Outline

1. What is a Query? Query Language?
2. Example Database Tables
3. SQL Overview: 3 Components
4. SELECT statement with 1 table
5. Multi-table SELECT statements
6. Why spatial extensions are needed?
7. 1-table spatial queries
8. Multi-table spatial queries
9. Trends
Learning Objectives

Upon completion of this module, students will be able to:

- Determine output of a single-Table SQL query
- Compose a single-Table SQL query
SQL Data Manipulation Language: SELECT Statement

• Purpose: Query data from database tables
  • Returns a table as result

• Features
  • Has many clauses
  • Can refer to many operators and functions
  • Allows nested queries
SQL SELECT Statement: Scope of Our Discussion

• Learn enough to appreciate spatial aspects
  • Observe example queries
• Read & compose simple SELECT statement
  • With frequently used clauses
    • e.g., SELECT, FROM, WHERE, ...
  • And a few operators and functions
Clauses of SELECT Statement

- Mandatory Clauses
  - SELECT specifies desired columns
  - FROM specifies relevant tables

- Optional Clauses
  - WHERE specifies qualifying conditions for results
  - ORDER BY specifies sorting columns for results
  - GROUP BY, HAVING specifies aggregation and statistics
SELECT Statement- operators, functions

- Arithmetic operators, e.g. +, -, ...
- Comparison operators, e.g. =, <, >, BETWEEN, LIKE, ...
- Logical operators, e.g. AND, OR, NOT, EXISTS,
- Statistical functions, e.g. SUM, COUNT, ...
- Set operators, e.g. UNION, IN, ALL, ANY,...
- Many other operators on strings, data, currency, ...
Ex. 1: Simplest SELECT query

- **Query**: List all the cities with their country.

<table>
<thead>
<tr>
<th>CITY</th>
<th>Name</th>
<th>Country</th>
<th>Pop (millions)</th>
<th>Captial</th>
<th>Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Havana</td>
<td>Cuba</td>
<td>2.1</td>
<td>Y</td>
<td></td>
<td>Pointid-1</td>
</tr>
<tr>
<td>Washington</td>
<td>USA</td>
<td>3.2</td>
<td>Y</td>
<td></td>
<td>Pointid-2</td>
</tr>
<tr>
<td>Monterrey</td>
<td>Mexico</td>
<td>2.0</td>
<td>N</td>
<td></td>
<td>Pointid-3</td>
</tr>
<tr>
<td>Toronto</td>
<td>Canada</td>
<td>3.4</td>
<td>N</td>
<td></td>
<td>Pointid-4</td>
</tr>
<tr>
<td>Brasilia</td>
<td>Brazil</td>
<td>1.5</td>
<td>Y</td>
<td></td>
<td>Pointid-5</td>
</tr>
<tr>
<td>Rosario</td>
<td>Argentina</td>
<td>1.1</td>
<td>N</td>
<td></td>
<td>Pointid-6</td>
</tr>
<tr>
<td>Ottawa</td>
<td>Canada</td>
<td>0.8</td>
<td>Y</td>
<td></td>
<td>Pointid-7</td>
</tr>
<tr>
<td>Mexico City</td>
<td>Mexico</td>
<td>14.1</td>
<td>Y</td>
<td></td>
<td>Pointid-8</td>
</tr>
<tr>
<td>Buenos Aires</td>
<td>Argentina</td>
<td>10.75</td>
<td>Y</td>
<td></td>
<td>Pointid-9</td>
</tr>
</tbody>
</table>
Commonly 3 clauses (SELECT, FROM, WHERE) are used

- **Query**: List the names of the capital cities in the CITY table.

SELECT *
FROM CITY
WHERE CAPITAL='Y'

<table>
<thead>
<tr>
<th>Name</th>
<th>Country</th>
<th>Pop (millions)</th>
<th>Capital</th>
<th>Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Havana</td>
<td>Cuba</td>
<td>2.1</td>
<td>Y</td>
<td>Point</td>
</tr>
<tr>
<td>Washington, D.C.</td>
<td>USA</td>
<td>3.2</td>
<td>Y</td>
<td>Point</td>
</tr>
<tr>
<td>Brasilia</td>
<td>Brazil</td>
<td>1.5</td>
<td>Y</td>
<td>Point</td>
</tr>
<tr>
<td>Ottawa</td>
<td>Canada</td>
<td>0.8</td>
<td>Y</td>
<td>Point</td>
</tr>
<tr>
<td>Mexico City</td>
<td>Mexico</td>
<td>14.1</td>
<td>Y</td>
<td>Point</td>
</tr>
<tr>
<td>Buenos Aires</td>
<td>Argentina</td>
<td>10.75</td>
<td>Y</td>
<td>Point</td>
</tr>
</tbody>
</table>
**SELECT with Aliasing**

**Query:** List names and Life-expectancy for countries, where the life-expectancy is less than seventy years.

```
SELECT Co.Name, Co.Life-Exp
FROM Country Co
WHERE Co.Life-Exp < 70
```

Note: use of alias ‘Co’ for Table ‘Country’

<table>
<thead>
<tr>
<th>Name</th>
<th>Life-Exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mexico</td>
<td>69.36</td>
</tr>
<tr>
<td>Brazil</td>
<td>65.60</td>
</tr>
</tbody>
</table>
SELECT: Aggregate Queries

**Query**: What is the average population of the capital cities?

```sql
SELECT AVG(Ci.Pop) 
FROM City Ci 
WHERE Ci.Capital = 'Y'
```

**Query**: For each continent, find the average GDP.

```sql
SELECT Co.Cont, Avg (Co.GDP) AS Continent-GDP 
FROM Country Co 
GROUP BY Co.Cont
```
SELECT: HAVING Clause

Query: For each country in which at least two rivers originate, find the length of the smallest river.

SELECT R.Origin, MIN (R.length) AS Min-length
FROM River
GROUP BY R.Origin
HAVING COUNT(*) > 1