PostgreSQL Commands Overview (Saichandan Gorli)

Quick Links

Login Steps in postresql :

1. Open the terminal with this path "C:\Program Files\PostgreSQL\16\bin"

2. Type the command "psql -U username"

3. Enter your password when prompted

4. If the username is not specified, it will prompt for it.

some useful commands

1. To list all databases, type \l

2. To connect to a database, type \c database_name

3. To list all tables, type \dt

4. To list all columns in a table, type \d table_name

5. To exit the psql prompt, type \q

6. To get help on a command, type ?

7. To clear the screen, type \clear

8. To get the current database, type \c

9. To get the current user, type \u

Administrative Commands

- **CREATE DATABASE**: Create a new database.

```
CREATE DATABASE database_name;
```

- **DROP DATABASE**: Delete a database.

```
DROP DATABASE database_name;
```

- **ALTER DATABASE**: Modify a database's properties.

```
ALTER DATABASE database_name SET parameter_name TO value;
```

Data Definition Language (DDL)

- **CREATE**: Create a new table, view, index, or database.

```
CREATE TABLE table_name (...);
```

- **ALTER**: Modify an existing database object.

```
ALTER TABLE table_name ADD COLUMN column_name data_type;
```

- **ADD PRIMARY KEY**: Add a primary key to a table.

```
ALTER TABLE table_name ADD PRIMARY KEY (column_name);
```

- **DROP CONSTRAINT**: Remove a constraint from a table.

```
ALTER TABLE table_name DROP CONSTRAINT constraint_name;
```

- **DROP**: Delete a database object.

```
DROP TABLE table_name;
```

- **TRUNCATE**: Remove all records from a table.

```
TRUNCATE TABLE table_name;
```

Data Manipulation Language (DML)

- **INSERT INTO**: Insert new records into a table.

```
INSERT INTO table_name VALUES (value1, value2, value3, ...);
```

- **Example**:

```
INSERT INTO employees VALUES ('John Doe', 30, 'Sales');
```

- **UPDATE**: Modify records in a table.

```
UPDATE table_name SET column1 = value1 WHERE condition;
```

- **Example**:

```
UPDATE employees SET age = 31 WHERE name = 'John Doe';
```

- **DELETE**: Remove records from a table.

```
DELETE FROM table_name WHERE condition;
```

- **Example**:

```
DELETE FROM employees WHERE age = 31;
```

- **SELECT**: Retrieve data from one or more tables.

```
SELECT * FROM table_name;
```

SELECT Command Variations

- **SELECT DISTINCT**: Retrieve unique records from a table.

```
SELECT DISTINCT column_name FROM table_name;
```

- **SELECT ALL**: Retrieve all records, including duplicates (default behavior).

```
SELECT ALL column_name FROM table_name;
```

- **WHERE Clause**: Filter records based on a condition.

```
SELECT * FROM table_name WHERE condition;
```

- **Example**:

```
SELECT * FROM employees WHERE amount > 200000;
```

- **LIKE Operator**: Search for a specified pattern in a column.

```
SELECT * FROM table_name WHERE column_name LIKE '<pattern>%';
```

- **Example**:

```
SELECT * FROM employees WHERE name LIKE 'S%';
```

- **IN Operator**: Check if a value exists in a list.

```
SELECT * FROM table_name WHERE column_name IN (value1, value2, ...);
```

- **Example**:

```
SELECT * FROM employees WHERE department IN ('CS', 'IT');
```

- **NOT Operator**: Exclude records that meet a condition.

```
SELECT * FROM table_name WHERE NOT condition;
```

- **Example**:

```
SELECT * FROM employees WHERE NOT department = 'Sales';
```

- **Order By** :Order By is used to sort the result set of a query based on one or more columns

```
SELECT cust_name FROM depositor ORDER BY cust_name;
```

Aggregate Functions

- **COUNT**: Count the number of rows in a table.

```
SELECT COUNT(column_name) FROM table_name;
```

- **SUM**: Calculate the sum of a column's values.

```
SELECT SUM(column_name) FROM table_name;
```

- **AVG**: Calculate the average value of a column's values.

```
SELECT AVG(column_name) FROM table_name;
```

- **MAX**: Find the maximum value in a column.

```
SELECT MAX(column_name) FROM table_name;
```

- **MIN**: Find the minimum value in a column.

```
SELECT MIN(column_name) FROM table_name;
```

Set Operations

- **INTERSECT**: Retrieve records common to both queries.

```
SELECT column_name FROM table_name1 INTERSECT SELECT column_name FROM table_name2;
```

- **UNION**: Combine results from two or more SELECT queries, excluding duplicates.

```
SELECT column_name FROM table_name1 UNION SELECT column_name FROM table_name2;
```

- **UNION ALL**: Combine results from two or more SELECT queries, including duplicates.

  ```
  SELECT column_name FROM table_name1 UNION ALL SELECT column_name FROM table_name2;
  ```

- **EXCEPT**: Retrieve records from the first query that are not in the second query.

  ```
  SELECT column_name FROM table_name1 EXCEPT SELECT column_name FROM table_name2;
  ```

- **EXCEPT ALL**: Retrieve records from the first query that are not in the second, including duplicates.

  ```
  SELECT column_name FROM table_name1 EXCEPT ALL SELECT column_name FROM table_name2;
  ```

Joins

- **INNER JOIN**: Retrieve records that have matching values in both tables.

  ```
  SELECT column_name FROM table_name1 INNER JOIN table_name2 ON
  table_name1.column_name = table_name2.column_name;
  ```

- **LEFT JOIN**: Retrieve all records from the left table and the matched records from the right table.

  ```
  SELECT column_name FROM table_name1 LEFT JOIN table_name2 ON
  table_name1.column_name = table_name2.column_name;
  ```

- **RIGHT JOIN**: Retrieve all records from the right table and the matched records from the left table.

  ```
  SELECT column_name FROM table_name1 RIGHT JOIN table_name2 ON
  table_name1.column_name = table_name2.column_name;
  ```