**LAB-13**

**Aim:** Implement Snow-flake schema in Data warehousing.

**Theory:** The snowflake schema is a variant of the star schema. In this, the centralized fact table is connected to multiple dimensions, presented in a normalized form in multiple related tables. The snowflake structure is formed when the dimensions of a star schema are detailed and densely structured, having several levels of relationship, and the child tables have multiple parent tables. The snowflake effect influences only the dimension tables and not the fact tables.

Characteristics of Snowflake Schema:

* The major benefit is that it employs smaller disk space.
* It is easier to implement the addition of a dimension into the Schema.
* Presence of multiple tables reduces query performance.
* The primary challenge is to bear the maintenance efforts because of the larger number of lookup tables.

**Program Code:** The following DMQL program code can be used to define Snowflake schema -

DEFINE CUBE Sales [Timestamp, Product, Branch, Location]:

Dollars\_sold = sum(sales in dollars), Units\_sold = count(\*), Avg\_sales = avg(sales in dollars)

DEFINE DIMENSION Timestamp as (Time ID, day, day of week, month, quarter, year)

DEFINE DIMENSION Product as (Product ID, name, type, Supplier (Supplier ID, supplier type))

DEFINE DIMENSION Branch as (Branch ID, branch name)

DEFINE DIMENSION Location as (Location ID, City (City ID, city, state, country))

**Output:** The DMQL code generates following Snowflake Schema -

|  |
| --- |
| **SALES** |
| Time ID |
| Product ID |
| Location ID |
| Branch ID |
| Dollars\_sold |
| Units\_sold |
| Avg\_sales |

|  |
| --- |
| **TIMESTAMP** |
| Time ID |
| day |
| day of week |
| month |
| quarter |
| year |

|  |
| --- |
| **PRODUCT** |
| Product ID |
| name |
| type |
| Supplier ID |

|  |
| --- |
| **SUPPLIER** |
| Supplier ID |
| Supplier name |
| supplier type |

|  |
| --- |
| **BRANCH** |
| Branch ID |
| branch name |

|  |
| --- |
| **LOCATION** |
| Location ID |
| City name |
| City ID |

|  |
| --- |
| **CITY** |
| City ID |
| city |
| state |
| country |

**LAB-29**

**Aim:** Write a LISP program to compute factorial of a given number using recursion.

**Theory:** The factorial of a non-negative integer n (denoted by n!), is the product of all positive integers less than or equal to n.

Given by: n! =

The program code employs recursion to calculate the factorial, by using conditional statements to check the value of n. The function is recalled when n is greater than 0.

**Program Code:** The LISP Code will be as follows-

(defun factorial(n)

(if (= n 0) 1

(\* n (factorial(- n 1)))

)

)

(write-line "Factorial of 5 is: ")

(write (factorial 5))

(terpri)

(write-line "Factorial of 13 is: ")

(write (factorial 13))

**Output:**

