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**

<table>
<thead>
<tr>
<th>country</th>
<th>id</th>
<th>name</th>
<th>population</th>
</tr>
</thead>
<tbody>
<tr>
<td>France</td>
<td>1</td>
<td>Paris</td>
<td>8.07M</td>
</tr>
<tr>
<td>Germany</td>
<td>2</td>
<td>Berlin</td>
<td>0.35M</td>
</tr>
</tbody>
</table>

**FILTERING THE OUTPUT**

**COMPARISON OPERATORS**

- `=`: Select names of cities that have a rating above 3.
  ```sql
  SELECT name
  FROM city
  WHERE rating > 3;
  ```
- `!=`: Select names of cities that are neither Berlin nor Madrid.
  ```sql
  SELECT name
  FROM city
  WHERE name != 'Berlin' AND name != 'Madrid';
  ```

**TEXT OPERATORS**

- `LIKE`:
  ```sql
  SELECT name
  FROM city
  WHERE name LIKE '_ublin';
  ```
- `ILIKE`:
  ```sql
  SELECT name
  FROM city
  WHERE name ILIKE 'B%';
  ```

**OTHER OPERATORS**

- `IS NULL`:
  ```sql
  SELECT name
  FROM city
  WHERE rating IS NULL;
  ```
- `IS NOT NULL`:
  ```sql
  SELECT name
  FROM city
  WHERE rating IS NOT NULL;
  ```

**AGGREGATE FUNCTIONS**

- `count()`:
  ```sql
  SELECT count(id)
  FROM city;
  ```
- `sum()`:
  ```sql
  SELECT sum(population)
  FROM country;
  ```
- `avg()`:
  ```sql
  SELECT avg(rating)
  FROM city
  HAVING avg(rating) > 3.0;
  ```
- `min()`:
  ```sql
  SELECT min(population)
  FROM country;
  ```
- `max()`:
  ```sql
  SELECT max(population)
  FROM country;
  ```

**EXAMPLE QUERIES**

**FINDING THE NUMBER OF CITIES**

Find the number of cities:
```sql
SELECT COUNT(name)
FROM city;
```
Find the number of cities with null ratings:
```sql
SELECT COUNT(name)
FROM city
WHERE rating IS NULL;
```
Find the number of cities with non-null ratings:
```sql
SELECT COUNT(name)
FROM city
WHERE rating IS NOT NULL;
```
Find the number of cities with null population values:
```sql
SELECT COUNT(id)
FROM country
WHERE population IS NULL;
```
Find the number of cities with population values:
```sql
SELECT COUNT(id)
FROM country
WHERE population IS NOT NULL;
```
Find the number of cities with population values:
```sql
SELECT COUNT(id)
FROM country
WHERE population IS NOT NULL;
```

**AGGREGATION AND GROUPING**

**GROUP BY**:

Find the number of cities by country:
```sql
SELECT country, COUNT(id)
FROM city
GROUP BY country;
```
Find the average city rating for cities in France:
```sql
SELECT AVG(rating)
FROM city
WHERE country = 'FR';
```
Find the average city rating for cities in Germany:
```sql
SELECT AVG(rating)
FROM city
WHERE country = 'DE';
```

**SUBQUERIES**

A subquery is a query that is nested inside another query. 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 (>, <, <=, >=, =). Example:
  ```sql
  SELECT name
  FROM city
  WHERE rating = (SELECT AVG(rating)
  FROM city
  WHERE rating = 'Paris');
  ```

**Multiple values**

- A subquery can also return multiple rows. Such subqueries can be used with operators IN, EXISTS, ALL, or ANY. Example:
  ```sql
  SELECT name
  FROM city
  WHERE country IN (SELECT country_id
  FROM city
  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. Example:
```sql
SELECT country, city
FROM city, country
WHERE average.city.country_id IN (SELECT country_id
FROM city
WHERE rating < 5);"