Snowsql

We load the file US-SuperStore to Snowflake database after splitting it and converting them to a .csv files.

Goal: We will use the 2 Excel sheets "ORDER" and "PEOPLE" to form 1 fact and 6 dimension tables.

Load the "PEOPLE" excel sheet as DIM_MANAGER table.(to get our first Dimension table)
Load the "ORDER" sheet as FactStoreSales table.

1) The spaces in the column names must be replaced by an underscore (_).
2) Change the names of the column SUB-CATEGORY to SUB_CATEGORY ..
3) The column COUNTRY (redundant) since the data is taken from superstores across the USA only → column COUNTRY will not be loaded.

SnowSQL QUERY:

```
/* Split the column "REGIONAL_MANAGER" into two : "MANAGER_FIRST_NAME" and "MANAGER_LAST_NAME" in DIM_MANAGER */
//Add new columns
        ALTER TABLE DIM_MANAGER
        ADD (MANAGER_FIRST_NAME VARCHAR2(50), MANAGER_LAST_NAME VARCHAR2(50));
//Update the new columns with the split values
        UPDATE DIM_MANAGER
        SET
                MANAGER_FIRST_NAME = SUBSTRING(Regional_Manager, 1, POSITION(' ' IN Regional_Manager) - 1),
                MANAGER_LAST_NAME = SUBSTRING(Regional_Manager, POSITION(' ' IN Regional_Manager) + 1);
//Drop the original column
        ALTER TABLE DIM_MANAGER
        DROP COLUMN REGIONAL_MANAGER;

//Add a new column for the primary key and generate unique values

//Creating the Column
        ALTER TABLE DIM_MANAGER
        ADD (MANAGER_PK NUMBER(20));
//Generate unique values
        MERGE INTO DIM_MANAGER USING (
        SELECT
        ROW_NUMBER() OVER (ORDER BY REGION) AS RowNum,
        REGION
        FROM DIM_MANAGER
        ) AS SOURCE
```

```
        ON DIM_MANAGER.REGION = SOURCE.REGION
                WHEN MATCHED THEN UPDATE SET DIM_MANAGER.MANAGER_PK =
        SOURCE.RowNum;
//Assigning a primary key
        ALTER TABLE DIM_MANAGER
        ADD CONSTRAINT PK_Manager PRIMARY KEY (MANAGER_PK)

/* Add a new column in the table FactStoreSales that will be the foreign key linking to
DIM_MANAGER */
        ALTER TABLE FactStoreSales
        ADD (MANAGER_FK NUMBER(20));
//Update the new column with corresponding primary key values
        UPDATE FactStoreSales fs
        SET MANAGER_FK = dm.MANAGER_PK
        FROM DIM_MANAGER dm
        WHERE fs.REGION = dm.REGION;
//Add the foreign key constraint
        ALTER TABLE FactStoreSales
        ADD CONSTRAINT FK_Manager FOREIGN KEY(MANAGER_FK) REFERENCES
        DIM_MANAGER(MANAGER_PK)
/* Drop the REGION column from DIM_MANAGER as we will include it in another
dimension later.*/
        ALTER TABLE DIM_MANAGER
        DROP COLUMN REGION

//We Extract 5 dimensions from the main FactStoreSales : Picking adequate primary keys
is crucial.

/* Extracting Customer table: each CUSTOMER_ID references a unique
CUSTOMER_NAME making it perfect as a primary key.*/

        CREATE TABLE DIM_CUSTOMER AS
        SELECT DISTINCT CUSTOMER_ID, CUSTOMER_NAME , Segment
        FROM FactStoreSales;

        ALTER TABLE DIM_CUSTOMER
        ADD CONSTRAINT PK_Customer PRIMARY KEY (CUSTOMER_ID);

        ALTER TABLE FactStoreSales
        ADD CONSTRAINT FK_Customer FOREIGN KEY(CUSTOMER_ID) REFERENCES
        DIM_CUSTOMER(CUSTOMER_ID);

/* Extracting Product table: The values of Product_ID can not represent a primary key and
the reason for that is in certain instances a Product_ID may reference
```

more than one product name (This is because throughout the years in this business, a product may no longer be sold leaving the newly available Product_ID ready to be used again) */

/* Solution : Create a new column in table FactStoreSales (will play the role of a foreign key referencing Product table later) --> assign a unique key to each distinct combination --> Extract the needed columns to the new Product Table. As follows */

```
ALTER TABLE FactStoreSales
ADD (PRODUCT_PK INTEGER);

MERGE INTO FactStoreSales o
        USING ( SELECT ROW_ID as rid, DENSE_RANK() OVER (ORDER BY
        PRODUCT_ID,PRODUCT_NAME) as new_id
FROM FactStoreSales
          ) t
ON (o.ROW_ID = t.rid)
WHEN MATCHED THEN UPDATE SET o.PRODUCT_PK = t.new_id;

CREATE TABLE DIM_PRODUCT AS
SELECT DISTINCT PRODUCT_PK,PRODUCT_ID, PRODUCT_NAME, CATEGORY,
SUB_CATEGORY
FROM FactStoreSales;
```

//Adding Primary and Foreign keys

```
ALTER TABLE DIM_PRODUCT
ADD CONSTRAINT PK_Product PRIMARY KEY (PRODUCT_PK);

ALTER TABLE FactStoreSales
ADD CONSTRAINT FK_Product FOREIGN KEY(PRODUCT_PK) REFERENCES
DIM_PRODUCT(PRODUCT_PK);
```

/* Extracting Location table: we want to use POSTAL_CODE as a primary key , however after checking */

```
SELECT POSTAL_CODE, COUNT(*) FROM
(SELECT distinct POSTAL_CODE,CITY FROM FactStoreSales)
GROUP BY POSTAL_CODE
HAVING COUNT(*) > 1;
```

→ the result : 92024 **/* which means that this postal code references more than 1 city (San Francisco & Encinitas) so we can not use it as a primary key*/**

// We will use same solution as Product table

```
ALTER TABLE FactStoreSales
ADD (LOCATION_PK INTEGER);


MERGE INTO FactStoreSales o
        USING (SELECT ROW_ID as rid,DENSE_RANK() OVER (ORDER BY
        POSTAL_CODE, CITY) as new_id
FROM FactStoreSales
                ) t
ON (o.ROW_ID = t.rid)
WHEN MATCHED THEN UPDATE SET o.LOCATION_PK = t.new_id;

CREATE TABLE DIM_LOCATION AS
SELECT DISTINCT LOCATION_PK,POSTAL_CODE,CITY,STATE,REGION
FROM FactStoreSales
```

//Add primary and foreign keys

```
ALTER TABLE DIM_LOCATION
ADD CONSTRAINT PK_Location PRIMARY KEY (LOCATION_PK)

ALTER TABLE FactStoreSales
ADD CONSTRAINT FK_Location FOREIGN KEY(LOCATION_PK) REFERENCES
DIM_LOCATION(LOCATION_PK)
```

//Extracting Date table

```
CREATE TABLE DIM_DATE (FULL_DATE_PK DATE);
```

/* Creating a sequence and generating date values from 01/01/2019 up to 31/12/2023 in the column FULL_DATE_PK */

```
INSERT INTO DIM_DATE (FULL_DATE_PK)
WITH RECURSIVE DateSequence AS (
SELECT TO_DATE('01/01/2019', 'DD/MM/YYYY') AS date_value
UNION ALL
SELECT date_value + 1
FROM DateSequence
WHERE date_value + 1 <= TO_DATE('31/12/2023', 'DD/MM/YYYY')
)
SELECT date_value
FROM DateSequence;
```

//Adding primary and foreign key constraints

```
ALTER TABLE DIM_DATE
ADD CONSTRAINT PK_Date PRIMARY KEY(FULL_DATE_PK);
```

```
        ALTER TABLE FactStoreSales
        ADD CONSTRAINT FK_SaleDate FOREIGN KEY(ORDER_DATE) REFERENCES
        DIM_DATE(FULL_DATE_PK);

        ALTER TABLE FactStoreSales
        ADD CONSTRAINT FK_ShipDate FOREIGN KEY(SHIP_DATE) REFERENCES
         DIM_DATE(FULL_DATE_PK);
//Extracting Shipment table
        CREATE TABLE DIM_SHIPMENT AS
        SELECT DISTINCT SHIP_MODE FROM FactStoreSales

//Adding primary and foreign keys
        ALTER TABLE DIM_SHIPMENT
        ADD CONSTRAINT PK_Shipment PRIMARY KEY(SHIP_MODE);

        ALTER TABLE FactStoreSales
        ADD CONSTRAINT FK_Shipment FOREIGN KEY(SHIP_MODE) REFERENCES
        DIM_SHIPMENT(SHIP_MODE);



/*After finishing the extraction of the needed tables.We change the names of some
columns for extra clarity and to avoid confusion: */

/* FactStoreSales:
ROW_ID --> SALE_PK
CUSTOMER_ID --> CUSTOMER_FK
ORDER_DATE --> SALE_DATE_FK
SHIP_DATE --> SHIP_DATE_FK
LOCATION_PK --> LOCATION_FK
PRODUCT_PK --> PRODUCT_FK
MANAGER_PK --> MANAGER_FK
DIM_CUSTOMER:
CUSTOMER_ID --> CUSTOMER_PK */

        ALTER TABLE FactStoreSales
        RENAME COLUMN ROW_ID TO SALE_PK

        ALTER TABLE FactStoreSales
        RENAME COLUMN CUSTOMER_ID to CUSTOMER_FK

        ALTER TABLE FactStoreSales
        RENAME COLUMN LOCATION_PK to LOCATION_FK
```

```
ALTER TABLE FactStoreSales
RENAME COLUMN ORDER_DATE to SALE_DATE_FK

ALTER TABLE FactStoreSales
RENAME COLUMN SHIP_DATE to SHIP_DATE_FK

ALTER TABLE FactStoreSales
RENAME COLUMN PRODUCT_PK to PRODUCT_FK

ALTER TABLE FactStoreSales
RENAME COLUMN SHIP_MODE to SHIP_MODE_FK
```

//DIM_CUSTOMER:

```
ALTER TABLE DIM_CUSTOMER
RENAME COLUMN CUSTOMER_ID TO CUSTOMER_PK
```

/* We drop the columns CUSTOMER_NAME,SEGMENT,CITY,REGION,STATE,POSTAL_CODE,PRODUCT_ID,CATEGORY,SUB_CATEGORY,PRODUCT_NAME from FactStoreSales */

```
ALTER TABLE FactStoreSales
DROP COLUMN CUSTOMER_NAME,SEGMENT,CITY,REGION,STATE,ORDER_ID
POSTAL_CODE,PRODUCT_ID,CATEGORY,SUB_CATEGORY,PRODUCT_NAME;
```

//We are left with the main table that contains business measures and the foreign keys for the other tables.
//We set a primary key for the table FactStoreSales.

```
ALTER TABLE FactStoreSales
ADD CONSTRAINT PK_Sale PRIMARY KEY(SALE_PK)
```

//The last step is to export our data in the desired format.

ROLAP QUERIES

//Display the total profit per manager

```
SELECT
    dm.MANAGER_FIRST_NAME,
    dm.MANAGER_LAST_NAME,
    SUM(fs.PROFIT) AS total_profit
FROM
    factstoresales fs
JOIN
    DIM_MANAGER dm ON fs.MANAGER_FK = dm.MANAGER_PK
GROUP BY
    dm.MANAGER_FIRST_NAME,
    dm.MANAGER_LAST_NAME;
```

// SELECT average delay per shipment mode

```
    SHIP_MODE_FK,
    AVG(fs.SHIP_DATE_FK - fs.SALE_DATE_FK) AS avg_shipping_time
FROM
    FactStoreSales fs
GROUP BY
    fs.SHIP_MODE_FK
ORDER BY
    avg_shipping_time DESC;
```

//Display quarter sales

```
SELECT
    EXTRACT(QUARTER FROM dd.FULL_DATE_PK) AS quarter,
    SUM(fs.SALES) AS total_sales
FROM
    DIM_DATE dd
LEFT JOIN
    FactStoreSales fs ON dd.FULL_DATE_PK = fs.SALE_DATE_FK
GROUP BY
    quarter
ORDER BY
    quarter;
```

//total sales, average profit, total quantity, minimum quantity, maximum quantity, and number of distinct customers for each city.

```
SELECT
    dl.CITY,
    SUM(fs.SALES) AS total_sales,
    AVG(fs.PROFIT) AS average_profit,
    SUM(fs.QUANTITY) AS total_quantity,
    MIN(fs.QUANTITY) AS min_quantity,
    MAX(fs.QUANTITY) AS max_quantity,
    COUNT(DISTINCT fs.CUSTOMER_FK) AS distinct_customers
FROM
    FactStoreSales fs
JOIN
    DIM_LOCATION dl ON fs.LOCATION_FK = dl.LOCATION_PK
GROUP BY
    dl.CITY
ORDER BY
    total_sales DESC;
```

// SELECT monthly sales and customers

```
    TO_CHAR(dd.FULL_DATE_PK, 'YYYY-MM') AS month,
    COUNT(DISTINCT fs.CUSTOMER_FK) AS distinct_customers,
    SUM(fs.SALES) AS total_sales
FROM
    DIM_DATE dd
LEFT JOIN
    FactStoreSales fs ON dd.FULL_DATE_PK = fs.SALE_DATE_FK
GROUP BY
    month
ORDER BY
    Month;
```

//Discount amount grouped by category and sub category

```
SELECT
    dp.CATEGORY,
    dp.SUB_CATEGORY,
    SUM(fs.DISCOUNT * fs.SALES) AS discount_amount
FROM
    FactStoreSales fs
JOIN
    DIM_PRODUCT dp ON fs.PRODUCT_FK = dp.PRODUCT_PK
GROUP BY
    dp.CATEGORY,
    dp.SUB_CATEGORY
ORDER BY
    dp.CATEGORY,
    dp.SUB_CATEGORY;
```