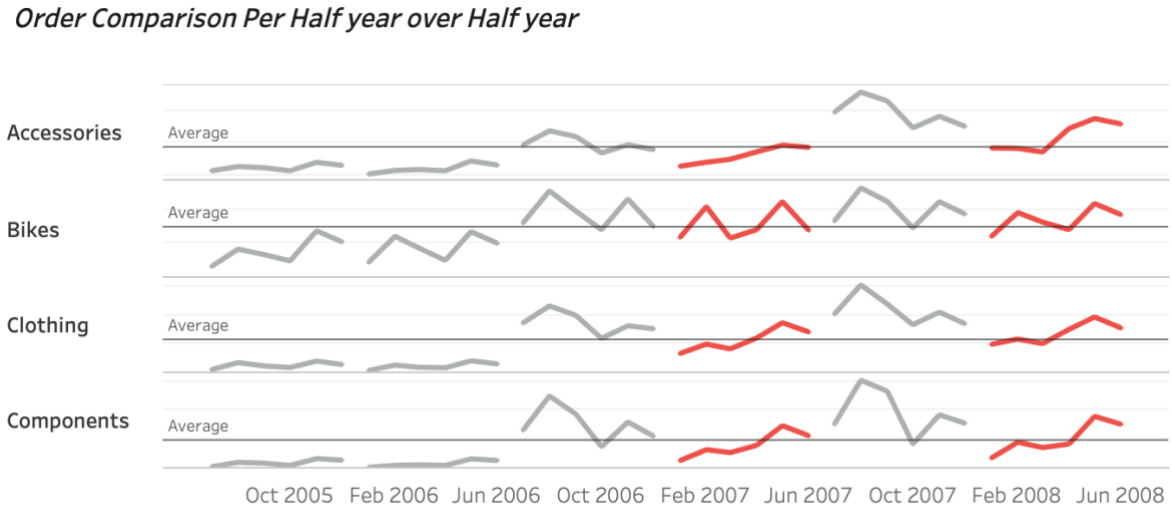
As I was able to do this task in SQL, I used company AWS account to create this table on the redshift, and to load csv data using S3 bucket. The credentials and the path are hidden.

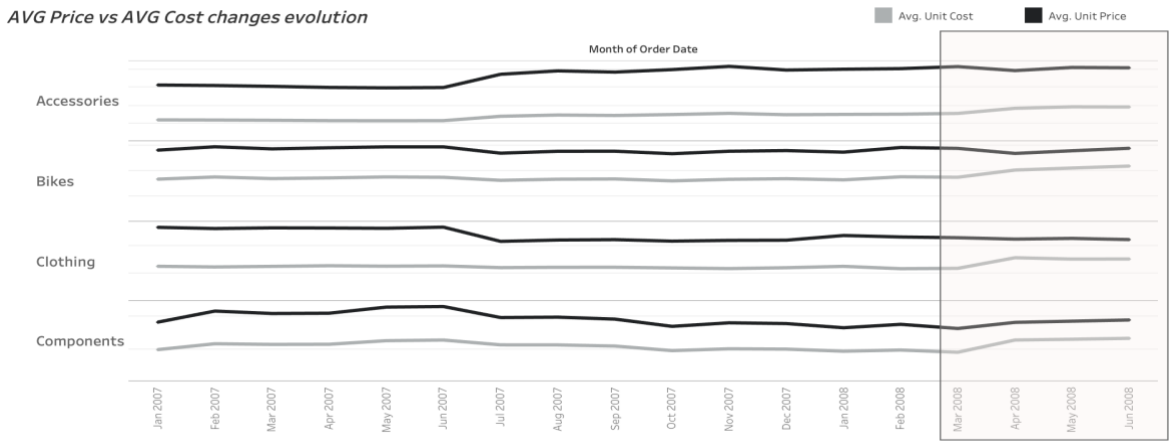
```
DROP TABLE IF EXISTS "sloba_test"."adventure_works_purchase_orders";
CREATE TABLE "sloba_test"."adventure_works_purchase_orders"
(
    "SalesOrderNumber"     VARCHAR(20) ENCODE ZSTD NOT NULL,
    "OrderDate"            DATE        ENCODE ZSTD NOT NULL,
    "DueDate"              VARCHAR(10) ENCODE ZSTD NOT NULL,  --I had
    "ShipDate"             VARCHAR(10) ENCODE ZSTD NOT NULL,  --I had
    "Sales_Person"         VARCHAR(30) ENCODE ZSTD NOT NULL,
    "Sales_Region"         VARCHAR(20) ENCODE ZSTD NOT NULL,
    "Sales_Province"       VARCHAR(20) ENCODE ZSTD NOT NULL,
    "Sales_City"           VARCHAR(20) ENCODE ZSTD NOT NULL,
    "Sales_Postal_Code"    VARCHAR(10) ENCODE ZSTD NOT NULL,
    "Customer_Code"        VARCHAR(20) ENCODE ZSTD NOT NULL,
    "Customer_Name"        VARCHAR(50) ENCODE ZSTD NOT NULL,
    "Customer_Region"      VARCHAR(20) ENCODE ZSTD NOT NULL,
    "Customer_Province"    VARCHAR(20) ENCODE ZSTD NOT NULL,
    "Customer_City"        VARCHAR(20) ENCODE ZSTD NOT NULL,
    "Customer_Postal_Code" VARCHAR(10) ENCODE ZSTD NOT NULL,
    "LineItem_Id"          VARCHAR(20) ENCODE ZSTD NOT NULL,
    "Product_Category"     VARCHAR(20) ENCODE ZSTD NOT NULL,
    "Product_Sub_Category" VARCHAR(20) ENCODE ZSTD NOT NULL,
    "Product_Name"         VARCHAR(40) ENCODE ZSTD NOT NULL,
    "Product_Code"         VARCHAR(20) ENCODE ZSTD NOT NULL,
    "Unit_Cost"            FLOAT       ENCODE ZSTD NOT NULL,
    "UnitPrice"            FLOAT       ENCODE ZSTD NOT NULL,
    "UnitPriceDiscount"    FLOAT       ENCODE ZSTD NOT NULL,
    "OrderQty"             INT         ENCODE ZSTD NOT NULL,
    "Unit_Freight_Cost"    FLOAT       ENCODE ZSTD NOT NULL
)
    DISTSTYLE ALL
    SORTKEY ("Sales_Region", "Product_Category", "OrderDate")
;

--I used copy command to get the data from S3 to redshift table
COPY "sloba_test"."adventure_works_purchase_orders"
    FROM 's3://mybucket/adventure_works_purchase_orders.csv'
    CREDENTIALS 'xxxxx-xxxxx'
    IGNOREHEADER 1
    CSV
    DATEFORMAT AS 'MM/DD/YYYY';
```

After looking at the data first I wanted to look was total order number, I noticing that there is no problem with orders in a total, I compared first half of 2008 year with first half of the 2007.



My next step was to see if there is some problem with Price, maybe prices decreased over some months?



Then I noticed that avg cost per product category increased in April, usually, we expect the price to increase too but here it wasn't the case.

In this table I didn’t find any data so conclusion is that price didn’t change per product name looking at the months from whole sample.

```sql
--Price changes trend over months
CREATE TEMP TABLE "tmp_analyze_price_change"
AS
SELECT *, "unitprice" - "price_before" AS "price_increase",
       "price_increase" / "price_before"::FLOAT AS "price_perc_increase"
FROM (
         SELECT "sales_region",
                "product_category",
                "product_name",
                "orderdate",
                "unitprice",
                LAG("unitprice") OVER (PARTITION BY "sales_region", "product_category", "product_name" ORDER BY "orderdate") AS "price_before"
         FROM "sloba_test"."adventure_works_purchase_orders"
         GROUP BY 1, 2, 3, 4, 5
         ORDER BY 1, 2, 3, 4, 5 DESC)
;

--used this query to check if there was increase of price for specific product, returned no data

-- SELECT *
-- FROM "tmp_analyze_price_change"
-- WHERE "unitprice" != "price_before";
```

Then I did the same for the cost, and noticed that cost for specific products increased, while price stay the same. Which confirmed my theory. There is the question, if this was the technical mistake in the company or it was company decision.

I created this table to keep track on all products with cost change, so later I will be able to fix the price and to see what will be the final result at the end. I had “cost_perc_increase” field which will be used later to update new prices where it needs to be.

```sql
--Cost changes trend over month per product
CREATE TEMP TABLE "tmp_analyze_cost_change"
AS
SELECT "sales_region",
       "product_category",
       "product_name",
       "unit_cost",
       "unit_cost" - "cost_before"           AS "cost_increase",
       "cost_increase" / "cost_before"::FLOAT AS "cost_perc_increase"
FROM (
         SELECT "sales_region",
                "product_category",
                "product_name",
                "orderdate",
                "unit_cost",
                LAG("unit_cost")
                OVER (PARTITION BY "sales_region", "product_category", "product_name" ORDER BY "orderdate") AS "cost_before"
         FROM "sloba_test"."adventure_works_purchase_orders"
         GROUP BY 1, 2, 3, 4, 5
         ORDER BY 1, 2, 3, 4, 5 DESC)
WHERE "unit_cost" != "cost_before"
;
```

Before increasing price for products that were in a problem with cost increase, I wanted to see what was the usual difference between product price and cost before if there was pattern for same product in all months, like Bike Wash – Dissolver had the difference between price and cost 1.67, and now it was 1.39 if I apply later the percentage of cost increase to the price to create new price, will the difference be the same like before?

```
--difference in percentages between price and cost per product and months
CREATE TEMP TABLE "tmp_cost_and_price_difference"
AS
SELECT "sales_region",
       "product_category",
       "product_name",
       "orderdate",
       "unit_cost",
       "unitprice",
       ("unitprice" - "unit_cost") / "unit_cost" AS "difference"
FROM "sloba_test"."adventure_works_purchase_orders"
ORDER BY 1, 2, 3, 4
;
```

In this table I confirmed my theory, and I was able to proceed with the updating price for products as “corrected difference” was the same as “previous difference”. I created corrected price for that products.

```
CREATE TEMP TABLE "tmp_test_previous_difference"
AS
SELECT DISTINCT "sales_region",
                "product_category",
                "product_name",
                "unitprice",
                "cost_increase",
                a."unit_cost",
                "cost_perc_increase",
                "difference",
                "unitprice" + "unitprice" * "cost_perc_increase"              AS "corrected_price",
                ("corrected_price" -a."unit_cost") / a."unit_cost"            AS "corrected_difference",
                MAX("difference") AS "previous_difference"
FROM "tmp_analyze_cost_change" a
        INNER JOIN "tmp_cost_and_price_difference" b USING ("sales_region",
                                                            "product_category",
                                                            "product_name")
GROUP BY 1,2,3,4,5,6,7,8,9
ORDER BY 1, 2, 3;

--used query to check if previous difference between cost and price will be the same if we apply cost increase to create new price
-- SELECT * FROM "tmp_test_previous_difference"
-- ORDER BY 1, 2, 3
-- LIMIT 100;
```

My next step was to calculate total profit as it is, and new total profit which should be the result if there wasn't problem with price as "expected_profit"? To do that I needed to include discount in the calculation, I created "discounted price" which is used later instead of "unitprice" field and to get the "current_profit" of the datasource.

I calculated "price_cut" later used to get the "lost_money" amount,
"fixed_price" later used as new corrected price for products to get the "expected_profit",

```
--use discount to create discount_price, lost money because not increased, and expected profit if the price was up
DROP TABLE IF EXISTS "sloba_test"."discount_counted";
CREATE TABLE "sloba_test"."discount_counted"
AS
SELECT *, "unitprice" - "unitprice" * "unitpricediscount"       AS "discount_price",
       COALESCE("discount_price" * b."cost_perc_increase",0)    AS "price_cut",
       "discount_price" + "price_cut"                           AS "fixed_price",
       "orderqty" * "price_cut"                                 AS "lost_money",
       "orderqty" * ("discount_price" - a."unit_cost")          AS "current_profit",
       "orderqty" * ("fixed_price" - a."unit_cost")             AS "expected_profit"
FROM "sloba_test"."adventure_works_purchase_orders" a
        LEFT JOIN "tmp_analyze_cost_change" b USING ( "sales_region",
                                                      "product_category",
                                                      "product_name",
                                                      "unit_cost" )
ORDER BY 1,2,3
;
```

I created aggregated table, which Is used later to create table "tableau_analyse" which will represent the line of current_total_profit and the expected_total_profit in the tableau dashboard.

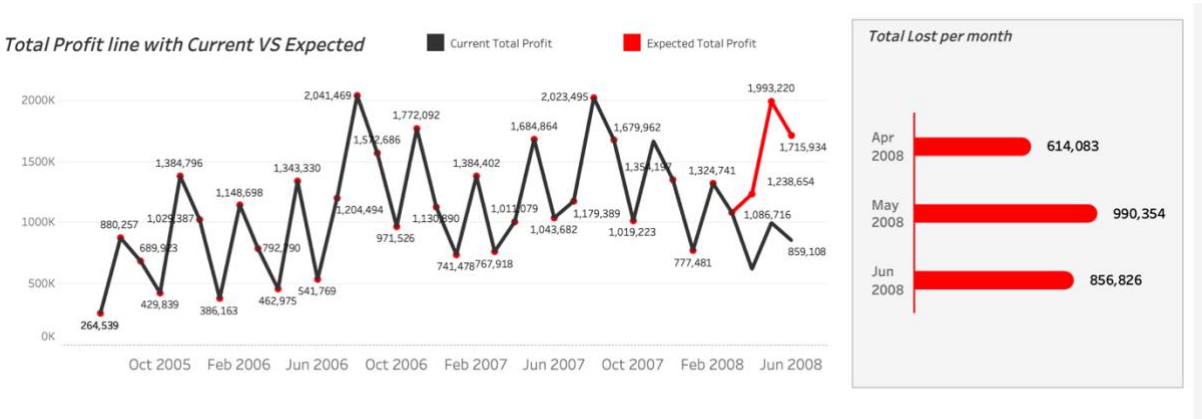
On top of it, to have full picture of profit, I subtracted sum of shipping cost, per region and order date, because, I noticed that per salesordernumber, we have for each product same amount for shipping, it was weird for me, so I counted 1 unit_freight_cost per salesordernumber, I believe that is how it works for the shipping costs.

```
CREATE TEMP TABLE "tmp_agregated"
AS
SELECT "sales_region",
       "orderdate",
       SUM("orderqty") AS "total_order",
       SUM("lost_money") AS "lost_money",
       SUM("current_profit") AS "current_profit",
       SUM("expected_profit") AS "expected_profit"
FROM "sloba_test"."discount_counted"
GROUP BY 1, 2;

--find total shipping cost per sales region and month
CREATE TEMP TABLE "tmp_shipping_costs"
AS
SELECT "sales_region",
       "orderdate",
       SUM("unit_freight_cost") AS "total_shipping_cost"
FROM (
        SELECT DISTINCT "sales_region",
                        "salesordernumber",
                        "orderdate",
                        "unit_freight_cost"
        FROM "sloba_test"."adventure_works_purchase_orders")
GROUP BY 1, 2;

--create final profit based on shipping costs
DROP TABLE IF EXISTS "sloba_test"."tableau_analyse";
CREATE TABLE "sloba_test"."tableau_analyse"
AS
SELECT "sales_region",
       "orderdate",
       "total_order",
       "lost_money",
       "current_profit",
       "expected_profit",
       "total_shipping_cost",
       "expected_profit" - "total_shipping_cost" AS "expected_total_profit",
       "current_profit" - "total_shipping_cost"  AS "current_total_profit"
FROM "tmp_agregated" a
        LEFT JOIN "tmp_shipping_costs" b USING ("sales_region", "orderdate");
```

Final picture of the profit line was this, and you can find it on my tableau public profile: https://public.tableau.com/profile/slobodan.ilic#!/vizhome/AWPO/Dashboard1



The conclusion was that Total Profit declined over past few months because cost price for some products increased while sell price remind the same.