Implementation of Lookup table

Follow Allan for more such content :

How lookup table is more beneficial than using switch case

1. **Simpler code:** As the number of fields and conditions increase, switch case statements can quickly become cluttered and difficult to read. A lookup table can simplify the code by providing a clear mapping between input and output values.
2. **Improved performance:** Lookup tables can be indexed for faster data retrieval, whereas switch case statements require the SQL engine to evaluate each condition in turn. This can result in slower query performance for larger datasets.
3. **Easier maintenance:** With a lookup table, changes to the mapping can be made in a single location, whereas with switch case statements, each query needs to be updated individually. This can save time and reduce the risk of errors in your code.

Overall, using a lookup table can help to make your SQL queries more efficient, easier to read and maintain, and faster to execute.

Creation of Table and inserting values

```
--creating table
Create TABLE Test(

  emp_id VARCHAR ,
    month_num INTEGER
)


--Insert values into table
Insert into test
  values (1001,01),(1002,02),(1003,03)
        ,(1004,04),(1005,05),(1006,06)
        ,(1007,07)
```

Output

emp_id	day_num

The output of the SQL code will be the creation of a table named `Test` with two columns and the insertion of eight rows of data into the table.

1001	01
1002	02
1003	03
1004	04
1005	05
1006	06
1007	07

Query using Switch case

> 💡 Here we are assuming that day starts from Monday (i.e 1) and ends at Sunday (i.e 7)

```
SELECT
emp_id
,CASE
    when day_num =1 then 'Monday'
      when day_num = 2 then 'Tuesday'
      when day_num = 3 then 'Wednesday'
      when day_num = 4 then 'Thursday'
      when day_num = 5 then 'Friday'
      when day_num = 6 then 'Saturday'
      when day_num = 7 then 'Sunday'
    else 'error'
      END AS day_str

from test
```

Output

The SQL code uses a CASE statement to convert the numerical values in the `day_num` column into corresponding weekday names, which are then displayed in the new `day_str` column. The resulting table shows the `emp_id` and `day_str` columns for all rows in the "test" table.

emp_id	day_str
1001	Monday
1002	Tuesday
1003	Wednesday
1004	Thursday
1005	Friday

1006	Saturday
1007	Sunday

Alternative to switch case : Lookup Table

Lookup Table Creation

```
--creating lookup table
create table lookup_table(
day_str VARCHAR ,
 day_num INTEGER
)



--inserting values into Lookup Table
Insert into lookup_table (

  Select 'Monday' as day_str,1 as day_num
  UNION ALL
  Select 'Tuesday' as day_str,2 as day_num
  UNION ALL
  Select 'Wednesday' as day_str,3 as day_num
  UNION ALL
   Select 'Thursday' as day_str,4 as day_num
  UNION ALL
  Select 'Friday' as day_str,5 as day_num
  UNION ALL

  Select 'Saturday' as day_str,6 as day_num
  UNION ALL
  Select 'Sunday' as day_str,7 as day_num
  )
```

Output

day_str	**day_num**
Monday	1
Tuesday	2
Wednesday	3
Thursday	4
Friday	5
Saturday	6

The SQL code creates a new table named `lookup_table` with two columns: `day_str` and `day_num`.

It then inserts seven rows of data into the table using a SELECT statement with UNION ALL, which combines the results of several SELECT statements into a single table.

Each SELECT statement generates one row of data with the weekday name in the `day_str` column and its

Sunday	7

corresponding numerical value in the `day_num` column.

The resulting table shows the contents of the `lookup_table` with seven rows, where each row represents a weekday and its numerical value.

Joining both the tables

```
Select test.*,lookup_table.day_str from test
LEFT JOIN lookup_table ON
test.day_num = lookup_table.day_num
```

Output

emp_id	**day_num**	**day_str**
1001	1	Monday
1002	2	Tuesday
1003	3	Wednesday
1004	4	Thursday
1005	5	Friday
1006	6	Saturday
1007	7	Sunday

The values in the `day_str` column will correspond to the day of the week for each `day_num` value, as specified in the `lookup_table`.