# SQL Tutorial on Single Row Functions

## Introduction:

Single row functions are powerful tools that operate on individual rows of a database table and return one result per row. They are especially useful for string manipulations, formatting, and data clean-up tasks. In this tutorial, we'll cover several essential SQL string functions with detailed examples and different parameter usages.

---

## 1. UPPER, LOWER, INITCAP

### UPPER Function:

The `UPPER` function converts all characters in a string to uppercase.

- **Syntax:**

  ```
  UPPER(string)
  ```

- **Example 1: Basic Usage**

  ```
  SELECT UPPER('hello world') AS result;
  ```

  **Output:**

  ```
  RESULT
  ------
  HELLO WORLD
  ```

- **Example 2: With Column Data** Suppose we have a table `employees` with a `name` column, and we want to retrieve all names in uppercase.

  ```
  SELECT UPPER(name) AS upper_name FROM employees;
  ```

---

### LOWER Function:

The `LOWER` function converts all characters in a string to lowercase.

- **Syntax:**

  ```
  LOWER(string)
  ```

- **Example 1: Basic Usage**

  ```
  SELECT LOWER('HELLO WORLD') AS result;
  ```

  **Output:**

```
RESULT
------
hello world
```

- **Example 2: Lowercase with Column Data** You can convert emails to lowercase in a table `users`.

  ```
  SELECT LOWER(email) AS normalized_email FROM users;
  ```

---

### INITCAP Function:

The `INITCAP` function capitalizes the first letter of each word in a string and converts the remaining letters to lowercase.

- **Syntax:**

  ```
  INITCAP(string)
  ```

- **Example 1: Basic Usage**

  ```
  SELECT INITCAP('this is a sql tutorial') AS result;
  ```

  **Output:**

  ```
  RESULT
  -----------------------
  This Is A Sql Tutorial
  ```

- **Example 2: Handling Mixed Case Input**

  ```
  SELECT INITCAP('wElCOme TO THE dAtaBAsE') AS result;
  ```

  **Output:**

  ```
  RESULT
  ------------------------
  Welcome To The Database
  ```

---

## 2. LEFT, RIGHT

### LEFT Function:

The `LEFT` function returns a specified number of characters from the left side of a string.

- **Syntax:**

  ```
  LEFT(string, number_of_characters)
  ```

- **Example 1: Basic Usage**

```
SELECT LEFT('Database', 4) AS result;
```

**Output:**

```
RESULT
------
Data
```

- **Example 2: Extracting Parts of Phone Numbers** Suppose we have a `contacts` table and we want to extract the country code from phone numbers.

```
SELECT LEFT(phone_number, 3) AS country_code FROM contacts;
```

### RIGHT Function:

The `RIGHT` function returns a specified number of characters from the right side of a string.

- **Syntax:**

```
RIGHT(string, number_of_characters)
```

- **Example 1: Basic Usage**

```
SELECT RIGHT('Database', 4) AS result;
```

**Output:**

```
RESULT
------
base
```

---

## 3. LPAD, RPAD, TRIM

### LPAD Function:

The `LPAD` function pads a string on the left side with a specified character until it reaches a desired length.

- **Syntax:**

```
LPAD(string, padded_length, pad_string)
```

- **Example 1: Padding with Zeroes**

```
SELECT LPAD('123', 6, '0') AS padded_value;
```

**Output:**

```
PADDED_VALUE
------------
000123
```

- **Example 2: Padding Employee IDs** For employee IDs with varying lengths, we can pad them with zeroes.

  ```
  SELECT LPAD(employee_id, 5, '0') AS formatted_id FROM employees;
  ```

**RPAD Function:**

The `RPAD` function pads a string on the right side with a specified character until it reaches a desired length.

- **Syntax:**

  ```
  RPAD(string, padded_length, pad_string)
  ```

- **Example 1: Padding with Dashes**

  ```
  SELECT RPAD('123', 6, '-') AS padded_value;
  ```

  **Output:**

  ```
  PADDED_VALUE
  ------------
  123---
  ```

- **Example 2: Padding Product Codes**

  ```
  SELECT RPAD(product_code, 10, '*') AS formatted_code FROM products;
  ```

**TRIM Function:**

The `TRIM` function removes leading, trailing, or both leading and trailing spaces (or other specified characters) from a string.

- **Syntax:**

  ```
  TRIM([LEADING | TRAILING | BOTH] trim_character FROM string)
  ```

- **Example 1: Removing Leading and Trailing Spaces**

  ```
  SELECT TRIM('   SQL Tutorial   ') AS trimmed_value;
  ```

  **Output:**

  ```
  TRIMMED_VALUE
  -------------
  SQL Tutorial
  ```

- **Example 2: Trimming Custom Characters**

  ```
  SELECT TRIM(BOTH 'x' FROM 'xxxSQLxxx') AS result;
  ```

  **Output:**

```
RESULT
------
SQL
```

---

## 4. LENGTH, REVERSE

### LENGTH Function:

The `LENGTH` function returns the number of characters in a string.

- **Syntax:**

  ```
  LENGTH(string)
  ```

- **Example 1: Finding String Length**

  ```
  SELECT LENGTH('Database Management') AS string_length;
  ```

  **Output:**

  ```
  STRING_LENGTH
  -------------
  21
  ```

- **Example 2: Checking Password Length**

  ```
  SELECT LENGTH(password) AS password_length FROM users WHERE
  LENGTH(password) < 8;
  ```

### REVERSE Function:

The `REVERSE` function reverses the characters in a string.

- **Syntax:**

  ```
  REVERSE(string)
  ```

- **Example 1: Reversing a String**

  ```
  SELECT REVERSE('SQL') AS reversed_string;
  ```

  **Output:**

  ```
  REVERSED_STRING
  ---------------
  LQS
  ```

- **Example 2: Reversing Email Addresses**

  ```
  SELECT REVERSE(email) AS reversed_email FROM users;
  ```

---

## 5. REPLACE

### REPLACE Function:

The `REPLACE` function replaces occurrences of a substring within a string with another substring.

- **Syntax:**

  ```
  REPLACE(string, search_string, replace_string)
  ```

- **Example 1: Replacing Words in a Sentence**

  ```
  SELECT REPLACE('Learn SQL with tutorials', 'tutorials', 'examples')
  AS replaced_value;
  ```

  **Output:**

  ```
  REPLACED_VALUE
  --------------
  Learn SQL with examples
  ```

- **Example 2: Replacing Product Codes** Suppose you have old product codes that need to be updated in a `products` table:

  ```
  SELECT REPLACE(product_code, 'OLD', 'NEW') AS updated_code FROM
  products;
  ```