SUPPLY CHAIN INVENTORY MANAGEMENT FOR AUTOMOTIVE DISTRIBUTORS

SHARUN
SANKET
GAYATRI
SHRUTHI

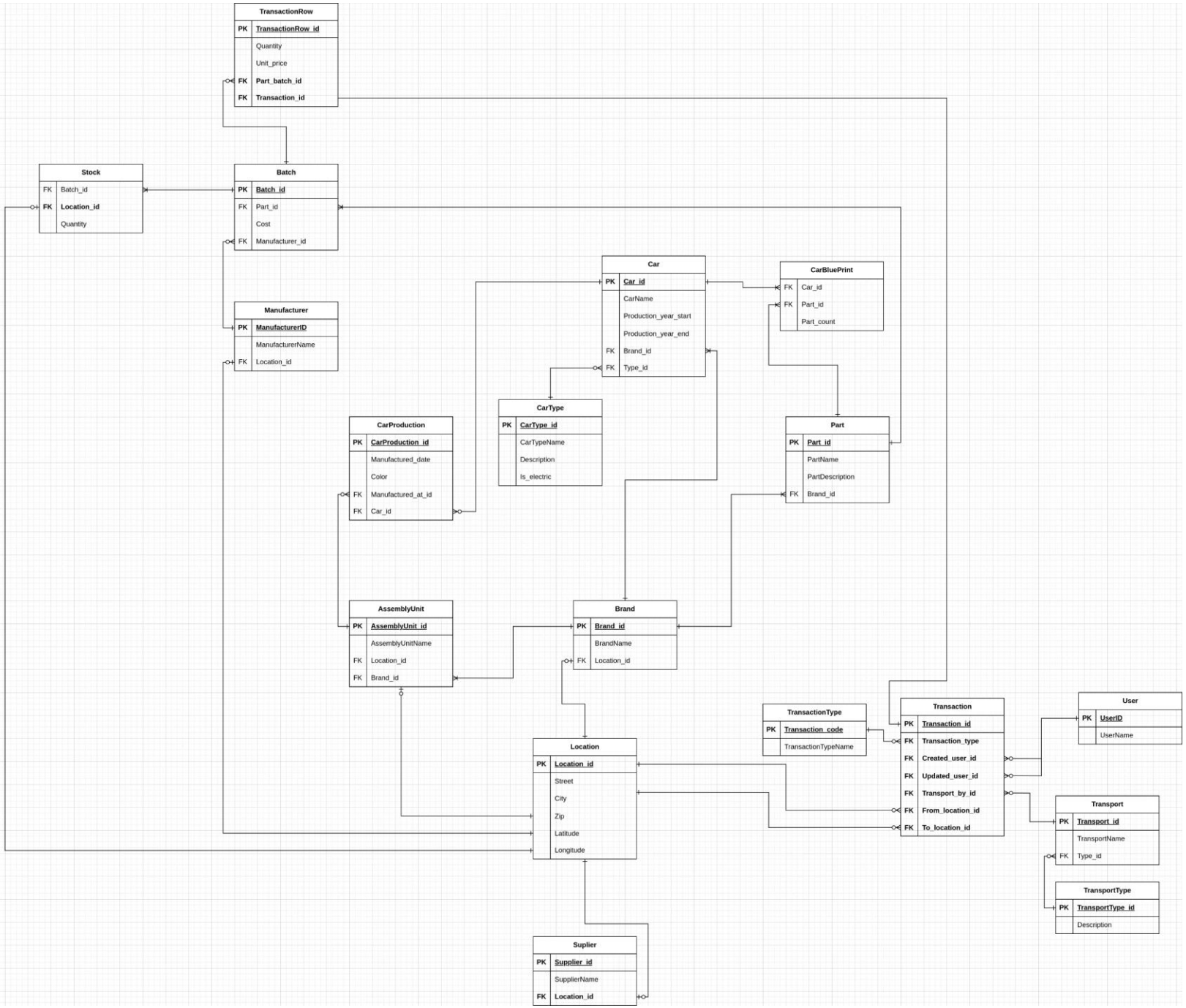
Project For DAMG6210 – Data Management and Database Design

Under the guidance of Manuel D Montrond

PROJECT HIGHLIGHTS

- Complex Data Models for Reduced Redundancy
- Constraint Checks for Ensuring Correctness of Data
- Triggers for Auto-Updating Related Tables (e.g. Stock Update)
- GUI for browsing data powered by Prisma

DESIGN DOCUMENTS



FINAL ERD

ERD Link:
https://github.com/sharunkumar/DAMG6210/blob/main/diagrams/erd.png

TRIGGER HIGHLIGHT - CARPRODUCTION

Create a car by inserting into `CarProduction` - it will automatically deduct the stock of the required parts from the `Stock` table, based on `CarBlueprint`

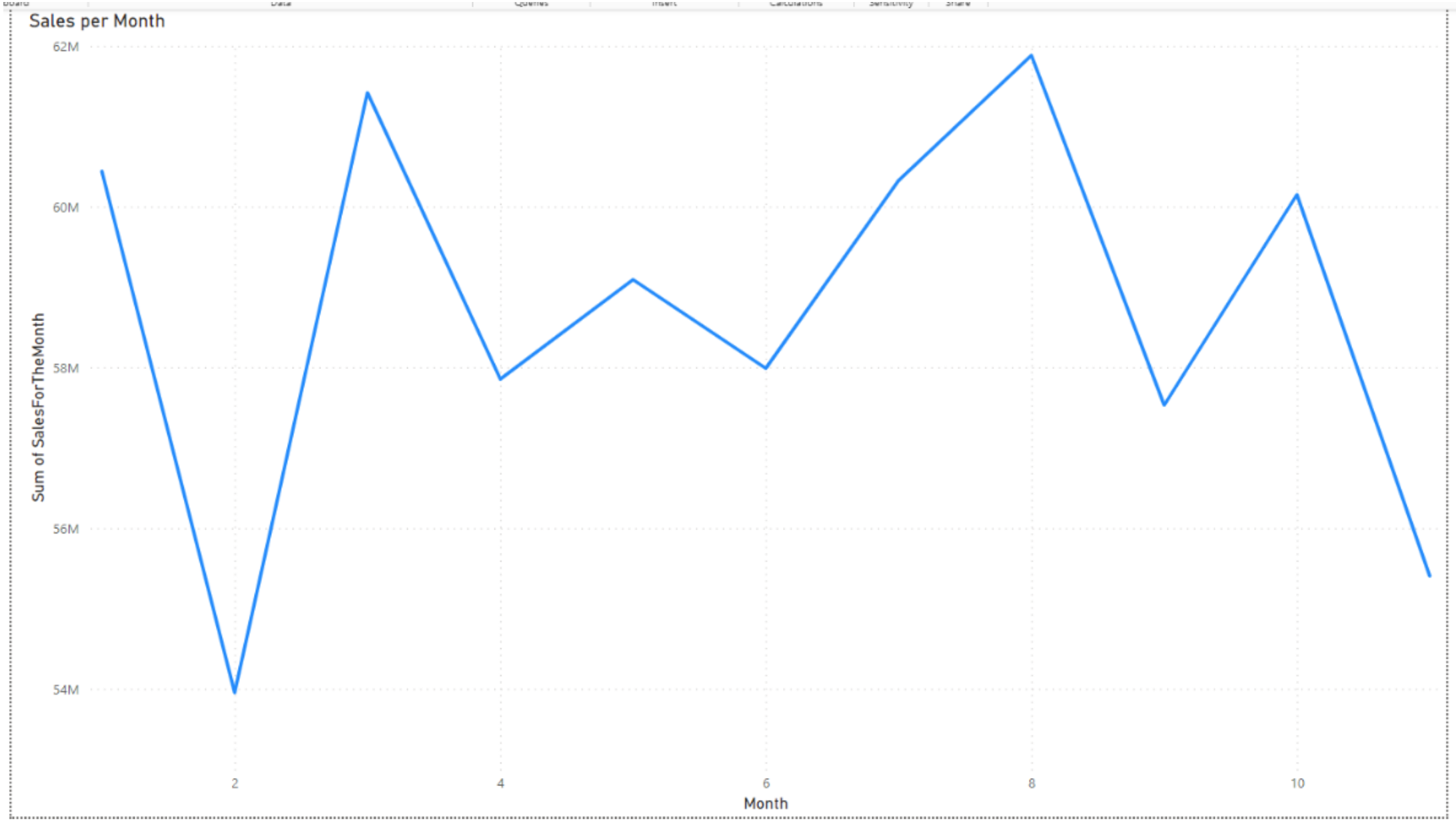
```sql
CREATE or ALTER TRIGGER CarProduction_Stock_Consume
ON CarProduction
AFTER INSERT
AS
BEGIN
    SET NOCOUNT ON;
    -- trigger for consuming parts (quantity) from stock when a car is produced

    WITH RankedRows AS ( -- select all candidate batches and rank them based on quantity
    select B.part_id, S.batch_id, S.location_id, S.quantity, CB.part_count, CB.car_id
    , ROW_NUMBER() OVER(PARTITION BY B.part_id ORDER BY quantity DESC) AS RowNumber
    from inserted I
    inner join AssemblyUnit A on A.AssemblyUnitID = I.manufactured_at_id
    inner join [Location] L on L.LocationID = A.location_id
    inner join CarBlueprint CB on CB.car_id = I.car_id
    inner join Batch B on B.part_id = CB.part_id
    inner join Stock S on S.batch_id = B.BatchID and S.location_id = L.LocationID
    )
    update RankedRows
    set quantity = quantity - (part_count * (select count(1) from inserted II where II.car_id = RankedRows.car_id)) -- in case multiple production of the same car is inserted
    where RowNumber = 1

END
```

TRIGGER HIGHLIGHT – STOCK UPDATE

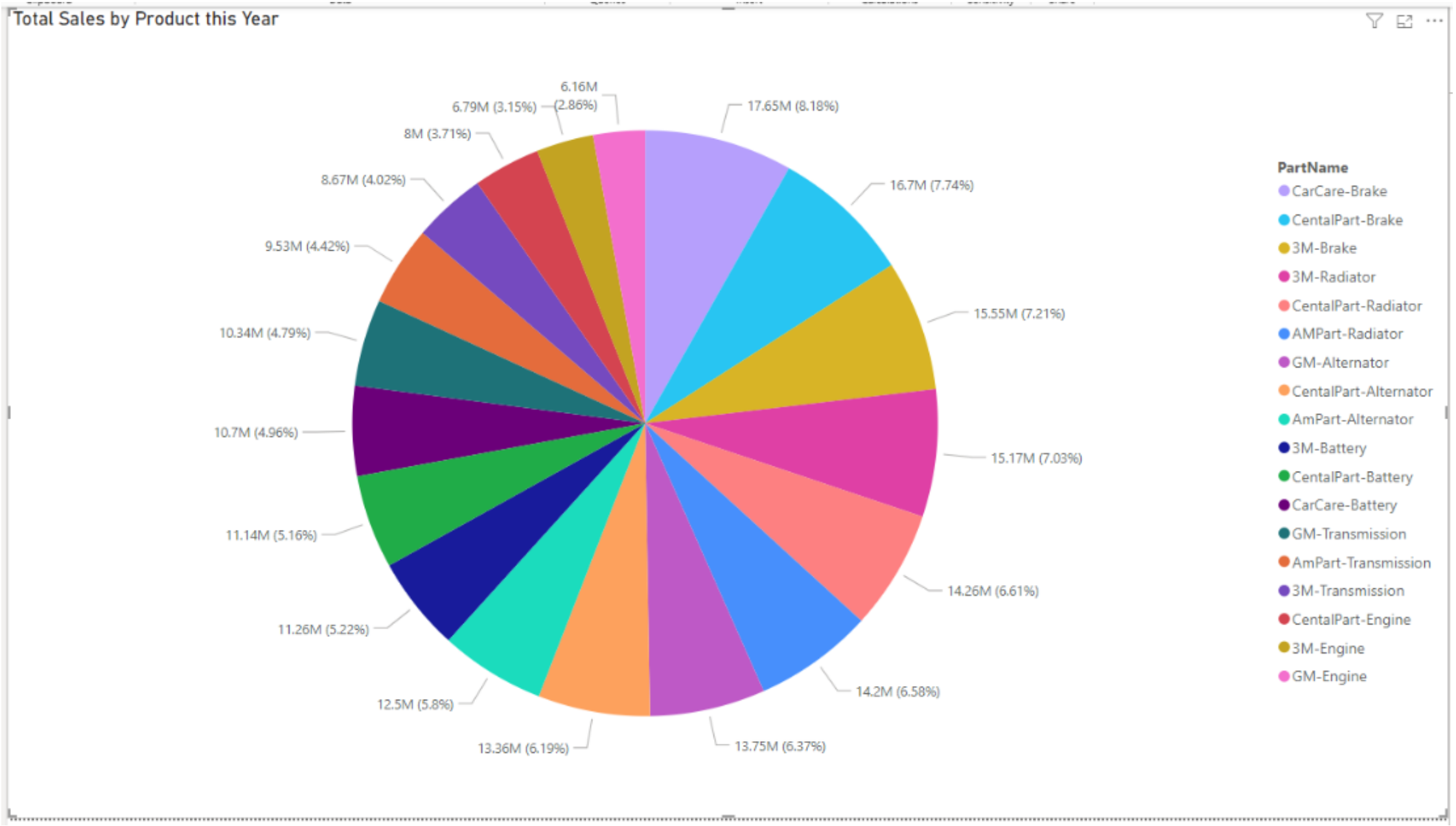
When inserting a row `TransactionRow`, the stock is automatically transferred from one location from the other in the `Stock` table

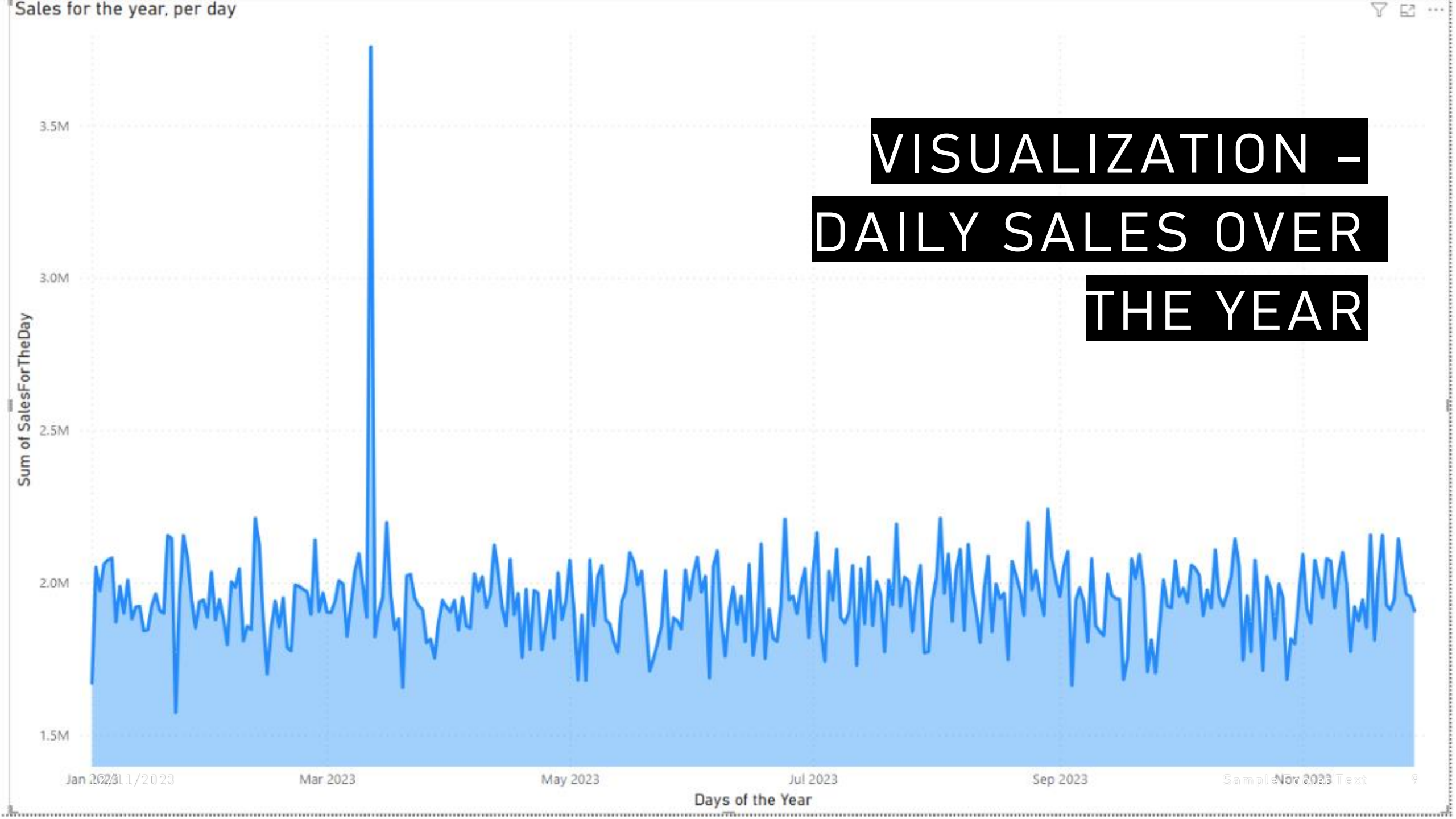
```
CREATE or ALTER TRIGGER StockUpdate
ON TransactionRow
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    SET NOCOUNT ON;

    -- handle updated rows
    update sf
    set sf.quantity = sf.quantity - (d.quantity - i.quantity)
    from inserted i inner join deleted d on i.TransactionRowID = d.TransactionRowID and i.quantity <> d.quantity
        inner join [Transaction] t on i.transaction_id = t.TransactionID
        inner join Stock sf on sf.location_id = t.from_location_id and i.part_batch_id = sf.batch_id

    update st
    set st.quantity = st.quantity + (d.quantity - i.quantity)
    from inserted i inner join deleted d on i.TransactionRowID = d.TransactionRowID and i.quantity <> d.quantity
        inner join [Transaction] t on i.transaction_id = t.TransactionID
        inner join Stock st on st.location_id = t.to_location_id and i.part_batch_id = st.batch_id


    -- handle deleted row
    update sf
        set sf.quantity = sf.quantity + d.quantity
         from deleted d inner join [transaction] t on d.TransactionRowID not in (select TransactionRowID
            from inserted) and t.TransactionID = d.transaction_id
        inner join stock sf on sf.location_id = t.from_location_id

    update st
        set st.quantity = st.quantity - d.quantity
         from deleted d inner join [transaction] t on d.TransactionRowID not in (select TransactionRowID
            from inserted) and t.TransactionID = d.transaction_id
        inner join stock st on st.location_id = t.to_location_id

    -- handle inserted row
    update sf
        set sf.quantity = sf.quantity - i.quantity
         from inserted i inner join [transaction] t on i.TransactionRowID not in (select TransactionRowID
            from deleted) and t.TransactionID = i.transaction_id
        inner join stock sf on sf.location_id = t.from_location_id

    update st
        set st.quantity = st.quantity + i.quantity
         from inserted i inner join [transaction] t on i.TransactionRowID not in (select TransactionRowID
            from deleted) and t.TransactionID = i.transaction_id
        inner join stock st on st.location_id = t.to_location_id

END
```

VISUALIZATION – SALES PER MONTH



VISUALIZATION – SALES BY PART




Sales for the year, per day
3.5M
3.0M
2.5M
2.0M
1.5M
Sum of SalesForTheDay
Jan 2023
Mar 2023
May 2023
Jul 2023
Sep 2023
Nov 2023
Days of the Year
VISUALIZATION – DAILY SALES OVER THE YEAR


LIVE DEMO