SQL Server

Arctech Info Private Limited

SELECT TOP

1) Using TOP with a constant value

```
SELECT TOP 10
    product_name,
    list_price
FROM
    production.products
ORDER BY
    list_price DESC;
```

2) Using TOP to return a percentage of rows

```
SELECT TOP 1 PERCENT
    product_name,
    list_price
FROM
    production.products
ORDER BY
    list_price DESC;
```

SELECT TOP

3) Using TOP WITH TIES to include rows that match the values in the last row

```
SELECT TOP 3 WITH TIES
    product_name,
    list_price
FROM
    production.products
ORDER BY
    list_price DESC;
```

Filtering data

- Distinct
  - select distinct values in one or more columns of a table.
- Where
  - filter rows in the output of a query based on one or more conditions.
- AND
  - combine two Boolean expressions and return true if all expressions are true.
- OR
  - combine two Boolean expressions and return true if either of conditions is true.
- IN
  - check whether a value matches any value in a list or a subquery.
- Between
  - test if a value is between a range of values.
- Like
  - check if a character string matches a specified pattern.
- Column & table aliases
  - show you how to use column aliases to change the heading of the query output and table alias to improve the readability of a query.

IN

- The IN operator is a logical operator that allows you to test whether a specified value matches any value in a list.

```
SELECT   product_name,  list_price
FROM   production.products
WHERE   list_price IN (89.99, 109.99, 159.99)
ORDER BY  list_price;
```

```
SELECT    product_name,    list_price
FROM    production.products
WHERE product_id IN (
        SELECT product_id
        FROM production.stocks
        WHERE  store_id = 1 AND quantity >= 30
    )
ORDER BY product_name;
```

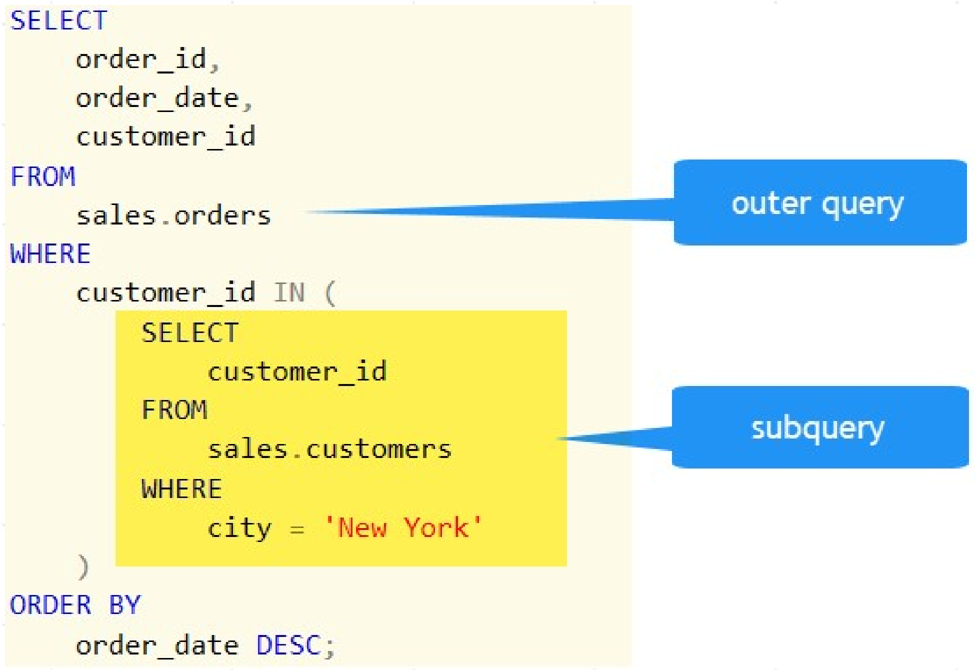
BETWEEN

- The BETWEEN operator is a logical operator that allows you to specify a range to test.
- Syntax –
- column | expression BETWEEN start_expression AND end_expression

```
SELECT product_id, product_name,
    list_price
FROM production.products
WHERE
    list_price BETWEEN 149.99 AND 199.99
ORDER BY
    list_price;
```

Subquery

- Subquery –
  - A subquery is a query nested inside another statement .



```
SELECT
    order_id,
    order_date,
    customer_id
FROM
    sales.orders
WHERE
    customer_id IN (
        SELECT
            customer_id
        FROM
            sales.customers
        WHERE
            city = 'New York'
    )
ORDER BY
    order_date DESC;
```

Nesting subquery

- A subquery can be nested within another subquery. SQL Server supports up to 32 levels of nesting.
- Example –
- Get all the product whoes list_price is more than the average price of product of brand 'Strider' or 'Trek'
- 1. get the brand id of bran names – 'Strider' or 'Trek'
- 2. get the average of list price whoes brand id falls in the above list
- 3. get all the products whoes list price is more than the average calculated above

```
SELECT   product_name,    list_price
FROM   production.products
WHERE   list_price > (
      SELECT  AVG (list_price) FROM production.products
      WHERE brand_id IN (
            SELECT brand_id FROM production.brands
            WHERE  brand_name = 'Strider'
            OR brand_name = 'Trek'
         )
   )
ORDER BY
   list_price;
```

EXISTS

- The EXISTS operator is a logical operator that allows you to check whether a subquery returns any row. The EXISTS operator returns TRUE if the subquery returns one or more rows.

- The following shows the syntax of the SQL Server EXISTS operator:
- EXISTS (subquery)

```
SELECT    customer_id,    first_name,     last_name
FROM    sales.customers c
WHERE     EXISTS (
        SELECT  COUNT (*)
        FROM  sales.orders o
        WHERE  customer_id = c.customer_id
        GROUP BY   customer_id
        HAVING  COUNT (*) > 2
    )
ORDER BY
    first_name,
    last_name;
```

SET OPERATOR – UNION and UNION ALL

- SQL Server UNION is one of the set operations that allow you to combine results of two SELECT statements into a single result set which includes all the rows that belong to the SELECT statements in the union.
- By default, the UNION operator removes all duplicate rows from the result sets. However, if you want to retain the duplicate rows, you need to specify the ALL keyword

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

SET OPERATOR – INTERSECT

- The SQL Server INTERSECT combines result sets of two or more queries and returns distinct rows that are output by both queries
- Both queries must have the same number and order of columns.
- The data type of the corresponding columns must be the same or compatible.

```
SELECT   city
FROM sales.customers
INTERSECT
SELECT  city
FROM  sales.stores
ORDER BY   city;
```

SET OPERATOR - EXCEPT

- The SQL Server EXCEPT compares the result sets of two queries and returns the distinct rows from the first query that are not output by the second query. In other words, the EXCEPT subtracts the result set of a query from another.
- The number and order of columns must be the same in both queries.
- The data types of the corresponding columns must be the same or compatible.

```
SELECT    product_id
FROM   production.products
EXCEPT
SELECT   product_id
FROM   sales.order_items;
```

DML – UPDATE

```
UPDATE table_name
SET c1 = v1, c2 = v2, ... cn = vn
[WHERE condition]
```

DML – UPDATE WITH JOIN

```
UPDATE
    sales.commissions
SET
    sales.commissions.commission =
        c.base_amount * t.percentage
FROM
    sales.commissions c
    INNER JOIN sales.targets t
        ON c.target_id = t.target_id;
```

- Create query for the following –
- Add a column StatusDetails in order_tems
- Update this column with help of data from oders
  - StatusDetails should be a string containing order date and order status from orders

DELETE

```
DELETE
FROM
    production.product_history
WHERE
    model_year = 2017;
```

DDL - ALTER TABLE ADD Column

```
ALTER TABLE table_name
ADD column_name data_type
column_constraint;
```

- First, specify the name of the table in which you want to add the new column.
- Second, specify the name of the column, its data type, and constraint if applicable.

```
CREATE TABLE sales.quotations (
    quotation_no INT IDENTITY PRIMARY KEY,
    valid_from DATE NOT NULL,
    valid_to DATE NOT NULL
);
```

```
ALTER TABLE sales.quotations
ADD description VARCHAR (255) NOT
NULL;
```

DDL – COMPUTED COLUMNS

```
SELECT
    person_id,
    first_name + ' ' + last_name AS full_name,
    dob
FROM
    persons
ORDER BY
    full_name;
```

- Instead of querying the table everytime with the expression to get the full name, SQL Server provides us with a feature called computed columns that allows you to add a new column to a table with the value derived from the values of other columns in the same table.

```
ALTER TABLE persons
ADD full_name AS (first_name + ' ' + last_name);
```

DDL – DROP TABLE

- Syntax
  - DROP TABLE [IF EXISTS] [database_name.][schema_name.]table_name;
- When SQL Server drops a table, it also deletes all data, triggers, constraints, permissions of that table. Moreover, SQL Server does not explicitly drop the views and stored procedures that reference the dropped table. Therefore, to explicitly drop these dependent objects, you must use the DROP VIEW and DROP PROCEDURE statement.

TRUNCATE TABLE

- Delete all rows from table
- Syntax-
  - TRUNCATE TABLE [database_name.][schema_name.]table_name;

- Truncate vs Delete
- Truncate removes all records and doesn't fire triggers. Truncate is faster compared to delete as it makes less use of the transaction log. Truncate is not possible when a table is referenced by a Foreign Key or tables are used in replication or with indexed views.