# Supply Chain Finance Management

**Introduction**

This project is focused on designing and implementing a database management system for sales, inventory, and cost tracking in a business environment. The goal is to efficiently track the movement of products, analyse sales data, calculate manufacturing and freight costs, and manage inventory levels. The system utilizes SQL queries, stored procedures, triggers, and user-defined functions to automate and optimize various business processes, providing insights into key performance indicators (KPIs) such as sales, profit margins, inventory turnover, and more.

The data model is based on a relational structure, connecting multiple tables such as products, sales, customers, cost factors, and freight costs. This enables complex analysis and reporting and facilitates inventory management and cost optimization.

**ERD Explanation**

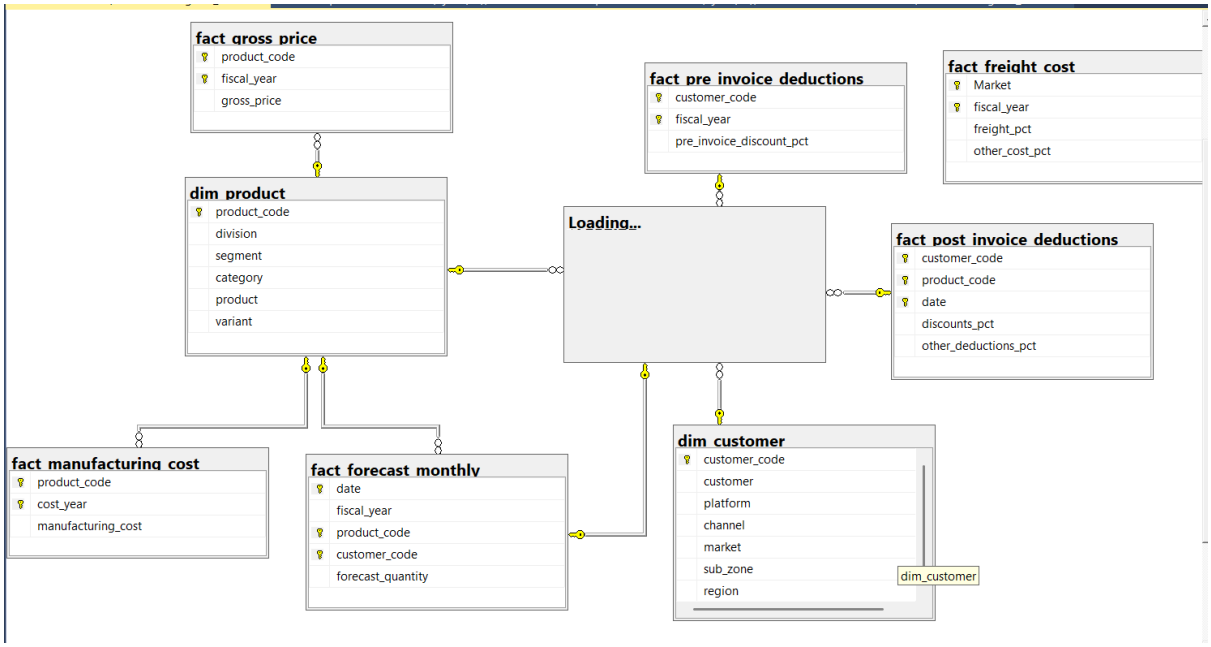
The **Entity-Relationship Diagram (ERD)** represents the database schema and the relationships between the various entities involved in the sales and inventory management system. The key tables in the system are:

1. **dim_product**: Contains product-related details, including product code, name, category, and variant.
2. **fact_sales_monthly**: Tracks monthly sales data, such as the quantity sold, fiscal year, and product code.
3. **fact_forecast_monthly**: Contains forecasted sales data for each product, allowing comparisons with actual sales.
4. **fact_gross_price**: Holds the gross prices of products for each fiscal year, used to calculate revenue.
5. **fact_manufacturing_cost**: Contains manufacturing costs for each product.
6. **fact_freight_cost**: Tracks freight costs by market and fiscal year.
7. **dim_customer**: Contains customer-related information, such as customer code, market, and platform.
8. **fact_post_invoice_deductions** and **fact_pre_invoice_deductions**: Handle invoice-related deductions and discounts.

**Relationships**:

- **dim_product** is connected to **fact_sales_monthly**, **fact_forecast_monthly**, **fact_gross_price**, and **fact_manufacturing_cost** via the product_code.
- **fact_sales_monthly** is linked to **dim_customer** by the customer_code, enabling sales tracking per customer.
- **fact_freight_cost** and **fact_manufacturing_cost** are connected to the sales data for calculating costs based on fiscal year and product.
- **fact_post_invoice_deductions** and **fact_pre_invoice_deductions** handle product-level and customer-level financial adjustments.

The relationships are designed to facilitate the accurate and efficient flow of data between tables, ensuring that key business operations such as sales tracking, inventory management, and cost calculations can be automated and analyzed in real-time.


fact gross price
product_code
fiscal_year
gross_price
fact pre invoice deductions
customer_code
fiscal_year
pre_invoice_discount_pct
fact freight cost
Market
fiscal_year
freight_pct
other_cost_pct
dim product
product_code
division
segment
category
product
variant
Loading...
fact post invoice deductions
customer_code
product_code
date
discounts_pct
other_deductions_pct
fact manufacturing cost
product_code
cost_year
manufacturing_cost
fact forecast monthly
date
fiscal_year
product_code
customer_code
forecast_quantity
dim customer
customer_code
customer
platform
channel
market
sub_zone
region
dim_customer

## Task-3

## Q1) Assume calendar_date is '2023-07-15'. Apply the function to this date and explain what value it will return as the fiscal year.

## select year(date) as fiscal_year from fact_sales_monthly

```
update fact_sales_monthly
set fiscal_year = YEAR(date)
```

## Q2) Analyzing Gross Sales: Monthly Product Transactions Report

## Write a Query for making report on monthly product transactions, including details such as date, product code,

## product name, variant, sold quantity, gross price, and gross price total. The query should involves joining several tables and

## filtering results based on customer code and fiscal year.

```
select
p.product_code,
p.product,
p.variant,
fsm.sold_quantity,
fgp.gross_price,
(fsm.sold_quantity * fgp.gross_price) as gross_price_total
from fact_sales_monthly fsm
join dim_product p on p.product_code = fsm.product_code
join fact_forecast_monthly ffm on ffm.product_code = p.product_code
join fact_gross_price fgp on fgp.product_code = p.product_code
join dim_customer c on c.customer_code = ffm.customer_code
```

## Task-4

## Sales Trend Analysis:

## Query the fact_monthly_sales table to identify the monthly sales trend for each product. How do the sales volumes fluctuate over time?

```
select p.product, p.variant ,DATENAME(MONTH, fsm.date) as month, year(date) as year, sum(fsm.sold_quantity) as
total_sold_quantity from fact_sales_monthly fsm
join dim_product p on p.product_code = fsm.product_code
group by p.product, p.variant, DATENAME(MONTH, fsm.date), year(date)
order by  year,month asc, total_sold_quantity desc
```

## Customer Segmentation:

## Utilizing the dim_customer table, segment customers based on their purchasing behavior. Which customer segments contribute the most to sales revenue?

```
select c.channel,c.region,c.market,round(SUM(fsm.sold_quantity * fgp.gross_price),2) as revenue from
dim_customer c
join fact_sales_monthly fsm on fsm.customer_code = c.customer_code
join fact_gross_price fgp on fgp.product_code = fsm.product_code
group by c.channel,c.region,c.market
```

## Product Performance Comparison:

## Compare the performance of products in terms of sales quantity and revenue generated. Which products are the top performers, and which ones need improvement?

```
select p.product,p.variant,SUM(fsm.sold_quantity) AS total_sold_quantity,round(cast(SUM(fsm.sold_quantity *
fgp.gross_price) as float),2) as revenue from dim_product p
```

```
join fact_sales_monthly fsm on p.product_code = fsm.product_code
join fact_gross_price fgp on fgp.product_code = fsm.product_code
group by p.product,p.variant
order by revenue desc, total_sold_quantity desc
```

## Market Expansion Opportunities:

## Analyze the fact_forecast_monthly table to identify potential market expansion opportunities. Which markets show the highest forecasted demand growth?

```
select c.market,ffm.fiscal_year, SUM(ffm.forecast_quantity) as forcast_quantity from dim_customer c
join fact_forecast_monthly ffm on ffm.customer_code =c.customer_code
group by c.market,ffm.fiscal_year
order by ffm.fiscal_year, forcast_quantity desc
```

## Cost Analysis:

## Calculate the total manufacturing cost for each product and compare it with the gross price to determine profitability. Which products have the highest profit margins?

```
select p.product,p.variant, fmc.cost_year,fgp.gross_price,fmc.manufacturing_cost,
(fgp.gross_price-fmc.manufacturing_cost) as profit_per_unit,
round(cast((fgp.gross_price-fmc.manufacturing_cost)/fgp.gross_price * 100 as float),2) as profit_percentage
from dim_product p
join fact_manufacturing_cost fmc on fmc.product_code = p.product_code
join fact_gross_price fgp on fgp.product_code = p.product_code
order by profit_percentage desc
```

## Discount Impact Analysis:

## Assess the impact of pre-invoice discounts on sales revenue.

## How do varying discount levels affect overall revenue and customer retention?

```
with revenue_discount as (
select c.customer,
c.customer_code,fsm.fiscal_year,p.product,p.product_code,p.variant,fsm.sold_quantity,fpid.pre_invoice_discount_p
ct,
round(cast( (fsm.sold_quantity * fgp.gross_price) as float),2) as Total_revenue,
round(cast((fsm.sold_quantity* fgp.gross_price * (1 - fpid.pre_invoice_discount_pct)) as float),2) as
Revenue_after_discount
from dim_customer c
join fact_sales_monthly fsm on fsm.customer_code = c.customer_code
join fact_pre_invoice_deductions fpid on fpid.customer_code = c.customer_code
join dim_product p on p.product_code = fsm.product_code
join fact_gross_price fgp on  fgp.product_code = p.product_code
),
customerRetention as(
select customer_code, count(distinct fiscal_year) as retention_years from fact_sales_monthly
group by customer_code
)
select rd.customer,rd.fiscal_year, cr.retention_years ,
SUM(rd.Total_revenue) as total_gross_revenue,
SUM(rd.Revenue_after_discount) as total_revenue_after_discount,
avg(rd.pre_invoice_discount_pct) * 100 as avg_dis_percentage
from  customerRetention cr
join revenue_discount rd on rd.customer_code = cr.customer_code
group by rd.customer,rd.fiscal_year,cr.retention_years
```

## Market-specific Freight Costs:

## Determine the average freight costs for different markets over the years. Are there any noticeable trends or outliers in freight expenses?

```
select market, fiscal_year, round(cast(AVG(freight_pct)*100 as float),0)  as avg_freight_cost_pct,
round(cast(AVG(other_cost_pct)*100 as float),0)  as avg_other_cost_pct,round(cast(AVG(freight_pct+other_cost_pct)
*100 as float),0) as avg_total_cost_pct
from fact_freight_cost
group by Market,fiscal_year
```

## Seasonal Sales Patterns:

## Explore the fact_monthly_sales table to identify seasonal sales patterns. How do sales volumes vary throughout the year, and are there any recurring trends?

```
select DATENAME(MONTH, date) AS month_name ,fiscal_year,
SUM(sold_quantity) as total_sales_volume
from fact_sales_monthly
group by fiscal_year, DATENAME(MONTH, date);
```

## Customer Loyalty Analysis:

## Analyze customer purchase frequency and retention rates over time. Which customers exhibit the highest levels of loyalty, and how can their behavior be leveraged for targeted marketing campaigns?

```
with CustomerPurchaseFrequency  as (
select customer_code, COUNT(distinct date) as Purchase_frequency, MIN(date) as first_purchase, MAX(date) as
last_purchase,
DATEDIFF(DAY,MIN(date),MAX(date)) as customer_life_span_days, COUNT(distinct fiscal_year) as active_years
from fact_sales_monthly
group by customer_code
)
select c.customer,c.customer_code,c.market,cpf.Purchase_frequency,cpf.active_years
from dim_customer  c
join CustomerPurchaseFrequency cpf on cpf.customer_code = c.customer_code
order by cpf.purchase_frequency DESC;
```

## Forecast Accuracy Evaluation:

## Evaluate the accuracy of sales forecasts by comparing forecasted quantities with actual sales data. Are there any significant discrepancies, and how can forecast models be improved?

```
with ForecastComparison  as(
select ffm.date, ffm.product_code, p.product, ffm.customer_code, c.customer, ffm.forecast_quantity,
coalesce(SUM(fsm.sold_quantity),0) as total_sold_quantity,
(coalesce(SUM(fsm.sold_quantity),0) -ffm.forecast_quantity) as forecast_error
from fact_forecast_monthly ffm
left join fact_sales_monthly fsm on ffm.customer_code = fsm.customer_code and ffm.date = fsm.date and
ffm.product_code = fsm.product_code
join dim_customer c on c.customer_code = ffm.customer_code
join dim_product p on p.product_code = ffm.product_code
group by ffm.date, ffm.product_code, p.product, ffm.customer_code, c.customer, ffm.forecast_quantity
),
ForecastError as(
select product, product_code,customer_code, customer,
round(SUM(case
            when total_sold_quantity >0 then ABS(cast(forecast_error as float))/total_sold_quantity
            else null
            end)/count(*) * 100,2) as error_pct
from ForecastComparison
group by product, product_code,customer_code, customer
)
select fc.product, fc.product_code,fc.customer_code,
fc.customer,fc.total_sold_quantity,fc.forecast_quantity,fc.forecast_error, fe.error_pct
from ForecastComparison fc
join ForecastError fe on fc.customer_code=fe.customer_code and fc.product_code = fe.product_code
```

## Channel Performance Assessment:

## Compare sales performance across different sales channels (e.g., E-Commerce vs. Brick & Mortar). Which channels are most effective in driving sales, and are there any opportunities for

## optimization?

```
select c.channel as sales_channel, count(distinct fsm.customer_code) as total_customers, SUM(fsm.sold_quantity) as total_sold_units,
cast(SUM(fsm.sold_quantity * fgp.gross_price)as float) as total_revenue,
round(cast(SUM(fsm.sold_quantity * fgp.gross_price)as float)/SUM(fsm.sold_quantity),2) as avg_price_per_unit
from dim_customer c
join fact_sales_monthly fsm on fsm.customer_code = c.customer_code
join fact_gross_price fgp on fgp.product_code = fsm.product_code and fgp.fiscal_year = fsm.fiscal_year
group by c.channel;
```

## Geographical Sales Distribution:

## Analyze sales distribution across different geographical regions. How does sales performance vary by region, and are there any emerging markets worth focusing on?

```
with TotalSales as (
select SUM(fsm.sold_quantity * fgp.gross_price) as total_revenue,
SUM(fsm.sold_quantity) as total_sold_unit
from fact_sales_monthly fsm
join fact_gross_price fgp on fgp.fiscal_year = fsm.fiscal_year and fgp.product_code = fsm.product_code
)
select c.region,c.market,SUM(fsm.sold_quantity) as total_sold_units, SUM(fsm.sold_quantity * fgp.gross_price) as total_revenue,
round(cast(SUM(fsm.sold_quantity * fgp.gross_price) as float) * 100.0 / (select total_revenue from totalSales),2) as revenue_pct,
round(cast(SUM(fsm.sold_quantity)as float)  *100.0/(select total_sold_unit from totalSales),2) as quantity_pct
from dim_customer c
join fact_sales_monthly fsm on fsm.customer_code = c.customer_code
join fact_gross_price fgp on fgp.product_code = fsm.product_code and fsm.fiscal_year = fgp.fiscal_year
group by c.region,c.market
order by total_revenue desc
```

## Customer Acquisition Cost Analysis:

## Calculate the customer acquisition cost (CAC) for each market and channel. Which acquisition channels provide the highest return on investment (ROI), and where should resources be

## allocated for customer acquisition?

```
with customer_first_purchase as(
select customer_code, min(date) as first_purchase from fact_sales_monthly
group by customer_code
),
new_customer as(
select c.customer_code,c.market,c.channel from dim_customer c
join customer_first_purchase cfp on c.customer_code = cfp.customer_code
),
new_customer_revenue as (
select nc.market,nc.channel,
SUM(fsm.sold_quantity * fgp.gross_price) AS revenue_by_new_customers,
COUNT(nc.customer_code) as new_customer_count
from new_customer nc
join fact_sales_monthly fsm on fsm.customer_code = nc.customer_code
```

```
join fact_gross_price fgp on fgp.product_code = fsm.product_code and fsm.fiscal_year = fgp.fiscal_year
group by nc.market, nc.channel
)
select market,channel,revenue_by_new_customers, new_customer_count,
ROUND(cast(revenue_by_new_customers * 0.2 as float), 2) AS estimated_acquisition_cost,  Assume 20% of revenue
for acquisition
ROUND(cast(revenue_by_new_customers * 0.2 / new_customer_count as float), 2) AS customer_acquisition_cost,
ROUND(cast((revenue_by_new_customers - (revenue_by_new_customers * 0.2)) * 100.0 as float) /
(revenue_by_new_customers * 0.2), 2) AS roi_percentage
from new_customer_revenue
order by roi_percentage desc
```

## Product Mix Optimization:

## Determine the optimal product mix based on sales volume, profitability, and market demand.

## How can product portfolios be adjusted to maximize overall revenue?

```
with productSales as(
select p.product_code, p.product, p.variant,fsm.fiscal_year ,c.market,c.region,c.channel ,
round(cast(SUM(fsm.sold_quantity)as float),2) as total_sold_quantity,
round(cast(SUM(fsm.sold_quantity * fgp.gross_price)as float),2) AS total_revenue,
round(cast(SUM(fsm.sold_quantity * fmc.manufacturing_cost)as float),2) AS total_cost
from dim_product p
join fact_sales_monthly fsm on fsm.product_code = p.product_code
join fact_gross_price fgp on fgp.product_code = p.product_code and fgp.fiscal_year = fsm.fiscal_year
join fact_manufacturing_cost fmc on fmc.product_code = p.product_code and fmc.cost_year = fsm.fiscal_year
join dim_customer c on c.customer_code = fsm.customer_code
group by p.product_code, p.product, p.variant,fsm.fiscal_year ,c.market,c.region,c.channel
)
Select product_code , product,fiscal_year,variant,total_sold_quantity,total_revenue,total_cost
,region,market,channel ,
(total_revenue - total_cost) AS total_profit,
ROUND(cast((total_revenue - total_cost) * 100.0 / total_revenue as float), 2) AS profit_margin
FROM ProductSales
order by fiscal_year,total_profit desc
```

## Customer Lifetime Value Calculation:

## Calculate the customer lifetime value (CLV) for each customer segment. Which segments are the most valuable in terms of long-term revenue generation?

```
with customer_revenue as (
select c.customer,c.customer_code,c.market,c.region,
round(cast(SUM(fsm.sold_quantity * fgp.gross_price)as float),2) AS total_revenue,
COUNT(DISTINCT YEAR(fsm.date)) AS active_years
from dim_customer c
join fact_sales_monthly fsm on fsm.customer_code = c.customer_code
join fact_gross_price fgp on fgp.product_code = fsm.product_code and fgp.fiscal_year =fsm.fiscal_year
GROUP BY c.customer,c.customer_code, c.market, c.region
)
select market,region,
round(cast(avg(total_revenue) as float),2) as Average_revenue_per_cus,
ROUND(cast(sum(total_revenue) as float),2) as Total_market_revenue,
ROUND(AVG(total_revenue / NULLIF(active_years, 0)), 2) AS avg_revenue_per_year,
COUNT(DISTINCT customer_code) AS total_customers
from customer_revenue
GROUP BY market, region
ORDER BY total_market_revenue DESC, Average_revenue_per_cus DESC;
```

## Task-5

### 1. Define a user-defined function to calculate the total forecasted quantity for a given product and fiscal year.

```
CREATE FUNCTION fn_total_forecast_quantity (
   @product_id varchar(10),       Product ID for which forecast is calculated
   @fiscal_year INT       Fiscal year to filter the forecasted data
)
returns int
as
begin
declare @total_forecast int;
select @total_forecast = SUM(forecast_quantity)
from fact_forecast_monthly
where product_code = @product_id and fiscal_year= @fiscal_year
RETURN @total_forecast;
END;
select dbo.fn_total_forecast_quantity('P001', 2022) as forcasted_result
```

### 2. Write a query to find the customers who made purchases exceeding the average monthly sales quantity across all products.

```
with avg_montly_sales_cte as(
select avg(monthly_sales) as avg_monthly_sales from
(select fiscal_year, MONTH(date) as month, SUM(sold_quantity) as monthly_sales from fact_sales_monthly group by
fiscal_year, MONTH(date)) AS MonthlySales)
select c.customer,c.customer_code,
SUM(fsm.sold_quantity) as total_purchase_quantity,
(select avg_monthly_sales from avg_montly_sales_cte) as average_monthly_sales
from dim_customer c
join fact_sales_monthly fsm on fsm.customer_code = c.customer_code
group by c.customer,c.customer_code
having SUM(fsm.sold_quantity) > (select avg_monthly_sales from avg_montly_sales_cte)
```

### 3. Create a stored procedure to update the gross price of a product for a specific fiscal year.

```
CREATE PROCEDURE UpdateGrossPrice
@product_id VARCHAR(10),
@fiscal_year INT,
@new_gross_price DECIMAL(18, 2)
AS
begin
update fact_gross_price
set gross_price = @new_gross_price
where fiscal_year = @fiscal_year and product_code = @product_id
end;
EXEC UpdateGrossPrice @product_id = 'P001', @fiscal_year = 2022, @new_gross_price = 150.75;
```

### 4. Implement a trigger that automatically inserts a record into the audit log table whenever a new entry is added to the sales table.

### 5. Use a window function to rank products based on their monthly sales quantity, partitioned by fiscal year.

```
select p.product,p.variant ,p.product_code,fsm.fiscal_year ,sum(fsm.sold_quantity)as  total_sold_quantity,
rank() over ( partition by fsm.fiscal_year order by sum(fsm.sold_quantity) desc) as rank
from dim_product p
join fact_sales_monthly fsm on fsm.product_code = p.product_code
group by p.product,p.variant ,p.product_code, fsm.fiscal_year
```

### 6. Utilize the STRING_AGG function to concatenate the names of all customers who purchased a specific product within a given timeframe.

```
select p.product, STRING_AGG(c.customer,',') as customers from dim_customer c
```

```
join fact_sales_monthly fsm on fsm.customer_code = c.customer_code
join dim_product p on p.product_code = fsm.product_code
where p.product_code = 'P002' and date between '2022-01-01' and '2022-01-04'
group by p.product
```

## 7. Develop a user-defined function that calculates the total manufacturing cost for a product over a specified range of years, using a subquery to retrieve the necessary data.

```
CREATE FUNCTION fn_total_manufacturing_cost (
  @product_code VARCHAR(10),
  @start_year INT,
  @end_year INT
)
RETURNS DECIMAL(10,2)
AS
BEGIN
DECLARE @total_cost DECIMAL(10,2);

SELECT @total_cost = SUM(manufacturing_cost)
FROM fact_manufacturing_cost
WHERE product_code = @product_code
AND cost_year BETWEEN @start_year AND @end_year;

RETURN @total_cost;
END;
SELECT dbo.fn_total_manufacturing_cost('P001', 2020, 2024) AS total_cost;
```

## 8. Design a stored procedure to insert new records into the sales table and use a trigger to enforce constraints on the quantity sold, ensuring it doesn't exceed the available inventory.

## 9. Apply the LEAD or LAG function to compare monthly sales quantities of a product with the previous month's sales.

```
SELECT fsm.fiscal_year,
MONTH(fsm.date) AS month,
fsm.product_code,
SUM(fsm.sold_quantity) AS current_month_sales,
LAG(SUM(fsm.sold_quantity), 1) OVER (PARTITION BY fsm.product_code ORDER BY fsm.fiscal_year,
MONTH(fsm.date)) AS previous_month_sales,
SUM(fsm.sold_quantity) - LAG(SUM(fsm.sold_quantity), 1) OVER (PARTITION BY fsm.product_code ORDER BY
fsm.fiscal_year, MONTH(fsm.date)) AS sales_diff
FROM fact_sales_monthly fsm
GROUP BY fsm.fiscal_year,MONTH(fsm.date),fsm.product_code
ORDER BY fsm.product_code, fsm.fiscal_year, MONTH(fsm.date);
```

## 10. Create a query to identify the top-selling products in each market based on their total sales quantity, utilizing subqueries and window functions.

```
select product_code, product, variant, market, total_sales_quantity, sales_rank
from (
select p.product_code, p.product, p.variant, c.market,
sum(fsm.sold_quantity) as total_sales_quantity,
rank() over (partition by c.market order by sum(fsm.sold_quantity) desc) as sales_rank
from dim_product p
join fact_sales_monthly fsm on p.product_code = fsm.product_code
join dim_customer c on c.customer_code = fsm.customer_code
group by p.product_code, p.product, p.variant, c.market
) as ranked_products
where sales_rank = 1
```

```
order by market, sales_rank;
```

## 11. Develop a user-defined function to calculate the total freight cost for a product based on its market and fiscal year. Then, integrate this function into a stored procedure to

## update the overall cost.

```
create function dbo.fn_total_freight_cost (
@market varchar(45),
@fiscal_year int
)
returns decimal(18, 4)
as
begin
declare @total_freight_cost decimal(18, 4);
select @total_freight_cost = sum(freight_pct + other_cost_pct)
from fact_freight_cost
where market = @market
and fiscal_year = @fiscal_year;
return isnull(@total_freight_cost, 0);
end;

create procedure dbo.sp_update_total_cost
@product_code varchar(50),
@market varchar(45),
@fiscal_year int,
@base_cost decimal(18, 4)
as
begin
declare @freight_cost decimal(18, 4);
declare @total_cost decimal(18, 4);
set @freight_cost = dbo.fn_total_freight_cost(@market, @fiscal_year);
set @total_cost = @base_cost + @freight_cost;
update fact_manufacturing_cost
set manufacturing_cost = @total_cost
where product_code = @product_code
and cost_year = @fiscal_year;
select @total_cost as updated_total_cost;
end;

exec dbo.sp_update_total_cost 'P001', 'north america', 2024, 1200.00;

select * from fact_manufacturing_cost
where product_code = 'P001' and cost_year = 2024;
```