Lesson 22: DAX Practice with ready-to-use measures/columns.

Assumptions: table Employee_Performance (EP); proper Date table related to EP[Hire_Date]. Rename column/table references if yours differ.

Helper measures (recommended once)

```
Employee Count := DISTINCTCOUNT ( Employee_Performance[Employment_id] )
Total Promotions := CALCULATE ( COUNTROWS ( Employee_Performance ), Employee_Performance[Promotions] > 0 )
Avg Satisfaction := AVERAGE ( Employee_Performance[Employee_Satisfaction_Score] )
Avg Salary := AVERAGE ( Employee_Performance[Monthly_Salary] )
```

1) Top Performer ID per Department (returns the Employment_id)

```
Top Performer ID (by Dept) :=
VAR T =
   TOPN (
      1,
      ADDCOLUMNS (
         VALUES ( Employee_Performance[Employment_id] ),
         "Score", CALCULATE ( MAX ( Employee_Performance[Performance_Score] ) )
      ),
      [Score], DESC
   )
RETURN
CONCATENATEX ( T, Employee_Performance[Employment_id], ", " )
```

Use in a matrix with Department on rows; it respects the department filter context.

2) YoY Promotion Growth (based on Hire_Date)

```
Promotions (This Year) :=
CALCULATE ( [Total Promotions], YEAR ( 'Date'[Date] ) = YEAR ( MAX ( 'Date'[Date] ) ) )

Promotions (LY) :=
CALCULATE ( [Total Promotions], SAMEPERIODLASTYEAR ( 'Date'[Date] ) )

Promotions YoY % :=
DIVIDE ( [Promotions (This Year)] - [Promotions (LY)], [Promotions (LY)] )
```

3) Avg Salary of Employees who Resigned ≤ 2 years

```
Avg Salary – Resigned <= 2y :=
CALCULATE (
   [Avg Salary],
   Employee_Performance[Resigned] = "Yes",
   Employee_Performance[Years_at_company] <= 2
)
```

4) Rank by Satisfaction within Department

(a) Measure (works nicely in visuals)

```
Rank by Satisfaction (in Dept) :=
RANKX (
   ALLEXCEPT ( Employee_Performance, Employee_Performance[Department] ),
   [Avg Satisfaction],    -- or AVERAGE(EP[Employee_Satisfaction_Score])
   ,
   DESC,
   DENSE
)
```

(b) Calculated column (static)

```
Rank by Satisfaction (Column) =
VAR Dept = Employee_Performance[Department]
RETURN
RANKX (
   FILTER ( Employee_Performance, Employee_Performance[Department] = Dept ),
   Employee_Performance[Employee_Satisfaction_Score],
   ,
   DESC,
   DENSE
)
```

5) Pearson correlation: Training_Hours vs Performance_Score

```
Correlation (Training vs Performance) :=
VAR T =
   SUMMARIZE (
      VALUES ( Employee_Performance[Employment_id] ),
      Employee_Performance[Employment_id],
      "x", CALCULATE ( AVERAGE ( Employee_Performance[Training_Hours] ) ),
      "y", CALCULATE ( AVERAGE ( Employee_Performance[Performance_Score] ) )
   )
VAR N   = COUNTROWS ( T )
```

```
VAR SX  = SUMX ( T, [x] )
VAR SY  = SUMX ( T, [y] )
VAR SXX = SUMX ( T, [x] * [x] )
VAR SYY = SUMX ( T, [y] * [y] )
VAR SXY = SUMX ( T, [x] * [y] )
RETURN
DIVIDE ( N * SXY - SX * SY,
    SQRT ( ( N * SXX - SX * SX ) * ( N * SYY - SY * SY ) ) )
```

6) % Employees with Frequent Remote Work (“Weekly” or “Daily”)

```
Remote Frequent % :=
VAR Num =
  CALCULATE (
    DISTINCTCOUNT ( Employee_Performance[Employment_id] ),
    Employee_Performance[Remote_Work_Frequency]  IN  {  "Weekly",
"Daily" }
  )
RETURN DIVIDE ( Num, [Employee Count] )
```

7) Consistently High Performance ≥ 4 over Tenure

With only the current row in HR_Analytics.csv, we simulate yearly scores by assuming the recorded Performance_Score represents each tenure year.

Flag measure (simulated):

```
Consistently High Performer (Sim) :=
VAR Tenure = MAX ( Employee_Performance[Years_at_company] )
VAR Score  = MAX ( Employee_Performance[Performance_Score] )
RETURN IF ( NOT ISBLANK ( Tenure ) && Score >= 4, 1, 0 )
```

If you have a yearly performance table (e.g., EP_PerfYear with one row per employee per year), use:

```
Consistently High Performer (True History) :=
VAR MinScore =
  CALCULATE (
    MIN ( EP_PerfYear[Performance_Score] ),
    ALLEXCEPT ( EP_PerfYear, EP_PerfYear[Employment_id] )
  )
RETURN IF ( MinScore >= 4, 1, 0 )
```

8) Dept-wise Salary Budget Utilization (against Budget table)

Budget table: Dept_Budget[Department], Dept_Budget[Annual_Budget] (create and relate by Department, or use lookup).

Total Salary (Dept) := SUM (Employee_Performance[Monthly_Salary])

Dept Budget :=
VAR Dept = SELECTEDVALUE (Employee_Performance[Department])
RETURN LOOKUPVALUE (Dept_Budget[Annual_Budget], Dept_Budget[Department], Dept)

Budget Utilization % :=
DIVIDE ([Total Salary (Dept)] * 12, [Dept Budget]) -- monthly → annual

9) Attrition Risk Index (custom score)
Calculated column (text label):

```
Attrition Risk =
VAR Sat = Employee_Performance[Employee_Satisfaction_Score]
VAR OT  = Employee_Performance[Overtime_Hours]
VAR SD  = Employee_Performance[Sick_Days]
RETURN
IF ( Sat < 3 && OT > 10 && SD > 5, "High",
    IF ( Sat < 4, "Medium", "Low" ) )
```

(If you prefer a numeric index, map High=3, Medium=2, Low=1 for sorting/CF).

10) Overworked but Unpromoted (count of employees)

```
Overworked & Unpromoted (Count) :=
VAR EmpSet =
   FILTER (
      SUMMARIZE ( Employee_Performance, Employee_Performance[Employment_id] ),
      CALCULATE ( MAX ( Employee_Performance[Work_Hours_per_Week] ) ) > 45
         && CALCULATE ( MAX ( Employee_Performance[Overtime_Hours] ) ) > 5
         && CALCULATE ( MAX ( Employee_Performance[Promotions] ) ) = 0
   )
RETURN COUNTROWS ( EmpSet )
```

The SUMMARIZE + MAX pattern ensures we count each person once even if there are multiple rows per employee.