7.Functions in MS-SQL Server.

7.1 Categories of Functions in SQL Server.

- ❑ Aggregate Functions.
- ❑ Date Functions.
- ❑ String Functions.
- ❑ System Functions.
- ❑ Window Functions.

1. **Aggregate Functions:** An aggregate function performs a calculation one or more values and returns a single value. The aggregate function is often used with the GROUP BY clause and HAVING clause of the SELECT statement.

Aggregate function	Description
AVG	The AVG() aggregate function calculates the average of non-NULL values in a set.
COUNT	The COUNT() aggregate function returns the number of rows in a group, including rows with NULL values.
MAX	The MAX() aggregate function returns the highest value (maximum) in a set of non-NULL values.
MIN	The MIN() aggregate function returns the lowest value (minimum) in a set of non-NULL values.
SUM	The SUM() aggregate function returns the summation of all non-NULL values a set.

AVG example

```
SELECT
    AVG(list_price) avg_product_price
FROM
    production.products;
```

COUNT example

```
SELECT
    COUNT(*) product_count
FROM
    production.products
WHERE
    list_price > 500;
```

MAX example

```
SELECT
    MAX(list_price) max_list_price
FROM
    production.products;
```

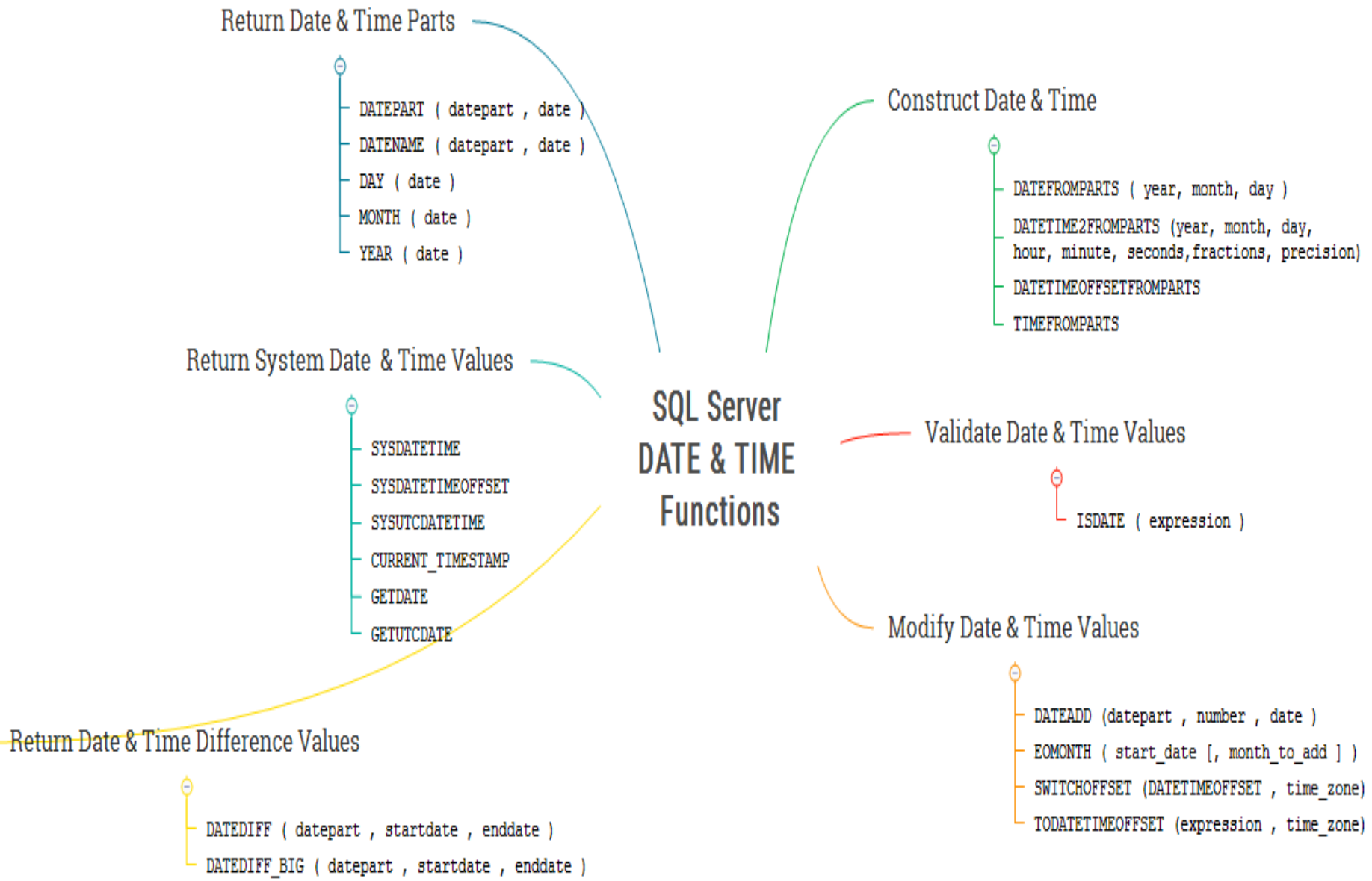
MIN example

```
SELECT
    MIN(list_price) min_list_price
FROM
    production.products;
```

SUM example

```
SELECT
    product_id,
    SUM(quantity) stock_count
FROM
    production.stocks
GROUP BY
    product_id
ORDER BY
    stock_count DESC;
```

2. Date Functions: This page lists the most commonly used SQL Server Date functions that allow you to handle date and time data effectively..



1.Returning the current date and time

Function	Description
CURRENT_TIMESTAMP	Returns the current system date and time without the time zone part.
GETUTCDATE	Returns a date part of a date as an integer number.
GETDATE	Returns the current system date and time of the operating system on which the SQL Server is running.
SYSDATETIME	Returns the current system date and time with more fractional seconds precision than the GETDATE() function.
SYSUTCDATETIME	Returns the current system date and time in UTC time
SYSDATETIMEOFFSET	Returns the current system date and time with the time zone.

SQL Server GETDATE() examples

```
1 SELECT
2    GETDATE() current_date_time;
```

2. Returning the date and time Parts

Function	Description
DATENAME	Returns a date part of a date as a character string
DATEPART	Returns a date part of a date as an integer number
DAY	Returns the day of a specified date as an integer
MONTH	Returns the month of a specified date as an integer
YEAR	Returns the year of the date as an integer.

SQL Server DATEPART() examples

```
DECLARE @d DATETIME = '2019-01-01 14:30:14';
SELECT
   DATEPART(year, @d) year,
   DATEPART(month, @d) month,
   DATEPART(day, @d) day,
   DATEPART(hour, @d) hour,
   DATEPART(minute, @d) minute,
   DATEPART(second, @d) second;
```

3. Returning a difference between two dates

Function	Return value
DATEDIFF	Returns a difference in date part between two dates.

```
DATEDIFF( date_part , start_date , end_date)
```

```
DECLARE
   @start_dt DATETIME2= '2019-12-31 23:59:59.9999999',
   @end_dt DATETIME2= '2020-01-01 00:00:00.0000000';

SELECT
   DATEDIFF(year, @start_dt, @end_dt) diff_in_year,
   DATEDIFF(month, @start_dt, @end_dt) diff_in_month,
   DATEDIFF(dayofyear, @start_dt, @end_dt) diff_in_dayofyear,
   DATEDIFF(day, @start_dt, @end_dt) diff_in_day
```

4. Modifying dates

Function	Description
DATEADD	Adds a value to a date part of a date and return the new date value.
EOMONTH	Returns the last day of the month containing the specified date, with an optional offset.
SWITCHOFFSET	Changes the time zone offset of a DATETIMEOFFSET value and preserves the UTC value.
TODATETIMEOFFSET	Transforms a DATETIME2 value into a DATETIMEOFFSET value.

SQL Server DATEADD() function examples

```
1 SELECT
2     DATEADD(day, 1, '2018-12-31 23:59:59') result;
```

5. Constructing date and time from their parts

Function	Description
DATEFROMPARTS	Return a DATE value from the year, month, and day.
DATETIME2FROMPARTS	Returns a DATETIME2 value from the date and time arguments
DATETIMEOFFSETFROMPARTS	Returns a DATETIMEOFFSET value from the date and time arguments
TIMEFROMPARTS	Returns a TIME value from the time parts with the precisions

SQL Server DATEFROMPARTS() function overview

```
DATEFROMPARTS(year, month, day)
```

```
SELECT
    DATEFROMPARTS(2020,12,31) a_date;
```

6.Validating date and time values

Function	Description
ISDATE	Check if a value is a valid date, time, or datetime value

Introduction to SQL Server ISDATE() function

```
ISDATE(expression)
```

```
SELECT
    ISDATE('2020-06-15') is_date
```

3. String Functions: The following SQL Server string functions process on an input string and return a string or numeric value:

function	Description
ASCII	Return the ASCII code value of a character
CHAR	Convert an ASCII value to a character
CONCAT	Join two or more strings into one string
DIFFERENCE	Compare the SOUNDEX() values of two strings
LEFT	Extract a given a number of characters from a character string starting from the left
LEN	Return a number of characters of a character string
LOWER	Convert a string to lowercase
REPLACE	Replace all occurrences of a substring, within a string, with another substring
REVERSE	Return the reverse order of a character string
SPACE	Returns a string of repeated spaces.
STR	Returns character data converted from numeric data.
SUBSTRING	Extract a substring within a string starting from a specified location with a specified length
TRIM	Return a new string from a specified string after removing all leading and trailing blanks
UPPER	Convert a string to uppercase

SQL Server ASCII() function overview

```
ASCII ( input_string )
```

```
SELECT
    ASCII('AB') A,
    ASCII('Z') Z;
```

Overview of SQL Server CONCAT() function

```
CONCAT ( input_string1, input_string2 [, input_stringN ] );
```

```
SELECT
    customer_id,
    first_name,
    last_name,
    CONCAT(first_name, ' ', last_name) full_name
FROM
    sales.customers
ORDER BY
    full_name;
```

SQL Serer REPLACE function overview

```
REPLACE(input_string, substring, new_substring);
```

```
SELECT
    REPLACE(
        'It is a good tea at the famous tea store.',
        'tea',
        'coffee'
    ) result;
```

SQL Server SUBSTRING() function overview

```
SUBSTRING(input_string, start, length);
```

```
SELECT
    SUBSTRING('SQL Server SUBSTRING', 5, 6) result;
```

7.2 User –Defined Functions (UDF):

The SQL Server user-defined functions help you simplify your development by encapsulating complex business logic and make them available for reuse in every query.

1.User-defined scalar functions – cover the user-defined scalar functions that allow you to encapsulate complex formula or business logic and reuse them in every query.

2.Table variables – learn how to use table variables as a return value of user-defined functions.

3.Table-valued functions – introduce you to inline table-valued function and multi-statement table-valued function to develop user-defined functions that return data of table types.

1. What are scalar functions:

SQL Server scalar function takes one or more parameters and returns a single value.

The scalar functions help you simplify your code. For example, you may have a complex calculation that appears in many queries. Instead of including the formula in every query, you can create a scalar function that encapsulates the formula and uses it in the queries.

Creating a scalar function

```
1 CREATE FUNCTION [schema_name.]function_name (parameter_list)
2 RETURNS data_type AS
3 BEGIN
4     statements
5     RETURN value
6 END
```

```sql
CREATE FUNCTION sales.udfNetSale(
    @quantity INT,
    @list_price DEC(10,2)

)
RETURNS DEC(10,2)
AS
BEGIN
    RETURN @quantity * @list_price;
END;
```

Calling a scalar function

```sql
SELECT
    sales.udfNetSale(10,100) net_sale;
```

```sql
SELECT
    order_id,
    SUM(sales.udfNetSale(quantity, list_price)) net_amount
FROM
    sales.order_items
GROUP BY
    order_id
ORDER BY
    net_amount DESC;
```

Modifying a scalar function

```sql
ALTER FUNCTION [schema_name.]function_name (parameter_list)
    RETURN data_type AS
    BEGIN
        statements
        RETURN value
    END
```

Removing a scalar function

```sql
DROP FUNCTION [schema_name.]function_name;
```

What are table variables:

Table variables are kind of variables that allow you to hold rows of data, which are similar to a temporary tables.

How to declare table variables

```
DECLARE @table_variable_name TABLE (
    column_list
);
```

Table variable example

```
DECLARE @product_table TABLE (
    product_name VARCHAR(MAX) NOT NULL,
    brand_id INT NOT NULL,
    list_price DEC(11,2) NOT NULL
)
```

Inserting data into the table variables

```
INSERT INTO @product_table
SELECT
    product_name,
    brand_id,
    list_price
FROM
    production.products
WHERE
    category_id = 1;
```

Querying data from the table variables

```
SELECT
    *
FROM
    @product_table;
```

```sql
DECLARE @product_table TABLE (
    product_name VARCHAR(MAX) NOT NULL,
    brand_id INT NOT NULL,
    list_price DEC(11,2) NOT NULL
);

INSERT INTO @product_table
SELECT
    product_name,
    brand_id,
    list_price
FROM
    production.products
WHERE
    category_id = 1;

SELECT
    *
FROM
    @product_table;
GO
```