NBA PostgreSQL Database CLI Documentation

Overview

This documentation provides comprehensive details on the use of the NBA PostgreSQL Database CLI (Command Line Interface). The system facilitates various operations such as inserting, deleting, updating, and querying data within the NBA database. Below, each function available through the CLI is detailed, explaining its purpose, usage, and example commands.

Prerequisites

Before using the CLI, ensure PostgreSQL is installed and configured on your system. Additionally, the database connection should be correctly configured in the db_connector module, which is utilized across the interface for establishing database connections.

Interface Functionality

1. Insert Data

Purpose: To add new records into any specified table within the database.

Command: 1

Inputs:

table_name: Name of the table.

columns: Comma-separated column names.

values: Corresponding values to the columns, comma-separated.

Example:

Enter the table name: Player

Enter the column names separated by commas: first_name,last_name,birthdate

Enter the values in the same order as columns: LeBron,James,1984-12-30

2. Delete Data

Purpose: To remove records from a table based on specified conditions.

Command: 2

Inputs:

table_name: Name of the table.

condition: Condition to match records for deletion.

Example:

Enter the table name: Player

Enter the condition for deletion: last_name = 'James'

3. Update Data

Purpose: To modify existing records in a table.

Command: 3

Inputs:

table_name: Name of the table.

set_clause: Changes to apply.

condition: Condition to select the records to update.

Example:

Enter the table name: Player

Enter the column and new value to set: height = 204

Enter the condition for the update: last_name = 'James'

4. Search Data

Purpose: To find records based on specified criteria.

Command: 4

Inputs:

table_name: Name of the table.

condition: Search condition.

Example:

Enter the table name: Player

Enter the search condition: last_name = 'James'

5. Aggregate Functions

Purpose: To perform calculations like sum, average, count, min, and max.

Command: 5

Inputs:

table_name: Name of the table.

column_name: Column to perform the aggregation on.

agg_function: Aggregate function (SUM, AVG, COUNT, MIN, MAX).

Example:

```
Enter the table name: Player
Enter the column name for aggregation: height
Enter the aggregate function to perform: AVG
```

6. Sorting

Purpose: To sort query results based on specified columns and order.

Command: 6

Inputs:

table_name: Name of the table.

column_name: Column to sort by.

sort_order: Order of sorting (ASC or DESC).

Example:

```
Enter the table name: Player
Enter the column name to sort by: last_name
Enter the sort order: ASC
```

7. Joins

Purpose: To combine data from multiple tables using specified conditions.

Command: 7

Inputs:

table1_name: First table name.

table2_name: Second table name.

join_condition: Condition for joining.

Example:

Enter the first table name: Player

Enter the second table name: Team

Enter the join condition: Player.team_id = Team.team_id

8. Grouping

Purpose: To group records based on specified columns, with optional aggregation.

Command: 8

Inputs:

table_name: Name of the table.

group_by_column: Columns to group by.

agg_function: Optional aggregate function.

Example:

Enter the table name: Player

Enter the column name(s) to group by: position

Optional - Enter the aggregate function and column: COUNT(person_id)

9. Subqueries

Purpose: To execute complex queries involving a subquery.

Command: 9

Inputs:

main_query: The outer SQL query with a placeholder for the subquery.

subquery: The inner query to embed.

Example:

Enter the main query, using '(SUBQUERY)' where the subquery should be inserted: SELECT * FROM Player WHERE person_id IN (SUBQUERY)

Enter the subquery: SELECT person_id FROM Player WHERE last_name = 'James'

10. Transactions

Purpose: To ensure the atomicity of multiple operations.

Command: 10

Inputs:

num_operations: Number of SQL operations in the transaction.

operations: Each SQL operation as per the sequence.

Example:

```
Enter the number of operations in the transaction: 2
Enter SQL operation 1: INSERT INTO Player (first_name, last_name) VALUES ('Kyrie', 'Irving')
Enter SQL operation 2: UPDATE Player SET team_id = 1 WHERE last_name = 'Irving'
```

11. Error Handling

Purpose: To manage exceptions and errors during database operations.

Command: 11

Inputs:

operation: SQL command that might raise an exception.

Example:

```
Enter the SQL command to execute: SELECT * FROM NonExistentTable
```

12. Exit

Purpose: To exit from the CLI interface.

Command: 12

Interpreting Query Outputs

Each time a query is executed, the query will be logged to Queries.log if successful. If unsuccessful, a corresponding error message to the query will be shown in the command line.

Additional Notes

This documentation serves as a guide for utilizing the CLI interface effectively. Users are encouraged to ensure all inputs are accurate to avoid errors during database operations. The system is designed to log all queries, providing a trace of operations for debugging and verification purposes.