SQL DATA WITH BARAA

DataWithBaraa

SQL Basics Part 1

```
--------------------
-- 01 What Is SQL --
--------------------
-- Database --> is a "container" that store the data
-- SQL --> is the "language" use in order to talk the database
-- SQL stands for structure query language, and is the language speak to the
database
-- Databases are secure, and speed, and you can control who can access, ...

------------------------------
-- What Is DBMS & SQL Server --
------------------------------
-- how many people are interacting with applications and the websites, we use DBMS
-- DBMS is the database management system, is the software that can "manage" the
database
-- DBMS determine which command will execute first,
-- Schema: helps you to organize your tables and your objects in the database, or it
is an organization of table
-- Table: like spread sheet that organize the data into columns, it collection of
data organized in rows and columns
-- Column: is a vertical part of the table, each column has one type of data.
-- Row: represents a single record in a table
-- Record: is the same as Row on the SQL, it represents on complete set of related
data in table
-- Data Types can be: INT, DECIMAL, CHAR, VARCHAR, DATE, TIME
```

```
/*
    ========================
    == 02 SELECT Statement ==
    ========================
    - used to filter the data from the table
    - syntax
        SELECT column1, column2, ..
        FROM table_name;
*/

-- select specific records from the table
SELECT
      CustomerID,
    CustomerName
FROM Customers;

-- select specific columns from the table
SELECT
      OrderID,
    OrderDate,
    CustomerID
FROM Orders;

-- select all columns from the table
SELECT *
FROM Customers;
```

```sql
---------------------
-- SELECT DISTINCT --
---------------------
-- used to select the different values from the table


-- find all different countries on the table
SELECT DISTINCT country
FROM customers;


-- find all different names on the table
SELECT DISTINCT first_name
FROM customers;


-- Get a list of unique scores
SELECT DISTINCT score
FROM customers;


-- Count how many unique countries exist
SELECT COUNT(DISTINCT country) AS unique_countries
FROM customers;


-- Get unique countries where score > 500
SELECT DISTINCT country
FROM customers
WHERE score > 500;


-- Count unique scores
SELECT COUNT(DISTINCT score) AS total_unique_scores
FROM customers;


-- sum of unique score
SELECT SUM(DISTINCT score) AS sum_unique_scores
FROM customers;


-- Sum of unique scores for Egypt only
SELECT SUM(DISTINCT score) AS egypt_unique_scores
FROM customers
WHERE country = 'Egypt';


-- Average of unique scores
SELECT AVG(DISTINCT score) AS avg_unique_scores
FROM customers;


-- Find the minimum unique score
SELECT MIN(DISTINCT score) AS min_unique_score
FROM customers;
```

```sql
-- Count unique scores per country
SELECT
	country,
	COUNT(DISTINCT score) AS unique_scores
FROM customers
GROUP BY country;


-- Count unique scores per country
SELECT
	country,
	COUNT(DISTINCT score) AS unique_scores
FROM customers
GROUP BY country
ORDER BY unique_scores DESC;


-- Highest unique score per country
SELECT
	country,
	MAX(DISTINCT score) AS max_unique_score
FROM customers
GROUP BY country
ORDER BY max_unique_score DESC;
```

```sql
/*
	=====================
	== 03 WHERE Clause ==
	=====================
	- used to make coniditions on SQL Statement
	- syntax
		SELECT col1, col2, ..
		FROM table_name
		WHERE condition;
*/

-- select all customers from USA
SELECT *
FROM customers
WHERE country = 'USA';


-- select all customers that's name is -> John
SELECT *
FROM customers
WHERE first= 'John';


-- select the first 4 customers
SELECT *
FROM customers
WHERE id < 5;
```

```sql
/*
	=========================
	== 04 ORDER BY Keyword ==
	=========================
	- used to sort the result-set
	- you can sort your data
		1- ASC 'ascending' sort, and this is the default sort
		2- DESC 'descending' sort
	- syntax
		SELECT column1, column2, …
		FROM table_name
		WHERE condition
		ORDER BY column_name ASC|DESC;
*/

-- sort the records from the largest id to smallest id
SELECT *
FROM customers
ORDER BY id DESC;


-- get all records from the USA, and sort the result-set from the largest score to
smallest score
SELECT *
FROM customers
WHERE country = 'Egypt'
ORDER BY score DESC;


-- sort the charchters from a-z
SELECT
	id,
	first_name
FROM customers
ORDER BY first_name ASC;

/*
	--------------------
	-- Nested Sorting --
	--------------------
	- The ORDER BY clause sorts the results based on one or more columns.
	- The results will be grouped and sorted alphabetically by the country column
	- Inside each country group, customers will be sorted by score from highest
to lowest
*/
SELECT *
FROM customers
ORDER BY
	country ASC,
	score DESC;

-------------------------------------------------------------------------------
----------
-- Retrieve all customers and sort the results by the country and then by the
highest score --
-------------------------------------------------------------------------------
----------
SELECT *
```

```sql
FROM customers
ORDER BY
	country ASC,
	score DESC;
```

```sql
/*
	========================
	== 05 GROUP BY Clause ==
	========================
	- used to GROUP rows that have the same values in one or more columns and
perform aggregate functions on them
	- combines the rows with the same values
	- aggregate columns with another column [total score by country]
	- the non-aggregated columns that you are adding in the select must be
mentioned on the GROUP BY
	- aggregate functions like: COUNT(), SUM(), AVG(), MAX(), MIN()
	- syntax
		SELECT column_name, AGGREGATE_FUNCTION(column_name)
		FROM table_name
		GROUP BY column_name;

	--------------
	- NOTE THAT --
	--------------
		Columns in SELECT must be either:
		- Inside an aggregate function,
		- or Listed in GROUP BY
*/

use MyDatabase;

SELECT
	country,
	SUM(score) AS total_score
FROM customers
GROUP BY country;

-- Count customers per country
SELECT
	country,
	COUNT(*) AS total_customers
FROM customers
GROUP BY country;

-- Average score per country
SELECT
	country,
	AVG(score) AS avg_score
FROM customers
GROUP BY country;


-- Total score per country
SELECT
	country,
	SUM(score) AS total_score
FROM customers
GROUP BY country;
```

```sql
-- Using GROUP BY with ORDER B
SELECT
	country,
	COUNT(*) AS total_customers
FROM customers
GROUP BY country
ORDER BY total_customers DESC;


-- Multiple Columns in GROUP BY
SELECT
	country, score,
	COUNT(*) AS count_score
FROM customers
GROUP BY country, score
ORDER BY country;



-- Find the highest score in each country:
SELECT
	country,
	MAX(score) AS highest_score
FROM customers
GROUP BY country;

-- Find the total score and the total number of customers for each country, and sort
the result-set
SELECT
	country,
	SUM(score) AS total_score,
	COUNT(id) AS total_customers
FROM customers
GROUP BY country
ORDER BY total_score DESC;
```

```sql
/*
	======================
	== 06 HAVING Clause ==
	======================
	- used to filter groups of data after using the GROUP BY statement
	- always used with the GROUP BY statement
	- WHERE can't used with the aggregation functions
	- HAVING usually always used with the aggregation functions
	- it is similar to WHERE clause but
		WHERE filter rows before grouping
		HAVING filter rows after grouping
	- syntax
		SELECT column_name, AGGREGATE_FUNCTION(column_name)
		FROM table_name
		GROUP BY column_name
		HAVING condition;

*/
```

```sql
-- Get countries where total sales > 750
SELECT
	country,
	SUM(score) AS total_sales
FROM customers
GROUP BY country
HAVING SUM(score) > 750;


/*
	- filter the data using "WHERE", then filter the data using "HAVING"
	- WHERE -> GROUP BY -> HAVING -> SELECT -> ORDER BY
*/
SELECT
	country,
	SUM(score) AS total_score
FROM customers
WHERE score > 500		-- Filters individual rows first
GROUP BY country
HAVING SUM(score) > 750	-- Filters final groups
ORDER BY total_score DESC;


/*
	find the average score for each country
	considering only customers with a score not equal to 0
	and return only those countries with an average score greater than 430
*/
SELECT
	country,
	AVG(score) AS avg_score
FROM customers
WHERE score != 0
GROUP BY country
HAVING AVG(score) > 430;
```

```sql
/*
	====================
	== 07 TOP Keyword ==
	====================
	- used to limit the number of rows returned by a query
	- syntax
		SELECT TOP (number) column1, column2, ...
		FROM table_name
		WHERE condition
		ORDER BY column_name ASC|DESC;

*/

-- Get the First 3 Rows
SELECT TOP 3 *
FROM customers;


-- Get Top 3 Customers by Highest Scores
SELECT TOP 3
	first_name,
```

```sql
	score
FROM customers
ORDER BY score DESC;


-- Get Top 3 Lowest Scores
SELECT TOP 3
	first_name,
	score
FROM customers
ORDER BY score ASC;


-- Get Top 50% of Customers by Highest Scores
SELECT TOP 50 PERCENT
	first_name,
	score
FROM customers
ORDER BY score DESC;


-- Returns the first 3 unique countries alphabetically.
SELECT DISTINCT TOP 3 country
FROM customers
ORDER BY country ASC;


-- Find the top 2 countries with the highest total scores
SELECT TOP 2
	country,
	SUM(score) AS total_score
FROM customers
GROUP BY country
ORDER BY total_score DESC;


-- Show top 2 countries where total scores > 800
SELECT TOP 2
	country,
	SUM(score) AS total_score
FROM customers
GROUP BY country
HAVING SUM(score) > 800
ORDER BY total_score DESC;


-- Top 2 Countries with Highest Average Scores (TOP + GROUP BY + ORDER BY)
SELECT TOP 2
	country,
	AVG(score) AS avg_score
FROM customers
GROUP BY country
ORDER BY avg_score DESC;


-- Top 3 Countries Having More Than 1 Customer (TOP + GROUP BY + HAVING + ORDER BY)
SELECT TOP 3
	country,
```

```
      COUNT(*) AS total_customers
FROM customers
GROUP BY country
HAVING COUNT(*) > 1
ORDER BY total_customers DESC, country ASC;
```

Course material, course Git repository:

- **Reference**: https://www.datawithbaraa.com/sql-introduction/sql-ultimate-course/