--- query single table

**Sample Data**

<table>
<thead>
<tr>
<th>COUNTRY</th>
<th>id</th>
<th>name</th>
<th>population</th>
<th>area</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>France</td>
<td>66600000</td>
<td>640680</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Germany</td>
<td>80700000</td>
<td>357000</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CITY</th>
<th>id</th>
<th>name</th>
<th>country_id</th>
<th>population</th>
<th>rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Paris</td>
<td>1</td>
<td>2243000</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Berlin</td>
<td>2</td>
<td>3460000</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

--- query multiple tables

--- city names with their country names

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

--- fetch only cities with non-null matching countries

```sql
city (INNER) JOIN country
```

--- fetch all cities even if no matching country exists

```sql
city LEFT (OUTER) JOIN country
```

--- fetch all countries even if no matching cities exist

```sql
city RIGHT (OUTER) JOIN country
```

--- fetch all cities and all countries even if no matching values exist in the other table

```sql
city FULL (OUTER) JOIN country
```
### AGGREGATION/GROUPING

-- cities with highest rating first

```
SELECT name FROM city ORDER BY rating DESC;
```

-- number of all cities

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

-- number of cities with non-null rating

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

-- number of distinctive country values

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

-- smallest and greatest country population

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

-- total population of cities in respective countries

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

-- 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;
```

### CORRELATED

-- cities with population greater than 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);
```

-- countries that have at least one city

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

### SET OPERATIONS

<table>
<thead>
<tr>
<th>CYCLING</th>
<th>SKATING</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>name</td>
</tr>
<tr>
<td>id</td>
<td>name</td>
</tr>
<tr>
<td>1</td>
<td>YK</td>
</tr>
<tr>
<td>2</td>
<td>ZG</td>
</tr>
<tr>
<td>3</td>
<td>WT</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

-- German cyclers together with German skaters (ALL shows repetitions)

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

-- German cyclers that are also German skaters at the same time

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

-- German cyclers unless they are also German skaters at the same time

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

### SUBQUERIES

#### SINGLE VALUE

-- cities with the same rating as Paris

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

#### MULTIPLE VALUES

-- 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);
```