USN NUMBER: 1RVU23CSE227
NAME: Kruthik S

Ex No: 9 Date: 06-11-2025	Transforming a Normalized Model into a Star Schema

Objective:

To transform a normalized relational data model into a **Star Schema** structure that enhances query performance, simplifies data retrieval, and supports efficient analytical processing by organizing data into **Fact** and **Dimension** tables suitable for data warehousing and business intelligence applications.

Outcomes:

1. Understand the differences between normalized and dimensional data models.
2. Design and implement a star schema from normalized tables.
3. Create fact and dimension tables optimized for analytical queries.
4. Load data from normalized schema into star schema tables.
5. Execute analytical queries to validate schema transformation and performance.

Materials

- **Tool: Snowflake / SQL Worksheet**
- **Database: SALES_DB**
- **Schema: SALES_SCHEMA**
- **Tables: Customers, Employees, Offices, Orders, OrderDetails, Products, ProductLines**
- **Artifacts: SQL scripts from Lab.txt**

Architecture

The Normalized Model consists of multiple tables representing entities such as Customers, Employees, Products, Orders, etc., linked via foreign keys.

The Star Schema transforms this model into:

- **Fact Table:** FACT_ORDERS — holds measurable business metrics (e.g., quantity, sales).
- **Dimension Tables:** DIM_CUSTOMER, DIM_PRODUCT, DIM_EMPLOYEE, etc., containing descriptive attributes.

Normalized Tables → Dimension Tables → Fact Table → Analytical Queries

This design denormalizes relationships for better query speed and readability.

USN NUMBER: 1RVU23CSE227
NAME: Kruthik S

Lab Procedure

Step 1: Create Database and Schema

```
CREATE DATABASE IF NOT EXISTS SALES_DB;
USE DATABASE SALES_DB;

CREATE SCHEMA IF NOT EXISTS SALES_SCHEMA;
USE SCHEMA SALES_SCHEMA;
```

Step 2: Create Normalized Tables (ClassicModels Example)

```
CREATE OR REPLACE TABLE CUSTOMERS (
  customerNumber INT PRIMARY KEY,
  customerName STRING,
  contactFirstName STRING,
  contactLastName STRING,
  phone STRING,
  addressLine1 STRING,
  addressLine2 STRING,
  city STRING,
  state STRING,
  postalCode STRING,
  country STRING,
  salesRepEmployeeNumber INT,
  creditLimit NUMBER(10,2)
);

CREATE OR REPLACE TABLE EMPLOYEES (
  employeeNumber INT PRIMARY KEY,
  lastName STRING,
  firstName STRING,
  extension STRING,
  email STRING,
  officeCode STRING,
  reportsTo INT,
  jobTitle STRING);
```

USN NUMBER: 1RVU23CSE227
NAME: Kruthik S

Step 3: Insert Sample Data

```
INSERT INTO CUSTOMERS VALUES
(103, 'Atelier graphique', 'Carine', 'Schmitt', '40.32.2555',
 '54, rue Royale', NULL, 'Nantes', NULL, '44000', 'France', 1370, 21000);

INSERT INTO EMPLOYEES VALUES
(1370, 'Schmitt', 'Carine', 'x1234', 'carine.schmitt@classicmodels.com', '1', NULL, 'Sales Rep');

INSERT INTO OFFICES VALUES
('1', 'San Francisco', '+1 650 219 4782', '100 Market St.', NULL, 'CA', 'USA', '94080', 'NA'),
('2', 'Boston', '+1 215 837 0825', '1550 Court Place', NULL, 'MA', 'USA', '02107', 'NA');

INSERT INTO ORDERS VALUES
(10100, '2025-10-01', '2025-10-10', '2025-10-05', 'Shipped', 103),
(10101, '2025-10-02', '2025-10-12', '2025-10-07', 'Shipped', 112),
(10102, '2025-10-03', '2025-10-13', NULL, 'In Process', 114);

INSERT INTO ORDERDETAILS VALUES
(10100, 'S10_1678', 30, 95.70, 1),
(10100, 'S10_1949', 50, 53.80, 2),
(10101, 'S12_1099', 22, 32.90, 1),
(10102, 'S18_1749', 49, 136.00, 1);

INSERT INTO PRODUCTLINES VALUES
('Motorcycles', 'Motorcycle products', '<html>Motorcycle products</html>'),
('Classic Cars', 'Classic car models', '<html>Classic car models</html>');
```

By transforming normalized data into a **Star Schema**, data redundancy is reduced, analytical queries become faster, and schema readability improves.

This schema supports **OLAP-style analysis**, enabling easy aggregations and drill-downs compared to complex joins required in normalized models.

USN NUMBER: 1RVU23CSE227
NAME: Kruthik S

Step 4: Transform to Star Schema

- Create dimension tables (DIM_CUSTOMER, DIM_PRODUCT, etc.)
- Create a fact table (FACT_ORDERS)

Example:

```
CREATE OR REPLACE TABLE FACT_ORDERS AS
SELECT
  o.orderNumber AS order_id,
  o.orderDate,
  c.customerNumber AS customer_id,
  p.productCode AS product_id,
  od.quantityOrdered AS quantity,
  od.priceEach AS price,
  (od.quantityOrdered * od.priceEach) AS total_amount
FROM ORDERS o
JOIN ORDERDETAILS od ON o.orderNumber = od.orderNumber
JOIN CUSTOMERS c ON o.customerNumber = c.customerNumber
JOIN PRODUCTS p ON od.productCode = p.productCode;
```

Step 5: Run Analytical Query

```
SELECT
  customer_id,
  product_id,
  SUM(quantity) AS total_quantity
FROM SALES_DB.SALES_SCHEMA.FACT_ORDERS
GROUP BY customer_id, product_id
LIMIT 50;
```

GitHub Link: https://github.com/kruth-s/Data-Engg-Lab