SQL Commands and Syntaxes

The Complete Tutorial
https://www.codecademy.com/articles/sql-commands

SELECT
Used to query or retrieve data from various columns of a table in the database.
SELECT column_list FROM table_name WHERE

- *column_name:* You can SELECT a single column, a list of columns (separated by commas) or all of the columns (using *)
- *table_name:* This is the table from which the data is SELECTed.
- *WHERE (optional):* used to select a particular row. You can filter the row using conditionals such as:
    - = equal
    - > greater than
    - < less than
    - >= greater or equal to
    - <= less or equal to
    - <> NOT equal to
    - LIKE select rows that starts/ends with a special char(s)
- ORDER BY: used to order the columns based on

Examples:

```
SELECT first, last, city FROM profiles WHERE age > 30
SELECT first FROM profiles WHERE first = 'Eric';
SELECT * FROM profiles WHERE last LIKE '%s'            #LIKE → ends with 's'
SELECT city FROM profiles WHERE first LIKE '%Er';      #LIKE → starts with 'Er'
SELECT first FROM profiles WHERE last '%John%';
```

SELECT DISTINCT
Specifies that SELECT statement is going to be a query that returns unique values in the specified columns(s).

ALTER TABLE
Used to add columns to a table

```
ALTER TABLE table name
ADD column_name datatype;
```

AND
Used to combine two conditions

```
...
WHERE column_1 = value_1
  AND column_2 = value_2;
```

AS
Allows you to rename a column or table until the end of the query
```
SELECT column_name AS 'Alias'
...
```

AVG()
Returns average value of a column
```
SELECT AVG(column_name)
...
```

BETWEEN
Used to filter result within a range of numbers, text or range
```
...
WHERE column_name BETWEEN value_1 AND value_2;
```

CASE (Pretty much IF-THEN logic)
Used to create different outputs (usually in the SELECT statement)
```
...
CASE
    WHEN condition THEN 'Result_1'
    WHEN condition THEN 'Result_2'
    ELSE 'Result_3'
END
```

COUNT()
Takes the name of the column as an argument counts the number of rows where the column is not NULL.
```
SELECT COUNT(column_name) ...
```

CREATE TABLE
Used to create a table (duh!)
```
CREATE TABLE table_name (
  column_1 datatype,
  column_2 datatype,
  column_3 datatype
);
```

DELETE
Removes rows from table
```
DELETE FROM table_name
WHERE some_column = some_value;
```

INSERT
Adds a new row to a table
```
INSERT INTO table_name (column_1, column_2, column_3)
VALUES (value_1, 'value_2', value_3);
```

GROUP BY
Used in collaboration with SELECT to arrange identical data into groups.
```
...
GROUP BY column_name;
```

INNER JOIN
Combines two rows from different tables if the condition is true
```
...
JOIN table_2
  ON table_1.column_name = table_2.column_name;
```

OUTER JOIN
Joins tables even if condition is not true. If condition is not met, NULL values are used to fill the columns.
```
...
LEFT JOIN table_2
  ON table_1.column_name = table_2.column_name;
```

IS NULL / IS NOT NULL
Used by WHERE clause to test for empty values
```
SELECT column_name(s)
FROM table_name
WHERE column_name IS NULL;
```

LIKE
Used with WHERE clause to search for a specific pattern in a column.
```
...
WHERE column_name LIKE pattern;
```

LIMIT
Used to specify maximum number of rows.

MAX() / MIN()
A function that takes the name of a column and return largest / smallest value in that column.
```
SELECT MAX(column_name)
FROM table_name;
```

OR
Operator that filters results set to include either conditions that are true
```
...
WHERE column_name = value_1
   OR column_name = value_2;
```

ORDER BY
ASC / DESC

ROUND()
Takes a column name and integer; It rounds the value in the column to the integer value specified
```
SELECT ROUND(column_name, integer)
...
```

SUM()
Returns the sum of all values in a column

UPDATE
Allows you to edit rows in a table
```
UPDATE table_name
SET some_column = some_value
WHERE some_column = some_value;
```

WHERE
Filters results to include rows where condition is true
```
...
WHERE column_name operator value;
```

WITH
Lets you store the results of a query in a temporary table using an alias.
```
WITH temporary_name AS (
   SELECT *
   FROM table_name)
SELECT *
FROM temporary_name
WHERE column_name operator value;
```

Running .sql file from psql
\i /path/to/file/file.sql