Lesson 16 – Advanced Charting Techniques with ready-to-use steps and DAX.

(Assumptions: tables Cards from Card_data.csv and Sales from sales.csv. Columns: card_brand, card_type, card_limit, client_id, card_number, account_opened_date, expire_dates; Sales has customer_id, sales_date.)

0) One Date table for the model

Disable Auto Date/Time → create and mark a proper Date table.

```
Date =
VAR Base = CALENDARAUTO()
RETURN
ADDCOLUMNS (
    Base,
    "Year",      YEAR ( [Date] ),
    "MonthNo",   MONTH ( [Date] ),
    "Month",     FORMAT ( [Date], "MMM" ),
    "YearMonth", FORMAT ( [Date], "YYYY-MM" )
)
```

Relationships

Cards[account_opened_date] → Date[Date]

Cards[expire_dates] → (no direct relationship needed for charts if you drive visuals by Date, otherwise create a role-playing copy of Date for Expiry)

Sales[sales_date] → Date[Date]

1) Measures for Cards

```
-- Core
Total Card Limit := SUM ( Cards[card_limit] )
Clients (Distinct) := DISTINCTCOUNT ( Cards[client_id] )
Cards Count := COUNTROWS ( Cards )
Cards Count (by Open Date) := CALCULATE ( [Cards Count] )  -- alias

-- Drilldown (Year→Month) uses Date hierarchy: Date[Year] > Date[MonthNo] or Date[YearMonth]
-- Tooltip measure
Clients in Context := DISTINCTCOUNT ( Cards[client_id] )
```

Stacked Column: Total Card Limit by Brand & Type

X: Cards[card_brand]

Y: [Total Card Limit]

Legend: Cards[card_type]

Tooltip: [Clients in Context]

2) Drill down into monthly trends (issues over time)

Count cards issued:

Cards Issued :=
CALCULATE ([Cards Count], NOT ISBLANK (Cards[card_number]))

Create hierarchy in Date: Year > MonthNo (sort Month by MonthNo) or use YearMonth (Continuous).

Visual: Stacked column

Axis: Date[Year] → enable Drill Down; clicking month shows monthly bars.

Values: [Cards Issued] (or [Cards Count])

3) Top 10 Clients by Total Card Limit (Bar)
Total Limit by Client := CALCULATE ([Total Card Limit])

Axis: Cards[client_id]

Values: [Total Limit by Client]

Filters pane → Top N → Top 10 by [Total Limit by Client]

Sort by value descending.

4) Client Drill-through page

Drill-through Page (create a new report page):

Add Drill-through field: Cards[client_id]

Table visual with columns: card_type, card_brand, card_limit, card_number, expire_dates

Add slicers (optional): card_type, Year of expire_dates (use a role-playing Date table or a calculated year column).

Tip: If you prefer not to add a second Date table, add:

Expire Year (calc column) = YEAR (Cards[expire_dates])

Then use it in slicers/tables.

5) Heatmap Matrix of Expiry Trends

Supporting measure:

Cards Expiring := COUNTROWS (Cards) -- context is expire_dates

Matrix

Rows: Cards[card_brand]

Columns: Year of expire_dates
(either from a role-playing Date table linked to expire_dates, or a calc column Expire Year)

Values: [Cards Expiring]

Conditional formatting → Color scale (e.g., red = higher counts).

6) Dynamic Top-N Card Brands (user-driven)
Option A (Recommended): What-If Parameter

Modeling → New parameter → What-If: TopN_Brands (Min 1, Max 50, Increment 1, Default 5).
This creates a table TopN_Brands with [TopN_Brands Value] and a slicer.

Rank measure and visual filter:

```
Brand Rank by Limit :=
RANKX (
   ALLSELECTED ( Cards[card_brand] ),
   [Total Card Limit],
   ,
   DESC,
```

```
    DENSE
)

Show Brand (TopN) :=
VAR N = SELECTEDVALUE ( TopN_Brands[TopN_Brands Value], 5 )
RETURN IF ( [Brand Rank by Limit] <= N, 1, 0 )
```

Column/Bar Chart

Axis: Cards[card_brand]

Values: [Total Card Limit]

Filters (visual level): [Show Brand (TopN)] = 1
The slicer from the What-If parameter controls N.

(Option B: build your own disconnected table with numbers and replace the SELECTEDVALUE reference.)

7) Sales dataset – Average days between sales per customer
A) Calculated columns (simplest & fast to learn)

```
-- Previous sale date per row
Prev Sales Date :=
VAR thisDate = Sales[sales_date]
VAR cust = Sales[customer_id]
RETURN
CALCULATE (
    MAX ( Sales[sales_date] ),
    FILTER ( ALL ( Sales ),
        Sales[customer_id] = cust && Sales[sales_date] < thisDate )
)

-- Day gap to previous sale
Days Since Prev :=
VAR prevD = Sales[Prev Sales Date]
RETURN IF ( ISBLANK ( prevD ), BLANK(), DATEDIFF ( prevD, Sales[sales_date], DAY ) )
```

Measure (average per customer):

```
Avg Days Between Sales :=
AVERAGE ( Sales[Days Since Prev] )
```

Use this measure in a table by customer_id (or in a Card with a filter for one customer).

Your example (05-05, 05-01, 04-21) gives gaps 4 and 10 → average 7.

B) Measure-only pattern (optional)

If you prefer a measure without calc columns:

```
Avg Days Between Sales (Measure Only) :=
VAR DatesTbl =
  CALCULATETABLE ( VALUES ( Sales[sales_date] ), REMOVEFILTERS ( Date ) )
VAR Ordered =
  ADDCOLUMNS ( DatesTbl, "RankD", RANKX ( DatesTbl, Sales[sales_date], , ASC ) )
VAR Gaps =
  ADDCOLUMNS (
    FILTER ( Ordered, [RankD] > 1 ),
    "PrevDate",
      CALCULATE ( MAX ( Sales[sales_date] ),
        FILTER ( Ordered, [RankD] = EARLIER ( [RankD] ) - 1 ) ),
    "Gap", DATEDIFF ( [PrevDate], Sales[sales_date], DAY )
  )
RETURN
AVERAGEX ( Gaps, [Gap] )
```

(Place customer_id on rows so the measure evaluates per customer.)