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

```
/* 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

```
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;
```

Your turn!

Exercise

- Import the file `OnlineNewsPopularitySmall`.
- Your task is to find:
  - Most/least shared article per year/month.
  - Months with the highest/lowest number of shares.
- You need to use string and date time functions to parse the information.
- If you had a lot of data, which auxiliary tables would you build (with different indices perhaps) to improve the performance of your queries?.