Awesome — here’s Lesson 14: DAX Optimization with ready-to-use measures and short explanations.

Assumptions (rename if your model differs):

Fact table: choco_sales with columns Date, Amount, Product, Boxes (if your boxes column is named differently, replace it below).

Date table: Date (created with CALENDARAUTO()), marked as Date Table, related choco_sales[Date] → Date[Date].

Base measures

```
Revenue := SUM ( choco_sales[Amount] )
Boxes Shipped := SUM ( choco_sales[Boxes] )
```

1) % Growth vs Last Year — using VAR

```
YoY % (VAR) :=
VAR Sales_Cur = [Revenue]
VAR Sales_LY  = CALCULATE ( [Revenue], SAMEPERIODLASTYEAR ( 'Date'[Date] ) )
RETURN DIVIDE ( Sales_Cur - Sales_LY, Sales_LY )
```

2) Difference between current month and previous month — using VAR

```
Monthly Diff (This vs Prev) :=
VAR Sales_CM =
   TOTALMTD ( [Revenue], 'Date'[Date] )
VAR Sales_PM =
   CALCULATE ( TOTALMTD ( [Revenue], 'Date'[Date] ), DATEADD ( 'Date'[Date], -1, MONTH ) )
RETURN Sales_CM - Sales_PM
```

3) One measure that shows both total boxes and average monthly boxes (using VAR)

Returns a formatted text like “Total: 12,345 | Avg/Month: 1,029” so you can drop it on a Card.

```
Boxes: Total & Avg/Month (Text) :=
VAR TotalBoxes = [Boxes Shipped]
VAR MonthsSelected =
   DISTINCTCOUNT ( 'Date'[YearMonth] )  -- Create 'YearMonth' column in Date table if you haven't.
VAR AvgPerMonth = DIVIDE ( TotalBoxes, MonthsSelected )
RETURN
"Total: " & FORMAT ( TotalBoxes, "#,0" ) &
" | Avg/Month: " & FORMAT ( AvgPerMonth, "#,0" )
```

4) Same calculation but return only Average Monthly Boxes

```
Boxes Avg per Month :=
VAR TotalBoxes = [Boxes Shipped]
VAR MonthsSelected = DISTINCTCOUNT ( 'Date'[YearMonth] )
RETURN DIVIDE ( TotalBoxes, MonthsSelected )
```

5) Growth % from last month

```
MoM % :=
VAR Sales_CM =
   TOTALMTD ( [Revenue], 'Date'[Date] )
VAR Sales_PM =
   CALCULATE ( TOTALMTD ( [Revenue], 'Date'[Date] ), DATEADD ( 'Date'[Date], -1, MONTH ) )
RETURN DIVIDE ( Sales_CM - Sales_PM, Sales_PM )
```

6) 3-Month Moving Average (rolling)

```
3M Moving Avg :=
AVERAGEX (
   DATESINPERIOD ( 'Date'[Date], MAX ( 'Date'[Date] ), -3, MONTH ),
   [Revenue]
)
```

7) Dynamic Card Message per Product (rank + YoY)

Drop this measure on a Card with a Product filter context (e.g., a product slicer or a small multiples layout).

Messages:

“Top Performer – Sales up by X%” (Top 5 & positive YoY)

“Consistent Performer” (YoY within ±5%)

“Needs Improvement” (otherwise)

```
Product Performance Message :=
VAR ProductName = SELECTEDVALUE ( choco_sales[Product] )
VAR SalesThis   = [Revenue]
VAR SalesLY     = CALCULATE ( [Revenue], SAMEPERIODLASTYEAR ( 'Date'[Date] ) )
VAR YoY         = DIVIDE ( SalesThis - SalesLY, SalesLY )
VAR RankTotal   =
   RANKX (
      ALL ( choco_sales[Product] ),
      [Revenue],
            ,
            DESC,
            DENSE
        )
    RETURN
    SWITCH (
        TRUE(),
        NOT ISBLANK(ProductName) && RankTotal <= 5 && YoY > 0,
            "Top Performer - Sales up by " & FORMAT ( YoY, "0%" ),
        NOT ISBLANK(ProductName) && ABS ( YoY ) <= 0.05,
            "Consistent Performer",
        NOT ISBLANK(ProductName),
            "Needs Improvement",
        BLANK()
    )
```

8) Top 5 manual DAX optimization tips (and why)

Use VAR to cache expensive expressions
– Prevents recalculating the same filter/context transitions multiple times (fewer storage engine/Formula engine calls).

Reduce row context → iterate only when needed
– Prefer filter/context modifiers (CALCULATE, KEEPFILTERS, TREATAS) over wide X-iterators; use SUMX only when you must.

Limit the evaluation scope with ALLSELECTED/REMOVEFILTERS precisely
– Avoid broad ALL() across entire tables when a single column is enough; reduces intermediate result size.

Precompute calendar attributes in the Date table (Year, MonthNo, YearMonth)
– Cleaner measures, fewer FORMAT()/YEAR() calls at query time; helps the engine hit cached group-bys.

Avoid volatile string concatenations in analytical measures
– Keep numeric logic numeric; use separate display measures (as in #3). Strings break query folding and cache re-use.

(Extras you'll feel in bigger models: create slim star schemas; avoid bi-directional relationships unless absolutely necessary; use calculated columns for static attributes, measures for dynamic logic.)

9) Why use DAX optimization tools?

DAX Studio — query the model directly, inspect server timings, measure CPU vs SE time, materialization, cache hits; export/trace to find bottlenecks.

Performance Analyzer (Power BI) — per-visual breakdown (DAX query, visual display, data fetch); helps rank visuals/measures by cost.

Tabular Editor — manage measure dependencies, Best Practices Analyzer, format DAX, create calculation groups (e.g., time-intel selectors) to reduce measure explosion and repeated logic.

10) Top-5 Flag (Yes/No) by Total Sales — compute rank once

```
Top 5 Flag (Yes/No) :=
VAR RankBySales =
    RANKX (
        ALL ( choco_sales[Product] ),
        [Revenue],
        ,
        DESC,
        DENSE
    )
RETURN IF ( RankBySales <= 5, "Yes", "No" )
```