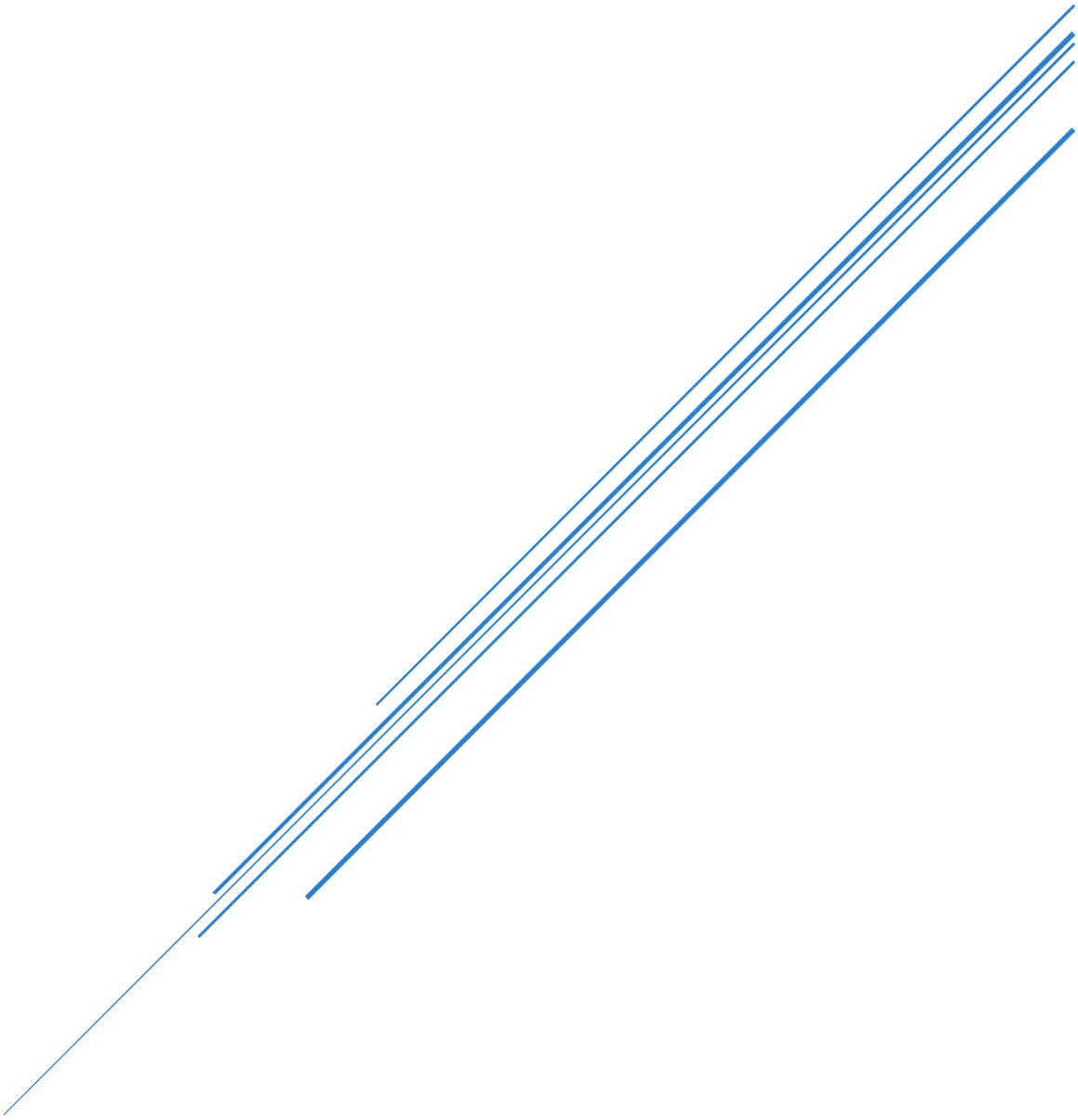
SQL
DataWithBaraa



SQL Part 2

```
/*
	==============
	== 27 UNION ==
	==============
	- used to combine the results of two or more SELECT queries into single
result set
	- it removes duplicate rows by default
	- all SELECT queries must have the same number of columns and compatible data
types
	- ORDER BY is allowed only once at the end of the query
	- data types of columns in each query must be compatible

	-- union returns distinct rows from both queries
	-- removes duplicate rows from the result

	------------
	-- syntax --
	------------
		SELECT column1, column2, ...
		FROM table1
		WHERE condition1

		UNION

		SELECT column1, column2, ...
		FROM table2
		WHERE condition2;

*/


USE SalesDB;


-- the number of columns in the first table must be equal to the number of columns
in the second table
-- the column names in the result set are determined by the column names specified
in the first query
SELECT
	FirstName AS first_name,
	LastName AS last_name
FROM Sales.Customers

UNION

SELECT
	FirstName,
	LastName
FROM Sales.Employees;
```

```sql
-- combine the data from employees and customers into one table
SELECT
	CustomerID AS id,
	FirstName AS first_name,
	LastName AS last_name
FROM Sales.Customers
UNION
SELECT
	EmployeeID,
	FirstName,
	LastName
FROM Sales.Employees;


/*
	---------------
	-- UNION ALL --
	---------------
	- is used to combine results from multiple SELECT queries and keep all rows,
including duplicates
	- it is faster than UNION because it does not remove duplicates

	------------
	-- syntax --
	------------
		SELECT column1, column2
		FROM table1

		UNION ALL

		SELECT column1, column2
		FROM table2;

*/
-- combine the data from employees and customers into one table
SELECT
	CustomerID AS id,
	FirstName AS first_name,
	LastName AS last_name
FROM Sales.Customers
UNION ALL
SELECT
	EmployeeID,
	FirstName,
	LastName
FROM Sales.Employees;
```

```sql
/*
	===============
	== 28 EXCEPT ==
	===============
	- is a SET operator that returns rows from the first query that do NOT exist
in the second query
	------------
	-- syntax --
	------------
		SELECT column1, column2
```

```
            FROM table1
            EXCEPT
            SELECT column1, column2
            FROM table2;
*/

USE SalesDB;

-- find the employees who are not customers at the same time
SELECT
      FirstName,
      LastName
FROM Sales.Customers
EXCEPT
SELECT
      FirstName,
      LastName
FROM Sales.Employees;

-- the order of queries in a EXCEPT does affect the result
SELECT
      FirstName,
      LastName
FROM Sales.Employees
EXCEPT
SELECT
      FirstName,
      LastName
FROM Sales.Customers;
```

```
/*
      ===============
      == INTERSECT ==
      ===============
      - is a SET operator that returns only rows that exists in BOTH queries
      ------------
      -- syntax --
      ------------
            SELECT col1, col2 FROM table1
            INTERSECT
            SELECT col1, col2 FROM table2;
*/

USE SalesDB;

-- find the employees that are also customers
SELECT
      FirstName,
      LastName
FROM Sales.Employees
INTERSECT
SELECT
      FirstName,
      LastName
FROM Sales.Customers;
```

➔ **Quick summary**

```
/*
      | Operator    | What it does                                |
      | ----------- | ------------------------------------------- |
      | `UNION`     | Combines results and removes duplicates     |
      | `UNION ALL` | Combines results and keeps duplicates       |
      | `INTERSECT` | Returns only common rows in both queries    |
      | `EXCEPT`    | Returns rows from first query not in second |
*/
```

```
/*
      ======================
      == String Functions ==
      ======================
      - CONCAT(string1, string2, ...): combines multiple strings into one
      - UPPER(text): convert all letters into upper case
      - LOWER(text): convert all letters into lower case
      - TRIM(text): remove the leading and trailling spaces from the left and right
side
      - REPLACE(text, old, new): replaces specific character with a new character
      - LEN(text): get the length of the characters
      - LEFT(string, length): extracts specific number of characters from the start
      - RIGHT(string, length):extracts specific number of characters from the end
      - SUBSTRING(value, start, final_length): extracts a part of string at a
specified position
*/

      USE MyDatabase;

      -- concatenate first name and country into one column
      SELECT
            first_name,
            country,
            CONCAT(first_name,'-', country) AS name_with_country
      FROM customers;

      -- you can write separator between columns
      SELECT
            first_name,
            country,
            CONCAT(first_name, ' - ', country) AS name_with_country
      FROM customers;


      -- convert the first name to upper case
      SELECT
            first_name,
            UPPER(first_name) AS upper_case
      FROM customers;

      -- convert the first name to lower case
      SELECT
            first_name,
            LOWER(first_name)
      FROM customers;
```

```sql
-- remove the spaces from the text
SELECT
      TRIM(' john ') AS trimed_data
FROM customers;

-- remove the special characters from the text
SELECT
      TRIM('-' FROM '--osama--') AS trimed_data
FROM customers;

-- replace word to another word
SELECT REPLACE('Hello World', 'World', 'SQL') AS Result;

-- replace the old phone syntax number to another syntax
SELECT
      '123-456-789' AS phone_number,
      REPLACE('123-456-789', '-', ' ') AS formatted_phone_number
FROM customers;

-- replace file extence from txt to csv
SELECT
      'file.txt' AS old_file,
      REPLACE('file.txt', '.txt', '.csv') AS new_file;


-- get the length of the text
SELECT
      first_name,
      LEN(first_name) AS text_length,
      LEN(TRIM(first_name)) AS text_after_trim
FROM customers;


-- extract the first 2 characters from the first name at the start
SELECT
      first_name,
      LEFT(first_name, 2) AS first_two_characters
FROM customers;


-- get the last character from the first name
SELECT
      first_name,
      RIGHT(first_name, 1) AS last_character
FROM customers;


-- get the first two characters after trim
SELECT
      first_name,
      LEFT(TRIM(first_name), 2) AS first_two_characters
FROM customers;
```

```sql
	-- get the first two characters before trim
	SELECT
		first_name,
		LEFT(first_name, 2) AS first_two_characters
	FROM customers;

	-- retrieve a list of customers'first names after removing the first
character
	SELECT
		first_name,
		SUBSTRING(first_name, 2, 4) AS new_string
	FROM customers;

	-- retrieve a list of customers's first name after removing the first
character
	SELECT
		first_name,
		SUBSTRING(first_name, 2, LEN(first_name))
	FROM customers;

-- make the first letter from the first_name capital, and the other letters small
SELECT
    UPPER(LEFT(first_name, 1)) + LOWER(SUBSTRING(first_name, 2, LEN(first_name))) AS
formatted_name
FROM customers;

SELECT
	first_name,
	LEN(first_name) AS 'length',
	LEN(TRIM(first_name)) AS 'length after trim',
	LEN(first_name) - LEN(TRIM(first_name)) AS 'flag'
FROM customers
WHERE LEN(first_name) != LEN(TRIM(first_name));
```

```sql
/*
	========================
	== 31 Number Functions ==
	========================
	- ROUND(): used to round a number to a specific number of decimal places
	- ABS(): returns the absolute value of a number, removing any negative sign
	- FLOOR() function rounds a number DOWN to the nearest integer.
	- SQRT() function is used to calculate the square root of a number.
	- CEILING() function rounds a number UP to the nearest integer.
	- POWER() function is used to raise a number to a specific power (exponent).
	- RAND(): Random number (0-1)
	- PI(): PI() function returns the mathematical constant (pi).
*/

USE MyDatabase;

SELECT
	12.4567 AS 'number',
	ROUND(12.4567, 2) AS result;
```

```sql
SELECT
	12.6 AS 'number',
	ROUND(12.6, 0) AS result;


-- get the positive number
SELECT
	-120 AS 'number',
	ABS(-120) AS 'positive number';


-- get the positive number
SELECT
	-120.52 AS 'number',
	ABS(-120.52) AS 'positive number';


-- round the number up, and down
SELECT
    CEILING(9.1) AS up,
    FLOOR(9.9) AS down;


SELECT POWER(10, 2);

-- get the the square root of 10
SELECT SQRT(10) as 'square root';


-- get the nearest number
SELECT CEILING(12.1);


-- (2 x 2 x 2)
SELECT POWER(2, 3);


-- (5 x 5)
SELECT POWER(5, 2);
```

```sql
/*
	==============================
	== 32 Date & Time Functions ==
	==============================
	- Date Format: 2025-08-20
	- Time Format: 18:55:45
	- DateTime: 2025-08-20  18:55:45

	- GETDATE(): returns the current system date and time of the SQL server

	| Format | Meaning      |
	| ------ | ------------ |
	| `ddd`  | Wed          |
	| `MMM`  | Jan          |
	| `yyyy` | 2025         |
	| `hh`   | 12-hour time |
	| `mm`   | minutes      |
```

```
	| `ss`	| seconds	|
	| `tt`	| AM / PM	|

*/

USE SalesDB;

SELECT
	OrderID,
	OrderDate,
	ShipDate,
	CreationTime
FROM Sales.Orders;


-- get current date and time
SELECT GETDATE() AS 'current DateTime';

/*
	-- insert the current date and time
	USE MyDatabase;
	INSERT INTO orders (order_date)
	VALUES (GETDATE());
*/

-- get the only date
SELECT CAST(GETDATE() AS DATE);


-- get the only time
SELECT CAST(GETDATE() AS TIME);


/*
	======================
	== Date Extraction ==
	======================
	DAY(): returns the day from a date
	MONTH(): returns the month from a date
	YEAR(): returns the year from a date
*/
USE SalesDB;
SELECT
	CreationTime,
	YEAR(CreationTime) AS 'Year',
	MONTH(CreationTime) AS 'Month',
	DAY(CreationTime) AS 'Day'
FROM Sales.Orders;


/*
	DATEPART(part, date): returns a specific part of a date as a number
	part: refers to the which part you need to extract like year, month, day, ...
	date: refers to the date or the column that contains the date
*/
SELECT
	OrderDate,
	DATEPART(YEAR, OrderDate) AS 'Year',
```

```sql
	DATEPART(MONTH, OrderDate) AS 'Month',
	DATEPART(DAY, OrderDate) AS 'Day',
	DATEPART(WEEK, OrderDate) AS 'Week'
FROM Sales.Orders;


-- also can use the abbreviation of part
SELECT
	CreationTime,
	DATEPART(YY, CreationTime) AS 'Year',
	DATEPART(MM, CreationTime) AS 'Month',
	DATEPART(DD, CreationTime) AS 'Day',
	DATEPART(HOUR, CreationTime) AS 'Hour',
	DATEPART(MINUTE, CreationTime) AS 'Minute',
	DATEPART(SECOND, CreationTime) AS 'Seconds',
	DATEPART(MILLISECOND, CreationTime) AS 'Milli Seconds',
	DATEPART(MICROSECOND, CreationTime) AS 'Micro Seconds'
FROM Sales.Orders;


-- Get the Quarter
SELECT
	CreationTime,
	DATEPART(QUARTER, CreationTime) AS 'Quarter'
FROM Sales.Orders;


-- get the weekday
SELECT
	CreationTime,
	DATEPART(WEEKDAY, CreationTime) AS 'Weekday'
FROM Sales.Orders;


/*
	DATENAME(part, date): returns the name of a specific part of a date
	the data type of the output is string
*/
SELECT
	CreationTime,
	DATENAME(MONTH, CreationTime) AS 'Month_DateName',
	DATENAME(WEEKDAY, CreationTime) AS 'Weekday_DateName'
FROM Sales.Orders;


/*
	DATETRUNC(part, date): truncate the date to the specific part, will keep the
part then reset the others values
	part: when define the part on DATRTRUNC() function, anything after that will
be reset
	if determine the part as MONTH, will reset the DAY as '01', because of there
is no DAY like 00
	if determine the part as YEAR, will reset the MONTH as '01', because of there
is no MONTH like 00
*/
```

```sql
SELECT
	CreationTime,
	DATETRUNC(YEAR, CreationTime) AS 'Hour'
FROM Sales.Orders;


/*
	count the number of orders based on the 'CreationTime'
	- will get one everywhere, because the level of details 'CreationTime' is
different
*/
SELECT
	CreationTime,
	COUNT(*) 'Number Of Orders'
FROM Sales.Orders
GROUP BY CreationTime;


-- aggregate the data based on the month level
SELECT
	DATETRUNC(MONTH, CreationTime) AS 'Month Level',
	COUNT(*) AS 'Number Of Orders'
FROM Sales.Orders
GROUP BY DATETRUNC(MONTH, CreationTime);


/*
	EOMONTH(date): return the last day of a month
	date: refers to the date or the column that contains the date
*/
SELECT
	CreationTime,
	EOMONTH(CreationTime) AS 'LastDay'
FROM Sales.Orders;


-- get the last day of the month
SELECT
	OrderDate,
	EOMONTH(OrderDate) AS 'LastDay'
FROM Sales.Orders;


-- get the start of month and end of month
SELECT
	CreationTime,
	DATETRUNC(MONTH, CreationTime) AS 'StartOfMonth',
	EOMONTH(CreationTime) AS 'EndOfMonth'
FROM Sales.Orders;
```

```sql
/*
	CAST(): used to convert the data types
*/
SELECT
	CreationTime,
	DATETRUNC(MONTH, CreationTime) AS 'StartOfDate',
	CAST(DATETRUNC(MONTH, CreationTime) AS DATE) AS 'StartOfData'
FROM Sales.Orders;

-- how many orders were placed each month
SELECT
	DATENAME(MONTH, OrderDate) AS OrderDate,
	COUNT(*) 'Number Of Orders'
FROM Sales.Orders
GROUP BY DATENAME(MONTH, OrderDate);


-- calculates how many orders were placed each month
SELECT
    YEAR(OrderDate) AS OrderYear,
    MONTH(OrderDate) AS OrderMonth,
    DATENAME(MONTH, OrderDate) AS MonthName,
    COUNT(*) AS NumberOfOrders
FROM Sales.Orders
GROUP BY
    YEAR(OrderDate),
    MONTH(OrderDate),
    DATENAME(MONTH, OrderDate)
ORDER BY
    OrderYear,
    OrderMonth;

-- show all orders that were placed during the month of February
SELECT *
FROM Sales.Orders
WHERE MONTH(OrderDate) = 2;


/*
	==============================
	== Date Formatting & Casting ==
	==============================
	- formatting: changing the format of a value from one to another (how the
data looks like)
	the formats can be as the following
		MONTH-DAY-YEAR
		DAY-MONTH-YEAR
	------------
	-- Syntax --
	------------
	FORMAT(value, format)

	- casting: convert the data type from one to another (string -> integer)
*/
```

```sql
SELECT
	OrderDate,

	-- change the arrange of the Date
	FORMAT(OrderDate, 'dd/MM/yy') AS 'Order Date',

	-- get the short name of the day
	FORMAT(OrderDate, 'ddd') AS 'Day',

	-- get the full name of the day
	FORMAT(OrderDate, 'dddd') AS 'Day',

	-- get the month as number
	FORMAT(OrderDate, 'MM') AS 'Month',

	-- get the short name of the month
	FORMAT(OrderDate, 'MMM') AS 'Month',

	-- get the full name of the month
	FORMAT(OrderDate, 'MMMM') AS 'Month',

	-- get the year from the date as number
	FORMAT(OrderDate, 'yy') AS 'Year',

	-- get the short name of the year
	FORMAT(OrderDate, 'yyy') AS 'Year'
FROM Sales.Orders;

/*s
	show the CreationTime using the following format:
	Day Wed Jan Q1 2025 12:34:56 PM
*/
SELECT
    CreationTime,
    CONCAT(
        'Day ',
        FORMAT(CreationTime, 'ddd MMM '),
        'Q', DATEPART(QUARTER, CreationTime), ' ',
        FORMAT(CreationTime, 'yyyy hh:mm:ss tt')
    ) AS [Creation Time]
FROM Sales.Orders;
```
