SQL Basics Cheat Sheet

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

COUNTRY							
id	nar	name		population		area	
1	Fran	France		66600000		640680	
2	Germ	Germany		80700000		357000	
CITY							
id	name	count	try_id	populatio	n	rating	
1	Paris		1	2243000		5	
2	Berlin		2	3460000		3	
• • •			• •				

QUERYING SINGLE TABLE

```
Fetch all columns from the country table:
```

SELECT *
FROM country;

Fetch id and name columns from the city table:

SELECT id, name
FROM city;

Fetch city names sorted by the rating column in the default ASCending order:

SELECT name
FROM city
ORDER BY rating [ASC];

Fetch city names sorted by the rating column in the DESCending order:

SELECT name FROM city ORDER BY rating DESC;

ALIASES

COLUMNS

SELECT name AS city_name
FROM city;

TABLES

```
SELECT co.name, ci.name
FROM city AS ci
JOIN country AS co
   ON ci.country_id = co.id;
```

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';

TEXT OPERATORS

Fetch names of cities that start with a 'P' or end with an 's': SELECT name FROM city

WHERE name LIKE 'P%'
OR name LIKE '%s';

Fetch names of cities that start with any letter followed by 'ublin' (like Dublin in Ireland or Lublin in Poland):

SELECT name
FROM city
WHERE name LIKE '_ublin';

OTHER OPERATORS

Fetch names of cities that have a population between 500K and 5M:

SELECT name
FROM city
WHERE population BETWEEN 500000 AND 5000000;

Fetch names of cities that don't miss a rating value:

SELECT name FROM city WHERE rating IS NOT NULL;

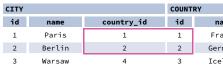
Fetch names of cities that are in countries with IDs 1, 4, 7, or 8: SELECT name FROM city WHERE country_id IN (1, 4, 7, 8);

QUERYING MULTIPLE TABLES

JOIN (or explicitly **INNER JOIN**) returns rows that have matching values in both tables.

SELECT city.name, country.name FROM city
[INNER] JOIN country

ON city.country_id = country.id;



LEFT JOIN

LEFT JOIN returns all rows from the left table with corresponding rows from the right table. If there's no mat row, **NULL**s are returned as values from the second table.

SELECT city.name, country.name
FROM city

LEFT JOIN country
ON city.country_id = country.id;

	COUNTRY			
name	country_id	id	n	
Paris	1	1	Fr	
Berlin	2	2	Ger	
Warsaw	4	NULL	N	
	Paris Berlin	Paris 1 Berlin 2	Paris 1 1 Berlin 2 2	

RIGHT JOIN

RIGHT JOIN returns all rows from the right table with corresponding rows from the left table. If there's no match row, **NULL**s are returned as values from the left table.

SELECT city.name, country.name
FROM city

RIGHT JOIN country

ON city.country_id = country.id;

CITY			COUNTRY	
id	name	country_id	id	na
1	Paris	1	1	Fra
2	Berlin	2	2	Ger
NULL	NULL	NULL	3	Ice

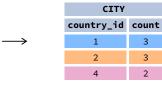
Try out the interactive **SQL Basics** course at **LearnSQL.com**, and check out our other SQL courses.

SQL Basics Cheat Sheet

AGGREGATION AND GROUPING

GROUP BY **groups** together rows that have the same values in specified columns. It computes summaries (aggregates) for each unique combination of values.

CITY				
id	name	country_id		
1	Paris	1		
101	Marseille	1		
102	Lyon	1		
2	Berlin	2		
103	Hamburg	2		
104	Munich	2		
3	Warsaw	4		
105	Cracow	4		



AGGREGATE FUNCTIONS

- avg(expr) average value for rows within the group
- count(expr) count of values for rows within the group
- max(expr) maximum value within the group
- min(expr) minimum value within the group
- sum(expr) sum of values within the group

EXAMPLE QUERIES

Find out the number of cities:

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

Find out the number of cities with non-null ratings:

```
SELECT COUNT(rating)
FROM city;
```

Find out the number of distinctive country values:

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

Find out the smallest and the greatest country populations:

```
SELECT MIN(population), MAX(population)
FROM country;
```

Find out the total population of cities in respective countries:

```
SELECT country_id, SUM(population)
FROM city
GROUP BY country_id;
```

Find out the average rating for cities in respective countries if the average is above 3.0:

```
SELECT country_id, AVG(rating)
FROM city
GROUP BY country_id
HAVING AVG(rating) > 3.0;
```

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 20M:

```
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 *
FROM city main_city
WHERE population > (
    SELECT AVG(population)
    FROM city average_city
    WHERE average_city.country_id = main_city.country_id
);
```

This query finds countries that have at least one city:

```
SELECT name
FROM country
WHERE EXISTS (
    SELECT *
    FROM city
    WHERE country_id = country.id
);
```

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

SET C

Set opera The comb types. The

CYCL:

10

UNION CO

doesn't re

This query SELECT FROM CY WHERE COUNION / SELECT

INTERSE

This quer

SELECT

FROM sk

WHERE C

FROM CY WHERE C INTERSE SELECT FROM Sk WHERE C

EXCEPT is second re

This query time: SELECT FROM CY WHERE C

SELECT FROM sk WHERE o