SQL Basics Cheat Sheet

SQL

SQL, or Structured Query Language, is a language to talk to databases. It allows you to select specific data and to build complex reports. Today, SQL is a universal language of data. It is used in practically all technologies that process data.

SAMPLE DATA

<table>
<thead>
<tr>
<th>City</th>
<th>Population</th>
<th>Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paris</td>
<td>2,140,000</td>
<td>108 km²</td>
</tr>
<tr>
<td>Berlin</td>
<td>3,530,000</td>
<td>895 km²</td>
</tr>
<tr>
<td>Madrid</td>
<td>3,200,000</td>
<td>663 km²</td>
</tr>
</tbody>
</table>

FILTERING THE OUTPUT

Comparison Operators

Fetch names of cities that have a rating above 3:

```
SELECT name FROM city WHERE rating > 3;
```

Fetch names of cities that are neither Berlin nor Madrid:

```
SELECT name FROM city WHERE name != 'Berlin' AND name != 'Madrid';
```

Fetch names of cities that start with "A" or end with an "s":

```
SELECT name FROM city WHERE name LIKE 'A%' OR name LIKE '%s';
```

Aliases

SELECT name AS city_name FROM city;

Tables

SELECT country.name FROM country;

AGGREGATION AND GROUPING

Groups by cities, returns all possible combinations of rows from both tables. There are two syntaxes available.

```
SELECT city.name, country.name
FROM city
CROSS JOIN country;
```

```
SELECT city.name, country.name
FROM city, country
ON city.country_id = country.id;
```

Subqueries

A subquery is a query that is nested inside another query, or inside another subquery. There are different types of subqueries.

Single Value

The simplest subquery returns exactly one column and exactly one row. It can be used with comparison operators <, <=, >, or >=

This query finds cities with the same rating as Paris:

```
SELECT name FROM city
WHERE rating = (SELECT rating FROM city
WHERE name = 'Paris');
```

Multiple Values

A subquery can also return multiple columns or multiple rows. Such subqueries can be used with operators IN, EXISTS, ALL, or ANY.

This query finds cities in countries that have a population above 20 million:

```
SELECT name FROM city
WHERE country_id IN (SELECT country_id FROM country
WHERE population > 20000000);
```

Correlated

A correlated subquery refers to the tables introduced in the outer query. A correlated subquery depends on the outer query. It cannot be run independently from the outer query.

This query finds cities with a population greater than the average population in the country:

```
SELECT name FROM city
WHERE population > (SELECT AVG(population) FROM city
WHERE country_id = city.country_id)
```

UNION

UNION combines the results of two result sets and removes duplicates. UNION ALL doesn't remove duplicates.

This query displays German cyclists together with German skaters:

```
SELECT name FROM cycling
WHERE country = 'DE';
UNION
SELECT name FROM skating
WHERE country = 'DE';
```

INTERSECT

INTERSECT returns only rows that appear in both result sets.

This query displays German cyclists who are also German skaters at the same time:

```
SELECT name FROM cycling
WHERE country = 'DE'
INTERSECT
SELECT name FROM skating
WHERE country = 'DE';
```

EXCEPT

EXCEPT returns only the rows that appear in the first result set but do not appear in the second result set.

This query displays German cyclists unless they are also German skaters at the same time:

```
SELECT name FROM cycling
WHERE country = 'DE'
EXCEPT
SELECT name FROM skating
WHERE country = 'DE';
```

Try out the interactive SQL Basics course at LearnSQL.com, and check out our other SQL courses.