DAX Formulas Reference
Hot Rolling Plant - Temper Line Analytics
Complete Measure Library

Table of Contents

---

1. PRODUCTION METRICS

Basic Counts

```dax
Total Pieces =
COUNTROWS(fact_production_coil)
```

```dax
Prime Pieces =
CALCULATE(
    COUNTROWS(fact_production_coil),
    fact_production_coil[is_prime] = TRUE
)
```

```dax
Scrap Pieces =
CALCULATE(
    COUNTROWS(fact_production_coil),
    fact_production_coil[is_scrap] = TRUE
)
```

```dax
Total Parent Coils =
DISTINCTCOUNT(fact_production_coil[parent_coil_id])
```

Rates and Ratios

```dax
Prime Rate % =
VAR PrimePieces = [Prime Pieces]
VAR TotalPieces = [Total Pieces]
RETURN
    DIVIDE(PrimePieces, TotalPieces, 0) * 100
```

```dax
Scrap Rate % =
VAR ScrapPieces = [Scrap Pieces]
VAR TotalPieces = [Total Pieces]
RETURN
    DIVIDE(ScrapPieces, TotalPieces, 0) * 100
```

```dax
Pieces per Parent =
VAR TotalPieces = [Total Pieces]
VAR ParentCoils = [Total Parent Coils]
RETURN
    DIVIDE(TotalPieces, ParentCoils, 0)
```

Production Volume

```dax
Total Mass Output =
SUM(fact_production_coil[mass_out_tons])
```

```dax
Average Mass per Piece =
AVERAGE(fact_production_coil[mass_out_tons])
```

```dax
```

Total Prime Mass =
CALCULATE(
    SUM(fact_production_coil[mass_out_tons]),
    fact_production_coil[is_prime] = TRUE
)

```dax
Total Scrap Mass =
CALCULATE(
    SUM(fact_production_coil[mass_out_tons]),
    fact_production_coil[is_scrap] = TRUE
)
```

Cycle Time Metrics

```dax
Avg Cycle Time =
AVERAGE(fact_production_coil[total_cycle_time_min])
```

```dax
Median Cycle Time =
MEDIAN(fact_production_coil[total_cycle_time_min])
```

```dax
Min Cycle Time =
MIN(fact_production_coil[total_cycle_time_min])
```

```dax
Max Cycle Time =
MAX(fact_production_coil[total_cycle_time_min])
```

```dax
StdDev Cycle Time =
STDEV.P(fact_production_coil[total_cycle_time_min])
```

```dax
Cycle Time Range =
[Max Cycle Time] - [Min Cycle Time]
```

---

2. TEMPO AND GAP ANALYSIS

Tempo Calculations

```dax
Pieces per Hour =
VAR AvgCycle = [Avg Cycle Time]
RETURN
    IF(
        AvgCycle > 0,
        DIVIDE(60, AvgCycle, 0),
        0
    )
```

```dax
Tempo Target = 4.0
// Note: Adjust based on plant-specific target
```

```dax
Tempo Achievement % =
VAR CurrentTempo = [Pieces per Hour]
VAR Target = [Tempo Target]
RETURN
    DIVIDE(CurrentTempo, Target, 0) * 100
```

```dax
Tempo Variance =
[Pieces per Hour] - [Tempo Target]
```

```dax
Tempo Status =
VAR Achievement = [Tempo Achievement %]
RETURN
    SWITCH(
        TRUE(),
        Achievement >= 100, "On Target",
        Achievement >= 90, "Near Target",
        Achievement >= 80, "Below Target",
        "Critical"
    )
```

Gap Analysis

```dax
Avg Completion Gap =
AVERAGE(fact_production_coil[gap_from_prev_completion_min])
```

```dax
Median Completion Gap =
MEDIAN(fact_production_coil[gap_from_prev_completion_min])
```

```dax
Avg Parent Gap =
AVERAGE(fact_production_coil[gap_from_prev_parent_min])
```

```dax
Median Parent Gap =
MEDIAN(fact_production_coil[gap_from_prev_parent_min])
```

```dax
Total Gap Time =
SUM(fact_production_coil[gap_from_prev_completion_min])
```

```dax
Gap Time as % of Total =
VAR TotalGapTime = [Total Gap Time]
VAR TotalCycleTime = SUM(fact_production_coil[total_cycle_time_min])
RETURN
    DIVIDE(TotalGapTime, TotalCycleTime + TotalGapTime, 0) * 100
```

```dax
Short Gaps Count =
CALCULATE(
    COUNTROWS(fact_production_coil),
    fact_production_coil[gap_from_prev_completion_min] < 2
)
```

```dax
Long Gaps Count =
CALCULATE(
    COUNTROWS(fact_production_coil),
    fact_production_coil[gap_from_prev_completion_min] > 10
)
```

---

3. EQUIPMENT UTILIZATION

Time Calculations

```dax
Total RUN Hours =
CALCULATE(
    SUM(fact_equipment_event_log[event_duration_sec]) / 3600,
    fact_equipment_event_log[event_type] = "RUN"
)
```

```dax
Total IDLE Hours =
CALCULATE(
    SUM(fact_equipment_event_log[event_duration_sec]) / 3600,
    fact_equipment_event_log[event_type] = "IDLE"
)
```

```dax
Total FAULT Hours =
CALCULATE(
    SUM(fact_equipment_event_log[event_duration_sec]) / 3600,
    fact_equipment_event_log[event_type] = "FAULT"
)
```

```dax
Total Equipment Hours =
[Total RUN Hours] + [Total IDLE Hours] + [Total FAULT Hours]
```

Utilization Percentages

```dax
Equipment Utilization % =
VAR RunHours = [Total RUN Hours]
VAR TotalHours = [Total Equipment Hours]
RETURN
```

```dax
    DIVIDE(RunHours, TotalHours, 0) * 100
```

```dax
Equipment Availability % =
VAR AvailableHours = [Total RUN Hours] + [Total IDLE Hours]
VAR TotalHours = [Total Equipment Hours]
RETURN
    DIVIDE(AvailableHours, TotalHours, 0) * 100
```

```dax
Downtime % =
VAR FaultHours = [Total FAULT Hours]
VAR TotalHours = [Total Equipment Hours]
RETURN
    DIVIDE(FaultHours, TotalHours, 0) * 100
```

```dax
Idle % =
VAR IdleHours = [Total IDLE Hours]
VAR TotalHours = [Total Equipment Hours]
RETURN
    DIVIDE(IdleHours, TotalHours, 0) * 100
```

Utilization Status

```dax
Utilization Status =
VAR Util = [Equipment Utilization %]
RETURN
    SWITCH(
        TRUE(),
        Util >= 80, "Excellent",
        Util >= 70, "Good",
        Util >= 60, "Fair",
        Util >= 50, "Poor",
        "Critical"
    )
```

```dax
Equipment Health Score =
VAR Utilization = [Equipment Utilization %]
VAR Availability = [Equipment Availability %]
```

VAR DowntimePct = [Downtime %]
RETURN
    (Utilization * 0.5) + (Availability * 0.3) - (DowntimePct * 0.2)

---

4. BOTTLENECK ANALYSIS

Operation Time Analysis

```dax
Total Operation Hours =
SUM(fact_coil_operation_cycle[operation_duration_sec]) / 3600
```

```dax
Avg Equipment Operation Time =
AVERAGE(fact_coil_operation_cycle[operation_duration_sec]) / 60
```

```dax
Bottleneck Operation Hours =
CALCULATE(
    SUM(fact_coil_operation_cycle[operation_duration_sec]) / 3600,
    dim_equipment[is_bottleneck_candidate] = TRUE
)
```

```dax
Non-Bottleneck Operation Hours =
CALCULATE(
    SUM(fact_coil_operation_cycle[operation_duration_sec]) / 3600,
    dim_equipment[is_bottleneck_candidate] = FALSE
)
```

Bottleneck Shares

```dax
Bottleneck Share % =
VAR BottleneckHours = [Bottleneck Operation Hours]
VAR TotalHours = [Total Operation Hours]
RETURN
    DIVIDE(BottleneckHours, TotalHours, 0) * 100
```

```dax
Equipment Time Share % =
VAR EquipmentTime = SUM(fact_coil_operation_cycle[operation_duration_sec])
VAR TotalTime =
    CALCULATE(
        SUM(fact_coil_operation_cycle[operation_duration_sec]),
        ALL(dim_equipment)
    )
RETURN
    DIVIDE(EquipmentTime, TotalTime, 0) * 100
```

Bottleneck Severity

```dax
Equipment Bottleneck Score =
VAR EquipmentOpsTime = SUM(fact_coil_operation_cycle[operation_duration_sec])
VAR TotalOpsTime =
    CALCULATE(
        SUM(fact_coil_operation_cycle[operation_duration_sec]),
        ALL(dim_equipment)
    )
VAR TimeShare = DIVIDE(EquipmentOpsTime, TotalOpsTime, 0)
VAR IsBottleneck = MAX(dim_equipment[is_bottleneck_candidate])
VAR UtilizationPct = [Equipment Utilization %]
VAR DowntimePct = [Downtime %]
RETURN
    IF(
        IsBottleneck,
        (TimeShare * 50) + (UtilizationPct * 0.3) + (DowntimePct * 0.2),
        TimeShare * 25
    )
```

```dax
Bottleneck Rank =
RANKX(
    ALL(dim_equipment),
    [Equipment Bottleneck Score],
    ,
    DESC,
    DENSE
)
```

```dax
Top Bottleneck =
```

VAR TopRank = [Bottleneck Rank]
RETURN
    IF(TopRank = 1, "Primary Bottleneck", "")

Cumulative Analysis

```dax
Cumulative Bottleneck Share =
VAR CurrentEquipment = MAX(dim_equipment[equipment_name])
VAR CurrentShare = [Equipment Time Share %]
VAR PriorShares =
    CALCULATE(
        SUMX(
            FILTER(
                ALL(dim_equipment),
                dim_equipment[process_order] < MAX(dim_equipment[process_order])
            ),
            [Equipment Time Share %]
        )
    )
RETURN
    PriorShares + CurrentShare
```

---

5. MAINTENANCE AND RELIABILITY

Event Counts

```dax
Total Maintenance Events =
COUNTROWS(fact_maintenance_event)
```

```dax
Unplanned Events =
CALCULATE(
    COUNTROWS(fact_maintenance_event),
    fact_maintenance_event[Category] <> "Planned downtime"
)
```

```dax
Planned Events =
CALCULATE(
    COUNTROWS(fact_maintenance_event),
    fact_maintenance_event[Category] = "Planned downtime"
)
```

Downtime Hours

```dax
Total Maintenance Hours =
SUM(fact_maintenance_event[duration_hours])
```

```dax
Unplanned Downtime Hours =
CALCULATE(
    SUM(fact_maintenance_event[duration_hours]),
    fact_maintenance_event[Category] <> "Planned downtime"
)
```

```dax
Planned Downtime Hours =
CALCULATE(
    SUM(fact_maintenance_event[duration_hours]),
    fact_maintenance_event[Category] = "Planned downtime"
)
```

```dax
Avg Downtime per Event =
DIVIDE(
    [Total Maintenance Hours],
    [Total Maintenance Events],
    0
)
```

Reliability Metrics

```dax
MTBF =
// Mean Time Between Failures
VAR TotalProductionHours = [Total RUN Hours] + [Total IDLE Hours]
VAR FailureCount = [Unplanned Events]
RETURN
    DIVIDE(TotalProductionHours, FailureCount, 0)
```

```dax
MTTR =
// Mean Time To Repair
CALCULATE(
    AVERAGE(fact_maintenance_event[duration_hours]),
    fact_maintenance_event[Category] <> "Planned downtime"
)
```

```dax
Equipment Reliability Score =
VAR MTBFValue = [MTBF]
VAR MTTRValue = [MTTR]
VAR MaxMTBF = 100  // Adjust based on plant standards
VAR MaxMTTR = 10   // Adjust based on plant standards
RETURN
    ((MTBFValue / MaxMTBF) * 70) + ((1 - (MTTRValue / MaxMTTR)) * 30)
```

Maintenance Effectiveness

```dax
Planned Maintenance % =
VAR PlannedHours = [Planned Downtime Hours]
VAR TotalMaintHours = [Total Maintenance Hours]
RETURN
    DIVIDE(PlannedHours, TotalMaintHours, 0) * 100
```

```dax
Maintenance Cost Impact =
// Estimate lost production value
VAR DowntimeHours = [Unplanned Downtime Hours]
VAR PiecesPerHour = [Pieces per Hour]
VAR LostPieces = DowntimeHours * PiecesPerHour
VAR AvgMassPerPiece = [Average Mass per Piece]
VAR LostMass = LostPieces * AvgMassPerPiece
VAR ValuePerTon = 800  // Adjust based on product value
RETURN
    LostMass * ValuePerTon
```

---

6. SHIFT PERFORMANCE

Shift Metrics

```dax
Pieces This Shift =
CALCULATE(
    [Total Pieces],
    USERELATIONSHIP(
        fact_production_coil[production_date],
        dim_date_crew_schedule[production_date]
    )
)
```

```dax
Shift Tempo =
VAR ShiftPieces = [Pieces This Shift]
VAR ShiftHours = 12  // 12-hour shifts
RETURN
    DIVIDE(ShiftPieces, ShiftHours, 0)
```

```dax
Shift Efficiency % =
VAR ShiftTempo = [Shift Tempo]
VAR Target = [Tempo Target]
RETURN
    DIVIDE(ShiftTempo, Target, 0) * 100
```

```dax
Shift Prime Rate % =
VAR ShiftPrimePieces =
    CALCULATE(
        [Prime Pieces],
        USERELATIONSHIP(
            fact_production_coil[production_date],
            dim_date_crew_schedule[production_date]
        )
    )
VAR ShiftTotalPieces = [Pieces This Shift]
RETURN
    DIVIDE(ShiftPrimePieces, ShiftTotalPieces, 0) * 100
```

Shift Comparison

```dax
```

```
Best Shift Tempo =
MAXX(
    VALUES(fact_production_coil[shift_code]),
    [Shift Tempo]
)
```

```dax
Worst Shift Tempo =
MINX(
    VALUES(fact_production_coil[shift_code]),
    [Shift Tempo]
)
```

```dax
Shift Tempo Range =
[Best Shift Tempo] - [Worst Shift Tempo]
```

```dax
Shift Rank =
RANKX(
    ALL(fact_production_coil[shift_code]),
    [Shift Tempo],
    ,
    DESC,
    DENSE
)
```

```dax
Shift Performance vs Average =
VAR ShiftTempo = [Shift Tempo]
VAR OverallTempo =
    CALCULATE(
        [Shift Tempo],
        ALL(fact_production_coil[shift_code])
    )
RETURN
    ShiftTempo - OverallTempo
```

Shift Handover Analysis

```dax
Shift Handover Loss =
// Estimate tempo loss during shift changes
VAR AvgGapOverall = [Avg Completion Gap]
VAR ShiftStartGaps =
    CALCULATE(
        AVERAGE(fact_production_coil[gap_from_prev_completion_min]),
        fact_production_coil[completion_ts].[Hour] >= 6,
        fact_production_coil[completion_ts].[Hour] < 7
    )
RETURN
    IF(
        NOT(ISBLANK(ShiftStartGaps)),
        ShiftStartGaps - AvgGapOverall,
        0
    )
```

```dax
Day Shift Pieces =
CALCULATE(
    [Total Pieces],
    fact_production_coil[completion_ts].[Hour] >= 6,
    fact_production_coil[completion_ts].[Hour] < 18
)
```

```dax
Night Shift Pieces =
CALCULATE(
    [Total Pieces],
    fact_production_coil[completion_ts].[Hour] >= 18 ||
    fact_production_coil[completion_ts].[Hour] < 6
)
```

```dax
Day vs Night % Difference =
VAR DayPieces = [Day Shift Pieces]
VAR NightPieces = [Night Shift Pieces]
VAR DayRate = DIVIDE(DayPieces, DayPieces + NightPieces, 0)
VAR NightRate = DIVIDE(NightPieces, DayPieces + NightPieces, 0)
RETURN
    (DayRate - NightRate) * 100
```

---

7. PRODUCT MIX ANALYSIS

Product Band Classification

```dax
Fast Band Pieces =
CALCULATE(
    COUNTROWS(fact_production_coil),
    fact_production_coil[thickness_mm] <= 2.0,
    fact_production_coil[width_mm] <= 1300
)
```

```dax
Standard Mix Pieces =
[Total Pieces] - [Fast Band Pieces]
```

```dax
Fast Band % =
DIVIDE(
    [Fast Band Pieces],
    [Total Pieces],
    0
) * 100
```

Product Performance

```dax
Avg Cycle Fast Band =
CALCULATE(
    AVERAGE(fact_production_coil[total_cycle_time_min]),
    fact_production_coil[thickness_mm] <= 2.0,
    fact_production_coil[width_mm] <= 1300
)
```

```dax
Avg Cycle Standard Mix =
CALCULATE(
    AVERAGE(fact_production_coil[total_cycle_time_min]),
    NOT(
        fact_production_coil[thickness_mm] <= 2.0 &&
        fact_production_coil[width_mm] <= 1300
    )
)
```

```dax
Cycle Time Difference =
[Avg Cycle Standard Mix] - [Avg Cycle Fast Band]
```

```dax
Tempo Fast Band =
DIVIDE(60, [Avg Cycle Fast Band], 0)
```

```dax
Tempo Standard Mix =
DIVIDE(60, [Avg Cycle Standard Mix], 0)
```

Product Diversity

```dax
Product Mix Complexity =
VAR UniqueWidths = DISTINCTCOUNT(fact_production_coil[width_mm])
VAR UniqueThickness = DISTINCTCOUNT(fact_production_coil[thickness_mm])
VAR UniqueGrades = DISTINCTCOUNT(fact_production_coil[Grade])
VAR UniqueTypes = DISTINCTCOUNT(fact_production_coil[type_code])
RETURN
    (UniqueWidths * 0.3) +
    (UniqueThickness * 0.3) +
    (UniqueGrades * 0.2) +
    (UniqueTypes * 0.2)
```

```dax
Unique Product Combinations =
DISTINCTCOUNT(
    fact_production_coil[thickness_mm] & "|" &
    fact_production_coil[width_mm] & "|" &
    fact_production_coil[Grade]
)
```

Type-Specific Metrics

```dax
HL Pieces =
CALCULATE(
    COUNTROWS(fact_production_coil),
    fact_production_coil[type_code] = "HL"
)
```

```dax
HX Pieces =
CALCULATE(
    COUNTROWS(fact_production_coil),
    fact_production_coil[type_code] = "HX"
)
```

```dax
Prime Type Distribution =
VAR HLCount = [HL Pieces]
VAR TotalPrime = [Prime Pieces]
RETURN
    DIVIDE(HLCount, TotalPrime, 0) * 100
```

---

8. TIME INTELLIGENCE

Period Comparisons

```dax
Previous Period Pieces =
CALCULATE(
    [Total Pieces],
    DATEADD(fact_production_coil[production_date], -1, MONTH)
)
```

```dax
Pieces MoM Change =
[Total Pieces] - [Previous Period Pieces]
```

```dax
Pieces MoM % =
DIVIDE(
    [Pieces MoM Change],
    [Previous Period Pieces],
    0
) * 100
```

```dax
Previous Year Pieces =
CALCULATE(
    [Total Pieces],
    DATEADD(fact_production_coil[production_date], -1, YEAR)
)
```

```dax
Pieces YoY Change =
[Total Pieces] - [Previous Year Pieces]
```

```dax
Pieces YoY % =
DIVIDE(
    [Pieces YoY Change],
    [Previous Year Pieces],
    0
) * 100
```

Rolling Averages

```dax
Rolling 7D Tempo =
CALCULATE(
    [Pieces per Hour],
    DATESINPERIOD(
        fact_production_coil[production_date],
        LASTDATE(fact_production_coil[production_date]),
        -7,
        DAY
    )
)
```

```dax
Rolling 30D Tempo =
CALCULATE(
    [Pieces per Hour],
    DATESINPERIOD(
        fact_production_coil[production_date],
        LASTDATE(fact_production_coil[production_date]),
        -30,
        DAY
    )
)
```

```dax
Rolling 7D Avg Cycle =
CALCULATE(
    [Avg Cycle Time],
    DATESINPERIOD(
        fact_production_coil[production_date],
        LASTDATE(fact_production_coil[production_date]),
        -7,
        DAY
    )
)
```

Period Aggregations

```dax
MTD Pieces =
CALCULATE(
    [Total Pieces],
    DATESMTD(fact_production_coil[production_date])
)
```

```dax
QTD Pieces =
CALCULATE(
    [Total Pieces],
    DATESQTD(fact_production_coil[production_date])
)
```

```dax
YTD Pieces =
CALCULATE(
    [Total Pieces],
    DATESYTD(fact_production_coil[production_date])
)
```

```dax
MTD vs Target % =
VAR MTDActual = [MTD Pieces]
VAR DaysInMonth = DAY(EOMONTH(TODAY(), 0))
VAR DaysElapsed = DAY(TODAY())
VAR MonthlyTarget = 15000  // Adjust based on plant target
VAR ProRataTarget = (MonthlyTarget / DaysInMonth) * DaysElapsed
RETURN
    DIVIDE(MTDActual, ProRataTarget, 0) * 100
```

---

9. ADVANCED CALCULATIONS

Statistical Measures

```dax
Cycle Time CV =
// Coefficient of Variation
VAR Mean = [Avg Cycle Time]
VAR StdDev = [StdDev Cycle Time]
RETURN
    DIVIDE(StdDev, Mean, 0) * 100
```

```dax
Cycle Time Percentile 90 =
PERCENTILEX.INC(
    fact_production_coil,
    fact_production_coil[total_cycle_time_min],
    0.90
)
```

```dax
Cycle Time Percentile 95 =
PERCENTILEX.INC(
    fact_production_coil,
    fact_production_coil[total_cycle_time_min],
    0.95
)
```

```dax
Gap Outlier Count =
CALCULATE(
    COUNTROWS(fact_production_coil),
    fact_production_coil[gap_from_prev_completion_min] >
        [Avg Completion Gap] + (3 * STDEV.P(fact_production_coil[gap_from_prev_completion_min]))
)
```

Weighted Averages

```dax
Weighted Avg Cycle by Mass =
SUMX(
    fact_production_coil,
    fact_production_coil[total_cycle_time_min] * fact_production_coil[mass_out_tons]
) / [Total Mass Output]
```

```dax
Weighted Tempo by Shift Size =
VAR TotalShiftPieces =
    SUMX(
        VALUES(fact_production_coil[shift_code]),
        [Pieces This Shift]
    )
RETURN
    SUMX(
        VALUES(fact_production_coil[shift_code]),
        [Shift Tempo] * DIVIDE([Pieces This Shift], TotalShiftPieces, 0)
    )
```

Conditional Aggregations

```dax
Avg Cycle Excluding Outliers =
CALCULATE(
    AVERAGE(fact_production_coil[total_cycle_time_min]),
    fact_production_coil[total_cycle_time_min] >= [Cycle Time Percentile 90] * 0.1,
    fact_production_coil[total_cycle_time_min] <= [Cycle Time Percentile 90]
)
```

```dax
Prime Pieces Above Target Tempo =
CALCULATE(
    [Prime Pieces],
    fact_production_coil[total_cycle_time_min] < DIVIDE(60, [Tempo Target], 999)
)
```

Parent Coil Metrics

```dax
```

```
Avg Prime Pieces per Parent =
VAR ParentPrimeTable =
    SUMMARIZE(
        fact_production_coil,
        fact_production_coil[parent_coil_id],
        "PrimePcs",
            CALCULATE(
                COUNTROWS(fact_production_coil),
                fact_production_coil[is_prime] = TRUE
            )
    )
RETURN
    AVERAGEX(ParentPrimeTable, [PrimePcs])
```

```dax
Parent Coils with Low Yield =
CALCULATE(
    DISTINCTCOUNT(fact_production_coil[parent_coil_id]),
    FILTER(
        VALUES(fact_production_coil[parent_coil_id]),
        CALCULATE([Prime Rate %]) < 70
    )
)
```

```dax
Avg Parent Cycle Time =
AVERAGEX(
    VALUES(fact_production_coil[parent_coil_id]),
    CALCULATE(
        DATEDIFF(
            MIN(fact_production_coil[completion_ts]),
            MAX(fact_production_coil[completion_ts]),
            MINUTE
        )
    )
)
```

---

10. KPI INDICATORS

Status Indicators

```dax
```

```
Tempo Indicator =
VAR Achievement = [Tempo Achievement %]
RETURN
    SWITCH(
        TRUE(),
        Achievement >= 100, "✓ On Target",
        Achievement >= 95, "△ Near Target",
        Achievement >= 85, "⚠ Below Target",
        "✗ Critical"
    )
```

```dax
Utilization Indicator =
VAR Util = [Equipment Utilization %]
RETURN
    SWITCH(
        TRUE(),
        Util >= 80, "🟢 Excellent",
        Util >= 70, "🟡 Good",
        Util >= 60, "🟠 Fair",
        "🔴 Poor"
    )
```

```dax
Prime Rate Indicator =
VAR Rate = [Prime Rate %]
RETURN
    SWITCH(
        TRUE(),
        Rate >= 85, "🟢 Target",
        Rate >= 80, "🟡 Acceptable",
        Rate >= 75, "🟠 Monitor",
        "🔴 Action Required"
    )
```

Traffic Light Indicators

```dax
Tempo Traffic Light =
VAR Achievement = [Tempo Achievement %]
RETURN
    SWITCH(
        TRUE(),
        Achievement >= 95, "#00B050",  // Green
        Achievement >= 85, "#FFC000",  // Yellow
        "#FF0000"                      // Red
    )
```

```dax
Maintenance Traffic Light =
VAR Downtime = [Downtime %]
RETURN
    SWITCH(
        TRUE(),
        Downtime <= 10, "#00B050",  // Green
        Downtime <= 15, "#FFC000",  // Yellow
        "#FF0000"                   // Red
    )
```

Trend Indicators

```dax
Tempo Trend =
VAR CurrentTempo = [Pieces per Hour]
VAR PreviousTempo =
    CALCULATE(
        [Pieces per Hour],
        DATEADD(fact_production_coil[production_date], -7, DAY)
    )
RETURN
    SWITCH(
        TRUE(),
        CurrentTempo > PreviousTempo * 1.05, "↑ Improving",
        CurrentTempo < PreviousTempo * 0.95, "↓ Declining",
        "→ Stable"
    )
```

```dax
Trend Arrow =
VAR Current = [Pieces per Hour]
VAR Previous =
    CALCULATE(
        [Pieces per Hour],
        DATEADD(fact_production_coil[production_date], -1, MONTH)
    )
VAR PercentChange = DIVIDE(Current - Previous, Previous, 0)
RETURN
    SWITCH(
        TRUE(),
        PercentChange > 0.05, "⬆",
        PercentChange < -0.05, "⬇",
        "➡"
    )
```

---

11. HELPER MEASURES

Formatting Helpers

```dax
Format Cycle Time =
FORMAT([Avg Cycle Time], "#,##0.0") & " min"
```

```dax
Format Tempo =
FORMAT([Pieces per Hour], "#,##0.0") & " pcs/hr"
```

```dax
Format Percentage =
FORMAT([Prime Rate %], "#,##0.0") & "%"
```

```dax
Format Hours =
FORMAT([Total RUN Hours], "#,##0") & " hrs"
```

Conditional Measures

```dax
Show Bottleneck Data =
IF(
    MAX(dim_equipment[is_bottleneck_candidate]) = TRUE,
    [Equipment Time Share %],
    BLANK()
)
```

```dax
Show Prime Data Only =
IF(
    MAX(fact_production_coil[is_prime]) = TRUE,
    [Total Pieces],
    BLANK()
)
```

Selection Context

```dax
Selected Equipment Count =
COUNTROWS(VALUES(dim_equipment[equipment_name]))
```

```dax
Is Single Equipment Selected =
[Selected Equipment Count] = 1
```

```dax
Selected Date Range =
VAR MinDate = MIN(fact_production_coil[production_date])
VAR MaxDate = MAX(fact_production_coil[production_date])
RETURN
    IF(
        MinDate = MaxDate,
        FORMAT(MinDate, "DD MMM YYYY"),
        FORMAT(MinDate, "DD MMM") & " - " & FORMAT(MaxDate, "DD MMM YYYY")
    )
```

---

12. RANK AND TOP N

Equipment Rankings

```dax
Equipment Downtime Rank =
RANKX(
    ALL(dim_equipment),
    [Total FAULT Hours],
    ,
    DESC,
    DENSE
)
```

```dax
Equipment Utilization Rank =
RANKX(
    ALL(dim_equipment),
    [Equipment Utilization %],
    ,
    DESC,
    DENSE
)
```

```dax
Top 5 Bottleneck Equipment =
CALCULATE(
    [Total Operation Hours],
    TOPN(
        5,
        ALL(dim_equipment),
        [Equipment Bottleneck Score],
        DESC
    )
)
```

Shift Rankings

```dax
Shift Performance Rank =
RANKX(
    ALL(fact_production_coil[shift_code]),
    [Shift Efficiency %],
    ,
    DESC,
    DENSE
)
```

```dax
Best Performing Crew =
VAR BestShift =
    TOPN(
        1,
        SUMMARIZE(
            fact_production_coil,
            fact_production_coil[shift_code],
            "Tempo", [Shift Tempo]
        ),
```

        [Tempo],
        DESC
)
RETURN
    MAXX(BestShift, fact_production_coil[shift_code])

Product Rankings

```dax
Type Code Rank by Volume =
RANKX(
    ALL(fact_production_coil[type_code]),
    [Total Pieces],
    ,
    DESC,
    DENSE
)
```

```dax
Top 3 Product Types =
CONCATENATEX(
    TOPN(
        3,
        VALUES(fact_production_coil[type_code]),
        [Total Pieces],
        DESC
    ),
    fact_production_coil[type_code],
    ", "
)
```

APPENDIX: MEASURE ORGANIZATION

Suggested Folder Structure in Measures Table

```
📁 Measures
├── 📁 1. Production
│   ├── Total Pieces
│   ├── Prime Pieces
│   ├── Scrap Pieces
│   └── ...
```

```
├── 📁 2. Tempo & Gaps
│   ├── Pieces per Hour
│   ├── Tempo Target
│   ├── Avg Completion Gap
│   └── ...
├── 📁 3. Equipment
│   ├── Total RUN Hours
│   ├── Equipment Utilization %
│   └── ...
├── 📁 4. Bottlenecks
│   ├── Bottleneck Share %
│   ├── Equipment Bottleneck Score
│   └── ...
├── 📁 5. Maintenance
│   ├── MTBF
│   ├── MTTR
│   └── ...
├── 📁 6. Shifts
│   ├── Shift Tempo
│   ├── Shift Efficiency %
│   └── ...
├── 📁 7. Product Mix
│   ├── Fast Band %
│   ├── Product Mix Complexity
│   └── ...
├── 📁 8. Time Intelligence
│   ├── MTD Pieces
│   ├── YTD Pieces
│   └── ...
└── 📁 9. Helpers
    ├── Format Tempo
    ├── Selected Date Range
    └── ...
```

USAGE NOTES

Performance Tips

1. Use variables to store intermediate calculations
2. Avoid iterating over large tables unnecessarily
3. Use CALCULATE efficiently with clear filter context
4. Reference base measures rather than recalculating
5. Consider creating aggregation tables for complex calculations

Testing Measures

Always test new measures with:
- Empty filter context
- Single equipment selection
- Multiple equipment selection
- Date range filters
- Shift filters
- Combined filters

Documentation

For each custom measure:
- Add description in measure properties
- Document business logic
- Note any assumptions or dependencies
- Specify target values where applicable

VERSION CONTROL

Version: 1.0
Date: December 2024
Last Updated: December 2024
Next Review: March 2025

END OF DAX FORMULAS REFERENCE