# Task-1: Database Creation and Table Structure

## Create a database:

CREATE DATABASE SupplyChainFinanceManagement;

## Create the nine tables:

### dim\_customer

CREATE TABLE dim\_customer (

customer\_id INT PRIMARY KEY,

customer\_name VARCHAR(255),

region VARCHAR(255),

customer\_segment VARCHAR(255),

purchase\_frequency INT,

total\_purchases DECIMAL(10, 2)

);

### dim\_product

CREATE TABLE dim\_product (

product\_id INT PRIMARY KEY,

product\_name VARCHAR(255),

product\_category VARCHAR(255),

product\_price DECIMAL(10, 2)

);

### fact\_sales\_monthly

CREATE TABLE fact\_sales\_monthly (

sale\_id INT PRIMARY KEY,

product\_id INT,

customer\_id INT,

sale\_date DATE,

quantity\_sold INT,

gross\_price DECIMAL(10, 2),

total\_sales DECIMAL(10, 2),

FOREIGN KEY (product\_id) REFERENCES dim\_product(product\_id),

FOREIGN KEY (customer\_id) REFERENCES dim\_customer(customer\_id)

);

### fact\_forecast\_monthly

CREATE TABLE fact\_forecast\_monthly (

forecast\_id INT PRIMARY KEY,

product\_id INT,

customer\_id INT,

forecasted\_quantity INT,

forecast\_date DATE,

fiscal\_year INT,

FOREIGN KEY (product\_id) REFERENCES dim\_product(product\_id),

FOREIGN KEY (customer\_id) REFERENCES dim\_customer(customer\_id)

);

### fact\_freight\_cost

CREATE TABLE fact\_freight\_cost (

freight\_id INT PRIMARY KEY,

product\_id INT,

market VARCHAR(255),

freight\_cost DECIMAL(10, 2),

fiscal\_year INT,

FOREIGN KEY (product\_id) REFERENCES dim\_product(product\_id)

);

### fact\_gross\_price

CREATE TABLE fact\_gross\_price (

price\_id INT PRIMARY KEY,

product\_id INT,

gross\_price DECIMAL(10, 2),

fiscal\_year INT,

FOREIGN KEY (product\_id) REFERENCES dim\_product(product\_id)

);

### fact\_manufacturing\_cost

CREATE TABLE fact\_manufacturing\_cost (

manufacturing\_cost\_id INT PRIMARY KEY,

product\_id INT,

manufacturing\_cost DECIMAL(10, 2),

freight\_cost DECIMAL(10, 2),

other\_cost DECIMAL(10, 2),

fiscal\_year INT,

FOREIGN KEY (product\_id) REFERENCES dim\_product(product\_id)

);

### fact\_post\_invoice\_deductions

CREATE TABLE fact\_post\_invoice\_deductions (

post\_invoice\_id INT PRIMARY KEY,

product\_id INT,

customer\_id INT,

post\_invoice\_deduction DECIMAL(10, 2),

deduction\_type VARCHAR(255),

fiscal\_year INT,

FOREIGN KEY (product\_id) REFERENCES dim\_product(product\_id),

FOREIGN KEY (customer\_id) REFERENCES dim\_customer(customer\_id)

);

### fact\_pre\_invoice\_deductions

CREATE TABLE fact\_pre\_invoice\_deductions (

pre\_invoice\_id INT PRIMARY KEY,

product\_id INT,

customer\_id INT,

pre\_invoice\_deduction DECIMAL(10, 2),

fiscal\_year INT,

FOREIGN KEY (product\_id) REFERENCES dim\_product(product\_id),

FOREIGN KEY (customer\_id) REFERENCES dim\_customer(customer\_id)

);

# Task-2: Insert Records

## Insert records into dim\_customer

INSERT INTO dim\_customer (customer\_id, customer\_name, region, customer\_segment, purchase\_frequency, total\_purchases)

VALUES

(1, 'Croma', 'North America', 'Retail', 15, 2500.00),

(2, 'Best Buy', 'North America', 'Retail', 10, 3000.00),

(3, 'Amazon', 'Global', 'E-Commerce', 25, 5000.00);

## Insert records into dim\_product

INSERT INTO dim\_product (product\_id, product\_name, product\_category, product\_price)

VALUES

(1, 'Wireless Mouse', 'Accessories', 29.99),

(2, 'Gaming Keyboard', 'Accessories', 49.99),

(3, 'Laptop X300', 'Computers', 899.99);

## Insert records into fact\_sales\_monthly

INSERT INTO fact\_sales\_monthly (sale\_id, product\_id, customer\_id, sale\_date, quantity\_sold, gross\_price, total\_sales)

VALUES

(1, 1, 1, '2023-07-01', 50, 29.99, 1499.50),

(2, 2, 2, '2023-07-02', 30, 49.99, 1499.70),

(3, 3, 3, '2023-07-03', 10, 899.99, 8999.90);

## Insert records into fact\_forecast\_monthly

INSERT INTO fact\_forecast\_monthly (forecast\_id, product\_id, customer\_id, forecasted\_quantity, forecast\_date, fiscal\_year)

VALUES

(1, 1, 1, 100, '2023-06-01', 2023),

(2, 2, 2, 150, '2023-06-01', 2023),

(3, 3, 3, 200, '2023-06-01', 2023);

## Insert records into fact\_freight\_cost

INSERT INTO fact\_freight\_cost (freight\_id, product\_id, market, freight\_cost, fiscal\_year)

VALUES

(1, 1, 'North America', 100.50, 2023),

(2, 2, 'Europe', 150.00, 2023),

(3, 3, 'Global', 250.00, 2023);

## Insert records into fact\_gross\_price

INSERT INTO fact\_gross\_price (price\_id, product\_id, gross\_price, fiscal\_year)

VALUES

(1, 1, 29.99, 2023),

(2, 2, 49.99, 2023),

(3, 3, 899.99, 2023);

## Insert records into fact\_manufacturing\_cost

INSERT INTO fact\_manufacturing\_cost (manufacturing\_cost\_id, product\_id, manufacturing\_cost, freight\_cost, other\_cost, fiscal\_year)

VALUES

(1, 1, 20.00, 5.00, 3.00, 2023),

(2, 2, 30.00, 7.00, 4.00, 2023),

(3, 3, 600.00, 50.00, 20.00, 2023);

## Insert records into fact\_post\_invoice\_deductions

INSERT INTO fact\_post\_invoice\_deductions (post\_invoice\_id, product\_id, customer\_id, post\_invoice\_deduction, deduction\_type, fiscal\_year)

VALUES

(1, 1, 1, 3.00, 'Performance Rebate', 2023),

(2, 2, 2, 2.50, 'Promotional Discount', 2023),

(3, 3, 3, 10.00, 'Placement Fee', 2023);

## Insert records into fact\_pre\_invoice\_deductions

INSERT INTO fact\_pre\_invoice\_deductions (pre\_invoice\_id, product\_id, customer\_id, pre\_invoice\_deduction, fiscal\_year)

VALUES

(1, 1, 1, 2.00, 2023),

(2, 2, 2, 1.50, 2023),

(3, 3, 3, 5.00, 2023);

# Task-3: Queries for Fiscal Year and Transactions

## Determine fiscal year from a date (2023-07-15):

Fiscal year starts in September.

SELECT CASE

WHEN MONTH('2023-07-15') >= 9 THEN YEAR('2023-07-15') + 1

ELSE YEAR('2023-07-15')

END AS fiscal\_year;

## Monthly product transaction report:

Query to join relevant tables for product transaction details:

SELECT

fs.sale\_date,

dp.product\_code,

dp.product\_name,

dp.variant,

fs.quantity\_sold,

fg.gross\_price,

(fs.quantity\_sold \* fg.gross\_price) AS gross\_price\_total

FROM

fact\_sales\_monthly fs

JOIN

dim\_product dp ON fs.product\_id = dp.product\_id

JOIN

fact\_gross\_price fg ON fs.product\_id = fg.product\_id

WHERE

fs.customer\_id = 1 -- Replace with customer code

AND YEAR(fs.sale\_date) = 2023; -- Replace with the fiscal year

# Task-4: Advanced Analysis Queries

## Monthly Sales Trend Analysis:

SELECT

fs.sale\_date,

dp.product\_name,

SUM(fs.quantity\_sold) AS total\_quantity\_sold

FROM

fact\_sales\_monthly fs

JOIN

dim\_product dp ON fs.product\_id = dp.product\_id

GROUP BY fs.sale\_date, dp.product\_name

ORDER BY fs.sale\_date;

## Customer Segmentation (using dim\_customer):

SELECT

customer\_segment,

SUM(total\_purchases) AS total\_revenue

FROM

dim\_customer

GROUP BY customer\_segment

ORDER BY total\_revenue DESC;

## Product Performance Comparison:

SELECT

dp.product\_name,

SUM(fs.quantity\_sold) AS total\_quantity,

SUM(fs.total\_sales) AS total\_revenue

FROM

fact\_sales\_monthly fs

JOIN

dim\_product dp ON fs.product\_id = dp.product\_id

GROUP BY dp.product\_name

ORDER BY total\_quantity DESC;

## Market Expansion Opportunities:

SELECT

region,

SUM(ff.forecasted\_quantity) AS total\_forecasted\_quantity

FROM

fact\_forecast\_monthly ff

JOIN

dim\_customer dc ON ff.customer\_id = dc.customer\_id

GROUP BY region

ORDER BY total\_forecasted\_quantity DESC;

# Task-5: Advanced SQL Procedures and Functions

## User-defined function to calculate total forecasted quantity:

CREATE FUNCTION total\_forecasted\_quantity(@product\_id INT, @fiscal\_year INT)

RETURNS DECIMAL(10,2)

AS

BEGIN

DECLARE @total DECIMAL(10,2);

SELECT @total = SUM(forecasted\_quantity)

FROM fact\_forecast\_monthly

WHERE product\_id = @product\_id AND fiscal\_year = @fiscal\_year;

RETURN @total;

END;

## Stored Procedure to Update Gross Price:

CREATE PROCEDURE UpdateGrossPrice

@product\_id INT,

@new\_gross\_price DECIMAL(10,2),

@fiscal\_year INT

AS

BEGIN

UPDATE fact\_gross\_price

SET gross\_price = @new\_gross\_price

WHERE product\_id = @product\_id AND fiscal\_year = @fiscal\_year;

END;

### Trigger for Audit Log on Sales Table:

CREATE TRIGGER SalesAuditTrigger

ON fact\_sales\_monthly

AFTER INSERT

AS

BEGIN

INSERT INTO audit\_log (action, log\_date, details)

VALUES ('New Sale Added', GETDATE(), 'A new sale has been added to fact\_sales\_monthly');

END;

# Task-6: Forecast Accuracy Analysis

## SQL Query for forecast accuracy using Pivot:

SELECT

p.product\_name,

MONTH(sale\_date) AS month,

SUM(fm.forecasted\_quantity) AS forecasted\_quantity,

SUM(fs.quantity\_sold) AS actual\_sold\_quantity,

(SUM(fs.quantity\_sold) / SUM(fm.forecasted\_quantity)) \* 100 AS forecast\_accuracy

FROM

fact\_forecast\_monthly fm

JOIN

fact\_sales\_monthly fs ON fm.product\_id = fs.product\_id

JOIN

dim\_product p ON p.product\_id = fm.product\_id

WHERE

fm.fiscal\_year = 2023 -- Replace with fiscal year

GROUP BY p.product\_name, MONTH(sale\_date)

ORDER BY month;