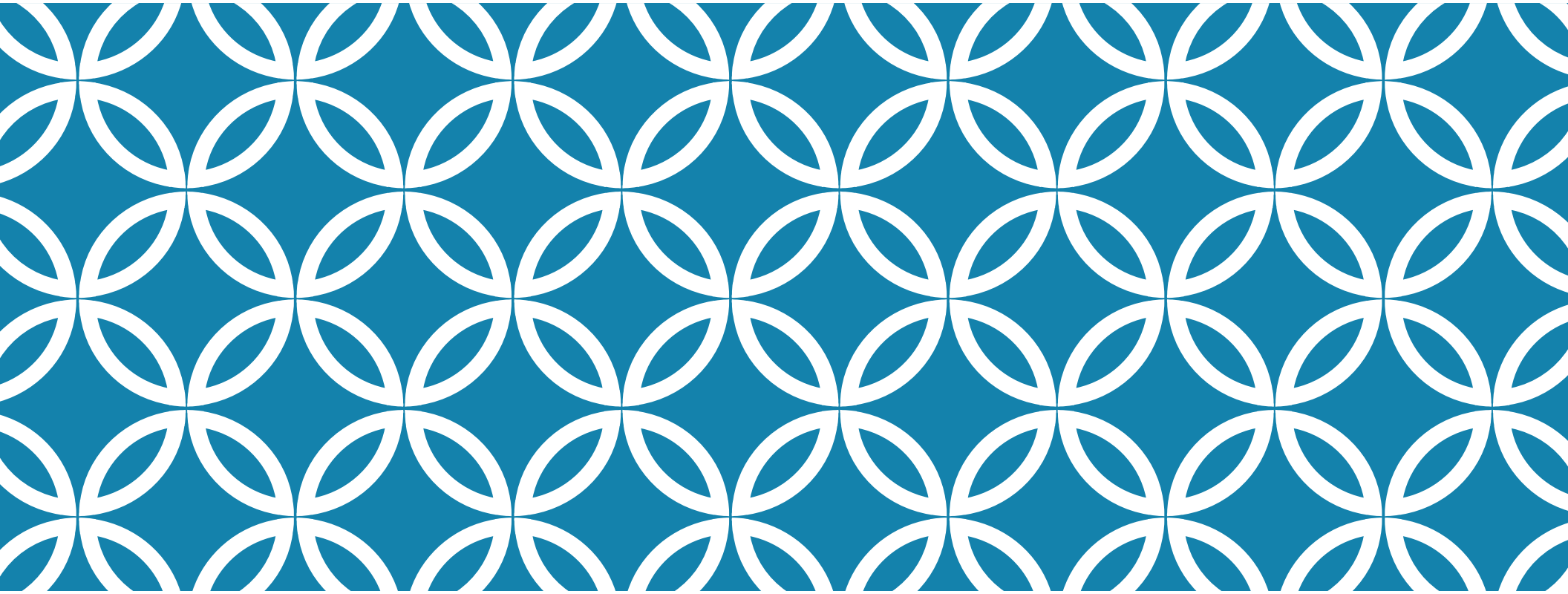


# SQL CASE STUDY

Extracting data from toy store chain inventory structure

# SQL CASE STUDY

This is the data from a Toy store chain with multiple locations in Barcelona, Spain.

We have access to data containing transactional records from January 2022 till Sept 2023, along with the information about products and store locations.

# ANALYSIS BY USING SQL

- **1. Production Performance Analysis**
- **Goal**- Identify top-performing products based on total sales and profit.

**2. Store Performance Analysis**

- **Goal**- Analyse sales performance for each store, including total revenue and profit margin.

**3. Complex Monthly Sales Trend Analysis:**

**Goal**- Examine monthly sales trends, considering the rolling 3-month average and identifying months with significant growth or decline.

**4. Cumulative Distribution of Profit Margin:**

**Goal-** Calculate the cumulative distribution of profit margin for each product category, consider where products are having profit.

**5. Store Inventory Turnover Analysis:**

**Goal:** Analyze the efficiency of inventory turnover for each store by calculating the Inventory Turnover Ratio.

# DATASET/S STRUCTURE

Tables- 5 nos

Table Names- Stores- 50 rows,

Sales- 8 lakh+ rows

Products- 36 rows

Calendar- 639 rows

Inventory- 1539 rows

# DATA DESCRIPTION

| Table | Field | Description |
|---|---|---|
| Products | Product_ID | Product ID |
| Products | Product_Name | Product name |
| Products | Product_Category | Product Category |
| Products | Product_Cost | Product cost ($USD) |
| Products | Product_Price | Product retail price ($USD) |
| Inventory | Store_ID | Store ID |
| Inventory | Product_ID | Product ID |
| Inventory | Stock_On_Hand | Stock quantity of the product in the store (inventory) |
| Stores | Store_ID | Store ID |
| Stores | Store_Name | Store name |
| Stores | Store_City | City in Mexico where the store is located |
| Stores | Store_Location | Location in the city where the store is located |
| Stores | Store_Open_Date | Date when the store was opened |
| Sales | Sale_ID | Sale ID |
| Sales | Date | Date of the transaction |
| Sales | Store_ID | Store ID |
| Sales | Product_ID | Product ID |
| Sales | Units | Units sold |
| Calendar | Date | Calendar date |

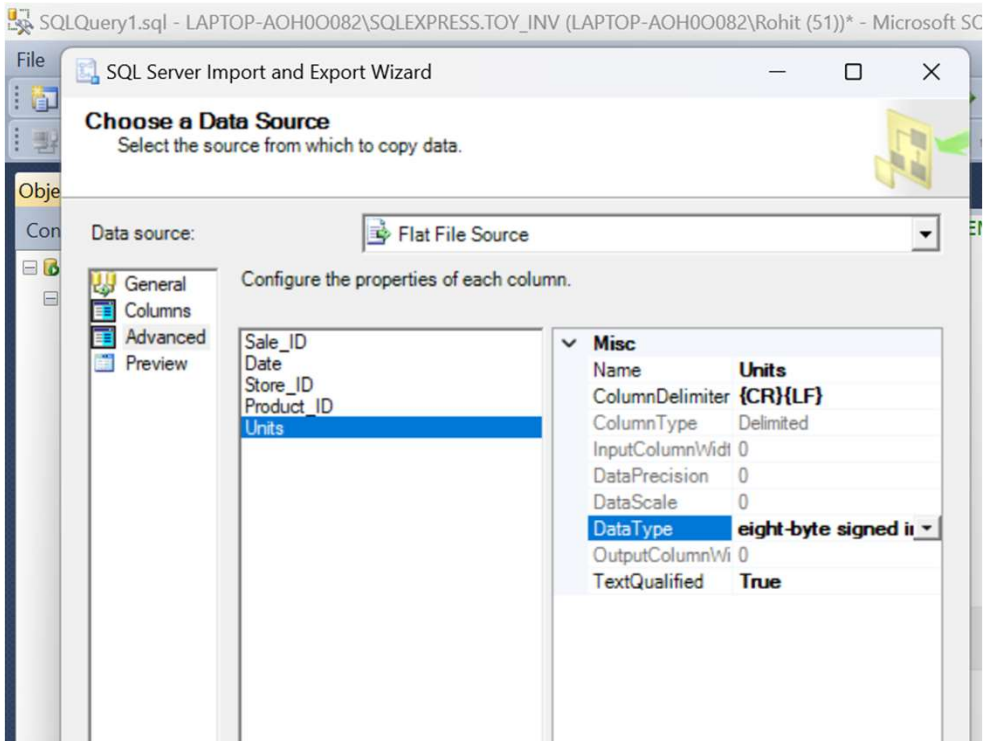
# IMPORT DATA TO SQL

Create database – **TOY_INVENTORY**

Import all 5 tables to this database . (Data Source – Flat File Source (csv))

Set data type for columns here

```
CREATE DATABASE TOY_INVENTORY;

USE TOY_INVENTORY;

-- load all tables to this database

SELECT * from INVENTORY;

SELECT * FROM PRODUCTS;

SELECT * FROM STORES;

SELECT * FROM SALES;
```



# ANALYSIS IN SQL

> Alter datatypes of various columns as needed to have tables in proper format
>
> Products cost , price – string → money → float

```sql
SELECT * FROM SALES;

SELECT TABLE_NAME,COLUMN_NAME,DATA_TYPE FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME='SALES';

ALTER TABLE SALES
ALTER COLUMN "Date" DATE;
```

100 %

Results | Messages

| | TABLE_NAME | COLUMN_NAME | DATA_TYPE |
|---|---|---|---|
| 1 | sales | Sale_ID | int |
| 2 | sales | Date | datetime |
| 3 | sales | Store_ID | int |
| 4 | sales | Product_ID | int |
| 5 | sales | Units | int |

```sql
/**** PRODUCTS TABLE  *********/

SELECT * FROM PRODUCTS;

SELECT TABLE_NAME,COLUMN_NAME,DATA_TYPE FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME='PRODUCTS';

--need to change data type of product cost and product price

ALTER TABLE PRODUCTS
ALTER COLUMN PRODUCT_COST MONEY;
ALTER TABLE PRODUCTS
ALTER COLUMN PRODUCT_COST FLOAT;

ALTER TABLE PRODUCTS
ALTER COLUMN PRODUCT_PRICE MONEY;
ALTER TABLE PRODUCTS
ALTER COLUMN PRODUCT_PRICE FLOAT;
```

100 %

Results | Messages

| | TABLE_NAME | COLUMN_NAME | DATA_TYPE |
|---|---|---|---|
| 1 | products | Product_ID | int |
| 2 | products | Product_Name | varchar |
| 3 | products | Product_Category | varchar |
| 4 | products | Product_Cost | float |
| 5 | products | Product_Price | float |
| 6 | products | PROFIT | float |

# ANALYSIS IN SQL

Alter datatypes of various columns as needed to have tables in proper format

```
SELECT * FROM PRODUCTS;

ALTER TABLE PRODUCTS
ADD PROFIT FLOAT;

UPDATE PRODUCTS SET PROFIT=PRODUCT_PRICE-PRODUCT_COST;
```

100 %

Results | Messages

| | Product_ID | Product_Name | Product_Category | Product_Cost | Product_Price | PROFIT |
|---|---|---|---|---|---|---|
| 1 | 1 | Action Figure | Toys | 9.99 | 15.99 | 6 |
| 2 | 2 | Animal Figures | Toys | 9.99 | 12.99 | 3 |
| 3 | 3 | Barrel O' Slime | Art & Crafts | 1.99 | 3.99 | 2 |
| 4 | 4 | Chutes & Ladders | Games | 9.99 | 12.99 | 3 |

# ANALYSIS IN SQL

All tables in proper format now.

```
SELECT * FROM STORES;
```

100 %

Results | Messages

| | Store_ID | Store_Name | Store_City | Store_Location | Store_Open_Date |
|---|---|---|---|---|---|
| 1 | 1 | Toys Guadalajara 1 | Guadalajara | Residential | 1992-09-18 |
| 2 | 2 | Toys Monterrey 1 | Monterrey | Residential | 1995-04-27 |
| 3 | 3 | Toys Guadalajara 2 | Guadalajara | Commercial | 1999-12-27 |
| 4 | 4 | Toys Saltillo 1 | Saltillo | Downtown | 2000-01-01 |

```
SELECT * FROM PRODUCTS;
```

100 %

Results | Messages

| | Product_ID | Product_Name | Product_Category | Product_Cost | Product_Price | PROFIT |
|---|---|---|---|---|---|---|
| 1 | 1 | Action Figure | Toys | 9.99 | 15.99 | 6 |
| 2 | 2 | Animal Figures | Toys | 9.99 | 12.99 | 3 |
| 3 | 3 | Barrel O' Slime | Art & Crafts | 1.99 | 3.99 | 2 |
| 4 | 4 | Chutes & Ladders | Games | 9.99 | 12.99 | 3 |
| 5 | 5 | Classic Dominoes | Games | 7.99 | 9.99 | 2 |

```
SELECT * FROM SALES;
```

100 %

Results | Messages

| | Sale_ID | Date | Store_ID | Product_ID | Units |
|---|---|---|---|---|---|
| 1 | 3921 | 2022-01-05 | 37 | 18 | 1 |
| 2 | 3922 | 2022-01-05 | 48 | 24 | 1 |
| 3 | 3923 | 2022-01-05 | 45 | 6 | 1 |
| 4 | 3924 | 2022-01-05 | 39 | 30 | 1 |
| 5 | 3925 | 2022-01-05 | 26 | 27 | 1 |

```
SELECT * from INVENTORY;
```

100 %

Results | Messages

| | Store_ID | Product_ID | Stock_On_Hand |
|---|---|---|---|
| 1 | 1 | 1 | 27 |
| 2 | 1 | 2 | 0 |
| 3 | 1 | 3 | 32 |
| 4 | 1 | 4 | 6 |

# ANALYSIS IN SQL

- **1. Production Performance Analysis**
- **Goal**- Identify top-performing products based on total sales and profit.

| Product_ID | Product_Name | Product_Category | Product_Cost | Product_Price | PROFIT |
|---|---|---|---|---|---|
| 1 | Action Figure | Toys | 9.99 | 15.99 | 6 |
| 2 | Animal Figures | Toys | 9.99 | 12.99 | 3 |
| 3 | Barrel O' Slime | Art & Crafts | 1.99 | 3.99 | 2 |

| Sale_ID | Date | Store_ID | Product_ID | Units |
|---|---|---|---|---|
| 3921 | 2022-01-05 | 37 | 18 | 1 |
| 3922 | 2022-01-05 | 48 | 24 | 1 |
| 3923 | 2022-01-05 | 45 | 6 | 1 |

```sql
-- Top performing products based on total profit earned


CREATE VIEW PRODUCT_DET
AS
SELECT T1.PRODUCT_ID,T1.PRODUCT_NAME,T1.PRODUCT_PRICE,T1.PROFIT,SUM(T2.UNITS) AS 'Total Units sold'
from PRODUCTS T1
INNER JOIN SALES T2
ON T1.PRODUCT_ID=T2.PRODUCT_ID
GROUP BY T1.PRODUCT_ID,T1.PRODUCT_NAME,T1.PRODUCT_PRICE,T1.PROFIT;

SELECT * FROM PRODUCT_DET
ORDER BY PROFIT*[Total Units sold] DESC;
```

100 %

Results | Messages

| | PRODUCT_ID | PRODUCT_NAME | PRODUCT_PRICE | PROFIT | Total Units sold |
|---|---|---|---|---|---|
| 1 | 6 | Colorbuds | 14.99 | 8 | 104368 |
| 2 | 1 | Action Figure | 15.99 | 6 | 57958 |
| 3 | 18 | Lego Bricks | 39.99 | 5 | 59737 |
| 4 | 8 | Deck Of Cards | 6.99 | 3 | 84034 |
| 5 | 14 | Glass Marbles | 10.99 | 5 | 37518 |

# ANALYSIS IN SQL

- **1. Production Performance Analysis**
- **Goal**- Identify top-performing products based on total sales and profit.

```sql
SELECT * FROM PRODUCT_DET
ORDER BY PRODUCT_PRICE*[Total Units sold] DESC;


-- Top performing products based on total sales (other way)

CREATE PROCEDURE TOP_BY_SALES
AS
BEGIN
SELECT T1.PRODUCT_ID,T1.PRODUCT_NAME,T1.PRODUCT_CATEGORY,T1.PRODUCT_PRICE,T1.PROFIT,SUM(T2.UNITS) AS 'Total Units sold'
from PRODUCTS T1
INNER JOIN SALES T2
ON T1.PRODUCT_ID=T2.PRODUCT_ID
GROUP BY T1.PRODUCT_ID,T1.PRODUCT_NAME,T1.PRODUCT_CATEGORY,T1.PRODUCT_PRICE,T1.PROFIT
ORDER BY PRODUCT_PRICE*SUM(T2.UNITS) DESC
END;

TOP_BY_SALES;
```

Results | Messages

| PRODUCT_ID | PRODUCT_NAME | PRODUCT_CATEGORY | PRODUCT_PRICE | PROFIT | Total Units sold |
|---|---|---|---|---|---|
| 18 | Lego Bricks | Toys | 39.99 | 5 | 59737 |
| 6 | Colorbuds | Electronics | 14.99 | 8 | 104368 |
| 19 | Magic Sand | Art & Crafts | 15.99 | 2 | 60598 |
| 1 | Action Figure | Toys | 15.99 | 6 | 57958 |
| 30 | Rubik's Cube | Games | 19.99 | 2 | 45672 |

# ANALYSIS IN SQL

## 2. Store Performance Analysis

- **Goal**- Analyse sales performance for each store, including total revenue and profit margin.

```
SELECT * FROM PRODUCTS;

SELECT * FROM SALES;


--creating a table to track product wise sales of each store


CREATE VIEW STORE_PRODUCTS
AS
SELECT T1.STORE_ID,T1.PRODUCT_ID,T1."Date" AS 'SALES_DATE',UNITS,PRODUCT_COST,PRODUCT_PRICE,PROFIT,
UNITS*PRODUCT_PRICE AS 'TOTAL_SALES',
UNITS*PROFIT AS 'TOTAL_PROFIT'
FROM SALES T1
LEFT JOIN PRODUCTS T2
ON T1.PRODUCT_ID=T2.PRODUCT_ID;

SELECT * FROM STORE_PRODUCTS
ORDER BY SALES_DATE;
```

| Sale_ID | Date | Store_ID | Product_ID | Units |
|---|---|---|---|---|
| 3921 | 2022-01-05 | 37 | 18 | 1 |
| 3922 | 2022-01-05 | 48 | 24 | 1 |
| 3923 | 2022-01-05 | 45 | 6 | 1 |

| Product_ID | Product_Name | Product_Category | Product_Cost | Product_Price | PROFIT |
|---|---|---|---|---|---|
| 1 | Action Figure | Toys | 9.99 | 15.99 | 6 |
| 2 | Animal Figures | Toys | 9.99 | 12.99 | 3 |
| 3 | Barrel O' Slime | Art & Crafts | 1.99 | 3.99 | 2 |

100 %

Results | Messages

| | STORE_ID | PRODUCT_ID | SALES_DATE | UNITS | PRODUCT_COST | PRODUCT_PRICE | PROFIT | TOTAL_SALES | TOTAL_PROFIT |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 24 | 4 | 2022-01-01 | 1 | 9.99 | 12.99 | 3 | 12.99 | 3 |
| 2 | 28 | 1 | 2022-01-01 | 1 | 9.99 | 15.99 | 6 | 15.99 | 6 |
| 3 | 6 | 8 | 2022-01-01 | 1 | 3.99 | 6.99 | 3 | 6.99 | 3 |
| 4 | 48 | 7 | 2022-01-01 | 1 | 11.99 | 15.99 | 4 | 15.99 | 4 |

# ANALYSIS IN SQL

## 2. Store Performance Analysis

- **Goal**- Analyse sales performance for each store, including total revenue and profit margin.

```sql
--each store total revenue and profit margin

SELECT T1.STORE_ID,T1.STORE_NAME,SUM(TOTAL_SALES) AS 'TOTAL REVENUE',SUM(TOTAL_PROFIT) AS 'PROFIT MARGIN'
FROM STORES T1
INNER JOIN STORE_PRODUCTS T2
ON T1.STORE_ID=T2.STORE_ID
GROUP BY T1.STORE_ID,T1.STORE_NAME
ORDER BY STORE_ID;
```

100 %

Results | Messages

| | STORE_ID | STORE_NAME | TOTAL REVENUE | PROFIT MARGIN |
|---|---|---|---|---|
| 1 | 1 | Toys Guadalajara 1 | 261842.889999964 | 69429 |
| 2 | 2 | Toys Monterrey 1 | 277959.139999947 | 73985 |
| 3 | 3 | Toys Guadalajara 2 | 262435.019999957 | 75752 |
| 4 | 4 | Toys Saltillo 1 | 330408.899999918 | 94252 |
| 5 | 5 | Toys La Paz 1 | 210897.829999974 | 57407 |

# ANALYSIS IN SQL

## 3. Complex Monthly Sales Trend Analysis:

**Goal**- Examine monthly sales trends, considering the rolling 3-month average and identifying months with significant growth or decline.

```sql
SELECT * FROM STORE_PRODUCTS;


-- Date wise total sales

CREATE VIEW DATE_WISE
AS
(SELECT SALES_DATE,SUM(TOTAL_SALES) AS 'TOTAL_SALES_EACH_DAY' FROM STORE_PRODUCTS
GROUP BY SALES_DATE);

SELECT * FROM DATE_WISE
ORDER BY SALES_DATE;
```

100 %

Results | Messages

| | STORE_ID | PRODUCT_ID | SALES_DATE | UNITS | PRODUCT_COST | PRODUCT_PRICE | PROFIT | TOTAL_SALES | TOTAL_PROFIT |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 24 | 4 | 2022-01-01 | 1 | 9.99 | 12.99 | 3 | 12.99 | 3 |
| 2 | 28 | 1 | 2022-01-01 | 1 | 9.99 | 15.99 | 6 | 15.99 | 6 |
| 3 | 6 | 8 | 2022-01-01 | 1 | 3.99 | 6.99 | 3 | 6.99 | 3 |
| 4 | 48 | 7 | 2022-01-01 | 1 | 11.99 | 15.99 | 4 | 15.99 | 4 |

100 %

Results | Messages

| | SALES_DATE | TOTAL_SALES_EACH_DAY |
|---|---|---|
| 1 | 2022-01-01 | 21076.1500000002 |
| 2 | 2022-01-02 | 19750.9400000001 |
| 3 | 2022-01-03 | 11759.4999999999 |
| 4 | 2022-01-04 | 14814.3999999999 |

# ANALYSIS IN SQL

## 3. Complex Monthly Sales Trend Analysis:

**Goal**- Examine monthly sales trends, considering the rolling 3-month average and identifying months with significant growth or decline.

```sql
-- MONTH wise total sales and 3 month rolling average of sales

CREATE VIEW MONTHLY_TREND_3
AS
WITH MONTH_SALES1 AS
(SELECT YEAR(SALES_DATE) AS 'SALES_YEAR',MONTH(SALES_DATE) AS 'SALES_MONTH',SUM(TOTAL_SALES_EACH_DAY) AS 'TOTAL_MONTHLY_SALES',
YEAR(SALES_DATE)*12+MONTH(SALES_DATE) AS 'YEARMONTH'
FROM DATE_WISE
GROUP BY MONTH(SALES_DATE),YEAR(SALES_DATE),YEAR(SALES_DATE)*12+MONTH(SALES_DATE)),
MONTH_SALES2 AS(
SELECT SALES_YEAR,SALES_MONTH,YEARMONTH,TOTAL_MONTHLY_SALES,AVG(TOTAL_MONTHLY_SALES) OVER(ORDER BY YEARMONTH ROWS BETWEEN
2 PRECEDING AND CURRENT ROW) AS '3 MONTH AVG' FROM MONTH_SALES1)
SELECT * FROM MONTH_SALES2;

SELECT * FROM MONTHLY_TREND_3;

--TREND ANALYSIS
```

100 %

Results | Messages

| | SALES_YEAR | SALES_MONTH | YEARMONTH | TOTAL_MONTHLY_SALES | 3 MONTH AVG |
|---|---|---|---|---|---|
| 1 | 2022 | 1 | 24265 | 542554.910000001 | 542554.910000001 |
| 2 | 2022 | 2 | 24266 | 541351.650000002 | 541953.280000001 |
| 3 | 2022 | 3 | 24267 | 589485.190000002 | 557797.250000002 |
| 4 | 2022 | 4 | 24268 | 681072.980000007 | 603969.940000003 |

# ANALYSIS IN SQL

## 3. Complex Monthly Sales Trend Analysis:

**Goal**- Examine monthly sales trends, considering the rolling 3-month average and identifying months with significant growth or decline.

```
--TREND ANALYSIS

SELECT SALES_YEAR,SALES_MONTH,TOTAL_MONTHLY_SALES,[3 MONTH AVG],CASE WHEN ([TOTAL_MONTHLY_SALES]-[3 MONTH AVG])>0 THEN '+1'
ELSE '-1'
END AS [TREND]
FROM MONTHLY_TREND_3;
```

100 %

Results | Messages

| | SALES_YEAR | SALES_MONTH | TOTAL_MONTHLY_SALES | 3 MONTH AVG | TREND |
|---|---|---|---|---|---|
| 1 | 2022 | 1 | 542554.910000001 | 542554.910000001 | -1 |
| 2 | 2022 | 2 | 541351.650000002 | 541953.280000001 | -1 |
| 3 | 2022 | 3 | 589485.190000002 | 557797.250000002 | +1 |
| 4 | 2022 | 4 | 681072.980000007 | 603969.940000003 | +1 |

# ANALYSIS IN SQL

## 4. Cumulative Distribution of Profit Margin:

**Goal-** Calculate the cumulative distribution of profit margin for each product category, consider where products are having profit.

| Product_ID | Product_Name | Product_Category | Product_Cost | Product_Price | PROFIT |
|---|---|---|---|---|---|
| 1 | Action Figure | Toys | 9.99 | 15.99 | 6 |
| 2 | Animal Figures | Toys | 9.99 | 12.99 | 3 |
| 3 | Barrel O' Slime | Art & Crafts | 1.99 | 3.99 | 2 |

| Sale_ID | Date | Store_ID | Product_ID | Units |
|---|---|---|---|---|
| 3921 | 2022-01-05 | 37 | 18 | 1 |
| 3922 | 2022-01-05 | 48 | 24 | 1 |
| 3923 | 2022-01-05 | 45 | 6 | 1 |

```sql
--Saving totals data for products to a VIEW

CREATE VIEW PRODUCT_TOTALS
AS
SELECT T1.PRODUCT_ID,T1.PRODUCT_NAME,T1.PRODUCT_CATEGORY,T1.PRODUCT_PRICE,T1.PROFIT,SUM(T2.UNITS) AS 'TOTAL_UNITS'
from PRODUCTS T1
INNER JOIN SALES T2
ON T1.PRODUCT_ID=T2.PRODUCT_ID
GROUP BY T1.PRODUCT_ID,T1.PRODUCT_NAME,T1.PRODUCT_CATEGORY,T1.PRODUCT_PRICE,T1.PROFIT;

SELECT * FROM PRODUCT_TOTALS;
```

100 %

Results | Messages

| | PRODUCT_ID | PRODUCT_NAME | PRODUCT_CATEGORY | PRODUCT_PRICE | PROFIT | TOTAL_UNITS |
|---|---|---|---|---|---|---|
| 1 | 1 | Action Figure | Toys | 15.99 | 6 | 57958 |
| 2 | 2 | Animal Figures | Toys | 12.99 | 3 | 39089 |
| 3 | 3 | Barrel O' Slime | Art & Crafts | 3.99 | 2 | 91663 |
| 4 | 4 | Chutes & Ladders | Games | 12.99 | 3 | 3829 |

# ANALYSIS IN SQL

## 4. Cumulative Distribution of Profit Margin:

**Goal-** Calculate the cumulative distribution of profit margin for each product category, consider where products are having profit.

$$\text{Gross Profit Margin} = \frac{\text{Gross Profit}}{\text{Total Revenue}} \times 100$$

```sql
--category wise total revenue and profit margin

SELECT PRODUCT_CATEGORY,SUM(PRODUCT_PRICE*TOTAL_UNITS) AS 'TOTAL_REVENUE_CATEGORY_WISE' ,
SUM(PROFIT*TOTAL_UNITS) AS 'TOTAL_PROFIT_CATEGORY_WISE',
ROUND(SUM(PROFIT*TOTAL_UNITS)/SUM(PRODUCT_PRICE*TOTAL_UNITS)*100,2) AS 'PROFIT_MARGIN_CATEGORY_WISE'
FROM PRODUCT_TOTALS
GROUP BY PRODUCT_CATEGORY;
```

100 %

Results | Messages

| | PRODUCT_CATEGORY | TOTAL_REVENUE_CATEGORY_WISE | TOTAL_PROFIT_CATEGORY_WISE | PROFIT_MARGIN_CATEGORY_WISE |
|---|---|---|---|---|
| 1 | Art & Crafts | 2705364.26 | 753354 | 27.85 |
| 2 | Electronics | 2246771.25 | 1001437 | 44.57 |
| 3 | Games | 2226836.27 | 673993 | 30.27 |
| 4 | Sports & Outdoors | 2172359.57 | 505718 | 23.28 |

# ANALYSIS IN SQL

**5. Store Inventory Turnover Analysis:**

**Goal:** Analyze the efficiency of inventory turnover for each store by calculating the Inventory Turnover Ratio.

Inventory turnover is a financial ratio showing how many times a company turned over its inventory relative to its cost of goods sold (COGS) in a given period. A company can then divide the days in the period, typically a fiscal year, by the inventory turnover ratio to calculate how many days it takes, on average, to sell its inventory.

$$\text{Inventory Turns} = \frac{\text{COGS}}{\text{Inventory Cost}} = \frac{\text{[Units Sold]} * \text{[Unit Cost]}}{\text{[Units in Stock]} * \text{[Unit Cost]}}$$

# ANALYSIS IN SQL

**5. Store Inventory Turnover Analysis:**

**Goal:** Analyze the efficiency of inventory turnover for each store by calculating the Inventory Turnover Ratio.

```
SELECT * FROM STORE_PRODUCTS;
SELECT * FROM INVENTORY;

CREATE VIEW INV_TURNOVER
AS
WITH STORE_UNITS AS
(SELECT STORE_ID,PRODUCT_ID,SUM(UNITS) AS 'UNITS SOLD'
FROM STORE_PRODUCTS
GROUP BY STORE_ID,PRODUCT_ID),
STORE_UNITS1 AS
(SELECT T1.STORE_ID,T1.PRODUCT_ID,STOCK_ON_HAND,[UNITS SOLD]
FROM INVENTORY AS T1
INNER JOIN STORE_UNITS AS T2
ON T1.Store_ID=T2.STORE_ID AND T1.PRODUCT_ID=T2.PRODUCT_ID)
SELECT STORE_ID,PRODUCT_ID,STOCK_ON_HAND,[UNITS SOLD],ISNULL([UNITS SOLD]*1.0/NULLIF(STOCK_ON_HAND,0),0) AS 'ITR'
FROM STORE_UNITS1;

SELECT STORE_ID,PRODUCT_ID,Stock_On_Hand,[UNITS SOLD],CAST(ITR AS DECIMAL(10,2)) AS 'ITR' FROM INV_TURNOVER;
```

100 %

Results | Messages

| | STORE_ID | PRODUCT_ID | SALES_DATE | UNITS | PRODUCT_COST | PRODUCT_PRICE | PROFIT | TOTAL_SALES | TOTAL_PROFIT |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 24 | 4 | 2022-01-01 | 1 | 9.99 | 12.99 | 3 | 12.99 | 3 |
| 2 | 28 | 1 | 2022-01-01 | 1 | 9.99 | 15.99 | 6 | 15.99 | 6 |
| 3 | 6 | 8 | 2022-01-01 | 1 | 3.99 | 6.99 | 3 | 6.99 | 3 |
| 4 | 48 | 7 | 2022-01-01 | 1 | 11.99 | 15.99 | 4 | 15.99 | 4 |

| | Store_ID | Product_ID | Stock_On_Hand |
|---|---|---|---|
| 1 | 1 | 1 | 27 |
| 2 | 1 | 2 | 0 |
| 3 | 1 | 3 | 32 |

100 %

Results | Messages

| | STORE_ID | PRODUCT_ID | Stock_On_Hand | UNITS SOLD | ITR |
|---|---|---|---|---|---|
| 1 | 9 | 31 | 6 | 2298 | 383.00 |
| 2 | 32 | 10 | 138 | 803 | 5.82 |
| 3 | 10 | 24 | 20 | 636 | 31.80 |
| 4 | 42 | 34 | 34 | 205 | 6.03 |

# THANK YOU