CHAPTER 13: FUNCTIONS

2023

CHAPTER 13: FUNCTIONS

Contents:

1. Index
2. Create/drop index
3. Date functions

INDEX

Indexes are used to speed up the database's *search* operations:

By indexing the columns that are frequently used in search operations, the database can quickly locate the required data.

Indexes can improve performance but can also impact data modification operations:

While indexes can significantly improve query performance, they can also *slow down data modification* operations such as INSERT, UPDATE, and DELETE. This is because indexes must be updated whenever the underlying data changes.

CREATE INDEX

```
CREATE INDEX index_name
ON table_name (column1,  column2  ...);
```

```
CREATE INDEX ix_customers_name

ON sales.customers(last_name,  first_name);
```

DROP INDEX

SQL Server

```
DROP INDEX index_name ON table_name;
```

MySQL

```
ALTER TABLE table_name
DROP INDEX index_name;
```

DATE FUNCTIONS

Function	Description
CURDATE	Returns the current date.
DATEDIFF	Calculates the number of days between two DATE values.
DAY	Gets the day of the month of a specified date.
DATE_ADD	Adds a time value to date value.
DATE_SUB	Subtracts a time value from a date value.
DATE_FORMAT	Formats a date value based on a specified date format.
DAYNAME	Gets the name of a weekday for a specified date.
DAYOFWEEK	Returns the weekday index for a date.
EXTRACT	Extracts a part of a date.

Function	Description
LAST_DAY	Returns the last day of the month of a specified date
NOW	Returns the current date and time at which the statement executed.
MONTH	Returns an integer that represents a month of a specified date.
STR_TO_DATE	Converts a string into a date and time value based on a specified format.
SYSDATE	Returns the current date.
TIMEDIFF	Calculates the difference between two TIME or DATETIME values.
TIMESTAMPDIFF	Calculates the difference between two DATE or DATETIME values.
WEEK	Returns a week number of a date.
WEEKDAY	Returns a weekday index for a date.
YEAR	Return the year for a specified date

DATE FUNCTIONS

Date Functions	Desc	Return Value Data Type
DAY (date or datetime)	Returns the day of the week for a given date	Integer like 1 - 31
MONTH (date or datetime)	Returns the month of a given date	Integer like 1 - 12
YEAR (date or datetime)	Returns the year of a given date	Integer for year like 2021
DATEPART (date part, date or datetime)	Returns the date part specified in int format	Integer like 1 – 12 for month, 1 – 31 for day, or year like 2021
DATENAME (date part, date or datetime)	Returns the date part specified in character format	Character like April, May, '1', '2', '31', '2020', '2021'
EOMONTH (date [,months to add)	Returns the last do of the month with an optional parameter to add months (+ or -).	01/31/2021
DATEADD (date part, units, date or datetime)	Return date math results	datetime

DATE FUNCTIONS

DATEDIFF (date part, start date, end date)	Give the difference between 2 dates in units specified by date part	Integer of date part units
DATEDIFF_BIG	Give the difference between 2 dates in units specified by date part	Big Integer of date part units
CONVERT (date type, value [, style]	Used to convert date output to a specified mask	Typically, a character datatype is specified when converting dates.
FORMAT (value, format [, culture])	Used to convert date output to a specified mask	Returns a date formatted string based on the mask specified.
CAST (value as data type)	Used to convert different data types to a date or datetime data type.	Returns data in the data type specified.
ISDATE (potential date string)	Use to validate a date string	Returns 1 if the string is a valid date or 0 if not a valid date.

DATE FUNCTIONS: CURRENT_TIMESTAMP

The CURRENT_TIMESTAMP function returns the current timestamp of the operating system of the server on which the SQL Server Database runs.

The CURRENT_TIMESTAMP function takes no argument

```
SELECT
    CURRENT_TIMESTAMP AS current_date_time;
```

100 %

Results | Messages

	current_date_time
1	2021-12-22 16:20:54.357

The values in the [created_at] column took the timestamp returned by the CURRENT_TIMESTAMP function of when the values were created.

```
CREATE TABLE current_timestamp_tbl
(
    id         INT IDENTITY,
    msg        VARCHAR(255) NOT NULL,
    created_at DATETIME NOT NULL
                DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY(id)
);

INSERT INTO current_timestamp_tbl(msg)
VALUES('This is the first message.');

INSERT INTO current_timestamp_tbl(msg)
VALUES('current_timestamp demo');

select * from current_timestamp_tbl
```

100 %

Results | Messages

	id	msg	created_at
1	1	This is the first message.	2021-12-22 16:24:38.847
2	2	current_timestamp demo	2021-12-22 16:24:38.850

GETDATE()

The GETDATE() function returns the current system timestamp as a DATETIME value without the database time zone offset.

The DATETIME value is derived from the Operating System (OS) of the server on which the instance of SQL Server is running.

```
SELECT
    GETDATE() current_date_time;
```

100 %

Results | Messages

	current_date_time
1	2021-12-22 16:41:11.560

SYSDATETIME()

The SYSDATETIME() function returns a value of DATETIME2 that represents the current system date and time of the server on which the SQL Server instance is running.

```
SELECT
    CONVERT(DATE, SYSDATETIME()) as dates,
    CONVERT(TIME, SYSDATETIME()) times;
```

100 %

Results | Messages

	dates	times
1	2021-12-22	16:46:37.9227177

DATENAME()

The DATENAME() characteristic returns a string NVARCHAR type that represents a particular date component e.g. Seconds, nanoseconds, year, month and day of a particular date etc.

```sql
DECLARE @dt DATETIME2= '2020-10-02 10:20:30.1234567 +08:10';
SELECT 'year,yyy,yy' date_part,
    DATENAME(year, @dt) result
UNION
SELECT 'quarter, qq, q',
    DATENAME(quarter, @dt)
UNION
SELECT 'month, mm, m',
    DATENAME(month, @dt)
UNION
SELECT 'dayofyear, dy, y',
    DATENAME(dayofyear, @dt)
UNION
SELECT 'day, dd, d',
    DATENAME(day, @dt)
UNION
SELECT 'week, wk, ww',
    DATENAME(week, @dt)
UNION
SELECT 'weekday, dw, w',
    DATENAME(weekday, @dt)
UNION
SELECT 'hour, hh' date_part,
    DATENAME(hour, @dt)
UNION
SELECT 'minute, mi,n',
    DATENAME(minute, @dt)
UNION
SELECT 'second, ss, s',
    DATENAME(second, @dt)
UNION
SELECT 'millisecond, ms',
    DATENAME(millisecond, @dt)
UNION
SELECT 'microsecond, mcs',
    DATENAME(microsecond, @dt)
UNION
SELECT 'nanosecond, ns',
    DATENAME(nanosecond, @dt)
UNION
SELECT 'TZoffset, tz',
    DATENAME(tz, @dt)
UNION
SELECT 'ISO_WEEK, ISOWK, ISOWW',
    DATENAME(ISO_WEEK, @dt);
```

100 %

Results | Messages

	date_part	result
1	day, dd, d	2
2	dayofyear, dy, y	276
3	hour, hh	10
4	ISO_WEEK, ISOWK, ISOWW	40
5	microsecond, mcs	123456
6	millisecond, ms	123
7	minute, mi,n	20
8	month, mm, m	October
9	nanosecond, ns	123456700
10	quarter, qq, q	4
11	second, ss, s	30
12	TZoffset, tz	+00:00
13	week, wk, ww	40
14	weekday, dw, w	Friday
15	year,yyy,yy	2020

DATEPART ()

The DATEPART () function returns an integer that is part of a date. E.g. Day, month , year.

```sql
SELECT DATEPART(year, shipped_date) [year],
       DATEPART(quarter, shipped_date) [quarter],
       DATEPART(month, shipped_date) [month],
       DATEPART(day, shipped_date) [day],
       SUM(quantity * list_price) gross_sales
FROM sales.orders o
     INNER JOIN sales.order_items i ON i.order_id = o.order_id
WHERE shipped_date IS NOT NULL
GROUP BY DATEPART(year, shipped_date),
         DATEPART(quarter, shipped_date),
         DATEPART(month, shipped_date),
         DATEPART(day, shipped_date)
ORDER BY [year] DESC;
```

Results | Messages

	year	quarter	month	day	gross_sales
1	2018	1	1	1	3259.96
2	2018	1	1	2	11963.92
3	2018	1	1	3	20810.85
4	2018	1	1	5	8819.93
5	2018	1	1	6	5126.94
6	2018	1	1	7	17117.84
7	2018	1	1	8	11869.93
8	2018	1	1	9	2909.94
9	2018	1	1	10	21789.92
10	2018	1	1	12	20839.92
11	2018	1	1	13	20919.92
12	2018	1	1	14	26655.85

DATENAME and DATEPART are SQL Server functions that return the same information but in a different format.

The DATENAME function will return the character string-based date and time of a specified date whereas the DATEPART function will return an integer-based date and time of a specified date.

DAY()

The DAY() function returns an integer value which represents the day of a month (1-31) of a specified date.

```sql
SELECT
    DAY(shipped_date) [day],
    SUM(list_price * quantity) gross_sales
FROM
    sales.orders o
    INNER JOIN sales.order_items i ON i.order_id = o.order_id
WHERE
    shipped_date IS NOT NULL
    AND YEAR(shipped_date) = 2017
    AND MONTH(shipped_date) = 2
GROUP BY
    DAY(shipped_date)
ORDER BY [day];
```

Results | Messages

	day	gross_sales
1	1	1499.98
2	2	23966.90
3	4	9451.89
4	5	2929.96
5	6	19386.81
6	7	5319.96
7	8	27993.85
8	9	28766.78
9	10	3159.96
10	11	28721.80
11	12	7579.95
12	14	4481.94

MONTH()

The MONTH() function returns an integer value which represents the month of a specified date. The following shows the syntax of the MONTH() function:

```sql
SELECT MONTH(shipped_date) [month],
    SUM(list_price * quantity) gross_sales
FROM sales.orders o
    INNER JOIN sales.order_items i ON i.order_id = o.order_id
WHERE shipped_date IS NOT NULL
    AND YEAR(shipped_date) = 2017
GROUP BY MONTH(shipped_date)
ORDER BY [month];
```

100 %

Results | Messages

	month	gross_sales
1	1	291487.89
2	2	294607.77
3	3	368210.07
4	4	266578.50
5	5	282580.74
6	6	420611.08
7	7	221318.92
8	8	326123.26
9	9	317060.85
10	10	311766.19
11	11	332052.61
12	12	293492.88

YEAR()

The YEAR() function returns an integer value which represents the year of the specified date.

```
SELECT YEAR(shipped_date) [year],
       SUM(list_price * quantity) gross_sales
FROM sales.orders o
     INNER JOIN sales.order_items i ON i.order_id = o.order_id
WHERE shipped_date IS NOT NULL
GROUP BY YEAR(shipped_date)
order by [year];
```

100 %

Results | Messages

	year	gross_sales
1	2016	2649649.97
2	2017	3725890.76
3	2018	1062469.33

DATEDIFF ()

The DATEDIFF() function returns a value of integer indicating the difference between the start_date and end_date with the unit specified by date_part

DATEDIFF(*interval*, *date1*, *date2*)

Datediff(interval, old date, new date)

Datediff(y, 2000/01/01, getdate())

```sql
SELECT
    order_id,
    required_date,
    shipped_date,
    CASE
        WHEN DATEDIFF(day, required_date, shipped_date) < 0
        THEN 'Late'
        ELSE 'OnTime'
    END shipment
FROM
    sales.orders
WHERE
    shipped_date IS NOT NULL
ORDER BY
    required_date;
```

100 %

Results | Messages

	order_id	required_date	shipped_date	shipment
1	1	2016-01-03	2016-01-03	OnTime
2	2	2016-01-04	2016-01-03	Late
3	4	2016-01-04	2016-01-05	OnTime
4	3	2016-01-05	2016-01-03	Late
5	8	2016-01-05	2016-01-05	OnTime
6	5	2016-01-06	2016-01-06	OnTime
7	10	2016-01-06	2016-01-06	OnTime
8	6	2016-01-07	2016-01-05	Late
9	7	2016-01-07	2016-01-05	Late
10	9	2016-01-08	2016-01-08	OnTime
11	11	2016-01-08	2016-01-07	Late
12	12	2016-01-08	2016-01-09	OnTime

Interval → *Can be one of the following values:*

- year, yyyy, yy = Year
- quarter, qq, q = Quarter
- month, mm, m = month
- dayofyear = Day of the year
- day, dy, y = Day
- week, ww, wk = Week
- weekday, dw, w = Weekday
- hour, hh = hour
- minute, mi, n = Minute
- second, ss, s = Second
- millisecond, ms = Millisecond

DATEADD()

The DATEADD() function adds a number to a specified date part of an input date and returns the modified value.

If the number is a decimal or a floating-point number, the DATEADD () function truncates the decimal.

DATEADD(*interval*, *number*, *date*)

```sql
SELECT
    DATEADD(month, 4, '2021-12-22') AS result;
```

100 %

Results | Messages

	result
1	2022-04-22 00:00:00.000

```sql
SELECT
    order_id,
    customer_id,
    order_date,
    DATEADD(day, 2, order_date) estimated_shipped_date
FROM
    sales.orders
WHERE
    shipped_date IS NULL
ORDER BY
    estimated_shipped_date DESC;
```

100 %

Results | Messages

	order_id	customer_id	order_date	estimated_shipped_date
1	1615	136	2018-12-28	2018-12-30
2	1614	135	2018-11-28	2018-11-30
3	1613	1	2018-11-18	2018-11-20
4	1612	3	2018-10-21	2018-10-23
5	1611	6	2018-09-06	2018-09-08
6	1610	15	2018-08-25	2018-08-27
7	1609	10	2018-08-23	2018-08-25
8	1608	53	2018-07-12	2018-07-14
9	1607	33	2018-07-11	2018-07-13
10	1606	119	2018-07-10	2018-07-12
11	1605	123	2018-07-01	2018-07-03
12	1604	7	2018-06-17	2018-06-19
13	1602	55	2018-04-30	2018-05-02
14	1603	74	2018-04-30	2018-05-02
15	1598	239	2018-04-29	2018-05-01

Query executed successfully. | PLJHBTRALPT017\SQLEXPRESS (... | PLATINUMLIFE\Masego.Di... | master

EOMONTH ()

The EOMONTH () function returns the last day of the specified date using an optional offset.

The EOMONTH () function accepts two arguments.

1. The start_date is a date expression that is evaluated for a date. The EOMONTH () function returns the last day of the date.

2. The offset is an integer that indicates the number of months to add to start_date.

If adding offset and start_date results in an invalid date, the EOMONTH () function returns an error.

The syntax: EOMONTH(start_date, [offset]);

EOMONTH () EXAMPLES

```
SELECT
    EOMONTH('2020-02-09') end_of_month_feb2020;
```

100 %

Results | Messages

	end_of_month_feb2020
1	2020-02-29

2. EOMONTH() function with an offset of 2 months:

```
SELECT
    EOMONTH('2019-02-15', 2) eomonth_next_2_months;
```

100 %

Results | Messages

	eomonth_next_2_months
1	2019-04-30

3. To get the number of days in the specified month of the date:

```
SELECT
    DAY(EOMONTH('2020-02-09')) days;
```

100 %

Results | Messages

	days
1	29

DATEFROMPARTS()

The DATEFROMPARTS() function returns a DATE value that maps to a year, month and day values.

The syntax: DATEFROMPARTS(year, month, day)

```
SELECT
    DATEFROMPARTS(2021,12,28) a_date;
```

100 %

Results | Messages

	a_date
1	2021-12-28

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

100 %

Results | Messages

	a_date
1	NULL

TIMEFROMPARTS()

The TIMEFROMPARTS() function returns a fully initialized time value.

The syntax: TIMEFROMPARTS (hour minute seconds fractions precision)

```
SELECT
    TIMEFROMPARTS(10, 36, 15, 5, 2) Time;
```

100 %

Results | Messages

	Time
1	10:36:15.05

In this example when the fractions has a value of 5 and precision has a value of 2 then the value of fractions represents 5/100 or 0.05 of a second.

ISDATE()

The ISDATE() function accepts an argument and returns 1 if that argument is a valid DATE, TIME or DATETIME value; otherwise it returns 0.

The syntax: ISDATE(expression)

```
SELECT
    ISDATE('2020-12-05 11:20:30') is date
```

100 %

Results | Messages

	is_date
1	1

Practice DATE FUNCTIONS

1. How long have each employee been working for the company?
2. On which date would we celebrate the “30 year anniversary” of each employee
3. How many employees have been hired in each year?
4. If they all get their first salary the last day of the month after they’ve been hired, which day would they have received their first pay-check?

Practice DATE FUNCTIONS

1. How many days did it take for customer 103 to make payment for his product with the order number 10123
2. Determine the customers who received their orders earlier than the required date. (*HINT: The customers are considered to have received the product earlier if the product was shipped 7 days before the required date and the status must be shipped*.)
3. Determine all the customers who received their product later than the required date.
4. Bonus question: Now write a query to categorize the payment received as "early" or "late". The payment is made early if the number of days is before 7 days the order date otherwise the payment is.

L&D SHOP