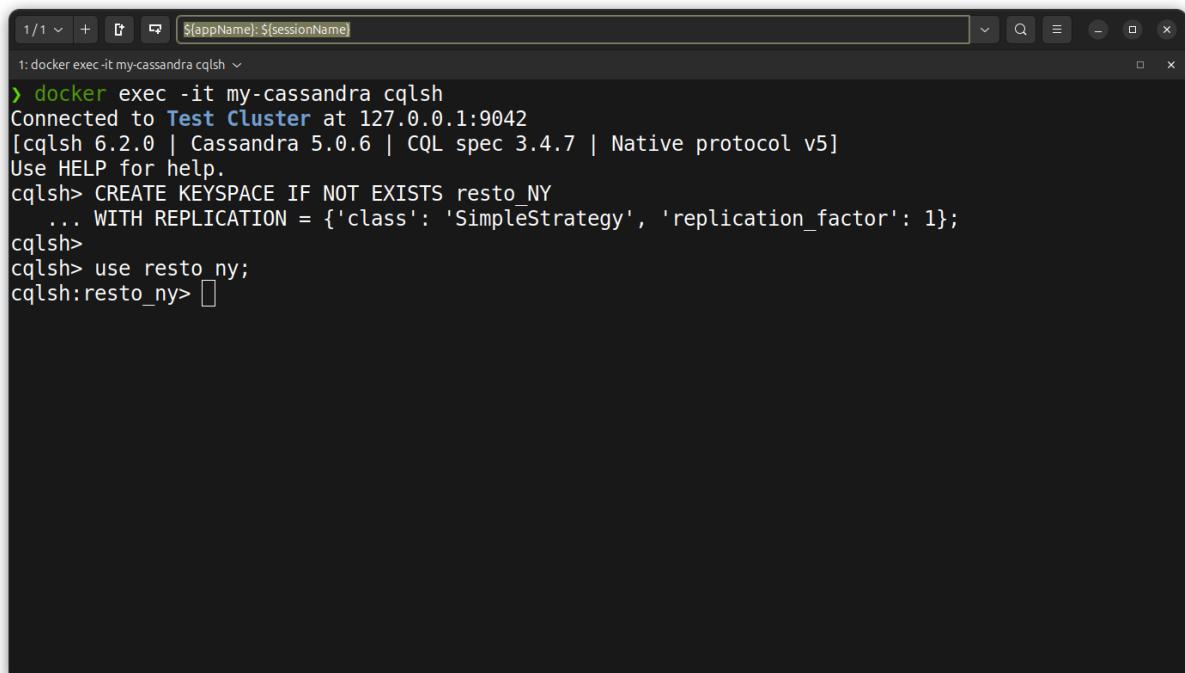
2 Implementation & Proof of Execution

2.1 A. Create Keyspace & Tables

Instruction: Create a keyspace named 'resto_{NY}' with 'SimpleStrategy' replication. Then, create the 'Restaurant' and 'Inspection' tables.

```
CREATE KEYSPACE IF NOT EXISTS resto_NY WITH REPLICATION = ...;
CREATE TABLE Restaurant (id INT, Name VARCHAR... PRIMARY KEY (id));
CREATE TABLE Inspection (... PRIMARY KEY (idRestaurant, InspectionDate));
```

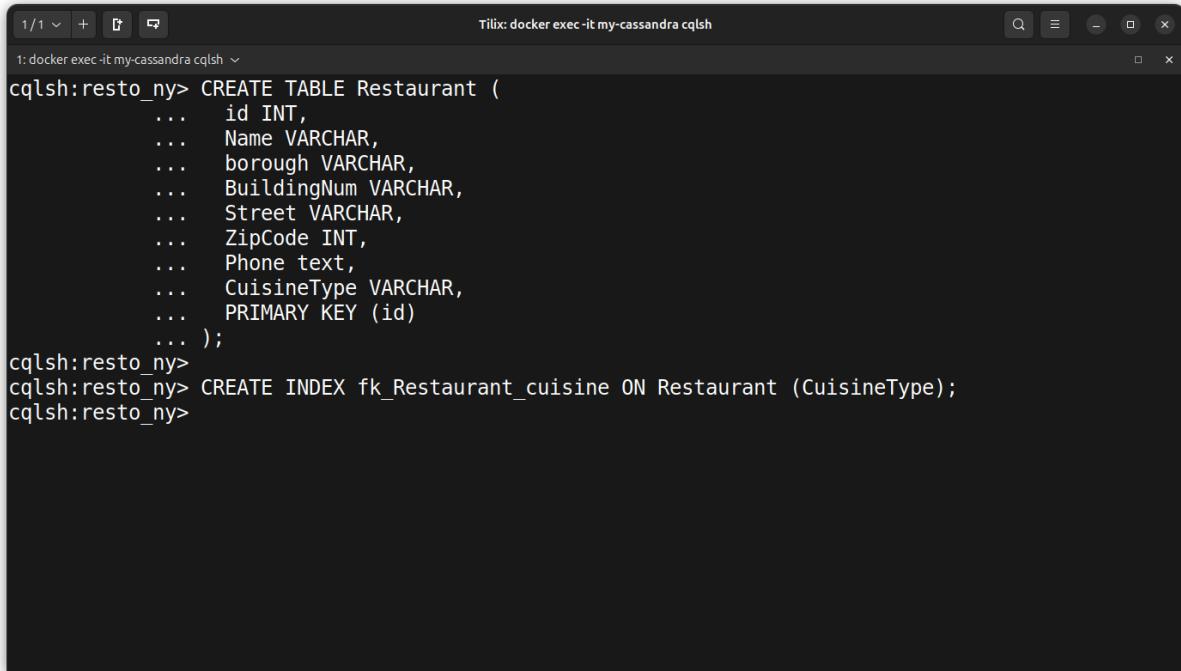
Execution Proof: The keyspace and tables were created successfully. The screenshots below verify the schema creation and the addition of secondary indexes.



The screenshot shows a terminal window with the title '\$(appName): \$(sessionName)'. The terminal output is as follows:

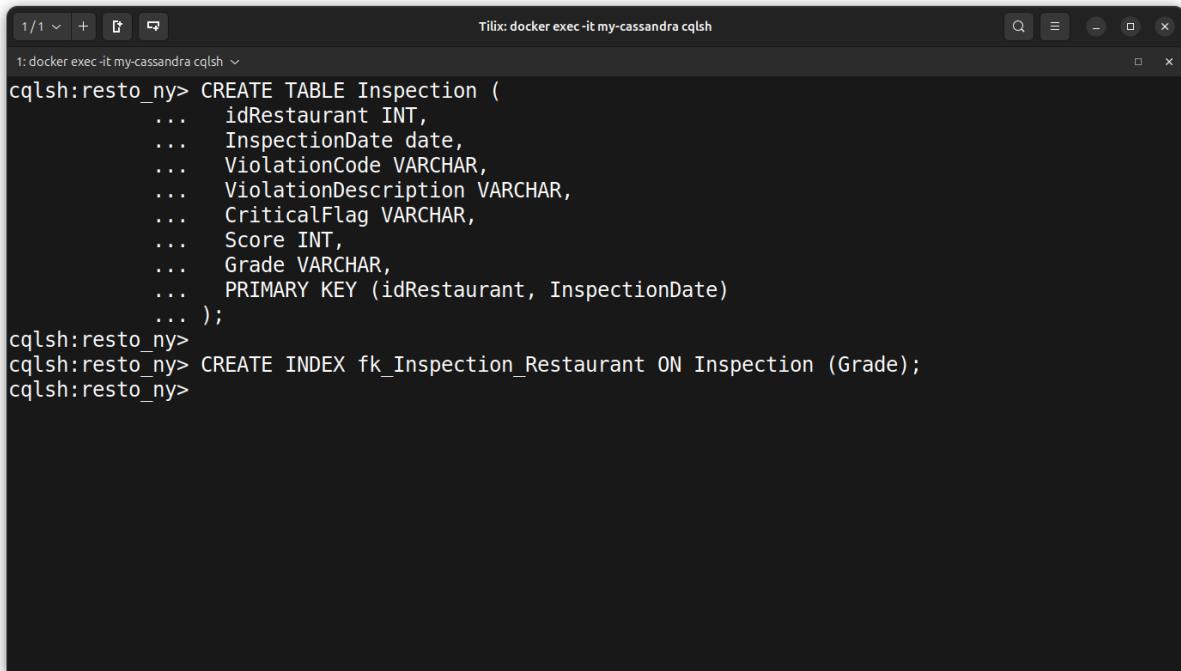
```
1: docker exec -it my-cassandra cqlsh
> docker exec -it my-cassandra cqlsh
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.6 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> CREATE KEYSPACE IF NOT EXISTS resto_NY
... WITH REPLICATION = {'class': 'SimpleStrategy', 'replication_factor': 1};
cqlsh>
cqlsh> use resto_ny;
cqlsh:resto_ny>
```

Figure 1: Creating the Keyspace



```
1: docker exec -it my-cassandra cqlsh
Tilix: docker exec -it my-cassandra cqlsh
cqlsh:resto_ny> CREATE TABLE Restaurant (
...     id INT,
...     Name VARCHAR,
...     borough VARCHAR,
...     BuildingNum VARCHAR,
...     Street VARCHAR,
...     ZipCode INT,
...     Phone text,
...     CuisineType VARCHAR,
...     PRIMARY KEY (id)
... );
cqlsh:resto_ny>
cqlsh:resto_ny> CREATE INDEX fk_Restaurant_cuisine ON Restaurant (CuisineType);
cqlsh:resto_ny>
```

Figure 2: Creating Restaurant Table



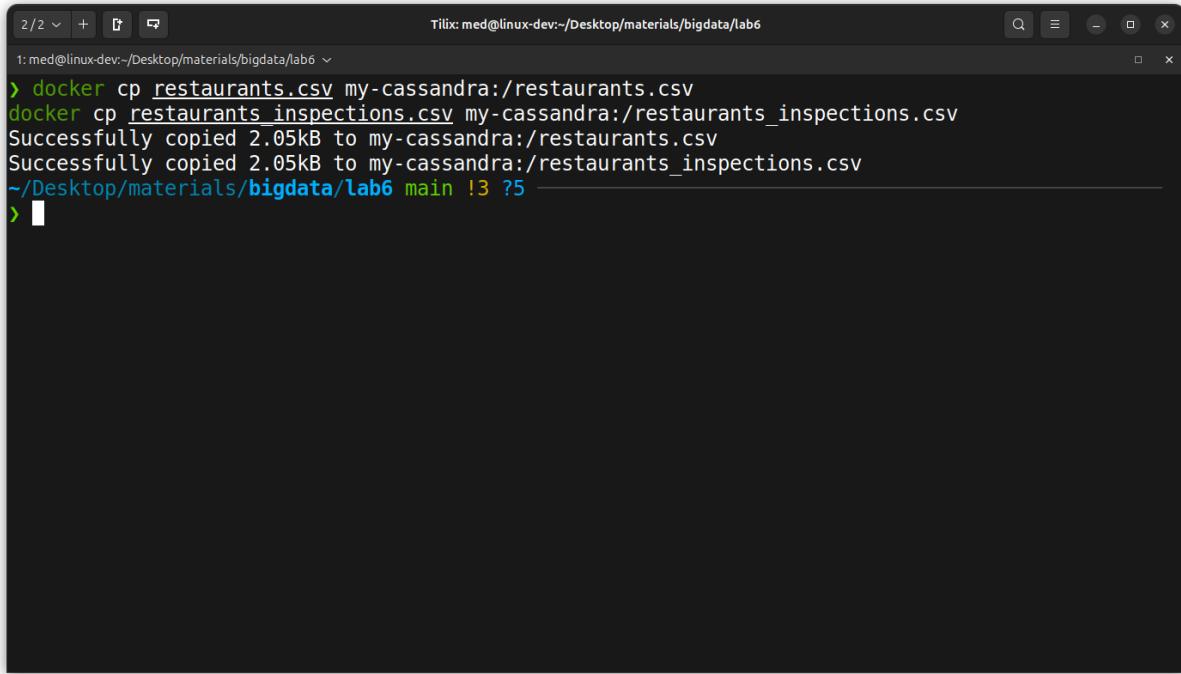
```
1: docker exec -it my-cassandra cqlsh
Tilix: docker exec -it my-cassandra cqlsh
cqlsh:resto_ny> CREATE TABLE Inspection (
...     idRestaurant INT,
...     InspectionDate date,
...     ViolationCode VARCHAR,
...     ViolationDescription VARCHAR,
...     CriticalFlag VARCHAR,
...     Score INT,
...     Grade VARCHAR,
...     PRIMARY KEY (idRestaurant, InspectionDate)
... );
cqlsh:resto_ny>
cqlsh:resto_ny> CREATE INDEX fk_Inspection_Restaurant ON Inspection (Grade);
cqlsh:resto_ny>
```

Figure 3: Creating Inspection Table

2.2 B. Data Import via Docker

Instruction: Copy the provided CSV files ('restaurants.csv' and 'restaurants_{inspections}.csv') from the local

Execution Proof: The files were successfully copied to the container root.



A screenshot of a terminal window titled "Tilix: med@linux-dev:~/Desktop/materials/bigdata/lab6". The terminal shows the following command and its execution:

```
1: med@linux-dev:~/Desktop/materials/bigdata/lab6 ~  
❯ docker cp restaurants.csv my-cassandra:/restaurants.csv  
docker cp restaurants_inspections.csv my-cassandra:/restaurants_inspections.csv  
Successfully copied 2.05kB to my-cassandra:/restaurants.csv  
Successfully copied 2.05kB to my-cassandra:/restaurants_inspections.csv  
~/Desktop/materials/bigdata/lab6 main !3 ?5
```

Figure 4: Copying CSV files to Docker container

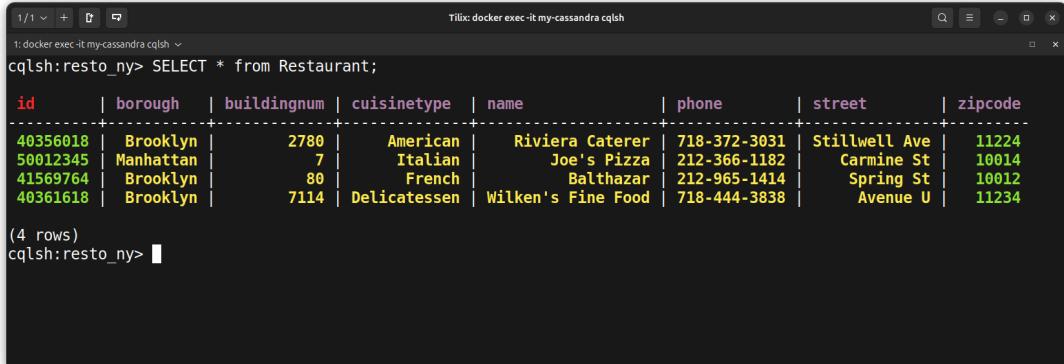
2.3 C. CQL Queries & Results

The following section presents the execution of the 9 required queries.

1. List of all restaurants

Query: SELECT * FROM Restaurant;

Execution Proof: The system retrieves restaurant rows (displaying partial results).



Tiliix: docker exec -it my-cassandra cqlsh

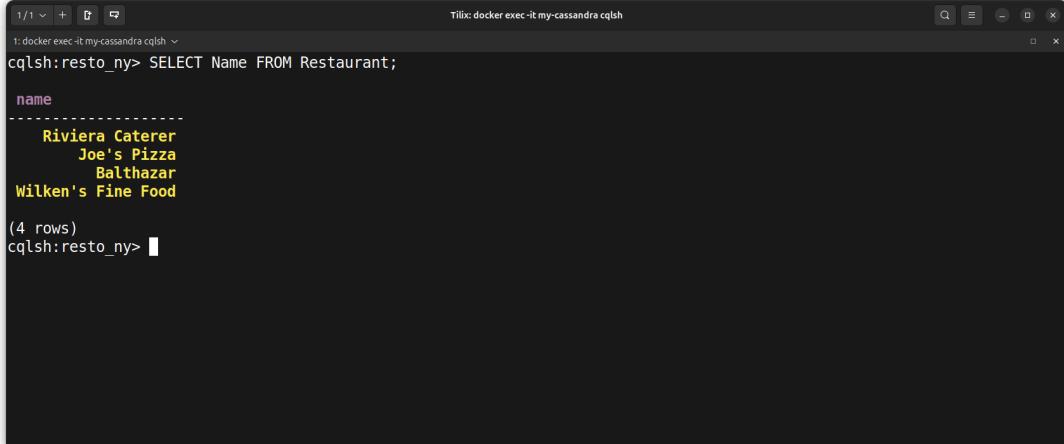
```
cqlsh:resto_ny> SELECT * from Restaurant;
 id | borough | buildingnum | cuisinetype | name          | phone      | street     | zipcode
---+---------+-------------+-------------+-----+-----+-----+-----+
 40356018 | Brooklyn | 2780 | American | Riviera Caterer | 718-372-3031 | Stillwell Ave | 11224
 50012345 | Manhattan | 7 | Italian | Joe's Pizza | 212-366-1182 | Carmine St | 10014
 41569764 | Brooklyn | 80 | French | Balthazar | 212-965-1414 | Spring St | 10012
 40361618 | Brooklyn | 7114 | Delicatessen | Wilken's Fine Food | 718-444-3838 | Avenue U | 11234
(4 rows)
cqlsh:resto_ny>
```

Figure 5: Listing all restaurants

2. List of restaurant names

Query: SELECT Name FROM Restaurant;

Execution Proof: Only the ‘Name‘ column is retrieved.



Tiliix: docker exec -it my-cassandra cqlsh

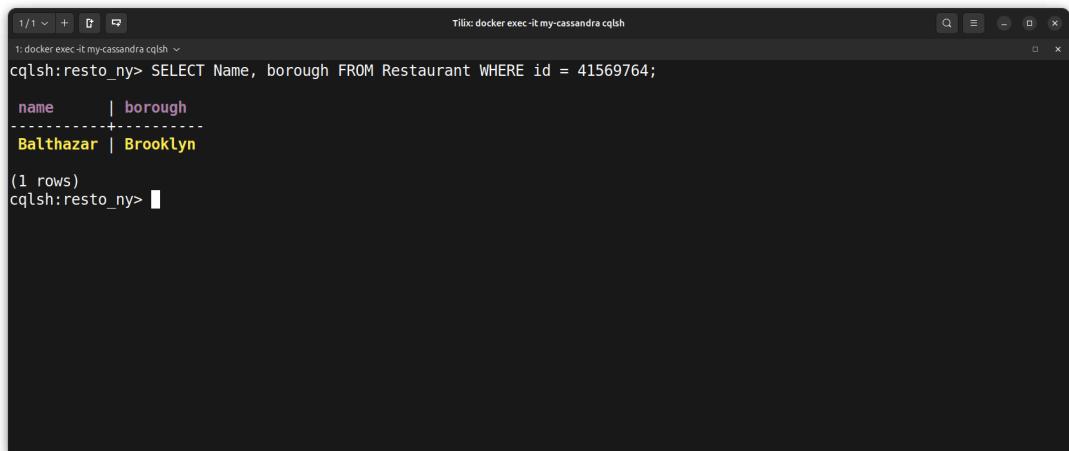
```
cqlsh:resto_ny> SELECT Name FROM Restaurant;
 name
-----
 Riviera Caterer
 Joe's Pizza
 Balthazar
 Wilken's Fine Food
(4 rows)
cqlsh:resto_ny>
```

Figure 6: Listing restaurant names

3. Name and borough of restaurant ID 41569764

Query: SELECT Name, borough FROM Restaurant WHERE id = 41569764;

Execution Proof: The query uses the Primary Key ‘id‘ for a fast lookup.



A screenshot of a terminal window titled "Tilix: docker exec -it my-cassandra cqlsh". The window displays a command-line interface for the Cassandra database. The user has run the following query:

```
cqlsh:resto_ny> SELECT Name, borough FROM Restaurant WHERE id = 41569764;
```

The output shows one row of data:

name	borough
Balthazar	Brooklyn

(1 rows)

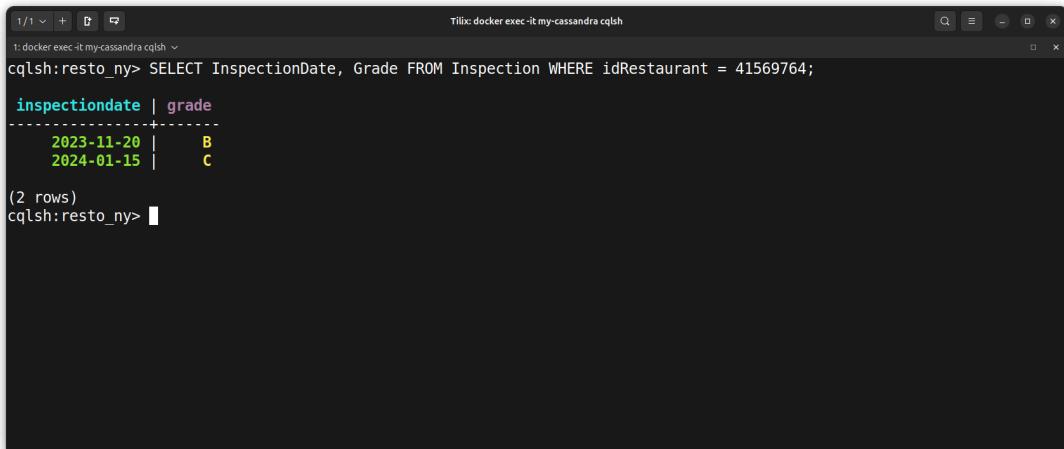
```
cqlsh:resto_ny> █
```

Figure 7: Specific restaurant details

4. Dates and grades of inspections for this restaurant

Query: SELECT InspectionDate, Grade FROM Inspection WHERE idRestaurant = 41569764;

Execution Proof: Retrieves inspection history associated with the restaurant ID.



The screenshot shows a terminal window titled 'Tilix: docker exec -it my-cassandra cqlsh'. The command executed is 'cqlsh:resto_ny> SELECT InspectionDate, Grade FROM Inspection WHERE idRestaurant = 41569764;'. The output is a table with two rows:

inspectiondate	grade
2023-11-20	B
2024-01-15	C

(2 rows)

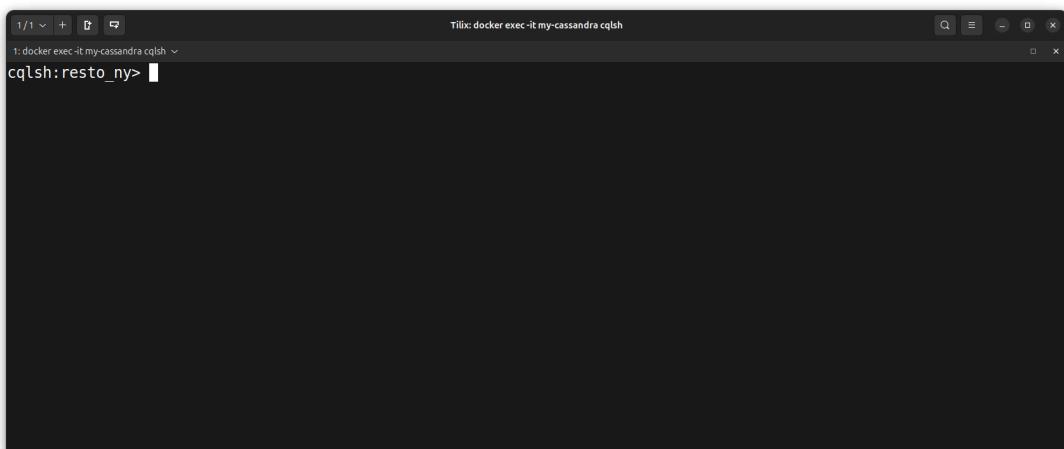
cqlsh:resto_ny>

Figure 8: Inspection history

5. Restaurants with French cuisine

Query: SELECT Name FROM Restaurant WHERE CuisineType = 'French';

Execution Proof: This query works efficiently because we created a secondary index on 'CuisineType'.



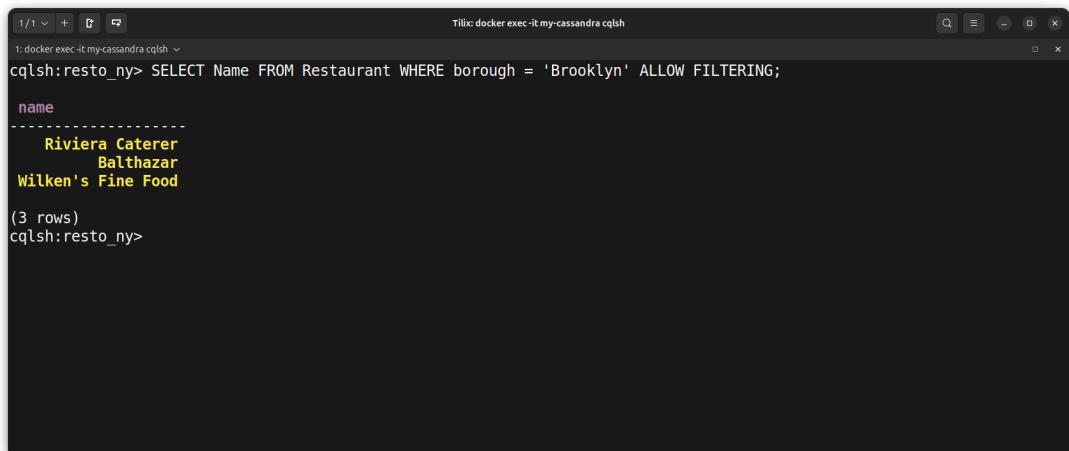
The screenshot shows a terminal window titled 'Tilix: docker exec -it my-cassandra cqlsh'. The command executed is 'cqlsh:resto_ny>'. The output is empty, indicating no results were found.

Figure 9: French Cuisine Restaurants

6. Restaurants in BROOKLYN (Allow Filtering)

Query: SELECT Name FROM Restaurant WHERE borough = 'BROOKLYN' ALLOW FILTERING;

Execution Proof: Since 'borough' is not indexed, 'ALLOW FILTERING' is required.



A screenshot of a terminal window titled "Tilix: docker exec -it my-cassandra cqlsh". The command run is "SELECT Name FROM Restaurant WHERE borough = 'Brooklyn' ALLOW FILTERING;". The output shows a table with one column "name" containing three rows: "Riviera Caterer", "Balthazar", and "Wilken's Fine Food". A message "(3 rows)" is displayed below the table. The prompt "cqlsh:resto_ny>" is at the bottom.

name
Riviera Caterer
Balthazar
Wilken's Fine Food

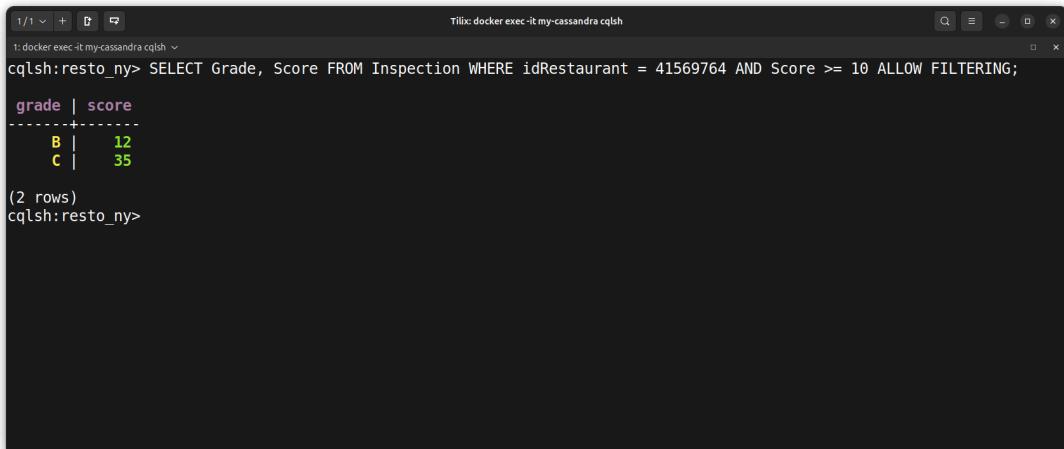
(3 rows)
cqlsh:resto_ny>

Figure 10: Brooklyn Restaurants

7. Grades for restaurant 41569764 with score ≥ 10

Query: SELECT Grade, Score FROM Inspection WHERE idRestaurant = 41569764 AND Score ≥ 10 ALLOW FILTERING;

Execution Proof: Filters inspections by both ID and score.



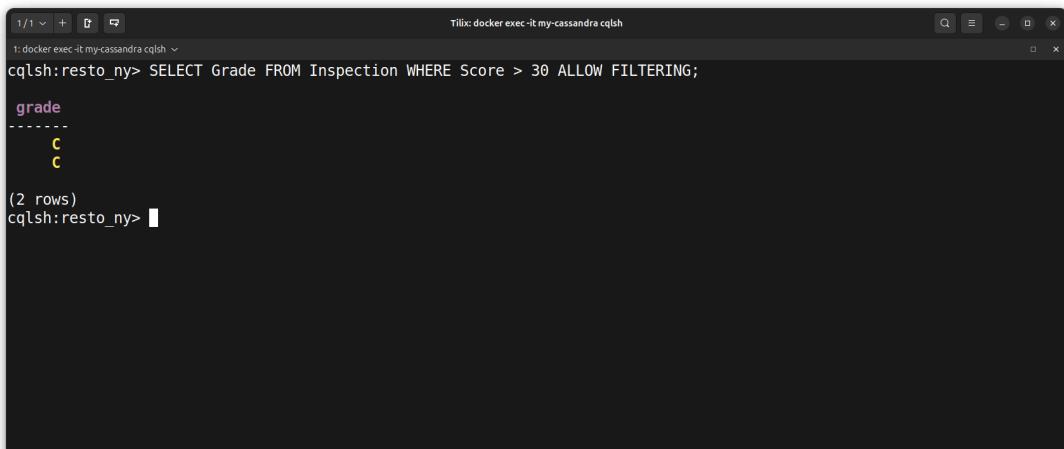
```
1/1 + ↻ Tilix: docker exec -it my-cassandra cqlsh
1: docker exec -it my-cassandra cqlsh >
cqlsh:resto_ny> SELECT Grade, Score FROM Inspection WHERE idRestaurant = 41569764 AND Score >= 10 ALLOW FILTERING;
grade | score
-----+-----
  B  |    12
  C  |    35
(2 rows)
cqlsh:resto_ny>
```

Figure 11: High scoring inspections

8. Grades where score < 30

Query: SELECT Grade FROM Inspection WHERE Score > 30 ALLOW FILTERING;

Execution Proof: Scans the table for high-risk inspections.



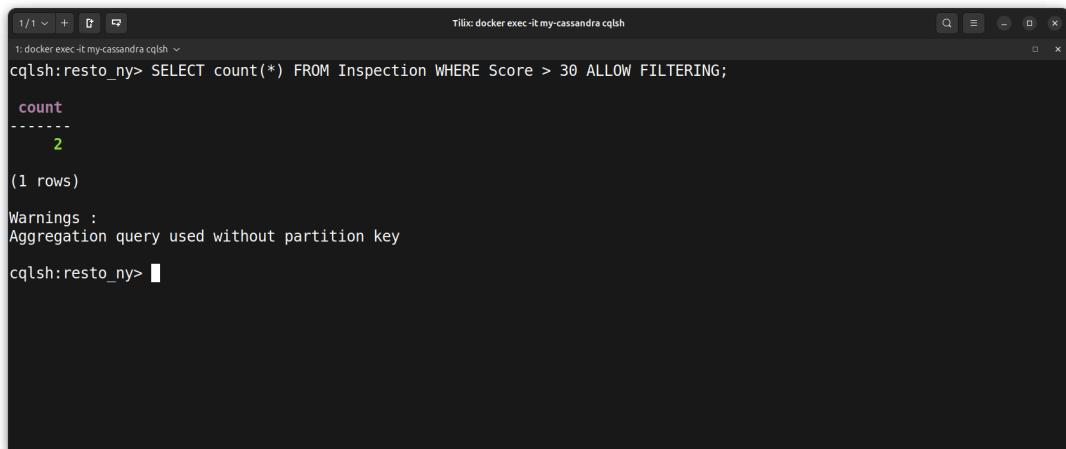
```
1/1 + ↻ Tilix: docker exec -it my-cassandra cqlsh
1: docker exec -it my-cassandra cqlsh >
cqlsh:resto_ny> SELECT Grade FROM Inspection WHERE Score > 30 ALLOW FILTERING;
grade
-----
  C
  C
(2 rows)
cqlsh:resto_ny>
```

Figure 12: Inspections with Score < 30

9. Count of rows from previous query

Query: SELECT COUNT(*) FROM Inspection WHERE Score > 30 ALLOW FILTERING;

Execution Proof: Returns the number of matching inspections.



A screenshot of a terminal window titled "Tilix: docker exec -it my-cassandra cqlsh". The window displays a command-line interface for the Cassandra database. The user has run the following query:

```
cqlsh:resto_ny> SELECT count(*) FROM Inspection WHERE Score > 30 ALLOW FILTERING;
```

The output shows a single row with the value "2". A warning message follows:

```
count
-----
 2
(1 rows)

Warnings :
Aggregation query used without partition key
```

```
cqlsh:resto_ny> █
```

Figure 13: Count result