```sql
-- Step 1: Create Database
DROP DATABASE IF EXISTS RetailSalesDW;
CREATE DATABASE RetailSalesDW;
USE RetailSalesDW;

-- Step 2: Create Dimension Tables

-- Customer Dimension Table
CREATE TABLE dim_customer (
    customer_id VARCHAR(20) PRIMARY KEY,
    gender ENUM('Male', 'Female', 'Other'),
    age INT
) ENGINE=InnoDB;

-- Product Dimension Table
CREATE TABLE dim_product (
    product_category VARCHAR(100) PRIMARY KEY
) ENGINE=InnoDB;

-- Date Dimension Table
CREATE TABLE dim_date (
    date DATE PRIMARY KEY,
    year INT,
    month INT,
    day INT,
    weekday VARCHAR(10)
) ENGINE=InnoDB;

-- Step 3: Create Fact Table
CREATE TABLE fact_sales (
    transaction_id INT PRIMARY KEY,
    date DATE,
    customer_id VARCHAR(20),
    product_category VARCHAR(100),
    quantity INT,
    price_per_unit DECIMAL(10,2),
    total_amount DECIMAL(10,2),
    FOREIGN KEY (date) REFERENCES dim_date(date),
    FOREIGN KEY (customer_id) REFERENCES dim_customer(customer_id),
    FOREIGN KEY (product_category) REFERENCES
dim_product(product_category)
) ENGINE=InnoDB;

-- Step 4: Create a Temporary Table for Raw Data
DROP TEMPORARY TABLE IF EXISTS temp_sales;
CREATE TEMPORARY TABLE temp_sales (
    transaction_id VARCHAR(50),
    date_str VARCHAR(50),
    customer_id VARCHAR(50),
    gender VARCHAR(10),
    age VARCHAR(10),
    product_category VARCHAR(100),
    quantity VARCHAR(20),
    price_per_unit VARCHAR(20),
    total_amount VARCHAR(20)
);

-- Step 5: Load Data into the Temporary Table
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server
8.0/Uploads/retail_sales_dataset.csv'
INTO TABLE temp_sales
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

-- Step 6: Populate Dimension Tables from temp_sales

-- Populate dim_date
INSERT IGNORE INTO dim_date (date, year, month, day, weekday)
SELECT DISTINCT
       STR_TO_DATE(date_str, '%Y-%m-%d') AS date,
       YEAR(STR_TO_DATE(date_str, '%Y-%m-%d')) AS year,
       MONTH(STR_TO_DATE(date_str, '%Y-%m-%d')) AS month,
       DAY(STR_TO_DATE(date_str, '%Y-%m-%d')) AS day,
       DAYNAME(STR_TO_DATE(date_str, '%Y-%m-%d')) AS weekday
FROM temp_sales
WHERE date_str IS NOT NULL;

-- Populate dim_customer
INSERT IGNORE INTO dim_customer (customer_id, gender, age)
SELECT DISTINCT
       TRIM(customer_id) AS customer_id,
       CASE
         WHEN gender = 'M' THEN 'Male'
         WHEN gender = 'F' THEN 'Female'
         ELSE 'Other'
       END AS gender,
       CAST(age AS UNSIGNED) AS age
FROM temp_sales
WHERE customer_id IS NOT NULL;

-- Populate dim_product
INSERT IGNORE INTO dim_product (product_category)
SELECT DISTINCT TRIM(product_category) AS product_category
FROM temp_sales
WHERE product_category IS NOT NULL;

-- Step 7: Populate Fact Table from temp_sales
INSERT INTO fact_sales (transaction_id, date, customer_id,
product_category, quantity, price_per_unit, total_amount)
SELECT
    CAST(transaction_id AS UNSIGNED),
    STR_TO_DATE(date_str, '%Y-%m-%d'),
    TRIM(customer_id),
    TRIM(product_category),
    CAST(quantity AS UNSIGNED),
    CAST(price_per_unit AS DECIMAL(10,2)),
    CAST(total_amount AS DECIMAL(10,2))
```

```
FROM temp_sales;

-- Step 8: Indexing for Performance
CREATE INDEX idx_date ON fact_sales(date);
CREATE INDEX idx_customer_id ON fact_sales(customer_id);
CREATE INDEX idx_product_category ON fact_sales(product_category);

-- Step 9: Analytical Queries

-- Total Sales by Product Category
SELECT dp.product_category, SUM(fs.total_amount) AS total_sales
FROM fact_sales fs
JOIN dim_product dp ON fs.product_category = dp.product_category
GROUP BY dp.product_category;

-- Sales Over Time
SELECT dd.year, dd.month, SUM(fs.total_amount) AS monthly_sales
FROM fact_sales fs
JOIN dim_date dd ON fs.date = dd.date
GROUP BY dd.year, dd.month
ORDER BY dd.year, dd.month;

-- Customer Demographics Analysis
SELECT dc.gender, AVG(dc.age) AS average_age, SUM(fs.total_amount) AS
total_spent
FROM fact_sales fs
JOIN dim_customer dc ON fs.customer_id = dc.customer_id
GROUP BY dc.gender;

-- Step 10: Stored Procedure for Monthly Sales Report
DELIMITER //
CREATE PROCEDURE GetMonthlySalesReport(IN report_year INT, IN report_month
INT)
BEGIN
    SELECT dd.year, dd.month, dp.product_category, SUM(fs.total_amount) AS
total_sales
    FROM fact_sales fs
    JOIN dim_date dd ON fs.date = dd.date
    JOIN dim_product dp ON fs.product_category = dp.product_category
    WHERE dd.year = report_year AND dd.month = report_month
    GROUP BY dd.year, dd.month, dp.product_category;
END //
DELIMITER ;

-- Step 11: Call the Stored Procedure for January 2023
CALL GetMonthlySalesReport(2023, 1);
```