

# SQL Server

Introduction

Ahmed Fekry

[ahmedfikry78@gmail.com](mailto:ahmedfikry78@gmail.com)


+201120590421

# OFFSET-FETCH Clause

**Description:** Used to skip a specific number of rows and return a set number of rows after that.

- Syntax:

sql


 Copy code

```
SELECT columns
FROM table
ORDER BY column
OFFSET number_of_rows_to_skip ROWS
FETCH NEXT number_of_rows_to_return ROWS ONLY;
```

# OFFSET-FETCH Clause

- Example:

sql

 Copy code

```
SELECT order_id, order_date, customer_id
FROM sales.orders
ORDER BY order_date DESC
OFFSET 10 ROWS
FETCH NEXT 5 ROWS ONLY;
```

# SQL Conversion Functions

## Overview

Conversion functions in SQL are used to explicitly convert data from one data type to another. This is essential when dealing with data that needs to be formatted or manipulated in a specific way.


# SQL Conversion Functions

## 1. CAST()

- **Description:** Converts an expression from one data type to another.

- Syntax:

sql

 Copy code


```
CAST(expression AS data_type)
```

# SQL Conversion Functions

## 2. CONVERT( )

- **Description:** Converts an expression from one data type to another with an optional style parameter for date formatting.
- Syntax:

sql

 Copy code


```
CONVERT(data_type(length), expression, style)
```

# SQL Conversion Functions

## 3. **PARSE()**

- **Description:** Converts an expression to the specified data type and can be used with a specific culture for formatting.
- Syntax:

sql

 Copy code

```
PARSE(expression AS data_type USING culture)
```

# SQL Conversion Functions

## Summary

- **CAST** is useful for standard type conversions and is more portable across different SQL databases.
- **CONVERT** is specific to SQL Server and offers additional formatting options, especially for date and time types.
- **PARSE** is used for converting strings to date/time and numeric types with respect to specific cultural formats but is generally slower due to its complexity.



# Writing Self-Contained Subqueries

- Subqueries are nested queries: queries within queries
- Results of inner query passed to outer query
- Inner query acts like an expression from perspective of outer query
- Subqueries can be self-contained or correlated
- Self-contained subqueries have no dependency on outer query
- Correlated subqueries depend on values from outer query
- Subqueries can be scalar, multi-valued, or table-valued

# Writing Self-Contained Subqueries

Outer Query:

```
SELECT orderid, productid,  
unitprice, qty  
FROM Sales.OrderDetails  
WHERE orderid = ( )
```



Inner Query:

```
SELECT MAX(orderid)  
AS lastorder  
FROM Sales.Orders
```

Outer Query:

```
SELECT orderid, empid,  
orderdate  
FROM Sales.Orders AS O1  
WHERE  
orderdate = ( )
```



Inner Query:

```
SELECT MAX(orderdate)  
FROM Sales.Orders AS O2  
WHERE O2.empid =  
O1.empid
```



# Writing Self-Contained Subqueries

## Writing Scalar Subqueries

- 1- Scalar subquery returns single value to outer query
- 2- If inner query returns an empty set, result is converted to NULL

```
SELECT orderid, productid, unitprice, qty
FROM Sales.OrderDetails
WHERE orderid =
    (SELECT MAX(orderid) AS lastorder
     FROM Sales.Orders);
```

# Writing Self-Contained Subqueries

## Writing Multi-Valued Subqueries

1- Multi-valued subquery returns multiple values as a single column set to the outer query

2- If any value in the subquery result matches IN predicate expression, the predicate returns TRUE

```
SELECT custid, orderid
FROM Sales.orders
WHERE custid IN (
    SELECT custid
    FROM Sales.Customers
    WHERE country = N'Mexico');
```

# Writing Correlated Subqueries

- Correlated subqueries refer to elements of tables used in outer query
- Dependent on outer query, cannot be executed separately
- Behaves as if inner query is executed once per outer row
- May return scalar value or multiple values

```
SELECT orderid, empid, orderdate
FROM Sales.Orders AS O1
WHERE orderdate =
    (SELECT MAX(orderdate)
     FROM Sales.Orders AS O2
     WHERE O2.empid = O1.empid)
ORDER BY empid, orderdate;
```

# Writing Subqueries using *Exists*

When a subquery is used with the keyword EXISTS, it functions as an existence test

True or false only—no rows passed back to outer query

EXISTS evaluates to TRUE or FALSE (not UNKNOWN)

If any rows are returned by the subquery, EXISTS returns TRUE

If no rows are returned, EXISTS returns FALSE

Syntax:

```
WHERE [NOT] EXISTS (subquery)
```

# CTE in SQL Server

CTE stands for common table expression. A CTE allows you to define a temporary named result set that available temporarily in the execution scope of a statement such as **SELECT**, **INSERT**, **UPDATE**, **DELETE**, or **MERGE**.

```
WITH expression_name[(column_name [, ...])]  
AS  
    (CTE_definition)  
SQL_statement;
```

# CTE in SQL Server

## Advantages of CTEs

1. **Improved readability:** Break complex queries into simpler, more understandable parts.
2. **Recursive queries:** Solve problems involving hierarchical data or recursive relationships.
3. **Reusability:** Refer to the same result set multiple times within a query.




# CTE in SQL Server

## Example 1: Simple CTE

Find products with a list price higher than the average list price:

sql

 Copy code

```
WITH AveragePrice AS
(
    SELECT AVG(list_price) AS avg_price
    FROM production.products
)
SELECT product_id, product_name, list_price
FROM production.products
WHERE list_price > (SELECT avg_price FROM AveragePrice);
```

# SQL Set Operations: UNION, UNION ALL, INTERSECT, and EXCEPT

## Overview

Set operations in SQL allow you to combine the results of two or more queries into a single result set. The most common set operations are UNION, UNION ALL, INTERSECT, and EXCEPT.

# SQL Set Operations: UNION, UNION ALL, INTERSECT, and EXCEPT

## Types of Set Operations


1. **UNION**: Combines the result sets of two or more queries and removes duplicate rows.
2. **UNION ALL**: Combines the result sets of two or more queries, including duplicates.
3. **INTERSECT**: Returns only the rows that are present in both result sets.
4. **EXCEPT**: Returns the rows from the first result set that are not present in the second result set.

# SQL Set Operations: UNION, UNION ALL, INTERSECT, and EXCEPT

## Example 1: Using UNION

Combine customers and staffs' names into a single list:

sql

 Copy code


```
SELECT first_name, last_name
FROM sales.customers
UNION
SELECT first_name, last_name
FROM sales.staffs;
```

# SQL Set Operations: UNION, UNION ALL, INTERSECT, and EXCEPT

## Example 2: Using UNION ALL

Combine customers and staffs' names into a single list, including duplicates:

sql

 Copy code


```
SELECT first_name, last_name
FROM sales.customers
UNION ALL
SELECT first_name, last_name
FROM sales.staffs;
```

# SQL Set Operations: UNION, UNION ALL, INTERSECT, and EXCEPT

## Example 3: Using INTERSECT

Find products available in both 2020 and 2021:

sql

 Copy code


```
SELECT product_id, product_name
FROM production.products
WHERE model_year = 2020
INTERSECT
SELECT product_id, product_name
FROM production.products
WHERE model_year = 2021;
```

# SQL Set Operations: UNION, UNION ALL, INTERSECT, and EXCEPT

## Example 4: Using EXCEPT

Find products available in 2020 but not in 2021:

sql

 Copy code

```
SELECT product_id, product_name
FROM production.products
WHERE model_year = 2020
EXCEPT
SELECT product_id, product_name
FROM production.products
WHERE model_year = 2021;
```

# SQL Server Stored Procedures

A stored procedure is a prepared SQL code that you can save and reuse. It can accept parameters, perform operations, and return results. Stored procedures improve performance, security, and code management in SQL Server.



# SQL Server Stored Procedures

## 1. **Benefits:**

- **Performance:** Precompiled and cached execution plans improve performance.
- **Security:** Restricts direct access to tables and allows permissions management.
- **Reusability:** Write once, use multiple times.
- **Maintainability:** Simplifies code maintenance and management.