SQL Basics Cheat Sheet

LearnSQL
• com

SQL

SQL [redacted]*ctured Query Language* [redacted]
databases. It allows you to select specific data and to build
[redacted]

SAMPLE DATA

COUNTRY			
id	name	population	area
1	France	66600000	640680
2	Germany	80700000	357000
...	...	...	...

CITY				
id	name	country_id	population	rating
1	Paris	1	2243000	5
2	Berlin	2	3460000	3
...	...	...	...	...

QUERYING SINGLE TABLE

[redacted]ntry [redacted]

```
SELECT *
FROM country;
```

[redacted]e [redacted]y [redacted]

```
SELECT id, name
FROM city;
```

[redacted]ing [redacted]

```
SELECT name
FROM city
ORDER BY rating [ASC];
```

[redacted]ing [redacted]C [redacted]

```
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

[redacted]

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

[redacted]

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

TEXT OPERATORS

[redacted]

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

[redacted]

```
SELECT name
FROM city
WHERE name LIKE '_ublin';
```

OTHER OPERATORS

[redacted]

```
SELECT name
FROM city
WHERE population BETWEEN 500000 AND 5000000;
```

[redacted]

```
SELECT name
FROM city
WHERE rating IS NOT NULL;
```

[redacted]

```
SELECT name
FROM city
WHERE country_id IN (1, 4, 7, 8);
```

QUERYING MULTIPLE TABLES

INNER JOIN

JOIN [redacted]**ER JOIN** [redacted]

```
SELECT city.name, country.name
FROM city
[INNER] JOIN country
  ON city.country_id = country.id;
```

CITY			COUNTRY	
id	name	country_id	id	name
1	Paris	1	1	France
2	Berlin	2	2	Germany
3	Warsaw	4	3	Iceland

LEFT JOIN

LEFT JOIN returns all rows from the left table with
[redacted]

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

CITY			COUNTRY	
id	name	country_id	id	name
1	Paris	1	1	France
2	Berlin	2	2	Germany
3	Warsaw	4	NULL	NULL

RIGHT JOIN

RIGHT JOIN [redacted]
corresponding rows from the left table. If there's no
[redacted]s are returned as values from the left
[redacted]

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

CITY			COUNTRY	
id	name	country_id	id	name
1	Paris	1	1	France
2	Berlin	2	2	Germany
NULL	NULL	NULL	3	Iceland

FULL JOIN

FULL JOIN [redacted] **OUTER JOIN** [redacted]

```
SELECT city.name, country.name
FROM city
FULL [OUTER] JOIN country
  ON city.country_id = country.id;
```

CITY			COUNTRY	
id	name	country_id	id	name
1	Paris	1	1	France
2	Berlin	2	2	Germany
3	Warsaw	4	NULL	NULL
NULL	NULL	NULL	3	Iceland

CROSS JOIN

CROSS JOIN [redacted]

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

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

CITY			COUNTRY	
id	name	country_id	id	name
1	Paris	1	1	France
1	Paris	1	2	Germany
2	Berlin	2	1	France
2	Berlin	2	2	Germany

NATURAL JOIN

NATURAL JOIN [redacted]

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

CITY			COUNTRY	
country_id	id	name	name	id
6	6	San Marino	San Marino	6
7	7	Vatican City	Vatican City	7
5	9	Greece	Greece	9
10	11	Monaco	Monaco	10

NATURAL JOIN [redacted]
city.id, city.name, country.id, country.name [redacted]
NATURAL JOIN [redacted]

SQL Basics Cheat Sheet

LearnSQL • com

AGGREGATION AND GROUPING

GROUP BY gro[illegible]ups together rows that have the same values in specified columns. [illegible]

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

→

CITY

country_id	count
1	3
2	3
4	2

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

[illegible]

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

[illegible]

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

[illegible]

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

[illegible]

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

[illegible]

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

[illegible]

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

SUBQUERIES

[illegible]
There are different types of subqueries.

SINGLE VALUE

[illegible]

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

[illegible] E[illegible]STS A[illegible]

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

[illegible]

This query finds cities with a population greater than the average population in the [illegible]

```
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
);
```

SET OPERATIONS

[illegible] compatible data types. The names of the corresponding columns can be different.

CYCLING

id	name	country
1	YK	DE
2	ZG	DE
3	WT	PL
...	...	...

SKATING

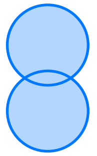
id	name	country
1	YK	DE
2	DF	DE
3	AK	PL
...	...	...

UNION

UNION [illegible]
UNION ALL [illegible]

[illegible]

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

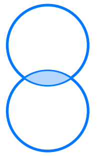


INTERSECT

INTERSECT [illegible]

[illegible]

```
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 [illegible]

[illegible]

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