datacamp

SQL for Data Science

SQL Basics Cheat Sheet

Learn SQL online at www.DataCamp.com

What is SQL?

SQL stands for “structured query language”. It is a language used to query, analyze, and manipulate data from databases. Today, SQL is one of the most widely used tools in data.

> The different dialects of SQL

Although SQL languages all share a basic structure, some of the specific commands and styles can differ slightly. Popular dialects include MySQL, SQLite, SQL Server, Oracle SQL, and more. PostgreSQL is a good place to start—since it’s close to standard SQL syntax and is easily adapted to other dialects.

Analytics engines such as DuckDB, BigQuery, Snowflake, and Databricks SQL use the same core syntax shown here.

> Sample Data

Throughout this cheat sheet, we’ll use the columns listed in this sample table of `airbnb_listings`

airbnb_listings				
id	city	country	number_of_rooms	year_listed
1	Paris	France	5	2018
2	Tokyo	Japan	2	2017
3	New York	USA	2	2022

> Querying tables

1. Get all the columns from a table

```
SELECT *
FROM airbnb_listings;
```

2. Return the `city` column from the table

```
SELECT city
FROM airbnb_listings;
```

3. Get the `city` and `year_listed` columns from the table

```
SELECT city, year_listed
FROM airbnb_listings;
```

4. Get the listing `id`, `city`, ordered by the `number_of_rooms` in ascending order

```
SELECT id, city
FROM airbnb_listings
ORDER BY number_of_rooms ASC;
```

5. Get the listing `id`, `city`, ordered by the `number_of_rooms` in descending order

```
SELECT id, city
FROM airbnb_listings
ORDER BY number_of_rooms DESC;
```

6. Get the first 5 rows from the `airbnb_listings` table

```
SELECT *
FROM airbnb_listings
LIMIT 5;
```

7. Get a unique list of cities where there are listings

```
SELECT DISTINCT city
FROM airbnb_listings;
```

8. Get the listing `id`, `city`, ordered by `country`, then by `number_of_rooms` in descending order

```
SELECT id, city
FROM airbnb_listings
ORDER BY country ASC, number_of_rooms DESC;
```

9. Skip the first 5 rows, then get the next 10 rows

```
SELECT *
FROM airbnb_listings
LIMIT 10 OFFSET 5;
```

10. Rename the `city` column to `location` using an alias

```
SELECT city AS location
FROM airbnb_listings;
```

11. Label each listing as `'Large'` or `'Small'` based on its `number_of_rooms`

```
SELECT city,
  CASE WHEN number_of_rooms >= 3 THEN 'Large'
       ELSE 'Small'
  END AS size
FROM airbnb_listings;
```

12. Add comments to explain your code

```
-- A single-line comment
/* A comment that spans
   several lines */
SELECT *
FROM airbnb_listings;
```

> Filtering Data

Filtering on numeric columns

1. Get all the listings where `number_of_rooms` is more or equal to 3

```
SELECT *
FROM airbnb_listings
WHERE number_of_rooms >= 3;
```

2. Get all the listings where `number_of_rooms` is more than 3

```
SELECT *
FROM airbnb_listings
WHERE number_of_rooms > 3;
```

3. Get all the listings where `number_of_rooms` is exactly equal to 3

```
SELECT *
FROM airbnb_listings
WHERE number_of_rooms = 3;
```

4. Get all the listings where `number_of_rooms` is lower or equal to 3

```
SELECT *
FROM airbnb_listings
WHERE number_of_rooms <= 3;
```

5. Get all the listings where `number_of_rooms` is lower than 3

```
SELECT *
FROM airbnb_listings
WHERE number_of_rooms < 3;
```

6. Get all the listings with 3 to 6 rooms

```
SELECT *
FROM airbnb_listings
WHERE number_of_rooms BETWEEN 3 AND 6;
```

Filtering on text columns

7. Get all the listings that are based in `'Paris'`

```
SELECT *
FROM airbnb_listings
WHERE city = 'Paris';
```

8. Get the listings based in the `'USA'` and in `'France'`

```
SELECT *
FROM airbnb_listings
WHERE country IN ('USA', 'France');
```

9. Get all the listings where the city starts with `'j'` and where the city does not end in `'t'`

```
SELECT *
FROM airbnb_listings
WHERE city LIKE 'j%' AND city NOT LIKE '%t';
```

10. Get all the listings where the city is `'P'`, any single character, then `'ris'`

```
SELECT *
FROM airbnb_listings
WHERE city LIKE 'P_ris';
```

Filtering on multiple columns

11. Get all the listings in `'Paris'` where `number_of_rooms` is bigger than 3

```
SELECT *
FROM airbnb_listings
WHERE city = 'Paris' AND number_of_rooms > 3;
```

12. Get all the listings in `'Paris'` OR the ones that were listed after 2012

```
SELECT *
FROM airbnb_listings
WHERE city = 'Paris' OR year_listed > 2012;
```

Filtering on missing data

13. Return the listings where `number_of_rooms` is missing

```
SELECT *
FROM airbnb_listings
WHERE number_of_rooms IS NULL;
```

14. Return the listings where `number_of_rooms` is not missing

```
SELECT *
FROM airbnb_listings
WHERE number_of_rooms IS NOT NULL;
```

> Aggregating Data

Simple aggregations

1. Get the total number of rooms available across all listings

```
SELECT SUM(number_of_rooms)
FROM airbnb_listings;
```

2. Get the average number of rooms per listing across all listings

```
SELECT ROUND(AVG(number_of_rooms), 1)
FROM airbnb_listings;
```

3. Get the listing with the highest number of rooms across all listings

```
SELECT MAX(number_of_rooms)
FROM airbnb_listings;
```

4. Get the listing with the lowest number of rooms across all listings

```
SELECT MIN(number_of_rooms)
FROM airbnb_listings;
```

5. Count all the rows in the table

```
SELECT COUNT(*)
FROM airbnb_listings;
```

6. Count the listings where `number_of_rooms` is not missing

```
SELECT COUNT(number_of_rooms)
FROM airbnb_listings;
```

7. Count the number of unique cities

```
SELECT COUNT(DISTINCT city)
FROM airbnb_listings;
```

Grouping, filtering, and sorting

8. Get the total number of rooms for each country

```
SELECT country, SUM(number_of_rooms)
FROM airbnb_listings
GROUP BY country;
```

9. Get the average number of rooms for each country

```
SELECT country, ROUND(AVG(number_of_rooms), 1)
FROM airbnb_listings
GROUP BY country;
```

10. Get the listing with the maximum number of rooms per country

```
SELECT country, MAX(number_of_rooms)
FROM airbnb_listings
GROUP BY country;
```

11. Get the listing with the lowest amount of rooms per country

```
SELECT country, MIN(number_of_rooms)
FROM airbnb_listings
GROUP BY country;
```

12. For each country, get the average number of rooms per listing, sorted by ascending order

```
SELECT country, ROUND(AVG(number_of_rooms), 1) AS avg_rooms
FROM airbnb_listings
GROUP BY country
ORDER BY avg_rooms ASC;
```

13. For Japan and the USA, get the average number of rooms per listing in each country

```
SELECT country, ROUND(AVG(number_of_rooms), 1)
FROM airbnb_listings
WHERE country IN ('USA', 'Japan')
GROUP BY country;
```

14. Get the number of listings per country

```
SELECT country, COUNT(id) AS number_of_listings
FROM airbnb_listings
GROUP BY country;
```

15. Get all the years where there were more than 100 listings per year

```
SELECT year_listed
FROM airbnb_listings
GROUP BY year_listed
HAVING COUNT(id) > 100;
```

Order of SQL clauses

Clauses always appear in this order. `WHERE` filters rows before grouping; `HAVING` filters groups after

```
SELECT columns
FROM table
WHERE row_condition
GROUP BY columns
HAVING group_condition
ORDER BY columns
LIMIT n;
```

Keep learning

Combining tables? Get the **SQL Joins Cheat Sheet**

Build your skills with the **SQL Fundamentals** skill track

Learn Data Skills Online at www.DataCamp.com