SUB QUERY

```sql
-- A subquery is a query that is written inside another query. Subqueries are also called Inner queries or Nested queries. They provide data
-- to the main query, which is called the Parent query or Outer query. A subquery can be inside the SELECT, FROM, JOIN, or WHERE clause or
-- inside another subquery.

-- Subquery in SELECT statement, a.k.a. Scalar Subquery
-- Generates only one column in the SELECT statement that contains a Scalar (single) value across all the rows of the main query

WITH data AS (
  SELECT 'iPhone 15 Pro 128GB' product, 999 price UNION SELECT 'iPhone 15 Pro 256GB' product, 1099 price UNION
  SELECT 'iPhone 15 Pro 512GB' product, 1299 price UNION SELECT 'iPhone 15 Pro 1TB'   product, 1499 price
)
SELECT
  product, price, (SELECT MAX(price) FROM data) max_price,        -- We are generating a scalar value of max_price using a subquery
  (SELECT MAX(price) FROM data) - price AS price_diff             -- This scalar value can be used for an arithmetic operation
FROM data;                                                        -- No GROUP BY clause is required to get the MAX(price) aggregation

-- | product             | price | max_price | price_diff |
-- |---------------------|-------|-----------|------------|
-- | iPhone 15 Pro 128GB | 999   | 1499      | 500        |
-- | iPhone 15 Pro 256GB | 1099  | 1499      | 400        |
-- | iPhone 15 Pro 512GB | 1299  | 1499      | 200        |
-- | iPhone 15 Pro 1TB   | 1499  | 1499      | 0          |

-- Subquery in WHERE clause
WITH data AS (
  SELECT 'John' name, 10 age UNION SELECT 'Sam', 15 UNION SELECT 'Tim', 20
)
SELECT name
FROM data
WHERE name = (SELECT name                                         -- Subquery in WHERE clause that returns a single row
              FROM data
              WHERE age = 10);

WITH data AS (
  SELECT 'John' name, 10 age UNION SELECT 'Sam', 15 UNION SELECT 'Tim', 20
)
SELECT name
FROM data
WHERE name IN (SELECT name                                        -- If the subqyery is returning more than a single row then the
              FROM data                                           -- IN operator is required
              WHERE age < 20);


--
```

```sql
    -- List - WHERE id IN (12, 25, 36)
    -- A Table
-- Used for comparing groups to summarized values, reshaping data, combining data that cannot be joined

-- Types of Subquery
-- Subquery Types
  -- In place of an Expression
  -- With IN or NOT IN
  -- In UPDATE, DELETE, or INSERT statement
  -- With EXISTS or NOT EXISTS
  -- With ANY or ALL
  -- In the FROM clause (Derived Table)
  -- Correlated Subquery


DROP TABLE #products
CREATE TABLE #products (product VARCHAR(10), cost INT)
INSERT INTO #products VALUES ('ABC', 10), ('ABC', 20), ('ABC', 30),
('XYZ', 15), ('XYZ', 25), ('XYZ', 35)
SELECT * FROM #products;

-- SUBQUERY in SELECT
-- Subquery in SELECT requires a Scalar (Single) value to be returned
-- as it needs to apply to all rows of the query
SELECT *, (SELECT MAX(cost)
           FROM #products) AS max_cost
```

```
FROM #products;

-- SUBQUERY in WHERE
-- To find products that cost more than average cost
SELECT product, cost, (SELECT AVG(cost) FROM #products) AS avg_cost
FROM #products
WHERE cost > (SELECT AVG(cost) FROM #products);

-- If expecting multiple values use the IN operator in subquery
-- Make sure to use DISTINCT to show only relevant products
SELECT DISTINCT product
FROM #products
WHERE product IN (SELECT DISTINCT product
                  FROM #products
                  WHERE cost > 20);

-- NESTED SUBQUERY
SELECT DISTINCT product
FROM #products
WHERE product IN (SELECT product
                  FROM #products
                  GROUP BY product
                  HAVING MAX(cost) > (SELECT AVG(cost)
                                      FROM #products));
```

DERIVED TABLE

```
-- Derived table is an expression that generats a table within the scope of a query FROM clause.
-- SELECT ... FROM (subquery) [AS] tbl_name ...

SELECT
    AVG(t.max_cost * 1.0) as avg_max_cost
FROM (SELECT product, MAX(cost) as max_cost
      FROM #products
      GROUP BY product) t
```

CORRELATED SUBQUERY

```
-- For Correlated Subquery, there is no JOIN condition, but join is performed in the WHERE clause
-- To find products that cost more than average cost of the same product

SELECT product, cost, (SELECT AVG(cost)
                       FROM #products b
                       WHERE a.product = b.product) as prod_avg_cost
FROM #products a
WHERE cost > (SELECT AVG(cost)
              FROM #products b
              WHERE a.product = b.product)
```

SUBQUERY

```
--- A Subquery returns a variety of information
    -- Scalar Values - 3.14159, -2, 0.001
    -- List - WHERE id IN (12, 25, 36)
    -- A Table
-- Used for comparing groups to summarized values, reshaping data, combining data that cannot be joined

-- Types of Subquery
-- Subquery Types
  -- In place of an Expression
  -- With IN or NOT IN
  -- In UPDATE, DELETE, or INSERT statement
  -- With EXISTS or NOT EXISTS
  -- With ANY or ALL
  -- In the FROM clause (Derived Table)
  -- Correlated Subquery

DROP TABLE #products
CREATE TABLE #products (product VARCHAR(10), cost INT)
INSERT INTO #products VALUES ('ABC', 10), ('ABC', 20), ('ABC', 30),
```

```
('XYZ', 15), ('XYZ', 25), ('XYZ', 35)
SELECT * FROM #products;

-- SUBQUERY in SELECT
-- Subquery in SELECT requires a Scalar (Single) value to be returned
-- as it needs to apply to all rows of the query
SELECT *, (SELECT MAX(cost)
           FROM #products) AS max_cost
FROM #products;

-- SUBQUERY in WHERE
-- To find products that cost more than average cost
SELECT product, cost, (SELECT AVG(cost) FROM #products) AS avg_cost
FROM #products
WHERE cost > (SELECT AVG(cost) FROM #products);

-- If expecting multiple values use the IN operator in subquery
-- Make sure to use DISTINCT to show only relevant products
SELECT DISTINCT product
FROM #products
WHERE product IN (SELECT DISTINCT product
                  FROM #products
                  WHERE cost > 20);

-- NESTED SUBQUERY
SELECT DISTINCT product
FROM #products
WHERE product IN (SELECT product
                  FROM #products
                  GROUP BY product
                  HAVING MAX(cost) > (SELECT AVG(cost)
                                      FROM #products));
```

DERIVED TABLE

```
-- Derived table is an expression that generats a table within the scope of a query FROM clause.
-- SELECT ... FROM (subquery) [AS] tbl_name ...

SELECT
    AVG(t.max_cost * 1.0) as avg_max_cost
FROM (SELECT product, MAX(cost) as max_cost
      FROM #products
      GROUP BY product) t
```

CORRELATED SUBQUERY

```
-- For Correlated Subquery, there is no JOIN condition, but join is performed in the WHERE clause
-- To find products that cost more than average cost of the same product

SELECT product, cost, (SELECT AVG(cost)
                       FROM #products b
                       WHERE a.product = b.product) as prod_avg_cost
FROM #products a
WHERE cost > (SELECT AVG(cost)
              FROM #products b
              WHERE a.product = b.product)
```

COMMON TABLE EXPRESSIONS (CTE)

```
-- Common Table Expressions is a temporary result set, that can be referenced within a SELECT, INSERT, UPDATE, or DELETE
-- statement, that immediately follows the CTE. The CTE is stored in-memory and not on disk. CTE improves query performance
-- and organization of complicated queries.

-- Features
-- More than one CTE can be defined in one WITH statement.
-- Combine several CTEs with UNION or JOIN
-- CTEs can be a substitute for a View
-- CTEs can reference other CTEs
-- Referencing itself (SELF JOIN) aka, Recursive CTEs
```

```sql
WITH cte_name (Column1, Column2,...) AS (
    CTE_query
)

-- If no column names are specified then column names from the sub query will be used
WITH cte_name AS (
    CTE_query
)
```

CORRELATED SUBQUERY

DERIVED TABLE

COMMON TABLE EXPRESSION

EXISTS

ANY, ALL

WINDOW FUNCTIONS

WHILE LOOP

RECURSIVE

DECLARING VARIABLES

TYPES OF LANGUAGES

DML, DDL, DCL, TCL