More SQL functions

In this lecture

- String manipulation.
- Date/time manipulation.
- `CASE` and `COALESCE`.

String manipulation

Function	Description
`SUBSTR`	Extracts a portion of a string (Teradata extension).
`SUBSTRING`	Extracts a portion of a string (ANSI standard).
`INDEX`	Locates the position of a character in a string (Teradata extension).
`POSITION`	Locates the position of a character in a string (ANSI standard).
`TRIM`	Trims blanks from a string.
`UPPER`	Converts a string to uppercase.
`LOWER`	Converts a string to lowercase.

String manipulation examples

String Function	Result
`SELECT SUBSTRING(‘warehouse’ FROM 1 FOR 4)`	Ware
`SELECT SUBSTR(‘warehouse’,1,4)`	Ware
`SELECT ‘data’ || ‘ ‘ || ‘warehouse’`	data warehouse
`SELECT UPPER(‘data’)`	DATA
`SELECT LOWER(‘DATA’)`	Data

More String / Regex examples

- https://dbmstutorials.com/teradata/teradata_string_functions.html
- https://dbmstutorials.com/teradata/teradata-regular-Expression-functions.html

Date/Time

Date Storage

- Dates are stored as integer internally using the following formula.
  ```
  ((YEAR - 1900) * 10000) + (MONTH * 100) + DAY
  ```
- You can use the following query to check how the dates are stored.
  ```
  SELECT CAST(CURRENT_DATE AS INTEGER);
  ```
- Since the dates are stored as integer, you can perform some arithmetic operations on them.

EXTRACT

```
SELECT EXTRACT(YEAR FROM CURRENT_DATE);
SELECT EXTRACT(MONTH FROM CURRENT_DATE);
SELECT EXTRACT(DAY FROM CURRENT_DATE);
SELECT EXTRACT(HOUR FROM CURRENT_TIMESTAMP);
SELECT EXTRACT(MINUTE FROM CURRENT_TIMESTAMP);
SELECT EXTRACT(SECOND FROM CURRENT_TIMESTAMP);
```

INTERVAL

```sql
/* Add three years */
SELECT CURRENT_DATE
, CURRENT_DATE + INTERVAL '03' YEAR;

/* Add three years and 1 month */
SELECT CURRENT_DATE
, CURRENT_DATE + INTERVAL '03-01' YEAR TO MONTH;

/* Add 01 days, 05 hours and 10 minutes
to current timestamp */
SELECT CURRENT_TIMESTAMP
,CURRENT_TIMESTAMP + INTERVAL '01 05:10' DAY TO MINUTE;
```

`CASE` and `COALESCE`

`CASE`

- `CASE` expression evaluates each row against a condition (a `WHEN` clause) and returns the result of the first match.
- If there are no matches then the result from `ELSE` is returned.

```
CASE <expression>
WHEN <expression> THEN result-1
WHEN <expression> THEN result-2
...
ELSE
result-n
END
```

- **To improve performance, most common values should go first!**

Example

```sql
SELECT
EmployeeNo,
CASE DepartmentNo
WHEN 1 THEN 'Admin'
WHEN 2 THEN 'IT'
ELSE 'Invalid Dept'
```

Aggregate functions in `CASE`

- Use `CASE` as a filter for aggregate functions.

```
SELECT
Sum(CASE WHEN categoryid = 'CY'
    THEN productprice
    ELSE 0 END) AS price_cy
FROM Product
```

Percentage of total

```
SELECT
100*Sum(CASE WHEN categoryid = 'CY'
        THEN productprice ELSE 0 END)/Sum(productprice)
        AS pct_cy
FROM Product
```

`COALESCE`

- `COALESCE` is used to check if the argument is `NULL`, if it is `NULL` then it takes the default value.
- It will check for `NOT NULL` values sequentially in the list and it will return the first `NOT NULL` value.

Example

```
SELECT Name,
COALESCE (HomePhone, OfficePhone, 'No Phone')
AS ContactPhone
FROM  PhoneDirectory;
```

`NULLIF`

- The following example returns `NULL` if the DepartmentNo is equal to 3.
- Otherwise, it returns the DepartmentNo value.

```
EmployeeNo,
NULLIF(DepartmentNo,3) AS department
FROM Employee;
```

Subqueries

Subqueries in Teradata

- Subqueries are nested SELECT statement in order to provide output to outer query for data filtering purpose.
    - All subqueries must be enclosed in parentheses.
    - Subqueries can have multiple columns to match with main query.
    - Subqueries will always return unique list of values.
- Subqueries can be broadly classified into 2 categories:
    - Basic / Noncorrelated subquery
    - Correlated subquery

Subqueries (cont.)

Subqueries can be used in following SQL statements

- SELECT Statements to filter required rows.
- DELETE Statements to delete rows as returned by subquery output.
- UPDATE Statements.
- View Definitions to restrict data.
- Table creation to restrict limited set of data in new table.
- Subqueries support qualifiers like ALL, ANY, SOME, LIKE, NOT LIKE for outer query.
- Subqueries can be objects of an IN, NOT IN, EXISTS and NOT EXISTS clause.

Restrictions for subqueries

- Subqueries can be nested up to a depth of 64(maximun) else it will fail with below error.
- TOP n option cannot be used in subqueries else it will fail with below error.
- Sample clause cannot be used in subqueries else it will fail with below error.
- ORDER BY clause cannot be used in subqueries else it will fail with below error.

Basic subquery

- A basic subquery is a subquery that is independent of outer query but provides data to outer query to restrict result of final main query.
- **Example:**

```
    SELECT * FROM table1
    WHERE id IN
          ( SELECT
                id
            FROM
                table2
          );
```

Basic subquery (cont.)

```
    SELECT * FROM table1
    WHERE (id,transaction_dt) NOT IN
          ( SELECT
                id,
                transaction_dt
            FROM
                table2
          );
```

Correlated subquery

- A correlated subquery is a subquery that uses values from the outer query to restrict result of final main query.
- Queries with EXISTS / NOT EXISTS clauses will generally have correlated subqueries.
- **Example**:

```
SELECT EmployeeName
  ,DeptNo
  ,Salary
FROM Employee AS emp
  WHERE Salary <= (
  SELECT AVG(Salary) FROM Employee AS e
  WHERE emp.DeptNo  = e.DeptNo
  )
```

The dark side of subqueries

- In most cases JOINs are faster than sub-queries and it is very rare for a sub-query to be faster. This is because they run *once* per every row returned.
- Usually Optimizer can create an execution plan that is better for your query and can predict what data should be loaded to be processed and save time.
- **100x faster** (JOIN vs correlated subquery) is not uncommon.
- They are more readable than JOINs, hence most new SQL people prefer them.
- Sometimes you have no choice, though.

Subquery to Join

```
/*Sub query*/
SELECT e.*
FROM employee
WHERE DeptNo IN
(SELECT DeptNo
FROM department
WHERE DeptName LIKE ‘IT’);
```

```
/*Subquery to JOIN*/
SELECT e.*
FROM employee e
INNER JOIN department d
ON e.DeptNo = d.DeptNo
WHERE d.DeptName LIKE ‘IT’;
```

Note: The `*` is also not recommended,used only for brevity.

Your turn!

Exercise

- Import the file `OnlineNewsPopularitySmall`.
- Create the following table:

YYMM	BestName	BestShares	WorstName	WorstShares

- You need to use string and date time functions to parse the information.
- **Bonus points:** Given what you know from Teradata's architecture, how could we run this query in a more efficient way? Suppose we have a table of millions of rows with the same structure.