

Nguyen Van Vien

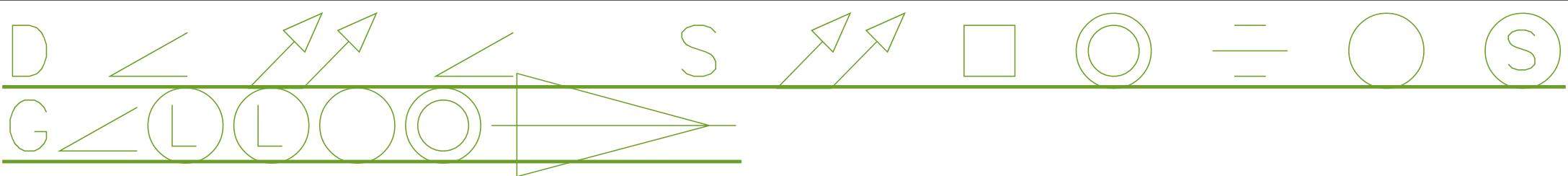
Email: viennv@ueh.edu.vn

Mobile: 0973 225 589



The goal is to turn data into information, and information into insight.

—Carly Fiorina

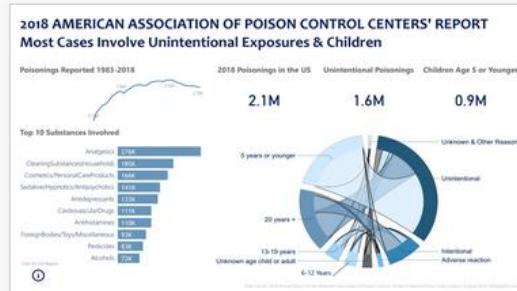


COVID-19 NSW Transport Impact

Tings

Featured

75



#ProjectHealthViz August 2020

Regiso

● Featured

45

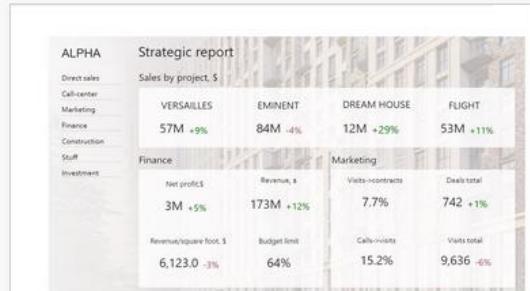


HANDBOL - ASOBAL

salabip1969

⟳ Featured

34



Property developer's sales analytics

alexkolokolov

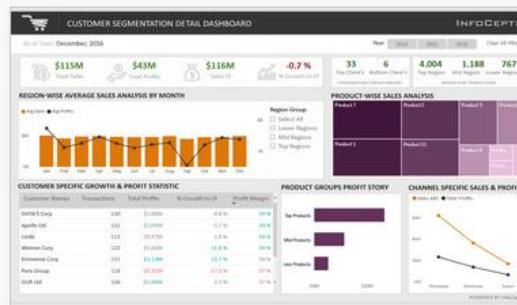
530



Construction management



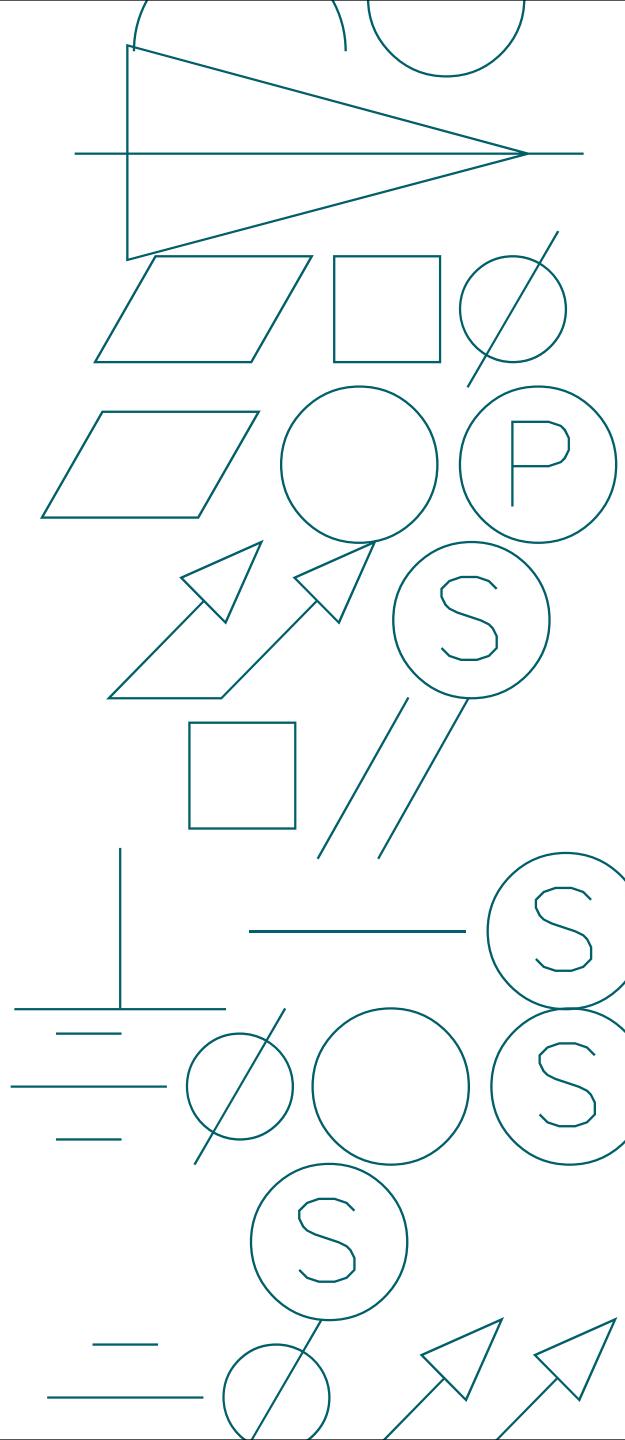
Airport Authority Performance Summary



Customer Analysis Dashboard



Trade in Europe



Domain

Data

Model

Analysis

Visualization



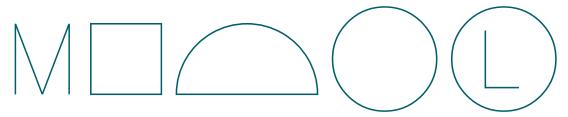
- A **domain** is simply the **context** where business intelligence is applied. Most businesses are composed of relatively standard business functions or departments, such as the following:
 - Sales
 - Marketing
 - Manufacturing/production
 - Supply chain/operations
 - Research and development
 - Human resources
 - Accounting/finance

D <--> (1)

- Once a domain has been decided upon, the next step is identifying and acquiring the **data** that's pertinent to that domain. This means identifying the sources of relevant data. These sources may be internal or external to an organization and may be structured, unstructured, or semi-structured in nature.
- Internal and external data
 - Internal data** is data that is generated within an organization by its business processes and operations.
 - External data** is data that is generated outside the boundaries of an organization's operations.

DATA (2)

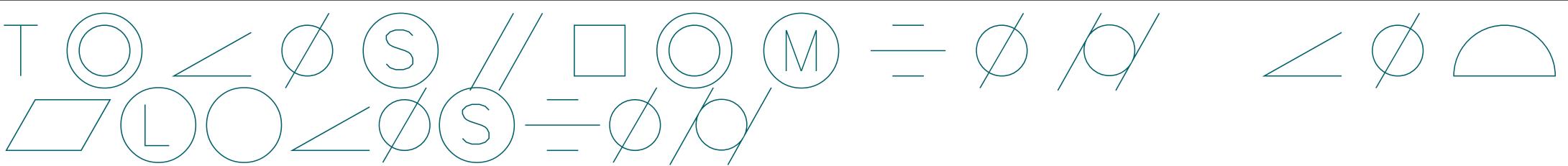
- Structured, unstructured, and semi-structured data
 - **Structured data** is data that conforms to a rather formal specification of tables with rows and columns: Open Database Connectivity (ODBC) and Object Linking and Embedding Database (OLE DB)
 - **Unstructured data** is effectively the opposite of structured data. Unstructured data cannot be organized into simple tables with rows and columns. Such data includes things such as video, audio, images, and text. This type of data is either stored as Binary Large Objects (BLOBs), online files or posts, or as files in a file system, such as the New Technology File System (NTFS) or the Hadoop Distributed File System (HDFS).
 - **Semi-structured data** has a structure but does not conform to the formal definition of structured data, that is, tables with rows and columns. Examples of semi-structured data include tab and delimited text files, XML, other markup languages such as HTML and XSL, JavaScript Object Notation (JSON), and Electronic Data Interchange (EDI).



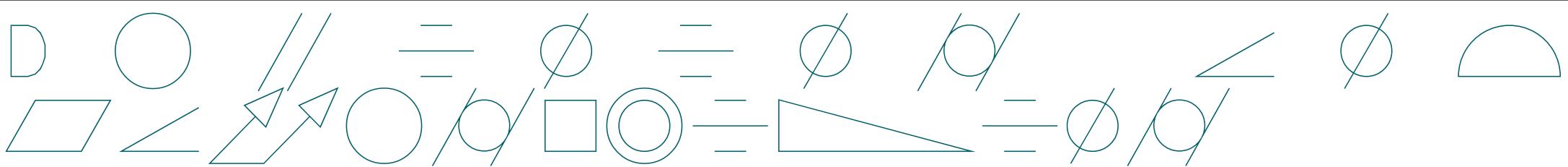
- A **model**, or data model, refers to the way in which one or more data sources are organized to support analysis and visualization. Models are built by transforming and cleansing data, helping to define the types of data within those sources, as well as the definition of data categories for specific data types. Building a model generally involves three elements:
 - Organizing
 - Transforming and cleansing
 - Defining and categorizing



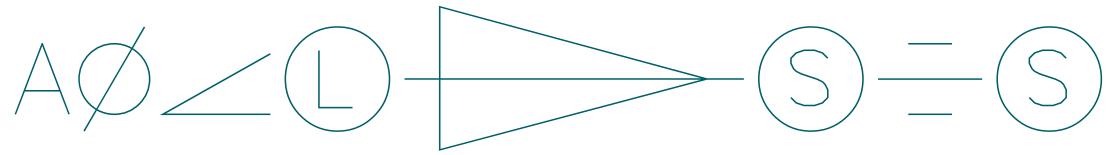
- The model becomes more complex as the various sources and tables of data must be combined into a cohesive whole. This is done by defining how each of the disparate sources of data relates to one another.
- As an example, let's say you have one data source that represents a customer's name, contact information, and perhaps the size of the business by revenue and/or the number of employees. This information might come from an organization's **Customer Relationship Management (CRM)** system. The second source of data might be order information, which includes the customer's name, units purchased, and the price that was paid. This second source of data comes from the organization's **Enterprise Resource Planning (ERP)** system. These two sources of data can be related to one another based on the unique name or ID of the customer.



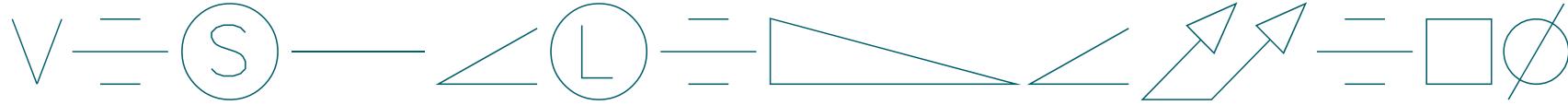
- Data is never clean – it must always be massaged for bad data to be removed or resolved.
- In addition, data may have errors, missing data, inconsistent formatting, or even have something as seemingly simple as trailing spaces.
- Business intelligence tools such as Power BI provide mechanisms for cleansing and reshaping the data to support analysis. This might involve replacing or removing errors in the data, pivoting, unpivoting, or transposing rows and columns, removing trailing spaces, or other types of transformation operations.



- Data models also formally define **the types of data within each table**. Data types generally include formats such as text, decimal number, whole number, percentage, date, time, date and time, duration, true/false, and binary. The definition of these data types is important as it defines what kind of analysis can be performed on the data.
- Data models also define **the data category of data types**. While a data type such as a postal code might be numeric or text, it is important for the model to define that the numeric data type represents a postal code. This further defines the type of analysis that can be performed upon this data, such as plotting the data on a map. Similarly, it might be important for the data model to define that a text data type represents a web or image **Uniform Resource Locator (URL)**. Typical data categories include such things as address, city, state, province, continent, country, region, place, county, longitude, latitude, postal code, web URL, image URL, and barcode.

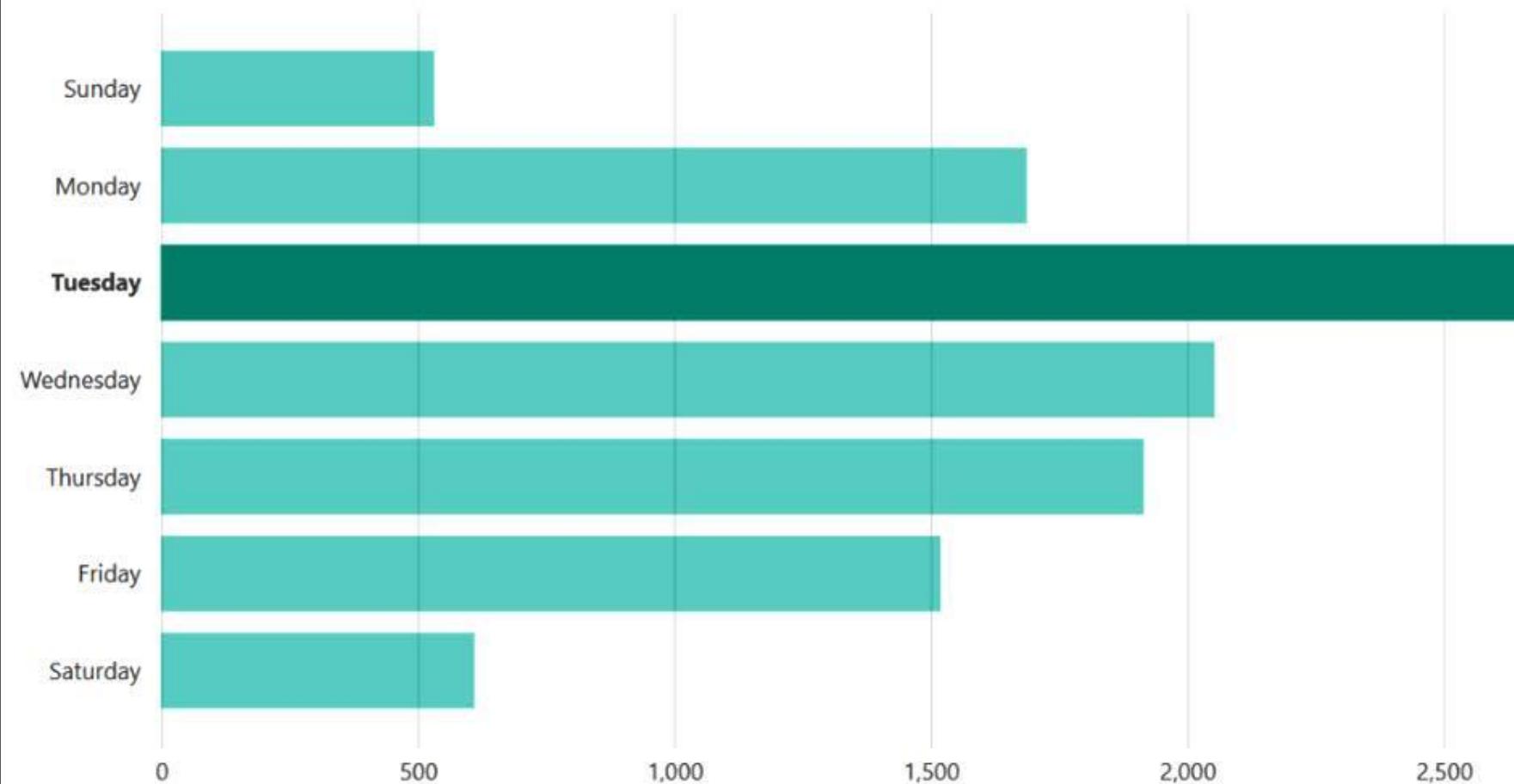


- This is a key process within business intelligence as this is when you attempt to answer questions **that are relevant to the business using internal and external data**. Simply having data about sales is not immediately useful to a business.
- Many times, organizations have, or wish to have, **Key Performance Indicators (KPIs)**, which are tracked by the business to help determine the organization's health or performance. KPIs might include such things as employee retention rate, net promoter score, new customer acquisitions per month, gross margin, and **Earnings Before Interest, Tax, Depreciation, and Amortization (EBITDA)**. Such KPIs generally require that the data is aggregated, has calculations performed on it, or both. These aggregations and calculations are called metrics or measures and are used to identify trends or patterns that can inform business decision-making.

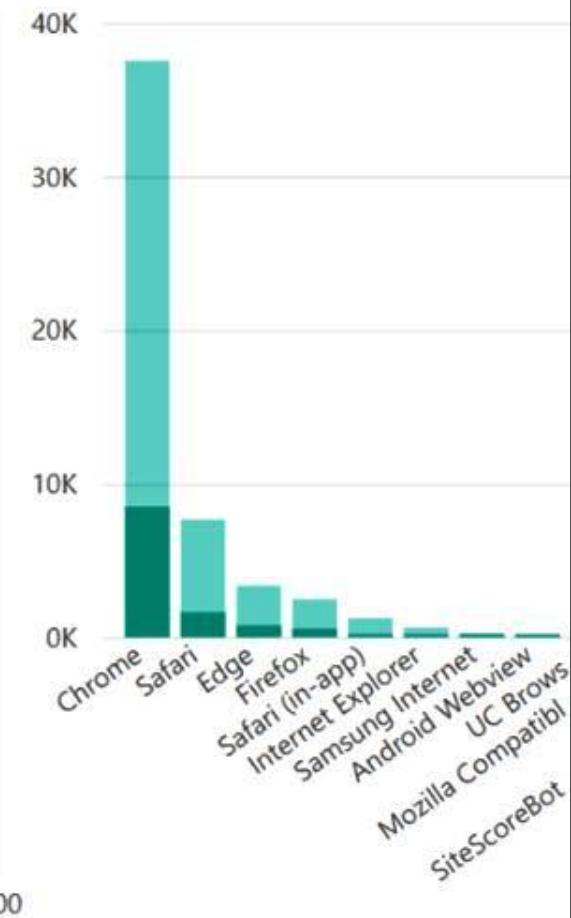


- Humans are visually oriented and thus it is advantageous to view the results of the analysis in the form of charts, reports, and dashboards.
- In the same way that a picture is worth a thousand words, visualizations allow thousands, millions, or even trillions of individual data points to be presented in a concise manner that is easily consumed and understandable.
- Visualization allows the analyst or report author to let the data tell a story. This story answers the questions that are originally posed by the business and thus delivers the insights that allow organizations to make better decisions.

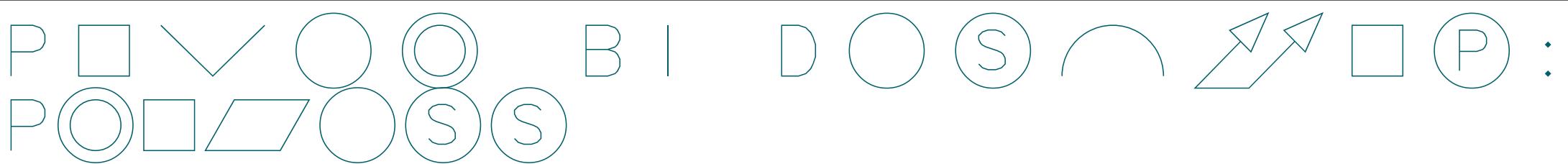
Sessions by DayOfWeek



Pageviews by Browser



Source: Deckler (2022)



- Getting data
- Creating a data model
- Analyzing data
- Creating and publishing reports

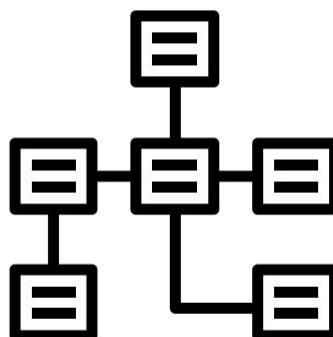
Data Sources

Power Query

Data Modeling

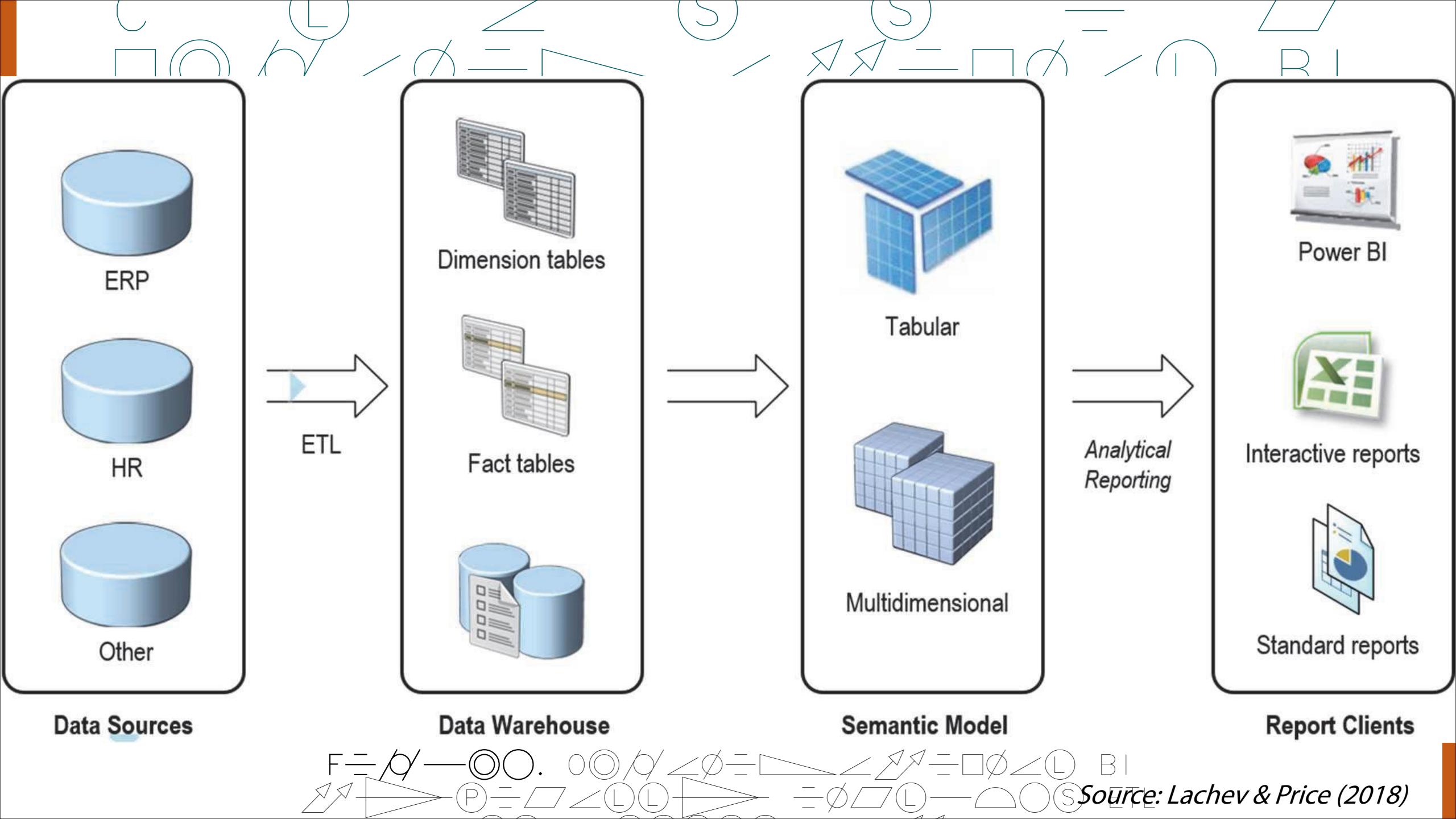
Dax Calculation

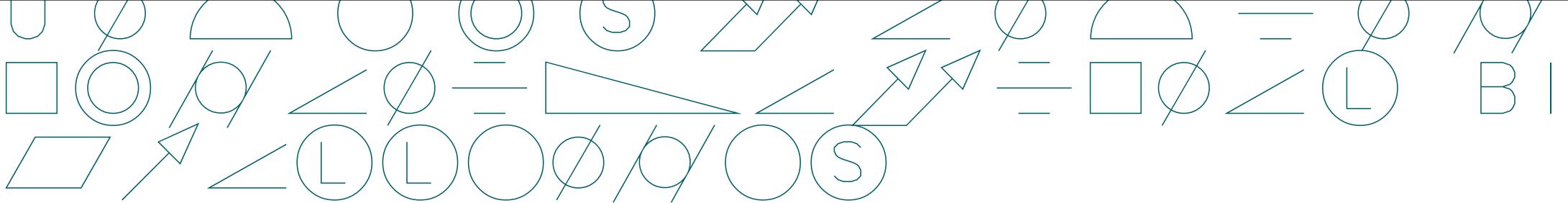
Visualization



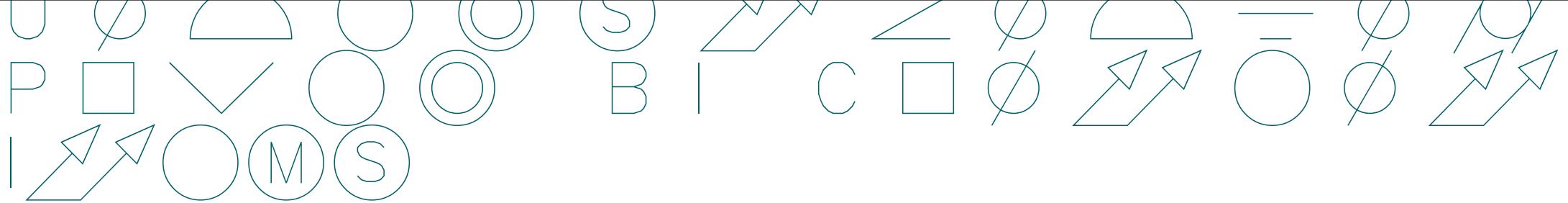


- Viewing and editing reports
- Creating dashboards
- Sharing and collaborating with others
- Accessing and creating apps
- Refreshing data





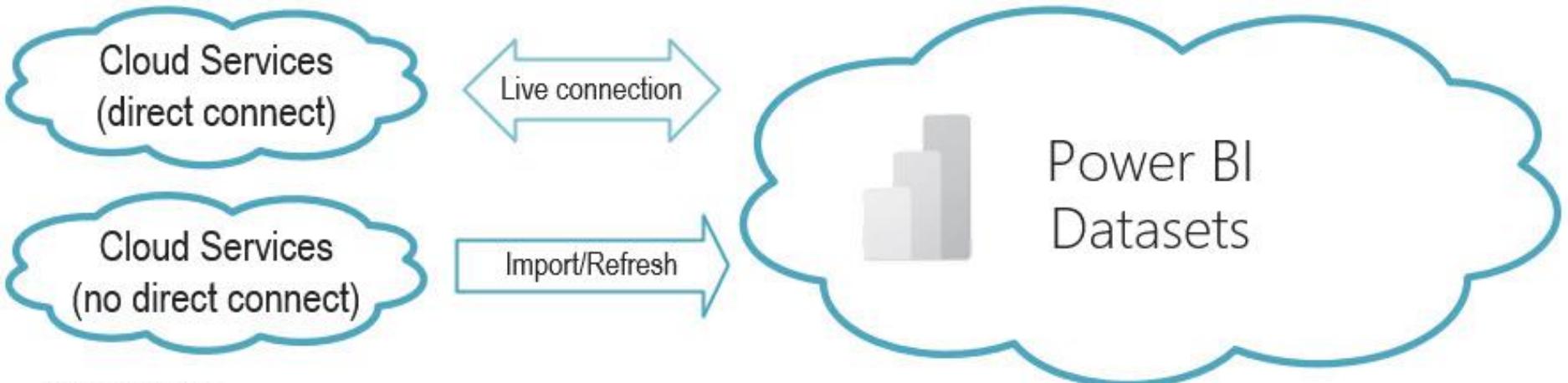
- **Upfront planning and implementation effort** - Depending on the data integration effort required, implementing an organizational BI solution might not be a simple task. Business users and IT pros must work together to derive requirements. Most of the implementation effort goes into data logistics processes to clean, verify, and load data.
- **Highly specialized skillset** - Organizational BI requires specialized talent, such as someone experienced in ETL, Analysis Services, and data warehousing. System engineers and developers must work together to plan the security, which sometimes might be more complicated than the actual BI solution.
- **Less flexibility** - Organization BI might not be flexible enough to react quickly to new or changing business requirements.



F = P / (1 - (1 + P)^{-n})

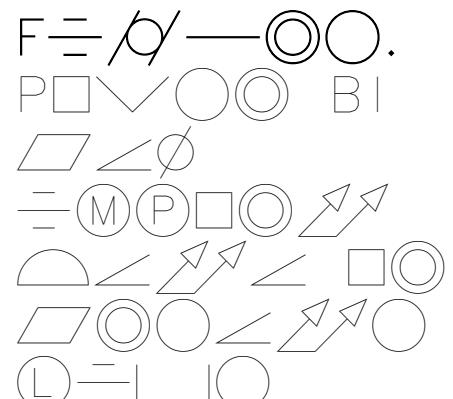
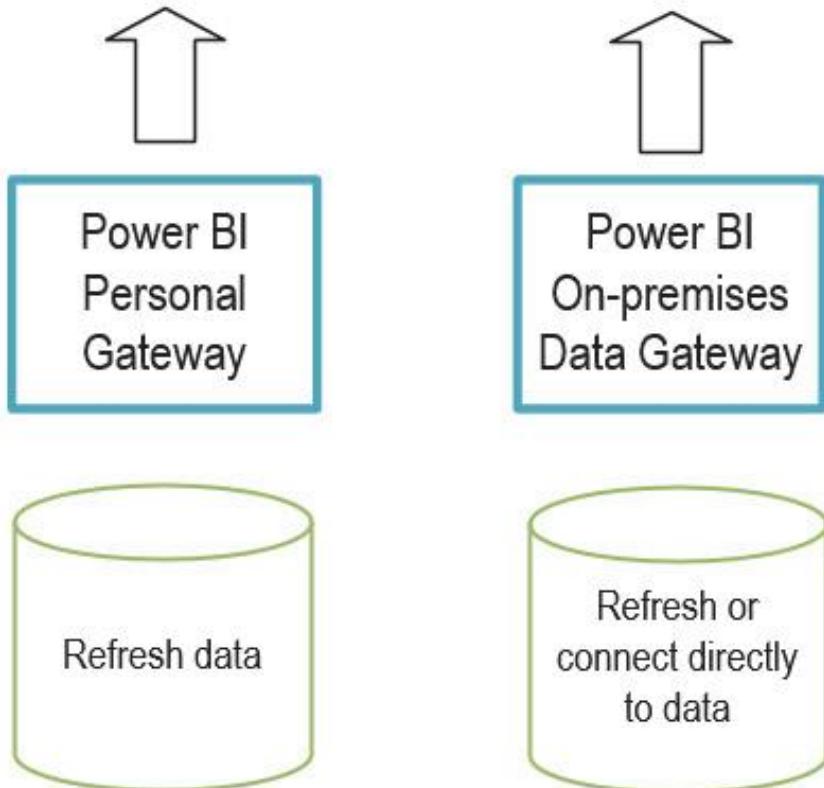
where F = Future Value, P = Periodic Payment, n = Number of Periods.

For example, if you make monthly payments of \$500 at 5% interest for 30 years, your future value would be approximately \$314,924.

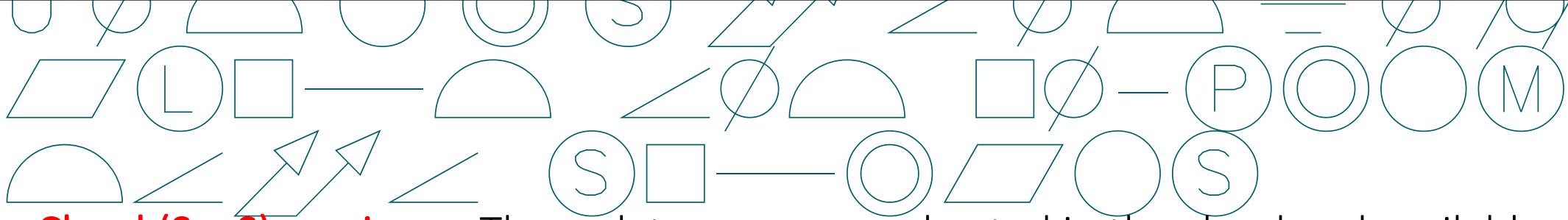


Cloud Data

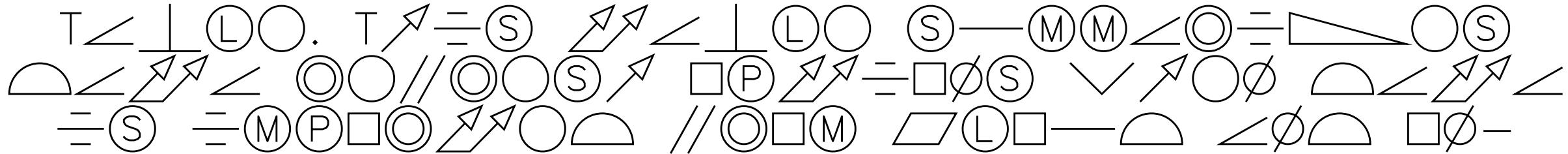
On-premises Data



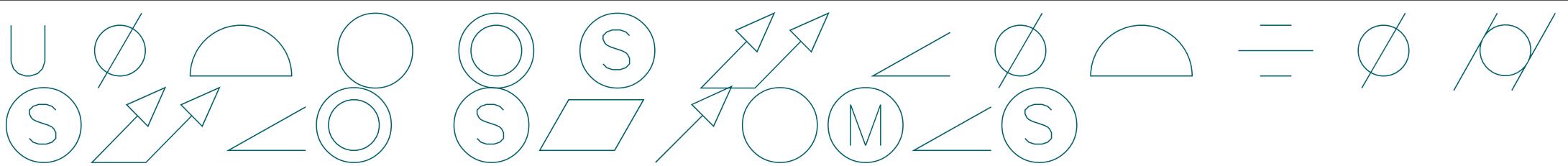
Source: Lachev & Price (2018)



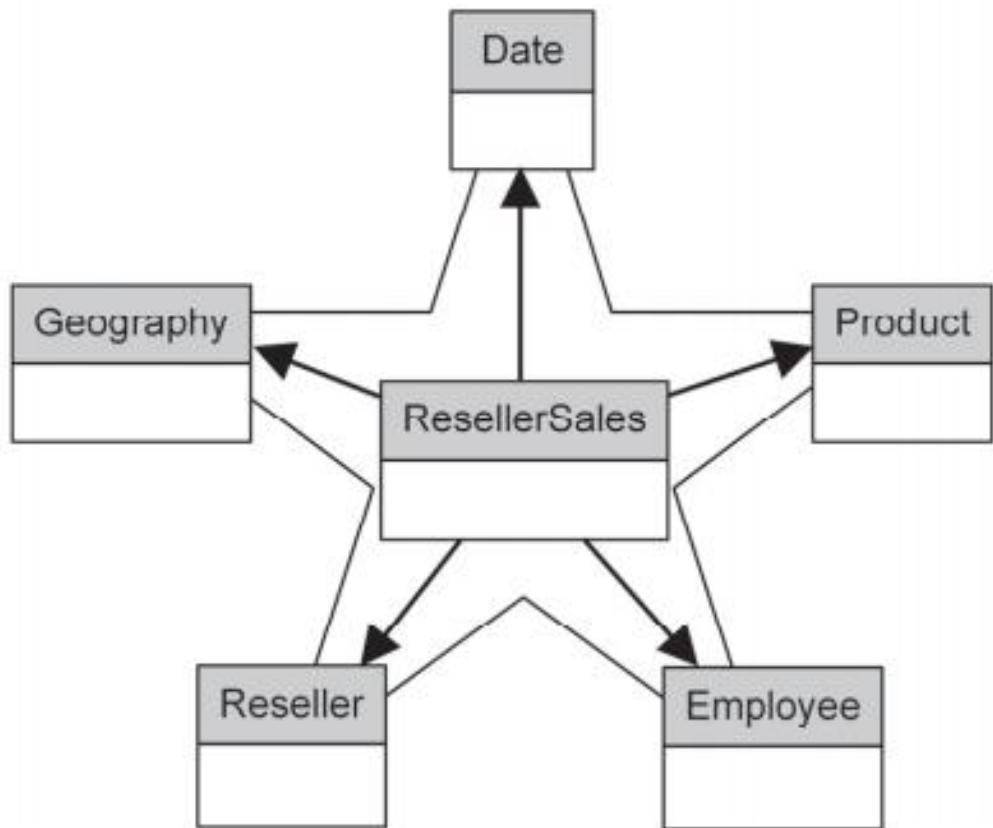
- **Cloud (SaaS) services** – These data sources are hosted in the cloud and available as online services. Examples of Microsoft cloud data sources that Power BI supports include OneDrive, Dynamics CRM, Azure SQL Database, Azure Synapse Analytics, and Spark on Azure HDInsight. Power BI can also access many popular cloud data sources from other vendors, such as Salesforce, Google Analytics, Marketo, and many others (the list is growing every month!).
- **On-premises data sources** – This category encompasses all other data sources that are internal to your organization, such as databases, cubes, Excel, and other files. For Power BI to access onpremises data sources, it needs a special connectivity software called a *gateway*.



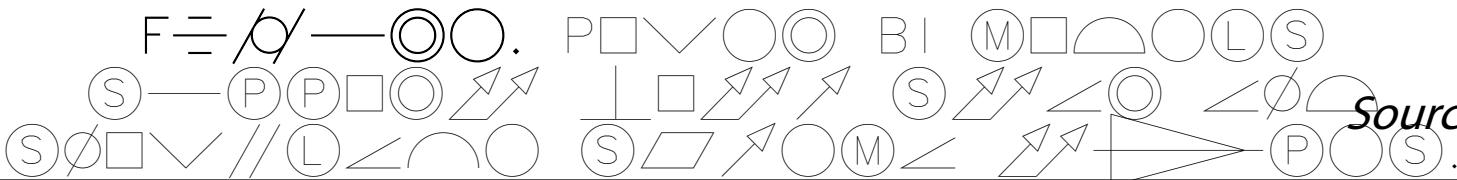
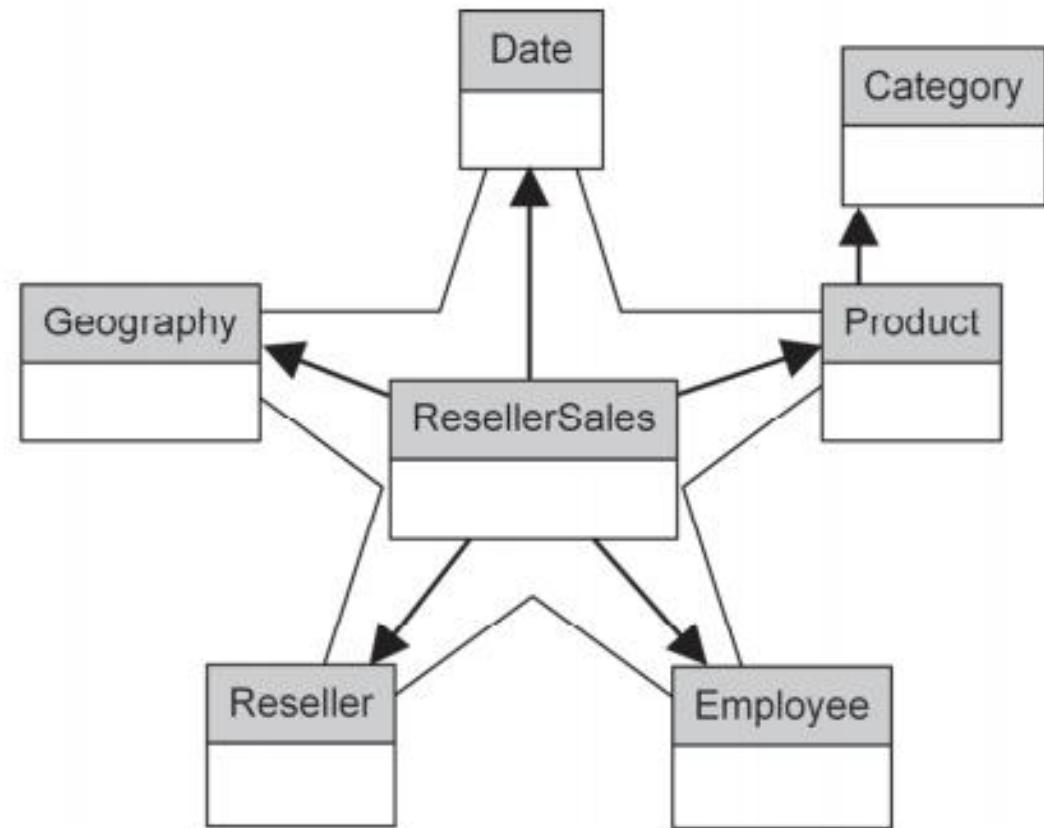
Location	Data Source	Refresh Type	Frequency
Cloud (Gateway not required)	Most cloud data sources, including Dynamics Online, Salesforce, Marketo, Zendesk, and many others.	Automatic	Once a day
	Excel, CSV, and Power BI Desktop files uploaded to OneDrive, OneDrive for Business, or SharePoint Online	Automatic	Once every hour
On premises (via gateway)	Supported data sources (see https://powerbi.microsoft.com/en-us/documentation/powerbi-refresh-data/)	Scheduled or manual	As configured by you up to 8/day or unlimited with Power BI Premium
	Excel 2013 (or later) Power Pivot data models with Power Query data connections or Power BI Desktop data models	Scheduled or manual	As configured by you up to 8/day or unlimited with Power BI Premium
	Local Excel files via Get Data in Power BI Service	Not supported	



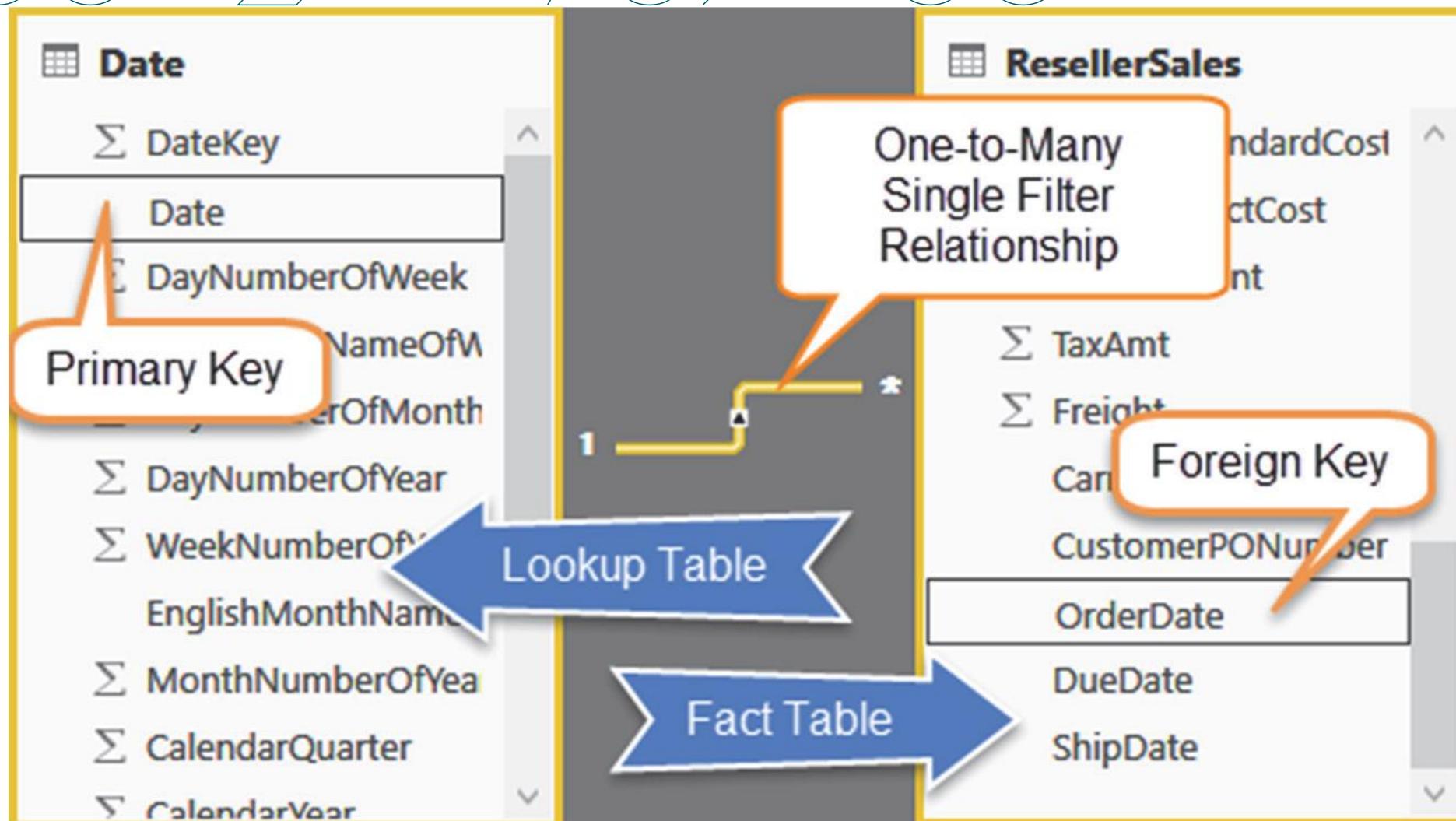
Star Schema



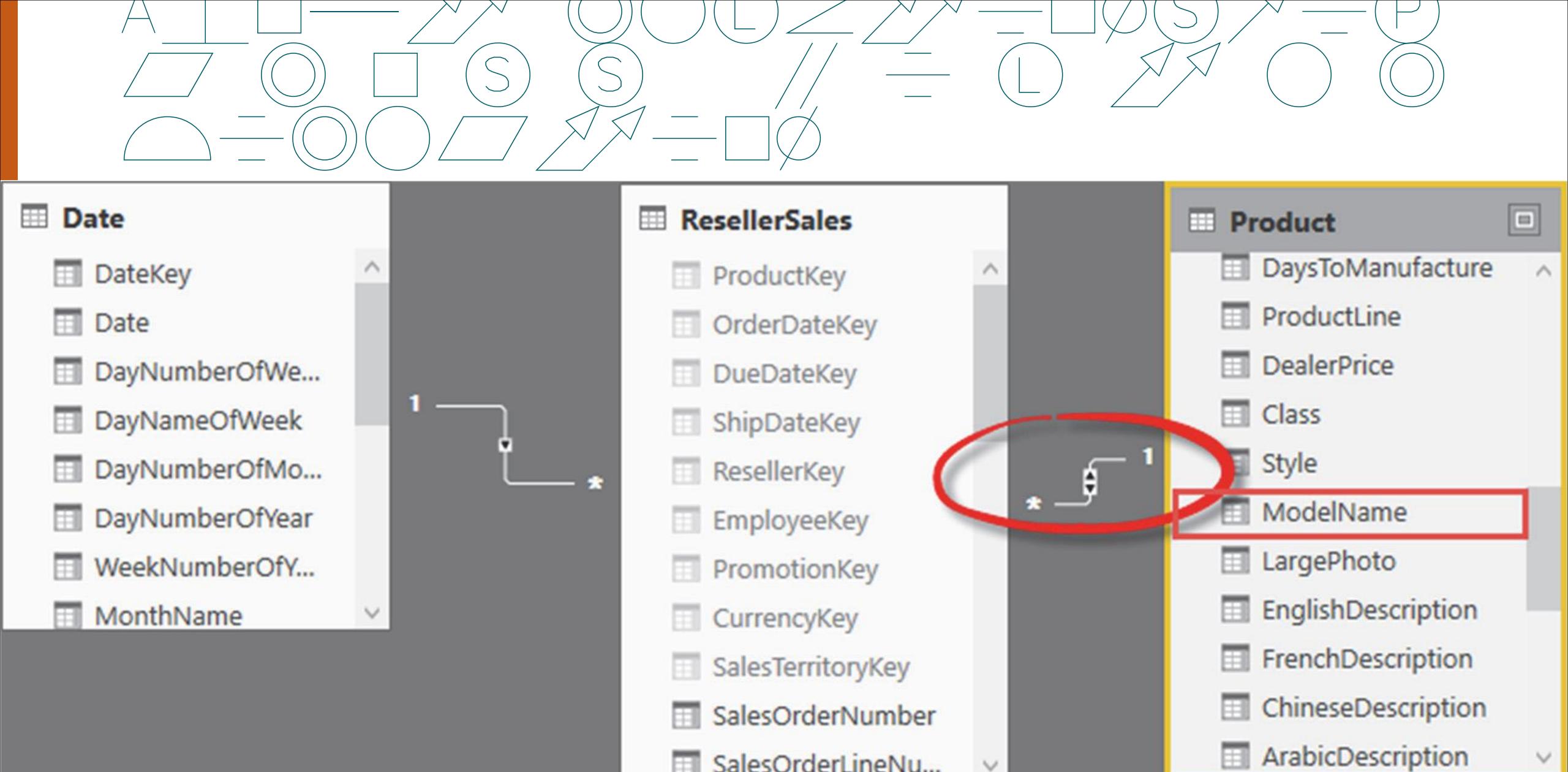
Snowflake Schema



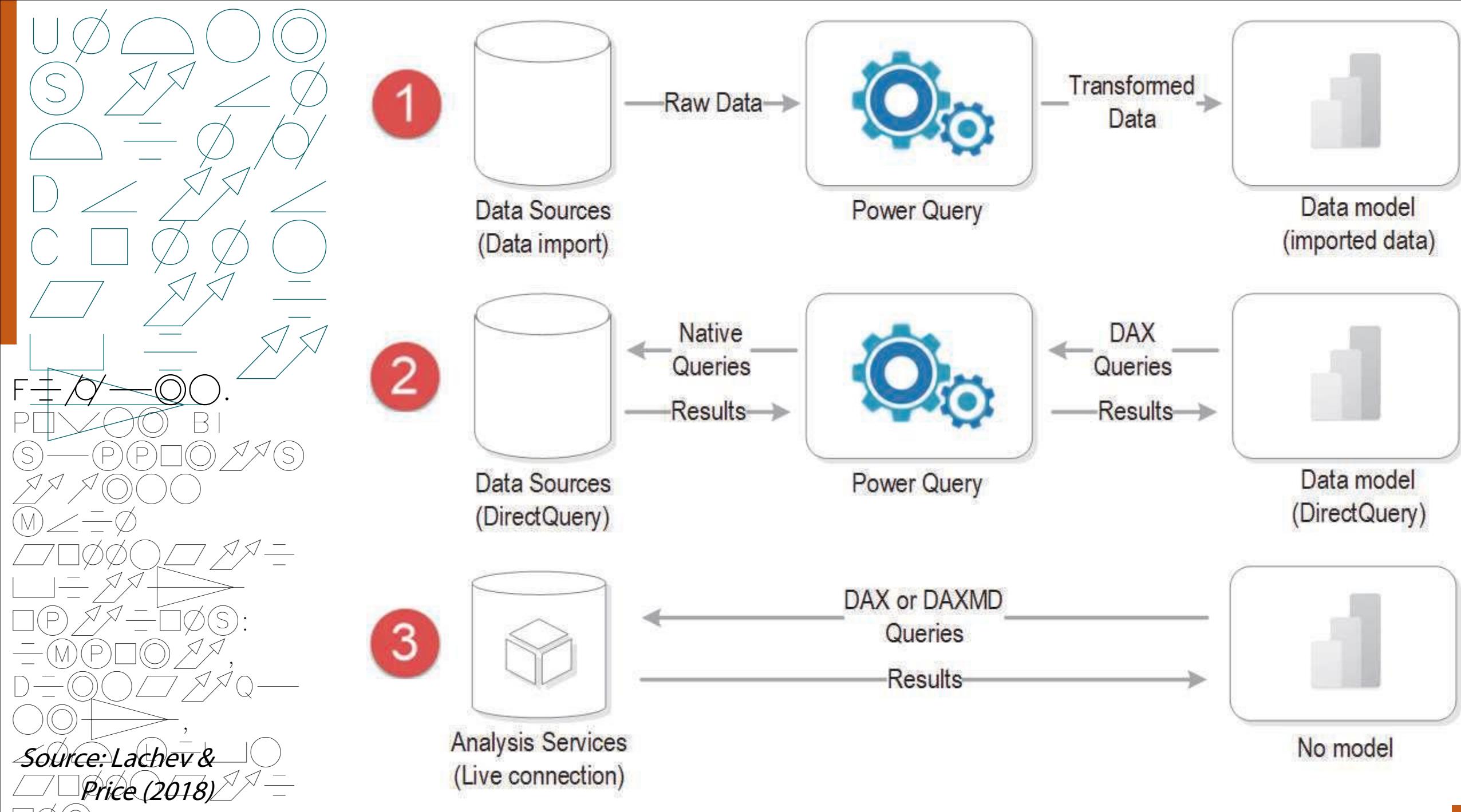
Source: Lachev & Price (2018)



Source: Lachev & Price (2018)

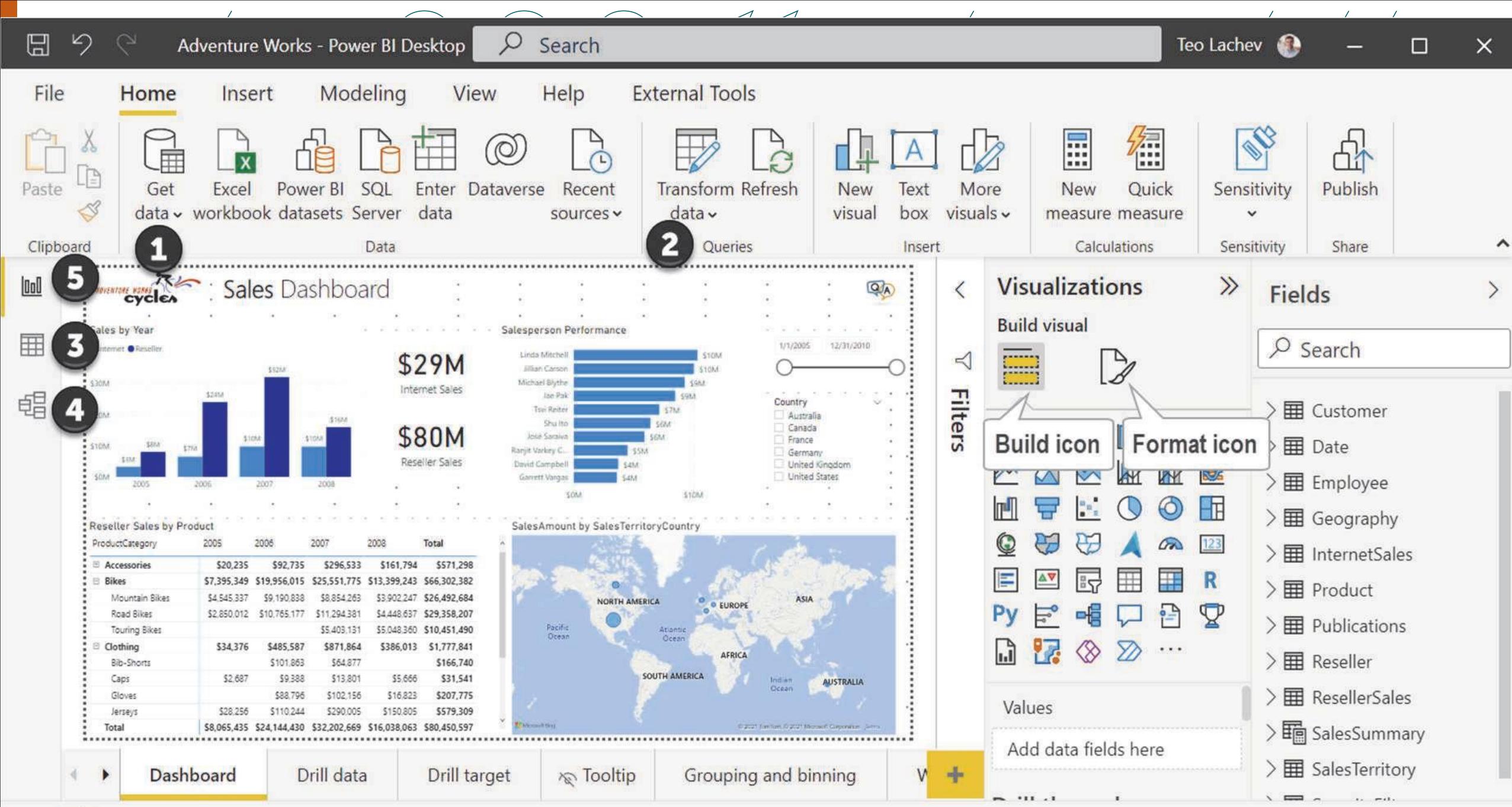


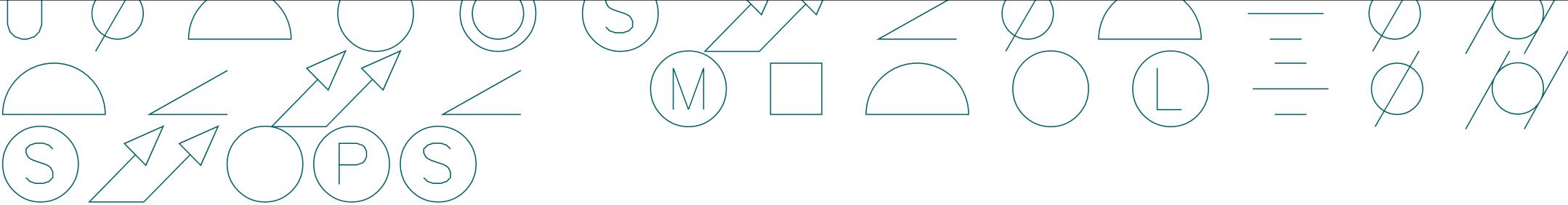
Source: Lachev & Price (2018)



Feature	Data import	DirectQuery	Connecting Live
Data sources	Any data source	SQL Server (on premises), Azure SQL Database, Azure Synapse Analytics, Spark on Azure HDInsight, Oracle, Teradata, Amazon Redshift, Impala, Snowflake, BigQuery	Analysis Services (Tabular and Multidimensional), published Power BI datasets, SAP Hana and DW (multidimensional connection)
Usage scenario	When you need all Power BI features and predictable performance	When you connect to a fast or large database that supports DirectQuery and you don't want to import data	When you connect to an existing model created by one of the above technologies.
Power Query for data transformations	Available (all features)	Available (basic transformations only)	Not available
Connect to multiple data sources	Yes	Yes	Yes, by switching local mode to DirectQuery (PBI datasets and AS only)
Implement hybrid storage (import and DirectQuery)	Yes	Yes	Yes, by switching local model to DirectQuery (PBI datasets and AS only)
Data storage	Data imported in xVelocity	Data is left at the data source	Data is left at the model
Connect to on-premises data from published models	Personal or standard gateway is required to refresh data	Standard (not personal) gateway is required to connect to data	Standard gateway is required to connect to on-prem SSAS models
Data modeling	Available	Available with limitations	Can't change the model but can augment it, such as with local measures
Relationships (Model tab)	Available	Available with limitations	Available but read-only
Business calculations in a data model	All DAX calculations	DAX calculations with limitations	DAX calculations both in the model and report (Tabular only)
Unsupported features	All features are supported	Clustering, Quick Insights, explain increase/decrease, Q&A, what-if, Key Influencers, Decomposition Tree (AI splits)	Model changes, binning, clustering, custom groups, Quick Insights, Explain increase/decrease, What-if, Key Influencers, Decomposition Tree limitations, synonyms
Impact on original data source	Reporting is handled internally by Power BI Desktop or Power BI Service when the model is deployed without making connections to the original data sources	Power BI Desktop/Power BI Service (published models) autogenerated native queries, sends them to the data source, and shows the results	Power BI Desktop/Power BI Service (published models) autogenerated queries, sends them to the data source, and shows the results

Source: Lachev & Price (2018)

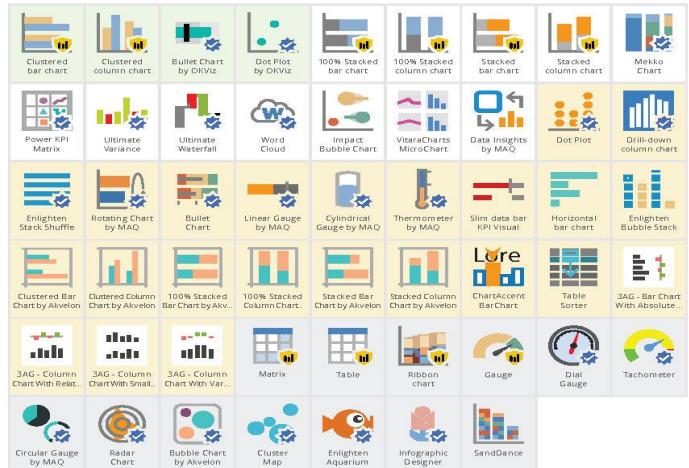




1. **Get data** – As in Power BI Service, the process starts with connecting to your data (the Get Data button).
2. **Transform data** – Remember that when you import the data or use DirectQuery, Power BI Desktop creates a query for each table you import (except when you connect live to certain data sources, in which case there is no underlying query). If the data requires cleansing or transformations, click "Transform data" to open Power Query Editor and perform data transformation tasks, such as replacing column values.
3. **Explore and refine data** – Switch to the Data tab to explore and refine the imported data, such as to see what data is loaded in a table and to change column data types. The Data tab isn't available with DirectQuery and live connections. However, in the case of DirectQuery data sources, you can still use the Power Query Editor to shape and transform the data, although not all transformations are available.
4. **Create relationships** – As a part of refining the data model, if you import multiple tables, you need to relate them either using the Model tab or the Manage Relationships button in the ribbon's Modeling tab.
5. **Visualize data** – Finally, you build reports to visualize your data and get insights by switching to the Report tab, which is available with all the data connectivity options.

COMPARISON

To compare the magnitude of measures



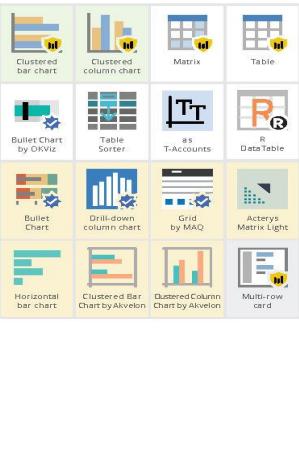
CHANGE OVER TIME

To display the changing trend of measures



RANKING

To rank measures in an order



SPATIAL

To display measures over spatial maps



PART-TO-WHOLE

To identify the parts making up a measure total



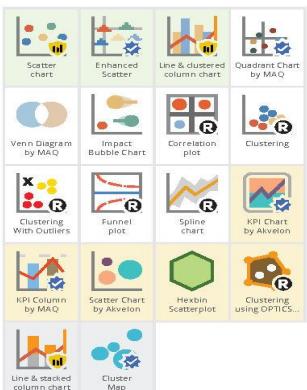
DISTRIBUTION

To display the distribution of values



CORRELATION

To show correlations between measures



SINGLE

To present single values



FILTER

To control report filters



NARRATIVE

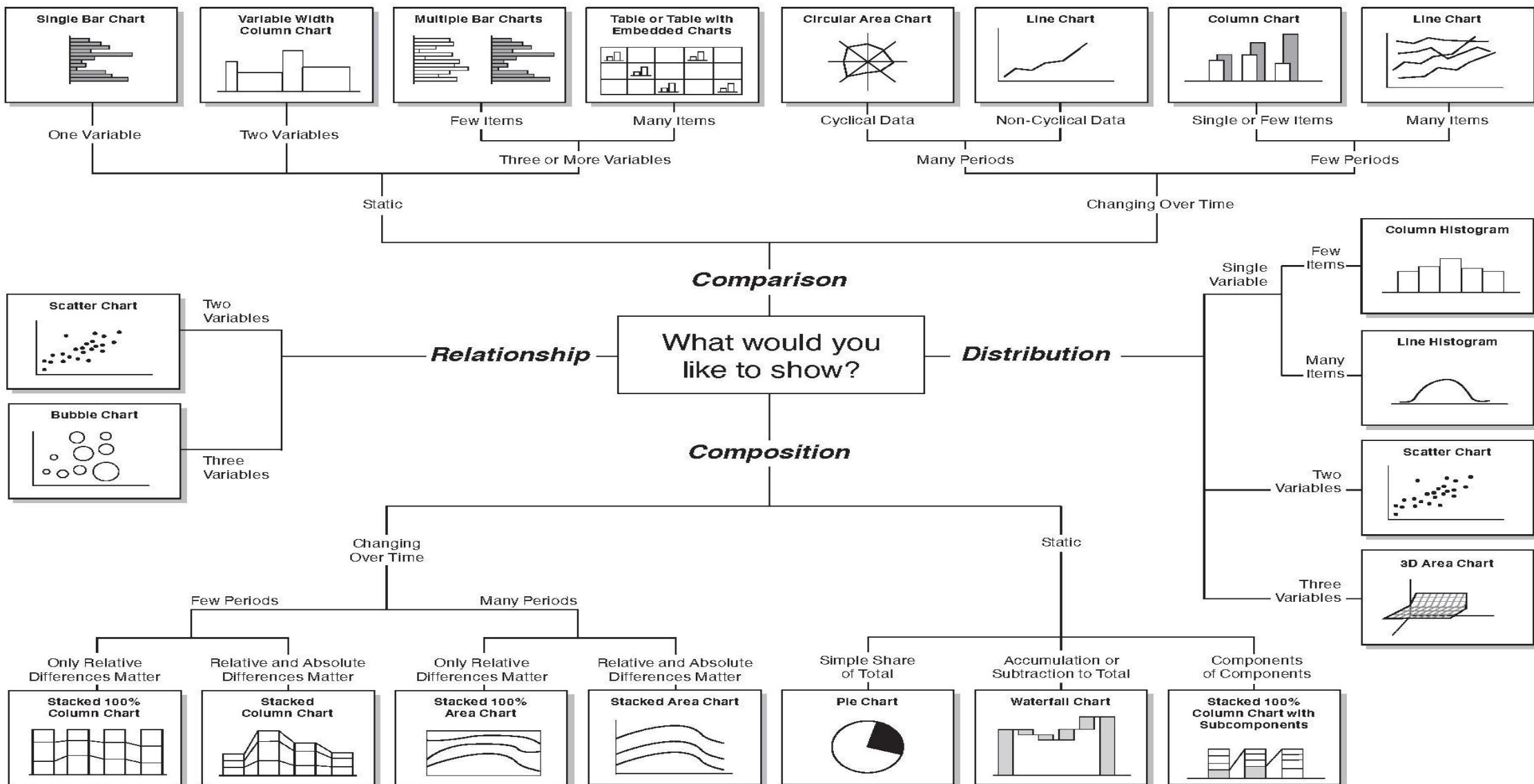
To tell a story with data

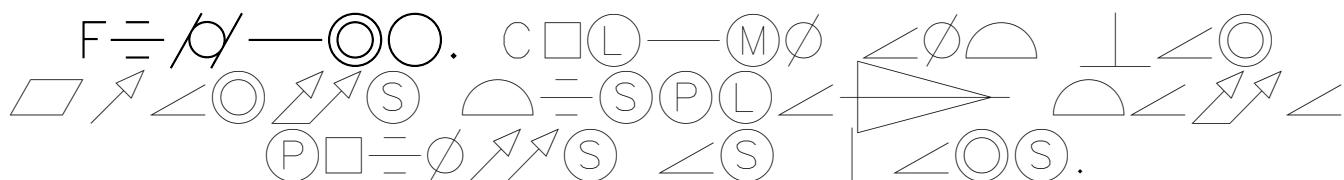
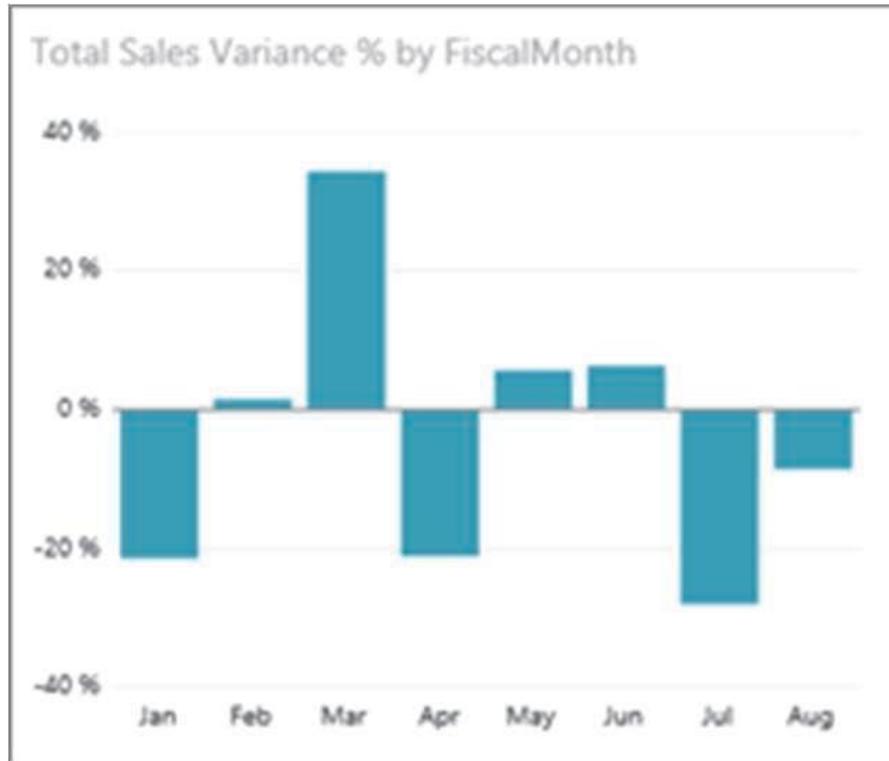
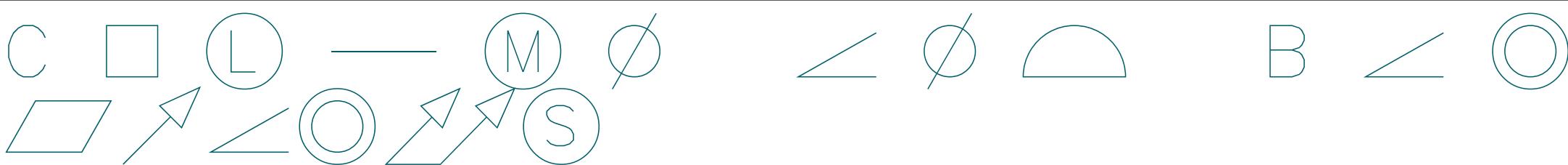


MISCELLANEOUS

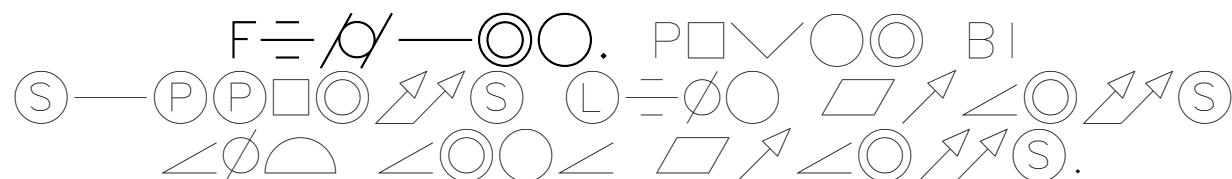
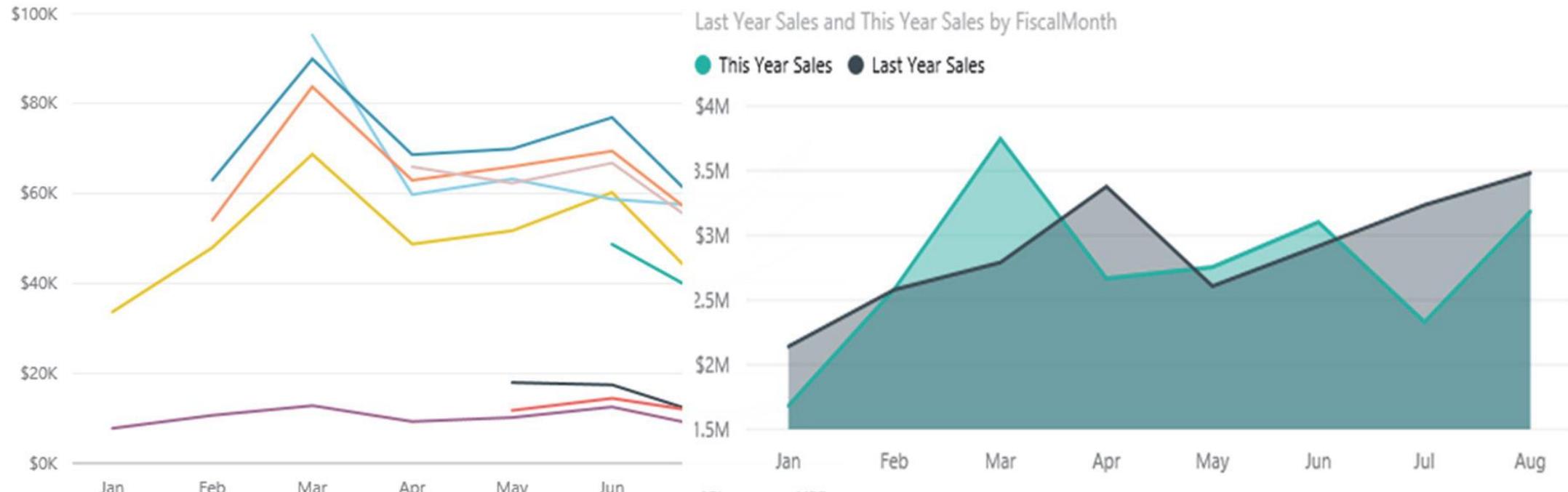


Chart Chooser





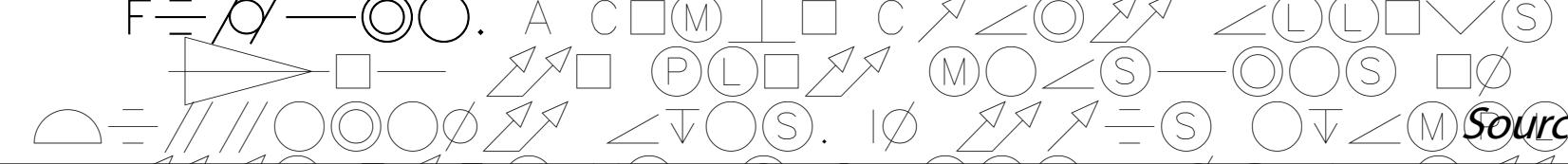
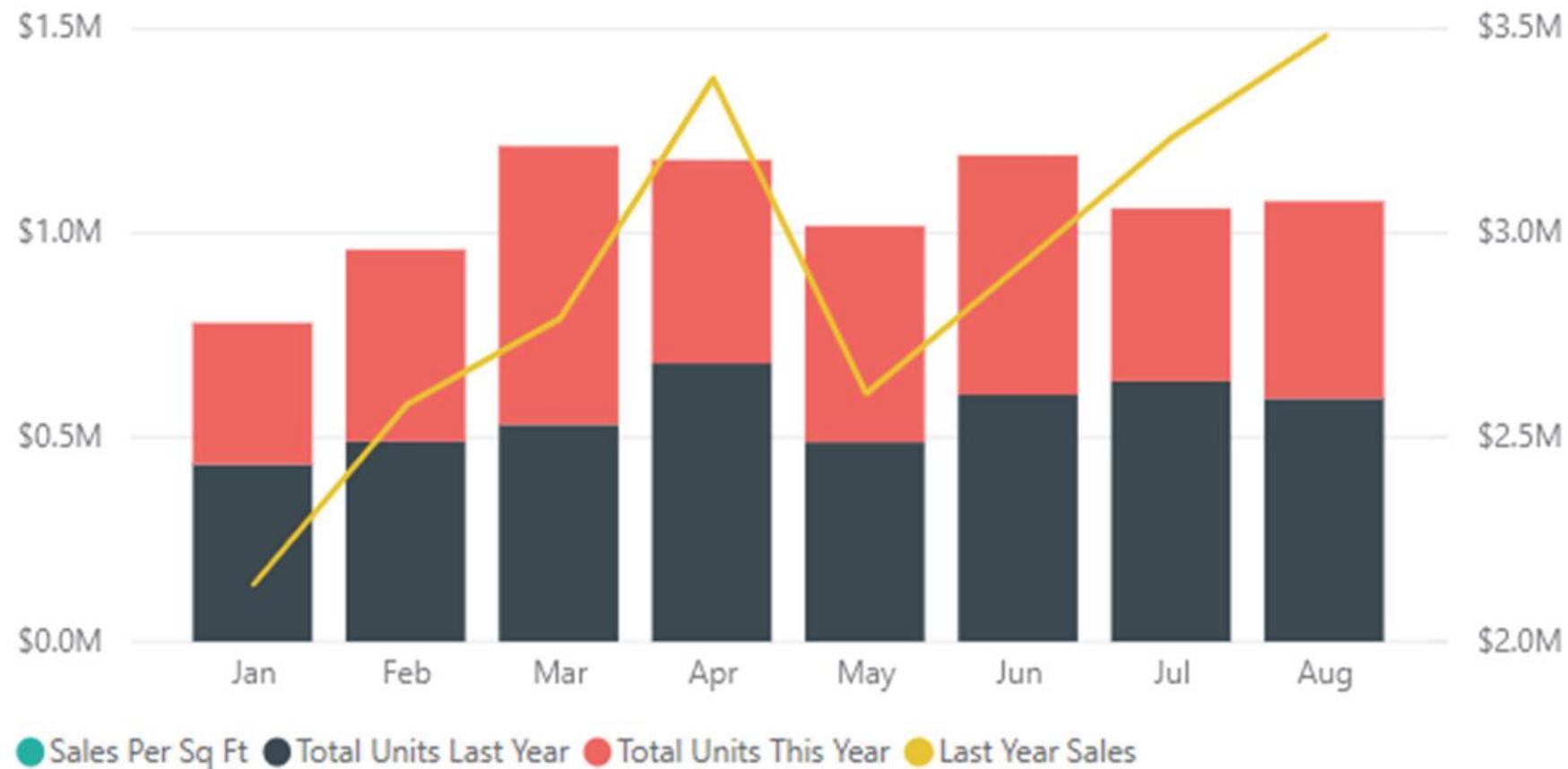
Source: Lachev & Price (2018)



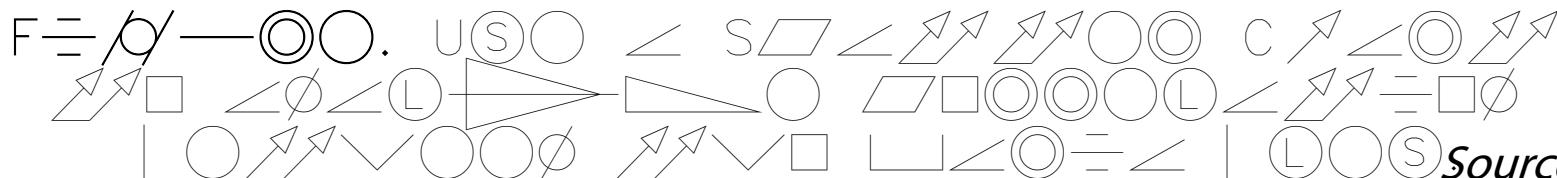
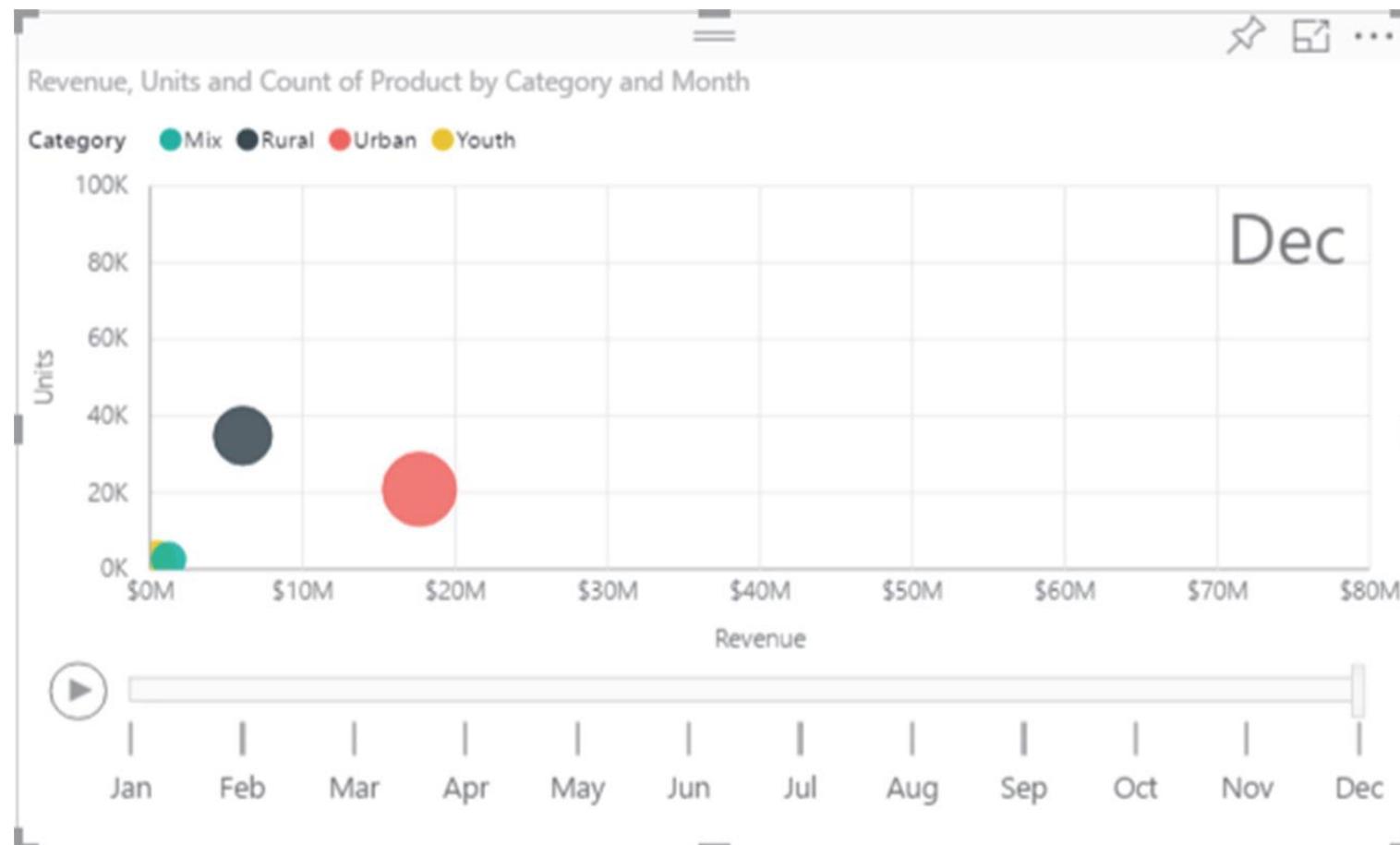
Source: Lachev & Price (2018)



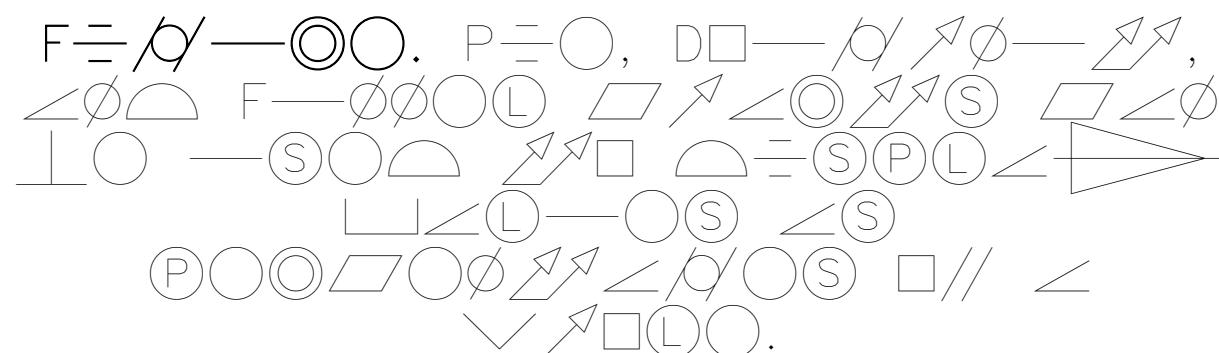
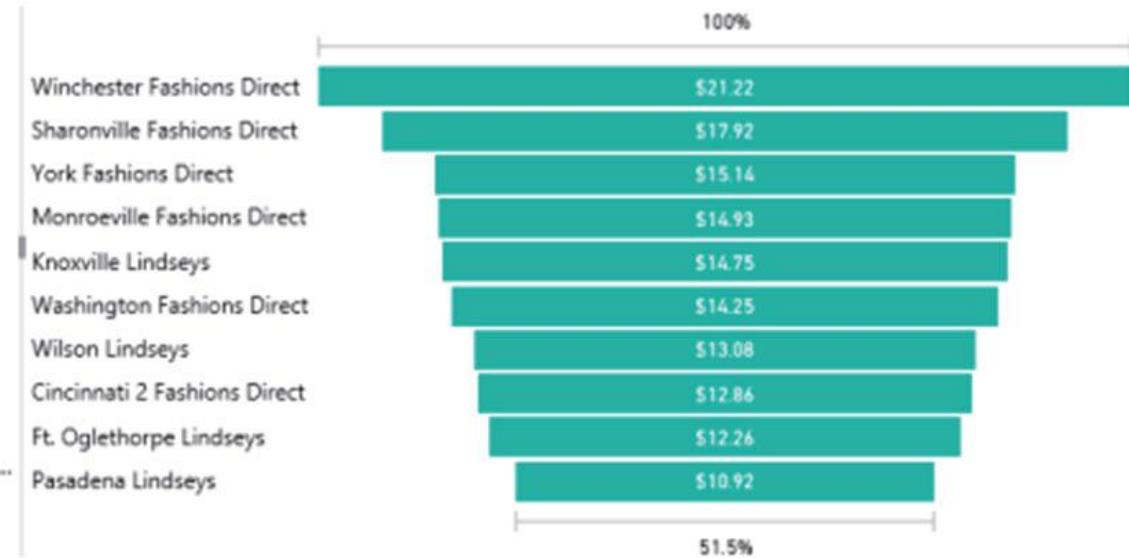
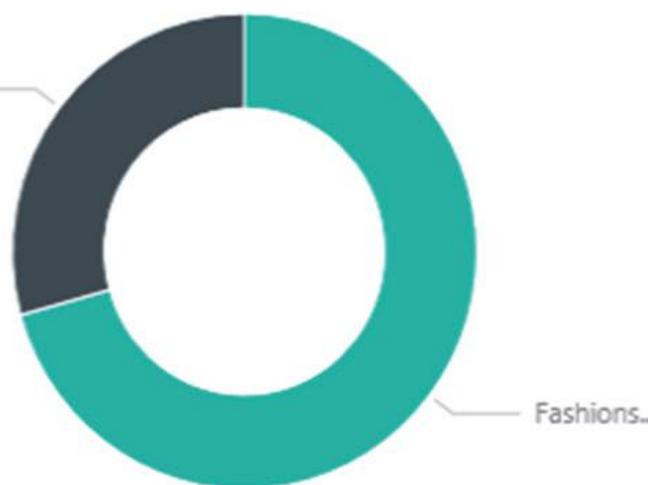
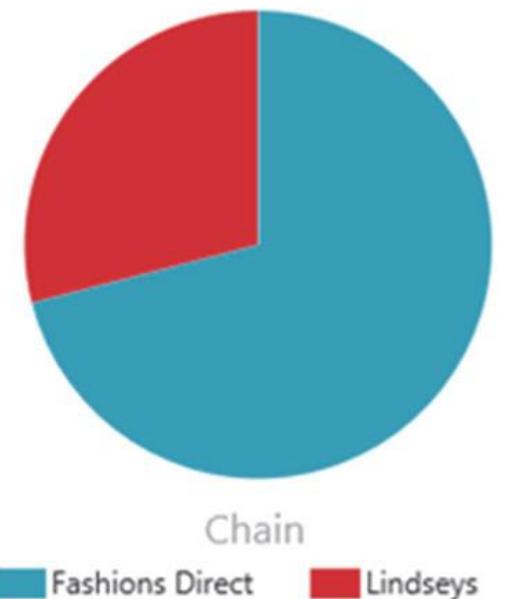
Sales Per Sq Ft, Total Units Last Year, Total Units This Year and Last Year Sales by Fiscal Month



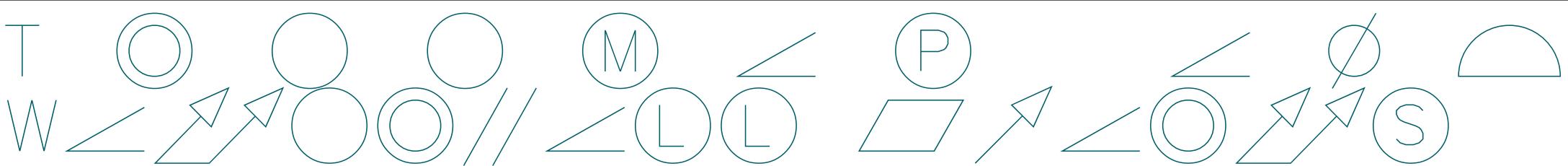
Source: Lachey & Price (2018)



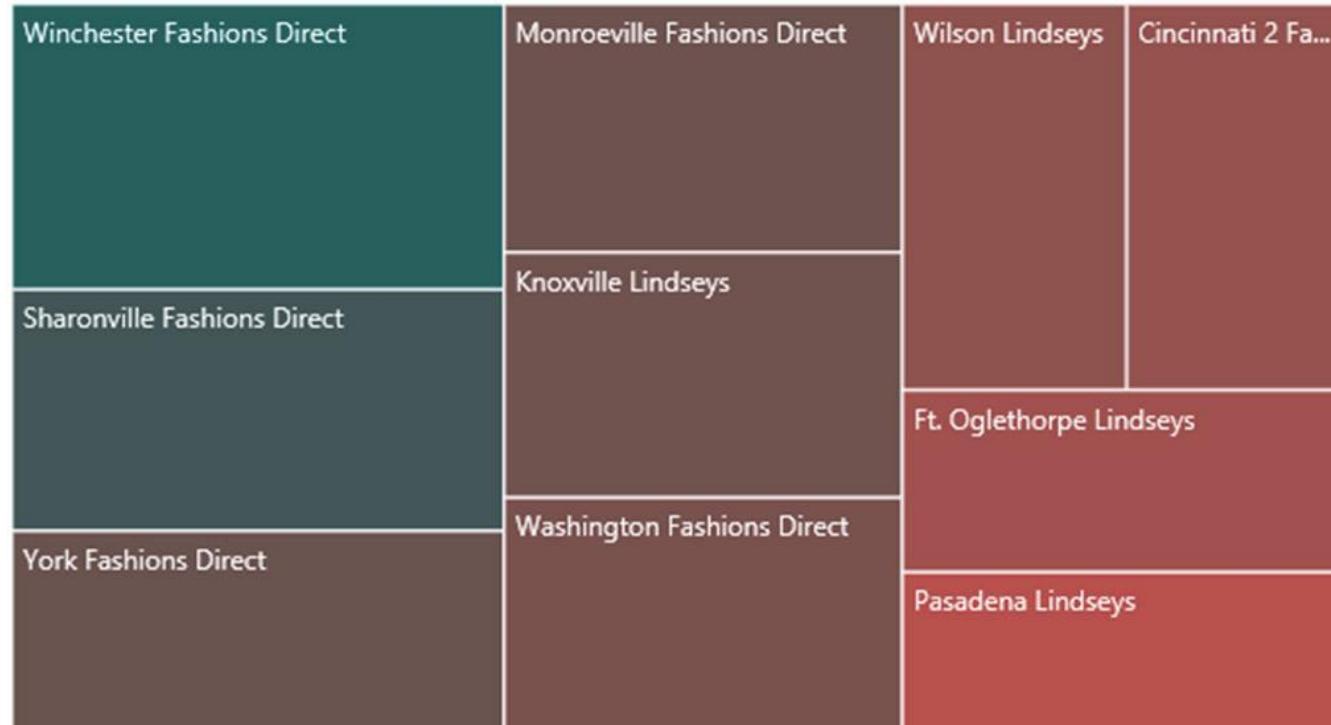
Source: Lachev & Price (2018)



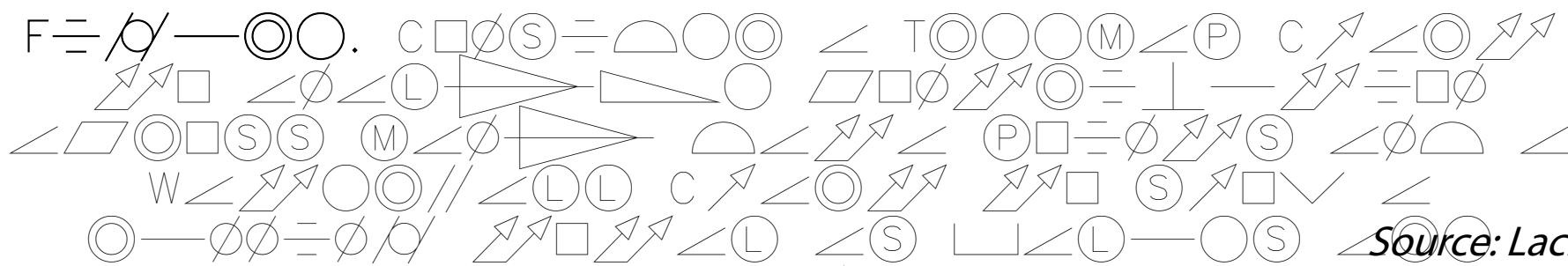
Source: Lachev & Price (2018)



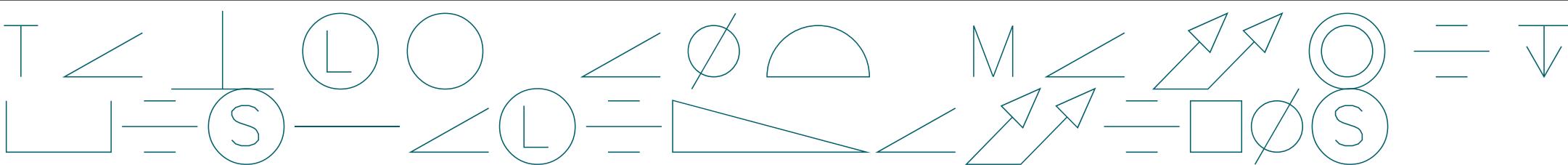
Sales Per Sq Ft and Sales Per Sq Ft by Name



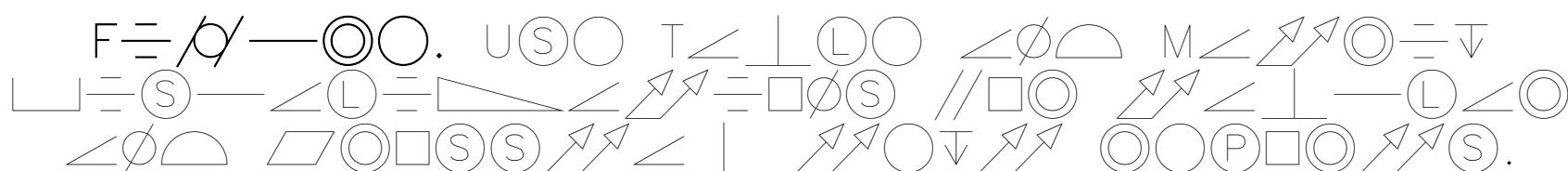
Total Sales Variance % by FiscalMonth



Source: Lachev & Price (2018)



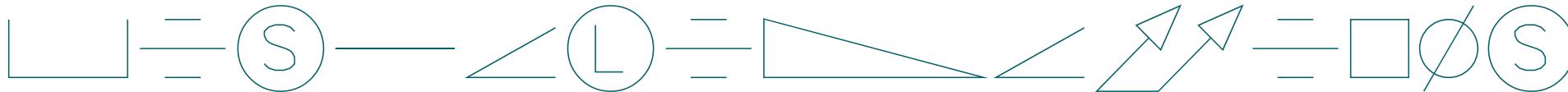
Segment	Product	Revenue	%GT Revenue	Year		2012		2013		2014	
				Segment	Revenue	%GT Revenue	Revenue	%GT Revenue	Revenue	%GT Revenue	
Moderation	Maximus UM-80	\$77,278,399	3.19%	+ Convenience	\$67,673,681	2.79%	\$80,468,276	3.32%	\$92,770,291	3.83%	
Moderation	Maximus UM-01	\$68,736,568	2.84%	+ Moderation	\$39,910,930	1.65%	\$48,928,754	2.02%	\$62,318,022	2.57%	
Moderation	Natura UM-10	\$47,733,220	1.97%	+ Extreme	\$30,445,067	1.26%	\$39,785,588	1.64%	\$49,741,872	2.05%	
Convenience	Maximus UC-00	\$41,304,725	1.70%	+ Productivity	\$22,056,108	0.91%	\$32,157,683	1.33%	\$42,761,440	1.76%	
Convenience	Maximus UC-69	\$40,796,332	1.68%	+ Select	\$7,412,775	0.31%	\$9,274,441	0.38%	\$10,006,674	0.41%	
Convenience	Maximus UC-74	\$39,648,312	1.64%	+ All Season	\$1,946,617	0.08%	\$2,731,482	0.11%	\$2,864,330	0.12%	
Convenience	Maximus UC-41	\$30,240,252	1.25%	+ Youth	\$1,270,201	0.05%	\$1,649,357	0.07%	\$2,208,297	0.09%	
Moderation	Maximus UM-50	\$28,764,264	1.19%	Total	\$173,345,354	7.15%	\$218,914,882	9.03%	\$266,956,375	11.02%	
Convenience	Maximus UC-16	\$27,920,144	1.15%								
Total		\$2,423,293,798	100.00%								



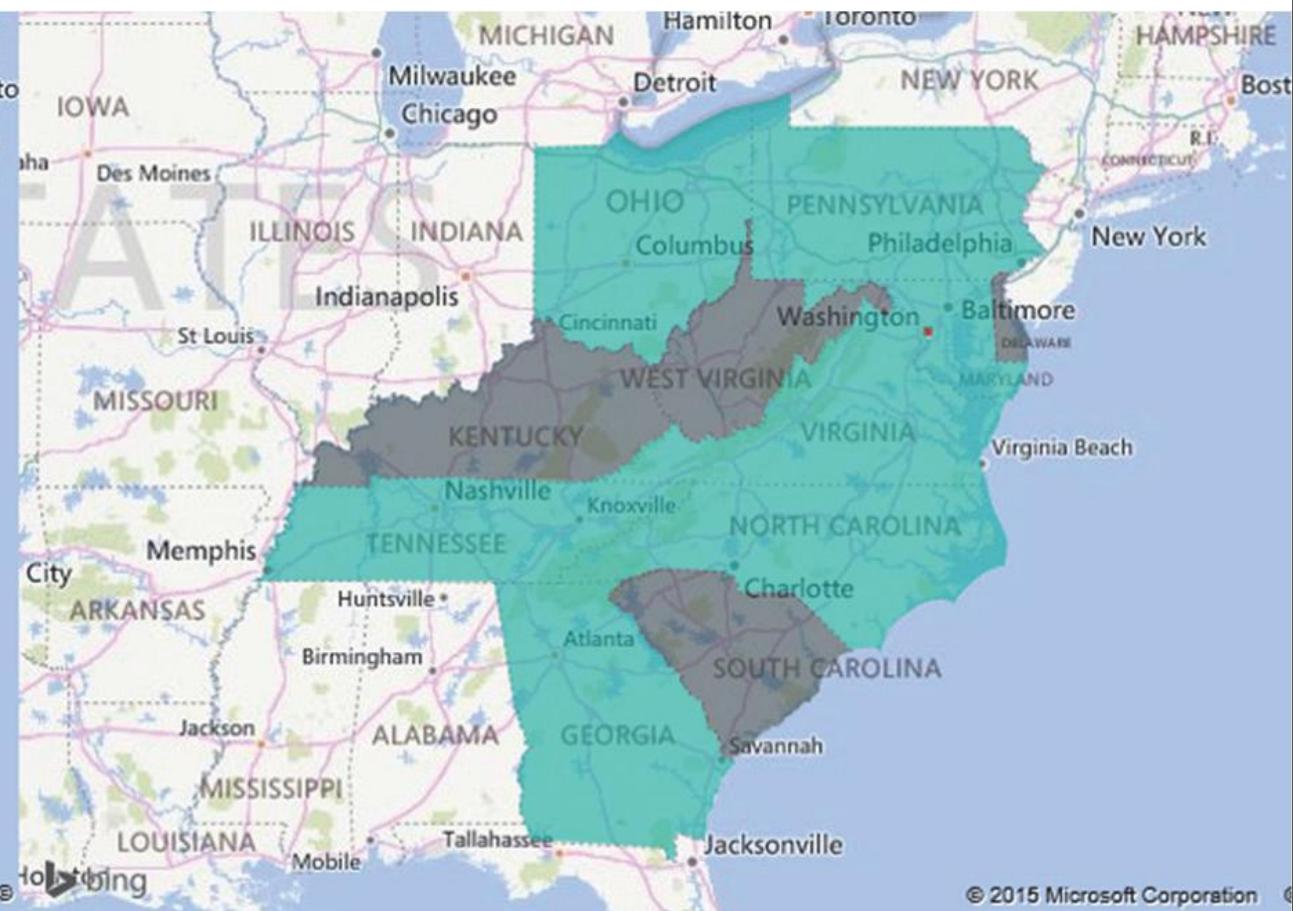
Source: Lachev & Price (2018)

M

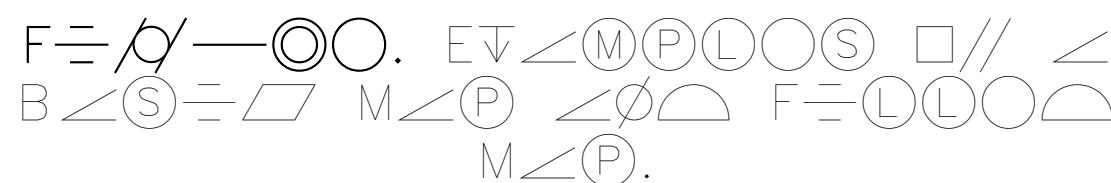
P



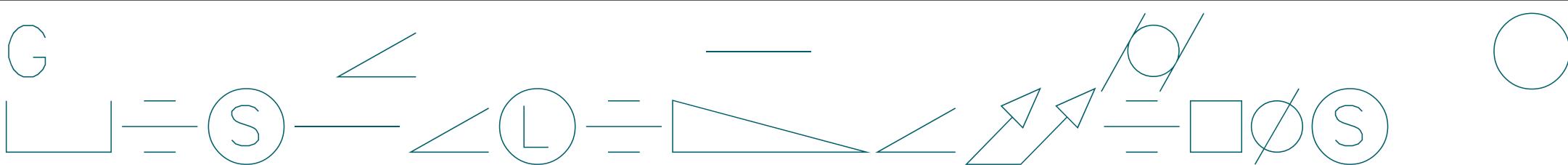
© 2015 Microsoft Corporation



© 2015 Microsoft Corporation



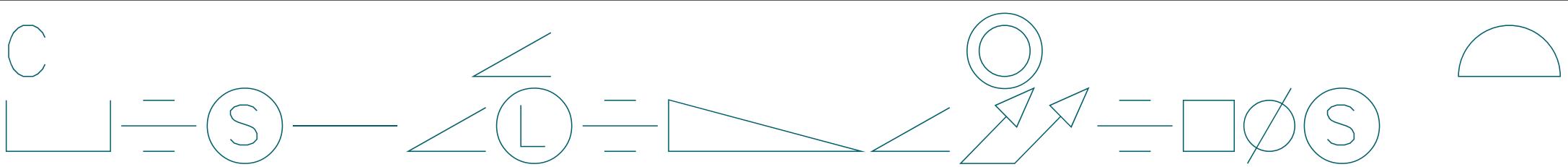
Source: Lachev & Price (2018)



Calendar Year	Sales Amount	Sales Amount Quota
2005	\$8,065,435	\$9,513,000
2006	\$24,144,430	\$29,009,000
2007	\$32,202,669	\$38,782,000
2008	\$16,038,063	\$18,410,000
Total	\$80,450,597	\$95,714,000



Source: Lachev & Price (2018)



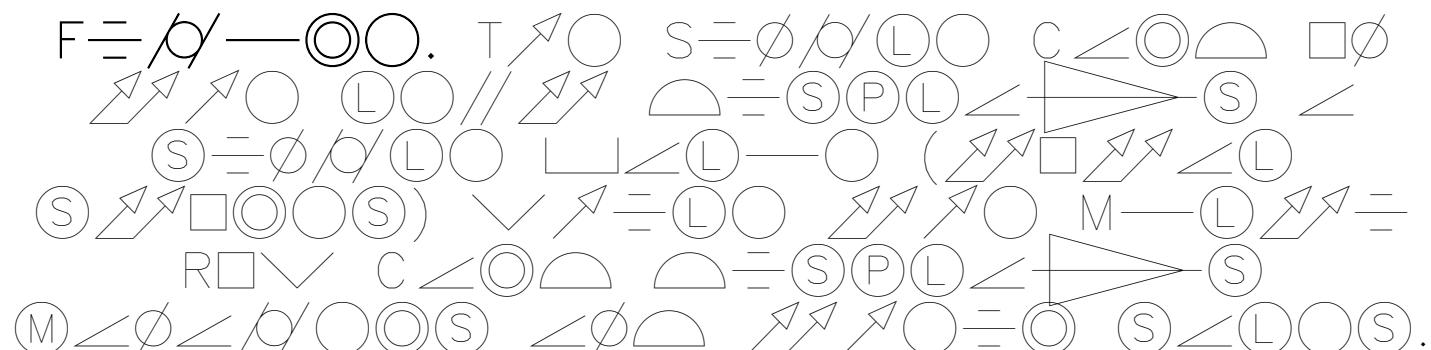
Total Stores
NEW & EXISTING STORES

104

Valery Ushakov \$3,841,281
District Manager This Year Sales

Tina Lassila \$3,717,414
District Manager This Year Sales

Carlos Grilo \$3,804,313
District Manager This Year Sales



Source: Lachev & Price (2018)

The screenshot displays a Power BI interface with several interactive elements:

- Date Filter:** A slider with two circular endpoints, currently set from 1/1/2005 to 12/31/2010.
- Date Selection:** A dropdown menu labeled "Date" with options "Last" (dropdown arrow), "1" (text input), and "Months" (dropdown arrow). Below it, a date range is shown as "10/27/2020 - 11/26/2020".
- Country Filter:** A dropdown menu titled "Country" with a search bar. It includes a magnifying glass icon and a "Search" button. The list of countries is:
 - Australia
 - Canada
 - France
 - Germany
 - United KingdomThe "List" button is highlighted in blue, while "Dropdown" is shown below it.
- Country Cards:** At the bottom, there are four dark grey rectangular cards with white text, representing selected countries: Australia, Canada, France, and Germany. To the right of these cards is a large, light grey rectangular area with a right-pointing arrow icon.

A decorative border composed of various geometric shapes such as circles, squares, and arrows, arranged in a repeating pattern along the bottom edge of the page.

Source: Lachev & Price (2018)



Key influencers Top segments

What influences Gross Margin This Year to Increase ?

When...

....the average of Gross Margin This Year increases by

Category is 020-Mens

\$12.17K

Territory is WV

\$9.73K

Territory is PA

\$8.38K

Territory is OH

\$7.76K



Visualizations >



Analyze

Gross Margin This Year

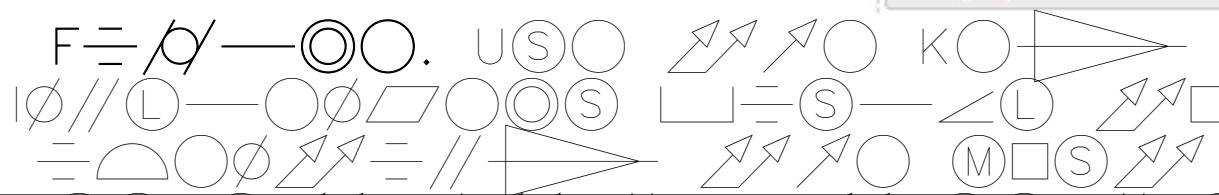
Explain by

City

Store Type

Territory

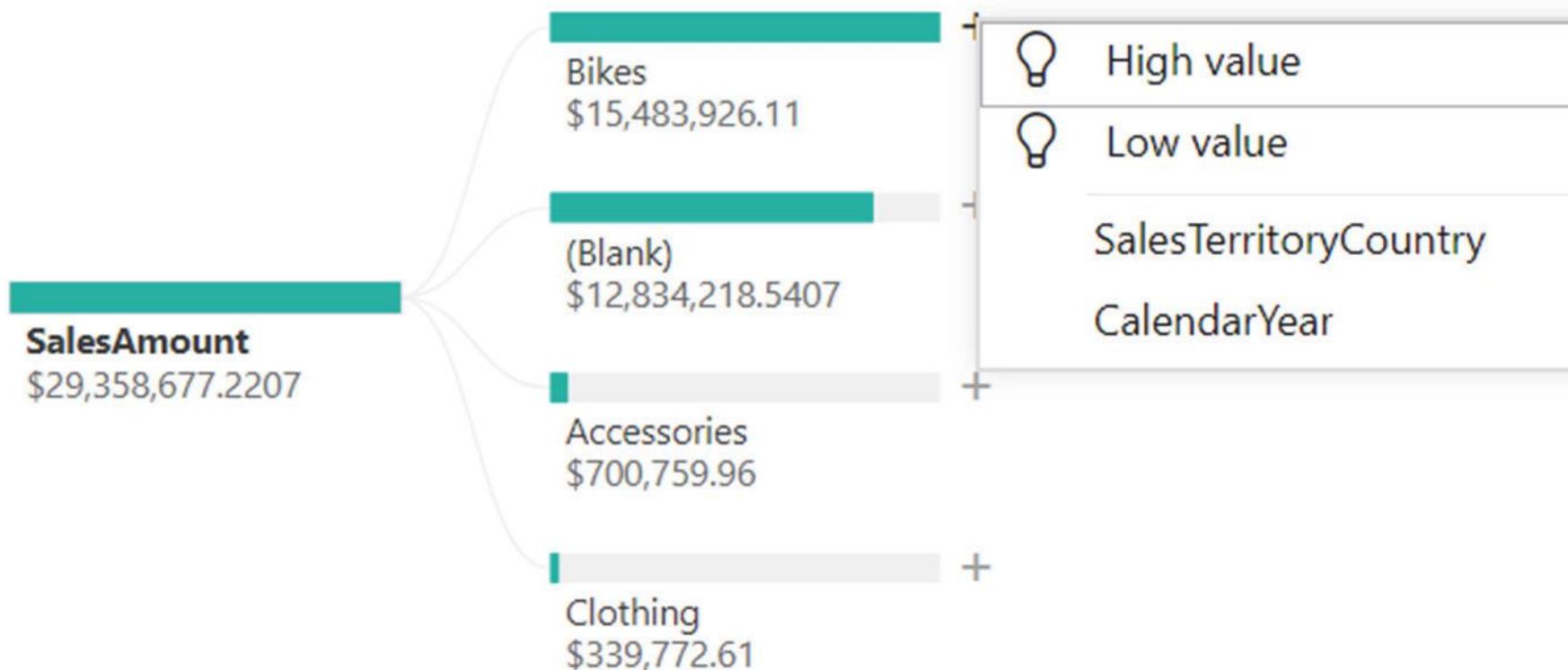
Category



Source: Lachev & Price (2018)

D O □ △ □ M P □ S = △ △ □ ○ T ○ ○

ProductCategory ×

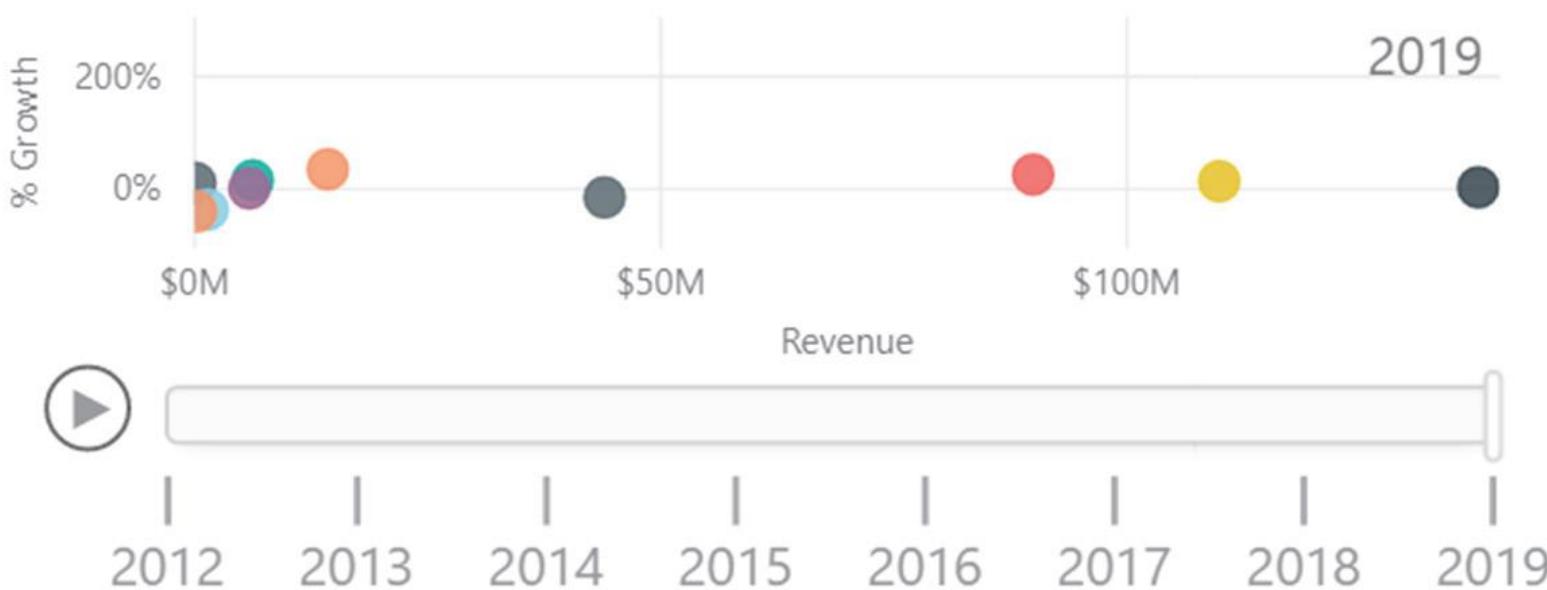


F = ○ — ○○. U○○ D○△□ M P □ S = △ △ □ ○ T ○ ○ ○ Source: Lachev & Price (2018)



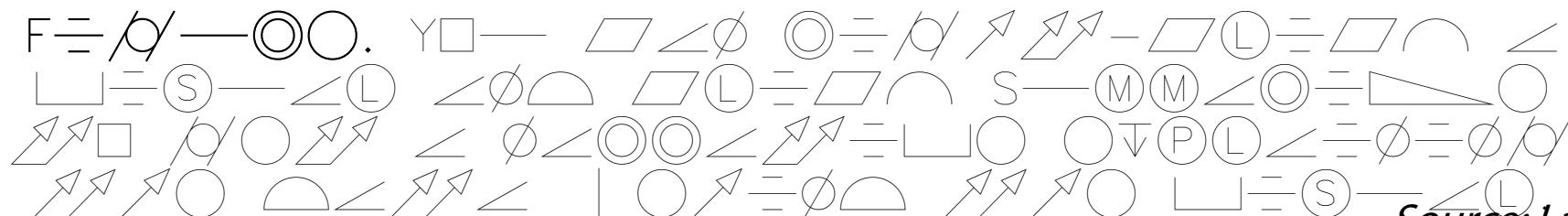
Revenue and % Growth by Category, Segment and Year

Segment ● All Season ● Convenience ● Extreme ● Moderation ● Productivity ● Regular



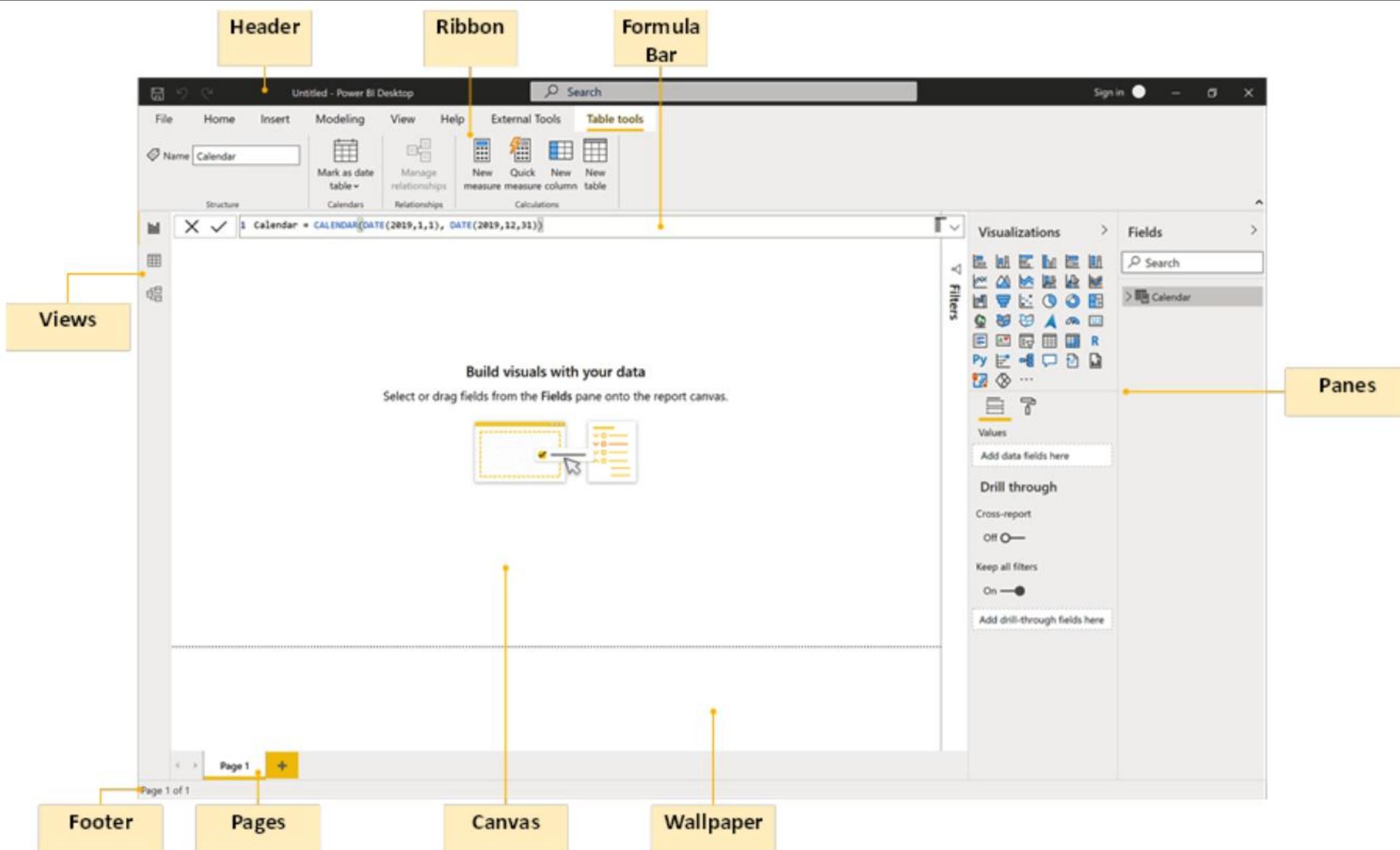
► 2019 had the highest Revenue (\$339,158,005) and 2015 had the highest % Growth (44.73%). The Category with the highest Revenue was Urban and the Category with the highest % Growth was Youth.

Most Year were below \$80,355,302 in Revenue and below 17.24% in % Growth.





Power BI Desktop



Power BI Desktop interface elements

Source: Deckler (2022)



- There are three different views available:
 - **Report:** The **Report** view allows for the authoring of reports through the creation of visualizations on one or more pages.
 - **Data:** The **Data** view provides an interface for exploring the data contained within the individual tables of a data model.
 - **Model:** The **Model** view provides an overall look at all of the tables in the data model and how those tables relate to one another.

P ϕ OS

- **Filters:** The **Filters** pane is available within the **Report** view and displays a list of filters currently active on reports, pages, and visualizations.
- **Visualizations:** The **Visualizations** pane is also available when in the **Report** view and provides access to the various visualization types available within the report. Sub-panes exist for configuring and formatting visualizations and adding analytics to visualizations, as well as for configuring filters and drill-through capabilities for reports, pages, and visualizations.
- **Bookmarks, Selection, Performance Analyzer, and Sync Slicers:** These panes are only available while in **Report** view and provide additional capabilities that we will explore later.
- **Properties:** The **Properties** pane is only present in the **Model** view. This pane provides the ability to associate metadata (data about data) for various fields or columns within the tables of the data model. This includes the ability to specify synonyms and descriptions as well as data types, data categories, and default aggregations or summarizations.



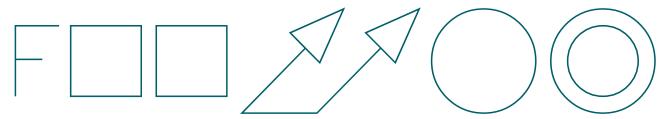
- The **Canvas** is the **main work area** within the desktop. This area is **contextual** depending upon the view. When in the **Report** view, this area is the place where visualizations are created in order to build reports. When in the **Data** view, this area displays the underlying data for a selected table within the data model. Finally, when in the **Model** view, this area displays all of the tables within the data model and their relationships to one another.



- The **Wallpaper** area is only available in the **Report** view. There is little functionality other than the ability to set a background color or image for a report.

PAGES

- The **Pages** area is only available when in the **Report** view. The **Pages** area provides the ability to create new pages, rename pages, and reorder pages within a report.



- The **Footer** area is **contextual** based upon the current view. In the **Report** view, the footer provides basic information regarding how many pages are in the report and which page is currently selected by the user. In the **Data** view, the footer provides basic statistics of a selected table and/or column, including the number of rows in the table and the number of distinct values in a column. Finally, in the **Model** view, the footer provides various viewing controls, such as the ability to zoom in and out, reset the view, and fit the model to the current display area.



- The ribbon consists of three permanent tabs and a number of contextual tabs. The permanent tabs are as follows:
 - **File**: The **File** tab actually displays a fly-out menu when clicked that allows overall file operations, such as opening and saving Power BI Desktop files. Power BI Desktop files have the .pbix file extension. Other operations include importing data, exporting, and publishing.
 - **Home**: The **Home** tab provides a variety of the most common operations, such as copying and pasting, getting data, inserting visuals, creating calculations, and other common actions.
 - **Help**: The **Help** tab includes useful links for getting help with Power BI, including links to the Power BI Community site, documentation, guided learning, and training videos.



- Contextual tabs appear in the ribbon depending upon the view selected, what items are selected in the interface, and whether or not additional tools are installed. These tabs include the following:
 - **Insert:** The **Insert** tab only appears in the **Report** view and has options for adding pages, visuals, and visual elements such as textboxes, images, and buttons.
 - **Modeling:** The **Modeling** tab only appears in the **Report** view and provides operations common to the data modeling process, including creating calculations, new tables, and new parameters, as well as operations related to security, questions, and answers.
 - **View:** The **View** tab only appears in the **Report** view and has actions related to themes, page views, and mobile page layout, as well as options related to laying out visual elements on a page, such as gridlines and snap to grid. Finally, the **View** tab includes options for showing or hiding panes, such as the **Filters** pane, **Bookmarks** pane, and **Performance Analyzer** pane.
 - **Format:** The **Format** tab is all about formatting how visuals interact with one another or are displayed in relation to one another. This tab appears while in **Report** view when a visual is selected on the canvas.
 - **Drill/Data:** The **Drill/Data** tab provides operations focused on Power BI's ability to drill into data and see the raw data that makes up a visualization. This tab only appears in **Report** view when a visual is selected on the canvas.
 - **Table tools:** The **Table tools** tab provides options for adjusting the properties of tables as well as creating new tables, measures, and columns. This tab is displayed in the **Report** and **Data** views when a table is selected in the **Fields** pane.
 - **Column tools:** The **Column tools** tab provides options for adjusting the properties of columns in the dataset. This tab is displayed in the **Report** and **Data** views when a column is selected in the **Fields** pane.
 - **Measure tools:** The **Measure tools** tab provides options for adjusting the properties of DAX measures. This tab is displayed in the **Report** and **Data** views when a measure is selected in the **Fields** pane.
 - **External Tools:** This tab is displayed in the **Report**, **Data**, and **Model** views when external tools, such as **DAX Studio** or **Tabular Editor**, are installed.



- The formula bar allows the user to enter **Data Analysis Expressions (DAX)** code in order to create columns, measures, and tables in the data model. DAX is a formula language comprised of functions, operators, and values and is used in Analysis Services (Tabular), Power BI Desktop, and Power Pivot in Excel.

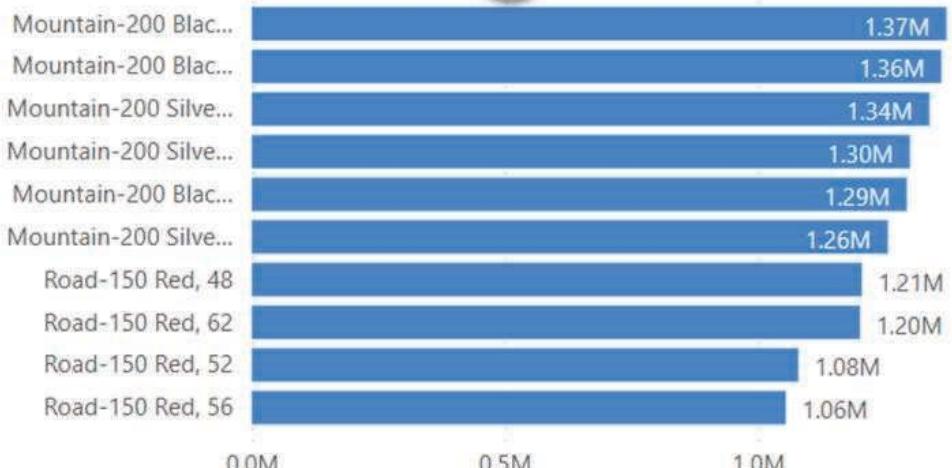


- While Excel's formula language is optimized for dealing with cells in a spreadsheet, **DAX (Data Analysis Expressions) is optimized to deal with tables of data consisting of columns and rows**. Hence, unlike Excel, it is not possible to reference an individual cell within a table. Instead, you use DAX to identify a table and a column and then filter down to a single row or rows.

Internet Sales Analysis

Sales by Product

1



2

29.36M

SalesAmount

3

60K

OrderQuantity

Sales and Order Quantity by Date

● SalesAmount ● OrderQuantity

2.0M

1.5M

1.0M

0.5M

0.0M

4

2006

2007

2008

6K

4K

2K

0K

Product	2005	2006	2007	2008	Total
Mountain-200 Black, 46		163,927.86	679,399.00	530,142.69	1,373,469.55
Mountain-200 Black, 42		159,829.66	627,269.94	576,042.49	1,363,142.09
Mountain-200 Silver, 38		136,713.69	613,471.64	589,277.46	1,339,462.79
Mountain-200 Silver, 46		103,570.98	617,531.62	579,997.50	1,301,100.10
Mountain-200 Black, 38		106,553.11	600,795.59	587,517.44	1,294,866.14
Mountain-200 Silver, 42		136,713.69	584,803.19	535,917.69	1,257,434.57
Road-150 Red, 48	547,475.31	658,401.68			1,205,876.99
Road-150 Red, 62	593,992.82	608,305.90			1,202,298.72
Road-150 Red, 52	472,331.64	608,305.90			1,080,637.54
Road-150 Red, 56	486,644.72	568,944.93			1,055,589.65
Total	3,266,373.66	6,530,343.53	9,791,060.30	9,770,899.74	29,358,677.22

5

6

Sales by Year

10M

5M

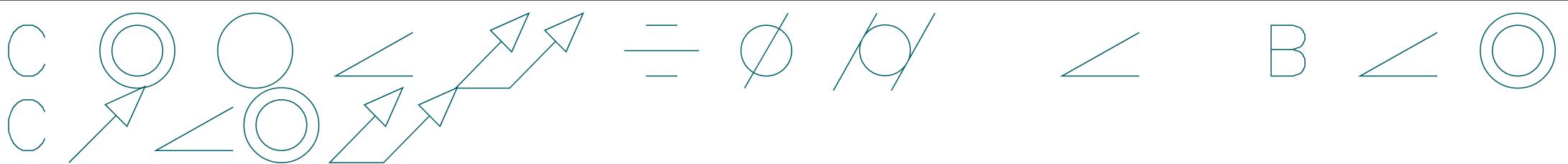
0M

2005

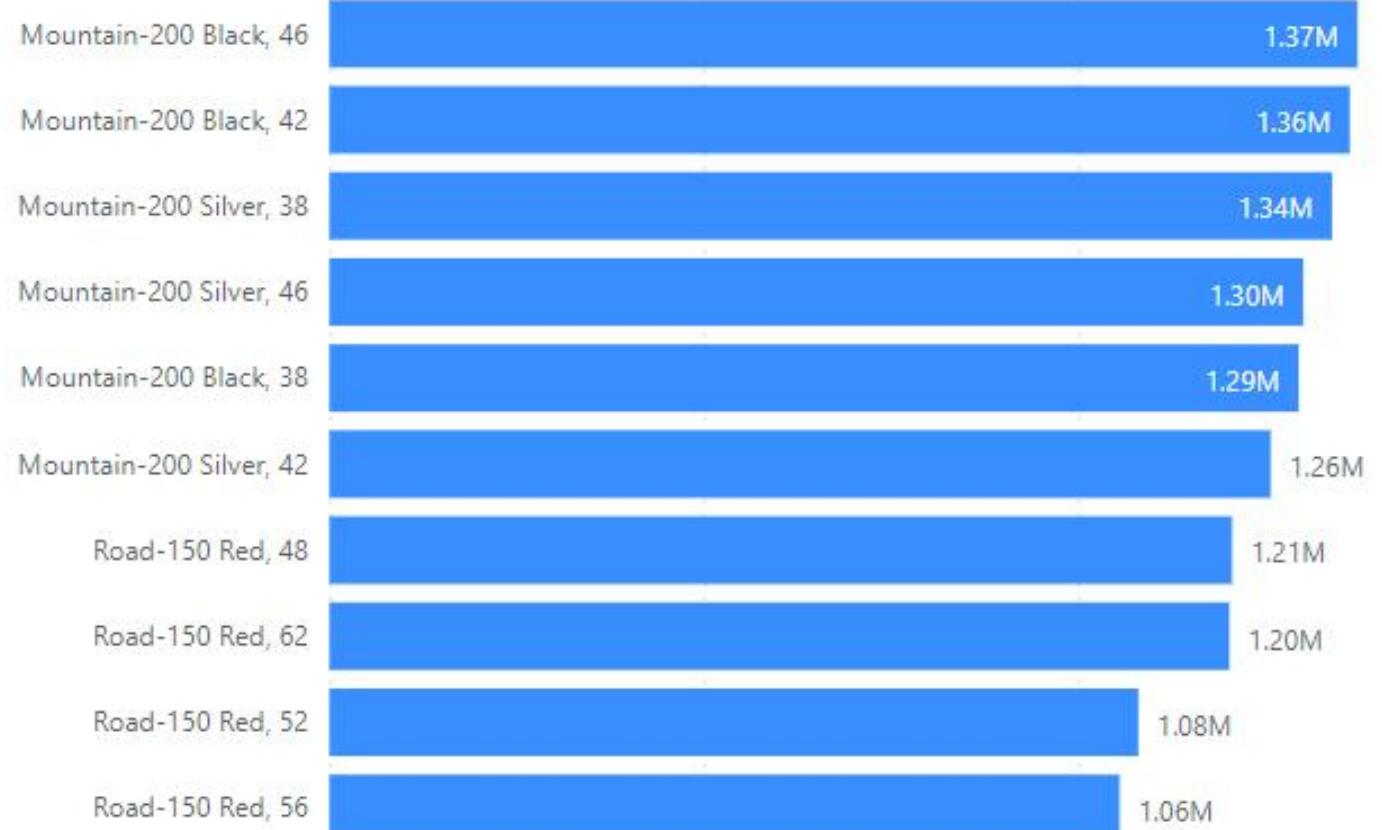
2006

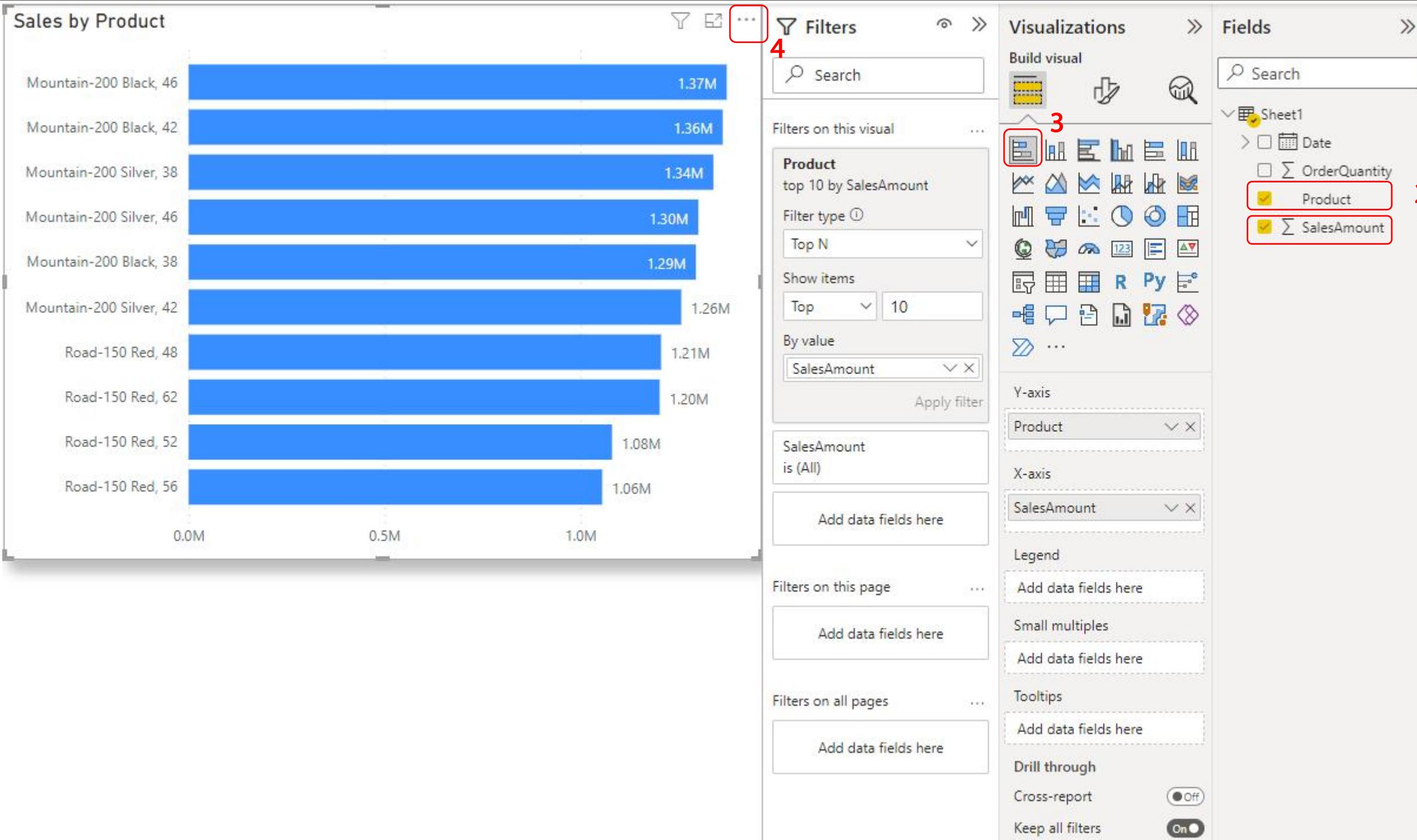
2007

2008



Sales by Product





Visualizations

Format visual

Search

Visual General ...

> Properties

Title **5**

Text **Sales by Product**

Heading **Heading 3**

Font **DIN** **14**

B **I** **U**

Text color

Background color

Horizontal alignment

Text wrap

Visualizations

Format visual

Search

Visual General ...

> Y-axis **On**

> X-axis **On**

> Legend **Off**

> Small multiples

> Gridlines

> Zoom slider **Off**

Bars **6**

> Data labels **On**

> Total labels

> Plot area background

Font **Segoe UI** **9**

B **I** **U**

Color

Max area width **25 %**

Switch axis position **Off** **7**

Values

Range

Minimum **Auto**

Maximum **Auto**

Logarithmic scale **Off**

Invert range **Off**

Title **Off** **8**

Reset to default

Visualizations

Format visual

Search

Visual General ...

> Y-axis **On**

> X-axis **On**

> Values

Font **Segoe UI** **9**

B **I** **U**

Color

Range

Minimum **Auto**

Maximum **Auto**

Logarithmic scale **Off**

Invert range **Off**

Reset to default

Filters

Search

Filters on this visual

Product top 10 by SalesAmount **9**

Filter type **10**

Show items **Top** **10** **11**

By value **SalesAmount**

Apply filter

SalesAmount is (All)

Add data fields here

Filters on this page

Add data fields here

Visualizations

Format visual

Search

Visual General ...

> Properties

> Title **On**

> Effects

Background **On**

Color

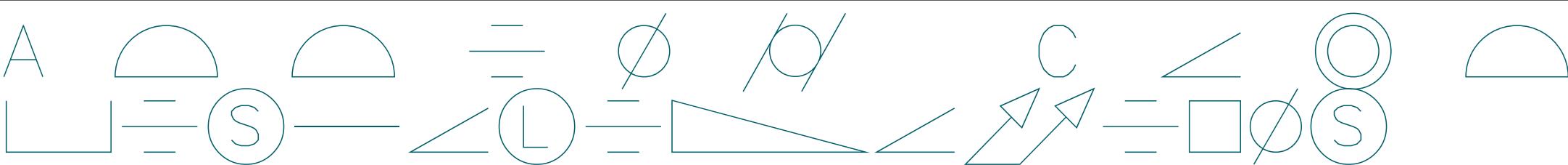
Transparency **0 %** **12**

Visual border **On** **13**

Color

Rounded corners **5 px** **14**

Shadow



29.36M
SalesAmount

60K
OrderQuantity

Filters

Search

Filters on this visual

SalesAmount
is (All)

Add data fields here

Filters on this page

Add data fields here

Visualizations

Build visual

Fields

Search

Sheet1

Date

\sum OrderQuantity

Product 1

\sum SalesAmount

The screenshot shows a Power BI interface with two main visualizations: a large number '29.36M' for SalesAmount and another for OrderQuantity. On the right, the 'Filters' pane shows a search bar and a dropdown for SalesAmount set to '(All)'. The 'Visualizations' pane displays a grid of icons for different chart types, with a red box highlighting the 'Table' icon. The 'Fields' pane lists fields from 'Sheet1': Date, \sum OrderQuantity, Product (with a value of 1), and \sum SalesAmount, with the latter also highlighted by a red box.

A decorative header featuring various light blue icons, including letters Q, S, M, P, L, and various shapes like circles, squares, and arrows.

Sales amount

29,358,677.22

60K

OrderQuantity

Filters

Search

Filters on this visual

SalesAmount
is (All)

Add data fields here

Filters on this page

Add data fields here

Visualizations

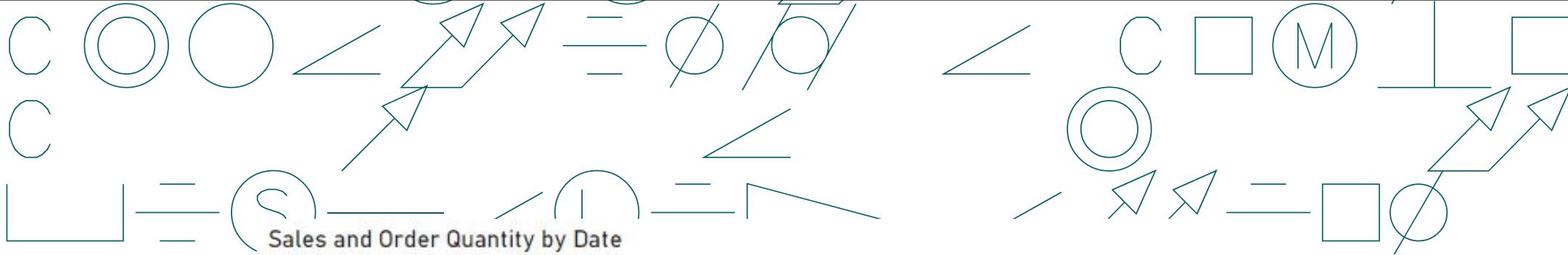
Build visual

Fields

Search

Sheet1

- Date
- \sum OrderQuantity
- Product
- \sum SalesAmount



Sales and Order Quantity by Date

● SalesAmount ● OrderQuantity

2,0M

1,5M

1,0M

0,5M

0,0M

6K

5K

4K

3K

2K

1K

0K

Jul 2005

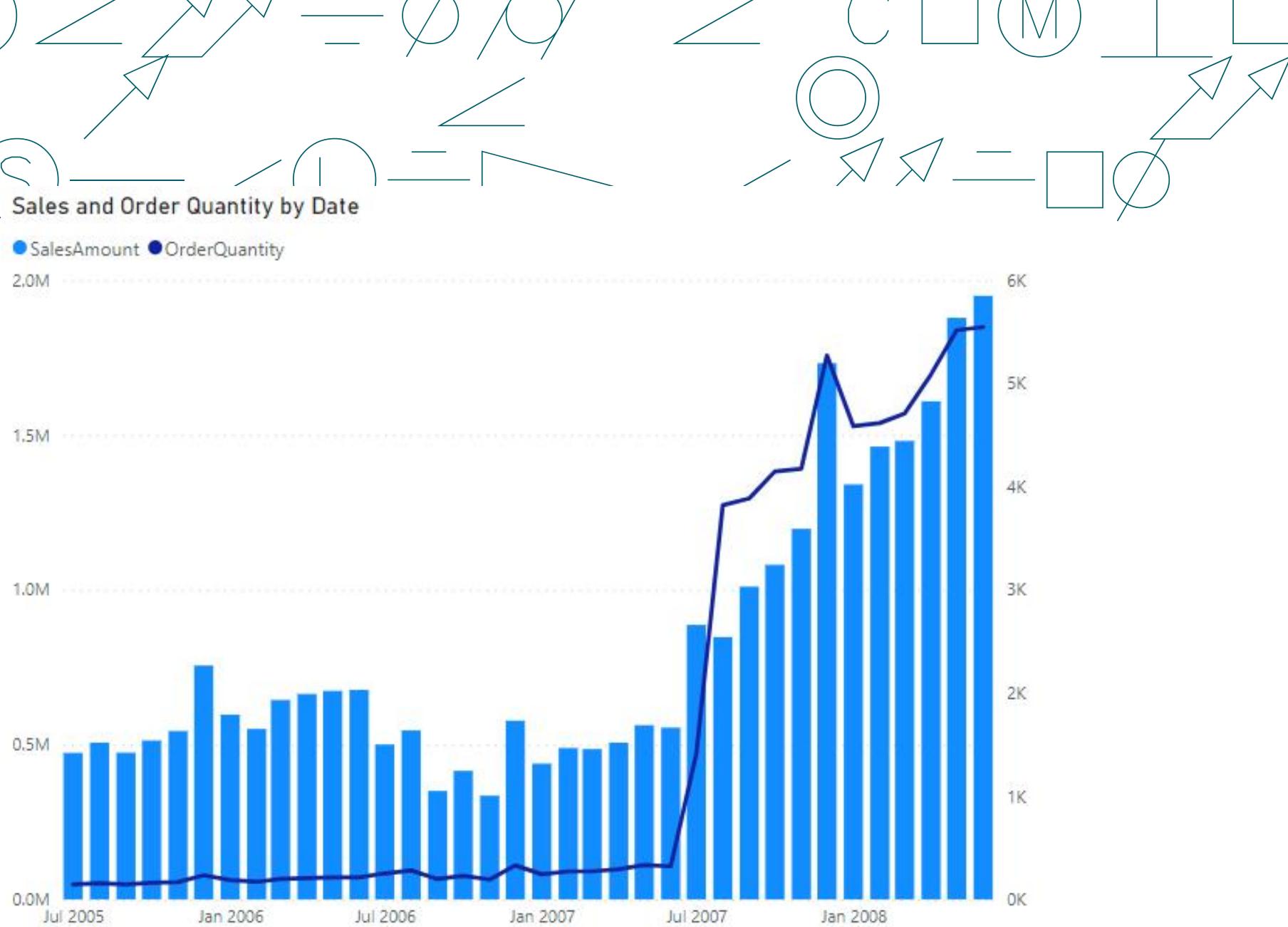
Jan 2006

Jul 2006

Jan 2007

Jul 2007

Jan 2008



Sales and Order Quantity by Date

● SalesAmount ● OrderQuantity

58,67...
60K
OrderQuantity

Product	2005	2006	2007	2008	Total
Mountain-200 Black, 46		163,927.86	679,399.00	530,142.69	1,373,469.55
Mountain-200 Black, 42		159,829.66	627,269.94	576,042.49	1,363,142.09
Mountain-200 Silver, 38		136,713.69	613,471.64	589,277.46	1,339,462.79
Mountain-200 Silver, 46		103,570.98	617,531.62	579,997.50	1,301,100.10
Mountain-200 Black, 38		106,553.11	600,795.59	587,517.44	1,294,866.14
Mountain-200 Silver, 42		136,713.69	584,803.19	535,917.69	1,257,434.57
Road-150 Red, 48	547,475.31	658,401.68			1,205,876.99
Road-150 Red, 62	593,992.82	608,305.90			1,202,298.72
Road-150 Red, 52	472,331.64	608,305.90			1,080,637.54
Road-150 Red, 56	486,644.72	568,944.93			1,055,589.65
Total	3,266,373.66	6,530,343.53	9,791,060.30	9,770,899.74	29,358,677.22

Filters

Search:

Filters on this visual

- Date - Day is (All)
- Date - Quarter is (All)
- Date - Month is (All)
- Date - Year is (All)
- Product is (All)
- SalesAmount is (All)
- Add data fields here

Filters on this page

Add data fields here

Visualizations

Build visual

Fields

Search:

Sheet1

Date

Date Hierarchy

Year (checked)
Quarter (unchecked)
Month (checked)

Day (unchecked)

OrderQuantity (unchecked)

Product (unchecked)

\sum SalesAmount (checked)

2

3

4

1

Red boxes highlight the SalesAmount field in the Fields pane (1), the Date field in the Date Hierarchy (4), and the columns Date, Year, and Month in the Columns section of the Filters pane (3).

Product	2005	2006	2007	2008	Total
All-Purpose Bike Stand					18,921.00
AWC Logo Cap					7,956.15
Bike Wash - Dissolver					3,044.85
Classic Vest, L					4,254.50
Classic Vest, M					4,572.00
Classic Vest, S					4,191.00
Fender Set - Mountain					19,408.34
Half-Finger Gloves, L					4,187.79
Half-Finger Gloves, M					4,775.55
Half-Finger Gloves, S					5,265.35
Hitch Rack - 4-Bike					16,440.00
HL Mountain Tire					20,300.00
HL Road Tire					12,877.00
Hydration Pack - 70 oz.					16,771.95
LL Mountain Tire					9,071.37
Total	3,266,373.66	6,530,343.53	9,791,060.30	9,770,899.74	29,358,677.22

Drill down

- Show as a table
- Show next level
- Expand to next level
- Include
- Exclude
- Group
- Summarize
- Copy

Visualizations

Format visual



Search

Visual General ...

Style presets

Minimal

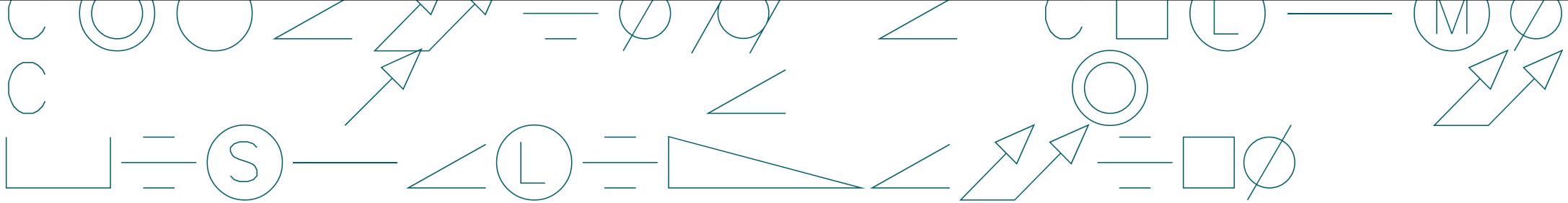
Reset to default

Grid

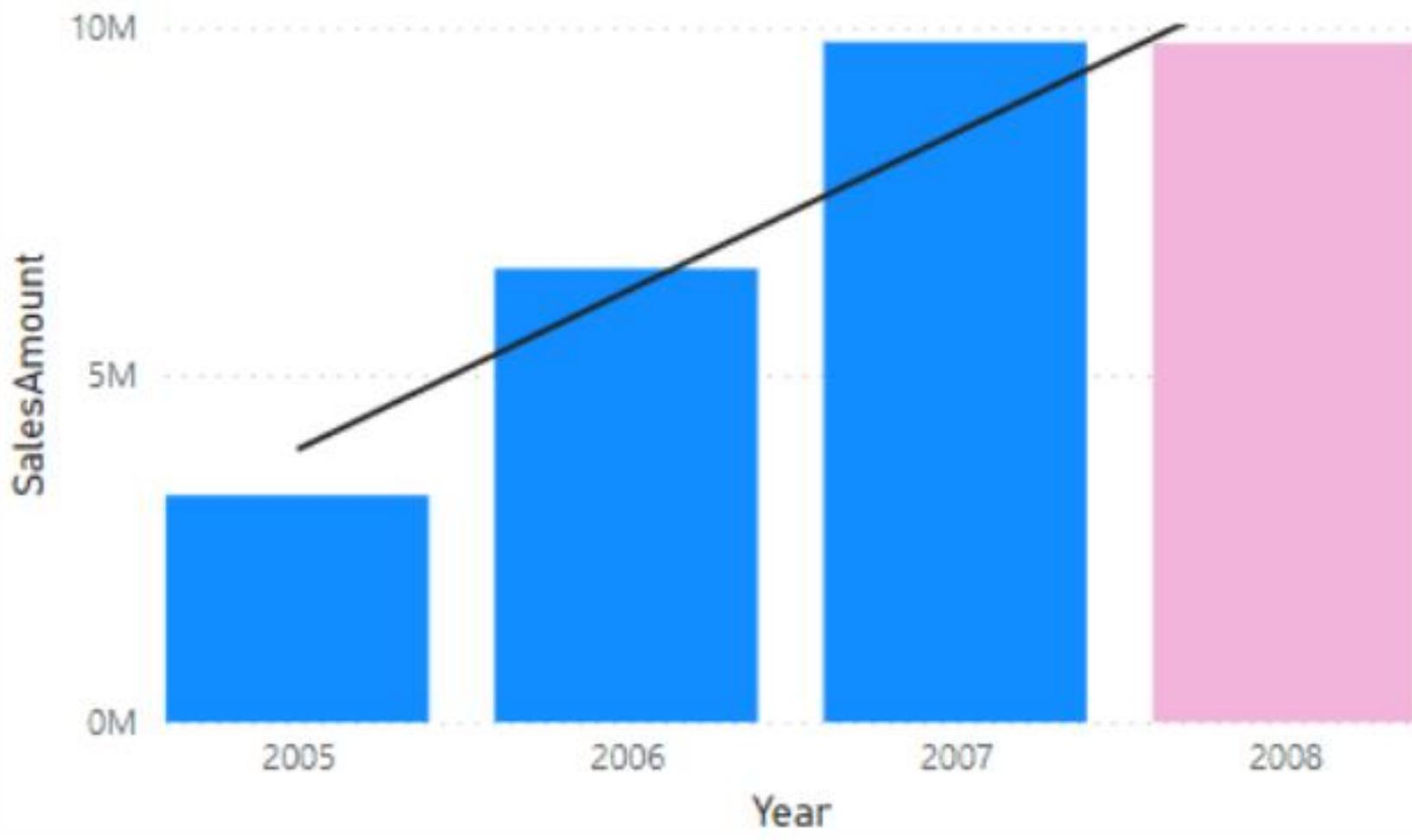
Values

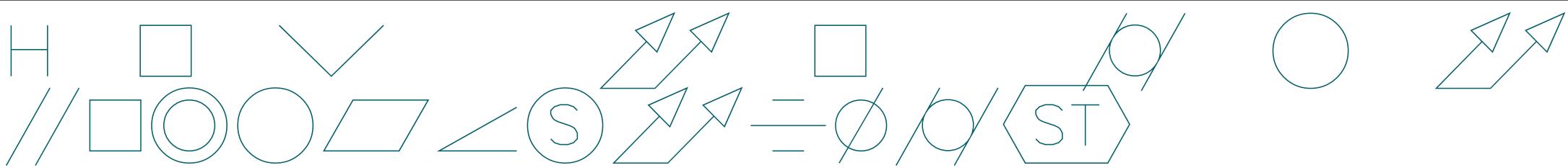


- Remove field
- Rename for this visual
- Move to >
- Add a sparkline
- Conditional formatting >
- Remove conditional formatting
- ✓ Sum
- Average
- Minimum
- Maximum
- Count (Distinct)
- Count
- Standard deviation
- Variance
- Median
- Show value as >
- New quick measure



Sales by Year





Sales by Year

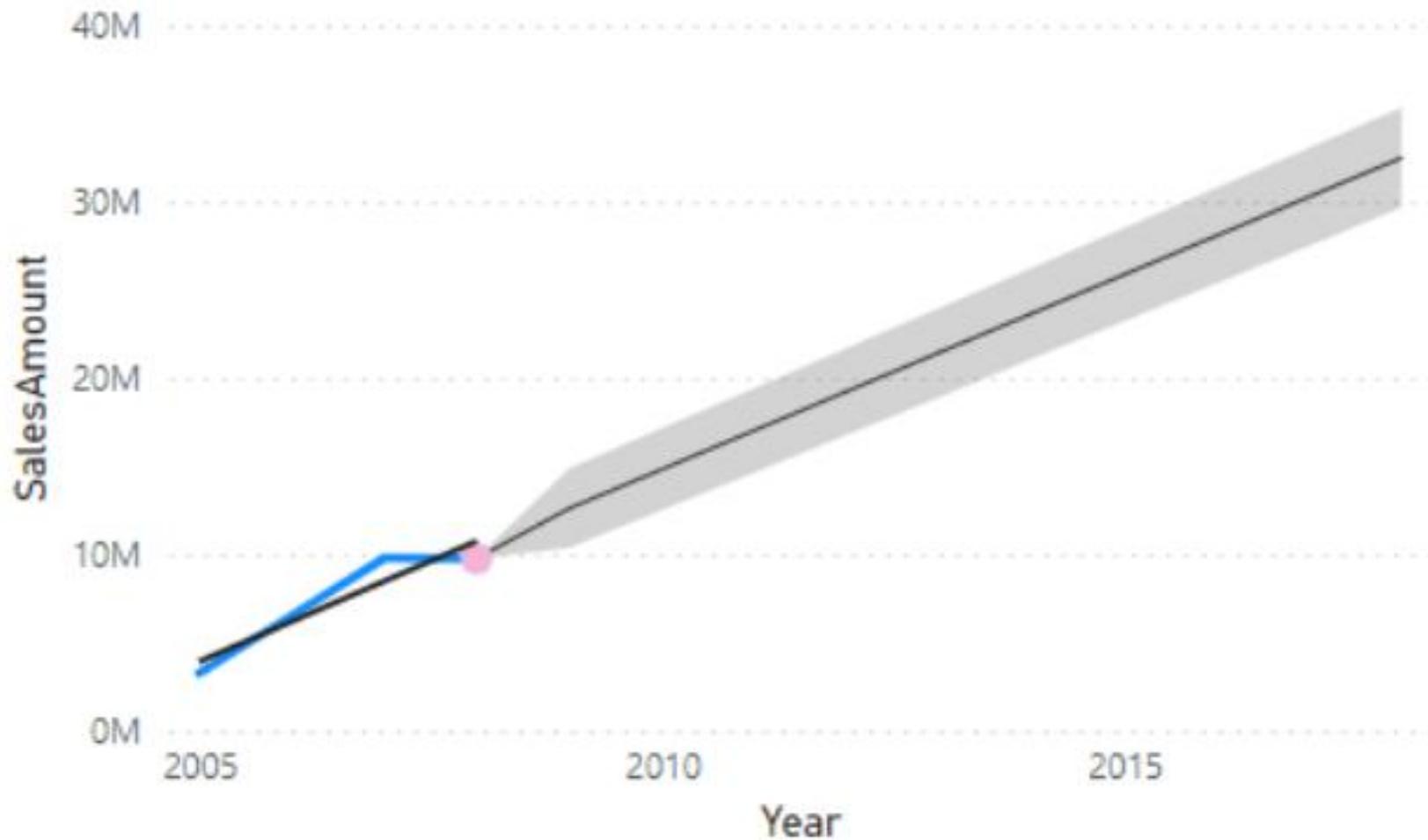


Diagram illustrating the creation of a SalesAmount by Product visualization in Power BI.

The visualization is titled "SalesAmount by Product".

Visualizations pane:

- Build visual: Bar chart icon (highlighted with a red box labeled 1).
- Sheet1: Date, OrderQuantity, Product, SalesAmount.

Fields pane:

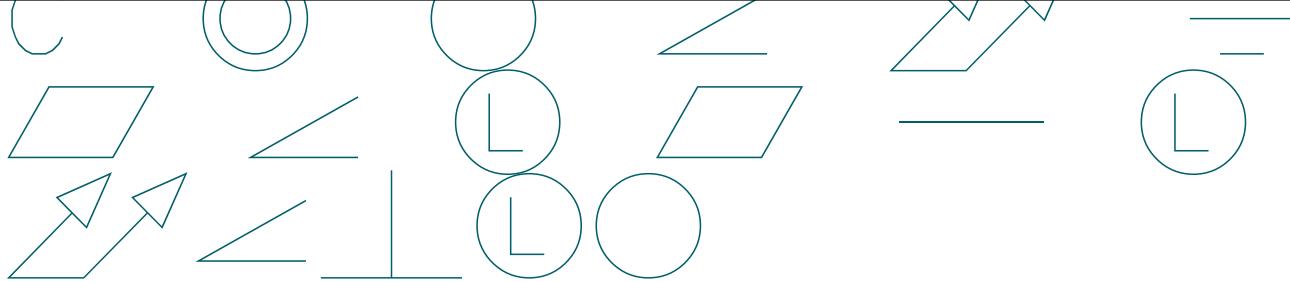
- Search: Search bar.
- Sheet1: Date, OrderQuantity, Product, SalesAmount.

Filters pane:

- Search: Search bar.
- Filters on this visual:
 - Product: is (All)
 - SalesAmount: is (All)
 - Add data fields here
- Filters on this page:
 - Add data fields here
- Filters on all pages:
 - Add data fields here

Details pane:

- Category: Product (selected).
- Details: Add data fields here.
- Values: SalesAmount (selected).
- Tooltips: Add data fields here.



- Power BI Desktop is all about connecting to data, modeling that data, and then visualizing that data. Therefore, it makes sense that you cannot really do much within Power BI without data.
- Report → Modeling → New Table
- Calendar = CALENDAR(DATE(2017 ,1 ,1),
DATE(2019 ,12 ,31))

Untitled - Power BI Desktop

File Home Help Table tools

Name: Calendar

Structure

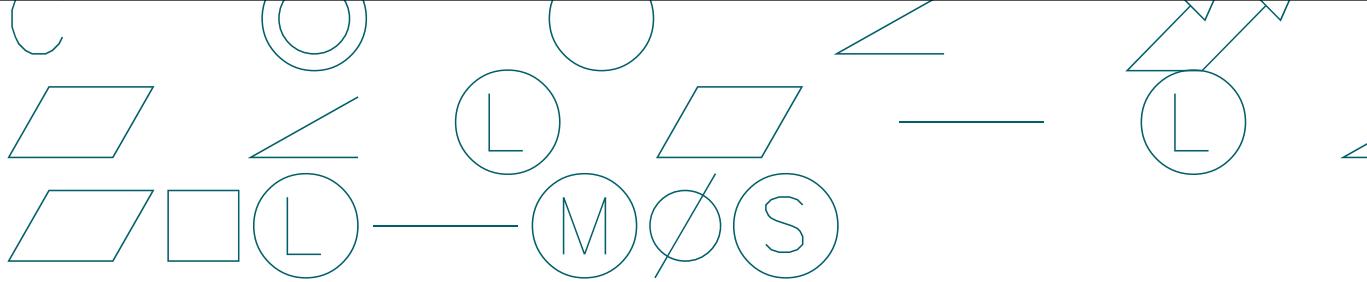
Date

Date
1/1/2017 12:00:00 AM
1/2/2017 12:00:00 AM
1/3/2017 12:00:00 AM
1/4/2017 12:00:00 AM
1/5/2017 12:00:00 AM
1/6/2017 12:00:00 AM
1/7/2017 12:00:00 AM
1/8/2017 12:00:00 AM
1/9/2017 12:00:00 AM
1/10/2017 12:00:00 AM
1/11/2017 12:00:00 AM
1/12/2017 12:00:00 AM
1/13/2017 12:00:00 AM
1/14/2017 12:00:00 AM
1/15/2017 12:00:00 AM
1/16/2017 12:00:00 AM
1/17/2017 12:00:00 AM
1/18/2017 12:00:00 AM
1/19/2017 12:00:00 AM
1/20/2017 12:00:00 AM
1/21/2017 12:00:00 AM
1/22/2017 12:00:00 AM
1/23/2017 12:00:00 AM
1/24/2017 12:00:00 AM
1/25/2017 12:00:00 AM
1/26/2017 12:00:00 AM
1/27/2017 12:00:00 AM

Table: Calendar (1,095 rows)

Mark as date table▼ Calendars Relationships

New measure Quick New measure column New table Calculations



- Calendar table, Fields pane, Table tools tab
- New Column
- Month = FORMAT([Date],"MMMM")

Untitled - Power BI Desktop

File Home Help Table tools Column tools

Name: Month Format: Text

Data type: Text \$ % , .00 Auto

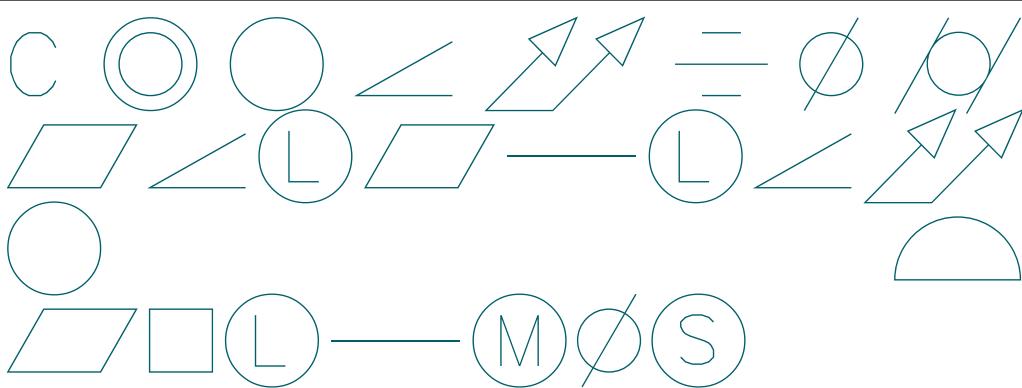
Structure:

X ✓ 1 Month = FORMAT([Date],"MMMM")

Date Month

Date	Month
1/1/2017 12:00:00 AM	January
1/2/2017 12:00:00 AM	January
1/3/2017 12:00:00 AM	January
1/4/2017 12:00:00 AM	January
1/5/2017 12:00:00 AM	January
1/6/2017 12:00:00 AM	January
1/7/2017 12:00:00 AM	January
1/8/2017 12:00:00 AM	January
1/9/2017 12:00:00 AM	January
1/10/2017 12:00:00 AM	January
1/11/2017 12:00:00 AM	January
1/12/2017 12:00:00 AM	January
1/13/2017 12:00:00 AM	January
1/14/2017 12:00:00 AM	January
1/15/2017 12:00:00 AM	January
1/16/2017 12:00:00 AM	January
1/17/2017 12:00:00 AM	January
1/18/2017 12:00:00 AM	January
1/19/2017 12:00:00 AM	January
1/20/2017 12:00:00 AM	January
1/21/2017 12:00:00 AM	January
1/22/2017 12:00:00 AM	January
1/23/2017 12:00:00 AM	January
1/24/2017 12:00:00 AM	January
1/25/2017 12:00:00 AM	January
1/26/2017 12:00:00 AM	January
1/27/2017 12:00:00 AM	January

Table: Calendar (1,095 rows) Column: Month (12 distinct values)



- Year = YEAR([Date])
- MonthNum = MONTH([Date])
- WeekNum = WEEKNUM([Date])
- Weekday = FORMAT([Date],"dddd")
- WeekdayNum = WEEKDAY([Date],2)
- IsWorkDay = IF([WeekdayNum]<=5 ,1, 0)
- The IF function works identically to Excel's IF function.

Untitled - Power BI Desktop

File Home Help Table tools Column tools

Name: Calendar

Mark as date table Calendars Relationships New measure New measure column Calculations

Date Month Year MonthNum WeekNum Weekday WeekdayNum IsWorkDay

Date	Month	Year	MonthNum	WeekNum	Weekday	WeekdayNum	IsWorkDay
1/1/2017 12:00:00 AM	January	2017	1	1	Sunday	7	0
1/2/2017 12:00:00 AM	January	2017	1	1	Monday	1	1
1/3/2017 12:00:00 AM	January	2017	1	1	Tuesday	2	1
1/4/2017 12:00:00 AM	January	2017	1	1	Wednesday	3	1
1/5/2017 12:00:00 AM	January	2017	1	1	Thursday	4	1
1/6/2017 12:00:00 AM	January	2017	1	1	Friday	5	1
1/7/2017 12:00:00 AM	January	2017	1	1	Saturday	6	0
1/8/2017 12:00:00 AM	January	2017	1	2	Sunday	7	0
1/9/2017 12:00:00 AM	January	2017	1	2	Monday	1	1
1/10/2017 12:00:00 AM	January	2017	1	2	Tuesday	2	1
1/11/2017 12:00:00 AM	January	2017	1	2	Wednesday	3	1
1/12/2017 12:00:00 AM	January	2017	1	2	Thursday	4	1
1/13/2017 12:00:00 AM	January	2017	1	2	Friday	5	1
1/14/2017 12:00:00 AM	January	2017	1	2	Saturday	6	0
1/15/2017 12:00:00 AM	January	2017	1	3	Sunday	7	0
1/16/2017 12:00:00 AM	January	2017	1	3	Monday	1	1
1/17/2017 12:00:00 AM	January	2017	1	3	Tuesday	2	1
1/18/2017 12:00:00 AM	January	2017	1	3	Wednesday	3	1
1/19/2017 12:00:00 AM	January	2017	1	3	Thursday	4	1
1/20/2017 12:00:00 AM	January	2017	1	3	Friday	5	1
1/21/2017 12:00:00 AM	January	2017	1	3	Saturday	6	0
1/22/2017 12:00:00 AM	January	2017	1	4	Sunday	7	0
1/23/2017 12:00:00 AM	January	2017	1	4	Monday	1	1
1/24/2017 12:00:00 AM	January	2017	1	4	Tuesday	2	1
1/25/2017 12:00:00 AM	January	2017	1	4	Wednesday	3	1
1/26/2017 12:00:00 AM	January	2017	1	4	Thursday	4	1
1/27/2017 12:00:00 AM	January	2017	1	4	Friday	5	1

Table: Calendar (1,095 rows) Column: IsWorkDay (2 distinct values)



- Date column, Column tools tab
- Data type
- Format

Data Type **Format**

The screenshot shows the Power BI Desktop interface with the 'Column tools' tab selected. In the 'Data Type' section, the 'Name' field is set to 'Date' and the 'Data type' dropdown is also set to 'Date'. In the 'Format' section, the 'Format' dropdown is set to 'Wednesday, h...'. The bottom part of the screenshot shows a preview of a table named 'Calendar' with the following data:

Date	Month	Year	MonthNum	WeekNum	Weekday	WeekdayNum	IsWorkDay
Sunday, January 1, 2017	January	2017	1	1	Sunday	7	0
Monday, January 2, 2017	January	2017	1	1	Monday	1	1
Tuesday, January 3, 2017	January	2017	1	1	Tuesday	2	1

Year
2017
2018

2019

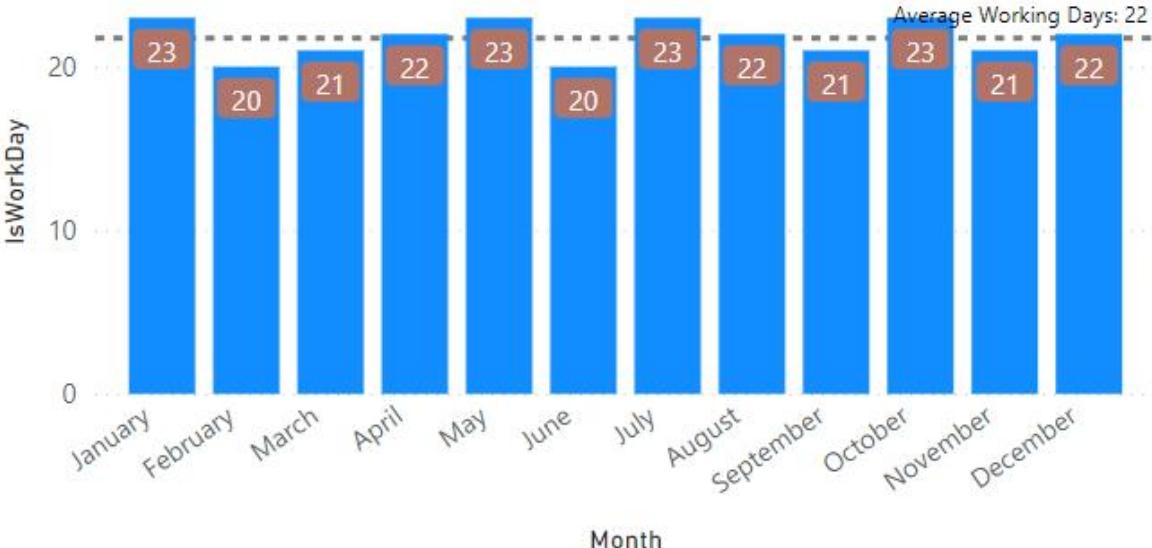
Date
1/1/2019 12/31/2019



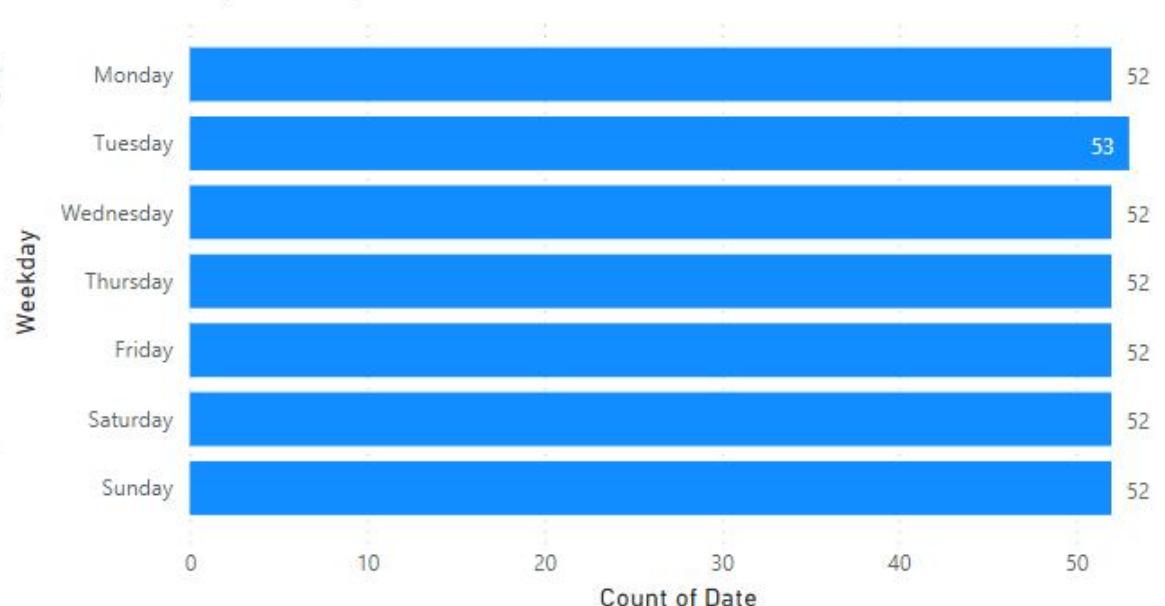
365

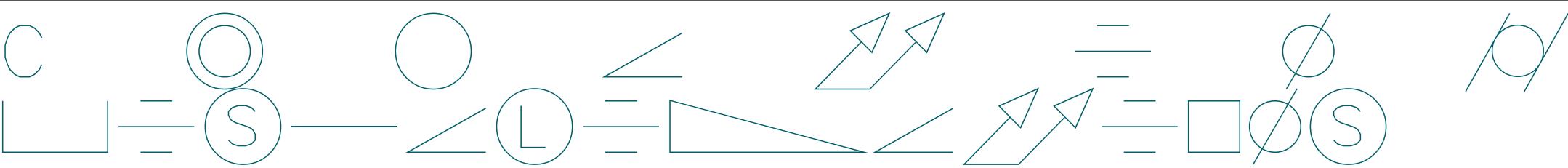
Count of Date

Number of Working Days by Month



Count of Date by Weekday





Practice 1 - Power BI Desktop

Vien Nguyen

Home File Insert Modeling View Help Format Data / Drill

Cut Copy Format painter Paste Get Excel Data SQL Enter Dataverse Recent sources Transform Refresh data New visual Text box More visuals Insert Calculations Sensitivity Share

Clipboard Data Queries

Visualizations Build visual Filters

Fields

Search

Calendar Date IsWorkDay Month MonthNum Weekday WeekdayNum WeekNum Year

X-axis Add data fields here

Y-axis IsWorkDay

Legend Add data fields here

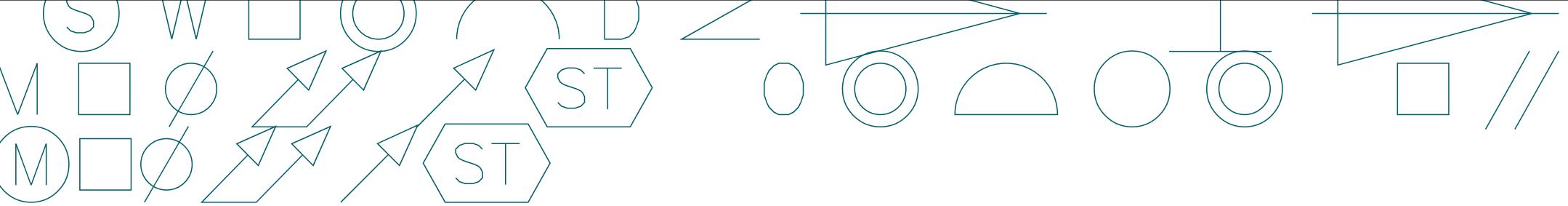
Small multiples Add data fields here

Tooltips Add data fields here

Page 1

Page 1 of 1

Source: Deckler (2022)



Practice 1 - Power BI Desktop

File Home Insert Modeling View Help Format Data / Drill

Visual table Data point table Apply drill down filters to Entire page Show Interactions

Drill actions Drill up Drill down Drill through Groups Data groups Find anomalies

IsWorkDay by Month

IsWorkDay

Month

40 30 20 10 0

April August December February January July June March May November October September

Month

Filters

Search

Filters on this visual

- IsWorkDay is (All)
- Month is (All)

Add data fields here

Filters on this page

Add data fields here

Filters on all pages

Add data fields here

Visualizations

Build visual

Fields

Search

Calendar

- Date
 - IsWorkDay
 - Month
 - MonthNum
 - Weekday
 - WeekdayNum
 - WeekNum
 - Year

X-axis

Month

Y-axis

IsWorkDay

Legend

Small multiples

Add data fields here

Tooltips

Drill through

Cross-report (Off)

Keep all filters (On)

Add drill-through fields here

Page 1

Page 1 of 1

103%

A screenshot of the Power BI Desktop interface showing a bar chart titled "IsWorkDay by Month". The chart displays the count of workdays for each month. The X-axis represents the months from April to September, and the Y-axis represents the count of workdays, ranging from 0 to 70. The bars are blue. The chart is located in the main workspace. On the right side, the "Fields" pane is open, showing the "Calendar" group expanded. The "Month" field is selected and highlighted with a red box. A red arrow points from the "X-axis" section of the chart's filter pane to the "Month" field in the Fields pane.

Practice 1 - Power BI Desktop

Vien Nguyen

File Home Insert Modeling View Help Format Data / Drill Table tools

Name: Month
Data type: Text

\$% Format: Text
Summarization: Don't summarize
Data category: Uncategorized

Structure
Formatting
Properties

Column tools 2

Sort by column
Data groups
Manage relationships
Relationships
New column
Calculations

Month

Date

IsWorkDay

MonthNum 3

Weekday

WeekdayNum

WeekNum

Year

1 Month = FORMAT([Date],"MMMM")

70
60
50
40
30
20
10
0

January February March April May June July August September October November December

Month

IsWorkDay

Filters on this visual

IsWorkDay is (All)

Month is (All)

Add data fields here

Filters on this page

Add data fields here

Filters on all pages

Add data fields here

Legend

Add data fields here

Small multiples

Add data fields here

Tooltips

Add data fields here

Drill through

Cross-report

Keep all filters On

Add drill-through fields here

Visualizations

Build visual

Search

Filters on this visual

IsWorkDay is (All)

Month is (All)

Add data fields here

Filters on this page

Add data fields here

Filters on all pages

Add data fields here

Legend

Add data fields here

Small multiples

Add data fields here

Tooltips

Add data fields here

Drill through

Cross-report

Keep all filters On

Add drill-through fields here

Fields

Search

Calendar

Date 1

IsWorkDay

Month 2

MonthNum

Weekday

WeekdayNum

WeekNum

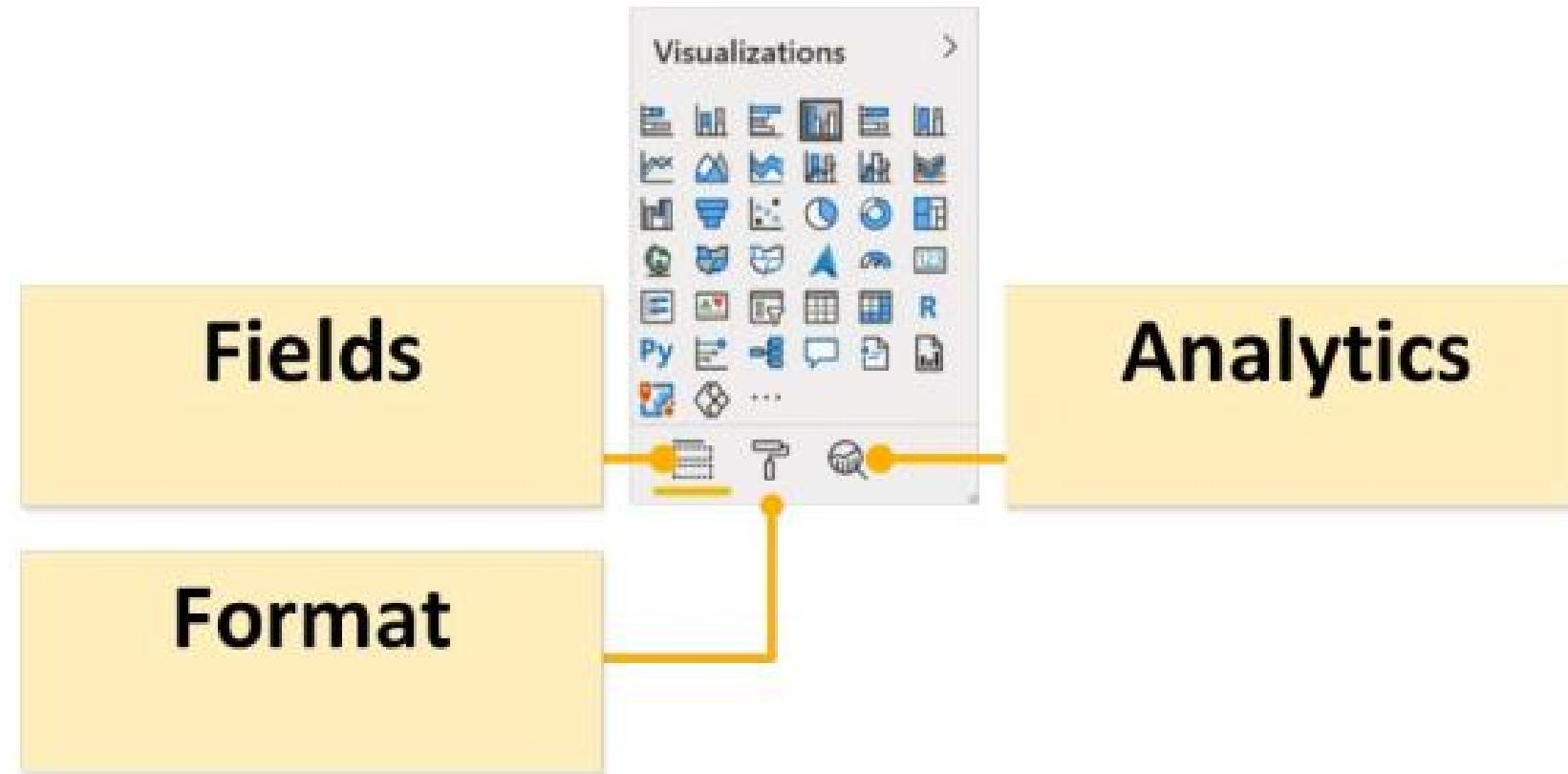
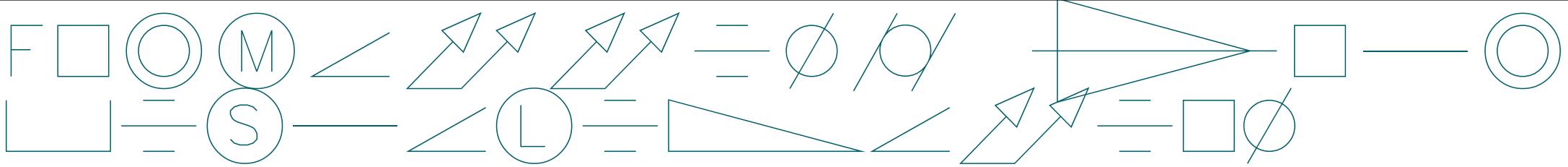
Year

Page 1 +

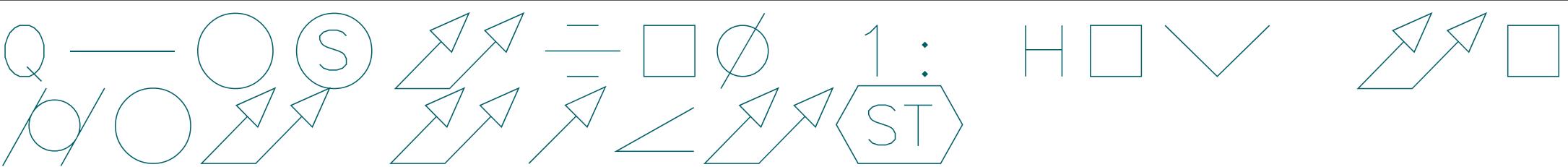
Page 1 of 1

103%

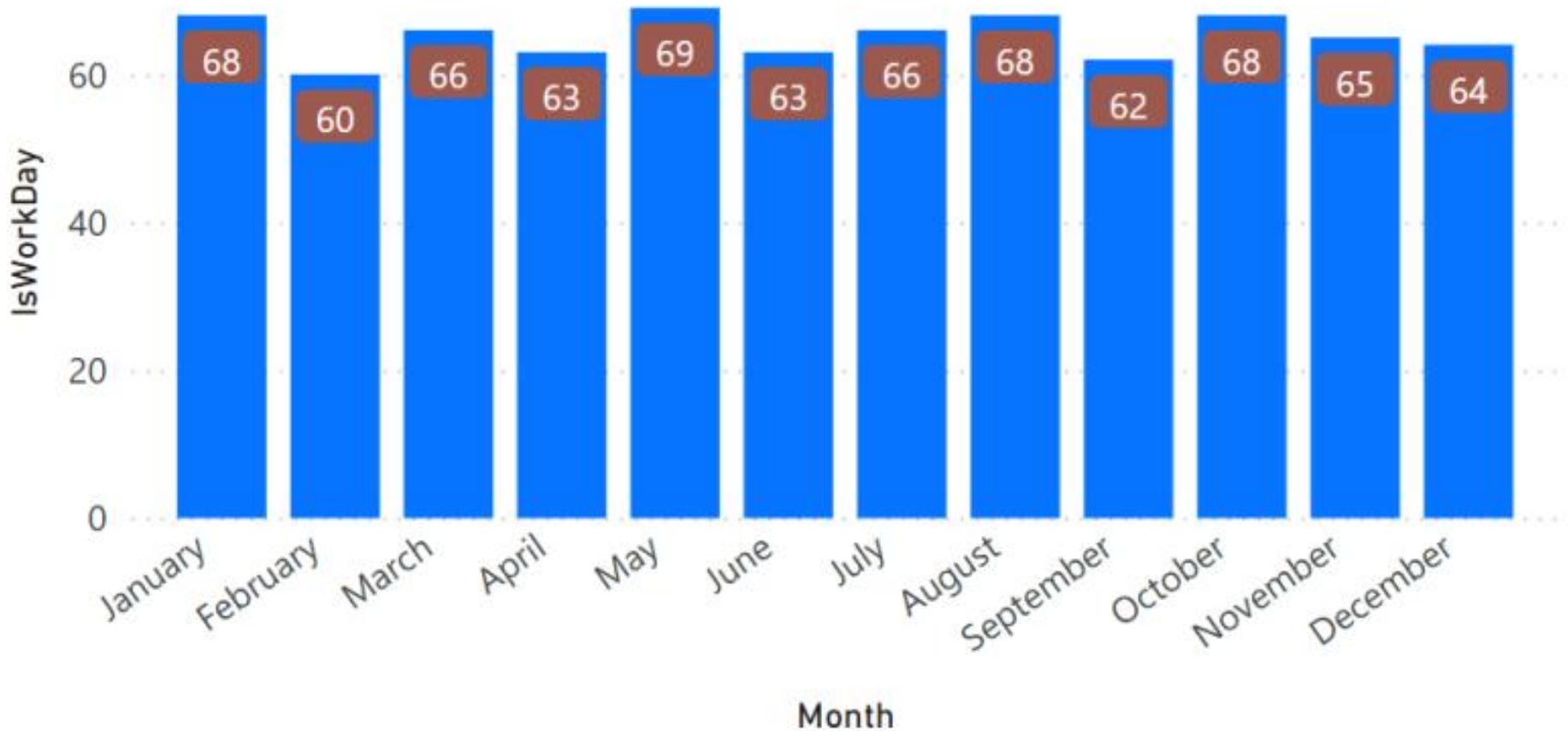
The screenshot shows a bar chart in Power BI Desktop with blue bars representing the number of workdays for each month from January to December. The Y-axis ranges from 0 to 70. The X-axis labels the months. A context menu, titled 'Column tools' and numbered '2', is open over the first bar. This menu includes options like 'Sort by column', 'Data groups', 'Manage relationships', 'New column', and 'Calculations'. A sub-menu for 'IsWorkDay' is also open, listing 'MonthNum' and 'Weekday', with 'MonthNum' highlighted and numbered '3'. The 'Fields' pane on the right lists various date-related fields under the 'Calendar' section, with 'Month' selected and highlighted with a red box and numbered '1'. The 'Visualizations' pane shows the current chart setup.



Source: Deckler (2022)

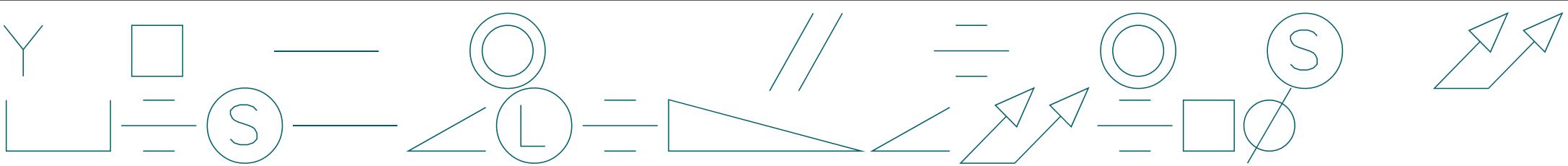


Number of Working Days by Month

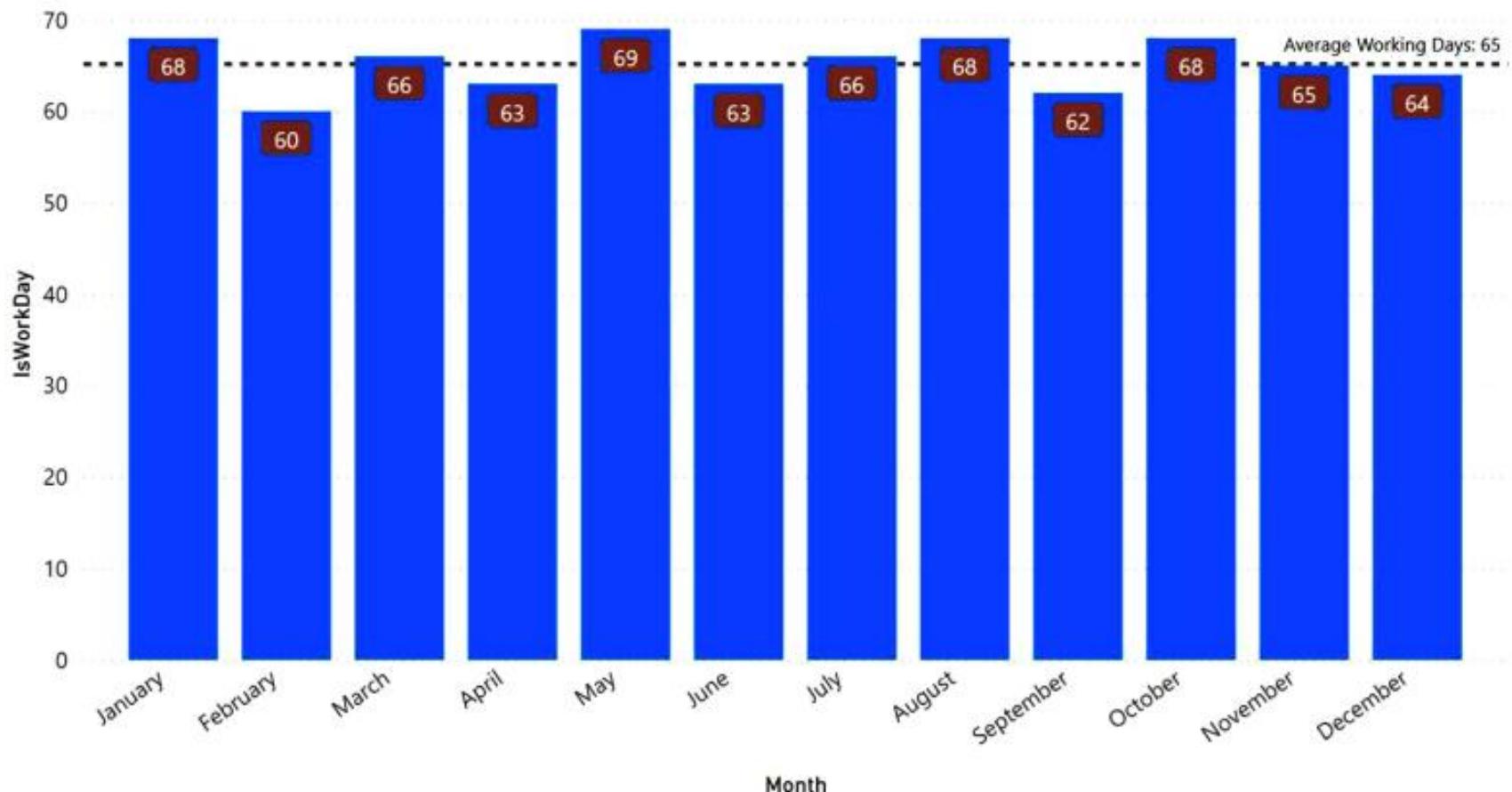


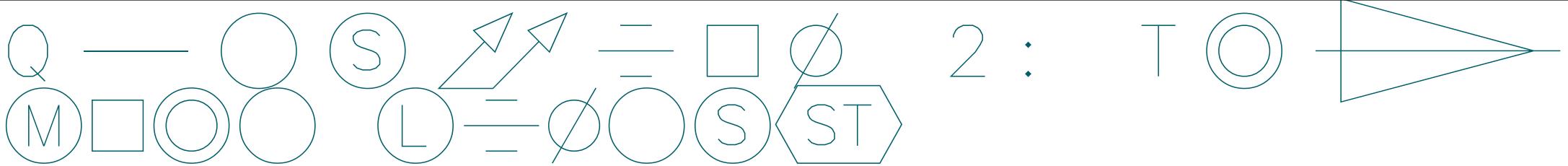


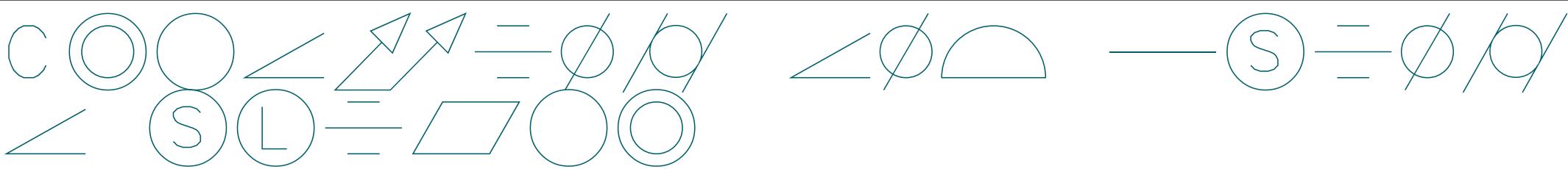
1. Ensure that your first visualization is selected in the canvas.
2. Click on the **Analytics** sub-pane. Here again, we see several expandable sections.
3. Expand the **Average Line** section and then click **+ Add**. Note the dotted line that appears on your visual.
4. Change the text **Average Line 1** to **Average Working Days**. Underneath this, note that **Measure** is set to **IsWorkDay**.
5. Below **Measure**, change **Color** to black and change **Position** to **Behind**.
6. Toggle on the **Data label**, change **Color** to black, change **Text** to **Both**, and finally, change **Horizontal Position** to **Right**.



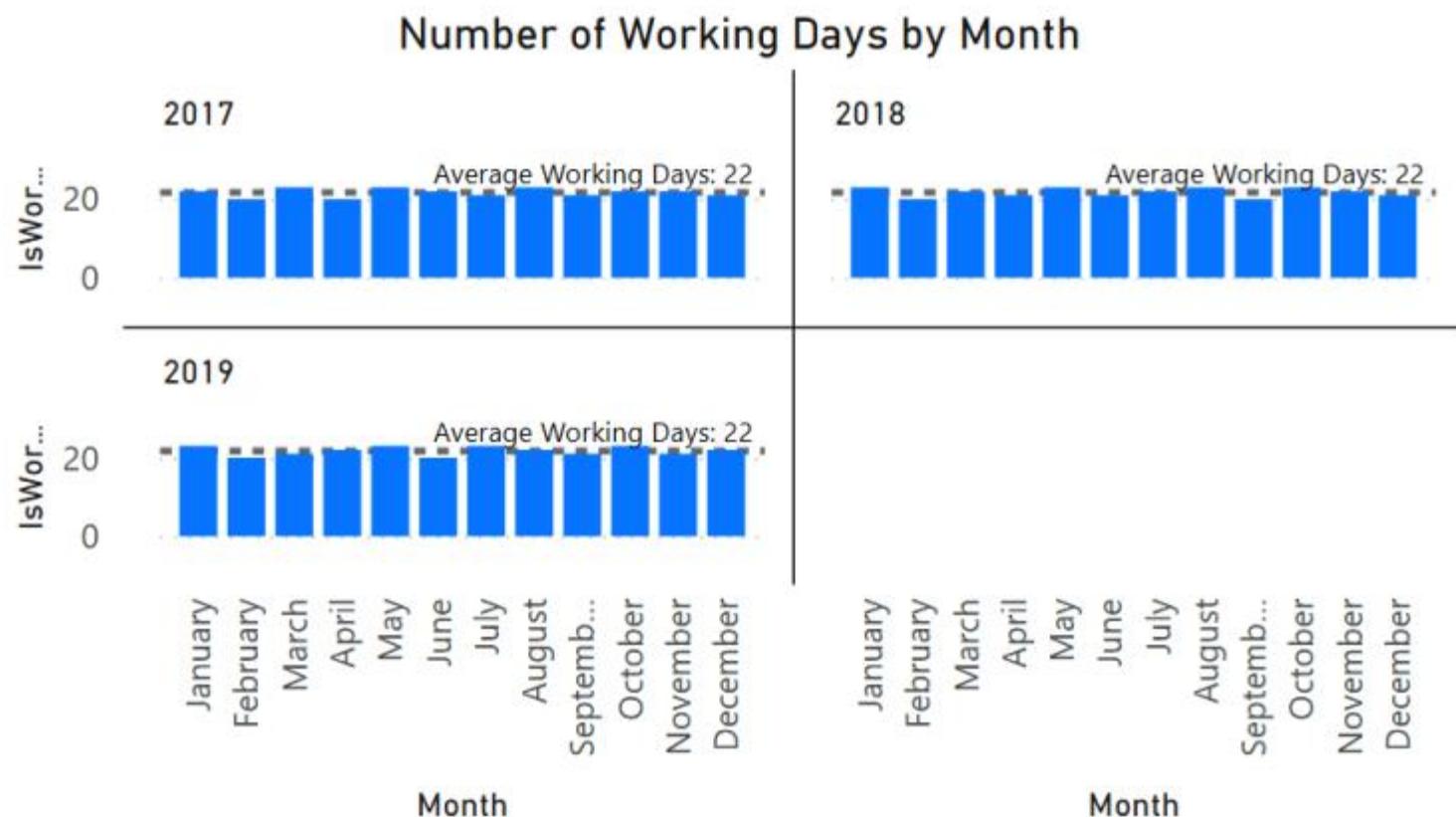
Number of Working Days by Month

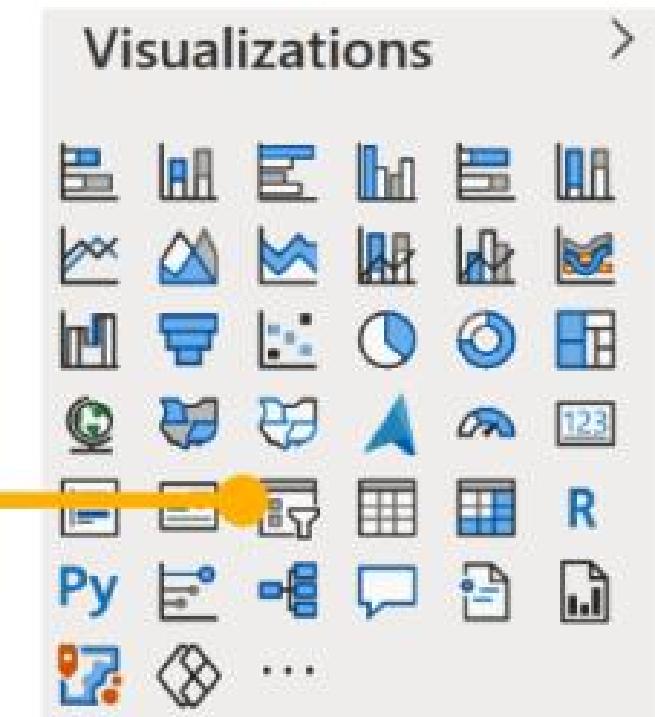






1. Ensure that your first visual is selected and change to the **Fields** sub-pane in the **Visualizations** pane.
2. Drag the **Year** column from the **Fields** pane into the **Small multiples** field in the **Visualizations** pane. Your visualization is now divided into four quadrants with each year displayed individually, as shown in the following diagram:





Source: Deckler (2022)

Practice 1 - Power BI Desktop

Vien Nguyen

File Home Insert Modeling View Help Format Data / Drill

Cut Copy Format painter Paste

Get data workbook hub Data SQL Server Enter Dataverse Recent sources Transform Refresh data New visual Text box More visuals New measure Quick Sensitivity Publish

Clipboard Data Queries Insert Calculations Sensitivity Share

Number of Working Days by Month

Year

Select all 2017 2018 2019

Filters

Visualizations

Build visual

Fields

Search

Calendar Date Date Hierarchy Year Quarter Month Day IsWorkDay Month MonthNum Weekday WeekdayNum WeekNum Year

Drill through

Cross-report Off

Keep all filters On

Add drill-through fields here

Page 1 +

Page 1 of 1

82%

Month	IsworkDay
January	22
February	20
March	23
April	20
May	23
June	22
July	21
August	23
September	21
October	22
November	22
December	21

Practice 1 - Power BI Desktop

Vien Nguyen

File Home Insert Modeling View Help Format Data / Drill Table tools Column tools

Name Month Format Text Summarization Don't summarize Data category Uncategorized

Data type Text \$ % , . , 00 Auto Sort by column Data groups Manage relationships New column

Structure Formatting Properties

Year, Month

Select all

2017

- January
- February
- March
- April
- May
- June
- July
- August
- September
- October
- November
- December

2018

- January
- February
- March
- April
- May
- June
- July
- August

Number of Working Days by Month

Highest Working Days: 23

Average Working Days: 22

Lowest Working Days: 20

IsworkDay

Month

January February March April May June July August September October November December

Visualizations

Build visual

Filters

Fields

Search

Calendar

Date

Date Hierarchy

- Year
- Quarter
- Month
- Day
- IsWorkDay
- Month

MonthNum

Weekday

WeekdayNum

WeekNum

Year

Drill through

Cross-report

Keep all filters

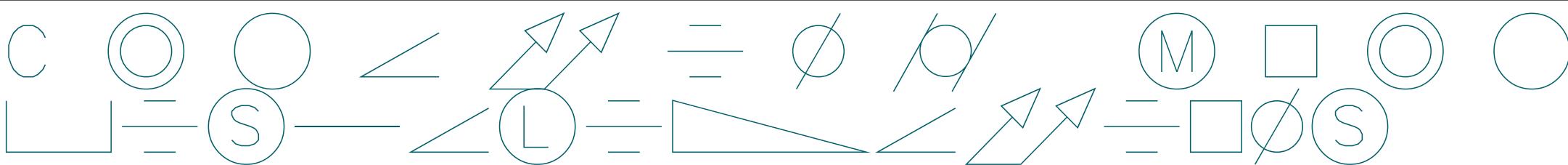
Add drill-through fields here

Page 1

Page 1 of 1

82%

Month	Working Days
January	23
February	20
March	22
April	21
May	23
June	21
July	21
August	23
September	21
October	22
November	22
December	21



Chevron

A screenshot of the Power BI interface showing a date hierarchy. On the left is a table of data from 2017 Qtr 1 January 1 to January 12. In the middle is the 'Values' pane with a 'Date' field selected. A context menu is open over the 'Date' field, with the 'Date Hierarchy' option highlighted. To the right is a list of available date-related fields: Weekday, WeekdayNum, WeekNum, and Year.

Year	Quarter	Month	Day
2017	Qtr 1	January	1
2017	Qtr 1	January	2
2017	Qtr 1	January	3
2017	Qtr 1	January	4
2017	Qtr 1	January	5
2017	Qtr 1	January	6
2017	Qtr 1	January	7
2017	Qtr 1	January	8
2017	Qtr 1	January	9
2017	Qtr 1	January	10
2017	Qtr 1	January	11
2017	Qtr 1	January	12

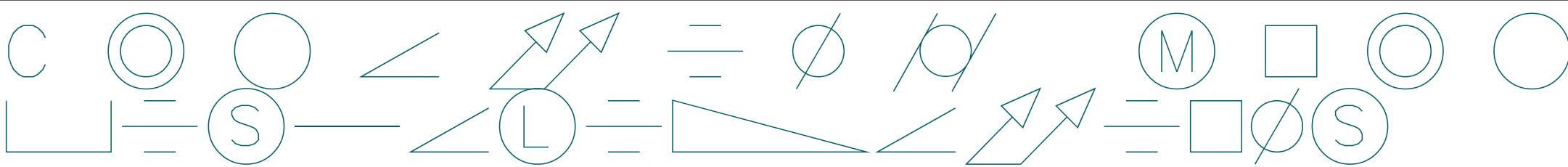
Clear selections

A screenshot of the Power BI interface showing a date selector. A context menu is open over a date range selector, with the 'Between' option highlighted. To the left is a list of dates from 1/1/2017 to 1/12/2017.

Date
1/1/2017
1/2/2017
1/3/2017
1/4/2017
1/5/2017
1/6/2017
1/7/2017
1/8/2017
1/9/2017
1/10/2017
1/11/2017
1/12/2017

Chevron

Source: Deckler (2022)



Practice 1 - Power BI Desktop

Vien Nguyen

Home

File Insert Modeling View Help

Cut Copy Format painter

Get data Excel Data Server Enter data Recent sources

Transform Refresh data New visual Text box More visuals

New measure measure Quick Sensitivity

Sensitivity Publish

Clipboard Data Queries Insert Calculations Sensitivity Share

Number of Working Days by Month

Highest Working Days: 69
Average Working Days: 65
Lowest Working Days: 60

IsWorkDay

Month	Working Days
January	68
February	60
March	66
April	63
May	69
June	65
July	66
August	68
September	62
October	68
November	65
December	64

Visualizations Fields

Build visual

Filters

Search

Calendar Date

IsWorkDay Month MonthNum Weekday WeekdayNum WeekNum Year

Values Add data fields here

Drill through Cross-report Off

Keep all filters On

Add drill-through fields here

Page 1 +

Page 1 of 1

82%

Practice 1 - Power BI Desktop

Vien Nguyen

File Home Insert Modeling View Help Format Data / Drill

Cut Copy Format painter

Get data Excel Data SQL Server Enter Dataverse Recent sources Transform Refresh data New visual Text box More visuals New measure Quick Sensitivity Publish

Clipboard

Year Quarter Month Day

Number of Working Days by Month

Highest Working Days: 69.

Average Working Days: 65

Lowest Working Days: 60

IsworkDay

Month

January February March April May June July August September October November December

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Visualizations

Build visual

Filters

Search

Calendar Date IsWorkDay Month MonthNum Weekday WeekdayNum WeekNum Year

Date

Remove field

New quick measure

Show items with no data

Date

Date Hierarchy

Drill through

Cross-report

Keep all filters

Add drill-through fields here

Page 1

Page 1 of 1

82%

Month	Working Days
January	68
February	60
March	66
April	63
May	69
June	63
July	66
August	68
September	62
October	68
November	65
December	64

Practice 1 - Power BI Desktop

Vien Nguyen

File Home Insert Modeling View Help Format Data / Drill

Cut Copy Format painter Clipboard

Get data Excel workbook hub Data SQL Server Enter Dataverse Recent sources Transform Refresh data New visual Text box More visuals New measure Quick Sensitivity Publish

Clipboard

Date

1/1/2017 12/31/2019

Back to report

Filters

2

Between

Before

After

List

Dropdown

Relative Date

Relative Time

Visualizations

Build visual

1

Calendar

Date

IsWorkDay

Month

MonthNum

Weekday

WeekdayNum

WeekNum

Year

Fields

Search

Date

Drill through

Cross-report

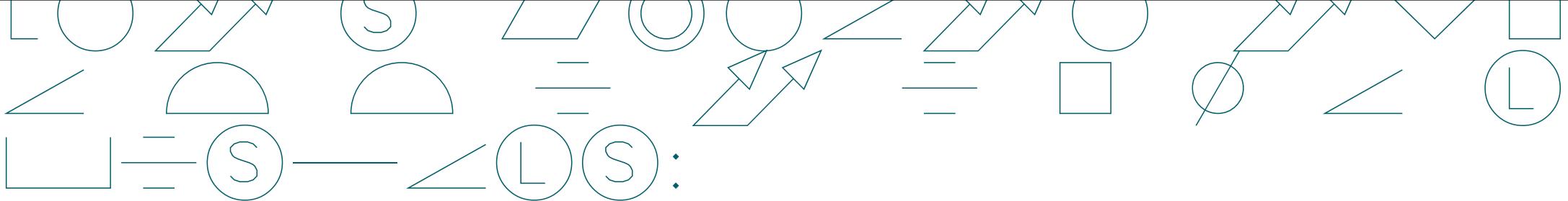
Keep all filters

Add drill-through fields here

Page 1 of 1

100%

The screenshot shows the Power BI Desktop interface with the 'Home' tab selected. A date filter is applied from 1/1/2017 to 12/31/2019. A context menu is open over the date range, with the 'Between' option highlighted. The Visualizations pane shows the 'Calendar' node expanded, with the 'Date' field selected. The Fields pane shows the 'Date' field listed under the 'Field' section.



1. For the first visual, drag the **Weekday** column in the **Fields** pane onto a blank area of the canvas. Note that a single-column table of the distinct values for the days of the week appears.
2. Drag the **Date** column from the **Fields** pane into this new visual. Four columns are automatically created for **Year**, **Quarter**, **Month**, and **Day**. As before, change this from **Date Hierarchy** to just **Date**.
3. Change this visual to **Clustered bar chart**. That is the third visual over in the top row.
4. In the **Fields** sub-pane within the **Visualizations** pane, drag and drop **Date** from the **Legend** field to the **Values** field. Note that the text changes to **Count of Date**.
5. Drag the visual so that the top of the visual lines up with the top of our original column chart visual and then resize it so that this visual extends to the middle of the page horizontally and all of the way to the bottom of the page.
6. Notice that the weekdays are not in order. Use the ellipsis (...) in the visual to change **Sort by** to **Weekday** and **Sort ascending**.
7. Click on the **Weekday** column in the **Fields** pane and then, from the **Column tools** tab, choose the **Sort by** column and then **WeekdayNum**. This new visual now displays Monday at the top and Sunday at the bottom.
8. Finally, format this visual to display **Data labels**.
9. Click on a blank area of the canvas and select the **Card** visual. This is the visual in the fourth row from the top and farthest to the right.
10. Drag **Date** from the **Fields** pane into this visual.
11. In the **Fields** sub-pane of the **Visualizations** pane, use the drop-down arrow next to **Earliest Date** and change this to **Count**.
12. Lastly, reposition the **Card** visual to be in the center of the blank area of the canvas and resize the visual to be just big enough to fully display the **Count of Date** text.

Source: Deckler (2022)

Practice 1 - Power BI Desktop

Vien Nguyen

File Home Insert Modeling View Help

Cut Copy Format painter Clipboard

Get data workbook hub Data SQL Server Enter Data Recent sources Transform Refresh data New visual Text box More visuals New measure Quick measure Sensitivity Share

Date 1/1/2017 12/31/2019

Year
2017
2018
2019

Count of Date 1095

Number of Working Days by Month

IsWorkDay

Month	IsWorkDay
January	68
February	60
March	66
April	63
May	69
June	63
July	66
August	68
September	62
October	68
November	65
December	64

Highest Working Days: 69
Average Working Days: 65
Lowest Working Days: 60

Count of Date by Weekday

Weekday

Weekday	Count of Date
Monday	150
Tuesday	150
Wednesday	150
Thursday	150
Friday	150
Saturday	150
Sunday	150

Filters

Search

Add data fields here

Filters on this page

Filters on all pages

Values

Add data fields here

Drill through

Cross-report Off

Keep all filters On

Add drill-through fields here

Visualizations

Build visual

Fields

Search

Calendar Date

IsWorkDay

Month

MonthNum

Weekday

WeekdayNum

WeekNum

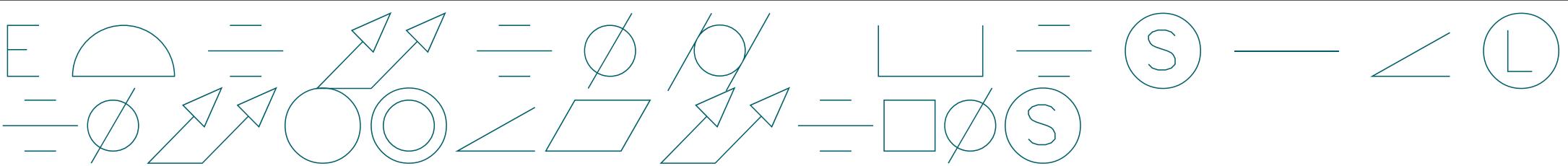
Year

Page 1

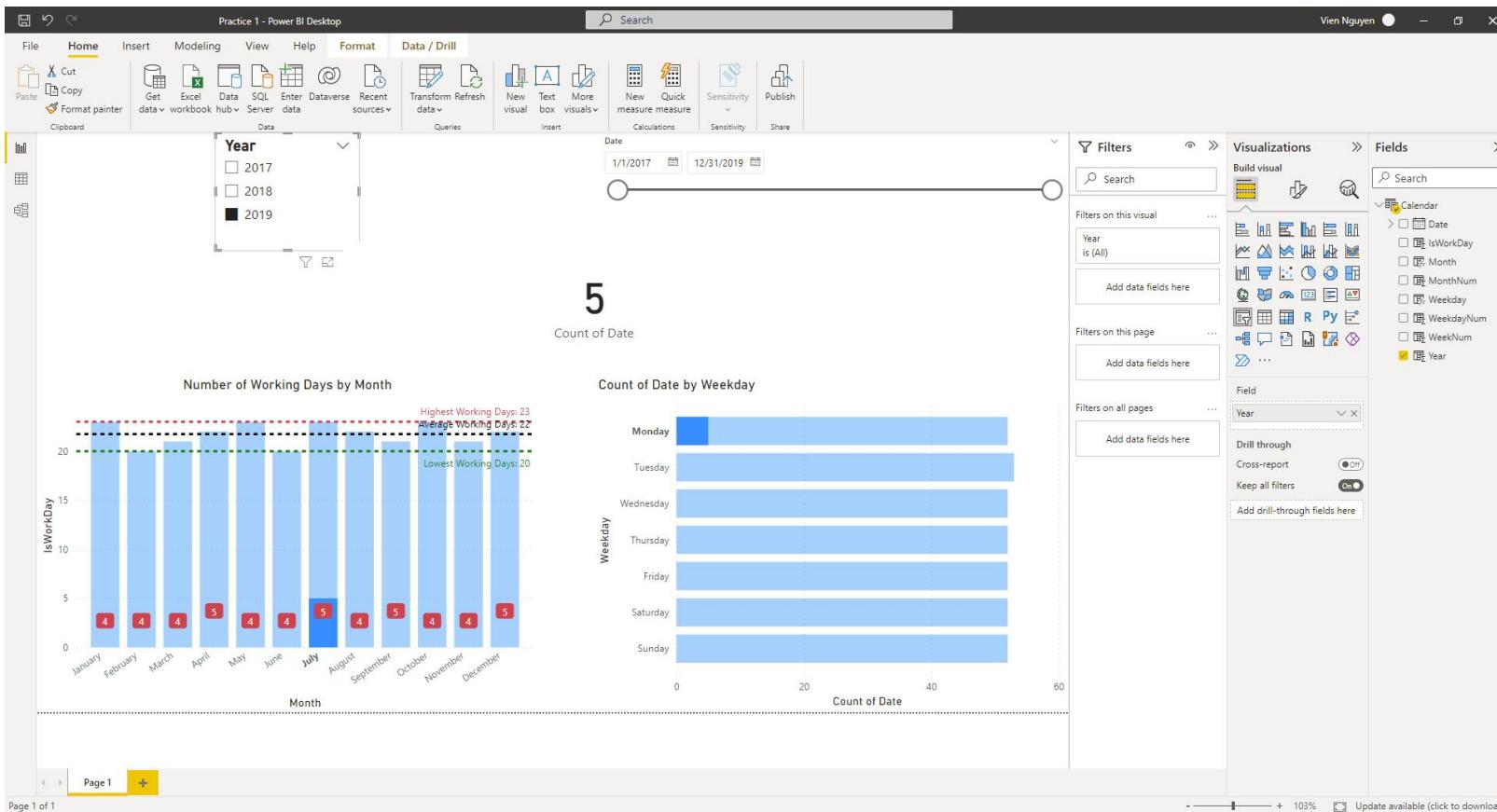
103%

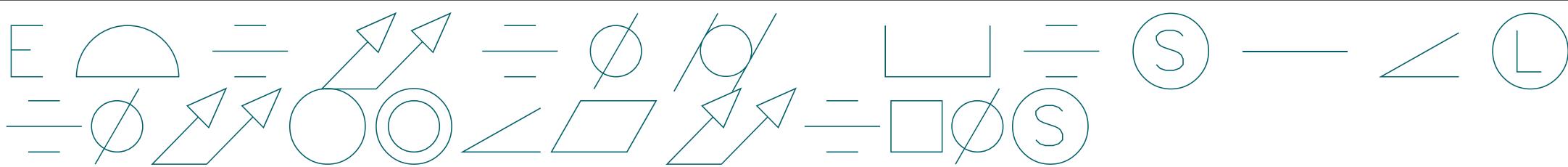
Update available (click to download)

Page 1 of 1

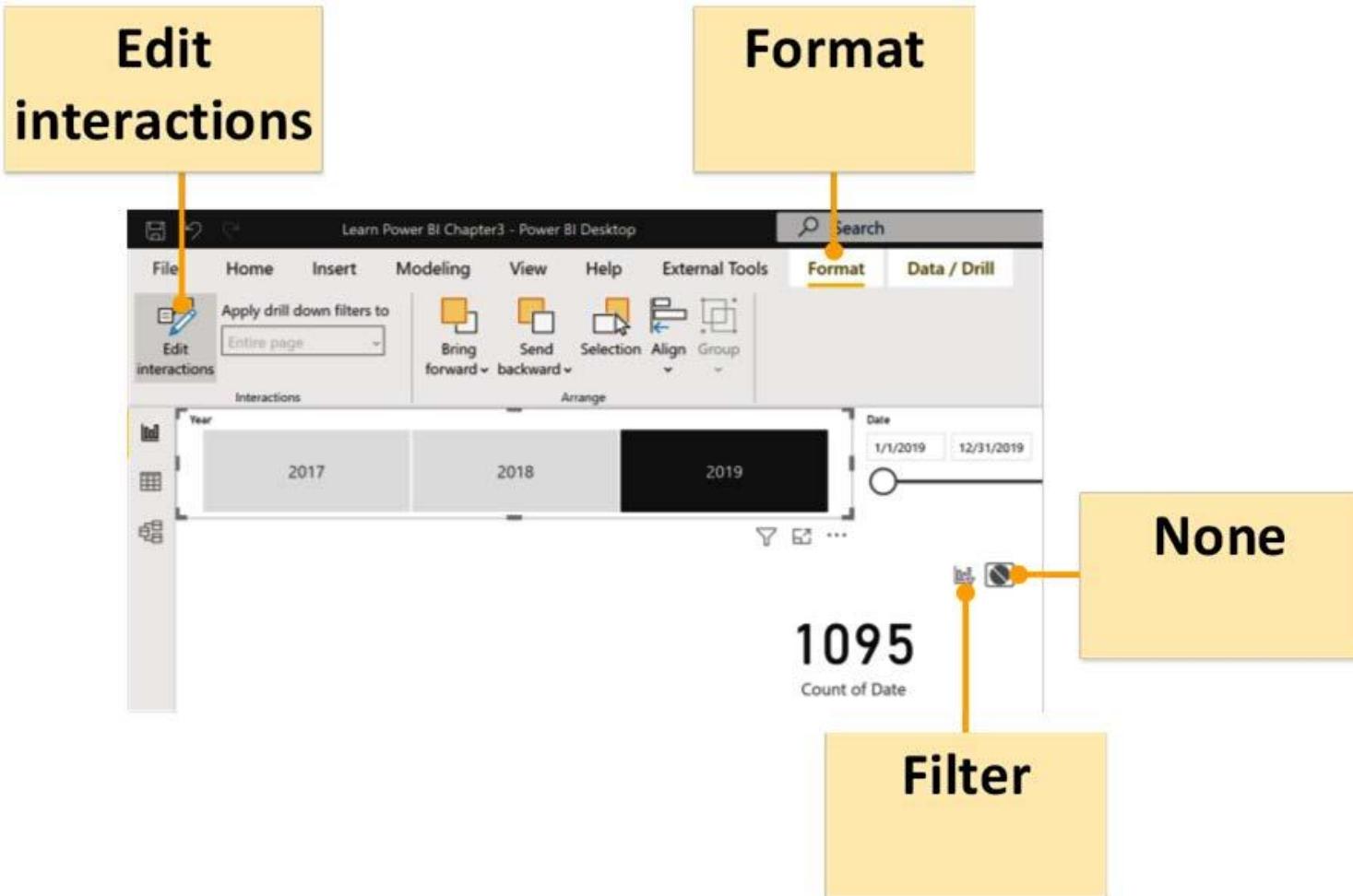


- With **Monday** selected in the bar chart, hover your mouse over **July** again, but, this time, hold down the *Ctrl* key and then click the **July** column. Note that the **Card** visual changes to read **5**. By using the *Ctrl* key, we can make values selected within visuals additive with one another.

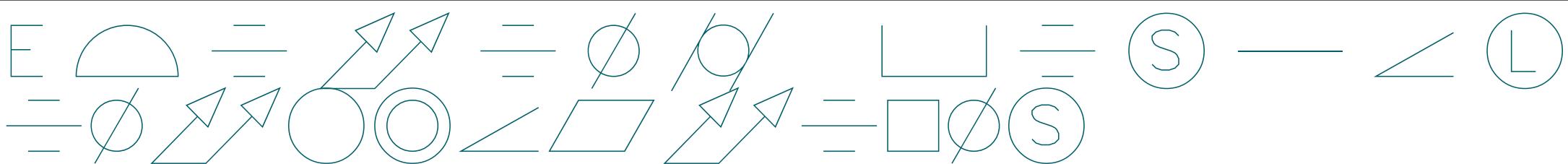




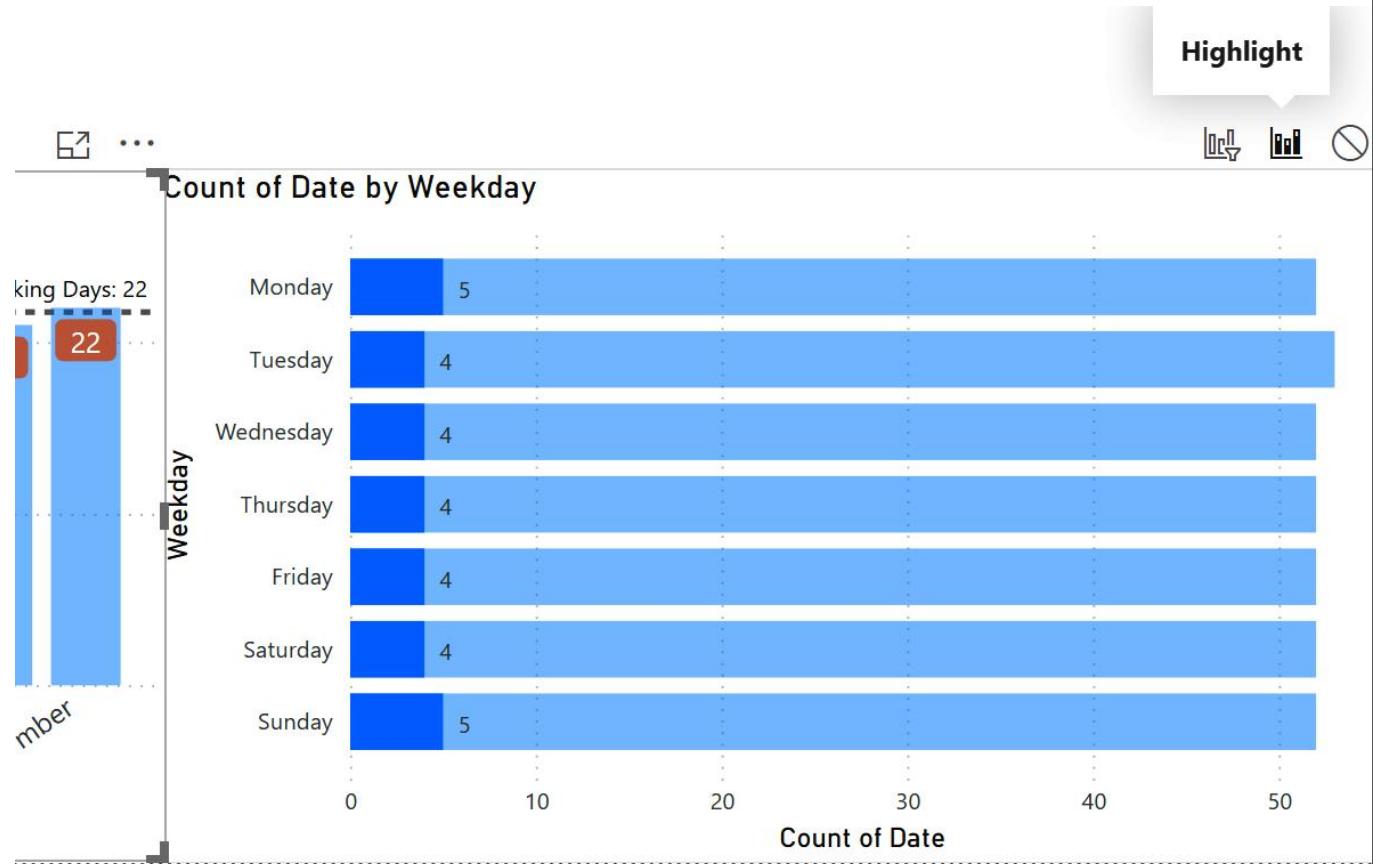
- One icon looks like a column chart with a small funnel, and the other is a circle with a line through it. If you hover over these icons, you will see that the chart icon is called **Filter** and the circle icon is called **None**. The icon that is shaded in gray is the currently active interaction mode.



Source: Deckler (2022)



- Select the column chart visual in the lower left corner and hover your mouse pointer over the bar chart. Notice that a third icon appears that is simply shaded in the column chart. This is the **Highlight** icon, which explains the behavior of this chart when we clicked on months within our column chart:
 - Card visuals cannot filter other visuals.



Source: Deckler (2022)

Practice 1 - Power BI Desktop

Vien Nguyen

File Home Insert Modeling View Help Format Data / Drill

Cut Copy Format painter Paste Get data workbook hub Data SQL Server Enter data Recent sources Transform Refresh data New visual Text box More visuals New measure Quick measure Sensitivity Publish

Clipboard Data Queries Insert Calculations Sensitivity Share

Year
2017
2018
2019

Date
1/1/2017 12/31/2019

Filters

Search

Filters on this visual
Year is (All)

Add data fields here

Filters on this page

Add data fields here

Filters on all pages

Add data fields here

Build visual

Visualizations

Fields

Search

Calendar

1095 Count of Date

Still 1095 days. Why?

Number of Working Days by Month

Count of Date by Weekday

Highest Working Days: 23
Average Working Days: 22
Lowest Working Days: 20

IsWorkDay

Month

January February March April May June July August September October November December

Weekday

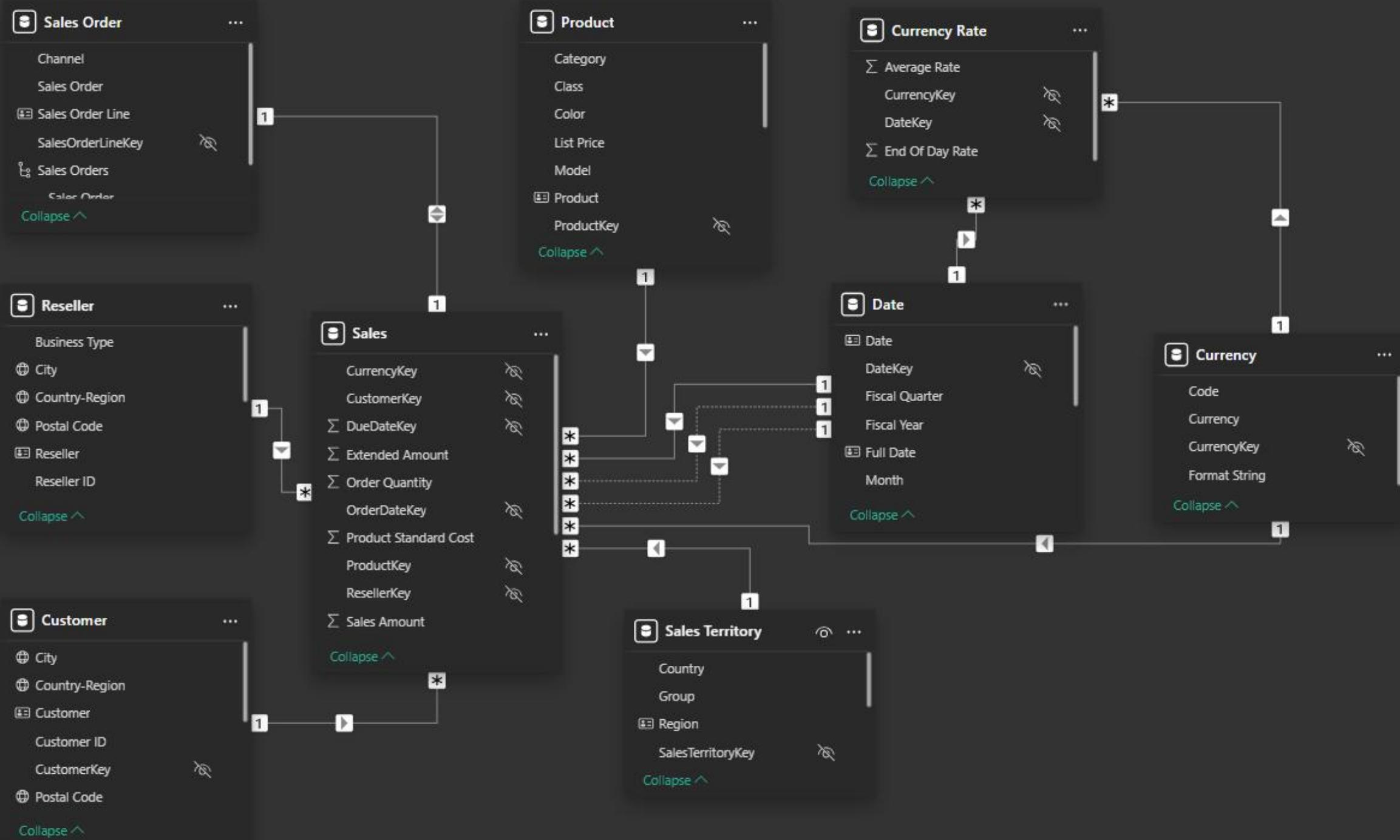
Monday Tuesday Wednesday Thursday Friday Saturday Sunday

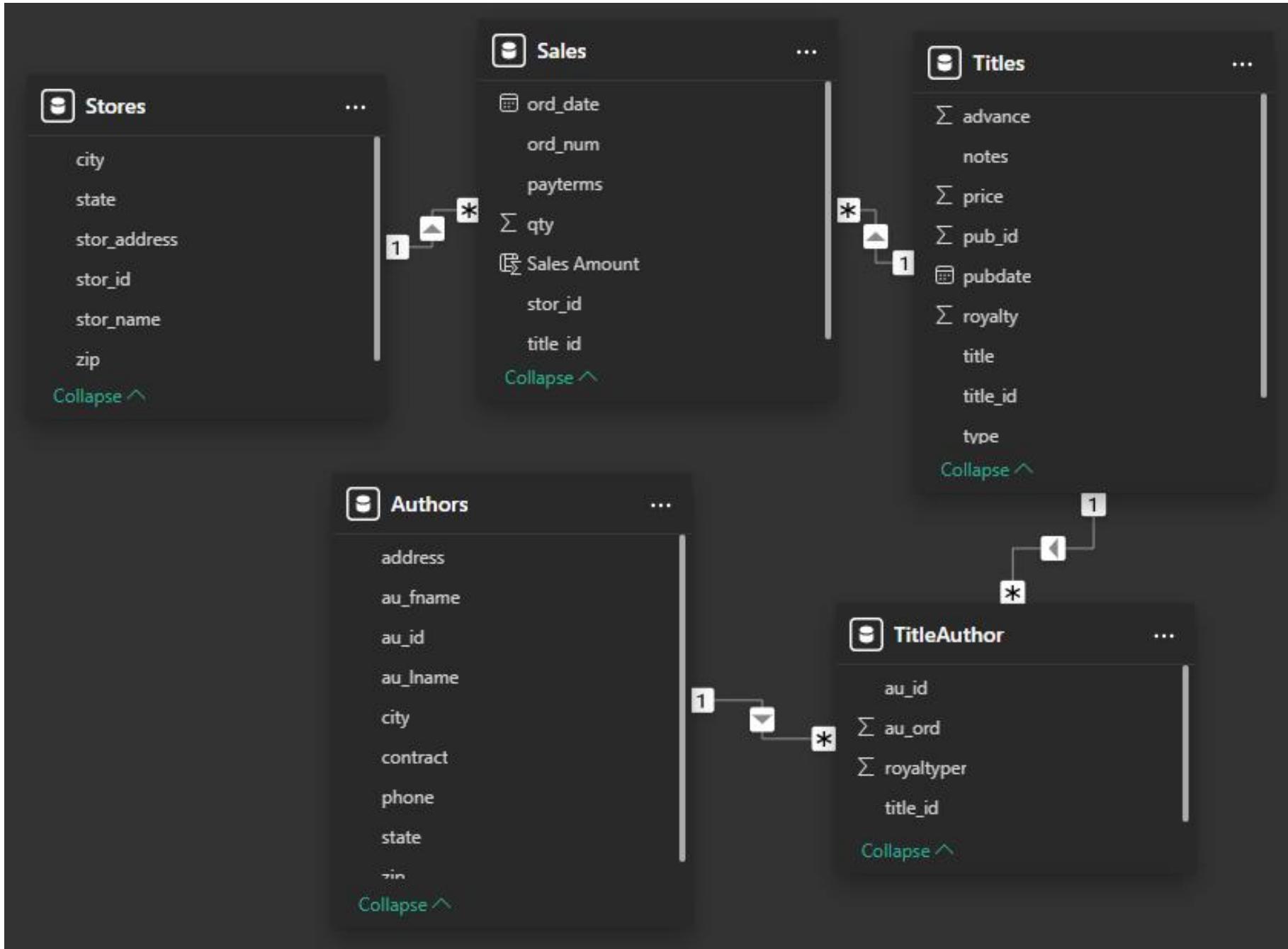
Count of Date

Page 1 +

103% Update available (click to download)

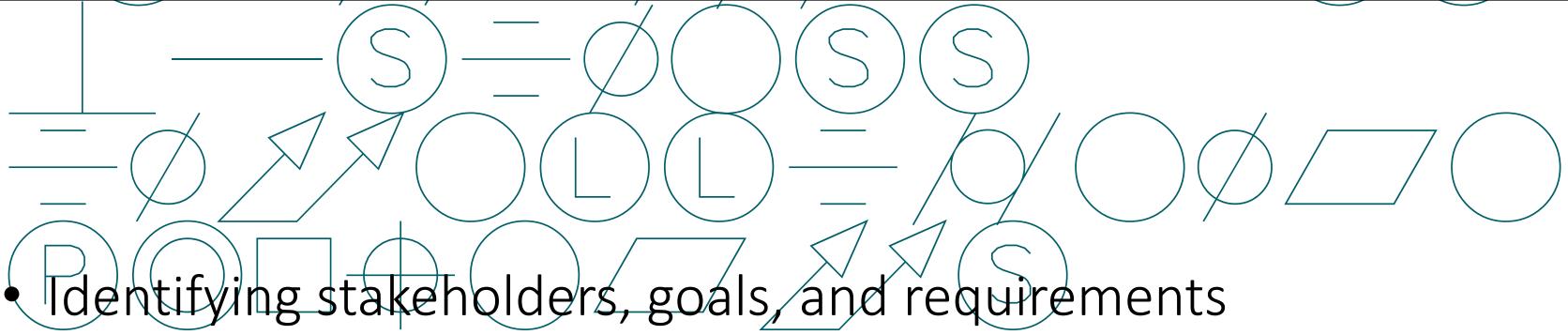
Page 1 of 1



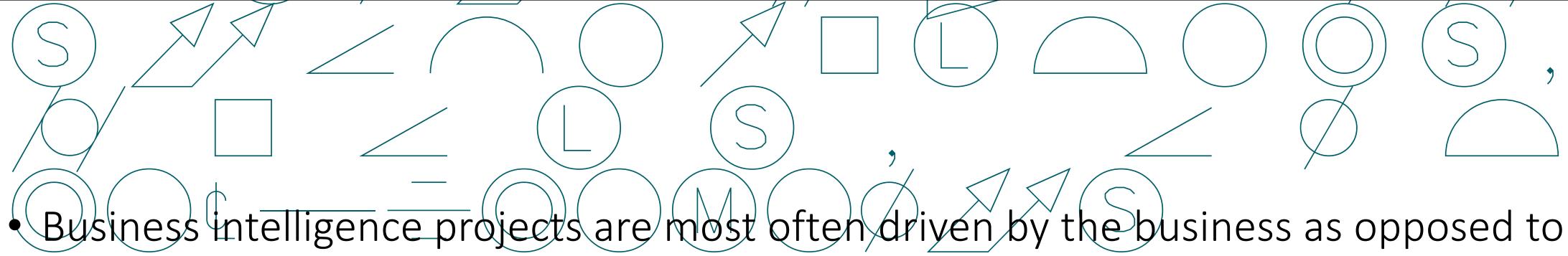




- Choose an enterprise you love (e.g. listed/ unlisted companies, SMEs, FDI enterprises, public/ private companies...)
- Find out and discuss on key concepts of business intelligence in that enterprise. For example:
 - Departments of enterprise?
 - Data sources being pertinent to those departments?
 - Discuss on how those sources of data give impacts on making decisions (or implications) to improve firm performance?
- Collect a piece of data on specific aspect (e.g. sales, human resources, production, finance...) of that enterprise, making visualization, and give some comments.
- Package for submission includes:
 - A report (MS Word file)
 - Data (MS Excel file)
 - Visualization (Power BI file)
- You could see several attached examples for reference.



- Identifying stakeholders, goals, and requirements
- Procuring the required resources
- Discovering the required data sources
- Designing a data model
- Planning reports and dashboards



- Business intelligence projects are most often driven by the business as opposed to IT. This means that one of the business domains, such as sales, marketing, manufacturing/ production, supply chain/operations, research and development, human resources, or accounting/finance, is attempting to answer specific questions about the business or better understand how to make the business more efficient, effective, and profitable. Therefore, it is imperative that any business intelligence project starts with identifying the specific **goals** or objectives of all interested parties or **stakeholders** that are championing the business intelligence project as well as any specific **requirements** in terms of data security, granularity of the data, the amount of historical data, and the availability of data.

Stakeholders

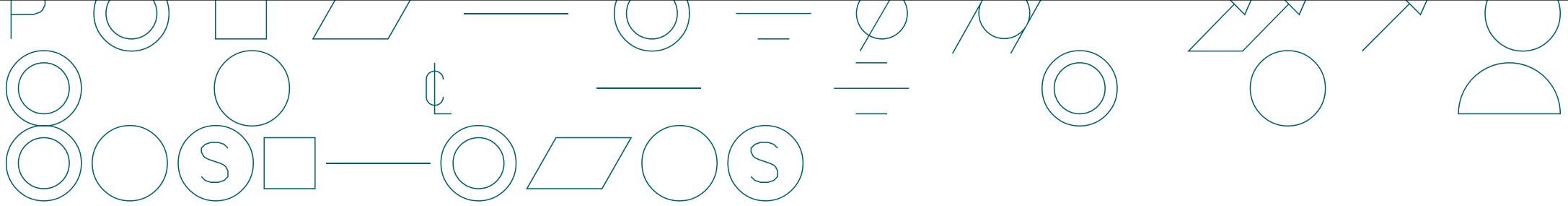
- It is not uncommon to have multiple stakeholders for a given business intelligence project, and even stakeholders that span multiple business domains. For example, sales and marketing often share similar interests and desire answers to similar questions. Start by identifying the business domains as stakeholders for the business intelligence project and then identify the specific individuals within those business domains that can provide the goals and requirements for the project.
- In most corporate environments, it is often advantageous to identify a single individual as a special kind of stakeholder, a **project sponsor**. Project sponsors help secure the funding and/or prioritization of resources for the business intelligence project.



- The goal or goals of the business intelligence project should be clearly stated and written down in one or two easily understood sentences (minimal business or technical jargon). The goal or goals should be ever-present in the minds of those implementing the business intelligence project in order to guide the project toward success and avoid increasing the scope of the project unnecessarily or, even worse, failing to fulfill the intended goals.
- Goals help determine what data will be required in order to fulfill the purpose of the business intelligence project. For example, in the case of the sales goal stated previously, this helps identify that all corporate sales transactions, as well as online and reseller sales transactions, are required as well as the yearly sales budget. For the marketing scenario, this means that marketing campaign dates, click-through rates, and website data are required. Finally, for the production scenario, the goal statement clearly indicates that both production and inventory data will be necessary.

RO_C ——= COMOS

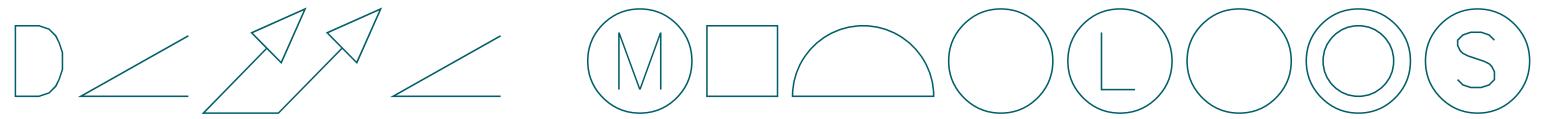
- Who will be accessing the reports and dashboards?
- Approximately how many individuals will need to access the reports and dashboards?
- How frequently will users be checking the reports and dashboards?
- Should some individuals or groups only see a subset of the data, reports, and dashboards?
- Are there any regulatory concerns regarding the data, such as PCI, HIPAA, or GDPR?
- What is the lowest level of data granularity required, such as individual transactions/ orders/records/lines or aggregated hourly, daily, weekly, monthly, or yearly?
- How will the data be analyzed, such as by date, customer, department, account, country, region, territory, city, or ZIP code?
- How much historical data is required – days, weeks, months, or years?
- How current must the data be, such as real-time, near real-time, daily, weekly, or monthly?
- What are the core **Key Performance Indicators (KPIs)** or business metrics required and what are their definitions?
- What calendar is used by the business?



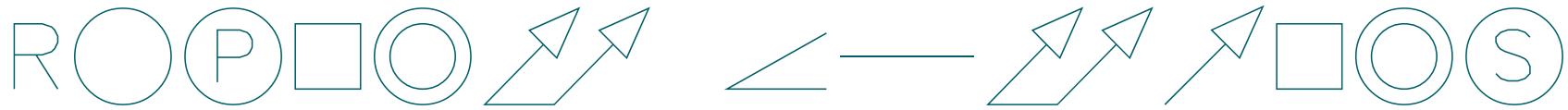
- Power BI projects require a number of different skill sets and areas of expertise. These skills sets can be broken down into the following roles that must be fulfilled in order to successfully execute a Power BI project:

- **Data modeler(s)**
- **Report author(s)**
- **Administrator(s)**

In smaller organizations, all of these roles may be fulfilled by the same individual, while in larger organizations, or for more complex projects, each role is fulfilled by one or more different individuals. In addition, these roles can be filled either by technical IT resources or business resources.



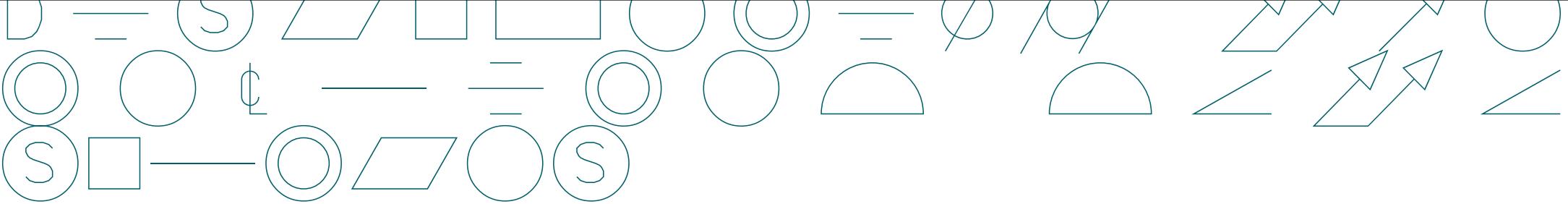
- Power BI projects require **connecting to source data systems via queries** that load data into structured data tables. These structured data tables and their relationships are called a **dataset** and it is the job of the data modeler to build these datasets. Data modelers are responsible for creating the queries that connect to the source data systems. In addition, data modelers transform or shape the data as necessary and define the relationships between the tables within the dataset. Data modelers are also responsible for defining the data types within the model (setting columns within data tables to be text or numbers), setting default summarizations such as sum, average, first, and last, and specifying data categories such as country, ZIP code, latitude, longitude, and web URL. Finally, data modelers often create any necessary calculations such as **Year-To-Date (YTD)** sales or gross margin percentage.



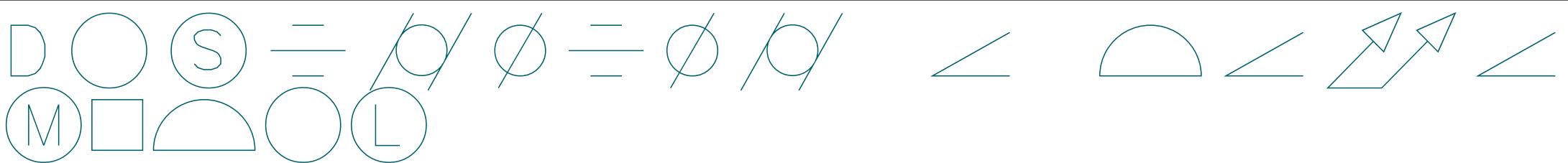
- Report authors are responsible for data analysis and **the creation of visuals that comprise reports and dashboards**. Generally, report authors serve as business analysts, gathering reporting requirements and understanding the KPIs and metrics desired by the business as well as how the business wishes to analyze and visualize the data to answer important business questions.
- Since the reports and dashboards created by report authors are the **most visible product of a Power BI project to the business**, it is important that report authors have solid skills in visualization and design best practices. Visualization choices, color choices, font choices, symmetry, and the layout of reports are key aspects of building a good user experience.



- Administrators are a special role within a Power BI tenant, the **Power BI administrator** role. Users assigned to this role can access the **admin portal** within the Power BI service. The admin portal allows administrators to monitor the usage of the Power BI service, as well as enforcing organizational governance policies through the **tenant settings**. For example, administrators can use the tenant settings to enable or disable certain functionality, such as the use of public embedded codes for reports, the use of R and Python visuals, the exporting of reports, and the creation of workspaces. Administrators are also responsible for setting up **enterprise data gateways** that facilitate data refresh for datasets, as well as creating and assigning permissions for data connections used by these data gateways. Finally, administrators can restrict the use of custom visuals and configure capacity settings used by Power BI Premium instances.

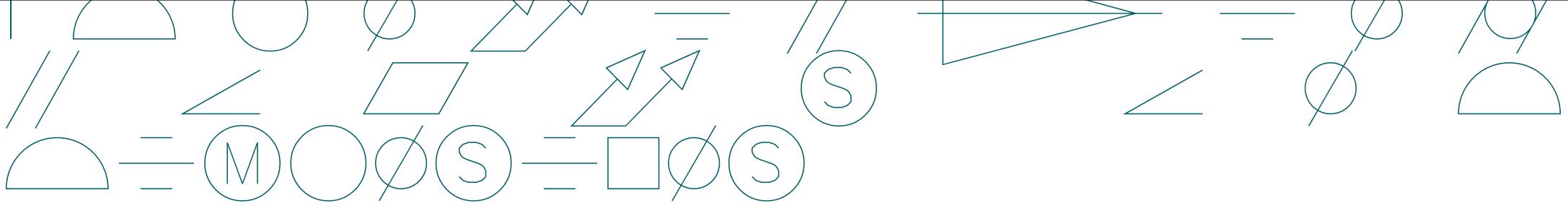


- Data modelers and report authors will need to collaborate with both IT and the business to discover the required source data repositories. These source data repositories may be transactional systems such as **Enterprise Resource Planning (ERP)** systems, **Customer Relationship Management (CRM)** systems, data warehouses, or even individual files maintained by the business, such as in Excel spreadsheets.
- An important aspect of this step is ensuring that **the required data sources are configured in such a way as to support access and/or data refresh within Power BI**. The Power BI reports generated will eventually be published to the cloud in the Power BI service or to an on-premises Power BI Report server. The service or server must be able to access these data sources.

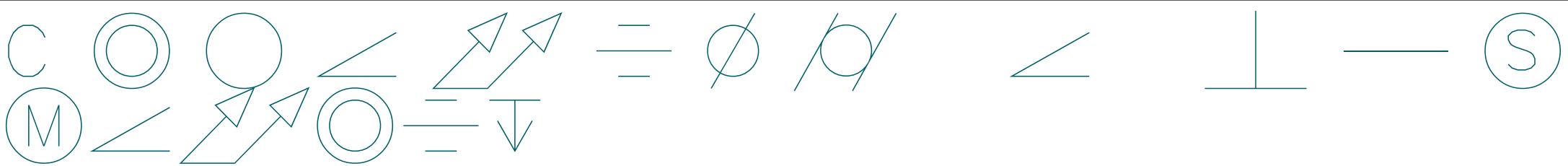


- A data model design includes the following steps:

- Identifying **facts** and **dimensions**
- Creating a **bus matrix**
- Determining a **dataset storage mode**

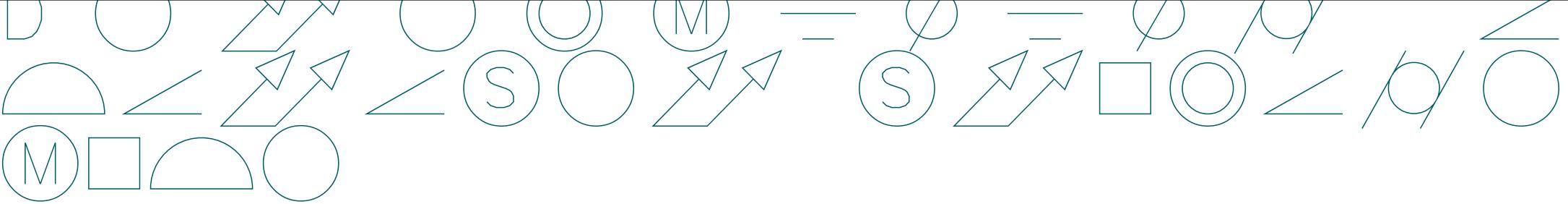


- Data modelers tend to think in terms of facts and dimensions. **Fact tables, or facts, are the tables that contain numeric information about particular business processes such as sales, marketing, production, and inventory.** Each row in a fact table represents a particular event associated with that business process, such as a change in inventory or a sale to a customer. Columns in the fact table include such things as the amount of inventory change, the number of units sold, and the unit sale price.
- **Dimension tables, or dimensions, contain information about people, places, or things involved in the business process.** As opposed to storing event information, dimensions store detailed information about people, places, and things. For example, a store dimension table would contain a row for every store in the organization and would include columns such as latitude, longitude, address, country, the date the store opened, and other detailed information that describes each store. Similarly, a product dimension table would include a row for every product sold and might include such columns as color, size, and category.

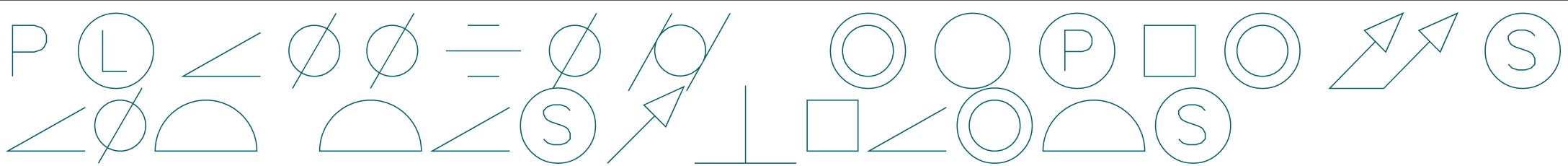


- The fact tables representing the business processes are on the left and columns describing the grain and important KPIs/measures are included as additional information. The dimension tables describing the people, places, and things associated with the facts are on the right. The Xs indicate which dimensions are associated with the different facts.

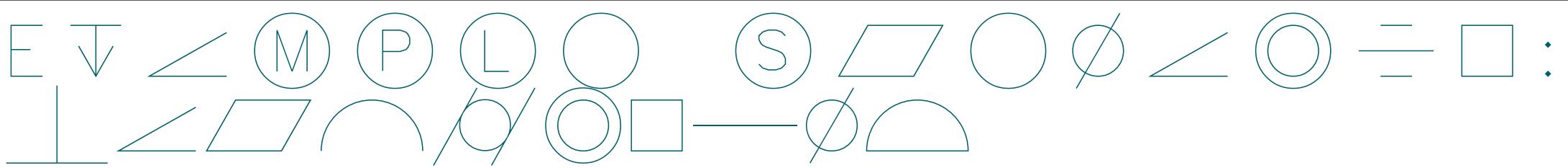
Facts	Grain	KPIs/Measures	Dimensions			
			Date	Product	Customer	Territory
Sales	Sales Order Line	# of Units, Unit Sale Price, Total Sale Price	X	X	X	X
Budget	Monthly	Budget Amount	X			X



- Deciding on dataset storage is an important design decision that should be addressed early in the planning process. Power BI supports the following three dataset storage modes:
 - **Import:** Power BI's default data storage mode is Import. This means that all of the data from source data systems is ingested into a local dataset to the Power BI Desktop file. Import mode allows the full functionality of all DAX functions and supports the extension of the dataset with DAX calculated columns as well as fast performance for DAX measures.
 - **DirectQuery:** DirectQuery mode datasets do not ingest any data into the local dataset. This avoids the dataset size limitations and refresh requirements associated with Import mode datasets.
 - **Composite:** Composite models allow you to mix DirectQuery and Import sources, or even multiple DirectQuery sources.



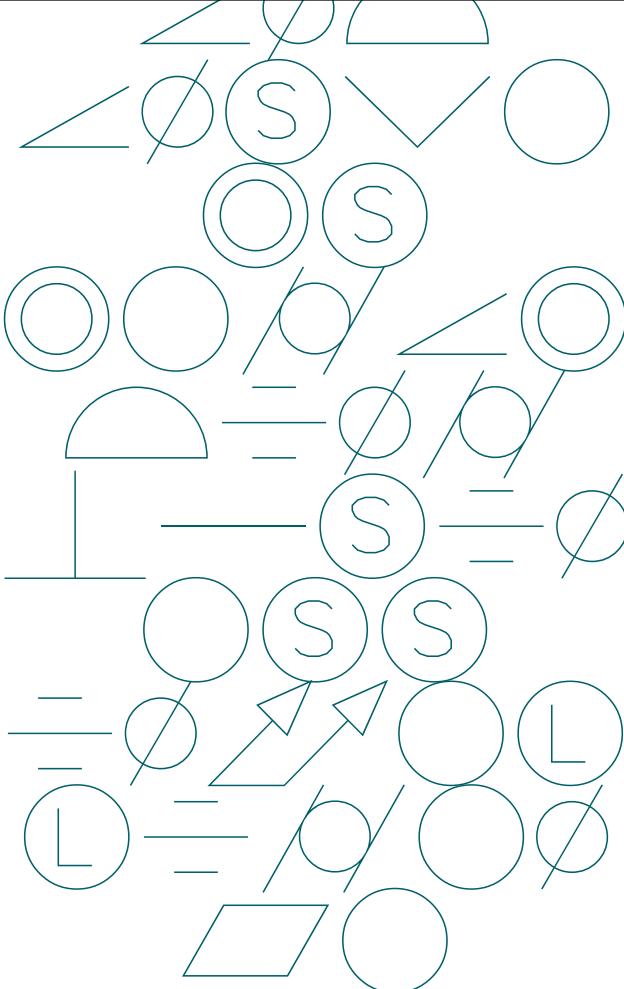
- Power BI lends itself to an agile, iterative development process where feedback from stakeholders is collected on a regular basis and used to inform and adjust the development process. This is especially true when authoring reports and dashboards. That said, it is still useful to include some upfront design work for reports. This often takes the form of one or more whiteboarding sessions with key stakeholders where different layouts for reports are discussed and mock-ups or wireframes of report pages are created. The goal of these design sessions is to understand the key KPIs and metrics the business wants to be displayed on reports, understand whether there will be multiple report pages, and discuss potential interactivity between visual elements and features in the report, such as the use of slicers, drillthrough, and drilldown.



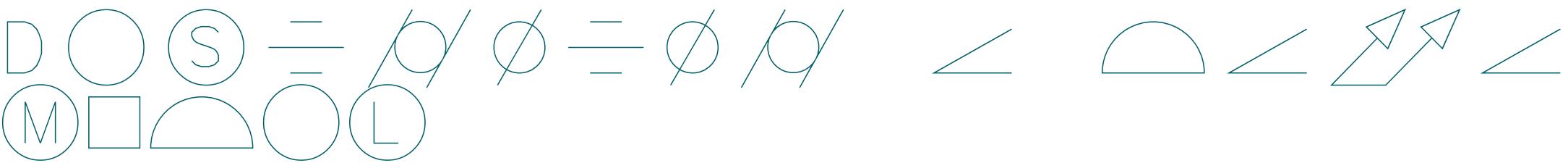
A firm recently conducted a yearly employee survey that highlighted a strong employee demand for increased flexibility regarding **Personal Time Off (PTO)**. As a result, human resources have decided to implement unlimited PTO. While time off still needs to be approved, employees no longer have a set amount of days of PTO per year. This is an enormous benefit to employees but, in order to stay profitable, it is now imperative that the organization is able to closely track and report on utilization so that managers can make informed business decisions around requests for time off.



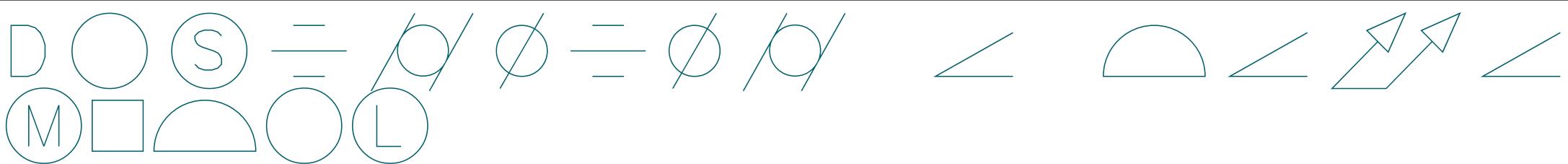
Report and analyze % utilization by employee, branch, and division across any date range, including the ability to identify trends and forecasts. The ability to compare revenue against budgets is desirable but not an absolute requirement at this time.



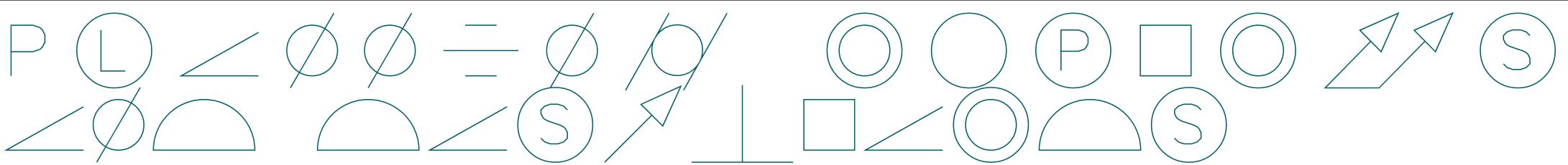
Question	Business Requirement
Who will be accessing the reports and dashboards?	Partners, executive management, branch managers, and division managers.
Approximately how many individuals will need to access the reports and dashboards?	10-12.
How frequently will users be checking the reports and dashboards?	Daily.
Should some individuals or groups only see a subset of the data, reports, and dashboards?	Yes, branch and division managers should only see the branches and divisions they are responsible for.
Are there any regulatory concerns regarding the data, such as PCI, HIPAA, or GDPR?	No.
What is the lowest level of data granularity required, such as individual transactions/orders/records/lines or aggregated hourly, daily, weekly, monthly, or yearly?	Daily for hours and % utilization. Budget data is by month.
How will the data be analyzed, such as by date, customer, department, account, country, region, territory, city, ... or ZIP code?	Date, division, branch, employee, project.
How much historical data is required– days, weeks, months, or years?	1 year.
How current must the data be, such as real-time, near real-time, daily, weekly, or monthly?	Weekly.
What are the core KPIs or business metrics required and what are their definitions?	% utilization. Utilization for sub-contractors and hourly employees is always 100%. For salaried employees, % utilization is defined as the sum of billable hours divided by the sum of potential billable hours. Each day is categorized as either a working day or a non-working day (holidays and weekends). A working day is assumed to have the potential for 8 billable hours.
What calendar is used by the business?	Standard calendar year.



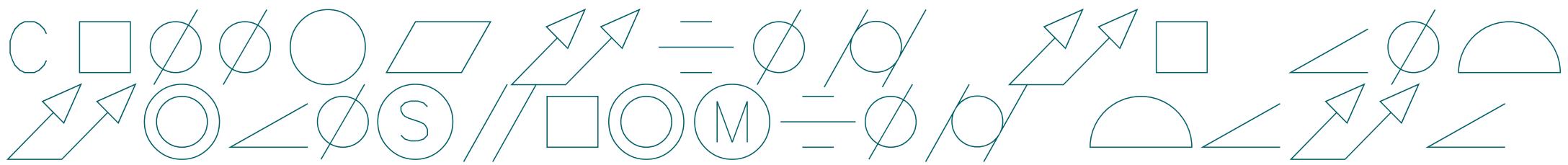
- **Hours:** Contains reported hours for every employee by day
 - **Budget:** Contains a forecasted budget by month for each division and branch
 - Dimensions for her model:
 - Date
 - Branch
 - Division
 - Employee
 - Project



Facts	Grain	KPIs/Measures	Dimensions				
			Date	Branch	Division	Employee	Project
Hours	Daily	Billable hours, % Utilization	X	X	X	X	X
Budget	Monthly	Budget Amount	X	X	X		



- All groups agree that report pages should be kept concise and clean with no more than four or five visuals on a page. The stakeholder groups also agree that there should be simple filtering of the reports by date range, branch, division, and employee type. A person suggests that the reports also include the ability for users to dynamically set the target utilization as different branches and divisions have slightly different utilization targets.
- Partners and executive management desire an executive summary page that includes the % utilization for the entire calendar year as well as a visual that displays the % utilization trend by month. Finally, partners and executive management also desire that the % utilization be displayed across branches and divisions.
- Division managers want similar information but displayed somewhat differently. Division managers are also interested in the % utilization for the year and % utilization broken down by branch. However, division managers also want visuals that display the total hours and % utilization by project code as well as employee type. Finally, division managers desire the ability to quickly see the total hours broken down by categories such as billable time, PTO, project non-billable time, bench time, and sales support.
- Branch managers desire extremely similar information to division managers, with the main difference being that branch managers want to see hours and % utilization by employee rather than by project code.
- Finally, all stakeholders feel that it is important to be able to drill down into additional detail pages that display hours and utilization per employee and per project code in order to analyze and determine the cause of utilization issues.



- Getting data
 - Transforming data
 - Merging, copying, and appending queries
 - Verifying and loading data

Adventure Works - Power Query Editor

File Home Transform Add Column View Tools Help

Close & Apply New Source Recent Enter Data Data source settings Manage Parameters Refresh Preview Manage

Properties Advanced Editor Manage

1

Manage Columns Reduce Rows Sort Split Column Group By Data Type: Whole Number Use First Row as Headers Replace Values

New Query Data Sources Parameters Query Transform

Text Analytics Vision Azure Machine Learning AI Transforms

Queries [6] **2**

FactResellerSales DimDate Resellers Employees SalesTerritories DimProduct

Table icon **4**

123 ResellerKey 123 GeographyKey

ResellerKey	GeographyKey	Count
1	1	637
2	2	635
3	3	584
4	4	572
5	5	322
6	6	303
7	7	599
8	8	409
9	9	568
10	10	44
11	11	96
12	12	96
13	13	211
14	14	167
15	15	6
16	16	247
17		

5

- Copy
- Remove
- Remove Other Columns
- Duplicate Column
- Add Column From Examples...
- Remove Duplicates
- Remove Errors
- Change Type
- Transform
- Replace Values... **6**
- Replace Errors...
- Group By...
- Fill
- Unpivot Columns
- Unpivot Other Columns
- Unpivot Only Selected Columns
- Rename...
- Move
- Drill Down
- Add as New Query

3

Query Settings

PROPERTIES

Name: Resellers

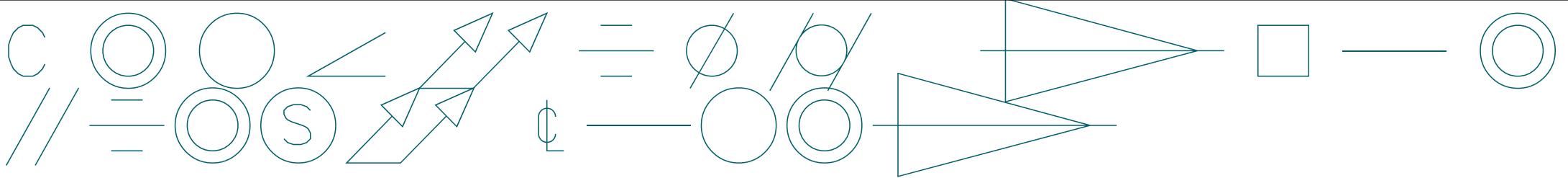
All Properties

APPLIED STEPS

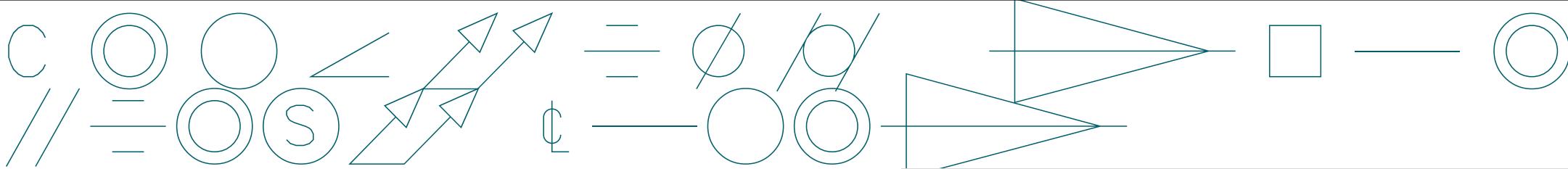
Source, Navigation, Promoted Headers, **Changed Type**

PREVIEW DOWNLOADED AT 9:08 AM

Source: Lachev & Price (2018)



- To determine whether or not **Personal Time Off (PTO)** should be approved, it is important to understand where the company is concerning budgets and forecasts. Whether or not the company, department, and/or location is on target in terms of its budget can be an important consideration when approving or denying time off.
- We will import data into our data model using a **query**. A query is simply a series of recorded steps for connecting to and transforming data.



- **All:** This lists all of the available connectors.
- **File:** File connectors, including Excel, text/CSV, XML, JSON, folder, PDF, and SharePoint folders.
- **Database:** The **Database** section lists sources such as SQL Server, Access, IBM DB2, IBM Informix, IBM Netezza, MySQL, PostgreSQL, Sybase, Teradata, SAP, Impala, Google BigQuery, Vertica, Snowflake, Essbase, and AtScale.
- **Power Platform:** Power Platform includes Power BI datasets and dataflows, as well as Dataverse.
- **Azure:** Azure lists many different services, such as Azure SQL Database, Azure Synapse Analytics, Azure Analysis Services, Azure Blob Storage, Azure Table Storage, Azure Cosmos DB, Azure Data Lake Storage, Azure HDInsights (HDFS), Azure HDInsights Spark, Azure Data Explorer (Kusto), Azure Databricks, and Azure Cost Management.
- **Online Services:** There is a substantial collection of online services, including Microsoft technologies such as SharePoint Online, Exchange Online, 365, Common Data Service, DevOps, and GitHub, as well as third parties such as Salesforce, Google, Adobe, QuickBooks, Smartsheet, Twilio, Zendesk, and many others.
- **Other:** other connectors include Web, OData, Spark, Hadoop (HDFS), ODBC, R, Python, and OLE DB.

Get Data

All

Icon	Connector Type
	Excel Workbook
	Text/CSV
	XML
	JSON
	Folder
	PDF
	Parquet
	SharePoint folder
	SQL Server database
	Access database
	SQL Server Analysis Services database
	Oracle database
	IBM Db2 database
	IBM Informix database (Beta)
	IBM Netezza
	MySQL database

Navigator

🔍
Display Options 📄
📁 Budgets and Forecasts.xlsx [1]
✔️ 📈 Budgets and Forecasts

1

Budgets and Forecasts

Location	Dept	Jan	Feb	Mar	Apr
Cleveland	1001	31855.1	33985.1	38741.21	
Cleveland	2001	14917.86	15200.85	19323.06	
Cleveland	3001	1787.64	1643.23	1907.65	
Charlotte	1001	6811.05	6859.54	9031.41	
Charlotte	2001	5753.77	6300.64	6142.28	
Nashville	1001	20926.68	19057.02	20915.76	
Nashville	2001	1181.7	1193.06	1266.72	
Nashville	3001	6333.42	5245.94	7358.03	
	null	null	null	null	null
Cleveland	1001	31989.38	31474.88	37386.81	
Cleveland	2001	13787.87	15212.4	19524.62	
Cleveland	3001	1946.36	1965.63	2012.09	
Charlotte	1001	6674.41	6881.78	6965.28	
Charlotte	2001	3894.15	3657.98	4275.02	
Nashville	1001	21822.36	19592.53	21140.26	
Nashville	2001	1054.44	863.36	1529.46	
Nashville	3001	6036.78	5409	4964.99	
	null	null	null	null	null
Forecast updated 4/12/2019		null	null	null	null

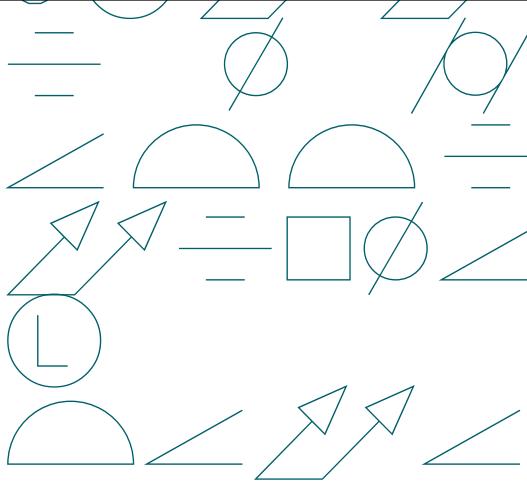
2

◀ ▶

Load

Transform Data

Cancel



Navigator

Display Options 🔍

📁 People and Tasks.xlsx [2]
 People
 Tasks

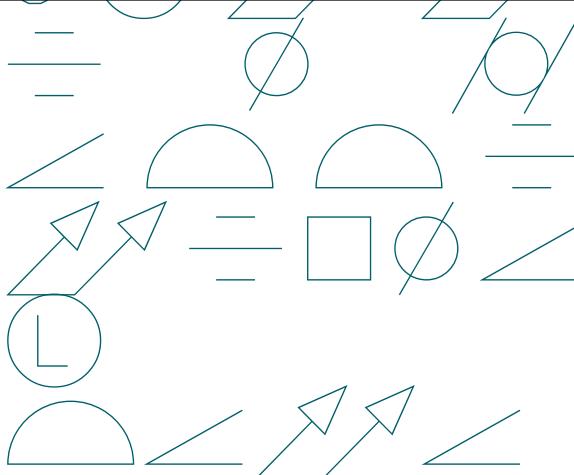
1

Tasks

Column1	Column2
TaskID	Category
PTO	PTO
INTERNAL	Int Admin
1001TM	Billable
1001TB	Billable
2001TM	Billable
3001FX	Billable
MEETING	Other
1001FX	Billable
SALESUP	Sales Support
TRAINING	Training
3001TM	Billable
2001FX	Billable
1001NB	PNB
BENCH	Bench
SICK	PTO
MARKETING	Other
8001TM	Billable
SALEPUR	Sales Pursuit
2001NB	PNB
OFFICE	Bench
3001NB	PNB
1001OT	Billable
TRAVEL	Other

2

Load Transform Data Cancel



Navigator

Display Options 🔍

- Hours.xlsx [3]
 - February
 - January
 - March

1

January

EmployeeID	Date	Hours	HourlyCost	HourlyRate	TaskID
GBRANCH	1/1/2019	8	136.972	0	PTO
GBRANCH	1/2/2019	8	136.972	0	INTERNAL
GBRANCH	1/3/2019	8	136.972	0	INTERNAL
GBRANCH	1/4/2019	8	136.972	0	INTERNAL
BREESE	1/2/2019	7	104.368	125	1001TM
BREESE	1/2/2019	3	104.368	75.5	1001TB
BREESE	1/3/2019	3	104.368	75.5	1001TB
BREESE	1/3/2019	8	104.368	125	1001TM
BREESE	1/4/2019	6	104.368	125	1001TM
FMAYNARD	1/2/2019	8	74.283	130	2001TM
FMAYNARD	1/3/2019	8	74.283	130	2001TM
FMAYNARD	1/4/2019	8	74.283	130	2001TM
FMAYNARD	1/1/2019	8	74.283	0	PTO
ACERVANTES	1/2/2019	8	67.067	105	1001TM
ACERVANTES	1/3/2019	8	67.067	105	1001TM
ACERVANTES	1/4/2019	8	67.067	105	1001TM
ACERVANTES	1/1/2019	8	67.067	0	PTO
JFRAZIER	1/2/2019	1.5	122.694	175	1001TM
JFRAZIER	1/4/2019	0.75	122.694	175	1001TM
EMORSE	1/2/2019	8	75.229	100	2001TM
EMORSE	1/3/2019	8	75.229	100	2001TM
EMORSE	1/4/2019	8	75.229	100	2001TM
EMORSE	1/1/2019	8	75.229	0	PTO

< >

2



- Power Query Editor can be launched from the Home tab by choosing Transform data in the Queries section of the Ribbon. Once launched, the following screen will be displayed:

The screenshot shows the Microsoft Power Query Editor window with several UI elements highlighted:

- Header:** The top navigation bar with tabs like File, Home, Transform, Add-Column, View, Tools, and Help.
- Ribbon:** The main menu bar above the ribbon area.
- Formula Bar:** The formula bar at the top of the main workspace.
- Queries pane:** A sidebar on the left showing a list of queries: Budgets and Forecasts, People, Tasks, and January.
- Date Canvas:** A small canvas at the bottom left.
- Footer:** The footer at the bottom of the window.
- Query Settings pane:** A pane on the right containing sections for Properties (Name: Budgets and Forecasts, All Properties) and Applied Steps (Source, Navigation, Promoted Head..., P: Changed Type).

The main workspace displays a table with columns: Location, Dept, Jan, and Feb. The data includes rows for Cleveland, Charlotte, and Nashville, along with a summary row for "Forecast updated 4/13/2019".

	Location	Dept	Jan	Feb
1	Cleveland		1001	114803.1
2	Cleveland		2001	14917.00
3	Cleveland		3001	1787.00
4	Charlotte		1001	6812.00
5	Charlotte		2001	1753.00
6	Nashville		1001	20946.68
7	Nashville		2001	11817.00
8	Nashville		3001	8333.40
9		null	null	null
10	Cleveland		1001	114803.00
11	Cleveland		2001	14917.00
12	Cleveland		3001	1787.00
13	Charlotte		1001	6812.00
14	Charlotte		2001	1753.00
15	Nashville		1001	21822.00
16	Nashville		2001	1054.00
17	Nashville		3001	6016.70
18		null	null	null
19	Forecast updated 4/13/2019			



- The **Formula bar** allows the user to view, enter, and modify the Power Query (**M**) code. The Power Query formula language, commonly called **M**, is a functional programming language comprised of functions, operators, and values. **M** is the underlying data connection and transformation technology for Microsoft Power Automate, PowerApps, Power BI Desktop, and Power Query in Excel.
- **M** is the language behind queries in Power BI. As you are building a query in **Power Query Editor**, behind the scenes, this is building an **M** script that executes to connect to, transform, and import your data. In reality, each of the applied steps in a query is a line of Power Query **M** language code. You do not need to worry about that just yet, but we will explore this in the *Merging, copying, and appending queries* section.

Practice 2 - Power Query Editor

File Home Transform Add Column View Tools Help

Close & Apply New Source Sources Enter Data Data source settings Manage Parameters Refresh Advanced Editor Choose Columns Remove Rows Keep Rows Reduce Rows Sort Split Column Group By Data Type: Text ▾ Merge Queries ▾ Text Analytics Close New Query Data Sources Parameters Preview Manage Query Manage Columns Manage Rows Use First Row as Headers ▾ Append Queries ▾ Vision Combine Files Combine AI Insights

Queries [4]

- Budgets and Forecasts
- Location
- Dept
- Jan
- Feb
- Mar
- Apr
- May

	Location	Dept	Jan	Feb	Mar	Apr	May
1	Cleveland	1001	31855.1	33985.1	38741.21	37042.63	
2	Cleveland	2001	14917.86	15200.85	19323.06	17673.31	
3	Cleveland	3001	1787.64	1643.23	1907.65	1756.22	
4	Charlotte	1001	6811.05	6859.54	9031.41	9292.51	
5	Charlotte	2001	5753.77	6300.64	6142.28	4986.36	
6	Nashville	1001	20926.68	19057.02	20915.76	21375.9	
7	Nashville	2001	1181.7	1193.06	1266.72	1547.7	
8	Nashville	3001	6333.42	5245.94	7358.03	7053.03	
9		null	null	null	null	null	
10	Cleveland	1001	31989.38	31474.88	37386.81	34260.07	
11	Cleveland	2001	13787.87	15212.4	19524.62	18215.71	
12	Cleveland	3001	1946.36	1965.63	2012.09	1541.85	
13	Charlotte	1001	6674.41	6881.78	6965.28	8891.79	
14	Charlotte	2001	3894.15	3657.98	4275.02	3736.48	
15	Nashville	1001	21822.36	19592.53	21140.26	27255.73	
16	Nashville	2001	1054.44	863.36	1529.46	2472.6	
17	Nashville	3001	6036.78	5409	4964.99	6308.77	
18		null	null	null	null	null	
19	Forecast updated 4/12/2019		null	null	null	null	

Query Settings

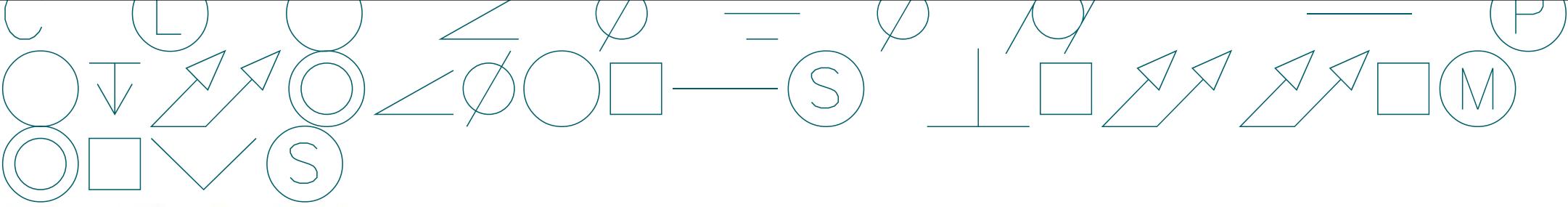
PROPERTIES

- Name: Budgets and Forecasts
- All Properties

APPLIED STEPS

- Source
- Navigation
- Promoted Headers
- Changed Type

15 COLUMNS, 19 ROWS Column profiling based on top 1000 rows PREVIEW DOWNLOADED AT 11:50 AM



A screenshot of Microsoft Power BI Data Editor showing a query editor and a transformation dialog.

File **Home** **Transform** **Add Column** **View** **Tools** **Help**

Close & Apply **New Source** **Recent Sources** **Enter Data** **Data source settings** **Manage Parameters** **Refresh Preview** **Advanced Editor** **Properties**

Choose Columns **Remove Columns** **Keep Rows** **Remove Rows** **Group By** **Split Column** **Replace Values** **Data Type: Text** **Use First Row as Headers**

Merge Queries **Append Queries** **Combine Files** **Combine** **Text Analytics** **Vision** **Azure Machine Learning** **AI Insights**

Queries [4]

Budgets and Forecasts

	1.2 Aug	1.2 Sep	1.2 Oct	1.2 Nov	1.2 Dec	AB_C Type
1	37041.17	36447.85	33505.11		34401	29134.96 Budget
2	15309.93	15228.32	13310.64		13291.05	12497.06 null
3	1741.04	1770.44	1636.96	1738.68	1675.68	1581.95 null
4	10972.17	12536.3	10626.19	12869.34	9227.87	9915.11 null
5	6624.38	8666.95	6670.85	7724.79	6612.46	5955.72 null
6	21812.23	23585.46	19490.02	21419.87	17237.91	17260.82 null
7	1625.25	1738.8	1386	1712.13	1009.8	1386.27 null
8	6103.01	6290.47	5415.92	5499.96	4789.29	5498.02 null
9	null	null	null	null	null	null null
10	35922.42	43346.91	35503.83	42702.37	37891.3	32484.06 Forecast
11	16806.93	14397.1	11442.78	17230.25	15230.08	12041.96 null
12	1974.89	2003.62	2011.16	2248.73	1760.16	1569.51 null

Transform

= Table.TransformColumnTypes(#"Promoted Headers", {{"L", Int64.Type}, {"Jan", type any}, {"Feb", type number}, {"Dept", Int64.Type}, {"Jan", type any}, {"Feb", type number}, {"Type", type any}})

Query Settings

PROPERTIES

Name: Budgets and Forecasts

All Properties

APPLIED STEPS

- Source
- Navigation
- Promoted Headers
- Changed Type

Remove Bottom Rows

Specify how many rows to remove from the bottom.

Number of rows:

OK Cancel



- Note that all of the distinct values that appear in the column are listed, including **(null)**, **Charlotte**, **Cleveland**, and **Nashville**. As datasets become larger, you may see a **List may be incomplete** warning message. This occurs because Power Query Editor samples the first 1,000 rows of data. If you see this message, you can click on the **Load more** link to have Power Query Editor analyze all the rows of data.



Cross-tab data: represented as a table with rows and columns, in which values are divided into cells corresponding to different dimensions, **wide format**.

Column %	Q3. Age									
	18 to 24	25 to 29	30 to 34	35 to 39	40 to 44	45 to 49	50 to 54	55 to 64	65 or more	
Coca Cola	65	41	52	58	29	28	46	36	36	
Diet Coke	2	10	9	17	23	4	8	12	23	
Coke Zero	9	23	18	19	17	28	28	16	14	
Pepsi Light	0	3	0	0	3	4	3	6	9	
Pepsi Max	16	18	12	0	11	8	13	24	14	
Pepsi	7	5	9	6	17	28+	3	6	5	
NET Sugared	72	46	61	64	46	56	49	42	41	
NET Sugarless	28	54	39	36	54	44	51	58	59	
NET	100	100	100	100	100	100	100	100	100	

Tabular data: represented as a table or list with independent records, **long format**.

	ABC Column %	ABC Attribute	123 Value
1	Coca Cola	18 to 24	65
2	Coca Cola	25 to 29	41
3	Coca Cola	30 to 34	52
4	Coca Cola	35 to 39	58
5	Coca Cola	40 to 44	29
6	Coca Cola	45 to 49	28
7	Coca Cola	50 to 54	46
8	Coca Cola	55 to 64	36
9	Coca Cola	65 or more	36
10	Diet Coke	18 to 24	2
11	Diet Coke	25 to 29	10
12	Diet Coke	30 to 34	9
13	Diet Coke	35 to 39	17
14	Diet Coke	40 to 44	23
15	Diet Coke	45 to 49	4
16	Diet Coke	50 to 54	8
17	Diet Coke	55 to 64	12
18	Diet Coke	65 or more	23
19	Coke Zero	18 to 24	9
20	Coke Zero	25 to 29	23



Unpivot: Convert data from wide format to long format.

Pivot: Convert data from long format to wide format.

Attribute

Value

Store	2020	2021
127 Sai Gon	10	15
27 Ha Noi	30	35

Unpivot



Store	Year	Quantity
127 Sai Gon	2020	10
127 Sai Gon	2021	15
27 Ha Noi	2020	30
27 Ha Noi	2021	35

Pivot





Practice 2 - Power Query Editor

File Home Transform Add Column View Tools Help

Data Type: Text Unpivot Columns Move

Transpose Reverse Rows Detect Data Type Fill Pivot Column Convert to List Group By Count Rows Rename Split Column Format ABC Extract 123 Parse Statistics Standard Scientific Trigonometry Rounding Date Time Duration Run R script Run Python script Py Scripts

Queries [4]

Budgets and Forecasts

People

Tasks

January

= Table.RenameColumns(#"Unpivoted Columns",{{"Attribute", "Month"}, {"Value", "Value"}})

	Location	Dept	Type	Month	Value	
1	Cleveland		1001	Budget	Jan	31855.1
2	Cleveland		1001	Budget	Feb	33985.1
3	Cleveland		1001	Budget	Mar	38741.21
4	Cleveland		1001	Budget	Apr	37042.63
5	Cleveland		1001	Budget	May	36216.9
6	Cleveland		1001	Budget	Jun	36919.86
7	Cleveland		1001	Budget	Jul	37041.17
8	Cleveland		1001	Budget	Aug	36447.85
9	Cleveland		1001	Budget	Sep	33505.11
10	Cleveland		1001	Budget	Oct	37192.39
11	Cleveland		1001	Budget	Nov	34401
12	Cleveland		1001	Budget	Dec	29134.96
13	Cleveland		2001		Jan	14917.86
14	Cleveland		2001		Feb	15200.85
15	Cleveland		2001		Mar	19323.06
16	Cleveland		2001		Apr	17673.31
17	Cleveland		2001		May	17411.26
18	Cleveland		2001		Jun	18658.18
19	Cleveland		2001		Jul	15309.93
20	Cleveland		2001		Aug	15228.32
21	Cleveland		2001		Sep	13310.64
22	Cleveland		2001		Oct	13667.68
23	Cleveland		2001		Nov	13291.05
24	Cleveland		2001		Dec	12497.06
25	Cleveland		3001		Jan	1787.64
26	Cleveland		3001		Feb	1643.23
27	Cleveland		3001		Mar	1907.65
28	Cleveland		3001		Apr	1756.22

Query Settings

Properties

Name: Budgets and Forecasts

All Properties

Applied Steps

Source

Navigation

Promoted Headers

Changed Type

Removed Bottom Rows

Filtered Rows

Unpivoted Columns

Renamed Columns

5 COLUMNS, 192 ROWS Column profiling based on top 1000 rows

PREVIEW DOWNLOADED AT 11:50 AM

U(S)=∅/F=LL

Practice 2 - Power Query Editor

File Home Transform Add Column View Tools Help

Table Any Column Text Column Number Column Date & Time Column Scripts

Queries [4]

Budgets and Forecasts

People

Tasks

January

= Table.RenameColumns(#"Unpivoted Columns",{{"Attribute", "Month"}, {"Value", "Value"}, {"Dept", "Type"}, {"Location", "Month"}, {"Month", "Month"}})

	Location	Dept	Type	Month	Value
1	Cleveland		1001	Budget	31855.1
2	Cleveland		1001	Budget	33985.1
3	Cleveland		1001	Budget	38741.21
4	Cleveland		1001	Budget	37042.63
5	Cleveland		1001	Budget	36216.9
6	Cleveland		1001	Budget	36919.86
7	Cleveland		1001	Budget	37041.17
8	Cleveland		1001	Budget	36447.85
9	Cleveland		1001	Budget	33505.11
10	Cleveland		1001	Budget	37192.39
11	Cleveland		1001	Budget	34401
12	Cleveland		1001	Budget	29134.96
13	Cleveland		2001		14917.86
14	Cleveland		2001		15200.85
15	Cleveland		2001		19323.06
16	Cleveland		2001		17673.31
17	Cleveland		2001		17411.26
18	Cleveland		2001		18658.18
19	Cleveland		2001		15309.93
20	Cleveland		2001		15228.32
21	Cleveland		2001		13310.64
22	Cleveland		2001		13667.68
23	Cleveland		2001	null Nov	13291.05
24	Cleveland		2001	null Dec	12497.06
25	Cleveland		3001	null Jan	1787.64
26	Cleveland		3001	null Feb	1643.23
27	Cleveland		3001	null Mar	1907.65
28	Cleveland		3001	null Apr	1756.22

Query Settings

PROPERTIES

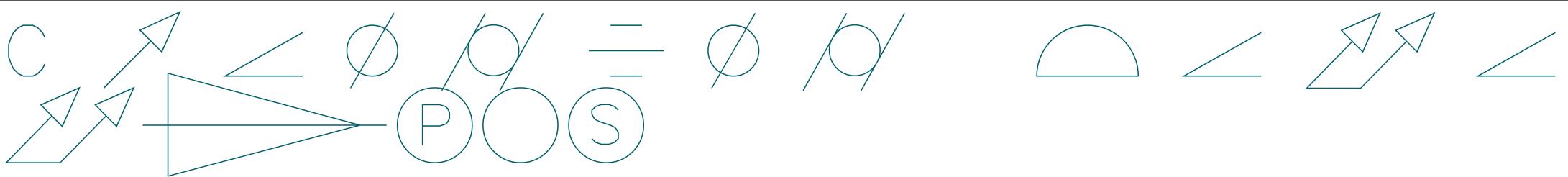
Name: Budgets and Forecasts

All Properties

APPLIED STEPS

Source, Navigation, Promoted Headers, Changed Type, Removed Bottom Rows, Filtered Rows, Unpivoted Columns, Renamed Columns

5 COLUMNS, 192 ROWS Column profiling based on top 1000 rows PREVIEW DOWNLOADED ON FRIDAY



Practice 2 - Power Query Editor

File Home Transform Add Column View Tools Help

Transposing Data Type: Any Replace Values Unpivot Columns
Reverse Rows Detect Data Type Fill Move
Group By Use First Row as Headers Count Rows Rename Pivot Column Convert to List

Split Column Format Merge Columns ABC Extract Statistics Standard Scientific Trigonometry
ABC 123 abc abc Parse Information Date Time Duration
Run R script Run Python script

Table Any Column Text Column Number Column Date & Time Column Scripts

Queries [4] x v fx = Table.FillDown(#"Renamed Columns", {"Type"})

Budgets and Forecasts

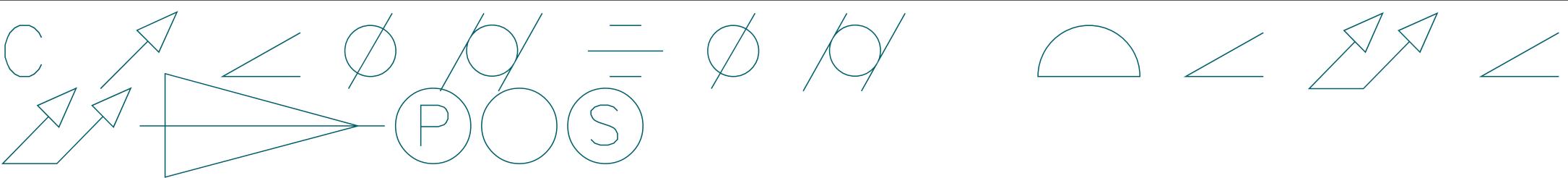
	ABC Location	123 Dept	ABC Type	ABC Month	Value	
1	Cleveland		1001	Budget	Jan	1.2 Decimal Number
2	Cleveland		1001	Budget	Feb	\$ Fixed decimal number
3	Cleveland		1001	Budget	Mar	123 Whole Number
4	Cleveland		1001	Budget	Apr	% Percentage
5	Cleveland		1001	Budget	May	>Date/Time
6	Cleveland		1001	Budget	Jun	Date
7	Cleveland		1001	Budget	Jul	Time
8	Cleveland		1001	Budget	Aug	Date/Time/Timezone
9	Cleveland		1001	Budget	Sep	Duration
10	Cleveland		1001	Budget	Oct	A B C Text
11	Cleveland		1001	Budget	Nov	True/False
12	Cleveland		1001	Budget	Dec	Binary
13	Cleveland		2001	Budget	Jan	Using Locale...
14	Cleveland		2001	Budget	Feb	19323.06
15	Cleveland		2001	Budget	Mar	17673.31
16	Cleveland		2001	Budget	Apr	17411.26
17	Cleveland		2001	Budget	May	18658.18
18	Cleveland		2001	Budget	Jun	15309.93
19	Cleveland		2001	Budget	Jul	15228.32
20	Cleveland		2001	Budget	Aug	13310.64
21	Cleveland		2001	Budget	Sep	13667.68
22	Cleveland		2001	Budget	Oct	13291.05
23	Cleveland		2001	Budget	Nov	12497.06
24	Cleveland		2001	Budget	Dec	1787.64
25	Cleveland		3001	Budget	Jan	1643.23
26	Cleveland		3001	Budget	Feb	1907.65
27	Cleveland		3001	Budget	Mar	1756.22
28	Cleveland		3001	Budget	Apr	

Query Settings

Properties Name Budgets and Forecasts All Properties

Applied Steps Source Navigation Promoted Headers Changed Type Removed Bottom Rows Filtered Rows Unpivoted Columns Renamed Columns Filled Down

5 COLUMNS, 192 ROWS Column profiling based on top 1000 rows PREVIEW DOWNLOADED ON FRIDAY



Practice 2 - Power Query Editor

File Home Transform Add Column View Tools Help

Queries [4]

Budgets and Forecasts

People

Tasks

January

Table Data Type: Whole Number 1 2 Replace Values 3 Unpivot Columns 4 Group By 5 Use First Row as Headers 6 Count Rows Transpose 7 Reverse Rows Detect Data Type 8 Fill 9 Move 10 Split Column 11 Pivot Column 12 Extract 13 ABC 14 abc 15 Parse 16 Statistics Standard Scientific 17 Trigonometry 18 10² 19 .00 Rounding 20 Date Time Duration 21 Run R script 22 Py 23 Information 24 Date & Time Column 25 Scripts

= Table.FillDown(#"Renamed Columns", {"Type"})

	Location	Dept	Type	Month	Value
1	Cleveland	1.2	Decimal Number	Jan	31855.1
2	Cleveland	\$	Fixed decimal number	Feb	33985.1
3	Cleveland	123	Whole Number	Mar	38741.21
4	Cleveland	%	Percentage	Apr	37042.63
5	Cleveland		Date/Time	May	36216.9
6	Cleveland		Date	Jun	36919.86
7	Cleveland	(⌚)	Time	Jul	37041.17
8	Cleveland	(🕒)	Date/Time/Timezone	Aug	36447.85
9	Cleveland	(🕒)	Duration	Sep	33505.11
10	Cleveland			Oct	37192.39
11	Cleveland			Nov	34401
12	Cleveland			Dec	29134.96
13	Cleveland			Jan	14917.86
14	Cleveland			Feb	15200.85
15	Cleveland			2001	Budget
16	Cleveland				Mar
17	Cleveland				19323.06
18	Cleveland				Apr
19	Cleveland				17673.31
20	Cleveland				May
21	Cleveland				17411.26
22	Cleveland				Jun
23	Cleveland				18658.18
24	Cleveland				Jul
25	Cleveland				15309.93
26	Cleveland				Aug
27	Cleveland				15228.32
28	Cleveland				Sep
					13310.64
					Oct
					13667.68
					Nov
					13291.05
					Dec
					12497.06
					3001
					Budget
					Jan
					1787.64
					Feb
					1643.23
					Mar
					1907.65
					Apr
					1756.22

Query Settings

PROPERTIES

Name: Budgets and Forecasts

All Properties

APPLIED STEPS

Source
Navigation
Promoted Headers
Changed Type
Removed Bottom Rows
Filtered Rows
Unpivoted Columns
Renamed Columns
Filled Down

5 COLUMNS, 192 ROWS Column profiling based on top 1000 rows

PREVIEW DOWNLOADED ON FRIDAY

Practice 2 - Power Query Editor

File Home Transform Add Column View Tools Help

Transpose Data Type: Text Replace Values Unpivot Columns Group Use First Row By as Headers Reverse Rows Detect Data Type Fill Move Split Column Format ABC Extract ABC 123 Parse Statistics Standard Scientific Trigonometry Rounding Information Date Time Duration Run R script Run Python script Scripts

Queries [4]

- Budgets and Forecasts
- People
- Tasks
- January

= Table.TransformColumnTypes(#"Promoted Headers",{{"ID", type text}, {"Name", type text}, {"Title", type text}, {"EmployeeType", type text}, {"TermDate", type date}, {"HireDate", type date}, {"Location", type text})

ID	Name	Title	EmployeeType	TermDate	HireDate	Location
KMCMAHON	McMahon, Karyn	CONSULTANT	CONSULTANT	1/1/1900	1/1/1900	Charlotte
CBRYANT	Bryant, Carolyn	CONSULTANT	SALARY	5/15/2015	1/1/1900	Charlotte
PLUCAS	Lucas, Pamela	SALES	ADMINISTRATION	1/1/1900	1/1/1900	Charlotte
ASHIELDS	Shields, Art	ADMIN	ADMINISTRATION	1/1/1900	1/1/1900	Charlotte
BFRANCO	Franco, Brenda	ADMIN	ADMINISTRATION	11/29/2018	1/1/1900	Charlotte
EBECKER	Becker, Eileen	CONSULTANT	SALARY	1/1/1900	1/1/1900	Charlotte
DVILLEGAS	Villegas, Desiree	CONSULTANT	CONSULTANT	11/18/2005	4/17/2000	Charlotte
MIMONTES	Montes, Megan	CONSULTANT	SALARY	3/8/2019	10/8/2001	Charlotte
TPARRISH	Parrish, Tamera	ADMIN	ADMINISTRATION	1/28/2011	6/9/2003	Charlotte
LGARRISON	Garrison, Lenny	ADMIN	ADMINISTRATION	1/1/9999	1/1/2004	Charlotte
TBENSON	Benson, Terra	ADMIN	ADMINISTRATION	8/18/2005	4/16/2004	Charlotte
CVALDEZ	Valdez, Carl	CONSULTANT	CONSULTANT	1/3/2006	8/2/2004	Charlotte
EKAISER	Kaiser, Evangeline	SALES	ADMINISTRATION	2/21/2007	10/25/2004	Charlotte
MVALENCIA	Valencia, Millicent	CONSULTANT	CONSULTANT	1/1/9999	10/28/2004	Charlotte
MPROCTOR	Proctor, Michele	CONSULTANT	SALARY	1/1/9999	1/4/2005	Charlotte
TCOOLEY	Cooley, Theron	CONSULTANT	SUB-CONTRACTOR	1/1/1900	2/28/2005	Charlotte
KBRUCE	Bruce, Katrina	SALES	ADMINISTRATION	1/1/9999	2/28/2005	Charlotte
HGOMEZ	Gomez, Hong	CONSULTANT	CONSULTANT	9/14/2007	4/1/2005	Charlotte
IBARRETT	Barrett, Isabelle	CONSULTANT	CONSULTANT	6/30/2006	4/1/2005	Charlotte
MSHAH	Shah, Miquel	CONSULTANT	SUB-CONTRACTOR	1/27/2006	4/18/2005	Charlotte
ESOSA	Sosa, Elma	CONSULTANT	CONSULTANT	3/31/2006	7/1/2005	Charlotte
BSTEVENS	Stevens, Beryl	ADMIN	ADMINISTRATION	6/22/2007	7/25/2005	Charlotte
IMCCANN	Mccann, Israel	ADMIN	ADMINISTRATION	1/1/9999	9/6/2005	Charlotte
KSERRANO	Serrano, Katina	CONSULTANT	SUB-CONTRACTOR	7/28/2006	10/20/2005	Charlotte
SSTEPHENSON	Stephenson, Shannon	CONSULTANT	CONSULTANT	2/10/2006	10/31/2005	Charlotte
EPOTTS	Potts, Elisa	CONSULTANT	ASSOCIATE	12/16/2005	12/19/2005	Charlotte
KAVILA	Avila, Kendrick	CONSULTANT	ASSOCIATE	11/17/2006	1/3/2006	Charlotte

7 COLUMNS, 999+ ROWS Column profiling based on top 1000 rows

Query Settings

PROPERTIES

- Name: People
- All Properties

APPLIED STEPS

- Source
- Navigation
- Promoted Headers
- Changed Type

PREVIEW DOWNLOADED ON FRIDAY

2 Practice 2 - Power Query Editor

File Home Transform Add Column View Tools Help

Group By Use First Row as Headers Table

Transpose Reverse Rows Count Rows Detect Data Type Fill Move Split Column Format Text Column Statistics Standard Scientific ABC Extract 10² Trigonometry ABC 123 Parse Number Column Date Time Duration Run R Run Python script Date & Time Column Scripts

Queries [4]

- Budgets and Forecasts
- People
- Tasks
- January

= Source{[Item="Tasks",Kind="Sheet"]}[Data]

ABC 123 Column1	ABC 123 Column2
1 TaskID	Category
2 PTO	PTO
3 INTERNAL	Int Admin
4 1001TM	Billable
5 1001TB	Billable
6 2001TM	Billable
7 3001FX	Billable
8 MEETING	Other
9 1001FX	Billable
10 SALESUP	Sales Support
11 TRAINING	Training
12 3001TM	Billable
13 2001FX	Billable
14 1001NB	PNB
15 BENCH	Bench
16 SICK	PTO
17 MARKETING	Other
18 8001TM	Billable
19 SALEPUR	Sales Pursuit
20 2001NB	PNB
21 OFFICE	Bench
22 3001NB	PNB
23 1001OT	Billable
24 TRAVEL	Other
25 BEREAVE	Other
26 2001TB	Billable
27 JURYDUTY	Other
28 VACNOPAY	PTO

Query Settings

PROPERTIES

- Name: Tasks
- All Properties

APPLIED STEPS 1

- Source
- Navigation

2 COLUMNS, 30 ROWS Column profiling based on top 1000 rows

PREVIEW DOWNLOADED ON FRIDAY

Practice 2 - Power Query Editor

File Home Transform Add Column View Tools Help

Transpose Reverse Rows Data Type: Any Replace Values Unpivot Columns Group Use First Row as Headers Count Rows Detect Data Type Fill Move Split Column Format Text Column Statistics Standard Scientific ABC Extract 10² Trigonometry ABC 123 Parse Number Column Date Time Duration Run R Run Python script Py

Any Column Pivot Column Convert to List Information Date & Time Column Scripts

Queries [4]

Budgets and Forecasts

People

Tasks

January

- Table.PromoteHeaders(Tasks_Sheet, [PromoteAllScalars=true])

	TaskID	Category
1	1.2	Decimal Number
2	\$	Fixed decimal number
3	123	Whole Number
4	%	Percentage
5	Date/Time	
6	Date	
7	Time	
8	Date/Time/Timezone	
9	Duration	
10	Text	
11	True/False	
12	Binary	
13	Using Locale...	
14	SICK	PTO
15	MARKETING	Other
16	8001TM	Billable
17	SALEPUR	Sales Pursuit
18	2001NB	PNB
19	OFFICE	Bench
20	3001NB	PNB
21	1001OT	Billable
22	TRAVEL	Other
23	BEREAVE	Other
24	2001TB	Billable
25	JURYDUTY	Other
26	VACNOPAY	PTO
27	QUALITY	Other

2 COLUMNS, 29 ROWS Column profiling based on top 1000 rows

Query Settings

PROPERTIES

Name: Tasks

All Properties

APPLIED STEPS

Source

Navigation

Promoted Headers

PREVIEW DOWNLOADED ON FRIDAY

Practice 2 - Power Query Editor

File Home Transform Add Column View Tools Help

Data Type: Decimal Number 1 Replace Values Unpivot Columns Transpose Reverse Rows Detect Data Type Fill Move Split Column Format ABC 123 Extract ABC Parse Statistics Standard Scientific Trigonometry 10² Rounding Date Time Duration Information Run R script Run Python script

Group By Use First Row as Headers Count Rows Rename Pivot Column Convert to List Table Any Column Text Column Number Column Date & Time Column Scripts

Queries [4]

Budgets and Forecasts

People

Tasks

January

EmployeeID Date Hours HourlyCost HourRate TaskID JobID

1 GBRANCH 1/1/2019 1.2 1.2 PTO 1001TECSOL

2 GBRANCH 1/2/2019 1.2 1.2 INTERNAL 1001TECSOL

3 GBRANCH 1/3/2019 1.2 1.2 INTERNAL 1001TECSOL

4 GBRANCH 1/4/2019 1.2 1.2 INTERNAL 1001TECSOL

5 BREESE 1/2/2019 1.2 1.2 125 1001TM CLEV003201

6 BREESE 1/2/2019 1.2 1.2 75.5 1001TB CLEV003201

7 BREESE 1/3/2019 1.2 1.2 75.5 1001TB CLEV003201

8 BREESE 1/3/2019 1.2 1.2 125 1001TM CLEV003201

9 BREESE 1/4/2019 1.2 1.2 125 1001TM CLEV003201

10 FMAYNARD 1/2/2019 1.2 1.2 130 2001TM CLEV003201

11 FMAYNARD 1/3/2019 1.2 1.2 130 2001TM CLEV003201

12 FMAYNARD 1/4/2019 1.2 1.2 130 2001TM CLEV003201

13 FMAYNARD 1/1/2019 1.2 1.2 0 PTO 2001ACCSOL

14 ACERVANTES 1/2/2019 1.2 1.2 105 1001TM CLEV002983

15 ACERVANTES 1/2/2019 1.2 1.2 105 1001TM CLEV002983

16 ACERVANTES 1.2 1.2 CLEV002983

17 ACERVANTES 1.2 1.2 1001TECSOL

18 JFRAZIER 1.2 1.2 CLEV003173

19 JFRAZIER 1.2 1.2 CLEV003173

20 EMORSE 1.2 1.2 CLEV002797

21 EMORSE 1.2 1.2 CLEV002797

22 EMORSE 1.2 1.2 CLEV002797

23 EMORSE 1.2 1.2 2001ACCSOL

24 LMOORE 1.2 1.2 CLEV003176

25 LMOORE 1.2 1.2 CLEV003176

26 LMOORE 1.2 1.2 CLEV003176

27 TOWEN 1.2 1.2 CLEV003092

28 <

Query Settings

PROPERTIES

Name: January

All Properties

APPLIED STEPS

Source

Navigation

Promoted Headers

Changed Type

Change Column Type

The selected column has an existing type conversion. Would you like to replace the existing conversion, or preserve the existing conversion and add the new conversion as a separate step?

Replace current Add new step Cancel

15 COLUMNS, 999+ ROWS Column profiling based on top 1000 rows PREVIEW DOWNLOADED AT 9:33 PM

МООДЫ = ПЛАНЫ

Result

Product ID	Order Date	Quantity	Name
2	01/01/2020	1	iPhone
7	02/01/2020	2	iPad
7	03/01/2020	10	iPad

Query 1

Product ID	Order Date	Quantity
2	01/01/2020	1
7	02/01/2020	2
7	03/01/2020	10

Product ID	Name
2	iPhone
7	iPad

Query 2

M O O D = Ø Ø Ⓜ — Ø Ø = Ø S

Practice 2 - Power Query Editor

File Home Transform Add Column View Tools Help

New Source Sources Enter Data Data source settings Manage Parameters Refresh Preview Advanced Editor Properties Choose Columns Remove Columns Keep Rows Remove Rows Sort Split Column Group By Data Type: Text Use First Row as Headers Append Queries Combine Files Text Analytics Vision Azure Machine Learning AI Insights

Queries [4]

Budgets and Forecasts

People

Tasks

January

Merge Queries

Query Settings

PROPERTIES

Name: January

All Properties

APPLIED STEPS

Source

Navigation

Promoted Headers

Changed Type

15 COLUMNS, 999+ ROWS Column profiling based on top 1000 rows

PREVIEW DOWNLOADED AT 9:33 PM

	EmployeeID	Date	12 Hours	\$ HourlyCost	\$ HourlyRate	TaskID	JobID
1	GBRANCH	1/1/2019	8	136.97	0.00	PTO	1001TECSOL
2	GBRANCH	1/2/2019	8	136.97	0.00	INTERNAL	1001TECSOL
3	GBRANCH	1/3/2019	8	136.97	0.00	INTERNAL	1001TECSOL
4	GBRANCH	1/4/2019	8	136.97	0.00	INTERNAL	1001TECSOL
5	BREESE	1/2/2019	7	104.37	125.00	1001TM	CLEVO03201
6	BREESE	1/2/2019	3	104.37	75.50	1001TB	CLEVO03201
7	BREESE	1/3/2019	3	104.37	75.50	1001TB	CLEVO03201
8	BREESE	1/3/2019	8	104.37	125.00	1001TM	CLEVO03201
9	BREESE	1/4/2019	6	104.37	125.00	1001TM	CLEVO03201
10	FMAYNARD	1/2/2019	8	74.28	130.00	2001TM	CLEVO03201
11	FMAYNARD	1/3/2019	8	74.28	130.00	2001TM	CLEVO03201
12	FMAYNARD	1/4/2019	8	74.28	130.00	2001TM	CLEVO03201
13	FMAYNARD	1/1/2019	8	74.28	0.00	PTO	2001ACCSOL
14	ACERVANTES	1/2/2019	8	67.07	105.00	1001TM	CLEVO02983
15	ACERVANTES	1/3/2019	8	67.07	105.00	1001TM	CLEVO02983
16	ACERVANTES	1/4/2019	8	67.07	105.00	1001TM	CLEVO02983
17	ACERVANTES	1/1/2019	8	67.07	0.00	PTO	1001TECSOL
18	JFRAZIER	1/2/2019	1.5	122.69	175.00	1001TM	CLEVO03173
19	JFRAZIER	1/4/2019	0.75	122.69	175.00	1001TM	CLEVO03173
20	EMORSE	1/2/2019	8	75.23	100.00	2001TM	CLEVO02797
21	EMORSE	1/3/2019	8	75.23	100.00	2001TM	CLEVO02797
22	EMORSE	1/4/2019	8	75.23	100.00	2001TM	CLEVO02797
23	EMORSE	1/1/2019	8	75.23	0.00	PTO	2001ACCSOL
24	LMOORE	1/2/2019	9	60.50	100.00	1001TM	CLEVO03176
25	LMOORE	1/3/2019	10	60.50	100.00	1001TM	CLEVO03176
26	LMOORE	1/4/2019	10	60.50	100.00	1001TM	CLEVO03176
27	TOWEN	1/2/2019	10	88.00	135.00	2001TM	CLEVO03092
28							

Merge

Select a table and matching columns to create a merged table.

January

EmployeeID	Date	Hours	HourlyCost	HourlyRate	TaskID	JobID	Division	PeriodStart
GBRANCH	1/1/2019	8	136.97	0.00	PTO	1001TECSOL	1001 Technology	12/31/2018
GBRANCH	1/2/2019	8	136.97	0.00	INTERNAL	1001TECSOL	1001 Technology	12/31/2018
GBRANCH	1/3/2019	8	136.97	0.00	INTERNAL	1001TECSOL	1001 Technology	12/31/2018
GBRANCH	1/4/2019	8	136.97	0.00	INTERNAL	1001TECSOL	1001 Technology	12/31/2018
GBRANCH	1/5/2019	7	136.97	100.00	1001TM	1001TECSOL	1001 Technology	12/31/2018

1

2



Tasks

TaskID	Category
PTO	PTO
INTERNAL	Int Admin
1001TM	Billable
1001TB	Billable
2001TM	Billable

2

Join Kind

Left Outer (all from first, matching from second)

Use fuzzy matching to perform the merge

> Fuzzy matching options

✓ The selection matches 10991 of 10991 rows from the first table.

OK

Cancel

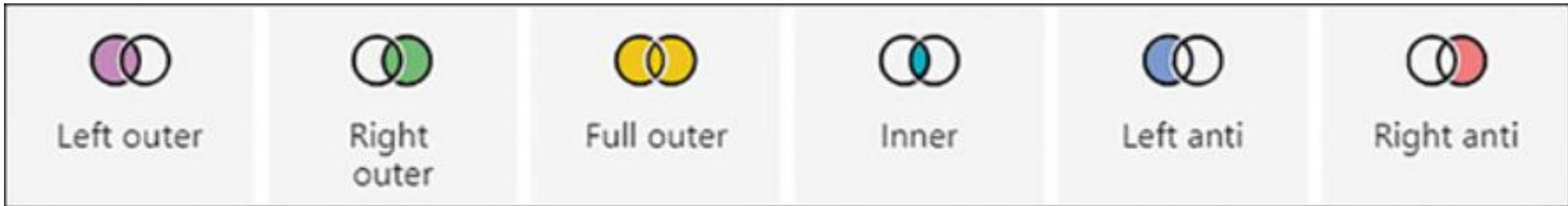
$$J \square = \emptyset \quad K = \emptyset$$

- A **fuzzy merge** allows similar but not identical items to be matched during a merge. Options include a **Similarity threshold**, which is optional. The **similarity threshold** is a number between **0.00** and **1.00**. A value of **0.00** causes all values to match, while a value of **1.00** causes only exact values to match. The default is a value of **0.80**. Additional options include the ability to **Ignore case**, as well as the ability to **Match by combining text parts**. For example, by ignoring case, mlcrOSoft could match Microsoft, and by combining text parts, Micro and soft could be combined to match Microsoft. When performing fuzzy merges, it is possible to have multiple values match. You can use the optional **Maximum number of matches** setting to control how many matching rows are returned for each input row. This is a number that can range from **1** to **2,147,483,647** (the default). Finally, there is an option to use a **Transformation table**. This allows you to specify a table of values with **From** and **To** columns that can be used during the merge process. For example, the merge table might contain a value in the **From** column for USA that maps to a **To** column of United States.

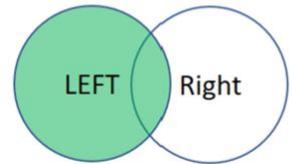
$$J \square = \emptyset \quad K = \emptyset$$

- **Similarity threshold** Can be valued from 0 to 1, where 0 will make all values match each other and 1 will allow exact matches only. The default is 0.8.
- **Ignore case** Lowercase and uppercase letters will be treated as the same.
- **Match by combining text parts** Power Query will try to combine separate words into one to find matches between keys.
- **Maximum number of matches** This option will limit the number of rows from the second table that are matched against the first table, and it can be useful if you expect multiple matches.
- **Transformation table** You can use a column with two columns—**From** and **To**—to map values during the matching process. For example, you can map *NZ* to *New Zealand*, and the two values will be considered the same for merge purposes.

$$J \square = \emptyset \quad K = \emptyset \cup$$

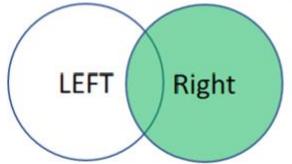


LEFT Outer



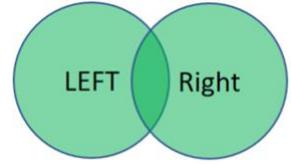
All rows from left and matching from right

RIGHT Outer



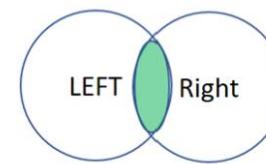
All rows from right and matching from left

Full Outer



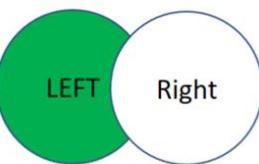
All rows from both: matching and not matching

Inner



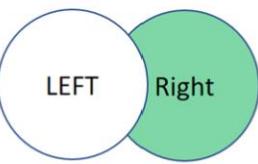
Only matching rows

Left Anti



Not matching rows from left

Right Anti



Not matching rows from right

EVP <∅ ⊥ = ∅ ∨ → ⊥ LOS

ks", JoinKind.LeftOuter)

A^B_C PayType Tasks

Search Columns to Expand A Z

Expand Aggregate

(Select All Columns)
 TaskID
 Category

Use original column name as prefix

OK Cancel

Query Settings X

PROPERTIES

Name
January

All Properties

APPLIED STEPS

Source Navigation Promoted Headers Changed Type Merged Queries

EV P ⊲ Ø ⊦ Ø Ø ↗ ↙ ⊥ LOS

- **Expand** You can select the columns from the joined table that you want to add to the current table. If there is more than one matching row in the joined table, the current table's rows will be duplicated after expansion.
- **Aggregate** This option aggregates rows and won't duplicate any rows in the current table. You can apply arithmetic and statistical functions to the columns of the joined table. For example, if it made business sense, you could take the average of **State Province Key** from the **State Province** table.

MOOBYO → OOS

- The new column:
 - Is of type Table
 - Has *Table* hyperlinks in each cell
 - Has a double-arrow button instead of a filter button on its header

JOINING TABLES

- **Expand** You can select the columns from the joined table that you want to add to the current table. If there is more than one matching row in the joined table, the current table's rows will be duplicated after expansion.
- **Aggregate** This option aggregates rows and won't duplicate any rows in the current table. You can apply arithmetic and statistical functions to the columns of the joined table. For example, if it made business sense, you could take the average of **State Province Key** from the **State Province** table.



- M is a case-sensitive language that is behind the Power Query engine.

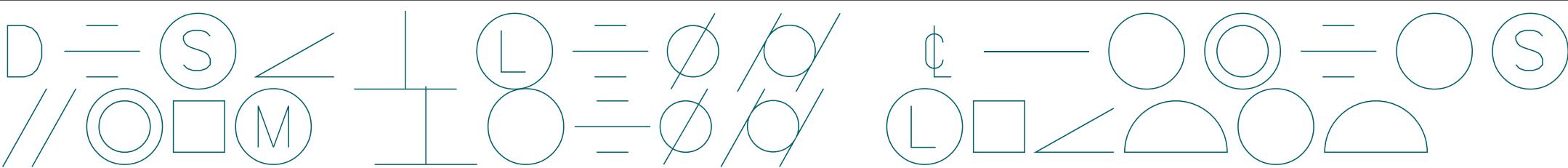
Advanced Editor

Date

```
let
    Source = Excel.Workbook(File.Contents("D:\OneDrive\1. Data\14. Tai lieu lop hoc\1. Data visualization and GIS\Textbook\9780136819684_exam.xlsx")),
    Date_Sheet = Source{[Item="Date",Kind="Sheet"]}[Data],
    #"Promoted Headers" = Table.PromoteHeaders(Date_Sheet, [PromoteAllScalars=true]),
    #"Changed Type" = Table.TransformColumnTypes(#"Promoted Headers",{{"Date", type date}, {"Month", type text}, {"Month Number", Int64.Type}})
in
    #"Changed Type"
```

No syntax errors have been detected.

Done Cancel



File Home Transform Add Column View Tools Help

Close & Apply New Source Recent Enter Data Data source settings Manage Parameters Refresh Preview Manage Choose Columns Remove Column Close New Query Data Sources Parameters Query Manage Column

Queries [4] < X ✓ fx = Table.PromoteHeaders(Tasks_Sheet, [Promo

Budgets and Forecasts

People

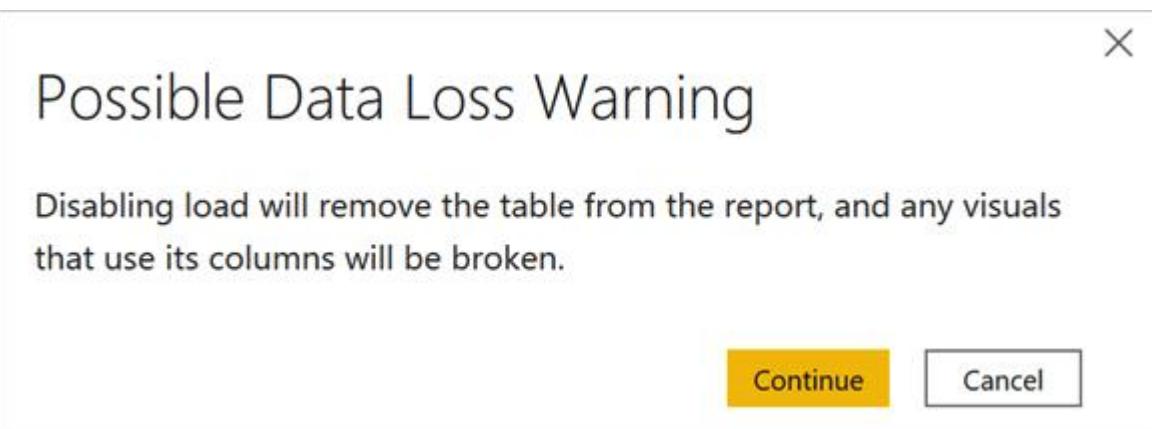
Tasks

January

ABC 123 TaskID ABC 123 Category

TaskID	Category
1	PTO
	PTO
	Int Admin
	Billable
	Billable
	Billable
	Billable
	Other
	Billable
	Sales Support
	Training
	Billable
	Billable
	PNB
	Bench
	PTO
	Other
	Billable
	Sales Pursuit
19	2001NB
20	OFFICE
21	3001NB

Copy Paste Delete Rename Enable load Include in report refresh Duplicate Reference Move To Group Move Up Move Down Create Function... Convert To Parameter Advanced Editor Properties...





Screenshot of the Microsoft Power Query Editor interface.

Queries [4]

- Budgets and Forecasts
- People
- Tasks
- January

The **January** query is selected and has a context menu open:

- Copy (highlighted with a red box)
- Paste
- Delete
- Rename
- Enable load
- Include in report refresh
- Duplicate
- Reference
- Move To Group
- Move Up
- Move Down
- Create Function...
- Convert To Parameter
- Advanced Editor
- Properties...

Queries [8]

= Table.NestedJoin(#"Changed Type", {"TaskID"}, {"Tasks (2)"}, {"Tasks", JoinKind.LeftOuter})

AB_C EmployeeID	Date	1.2 Hours	\$ HourlyCost	\$ HourlyRate	AB_C TaskID	AB_C JobID
1 GBRANCH	1/1/2019	8	136.97	0.00	PTO	1001TECSOL
2 GBRANCH	1/2/2019	8	136.97	0.00	INTERNAL	1001TECSOL
3 GBRANCH	1/3/2019	8	136.97	0.00	INTERNAL	1001TECSOL
4 GBRANCH	1/4/2019	8	136.97	0.00	INTERNAL	1001TECSOL
5 BREESE	1/2/2019	7	104.37	125.00	1001TM	CLEV003201
6 BREESE	1/2/2019	3	104.37	75.50	1001TB	CLEV003201
7 BREESE	1/3/2019	3	104.37	75.50	1001TB	CLEV003201
8 BREESE	1/3/2019	8	104.37	125.00	1001TM	CLEV003201
9 BREESE	1/4/2019	6	104.37	125.00	1001TM	CLEV003201
10 FMAYNARD	1/2/2019	8	74.28	130.00	2001TM	CLEV003201
11 FMAYNARD	1/3/2019	8	74.28	130.00	2001TM	CLEV003201
12 FMAYNARD	1/4/2019	8	74.28	130.00	2001TM	CLEV003201
13 FMAYNARD	1/1/2019	8	74.28	0.00	PTO	2001ACCSOL
14 ACERVANTES	1/2/2019	8	67.07	105.00	1001TM	CLEV002983
15 ACERVANTES	1/3/2019	8	67.07	105.00	1001TM	CLEV002983
16 ACERVANTES	1/4/2019	8	67.07	105.00	1001TM	CLEV002983
17 ACERVANTES	1/1/2019	8	67.07	0.00	PTO	1001TECSOL
18 JFRAZIER	1/2/2019	1.5	122.69	175.00	1001TM	CLEV003173
19 JFRAZIER	1/4/2019	0.75	122.69	175.00	1001TM	CLEV003173
20 EMORSE	1/2/2019	8	75.23	100.00	2001TM	CLEV002797
21 EMORSE	1/3/2019	8	75.23	100.00	2001TM	CLEV002797
22 EMORSE	1/4/2019	8	75.23	0.00	PTO	2001ACCSOL
23 EMORSE	1/1/2019	8	75.23	0.00	PTO	2001ACCSOL
24 LMOORE	1/2/2019	9	60.50	100.00	1001TM	CLEV003176
25 LMOORE	1/3/2019	10	60.50	100.00	1001TM	CLEV003176
26 LMOORE	1/4/2019	10	60.50	100.00	1001TM	CLEV003176
27 TOWEN	1/2/2019	10	88.00	135.00	2001TM	CLEV003092
28						

16 COLUMNS, 999+ ROWS Column profiling based on top 1000 rows

PREVIEW DOWNLOADED AT 11:14 PM

Query Settings

PROPERTIES

- Name: January (2)
- All Properties

APPLIED STEPS

- Source
- Navigation
- Promoted Headers
- Changed Type
- Merged Queries (highlighted with a red box)
- Expanded Tasks

Merge

Select a table and matching columns to create a merged table.

January (2)

EmployeeID	Date	Hours	HourlyCost	HourlyRate	TaskID	JobID	Division
GBRANCH	1/1/2019	8	136.97	0.00	PTO	1001TECSOL	1001 Technology
GBRANCH	1/2/2019	8	136.97	0.00	INTERNAL	1001TECSOL	1001 Technology
GBRANCH	1/3/2019	8	136.97	0.00	INTERNAL	1001TECSOL	1001 Technology
GBRANCH	1/4/2019	8	136.97	0.00	INTERNAL	1001TECSOL	1001 Technology

Tasks (2)

Budgets and Forecasts

January

January (2) (Current)

January (3)

People

Tasks

Tasks (2)

Tasks (3)

Join Kind

Left Outer (all from first, matching from second)

Use fuzzy matching to perform the merge

> Fuzzy matching options

Merge

Select a table and matching columns to create a merged table.

January (2)

EmployeeID	Date	Hours	HourlyCost	HourlyRate	TaskID	JobID	Division	PeriodStart
GBRANCH	1/1/2019	8	136.97	0.00	PTO	1001TECSOL	1001 Technology	12/31/2018
GBRANCH	1/2/2019	8	136.97	0.00	INTERNAL	1001TECSOL	1001 Technology	12/31/2018
GBRANCH	1/3/2019	8	136.97	0.00	INTERNAL	1001TECSOL	1001 Technology	12/31/2018
GBRANCH	1/4/2019	8	136.97	0.00	INTERNAL	1001TECSOL	1001 Technology	12/31/2018

Tasks

TaskID	Category
PTO	PTO
INTERNAL	Int Admin
1001TM	Billable
1001TB	Billable
2001TM	Billable

Join Kind

Left Outer (all from first, matching from second)

Use fuzzy matching to perform the merge

> Fuzzy matching options

✓ The selection matches 10991 of 10991 rows from the first table.

OK

Cancel

File Home Transform Add Column File Home Transform Add Column View Tools Help

Close & Apply New Source Recent Sources Enter Data Data source settings Close & Apply Par Close & Apply New Source Recent Sources Enter Data Data source settings Manage Parameters Refresh Preview Manage Choose Remove Columns Keep Rows Remove Rows Split Column Group By Data Type: Table Use First Row as Headers Merge Queries Append Queries Combine Files Text Analytics Vision Combine Files Azure Machine Learning Close New Query Data Sources Parameters Query Manage Columns Reduce Rows Sort Transform Combine AI Insights

Queries [8]

- Budgets and Forecasts
- People
- Tasks
- January
- January (2)
- Tasks (2) Delete
- Janua
- Tasks

Queries [6]

- Budgets and Forecasts
- People
- Tasks
- January
- January (2)
- January (3)

Table

```
= Table.NestedJoin(#"Changed Type", {"TaskID"}, Tasks, {"TaskID"}, "Tasks", JoinKind.LeftOuter)
```

date	1.2 TotalHoursBilled	A ^B C TimesheetBatchID	A ^B C TimesheetID	1.2 TotalHours	A ^B C PayType	Tasks
1	1/4/2019	0	WB122918	TMS162977	40	SALARY
2	1/4/2019	0	WB122918	TMS162977	40	SALARY
3	1/4/2019	0	WB122918	TMS162977	40	SALARY
4	1/4/2019	0	WB122918	TMS162977	40	SALARY
5	1/4/2019	27	WB122918	TMS162955	27	SALARY
6	1/4/2019	27	WB122918	TMS162955	27	SALARY
7	1/4/2019	27	WB122918	TMS162955	27	SALARY
8	1/4/2019	27	WB122918	TMS162955	27	SALARY
9	1/4/2019	27	WB122918	TMS162955	27	SALARY
10	1/4/2019	24	WB122918	TMS163049	40	SALARY
11	1/4/2019	24	WB122918	TMS163049	40	SALARY
12	1/4/2019	24	WB122918	TMS163049	40	SALARY
13	1/4/2019	24	WB122918	TMS163049	40	SALARY
14	1/4/2019	24	WB122918	TMS162548	40	SALARY
15	1/4/2019	24	WB122918	TMS162548	40	SALARY
16	1/4/2019	24	WB122918	TMS162548	40	SALARY
17	1/4/2019	24	WB122918	TMS162548	40	SALARY
18	1/4/2019	2.25	WB122918LATE	TMS163151	2.25	ADMINISTRATION
19	1/4/2019	2.25	WB122918LATE	TMS163151	2.25	ADMINISTRATION
20	1/4/2019	24	WB122918	TMS163018	40	SALARY
21	1/4/2019	24	WB122918	TMS163018	40	SALARY
22	1/4/2019	24	WB122918	TMS163018	40	SALARY
23	1/4/2019	24	WB122918	TMS163018	40	SALARY
24	1/4/2019	37	WB122918	TMS163092	37	SUB-CONTRACTOR
25	1/4/2019	37	WB122918	TMS163092	37	SUB-CONTRACTOR
26	1/4/2019	37	WB122918	TMS163092	37	SUB-CONTRACTOR
27	1/4/2019	30	WB122918	TMS162942	30	SUB-CONTRACTOR
28						

16 COLUMNS, 999+ ROWS Column profiling based on top 1000 rows

PREVIEW DOWNLOADED AT 11:14 PM

Query Settings

PROPERTIES

- Name: January (3)
- All Properties

APPLIED STEPS

- Source
- Navigation
- Promoted Headers
- Changed Type
- Merged Queries
- Expanded Tasks



Practice 2 - Power Query Editor

File Home Transform Add Column View Tools Help

Close & Apply New Source Recent Enter Data Data source settings Manage Parameters Refresh Preview Advanced Editor Choose Columns Remove Columns Keep Rows Remove Rows Sort Data Type: Any Split Column Group By Use First Row as Headers Replace Values Merge Queries Append Queries Combine Files Text Analytics Vision Combine Azure Machine Learning AI Insights

Queries [6]

Budgets and Forecasts
People
Tasks
January
January (2)
January (3)

= Source{[Item="January",Kind="Sheet"]}[Data]

ABC 123 Column1	ABC 123 Column2	ABC 123 Column3	ABC 123 Column4	ABC 123 Column5	ABC 123 Column6	ABC 123 Column7
EmployeeID	Date	Hours	HourlyCost	HourlyRate	TaskID	JobID
1 GBRANCH	1/1/2019	8	136.972	0 PTO	1001TECSOL	
2 GBRANCH	1/2/2019	8	136.972	0 INTERNAL	1001TECSOL	
3 GBRANCH	1/3/2019	8	136.972	0 INTERNAL	1001TECSOL	
4 GBRANCH	1/4/2019	8	136.972	0 INTERNAL	1001TECSOL	
5 GBRANCH						
6 BREESE	1/2/2019	7	104.368	125 1001TM	CLEV003201	
7 BREESE	1/2/2019	3	104.368	75.5 1001TB	CLEV003201	
8 BREESE	1/3/2019	3	104.368	75.5 1001TB	CLEV003201	
9 BREESE	1/3/2019	8	104.368	125 1001TM	CLEV003201	
10 BREESE	1/4/2019	6	104.368	125 1001TM	CLEV003201	
11 FMAYNARD	1/2/2019	8	74.283	130 2001TM	CLEV003201	
12 FMAYNARD	1/3/2019	8	74.283	130 2001TM	CLEV003201	
13 FMAYNARD	1/4/2019	8	74.283	130 2001TM	CLEV003201	
14 FMAYNARD	1/1/2019	8	74.283	0 PTO	2001ACCSOL	
15 ACERVANTES	1/2/2019	8	67.067	105 1001TM	CLEV002983	
16 ACERVANTES	1/3/2019	8	67.067	105 1001TM	CLEV002983	
17 ACERVANTES	1/4/2019	8	67.067	105 1001TM	CLEV002983	
18 ACERVANTES	1/1/2019	8	67.067	0 PTO	1001TECSOL	
19 JFRAZIER	1/2/2019	1.5	122.694	175 1001TM	CLEV003173	
20 JFRAZIER	1/4/2019	0.75	122.694	175 1001TM	CLEV003173	
21 EMORSE	1/2/2019	8	75.229	100 2001TM	CLEV002797	
22 EMORSE	1/3/2019	8	75.229	100 2001TM	CLEV002797	
23 EMORSE	1/4/2019	8	75.229	100 2001TM	CLEV002797	
24 EMORSE	1/1/2019	8	75.229	0 PTO	2001ACCSOL	
25 LMOORE	1/2/2019	9	60.5	100 1001TM	CLEV003176	
26 LMOORE	1/3/2019	10	60.5	100 1001TM	CLEV003176	
27 LMOORE	1/4/2019	10	60.5	100 1001TM	CLEV003176	
28						

15 COLUMNS, 999+ ROWS Column profiling based on top 1000 rows

Query Settings

PROPERTIES

Name: January (2)

APPLIED STEPS

Source
Navigation (highlighted with red box)
Promoted Headers
Changed Type
Merged Queries
Expanded Tasks

PREVIEW DOWNLOADED AT 11:14 PM

Navigation

Specify a resource to navigate to.

Hours.xlsx [3]

February

January

March

OK

Cancel

Practice 2 - Power Query Editor

File **Home** **Transform** **Add Column** **View** **Tools** **Help**

Close & Apply **New Source** **Recent Sources** **Enter Data** **Data source settings** **Manage Parameters** **Refresh Preview** **Properties Advanced Editor** **Choose Columns** **Remove Columns** **Keep Rows** **Remove Rows** **A₁Z₁** **A₁Z₁** **Data Type: Text** **Merge Queries** **Text Analytics**
Close **New Query** **Data Sources** **Parameters** **Query** **Manage Columns** **Reduce Rows** **Sort** **Split Column** **Group By** **Use First Row as Headers** **Append Queries** **Vision**
Combine Files **Combine** **AI Insights** **Replace Values**

Queries [6]

= Table.ExpandTableColumn(#"Merged Queries", "Tasks", {"Category"}, {"Category"})

	EmployeeID	Date	1.2 Hours	\$ HourlyCost	\$ HourlyRate	TaskID	JobID
1	MELLIS	2/4/2019	8	59.04	105.00	1001TM	NASH0000852
2	MELLIS	2/5/2019	8	59.04	105.00	1001TM	NASH0000852
3	MELLIS	2/6/2019	8	59.04	105.00	1001TM	NASH0000852
4	MELLIS	2/7/2019	8	59.04	105.00	1001TM	NASH0000852
5	MELLIS	2/8/2019	8	59.04	105.00	1001TM	NASH0000852
6	RMARKS	2/4/2019	8	82.50	120.00	1001TM	NASH0000859
7	RMARKS	2/5/2019	8	82.50	120.00	1001TM	NASH0000859
8	RMARKS	2/6/2019	8	82.50	120.00	1001TM	NASH0000859
9	RMARKS	2/7/2019	8	82.50	120.00	1001TM	NASH0000859
10	RMARKS	2/8/2019	8	82.50	120.00	1001TM	NASH0000859
11	WBENNETT	2/4/2019	8	97.96	125.00	1001TM	NASH0000351
12	WBENNETT	2/5/2019	8	97.96	125.00	1001TM	NASH0000351
13	WBENNETT	2/6/2019	8	97.96	125.00	1001TM	NASH0000351
14	WBENNETT	2/7/2019	8	97.96	125.00	1001TM	NASH0000351
15	WBENNETT	2/8/2019	8	97.96	125.00	1001TM	NASH0000351
16	LHUFFMAN	2/4/2019	9	89.84	110.59	1001TM	NASH0000299
17	LHUFFMAN	2/5/2019	7	89.84	110.59	1001TM	NASH0000299
18	LHUFFMAN	2/6/2019	8	89.84	110.59	1001TM	NASH0000299
19	LHUFFMAN	2/7/2019	8	89.84	110.59	1001TM	NASH0000299
20	LHUFFMAN	2/8/2019	8	89.84	110.59	1001TM	NASH0000299
21	RCOMPTON	2/4/2019	6	67.21	110.00	2001TM	NASH0000762
22	RCOMPTON	2/5/2019	8	67.21	110.00	2001TM	NASH0000762
23	RCOMPTON	2/6/2019	8	67.21	110.00	2001TM	NASH0000762
24	RCOMPTON	2/7/2019	6	67.21	110.00	2001TM	NASH0000762
25	RCOMPTON	2/8/2019	8	67.21	110.00	2001TM	NASH0000762
26	MSELLERS	2/4/2019	8	64.35	100.00	3001TM	NASH0000641
27	MSELLERS	2/5/2019	8	64.35	100.00	3001TM	NASH0000641
28							

16 COLUMNS, 999+ ROWS Column profiling based on top 1000 rows

Query Settings

PROPERTIES

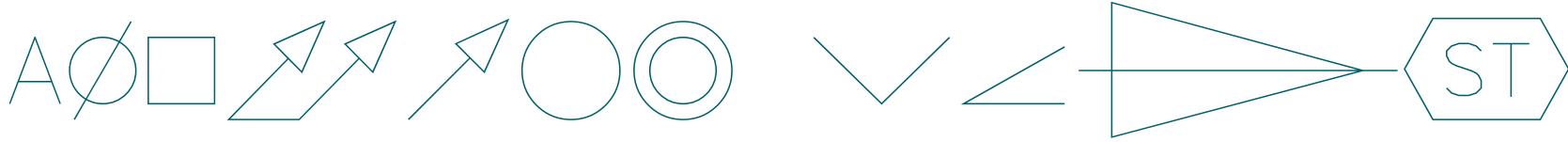
Name: February

All Properties

APPLIED STEPS

- Source
- Navigation
- Promoted Headers
- Changed Type
- Merged Queries
- Expanded Tasks

PREVIEW DOWNLOADED AT 11:14 PM



Practice 2 - Power Query Editor

File Home Transform Add Column View Tools Help

Close & Apply New Source Recent Enter Data Data source settings Manage Parameters Refresh Preview Advanced Editor Choose Columns Remove Columns Keep Rows Remove Rows Sort Split Column Group By Data Type: Any Use First Row as Headers Merge Queries Append Queries Text Analytics Vision Combine Files Combine Azure Machine Learning AI Insights Close New Query Data Sources Parameters Query Manage Columns Reduce Rows Sort Transform Combine AI Insights

Queries [6]

= Source{[Item="March",Kind="Sheet"]}[Data]

	ABC 123 Column1	ABC 123 Column2	ABC 123 Column3	ABC 123 Column4	ABC 123 Column5	ABC 123 Column6	ABC 123 Column7
1	EmployeeID	Date	Hours	HourlyCost	HourlyRate	TaskID	JobID
2	MELLIS	3/4/2019	8	59.037	105	1001TM	NASH0000852
3	MELLIS	3/5/2019	8	59.037	105	1001TM	NASH0000852
4	MELLIS	3/6/2019	8	59.037	105	1001TM	NASH0000852
5	MELLIS	3/7/2019	8	59.037	105	1001TM	NASH0000852
6	MELLIS	3/8/2019	8	59.037	105	1001TM	NASH0000852
7	RMARKS	3/4/2019	8	82.5	120	1001TM	NASH0000859
8	RMARKS	3/5/2019	8	82.5	120	1001TM	NASH0000859
9	RMARKS	3/6/2019	8	82.5	120	1001TM	NASH0000859
10	RMARKS	3/7/2019	8	82.5	120	1001TM	NASH0000859
11	RMARKS	3/8/2019	8	82.5	120	1001TM	NASH0000859
12	WBENNETT	3/4/2019	8	97.955	125	1001TM	NASH0000351
13	WBENNETT	3/5/2019	8	97.955	125	1001TM	NASH0000351
14	WBENNETT	3/6/2019	8	97.955	125	1001TM	NASH0000351
15	WBENNETT	3/7/2019	8	97.955	125	1001TM	NASH0000351
16	WBENNETT	3/8/2019	8	97.955	125	1001TM	NASH0000351
17	MSELLERS	3/4/2019	8	64.35	100	3001TM	NASH0000641
18	MSELLERS	3/5/2019	8	64.35	100	3001TM	NASH0000641
19	MSELLERS	3/6/2019	8	64.35	100	3001TM	NASH0000641
20	MSELLERS	3/7/2019	8	64.35	100	3001TM	NASH0000641
21	MSELLERS	3/8/2019	8	64.35	100	3001TM	NASH0000641
22	RCOMPTON	3/4/2019	8	67.21	110	2001TM	NASH0000762
23	RCOMPTON	3/5/2019	8	67.21	110	2001TM	NASH0000762
24	RCOMPTON	3/6/2019	8	67.21	110	2001TM	NASH0000762
25	RCOMPTON	3/7/2019	8	67.21	110	2001TM	NASH0000762
26	LHUFFMAN	3/4/2019	8	89.837	110.59	1001TM	NASH0000299
27	LHUFFMAN	3/5/2019	8	89.837	110.59	1001TM	NASH0000299
28							

15 COLUMNS, 999+ ROWS Column profiling based on top 1000 rows

Query Settings

PROPERTIES

Name: March

APPLIED STEPS

- Source
- Navigation
- Promoted Headers
- Changed Type
- Merged Queries
- Expanded Tasks

PREVIEW DOWNLOADED AT 11:14 PM

APPENDIX C – OS

Store	Order Date	Quantity
127 Sai Gon	01/02/2001	10
127 Sai Gon	02/02/2001	15

Store	Order Date	Quantity
07 Ha Noi	01/02/2001	25
07 Ha Noi	02/02/2001	30

Store	Order Date	Quantity
127 Sai Gon	01/02/2001	10
127 Sai Gon	02/02/2001	15
07 Ha Noi	01/02/2001	25
07 Ha Noi	02/02/2001	30

A P P O Ø = Ø Ø / ¢ — Ø Ø = Ø S

Practice 2 - Power Query Editor

File Home Transform Add Column View Tools Help

Close & Apply New Source Recent Enter Data Data source settings Manage Parameters Refresh Preview Advanced Editor Properties Choose Columns Remove Columns Keep Rows Remove Rows Split Column Group By Data Type: Text Use First Row as Headers Sort Merge Queries Append Queries Text Analytics Append Queries Machine Learning Append Queries as New Insights

Queries [6]

Budgets and Forecasts

People Tasks January February March

= Table.TransformColumnTypes(#"Filled Down",{{"Dept", type text}})

	Location	Dept	Type	Month	Value
1	Cleveland	1001	Budget	Jan	31855.1
2	Cleveland	1001	Budget	Feb	33985.1
3	Cleveland	1001	Budget	Mar	38741.21
4	Cleveland	1001	Budget	Apr	37042.63
5	Cleveland	1001	Budget	May	36216.9
6	Cleveland	1001	Budget	Jun	36919.86
7	Cleveland	1001	Budget	Jul	37041.17
8	Cleveland	1001	Budget	Aug	36447.85
9	Cleveland	1001	Budget	Sep	33505.11
10	Cleveland	1001	Budget	Oct	37192.39
11	Cleveland	1001	Budget	Nov	34401
12	Cleveland	1001	Budget	Dec	29134.96
13	Cleveland	2001	Budget	Jan	14917.86
14	Cleveland	2001	Budget	Feb	15200.85
15	Cleveland	2001	Budget	Mar	19323.06
16	Cleveland	2001	Budget	Apr	17673.31
17	Cleveland	2001	Budget	May	17411.26
18	Cleveland	2001	Budget	Jun	18658.18
19	Cleveland	2001	Budget	Jul	15309.93
20	Cleveland	2001	Budget	Aug	15228.32
21	Cleveland	2001	Budget	Sep	13310.64
22	Cleveland	2001	Budget	Oct	13667.68
23	Cleveland	2001	Budget	Nov	13291.05
24	Cleveland	2001	Budget	Dec	12497.06
25	Cleveland	3001	Budget	Jan	1787.64
26	Cleveland	3001	Budget	Feb	1643.23
27	Cleveland	3001	Budget	Mar	1907.65
28	Cleveland	3001	Budget	Apr	1756.22

5 COLUMNS, 192 ROWS Column profiling based on top 1000 rows

Query Settings

PROPERTIES

Name: Budgets and Forecasts

APPLIED STEPS

Source, Navigation, Promoted Headers, Changed Type, Removed Bottom Rows, Filtered Rows, Unpivoted Columns, Renamed Columns, Filled Down, **Changed Type1**

PREVIEW DOWNLOADED ON SATURDAY

APPEND = APPENDS

Append

Concatenate rows from three or more tables into a single table.

Two tables Three or more tables

Available tables

- Budgets and Forecasts
- People
- Tasks
- January
- February
- March

Add >>

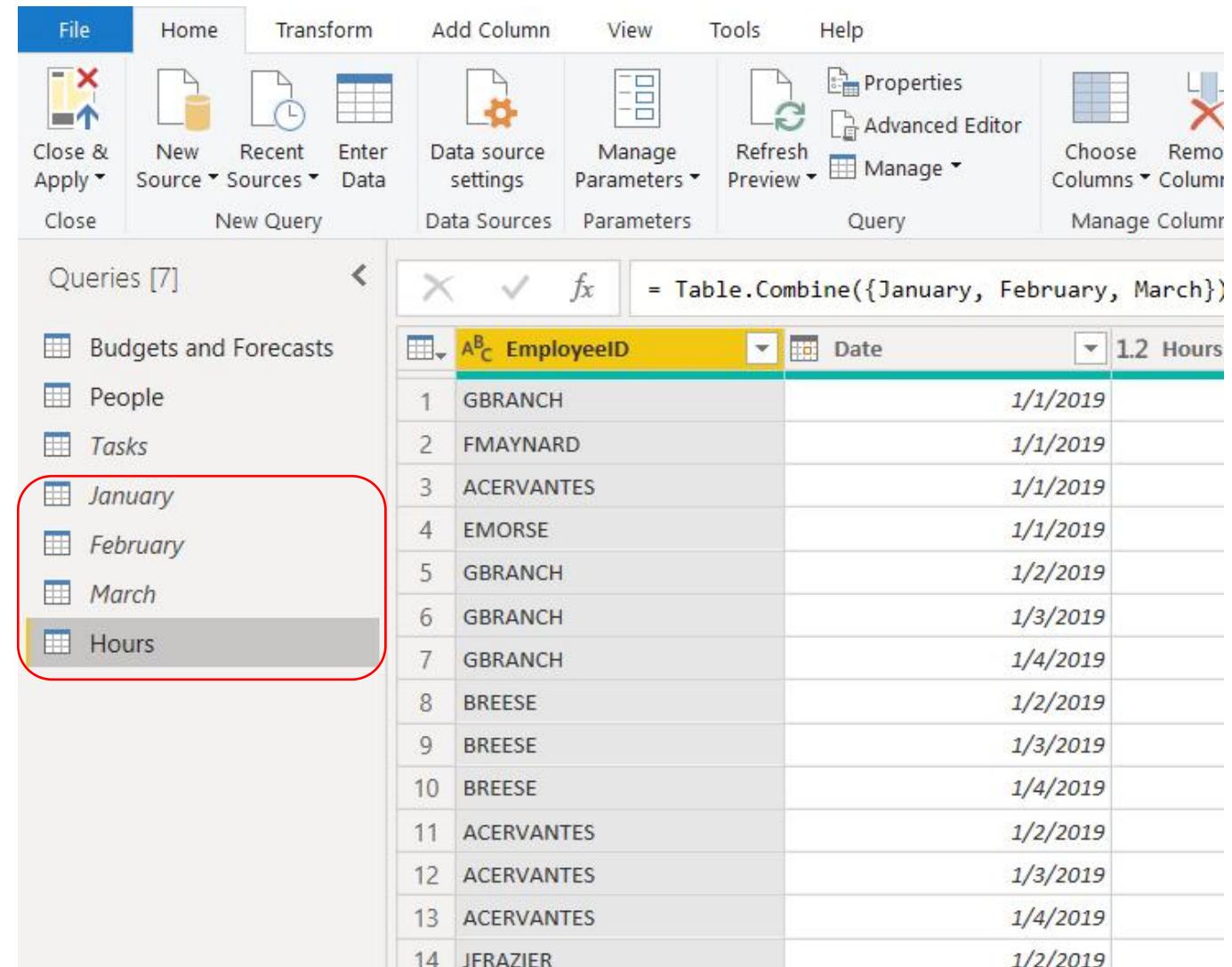
Tables to append

- January
- February
- March

OK Cancel

APPoD = PBI → ODS

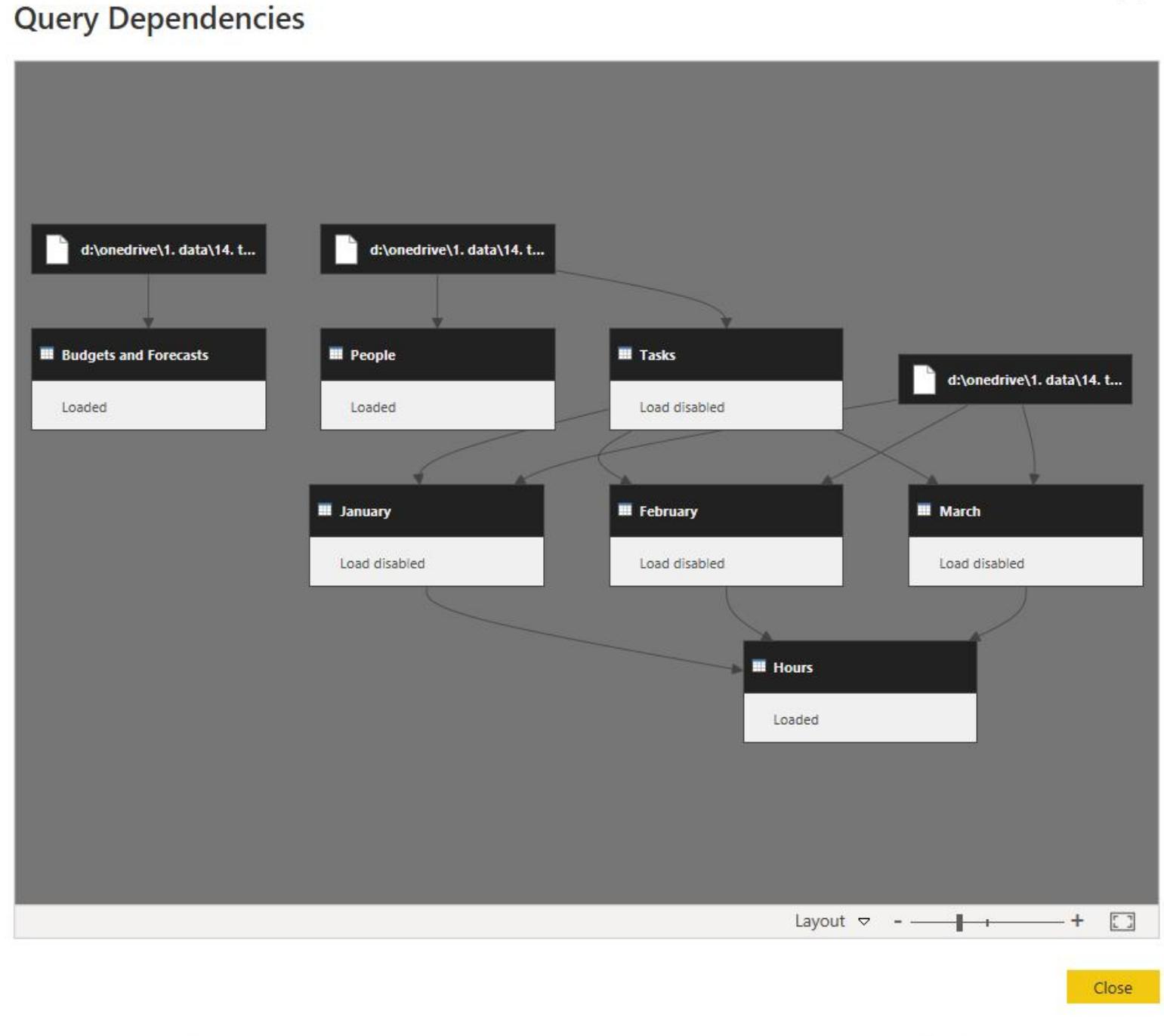
- We have used multiple pages in a single Excel workbook for the **Hours** query. In a production scenario, this would likely involve multiple Excel workbooks, one for each month. In this case, if you have many files in the same format, consider using a **Combine Binaries (Folder)** query:
<https://docs.microsoft.com/en-us/power-bi/desktop-combine-binaries>.

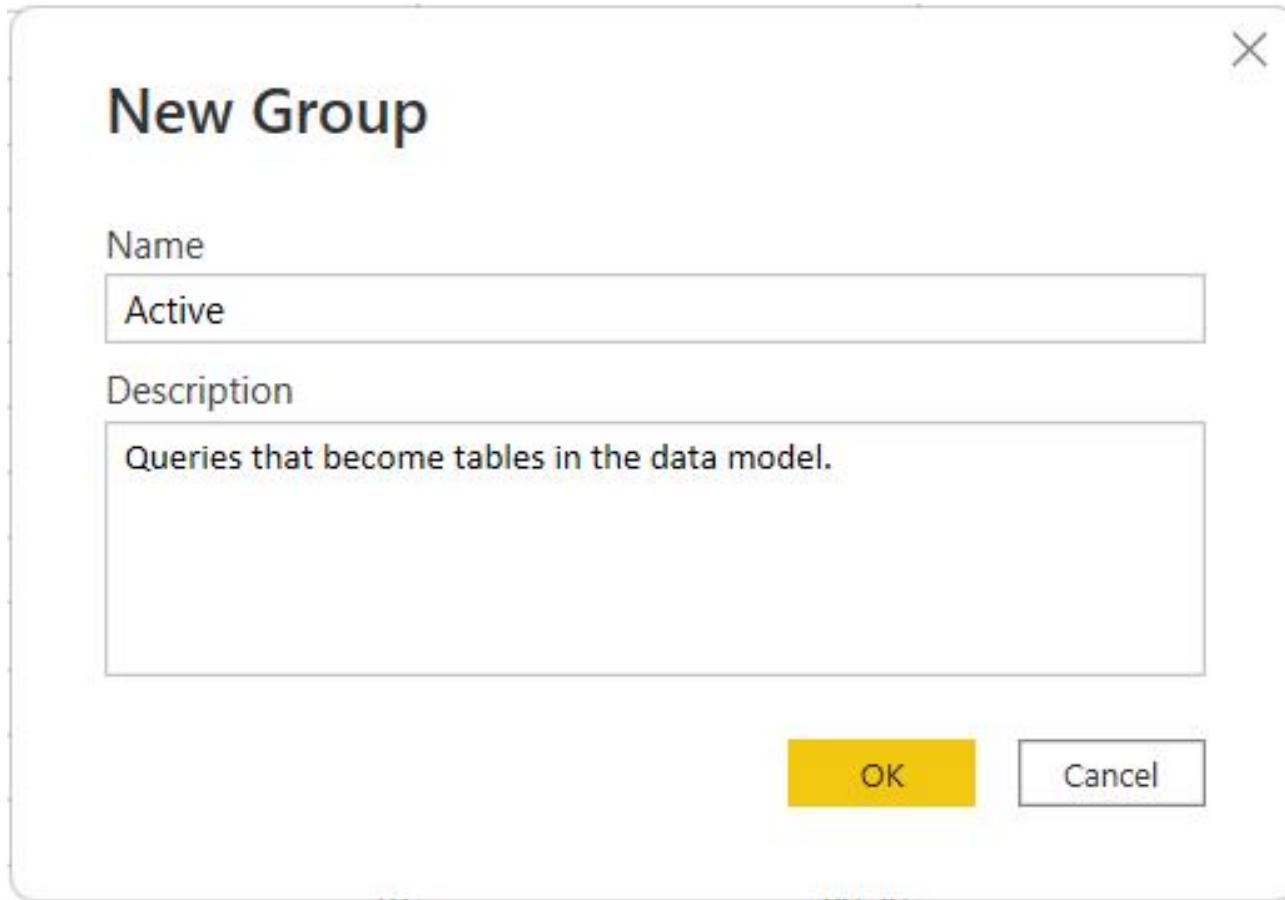


The screenshot shows the Power BI Desktop interface. The ribbon is visible at the top with tabs: File, Home, Transform, Add Column, View, Tools, and Help. The 'File' tab is selected. Below the ribbon, there's a toolbar with icons for Close & Apply, New Source, Recent Sources, Enter Data, Data source settings, Manage Parameters, Refresh Preview, Properties, Advanced Editor, Choose Columns, and Remove Columns. The 'Queries [7]' pane on the left lists seven queries: Budgets and Forecasts, People, Tasks, January, February, March, and Hours. The 'January', 'February', 'March', and 'Hours' queries are highlighted with a red rounded rectangle. To the right of the queries is a table preview showing columns: EmployeeID, Date, and Hours. The table contains 14 rows of data.

	EmployeeID	Date	Hours
1	GBRANCH	1/1/2019	
2	FMAYNARD	1/1/2019	
3	ACERVANTES	1/1/2019	
4	EMORSE	1/1/2019	
5	GBRANCH	1/2/2019	
6	GBRANCH	1/3/2019	
7	GBRANCH	1/4/2019	
8	BREESE	1/2/2019	
9	BREESE	1/3/2019	
10	BREESE	1/4/2019	
11	ACERVANTES	1/2/2019	
12	ACERVANTES	1/3/2019	
13	ACERVANTES	1/4/2019	
14	JFRAZIER	1/2/2019	

- The Query Dependencies window is useful for understanding how queries are related to one another and how queries are organized.





The screenshot shows the Power BI Query Editor interface. The top navigation bar includes File, Home, Transform, Add Column, View, Tools, and Help. The 'Query Settings' icon is selected. The 'Layout' tab is active, showing options for Formula Bar, Monospaced, Column distribution, Show whitespace, Column profile, and Column quality. The 'Data Preview' tab is also visible. In the center, there is a list of 'Queries [7]' and a preview pane showing a table with columns EmployeeID, Date, and Month. A context menu is open over the 'Hours' table in the preview pane, with the 'Move To Group' option highlighted. A red circle highlights the context menu options. The 'Hours' table row in the preview pane is also highlighted with a red circle.

EmployeeID	Date	Month
1	GBRANCH	
2	FMAYNARD	
3	ACERVANTES	
4	EMORSE	
5	GBRANCH	March
6	GBRANCH	

The screenshot shows the Power BI Query Editor interface. The ribbon tabs at the top are File, Home, Transform, Add Column, View (selected), and Tools. The 'Query Settings' icon is highlighted. Under 'View', 'Formula Bar' is checked, while 'Monospaced', 'Column distribution', 'Show whitespace', 'Column profile', and 'Column quality' are unchecked. The 'Layout' tab is selected. In the main area, the 'Queries [7]' pane shows two groups: 'Active [3]' (highlighted with a red box) containing 'Budgets and Forecast...', 'People', and 'Hours'; and 'Other Queries [4]' containing 'Tasks', 'January', 'February', and 'March'. Below the queries is a 'Data Preview' table with columns 'A' (Index), 'B' (Location), and 'C' (Value). The first 12 rows show data for 'Cleveland'.

A	B	C
1	Cleveland	100
2	Cleveland	100
3	Cleveland	100
4	Cleveland	100
5	Cleveland	100
6	Cleveland	100
7	Cleveland	100
8	Cleveland	100
9	Cleveland	100
10	Cleveland	100
11	Cleveland	100
12	Cleveland	100

The screenshot shows a context menu for the 'Budgets and Forecast...' query in the 'Active' group. The menu items are: Copy, Paste, Delete, Rename, Enable load (checked), Include in report refresh (checked), Duplicate, Reference, Move To Group (highlighted with a red box), Move Up, Move Down, Create Function..., Convert To Parameter, Advanced Editor, and Properties... A red box also highlights the 'Active' group in the 'Move To Group' submenu.

Practice 2 - Power Query Editor

File **Home** **Transform** **Add Column** **View** **Tools** **Help**

Query Settings

Formula Bar Monospaced Column distribution Always allow Show whitespace Column profile Column quality

Layout **Data Preview** **Columns** **Parameters** **Advanced** **Dependencies**

Queries [7]

- Active [3]
 - Budgets and Forecast...
 - People
 - Hours**
- Other Queries [4]
 - Tasks
 - January
 - February
 - March

= Table.Combine({January, February, March})

	A ^B C EmployeeID	Date	1.2 Hours	\$ HourlyCost	\$ HourlyRate	A ^B C TaskID	A ^B C JobID
1	GBRANCH	1/1/2019	8	136.97	0.00	PTO	1001TECSOL
2	FMaynard	1/1/2019	8	74.28	0.00	PTO	2001ACCSOL
3	ACERVANTES	1/1/2019	8	67.07	0.00	PTO	1001TECSOL
4	EMORSE	1/1/2019	8	75.23	0.00	PTO	2001ACCSOL
5	GBRANCH	1/2/2019	8	136.97	0.00	INTERNAL	1001TECSOL
6	GBRANCH	1/3/2019	8	136.97	0.00	INTERNAL	1001TECSOL
7	GBRANCH	1/4/2019	8	136.97	0.00	INTERNAL	1001TFCSOL

Column statistics

Count	1000
Error	0
Empty	0
Distinct	242
Unique	6
Empty string	0
Min	AANDRE...
Max	WSUAREZ

Value distribution

Name	Count
CBARRY	1000
MBRIGHT	~950
ARANDOLPH	~900
NBOND	~850
BARCHER	~800
CLEWIS	~750
NRITTER	~700
LIRWIN	~650
CGOODWIN	~600
JBEASLEY	~550
BHARRINGTON	~500

Query Settings

PROPERTIES

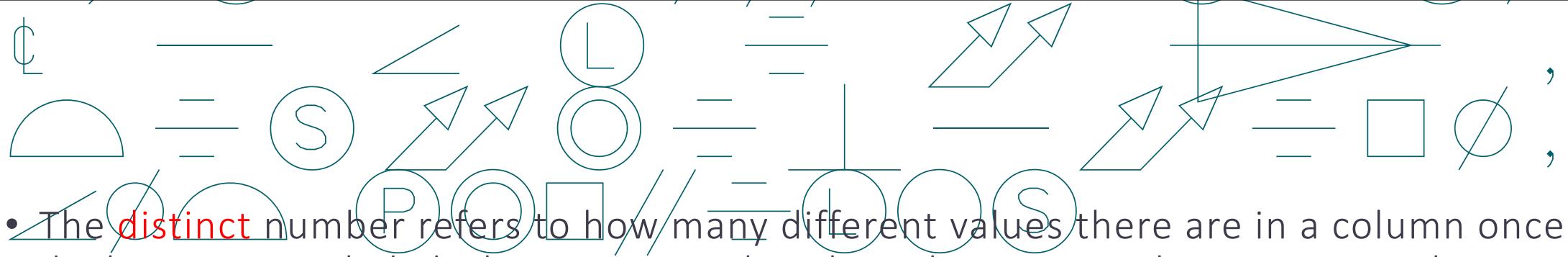
Name: Hours
All Properties

APPLIED STEPS

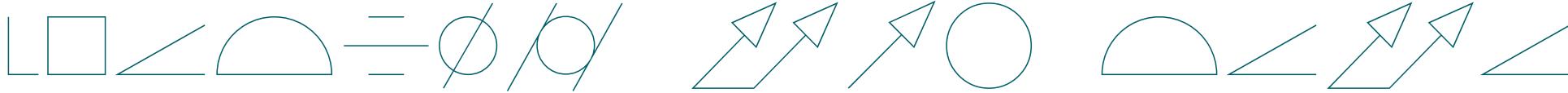
Source

16 COLUMNS, 999+ ROWS Column profiling based on top 1000 rows

PREVIEW DOWNLOADED ON SATURDAY



- The **distinct** number refers to how many different values there are in a column once duplicates are excluded. The **unique** number shows how many values occur exactly once. The number of distinct and unique values will be the same only if all values are unique.



LearnPowerBI_CH4Start - Power BI Desktop

Vien Nguyen

File Home Insert Modeling View Help

Paste Cut Copy Format painter

Get data workbook hub Data Server SQL Enter data Dataverse Recent sources

Transform Refresh data New visual Text box More visuals

New measure Quick Sensitivity Share

Clipboard Date 1/1/2019 12/31/2019

Year 2017 2018 2019

365 Count of Date

Number of Working Days by Month

Month	Working Days
January	23
February	20
March	21
April	22
May	23
June	20
July	23
August	22
September	21
October	23
November	21
December	22

Average Working Days: 22

Count of Date by Weekday

Weekday	Count of Date
Monday	52
Tuesday	53
Wednesday	52
Thursday	52
Friday	52
Saturday	52
Sunday	52

IsWorkDay

Page 1 +

Visualizations Fields

Build visual

Filters

Search

Budgets and Forecasts

Calendar

Hours

People

Add data fields here

Drill through

Cross-report Off

Keep all filters On

Add drill-through fields here

Page 1 of 1

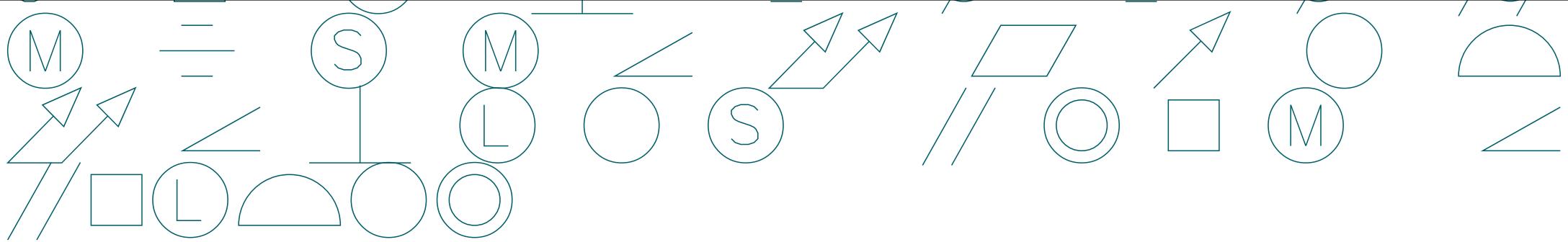
82%

	C	S	M	S	M
ID	Color	StandardCost			
WB-H098		1.87			
BC-M005		3.74			
BC-R205		3.36			
PK-7098		0.86			
RA-H123		44.88			
CL-9009		2.97			
FE-6654		8.22			
ST-1401		59.47			
HY-1023-70	Silver	20.57			
TT-M928		1.87			
TT-R982		1.49			
TT-T092		1.87			
TI-M267		9.35			
TI-M602		11.22			
TI-M823		13.09			
TI-R092		8.04			
TI-R628		9.35			
TI-R982		12.19			
TI-T723		10.84			
PA-T100	Grey	51.56			
LO-C100		10.31			
PU-0452		8.25			
PU-M044		10.31			
LT-T990		5.77			
LT-H902		14.43			
LT-H903		18.56			
HL-U509-B	Blue	13.09			
HL-U509-R	Red	13.09			
HL-U509	Black	13.09			

ID	Color	Cost
BK-T44U-60	Blue	755.15
BK-T79Y-46	Yellow	1481.94
BK-T79Y-50	Yellow	1481.94
BK-T79Y-54	Yellow	1481.94
BK-T79Y-60	Yellow	1481.94
BK-T18U-54	Blue	461.44
BK-T18U-58	Blue	461.44
BK-T18U-62	Blue	461.44
BK-T18Y-44	Yellow	461.44
BK-T18Y-50	Yellow	461.44
BK-T18Y-54	Yellow	461.44
BK-T18Y-58	Yellow	461.44
BK-T18Y-62	Yellow	461.44
BK-T79U-46	Blue	1481.94
BK-T79U-50	Blue	1481.94
BK-T79U-54	Blue	1481.94
BK-T79U-60	Blue	1481.94
BK-T44U-46	Blue	755.15
BK-T44U-50	Blue	755.15
BK-T44U-54	Blue	755.15
BK-R79Y-40	Yellow	1082.51
BK-R79Y-42	Yellow	1082.51
BK-R79Y-44	Yellow	1082.51

A _B C _C	ID	A _B C _C	Color	1.2 Cost
1	WB-H098		null	1.87
2	BC-M005		null	3.74
3	BC-R205		null	3.36
4	PK-7098		null	0.86
5	RA-H123		null	44.88
6	CL-9009		null	2.97
7	FE-6654		null	8.22
8	ST-1401		null	59.47
9	HY-1023-70		Silver	20.57
10	TT-M928		null	1.87
11	TT-R982		null	1.49
12	TT-T092		null	1.87
13	TI-M267		null	9.35
14	TI-M602		null	11.22
15	TI-M823		null	13.09
16	TI-R092		null	8.04
17	TI-R628		null	9.35
18	TI-R982		null	12.19
19	TI-T723		null	10.84
20	PA-T100		Grey	51.56
21	LO-C100		null	10.31
22	PU-0452		null	8.25
23	PU-M044		null	10.31
24	LT-T990		null	5.77
25	LT-H902		null	14.43
26	LT-H903		null	18.56
27	HL-U509-B		Blue	13.09

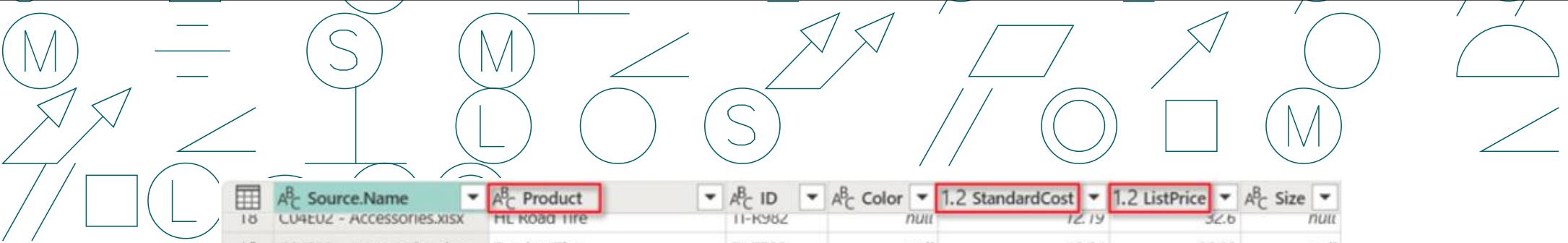
Source: Maslyuk and Raviv (2025)



Products Folder

Mismatched Column Names

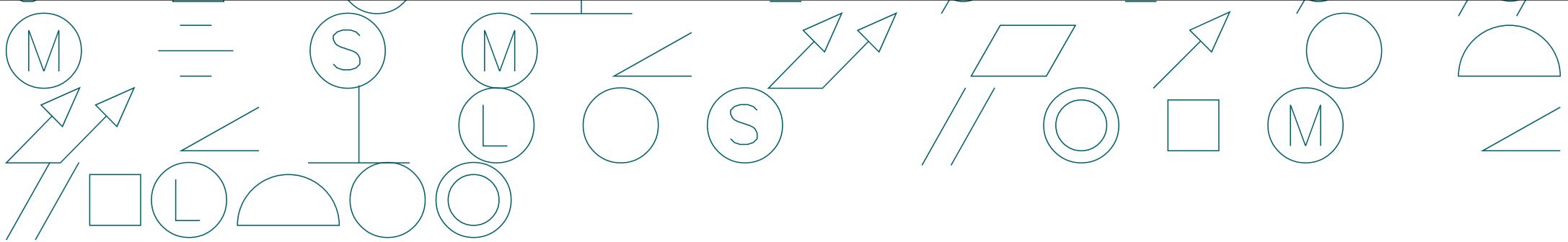
 C04E02 - Accessories.xlsx	→	Product	ID	Color	StandardCost	ListPrice
 C04E02 - Bikes.xlsx	→	Name	Product_Number	Color	Standard_Cost	List_Price
 C04E02 - Clothing.xlsx	→	Product_name	Prod_num	Pro_color	Prod_cost	Prod_price
 C04E02 - Components.xlsx	→	Name	Product Number	Color	Standard Cost	List Price



A _C Source.Name	A _C Product	A _C ID	A _C Color	1.2 StandardCost	1.2 ListPrice	A _C Size
18 C04E02 - Accessories.xlsx	HL Road Tire	II-K982	null	12.19	32.6	null
19 C04E02 - Accessories.xlsx	Touring Tire	TI-T723	null	10.84	28.99	null
20 C04E02 - Accessories.xlsx	Touring-Panniers, Large	PA-T100	Grey	51.56	125	null
21 C04E02 - Accessories.xlsx	Cable Lock	LO-C100	null	10.31	25	null
22 C04E02 - Accessories.xlsx	Minipump	PU-0452	null	8.25	19.99	null
23 C04E02 - Accessories.xlsx	Mountain Pump	PU-M044	null	10.31	24.99	null
24 C04E02 - Accessories.xlsx	Taillights - Battery-Po			5.77	13.99	null
25 C04E02 - Accessories.xlsx	Headlights - Dual-Be			14.43	34.99	null
26 C04E02 - Accessories.xlsx	Headlights - Weather			18.56	44.99	null
27 C04E02 - Accessories.xlsx	Sport-100 Helmet, Blue	HL-U509-B	Blue	13.09	34.99	null
28 C04E02 - Accessories.xlsx	Sport-100 Helmet, Red	HL-U509-R	Red	13.09	34.99	null
29 C04E02 - Accessories.xlsx	Sport-100 Helmet, Black	HL-U509	Black	13.09	34.99	null
30 C04E02 - Bikes.xlsx		null	null	60		
31 C04E02 - Bikes.xlsx		null	null	46		
32 C04E02 - Bikes.xlsx		null	null	50		
33 C04E02 - Bikes.xlsx		null	null	54		
34 C04E02 - Bikes.xlsx		null	null	60		
35 C04E02 - Bikes.xlsx		null	null	54		
36 C04E02 - Bikes.xlsx		null	null	58		
37 C04E02 - Bikes.xlsx		null	null	62		
38 C04E02 - Bikes.xlsx		null	null	44		
39 C04E02 - Bikes.xlsx		null	null	50		
40 C04E02 - Bikes.xlsx		null	null	54		
41 C04E02 - Bikes.xlsx		null	null	58		

Bikes data will have null values in columns that were used in Accessories

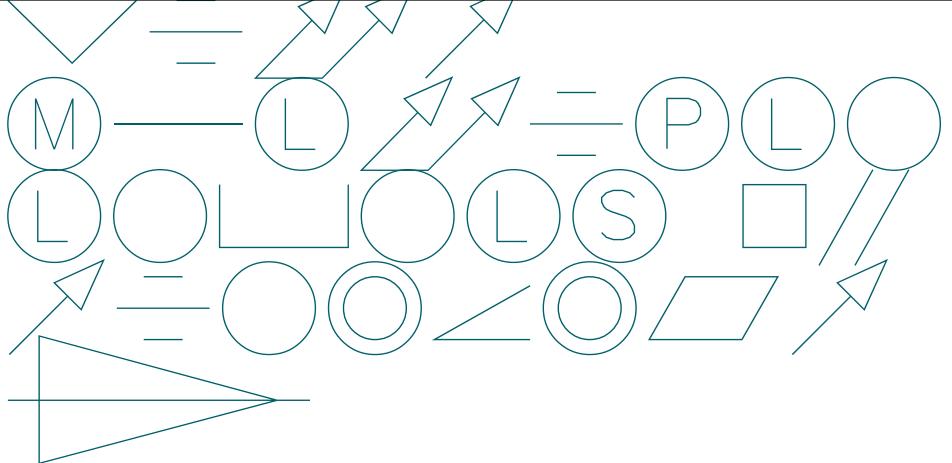
null	null	Blue	null	null	60
null	null	Yellow	null	null	46
null	null	Yellow	null	null	50
null	null	Yellow	null	null	54
null	null	Yellow	null	null	60
null	null	Blue	null	null	54
null	null	Blue	null	null	58
null	null	Blue	null	null	62
null	null	Yellow	null	null	44
null	null	Yellow	null	null	50
null	null	Yellow	null	null	54
null	null	Yellow	null	null	58



The column 'Product' of the table wasn't found.

```
Table.RemoveColumns("#Filtered Rows", {"C04E02 - Accessories.xlsx", "0"})
```

```
Table.RemoveColumns("#Filtered Rows", {Table.ColumnNames("#Filtered Rows"){0}, "0"})
```



1. Fill Down N-1 Columns

Column1	Column N-1	Column N		
		Black		
		Accessories	Bikes	
		Helmets	Mountain Bikes	Road Bikes
United Kingdom	England	Liverpool	238.659792	24566.028
		London	104.97	33290.832
		Milton Keynes		33038.094
				14012.916

2. Merge N Columns with a Delimiter

3. Transpose

Merged	Column2		
	Black		
	Accessories	Bikes	
	Helmets	Mountain Bikes	Road Bikes
United Kingdom:England:Liverpool	238.659792		24566.028
United Kingdom:England:London	104.97	33290.832	33038.094
United Kingdom:England:Milton Keynes			14012.916

4. Fill Down M-1 Columns

5. Use First Row as Headers

Column1	Column M-1	Column M	United Kingdom:England:Liverpool	United Kingdom:England:London	United Kingdom:...
Black	Accessories	Helmets	238.659792	104.97	
	Bikes	Mountain Bikes		33290.832	14012.916
		Road Bikes	24566.028	33038.094	

6. Select M Columns & Unpivot Other Columns

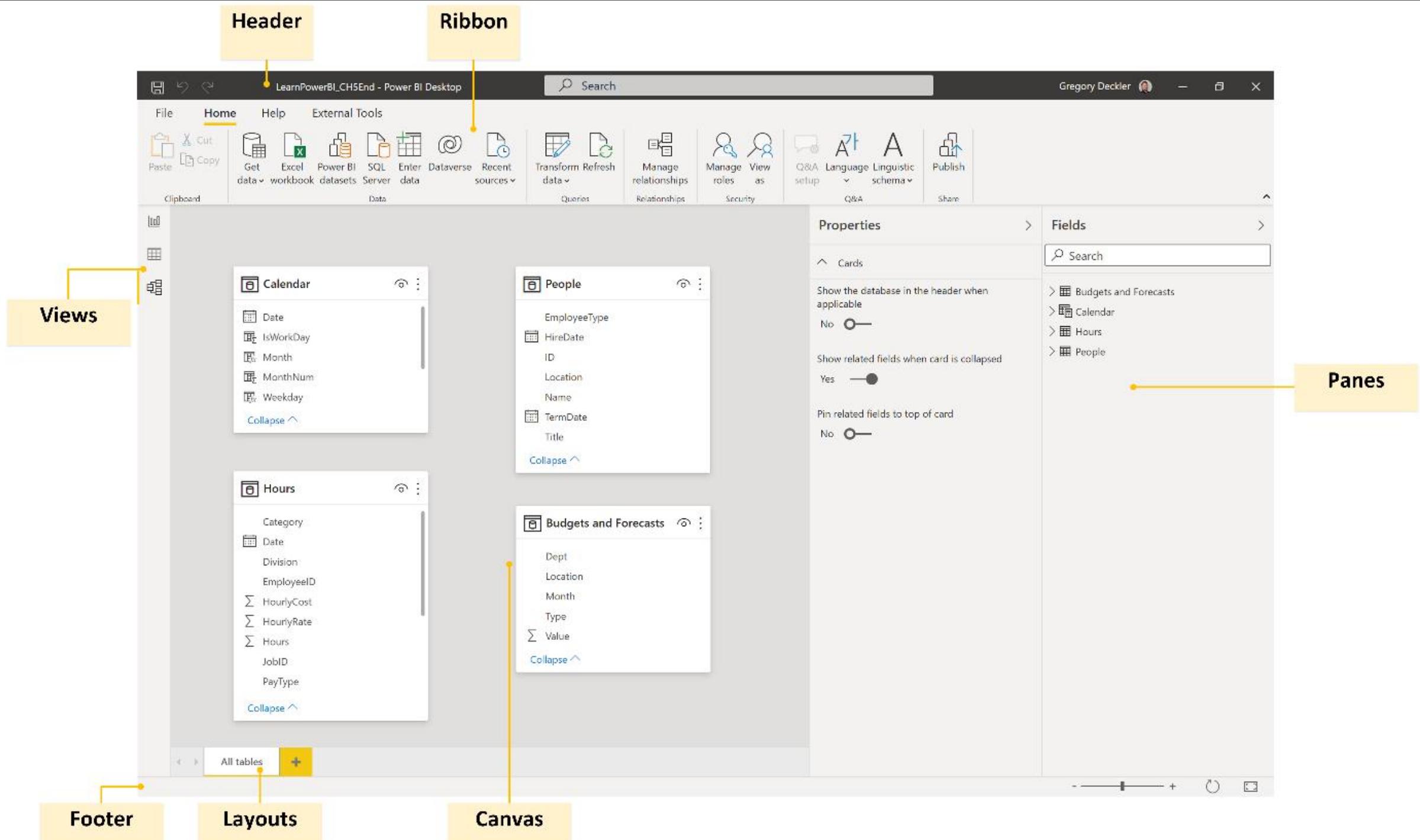
			United Kingdom:England:Liverpool	United Kingdom:England:London	United Kingdom:...
Black	Accessories	Helmets	238.659792	104.97	
Black	Bikes	Mountain Bikes		33290.832	14012.916
Black	Bikes	Road Bikes	24566.028	33038.094	

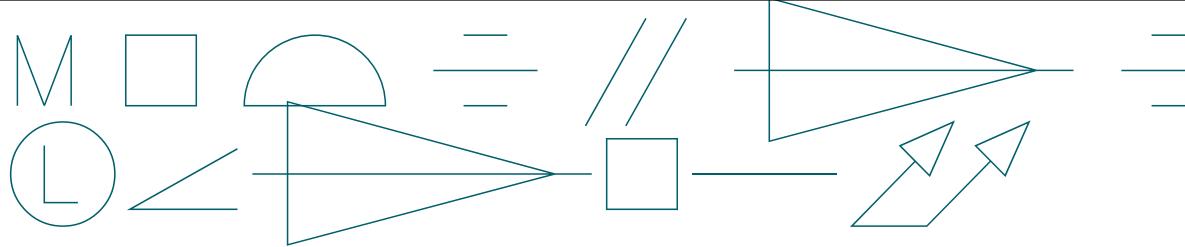
7. Split Attribute by Delimiter

Attribute	Value
United Kingdom:England:Liverpool	238.659792
United Kingdom:England:Liverpool	24566.028
United Kingdom:England:London	104.97
United Kingdom:England:London	33290.832
United Kingdom:England:London	33038.094
United Kingdom:England:Milton Keynes	14012.916
⋮	⋮

8. Rename Columns

--> Column Field 1	--> Column Field 2	--> Column Field M	Attribute.1 --> Row Field 1	Attribute.2 --> Row Field 2	Attribute.N --> Row Field N	Value
Black	Accessories	Helmets	United Kingdom	England	Liverpool	238.66
Black	Bikes	Road Bikes	United Kingdom	England	Liverpool	24566
Black	Accessories	Helmets	United Kingdom	England	Liverpool	104.97
Black	Bikes	Mountain Bikes	United Kingdom	England	Liverpool	33290.8
Black	Bikes	Road Bikes	United Kingdom	England	Liverpool	33038.1
Black	Bikes	Mountain Bikes	United Kingdom	England	Liverpool	14012.9



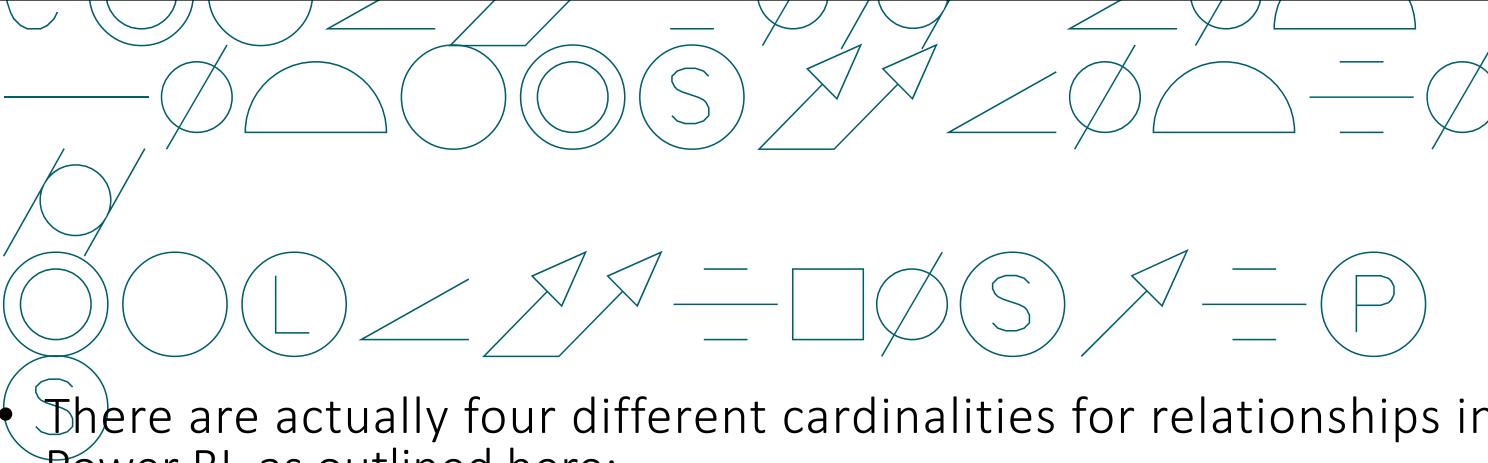


-  Date
-  IsWorkDay
-  Month
-  MonthNum
-  Weekday
-  WeekdayNum
-  WeekNum
-  Year

- Dept
- Location
- Month
- Type
- Σ Value

People
EmployeeType
HireDate
ID
Location
Name
TermDate
Title

- Hours
- Category
- Date
- Division
- EmployeeID
- HourlyCost
- HourlyRate
- Hours
- JobID
- PayType
- PeriodEndDate
- PeriodStartDate
- TaskID
- TimesheetBatchID
- TimesheetID
- TotalHours
- TotalHoursBilled

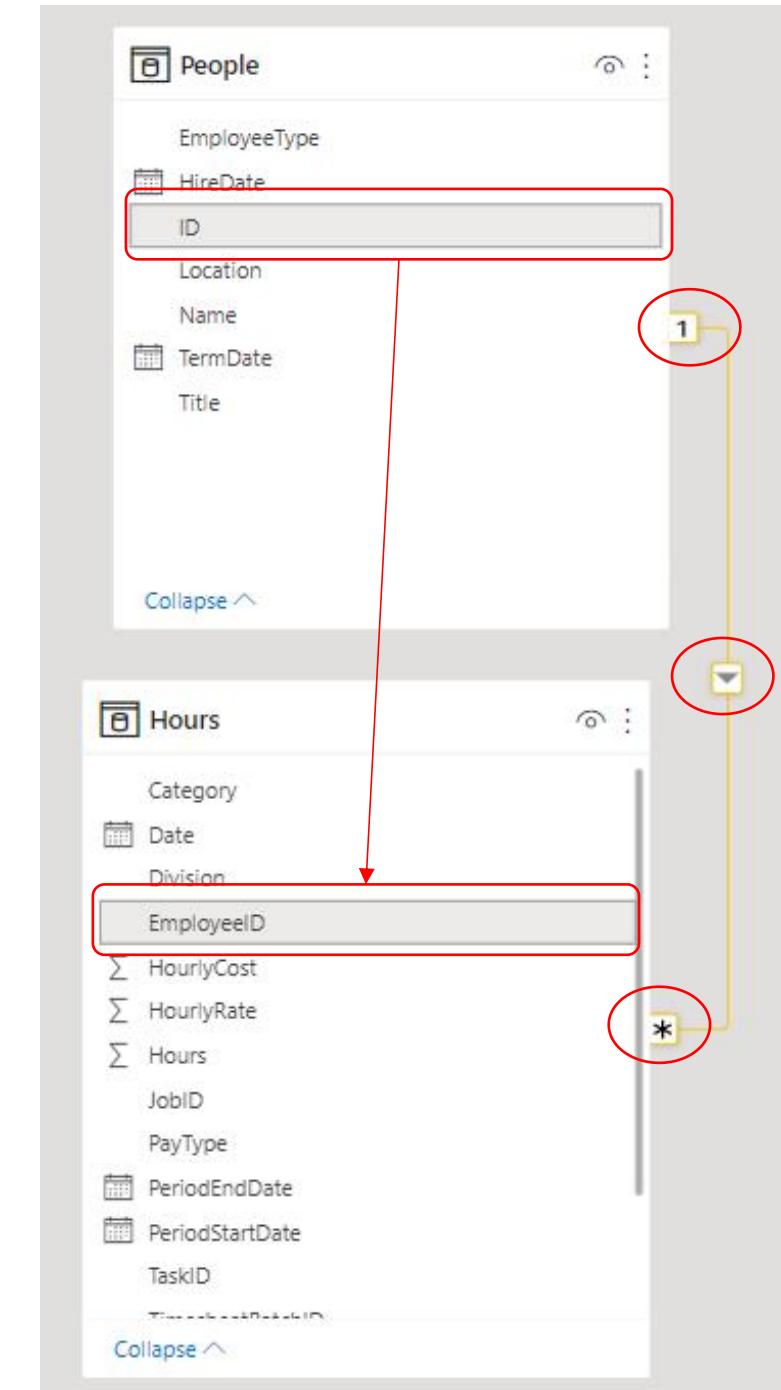


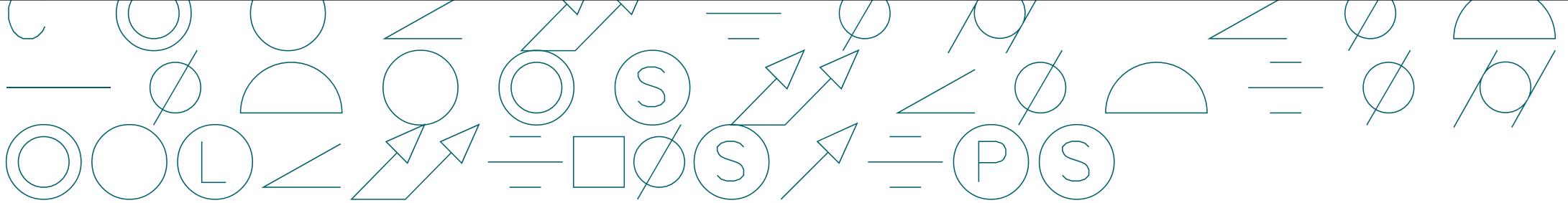
- There are actually four different cardinalities for relationships in Power BI, as outlined here:

- **One-to-one:** This means that there are unique values in each table.
- **Many-to-one:** This means that there are unique values in one table that match multiple rows in another table.
- **One-to-many:** This means that there are unique values in one table that match multiple rows in another table.
- **Many-to-many:** This means that neither table has unique values for rows. It is generally good practice to avoid these types of relationships because of their complexity and the amount of processing and resources required.

There is also an arrow icon in the middle of the line that points from **People** to **Hours**. This indicates that the **People** table filters the **Hours** table, but not vice versa. This is known as the cross filter direction. Cross filter directions can be either **Single** or **Both**, meaning that the filtering occurs only one way or bidirectionally. In simple data models, you generally don't have to worry about cross filter direction, but it can become very important if the complexity of the data model increases.

The line that forms the relationship between the tables is solid. A solid line indicates an **active relationship**. **Inactive relationships** can be created between tables, and these are represented as a dashed line. In Power BI, there can only be a single active path between tables. In other words, there can only be a single path of a relationship cross-filtering from one table to another. This is the active pathway.





- Each table can be on the *one* or the *many* side during a join. If a table is on the *one* side, it means the key in the table has a unique value for every row, and it's also known as the **primary key**. If a table is on the *many* side, it means that values in the key column are not necessarily unique for every row and may repeat, and such a key is often called a **foreign key**.



- Note that since the **Hours** table is defined first and the **People** table is defined second, the **Cardinality** value of our relationship is **Many to one (*:1)**. Also, note that the relationship is active and that the **Cross filter direction** value is **Single**.
- In many-to-one and one-to-many relationships, a **Cross filter direction** value of **Single** means that the one side of the relationship filters the many side. Finally, note that the EmployeeID column in the **Hours** table and the ID column in the **People** table are highlighted in gray. This shows us the columns that form a relationship between the two tables. Close the **Edit relationship** dialog by clicking the **Cancel** button.

Edit relationship

Select tables and columns that are related.

Hours								
EmployeeID	Date	Hours	HourlyCost	HourlyRate	TaskID	JobID	Division	
CBATES	Monday, March 4, 2019	8	\$77.088	\$120	1001TM	CLEV003201	1001 Technol	
CBATES	Tuesday, March 5, 2019	8	\$77.088	\$120	1001TM	CLEV003201	1001 Technol	
CBATES	Wednesday, March 6, 2019	8	\$77.088	\$120	1001TM	CLEV003201	1001 Technol	

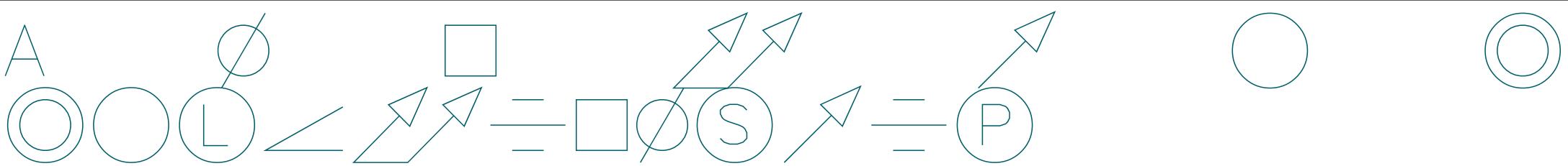
People						
ID	Name	Title	EmployeeType	TermDate	HireDate	
ACOHEN	Cohen, Arlen	CONSULTANT	SALARY	Friday, January 1, 9999	Saturday, April 1, 1995	Cl
KSHERMAN	Sherman, Kari	CONSULTANT	SALARY	Friday, January 1, 9999	Monday, October 16, 1995	Cl
HCUNNINGHAM	Cunningham, Hung	CONSULTANT	SALARY	Friday, January 1, 9999	Monday, October 16, 1995	Cl

Cardinality: Many to one (*:1) **Cross filter direction:** Single

Make this relationship active Apply security filter in both directions

Assume referential integrity

OK **Cancel**



Manage relationships

Active	From: Table (Column)	To: Table (Column)
<input checked="" type="checkbox"/>	Hours (EmployeeID)	People (ID)

New... Autodetect... Edit... Delete Close



Create relationship

Select tables and columns that are related.

Hours

EmployeeID	Date	Hours	HourlyCost	HourlyRate	TaskID	JobID	Division
CBATES	Monday, March 4, 2019	8	\$77.088	\$120	1001TM	CLEV003201	1001 Technol
CBATES	Tuesday, March 5, 2019	8	\$77.088	\$120	1001TM	CLEV003201	1001 Technol
CBATES	Wednesday, March 6, 2019	8	\$77.088	\$120	1001TM	CLEV003201	1001 Technol

Calendar

Date	Month	Year	MonthNum	WeekNum	Weekday	WeekdayNum	IsWorkDay	
1/1/2017	January	2017	1	1	Sunday	7	0	
1/2/2017	January	2017	1	1	Monday	1	1	
1/3/2017	January	2017	1	1	Tuesday	2	1	

Cardinality

Many to one (*:1)

Make this relationship active

Cross filter direction

Single

Apply security filter in both directions

Assume referential integrity

OK

Cancel

Calendar

- Date
- IsWorkDay
- Month
- MonthNum
- Weekday
- WeekdayNum
- WeekNum
- Year

[Collapse ^](#)

Budgets and Forecasts

- Dept
- Location
- Month
- Type
- Value

[Collapse ^](#)

People

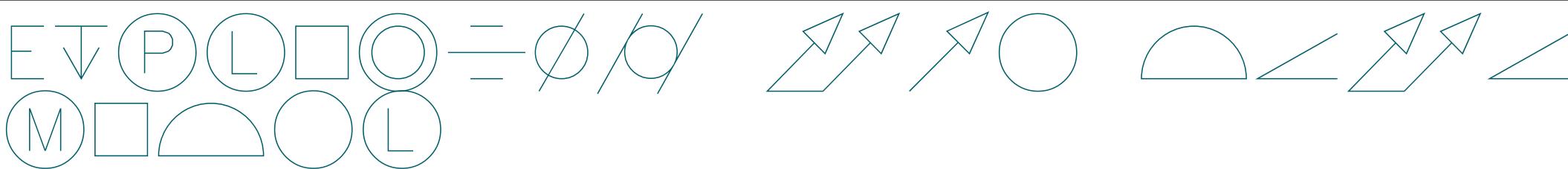
- EmployeeType
- HireDate
- ID
- Location
- Name
- TermDate
- Title

[Collapse ^](#)

Hours

- Category
- Date
- Division
- EmployeeID
- HourlyCost
- HourlyRate
- Hours
- JobID
- PayType
- PeriodEndDate
- PeriodStartDate
- TaskID

[Collapse ^](#)



Practice 3 - Power BI Desktop

Vien Nguyen

File Home Insert Modeling View Help Format Data / Drill

Cut Copy Paste Format painter Get data workbook hub Data SQL Server Enter data Dataverse Recent sources Transform Refresh data New visual Text box More visuals New measure Quick Sensitivity measure Sensitivity Publish

Clipboard Name Abbott, Eugenio Abbott, Isabelle Abbott, Pamela Acevedo, Dianne Acevedo, Laverne Acevedo, Randy Acosta, Angeline Acosta, Jaime Acosta, Rosalyn Adams, Joan Adams, Kurtis Adams, Marcy Adkins, Devon Adkins, Rita Aguilar, Jonah Aguilar, Mable Aguilar, Royce Aguirre, Burt Aguirre, Jami Aguirre, Thad Alexander, Buck Alexander, Darlene Alexander, Teresa Ali, Antonia Ali, Hannah Ali, Lenard Allen, Karen Allen, Nellie Allen, Susana Allison, Alison Allison, Gina Allison, Ophelia

Visualizations Fields

Build visual

Filters

Search

Budgets and Forecasts

Calendar

Hours

People

EmployeeType

HireDate

ID

Location

Name

TermDate

Title

Columns

Drill through

Cross-report Off

Keep all filters On

Add drill-through fields here

Utilization +

Page 1

Page 2 of 2

82%

Practice 3 - Power BI Desktop

Vien Nguyen

File Home Insert Modeling View Help Format Data / Drill

Cut Copy Format painter

Get data Excel workbook hub Data SQL Server Enter Dataverse Recent sources

New visual Text box More visuals

Transform Refresh data

Queries

Insert Calculations

Sensitivity

Quick measure

Publish

Clipboard

Name Hours

Adkins, Rita	464.00
Allen, Susana	72.00
Anderson, Pearlie	525.00
Andrews, Abram	482.75
Archer, Bryce	502.00
Arellano, Danilo	474.00
Arellano, Tanya	476.00
Atkins, Joan	486.00
Atkinson, Annmarie	461.60
Ayala, Kristi	471.50
Baird, Tami	367.00
Baker, Lionel	249.00
Ball, Jerri	510.00
Ballard, Charlie	472.00
Banks, Ambrose	472.00
Barnett, Tamika	472.00
Barr, Jack	393.50
Barrera, Tricia	184.00
Barron, Valarie	473.50
Barry, Colby	483.00
Barton, Sam	192.25
Bates, Clint	492.50
Bauer, Lacy	351.00
Bautista, Damion	360.00
Baxter, Zack	472.00
Beasley, Joni	368.00
Beck, Rae	472.00
Bell, Deshawn	169.00
Beltran, Arlie	472.00
Beltran, Elvira	387.00
Benitez, Flora	464.00
Total	177,997.15

Page 1 Utilization +

Visualizations Fields

Build visual

Filters

Hours

Category Date Division EmployeeID \sum HourlyCost \sum HourlyRate \sum Hours

Columns

Name Hours

Drill through

Cross-report Off

Keep all filters On

Add drill-through fields here

People

Page 2 of 2

82%

Practice 3 - Power BI Desktop

Vien Nguyen

File Home Insert Modeling View Help Format Data / Drill

Cut Copy Format painter

Get data Excel Data SQL Server Enter Dataverse Recent sources

Transform Refresh data New visual Text box More visuals

New measure Quick Sensitivity Publish

Sensitivity Calculations Share

Clipboard

Name Hours

Adkins, Rita	464.00
Allen, Susana	72.00
Anderson, Pearlie	525.00
Andrews, Abram	482.75
Archer, Bryce	502.00
Arellano, Danilo	474.00
Arellano, Tanya	476.00
Atkins, Joan	486.00
Atkinson, Annmarie	461.60
Ayala, Kristi	471.50
Baird, Tami	367.00
Baker, Lionel	249.00
Ball, Jerri	510.00
Ballard, Charlie	472.00
Banks, Ambrose	472.00
Barnett, Tameka	472.00
Barr, Jack	393.50
Barrera, Tricia	184.00
Barron, Valarie	473.50
Barry, Colby	483.00
Barton, Sam	192.25
Bates, Clint	492.50
Bauer, Lacy	351.00
Bautista, Damion	360.00
Baxter, Zack	472.00
Beasley, Joni	368.00
Beck, Rae	472.00
Bell, Deshawn	169.00
Beltran, Arlie	472.00
Beltran, Elvira	387.00
Benitez, Flora	464.00
Bennett, Wiley	444.00
Bentley, Eric	600.00
Total	177,997.15

Visualizations Fields

Build visual

Filters

Hours

- Category
- Date
- Division
- EmployeeID
- Σ HourlyCost
- Σ HourlyRate
- Σ Hours
- JobID
- PayType

Rows

Name

Columns

Add data fields here

Values

Hours

Drill through

Cross-report

Keep all filters

Add drill-through fields here

Page 1 Utilization +

Page 2 of 2

82%

Practice 3 - Power BI Desktop

Vien Nguyen

Home

Cut, Copy, Paste, Format painter, Clipboard

Get data, Excel, Data, SQL, Enter data, Dataverse, Recent sources, Transform data, Refresh data, New visual, Text box, More visuals, New measure, Quick measure, Sensitivity, Publish

Data / Drill

Name, Bench, Billable, Int Admin, Other, PNB, PTO, Sales Pursuit, Sales Support, Training, Total

Name	Bench	Billable	Int Admin	Other	PNB	PTO	Sales Pursuit	Sales Support	Training	Total	
Adkins, Rita		464.00								464.00	
Allen, Susana		72.00								72.00	
Anderson, Pearlie		388.00	10.00		32.00		95.00			525.00	
Andrews, Abram		398.25	6.50		78.00					482.75	
Archer, Bryce		356.75	28.50	65.50	24.00		0.75	26.50		502.00	
Arellano, Danilo		409.50	25.50	3.00	36.00					474.00	
Arellano, Tanya		403.50		19.00	53.50					476.00	
Atkins, Joan		454.00			16.00			16.00		486.00	
Atkinson, Annmarie		461.60								461.60	
Ayala, Kristi		471.50								471.50	
Baird, Tami		367.00								367.00	
Baker, Lionel		225.00			24.00					249.00	
Ball, Jerri		397.50	39.00		73.50					510.00	
Ballard, Charlie		440.00			32.00					472.00	
Banks, Ambrose		458.00		1.50	12.50					472.00	
Barnett, Tameka		456.00			16.00					472.00	
Barr, Jack		393.50								393.50	
Barrera, Tricia	18.50	152.50			8.00		5.00			184.00	
Barron, Valarie		408.00		1.50	16.00		48.00			473.50	
Barry, Colby		134.25	232.50	31.50	64.00		20.75			483.00	
Barton, Sam	25.00	110.50			40.00	16.75				192.25	
Bates, Clint		456.50				36.00				492.50	
Bauer, Lacy		232.00	1.00		13.00	8.00		97.00		351.00	
Bautista, Damion		360.00								360.00	
Baxter, Zack		456.00			16.00					472.00	
Beasley, Joni		2.00	304.00	20.00	8.00		19.00	15.00		368.00	
Beck, Rae		456.00				16.00				472.00	
Bell, Deshawn		126.50	9.50		33.00					169.00	
Beltran, Arlie		456.00				16.00				472.00	
Beltran, Elvira		22.00			45.00	232.00		88.00		387.00	
Benitez, Flora		464.00								464.00	
Bennett, Wiley		444.00								444.00	
Bentley, Eric		431.50	5.50	117.50	40.00			5.50	600.00		
Total		3,763.75	149,490.15	5,971.50	2,705.00	952.50	7,605.75	1,281.25	3,590.50	2,636.75	177,997.15

Utilization

Page 1

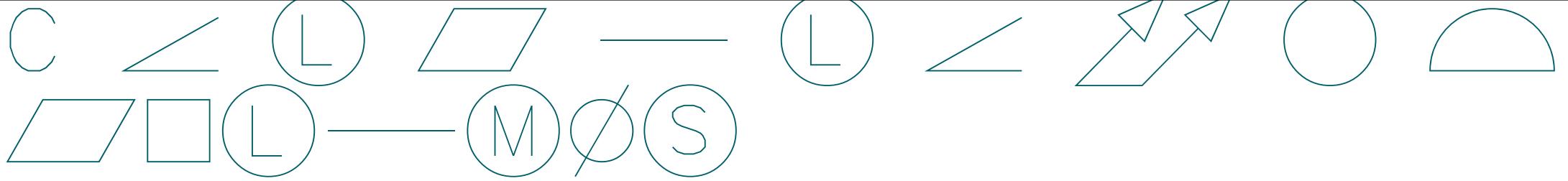
Visualizations: Build visual, Filters, Rows, Columns, Values, Drill through, Cross-report, Keep all filters, Add drill-through fields here.

Fields: Search, Budgets and Forecasts, Calendar, Hours (Category checked), Date, Division, EmployeeID, HourlyCost, HourlyRate, Hours (checked), JobID, PayType, PeriodEndDate, PeriodStartDate, TaskID, TimesheetBatchID, TimesheetID, TotalHours, TotalHoursBilled, People.

Keep all filters: On

Page 2 of 2

82%



Column = SUM([Hours])

Column 2 = [Hours]/[TotalHours]

Practice 3 - Power BI Desktop

Vien Nguyen

Table tools

Name: Hours

Mark as date table ▾ Manage relationships New measure Quick New column

Structure Calendars Relationships Calculations

Why?

1 Column = SUM([Hours])

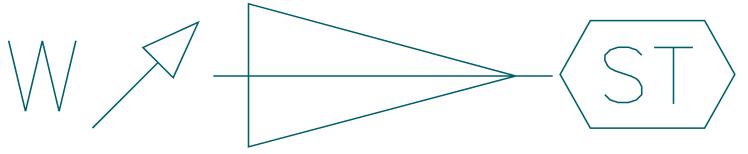
HourlyCost	HourlyRate	TaskID	JobID	Division	PeriodStartDate	PeriodEndDate	TotalHoursBilled	TimesheetBatchID	TimesheetID	TotalHours	PayType	Category	Column	
8	\$77.088	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165152	40	SALARY	Billable	177997.15
8	\$77.088	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165152	40	SALARY	Billable	177997.15
8	\$77.088	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165152	40	SALARY	Billable	177997.15
8	\$77.088	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165152	40	SALARY	Billable	177997.15
8	\$77.088	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165152	40	SALARY	Billable	177997.15
8	\$77.088	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165152	40	SALARY	Billable	177997.15
8	\$72.292	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165117	40	SALARY	Billable	177997.15
8	\$72.292	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165117	40	SALARY	Billable	177997.15
8	\$72.292	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165117	40	SALARY	Billable	177997.15
8	\$72.292	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165117	40	SALARY	Billable	177997.15
8	\$72.292	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165117	40	SALARY	Billable	177997.15
8	\$72.292	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165117	40	SALARY	Billable	177997.15
8	\$69.685	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165154	40	SALARY	Billable	177997.15
8	\$69.685	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165154	40	SALARY	Billable	177997.15
8	\$74.91	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165111	40	SALARY	Billable	177997.15
8	\$74.91	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165111	40	SALARY	Billable	177997.15
8	\$74.91	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165111	40	SALARY	Billable	177997.15
8	\$74.91	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165111	40	SALARY	Billable	177997.15
8	\$74.91	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165111	40	SALARY	Billable	177997.15
8	\$74.91	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165111	40	SALARY	Billable	177997.15
8	\$74.91	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165111	40	SALARY	Billable	177997.15
8	\$76.637	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165263	40	SALARY	Billable	177997.15
8	\$76.637	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165263	40	SALARY	Billable	177997.15
8	\$76.637	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165263	40	SALARY	Billable	177997.15
8	\$76.637	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165263	40	SALARY	Billable	177997.15
8	\$76.637	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165263	40	SALARY	Billable	177997.15
8	\$78.397	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165128	40	SALARY	Billable	177997.15
8	\$78.397	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165128	40	SALARY	Billable	177997.15
8	\$74.041	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165227	40	SALARY	Billable	177997.15
8	\$74.041	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165227	40	SALARY	Billable	177997.15
8	\$74.041	\$120	1001TM	CLEVO003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165227	40	SALARY	Billable	177997.15

Fields

Search: Hours

- Budgets and Forecasts
- Calendar
- Hours
- Category
- Column
- Date
- Division
- EmployeeID
- HourlyCost
- HourlyRate
- Hours
- JobID
- PayType
- PeriodEndDate
- PeriodStartDate
- TaskID
- TimesheetBatchID
- TimesheetID
- TotalHours
- TotalHoursBilled
- People

Table: Hours (27,869 rows) Column: Column (1 distinct values)



- To understand what is going on here, it is important to understand the concept of evaluation context. Evaluation context is the context in which a DAX formula evaluates a calculation.
- There are two types of evaluation contexts in DAX: row and filter contexts. Calculated columns, by default, evaluate within what is called row context. Row context means that when a DAX calculation evaluates, it only considers the current row, and nothing else.
- However, certain functions, such as SUM, override this default context. In this case, the SUM function is overriding the row context of the calculated column and imposing a filter context of the entire table. Hence, the output for each of the rows is the sum of all of the hours in the entire table. All of the standard aggregation functions—that is, SUM, AVERAGE, MAXIMUM, MINIMUM, COUNT, and so on—exhibit this behavior.

Practice 3 - Power BI Desktop

File Home Help Table tools Column tools

Name: Column 2 Format: General Summarization: Sum Data category: Uncategorized Sort by column: Sort Data groups: Groups Manage relationships: Relationships New column: Calculations

Structure: Column 2 = [Hours]/[TotalHours]

Formatting: \$ % Auto

Properties: Sort: Sort ascending, Sort descending, Clear sort, Clear filter, Clear all filters, Number filters

Sort: Sort ascending, Sort descending, Clear sort, Clear filter, Clear all filters, Number filters

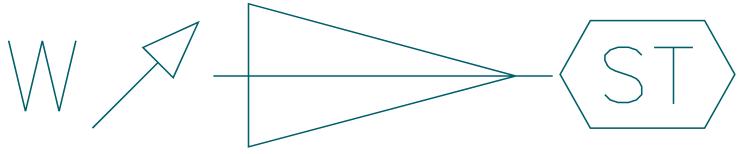
Search: Search

Fields: Search, Budgets and Forecasts, Calendar, Hours, Category, Column, Column 2, Date, Division, EmployeeID, HourlyCost, HourlyRate, Hours, JobID, PayType, PeriodEndDate, PeriodStartDate, TaskID, TimesheetBatchID, TimesheetID, TotalHours, TotalHoursBilled

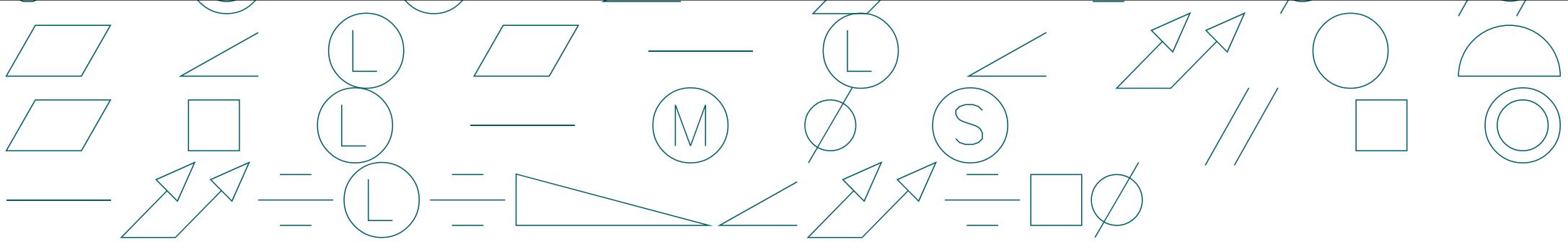
Why?

HourlyRate	TaskID	JobID	Division	PeriodStartDate	PeriodEndDate	TotalHoursBilled	TimesheetBatchID	TimesheetID	TotalHours
7.088	\$120	1001TM	CLEV003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165152
7.088	\$120	1001TM	CLEV003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165152
7.088	\$120	1001TM	CLEV003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165152
7.088	\$120	1001TM	CLEV003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165152
7.088	\$120	1001TM	CLEV003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165152
2.292	\$120	1001TM	CLEV003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165117
2.292	\$120	1001TM	CLEV003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165117
2.292	\$120	1001TM	CLEV003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165117
2.292	\$120	1001TM	CLEV003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165117
2.292	\$120	1001TM	CLEV003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165117
9.685	\$120	1001TM	CLEV003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165154
9.685	\$120	1001TM	CLEV003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165154
74.91	\$120	1001TM	CLEV003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165111
74.91	\$120	1001TM	CLEV003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165111
74.91	\$120	1001TM	CLEV003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165111
74.91	\$120	1001TM	CLEV003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165111
74.91	\$120	1001TM	CLEV003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165111
6.637	\$120	1001TM	CLEV003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165263
6.637	\$120	1001TM	CLEV003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165263
6.637	\$120	1001TM	CLEV003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165263
6.637	\$120	1001TM	CLEV003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165263
8.397	\$120	1001TM	CLEV003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165128
8.397	\$120	1001TM	CLEV003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165128
4.041	\$120	1001TM	CLEV003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165227
4.041	\$120	1001TM	CLEV003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165227
4.041	\$120	1001TM	CLEV003201	1001 Technology	Saturday, March 2, 2019	Friday, March 8, 2019	40	WB030219	TMS165227

Table: Hours (27,869 rows) Column: Column 2 (1,680 distinct values)



- In this calculation, the TotalHours column contains the total number of hours that were reported during the entire reporting period specified by the PeriodStartDate and PeriodEndDate columns. Because no functions have been used that change the evaluation context from row to filter, only the current row is considered during the evaluation of the formula.
- Hence, this formula simply divides the Hours column in each row by the TotalHours column in each row. Now, if we click on the drop-down arrow for **Column 2**, we will see that there are many different values.



- For the purposes of calculating utilization, we want the total hours billed by an employee divided by the total hours reported by an employee.
- In order to get the first part of this - that is, the total hours billed - we need to create a third calculated column using the following formula:
- Use *Shift + Enter* and the *Tab* key to create the correct formatting. *Shift + Enter* inserts a new line or break, while *Tab* indents text that's entered.

Column 3 =

`SUMX(`

`FILTER(`

`ALL('Hours'),`

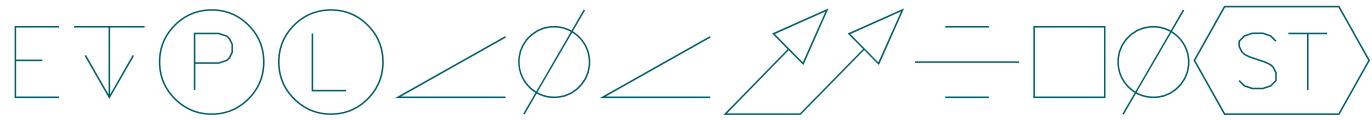
`[Category] = "Billable" && [EmployeeID] =`

`EARLIER([EmployeeID])`

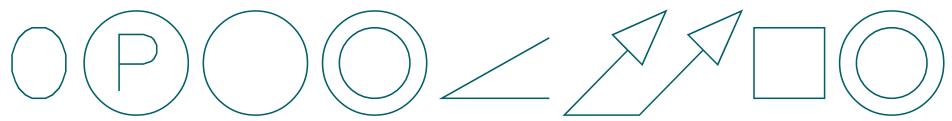
`),`

`[Hours]`

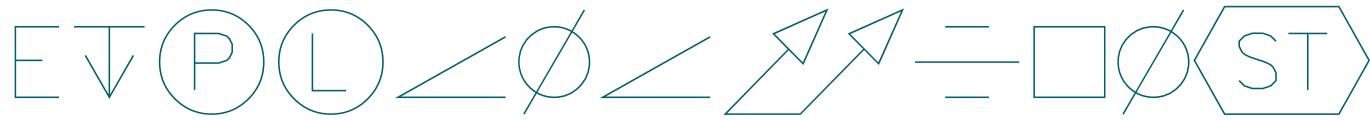
`)`



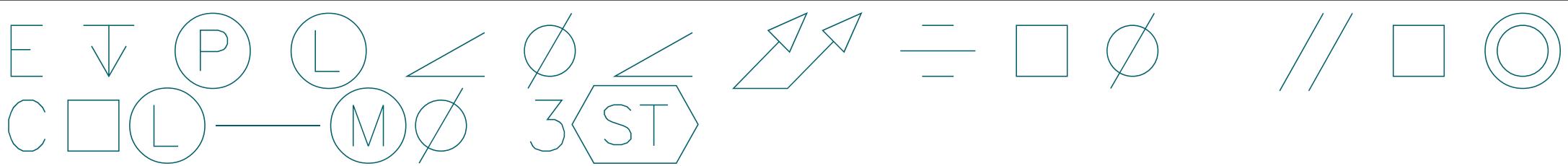
- By way of explanation, the SUMX function takes two parameters: **a table and a numeric column**. These parameters are separated by a comma (,). The SUMX function iterates over the specified table and sums the specified column. Hence, in the preceding formula, the first parameter is the FILTER function (*line 3*), while the second parameter is simply the Hours column (*line 7*).
- The FILTER function itself takes two parameters. The first parameter is the 'Hours' table, *on line 4*, and a filter, *on line 5*. The ALL function returns an entire table stripped of all of its row and filter contexts. Although technically not required in this context, the ALL function ensures that we are operating against all of the rows in the table. The second parameter is our applied filter. This filter actually contains two parts joined by logical AND criteria, &&. **In DAX, && is a logical AND operator, while || is a logical OR operator**. Hence, the first part ensures that the table that's returned by the FILTER function only contains rows that have a **Category** value of **Billable**. The second part ensures that the table that's returned by the FILTER function also only contains rows for the employee in the current row. To do this, this part of the filter uses the EARLIER function.



Operator category	samples	operation
Arithmetic	+	adds two numbers
	*	multiplies two numbers
	/	divides
Comparison	=	equal
	<>	not equal
	=>	greater than or equal
Text concatenation	&	concatenate texts
Logical	&&	AND
		OR
	IN	if the value is IN the list



- The EARLIER function is perhaps the worst named function in all of DAX. The logic behind the naming of this function makes some amount of sense if you are intimately familiar with the DAX evaluation context but generally makes no sense at all to the common layman. You should think of EARLIER as being the current row value for the column specified. Hence, I like to think of EARLIER as the current value. Another construct that can do the same thing is the use of variables or VAR statements. We will find out more about using DAX variables a little later.
- To understand what's going on here, we know that the default context of a calculated column is the row's context. However, because we are using functions such as SUMX and ALL, we have overridden this default row context and imposed a filter context. Hence, to ensure that the table that's returned only includes rows for the employee in our current row, we need to revert to the earlier evaluation context—that is, the row context. This is the purpose of the EARLIER function. Within the EARLIER function, the row context is reestablished, and hence EARLIER([EmployeeID]) returns the value of EmployeeID in the current row.



- Therefore, the **Column 3** formula can be read in the following way:
 1. Take the entire table (ALL) and filter this table down so that it only has rows where the **Category** column has a value of **Billable** and rows where the EmployeeID column matches the current row value for EmployeeID.
 2. Second, sum the Hours column for this table and return the result. Each row for an employee should return the exact same numeric value, which is the total hours that were reported by that employee that were billable hours.
 3. For the second part—that is, the total number of hours reported by an employee— we simply need to create another new column with a slight modification to remove the filter for only billable hours. This column can be written with the following formula:

Column 4 =

`SUMX(`

`FILTER(`

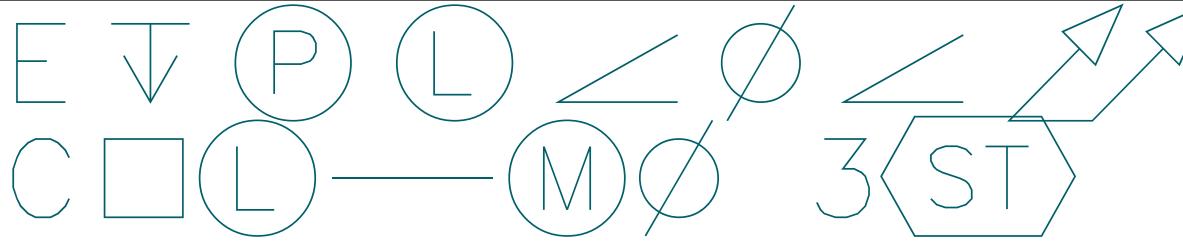
`ALL('Hours'),`

`[EmployeeID] = EARLIER([EmployeeID])`

`),`

`[Hours]`

`)`



- Finally, to create our utilization calculation, create a new calculated column that utilizes row context, as follows:

Column 5 = DIVIDE([Column 3], [Column 4],0)

- While the preceding formula could have been written as Column 5 = [Column 3] / [Column 4], it is a good idea to use the DIVIDE function instead of the divide (/) operator. The DIVIDE function takes a third parameter, which allows for the return of an alternate number instead of an error in the event of an attempt to divide by zero.
- We can now select **Column 5** by clicking on its header and change its formatting to be a percentage. To change the column to a percentage, we need to go to the **Formatting** section of the **Column tools** tab in the ribbon. From here, select the **%** icon.

Search

Column	Column 2	Column 3	Column 4	Column 5
177997.15	0.2	456.5	492.5	92.69%
177997.15	0.2	456.5	492.5	92.69%
177997.15	0.2	456.5	492.5	92.69%
177997.15	0.2	456.5	492.5	92.69%
177997.15	0.2	456.5	492.5	92.69%
177997.15	0.2	464	472	98.31%
177997.15	0.2	464	472	98.31%
177997.15	0.2	464	472	98.31%
177997.15	0.2	464	472	98.31%
177997.15	0.2	416	472	88.14%
177997.15	0.2	416	472	88.14%
177997.15	0.2	460	478	96.23%
177997.15	0.2	460	478	96.23%
177997.15	0.2	460	478	96.23%
177997.15	0.2	460	478	96.23%
177997.15	0.2	464	472	98.31%
177997.15	0.2	464	472	98.31%
177997.15	0.2	464	472	98.31%
177997.15	0.2	464	472	98.31%
177997.15	0.2	464	472	98.31%
177997.15	0.2	440	472	93.22%
177997.15	0.2	440	472	93.22%
177997.15	0.2	464	472	98.31%
177997.15	0.2	464	472	98.31%
177997.15	0.2	464	472	98.31%
177997.15	0.2	464	472	98.31%
177997.15	0.2	440	472	93.22%

Search

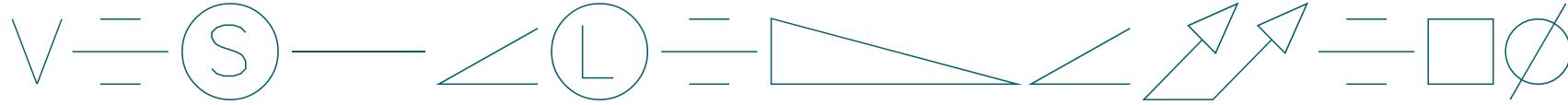
- > Budgets and Forecasts
- > Calendar
- > Hours
 - Category
 - Column
 - Column 2
 - Column 3
 - Column 4
 - Column 5
- Date
- Division
- EmployeeID
- HourlyCost
- HourlyRate
- Hours
- JobID
- PayType
- PeriodEndDate
- PeriodStartDate
- TaskID
- TimesheetBatchID
- TimesheetID
- TotalHours
- TotalHoursBilled
- People

C L O ↗ P

1. Right-click the header for **Column** and choose **Delete**.
2. Confirm the deletion of this column in the **Delete column** dialog box by clicking **Delete**.
3. Repeat this procedure for **Column 2**.
4. Double-click the TotalHours column and rename this column TotalHoursInPeriod.
5. Rename **Column 3** TotalBillableHours, **Column 4** TotalHours, and **Column 5** % Utilization.

The screenshot shows a Microsoft Power BI Data Editor interface. On the left is a table with six columns. The columns are: TotalHoursInPeriod, PayType, Category, TotalBillableHours, TotalHours, and Utilization. The Utilization column has a yellow background. On the right side of the screen is a navigation pane with several items listed:

- Search
- Budgets and Forecasts
- Calendar
- Hours
 - Category
 - Date
 - Division
 - EmployeeID
 - HourlyCost
 - HourlyRate
 - Hours
 - JobID
 - PayType
 - PeriodEndDate
 - PeriodStartDate
 - TaskID
 - TimesheetBatchID
 - TimesheetID
 - TotalBillableHours
 - TotalHours
 - TotalHoursBilled
 - TotalHoursInPeriod
 - Utilization
- People



1. Click on the **Report** view in the **Views** bar.
2. Shrink the existing matrix visualization horizontally to create space on the report canvas.
3. Click on an empty portion of the report canvas to deselect the existing table.
4. Click **Name** from the **People** table to create a new table visualization on the blank area of the report canvas.
5. With this new table selected, click on **TotalBillableHours**, **TotalHours**, and **% Utilization** from the **Hours** table to add these to this second table visualization.

Name	TotalBillableHours	TotalHours	Utilization
Adkins, Rita	26,912.00	26,912.00	5800.00%
Allen, Susana	1,440.00	1,440.00	2000.00%
Anderson, Pearlie	36,860.00	49,875.00	7020.95%
Andrews, Abram	26,682.75	32,344.25	5527.24%
Archer, Bryce	69,209.50	97,388.00	13786.75%
Arellano, Danilo	28,255.50	32,706.00	5961.08%
Arellano, Tanya	29,859.00	35,224.00	6272.90%
Atkins, Joan	31,780.00	34,020.00	6539.09%
Atkinson, Annmarie	25,849.60	25,849.60	5600.00%
Ayala, Kristi	29,704.50	29,704.50	6300.00%
Baird, Tami	16,882.00	16,882.00	4600.00%
Baker, Lionel	12,825.00	14,193.00	5150.60%
Total	9,833,628.00	12,502,645.75	2199848.33%

Why?



- Note that the values that are returned in this second table are really large! This is because the default aggregation for numeric fields is **Sum**. Since all of the rows for each employee contain our desired number, summing these values doesn't make sense. To correct this, do the following:
 1. In the **Values** field well of the **Visualizations** pane, select the drop-down arrow next to **TotalBillableHours** and choose **Average** instead of **Sum** as the aggregation.
 2. Repeat this procedure for **TotalHours** and **% Utilization**.

The screenshot shows the Power BI 'Visualizations' pane with the 'Values' field well open. The 'TotalBillableHours' field has a red box around it, indicating it is selected. The 'Aggregation' dropdown menu is open, showing options like Sum, Average, Minimum, Maximum, Count (Distinct), Count, Standard deviation, Variance, and Median. The 'Average' option is highlighted with a red box. Below the field well, there are sections for 'Columns', 'Drill through', 'Cross-report', and 'Keep all filters'. A 'New quick measure' section is also visible on the right.

Remove field
Rename for this visual
Move
Add a sparkline
Conditional formatting
Remove conditional formatting
Don't summarize
✓ Sum
Average
Minimum
Maximum
Count (Distinct)
Count
Standard deviation
Variance
Median
Show value as
Name
TotalBillableHours
TotalHours
Utilization
New quick measure
PeriodStartDate
TaskID
TimesheetBatchID
TimesheetID
TotalBillableHours
TotalHours
 \sum TotalHoursBilled
 \sum TotalHoursInPeriod
Utilization



- We did not create a relationship for the **Budgets and Forecasts** table. The main issue here is that the **Budgets and Forecasts** table has a data **granularity** of month while our **Hours** and **Calendar** tables have a data granularity of day. However, we can resolve this data granularity issue through the use of calculated columns as well as a calculated table.
- To resolve the data granularity issue, start by creating the following calculated column in the **Budgets and Forecasts** table:

`MonthYear = [Month] & "2017"`

- Here, we use DAX's & concatenation operator to concatenate each row's Month column value with the text value 2017 since the **Budgets and Forecasts** data is for the year **2017**.



- Similarly, we can create a calculated column in the **Calendar** table using the following formula:

`MonthYear = LEFT([Month], 3) & [Year]`

- We can now use the MonthYear columns in the **Budgets and Forecasts** table and the **Calendar** table to create a relationship between the tables. However, if we do that, we receive a warning when creating the relationship:

“This relationship has cardinality Many-Many. This should only be used if it is expected that neither column (MonthYear and MonthYear) contains unique values, and that the significantly different behavior of Many-many relationships is understood.”



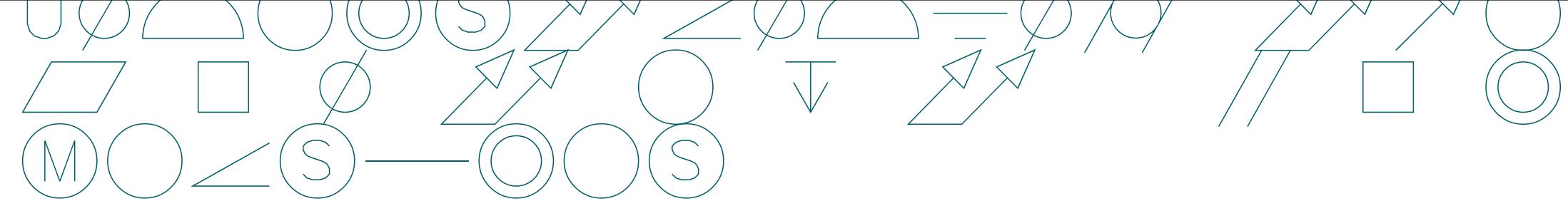
- Since the **Budgets and Forecasts** table contains a budget for each city for each month, there are multiple rows with duplicate MonthYear values. The same is true for the **Calendar** table. While relationships with **Many-Many** cardinality are allowed in Power BI since the July 2018 release of Power BI Desktop, they are often best avoided. To resolve the **Many-Many** relationship issue, perform the following steps:

1. Click **Cancel** to not create the relationship or otherwise delete the **Many-Many** relationship between the **Budgets and Forecast** table and **Calendar** table if already created.
2. While in the **Data** view, choose the **Table tools** tab in the ribbon and then select **New table** from the **Calculations** section.
3. Enter the following formula: `MonthYears = DISTINCT('Calendar'[MonthYear])`
4. Switch to the **Model** view and create a relationship between the MonthYear column in the **MonthYears** table and the MonthYear column in the **Calendar** table.
5. Create a relationship between the MonthYear column in the **MonthYears** table and the MonthYear column in the **Budgets and Forecasts** table.

Note that both of the relationships have a **Cardinality** value of **Many-to-one (*:1)** with the **MonthYears** table on the **1** side of the relationships.



- We observed implicit measures when we created our table visualizations in the previous section, *Calculated columns*. When we placed the TotalBillableHours, TotalHours, and % Utilization columns in our table, the values were calculated via an **implicit Sum measure** that we changed to Average.
- Unlike calculated columns, measures are **dynamic** in their calculation so that measures are calculated just in time, every time there is an interaction with the end user. Measures are not recalculated during data refresh but rather every time a user views or interacts with a report. This makes measures truly dynamic, and hence they are perfect for a scenario where end users may change the evaluation context on the fly.



A screenshot of a Microsoft Power BI interface showing a data visualization and a context menu.

The visualization displays a list of names and their corresponding values:

Name	Value 1	Value 2
Bell, Deshawn	126.50	9.50
Beltran, Arlie		
Beltran, Elvira	456.00	
Benitez, Flora		
Bennett, Wiley		
Bentley, Eric		
Total	3,76	117 50
		2,705.00

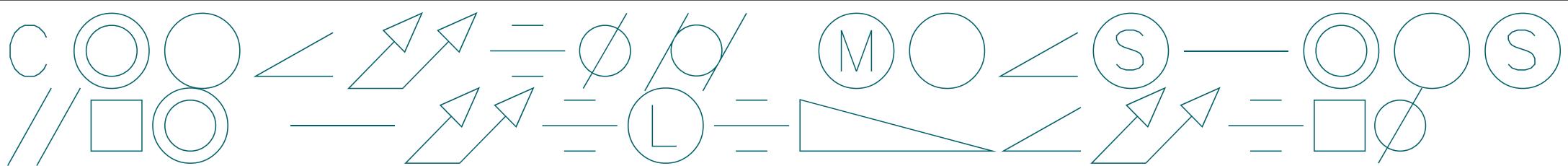
The context menu, which is open over the 'Total' row, contains the following items:

- New measure
- Duplicate Page (highlighted with a red box)
- Rename Page
- Delete Page
- Hide Page
- Dept
- Location
- Month
- MonthYear
- Type
- Value
- Date
- IsWorkDay
- Month
- MonthNum
- MonthYear
- Weekday
- WeekdayNum
- WeekNum
- Year
- Hours
- % Utilization

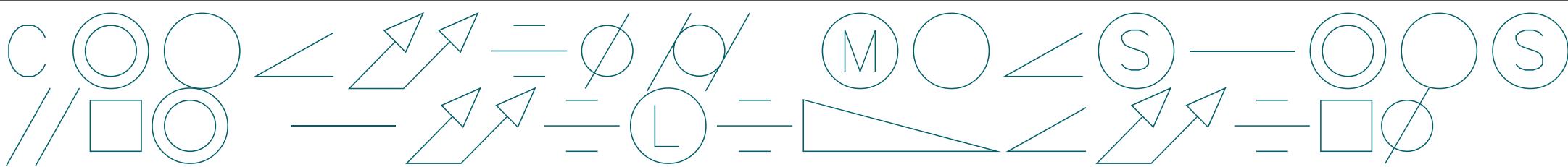
Measure =
 $\text{SUMX}(\text{FILTER}(\text{'Hours'}, \text{'Hours'}[\text{Category}] = "Billable"), \text{'Hours'}[\text{Hours}])$

Month	Name	Average of TotalBillableHours	Average of TotalHours	Average of % Utilization	Measure
<input type="checkbox"/> April	Adkins, Rita	464.00	464.00	100.00%	176.00
<input type="checkbox"/> August	Allen, Susana	72.00	72.00	100.00%	4.00
<input type="checkbox"/> December	Anderson, Pearlie	388.00	525.00	73.90%	164.00
<input type="checkbox"/> February	Andrews, Abram	398.25	482.75	82.50%	154.50
<input checked="" type="checkbox"/> January	Archer, Bryce	356.75	502.00	71.07%	120.75
<input type="checkbox"/> July	Arellano, Danilo	409.50	474.00	86.39%	148.50
<input type="checkbox"/> June	Arellano, Tanya	403.50	476.00	84.77%	150.00
<input type="checkbox"/> March	Atkins, Joan	454.00	486.00	93.42%	180.00
<input type="checkbox"/> May	Atkinson, Annmarie	461.60	461.60	100.00%	173.50
<input type="checkbox"/> November	Ayala, Kristi	471.50	471.50	100.00%	182.75
<input type="checkbox"/> October	Baird, Tami	367.00	367.00	100.00%	168.00
<input type="checkbox"/> September	Baker, Lionel	225.00	249.00	90.36%	23.00
	Ball, Jerri	397.50	510.00	77.94%	168.00
	Ballard, Charlie	440.00	472.00	93.22%	176.00
	Banks, Ambrose	458.00	472.00	97.03%	175.00
	Barnett, Tameka	456.00	472.00	96.61%	176.00
	Barr, Jack	393.50	393.50	100.00%	177.00
	Barrera, Tricia	152.50	184.00	82.88%	152.50
	Barron, Valarie	408.00	473.50	86.17%	136.00
	Barry, Colby	134.25	483.00	27.80%	32.25
	Barton, Sam	110.50	192.25	57.48%	110.50
	Bates, Clint	456.50	492.50	92.69%	168.50
	Bauer, Lacy	232.00	351.00	66.10%	88.00
	Bautista, Damion	360.00	360.00	100.00%	72.00
	Baxter, Zack	456.00	472.00	96.61%	176.00
	Beasley, Joni	2.00	368.00	0.54%	2.00
	Beck, Rae	456.00	472.00	96.61%	168.00
	Bell, Deshawn	126.50	169.00	74.85%	126.50
	Beltran, Arlie	456.00	472.00	96.61%	168.00
	Beltran, Elvira	22.00	387.00	5.68%	13.00
	Benitez, Flora	464.00	464.00	100.00%	176.00
	Total	360.24	449.79	80.40%	55,533.23





The screenshot shows the Power BI Query Settings dialog box. The 'Properties' section is highlighted with a red box, showing the 'Name' field set to 'Calculations'. The 'Applied Steps' section is also highlighted with a red box, showing a single step named 'Source'. On the left, the 'Common data sources' list includes 'Excel workbook', 'Power BI datasets', 'Power BI dataflows', 'Dataverse', 'SQL Server', 'Analysis Services', 'Text/CSV', 'Web', 'OData feed', and 'Blank query'. The 'Blank query' option is highlighted with a red box. On the right, a context menu is open for the 'Calculations' item, listing options like Copy, Paste, Delete, Rename, Enable load (which is checked and highlighted with a red box), Include in report refresh, Duplicate, Reference, and Move To Group.



File Home Insert Modeling View Help **Form Fields >**

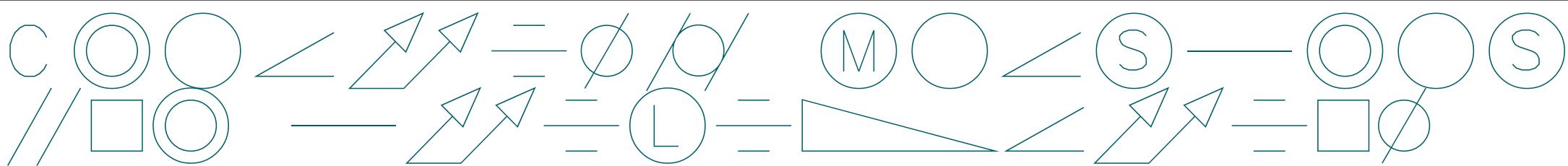
Name **Measure** Format General \$ % , .
Home table Calculations \$ % , .
Structure Formatting

Search Calculations
Measure ...
Budgets and Forecasts
Calendar
Hours

Search Calculations
Total Billable Hours
Budgets and Forecasts
Calendar

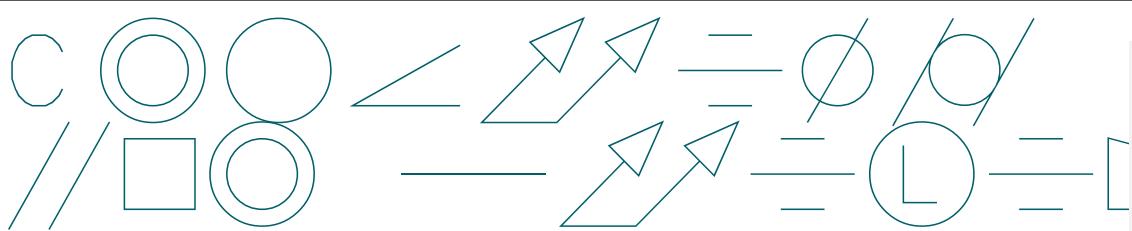
Measure =

```
1 Measure =
2 SUMX(
3   FILTER(
4     'Hours',
5     'Hours'[Category] = "Billable"
6   ),
7   'Hours'[Hours]
8 )
```



- We need to create a measure for the denominator of our utilization calculation—that is, the total potential billable hours available within a period. We know that we cannot simply use the total hours reported, though, and that we must calculate the number of potential billable hours for any period of time in a different way.
 - To do this, we must have an independent way of calculating the number of hours and thus utilization within any arbitrary period of days.

Name	WorkHours	Format	Whole number	Summarization	Sum	Sort by column				
Data type	Whole number	\$ % , .	0	Data category	Uncategorized	Data groups				
Structure			Formatting			Properties		Sort	Groups	Relations
X	✓	1 WorkHours = IF([IsWorkDay],8,0)								
Date	Month	Year	MonthNum	WeekNum	Weekday	WeekdayNum	IsWorkDay	MonthYear	WorkHours	
1/1/2017 12:00:00 AM	January	2017	1	1	Sunday	7	0	Jan2017	0	
1/2/2017 12:00:00 AM	January	2017	1	1	Monday	1	1	Jan2017	8	
1/3/2017 12:00:00 AM	January	2017	1	1	Tuesday	2	1	Jan2017	8	
1/4/2017 12:00:00 AM	January	2017	1	1	Wednesday	3	1	Jan2017	8	



File Home Help Table tools **Measure tools**

Name Measure Format Whole number

Home table Calculations \$ % Auto

Structure Formatting

1 Total Hours = SUM('Calendar'[WorkHours])

File Home Help Table tools **Measure tools**

Name % Utilization Format General

Home table Calculations \$ % Auto

Structure Formatting

1 % Utilization = DIVIDE([Total Billable Hours],[Total Hours],0)

Name % Utilization Format Percentage

Home table Calculations \$ %

Structure

1 % Utilization = DIVIDE([Total Billable Hours],[Total Hours],0)

<input type="checkbox"/>	Adkins, Rita
<input type="checkbox"/>	Allen, Susana
<input type="checkbox"/>	Anderson, Pearlie
<input type="checkbox"/>	Andrews, Abram
<input type="checkbox"/>	Archer, Bryce
<input type="checkbox"/>	Arellano, Danilo
<input type="checkbox"/>	Arellano, Tanya
<input type="checkbox"/>	Atkins, Joan
<input type="checkbox"/>	Atkinson, Annmarie
<input type="checkbox"/>	Ayala, Kristi
<input type="checkbox"/>	Baird, Tami
<input type="checkbox"/>	Baker, Lionel
<input type="checkbox"/>	Ball, Jerri
<input type="checkbox"/>	Ballard, Charlie
<input type="checkbox"/>	Banks, Ambrose
<input type="checkbox"/>	Barnett, Tameka
<input type="checkbox"/>	Barr, Jack
<input type="checkbox"/>	Barrera, Tricia
<input type="checkbox"/>	Barron, Valarie
<input type="checkbox"/>	Barry, Colby
<input type="checkbox"/>	Barton, Sam
<input type="checkbox"/>	Bates, Clint
<input type="checkbox"/>	Bauer, Lacy
<input type="checkbox"/>	Bautista, Damion
<input type="checkbox"/>	Baxter, Zack
<input type="checkbox"/>	Beasley, Joni
<input type="checkbox"/>	Beck, Rae
<input type="checkbox"/>	Bell, Deshawn
<input type="checkbox"/>	Beltran, Arlie
<input type="checkbox"/>	Beltran, Elvira
<input type="checkbox"/>	Benitez, Flora
<input type="checkbox"/>	Total

Name	Average of TotalBillableHours	Average of TotalHours	Average of % Utilization	Total Billable Hours	Total Hours	%
Abbott, Eugenio				6256		
Abbott, Isabelle				6256		
Abbott, Pamela				6256		
Acevedo, Dianne				6256		
Acevedo, Laverne				6256		
Acevedo, Randy				6256		
Acosta, Angeline				6256		
Acosta, Jaime				6256		
Acosta, Rosalyn				6256		
Adams, Joan				6256		
Adams, Kurtis				6256		
Adams, Marcy				6256		
Adkins, Devon				6256		
Adkins, Rita	464.00	464.00	100.00%	464.00	6256	7.42%
Aguilar, Jonah				6256		
Aguilar, Mable				6256		
Aguilar, Royce				6256		
Aguirre, Burt				6256		
Aguirre, Jami				6256		
Aguirre, Thad				6256		
Alexander, Buck				6256		
Alexander, Darlene				6256		
Alexander, Teresa				6256		
Ali, Antonia				6256		
Ali, Hannah				6256		
Ali, Lenard				6256		
Allen, Karen				6256		
Allen, Nellie				6256		
Allen, Susana	72.00	72.00	100.00%	72.00	6256	1.15%
Allison, Allison				6256		
Allison, Gina				6256		
Total	359.44	448.62	80.41%	149,490.15	6256	2389.55%

Error?

Visualizations

Build visual

Filters

Calculations

- % Utilization
- Total Billable Hours
- Total Hours

Budgets and Forecasts

Calendar

Date

IsWorkDay

Month

MonthNum

MonthYear

Weekday

WeekdayNum

WeekNum

WorkHours

Year

Hours

MonthYears

People

Columns

Name

Average of TotalBillab...

Average of TotalHours

Average of % Utilizati...

Total Billable Hours

Total Hours

% Utilization

Drill through

Cross-report

Off



Edit relationship

Select tables and columns that are related.

Hours

EmployeeID	Date	Hours	HourlyCost	HourlyRate	TaskID	JobID	Division
CBATES	Monday, January 7, 2019	8	\$77.088	120	1001TM	CLEV003201	1001 Technol
CBATES	Thursday, January 10, 2019	8	\$77.088	120	1001TM	CLEV003201	1001 Technol
CBATES	Friday, January 11, 2019	8	\$77.088	120	1001TM	CLEV003201	1001 Technol



Calendar

Date	Month	Year	MonthNum	WeekNum	Weekday	WeekdayNum	IsWorkDay
1/1/2017 12:00:00 AM	January	2017	1	1	Sunday	7	0
1/2/2017 12:00:00 AM	January	2017	1	1	Monday	1	1
1/3/2017 12:00:00 AM	January	2017	1	1	Tuesday	2	1



Cardinality

Many to one (*:1)

Make this relationship active

Assume referential integrity

Cross filter direction

Both

Apply security filter in both directions

OK

Cancel

Name	Average of TotalBillableHours	Average of TotalHours	Average of % Utilization	Total Billable Hours	Total Hours by Employee	% U
Adkins, Rita	464.00	464.00	100.00%	464.00	464	100.00%
Allen, Susana	72.00	72.00	100.00%	72.00	160	45.00%
Anderson, Pearlie	388.00	525.00	73.90%	388.00	472	82.20%
Andrews, Abram	398.25	482.75	82.50%	398.25	472	84.38%
Archer, Bryce	356.75	502.00	71.07%	356.75	472	75.58%
Arellano, Danilo	409.50	474.00	86.39%	409.50	472	86.76%
Arellano, Tanya	403.50	476.00	84.77%	403.50	472	85.49%
Atkins, Joan	454.00	486.00	93.42%	454.00	472	96.19%
Atkinson, Annmarie	461.60	461.60	100.00%	461.60	448	103.04%
Ayala, Kristi	471.50	471.50	100.00%	471.50	456	103.40%
Baird, Tami	367.00	367.00	100.00%	367.00	360	101.94%
Baker, Lionel	225.00	249.00	90.36%	225.00	336	66.96%
Ball, Jerri	397.50	510.00	77.94%	397.50	472	84.22%
Ballard, Charlie	440.00	472.00	93.22%	440.00	472	93.22%
Banks, Ambrose	458.00	472.00	97.03%	458.00	472	97.03%
Barnett, Tameka	456.00	472.00	96.61%	456.00	472	96.61%
Barr, Jack	393.50	393.50	100.00%	393.50	392	100.38%
Barrera, Tricia	152.50	184.00	82.88%	152.50	184	82.88%
Barron, Valarie	408.00	473.50	86.17%	408.00	472	86.44%
Barry, Colby	134.25	483.00	27.80%	134.25	472	28.44%
Barton, Sam	110.50	192.25	57.48%	110.50	192	57.55%
Bates, Clint	456.50	492.50	92.69%	456.50	472	96.72%
Bauer, Lacy	232.00	351.00	66.10%	232.00	472	49.15%
Bautista, Damion	360.00	360.00	100.00%	360.00	360	100.00%
Baxter, Zack	456.00	472.00	96.61%	456.00	472	96.61%
Beasley, Joni	2.00	368.00	0.54%	2.00	368	0.54%
Beck, Rae	456.00	472.00	96.61%	456.00	472	96.61%
Bell, Deshawn	126.50	169.00	74.85%	126.50	168	75.30%
Beltran, Arlie	456.00	472.00	96.61%	456.00	472	96.61%
Beltran, Elvira	22.00	387.00	5.68%	22.00	424	5.19%
Benitez, Flora	464.00	464.00	100.00%	464.00	464	100.00%
Total	359.44	448.62	80.41%	149,490.15	6256	2389.55%

Visualizations

Build visual

Filters

Calculations

Renamed

Total Billable Hours

Total Hours by Employee

Budgets and Forecasts

Calendar

Date

IsWorkDay

Month

MonthNum

MonthYear

Weekday

WeekdayNum

WeekNum

WorkHours

Year

Hours

MonthYears

People

Columns

Name

Average of TotalBillab...

Average of TotalHours

Average of % Utilizati...

Total Billable Hours

Total Hours by Emplo...

% Utilization

Drill through

Cross-report

Name: Measure

Home table: Calculations

Structure:

- April
- August
- December
- February
- January
- July
- June
- March
- May

Format: \$% Format: Data category: Uncategorized

Formatting: \$ % , .⁰⁰ Auto

Properties:

```
1 Total Hours =
2 IF(
3   HASONEVALUE(People[Name]),
4   [Total Hours by Employee],
5   SUMX(
6     SUMMARIZE(
7       'People', People[Name],
8       "_Hours", [Total Hours by Employee]
9     ),
10    [_Hours]
11  )
12 )
```

Name: % Utilization

Home table: Calculations

Structure:

- X ✓

Format: \$% Format: Percentage Data category: Unc

Formatting: \$ % , .⁰⁰ 2

Properties:

```
1 % Utilization = DIVIDE([Total Billable Hours],[Total Hours],0)
```

Name	Average of TotalBillableHours	Average of TotalHours	Average of % Utilization	Total Billable Hours	Total Hours by Employee	% Utilization	Total
Adkins, Rita	464.00	464.00	100.00%	464.00	464	100.00%	464
Allen, Susana	72.00	72.00	100.00%	72.00	160	45.00%	160
Anderson, Pearlie	388.00	525.00	73.90%	388.00	472	82.20%	472
Andrews, Abram	398.25	482.75	82.50%	398.25	472	84.38%	472
Archer, Bryce	356.75	502.00	71.07%	356.75	472	75.58%	472
Arellano, Danilo	409.50	474.00	86.39%	409.50	472	86.76%	472
Arellano, Tanya	403.50	476.00	84.77%	403.50	472	85.49%	472
Atkins, Joan	454.00	486.00	93.42%	454.00	472	96.19%	472
Atkinson, Annmarie	461.60	461.60	100.00%	461.60	448	103.04%	448
Ayala, Kristi	471.50	471.50	100.00%	471.50	456	103.40%	456
Baird, Tami	367.00	367.00	100.00%	367.00	360	101.94%	360
Baker, Lionel	225.00	249.00	90.36%	225.00	336	66.96%	336
Ball, Jerri	397.50	510.00	77.94%	397.50	472	84.22%	472
Ballard, Charlie	440.00	472.00	93.22%	440.00	472	93.22%	472
Banks, Ambrose	458.00	472.00	97.03%	458.00	472	97.03%	472
Barnett, Tameka	456.00	472.00	96.61%	456.00	472	96.61%	472
Barr, Jack	393.50	393.50	100.00%	393.50	392	100.38%	392
Barrera, Tricia	152.50	184.00	82.88%	152.50	184	82.88%	184
Barron, Valarie	408.00	473.50	86.17%	408.00	472	86.44%	472
Barry, Colby	134.25	483.00	27.80%	134.25	472	28.44%	472
Barton, Sam	110.50	192.25	57.48%	110.50	192	57.55%	192
Bates, Clint	456.50	492.50	92.69%	456.50	472	96.72%	472
Bauer, Lacy	232.00	351.00	66.10%	232.00	472	49.15%	472
Bautista, Damion	360.00	360.00	100.00%	360.00	360	100.00%	360
Baxter, Zack	456.00	472.00	96.61%	456.00	472	96.61%	472
Beasley, Joni	2.00	368.00	0.54%	2.00	368	0.54%	368
Beck, Rae	456.00	472.00	96.61%	456.00	472	96.61%	472
Bell, Deshawn	126.50	169.00	74.85%	126.50	168	75.30%	168
Beltran, Arlie	456.00	472.00	96.61%	456.00	472	96.61%	472
Beltran, Elvira	22.00	387.00	5.68%	22.00	424	5.19%	424
Benitez, Flora	464.00	464.00	100.00%	464.00	464	100.00%	464
Total	359.44	448.62	80.41%	149,490.15	6256	83.82%	178344

Filters

Visualizations

Build visual

Fields

Search

Calculations

- % Utilization
- Total Billable Hours
- Total Hours
- Total Hours by Employee

Budgets and Forecasts

Calendar

Date

IsWorkDay

Month

MonthNum

MonthYear

Weekday

WeekdayNum

WeekNum

WorkHours

Year

Hours

MonthYears

People

Columns

Name
Average of TotalBillab...
Average of TotalHours
Average of % Utilizati...
Total Billable Hours
Total Hours by Emplo...
% Utilization
Total Hours

Drill through

File Home Insert Modeling View Help Format Data

Name % Utilization Format Percentage \$, 00 2

Home table Calculations

Structure

Formatting

1 % Utilization =
2 VAR __utilization =
3 | DIVIDE([Total Billable Hours],[Total Hours],0)
4 RETURN
5 | __utilization + 0

Acevedo, Laverne

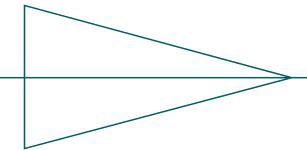
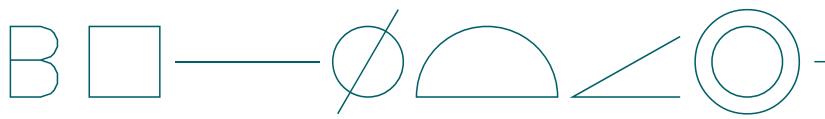
Name % Utilization Format Percentage \$, 00 2

Home table Calculations

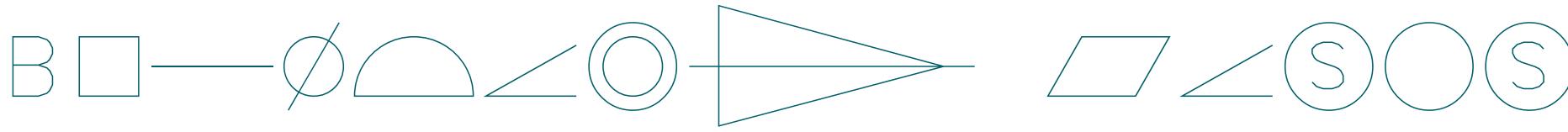
Structure

Formatting

1 % Utilization =
2 VAR __utilization =
3 | DIVIDE([Total Billable Hours],[Total Hours],0)
4 VAR __days = COUNTROWS('Hours')
5 RETURN
6 IF(
7 | ISBLANK(__days),
8 | BLANK(),
9 | __utilization + 0
10)

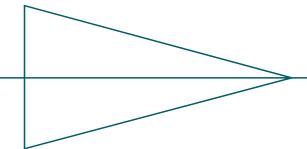


Name	Average of TotalBillableHours	Average of TotalHours	Average of % Utilization	Total Billable Hours	Total Hours by Employee	% Utilization	Total	...
Cole, Camille	14.00	14.00	100.00%	14.00	8	175.00%	8	
Perkins, Caitlin	169.00	169.00	100.00%	169.00	104	162.50%	104	
Lawson, Malik	494.00	494.00	100.00%	494.00	368	134.24%	368	
Moore, Lydia	29.00	29.00	100.00%	29.00	24	120.83%	24	
Shea, Bart	380.00	380.00	100.00%	380.00	320	118.75%	320	
Noble, Joyce	148.00	156.00	94.87%	148.00	128	115.63%	128	
Dominguez, Sheila	524.50	543.00	96.59%	524.50	464	113.04%	464	
Parker, Concetta	9.00	9.00	100.00%	9.00	8	112.50%	8	
Strong, Lela	296.50	296.50	100.00%	296.50	264	112.31%	264	
Riley, Veronica	516.00	516.00	100.00%	516.00	464	111.21%	464	
Owens, Betsy	400.00	400.00	100.00%	400.00	360	111.11%	360	
Fleming, Joyce	515.50	515.50	100.00%	515.50	464	111.10%	464	
Wells, Gail	408.35	409.10	99.82%	408.35	368	110.96%	368	
Lutz, Suzette	509.00	509.00	100.00%	509.00	464	109.70%	464	
Castro, Anne	490.00	490.00	100.00%	490.00	448	109.38%	448	
Jordan, Darrell	490.00	490.00	100.00%	490.00	448	109.38%	448	
Savage, Eunice	515.00	523.00	98.47%	515.00	472	109.11%	472	
Logan, Rosemarie	497.20	497.20	100.00%	497.20	456	109.04%	456	
Parker, Aurora	507.00	519.50	97.59%	507.00	472	107.42%	472	
Owen, Theresa	498.00	498.00	100.00%	498.00	464	107.33%	464	
Reed, Myron	470.00	470.00	100.00%	470.00	440	106.82%	440	
Sanford, Abram	503.50	511.50	98.44%	503.50	472	106.67%	472	
Kemp, Irving	486.00	486.00	100.00%	486.00	456	106.58%	456	
Fletcher, Anne	409.25	421.25	97.15%	409.25	384	106.58%	384	
Johns, Mariana	468.75	495.75	94.55%	468.75	440	106.53%	440	
Lawrence, Charley	502.50	539.50	93.14%	502.50	472	106.46%	472	
Reese, Bruce	450.00	450.00	100.00%	450.00	424	106.13%	424	
Goodwin, Carey	348.00	348.00	100.00%	348.00	328	106.10%	328	
Meyers, Chasity	456.00	456.00	100.00%	456.00	432	105.56%	432	
Houston, Candice	477.50	493.50	96.76%	477.50	456	104.71%	456	
Westerman, Scott	100.00	100.00	100.00%	100.00	96	104.17%	96	
Total	359.44	448.62	80.41%	149,490.15	6256	83.82%	178344	

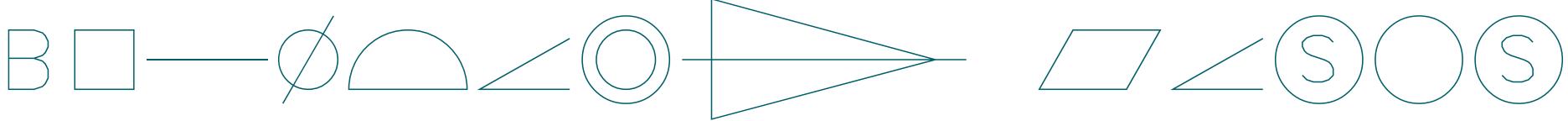


A screenshot of a Microsoft Excel spreadsheet showing employee time entries. The table has columns for EmployeeID, Date, Hours, HourlyCost, HourlyRate, TaskID, JobID, Division, PeriodStartDate, PeriodEndDate, and TotalHoursBilled.

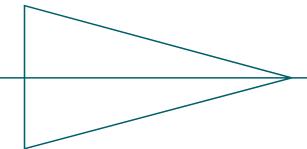
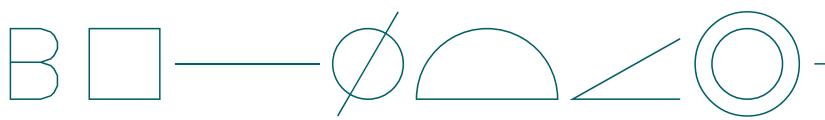
EmployeeID	Date	Hours	HourlyCost	HourlyRate	TaskID	JobID	Division	PeriodStartDate	PeriodEndDate	TotalHoursBilled
CCOLE	Sunday, February 24, 2019	4	\$117.854	145	1001TM	NASH000085	1001 Technology	Saturday, February 23, 2019	Friday, March 1, 2019	4
CCOLE	Friday, February 22, 2019	10	\$117.854	145	1001TM	NASH000085	1001 Technology	Saturday, February 16, 2019	Friday, February 22, 2019	10



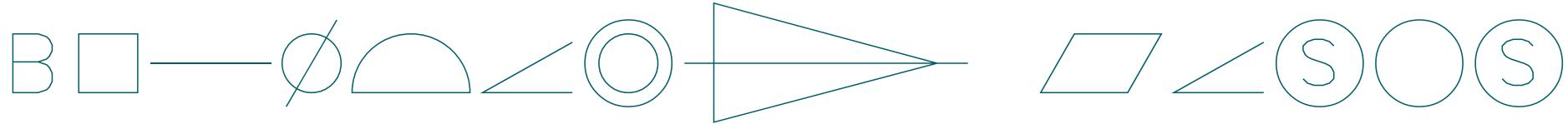
Name	Average of TotalBillableHours	Average of TotalHours	Average of % Utilization	Total Billable Hours	Total Hours by Employee	% Utilization	Total	...
Cole, Camille	14.00	14.00	100.00%	14.00	8	175.00%	8	
Perkins, Caitlin	169.00	169.00	100.00%	169.00	104	162.50%	104	
Lawson, Malik	494.00	494.00	100.00%	494.00	368	134.24%	368	
Moore, Lydia	29.00	29.00	100.00%	29.00	24	120.83%	24	
Shea, Bart	380.00	380.00	100.00%	380.00	320	118.75%	320	
Noble, Joyce	148.00	156.00	94.87%	148.00	128	115.63%	128	
Dominguez, Sheila	524.50	543.00	96.59%	524.50	464	113.04%	464	
Parker, Concetta	9.00	9.00	100.00%	9.00	8	112.50%	8	
Strong, Lela	296.50	296.50	100.00%	296.50	264	112.31%	264	
Riley, Veronica	516.00	516.00	100.00%	516.00	464	111.21%	464	
Owens, Betsy	400.00	400.00	100.00%	400.00	360	111.11%	360	
Fleming, Joyce	515.50	515.50	100.00%	515.50	464	111.10%	464	
Wells, Gail	408.35	409.10	99.82%	408.35	368	110.96%	368	
Lutz, Suzette	509.00	509.00	100.00%	509.00	464	109.70%	464	
Castro, Anne	490.00	490.00	100.00%	490.00	448	109.38%	448	
Jordan, Darrell	490.00	490.00	100.00%	490.00	448	109.38%	448	
Savage, Eunice	515.00	523.00	98.47%	515.00	472	109.11%	472	
Logan, Rosemarie	497.20	497.20	100.00%	497.20	456	109.04%	456	
Parker, Aurora	507.00	519.50	97.59%	507.00	472	107.42%	472	
Owen, Theresa	498.00	498.00	100.00%	498.00	464	107.33%	464	
Reed, Myron	470.00	470.00	100.00%	470.00	440	106.82%	440	
Sanford, Abram	503.50	511.50	98.44%	503.50	472	106.67%	472	
Kemp, Irving	486.00	486.00	100.00%	486.00	456	106.58%	456	
Fletcher, Anne	409.25	421.25	97.15%	409.25	384	106.58%	384	
Johns, Mariana	468.75	495.75	94.55%	468.75	440	106.53%	440	
Lawrence, Charley	502.50	539.50	93.14%	502.50	472	106.46%	472	
Reese, Bruce	450.00	450.00	100.00%	450.00	424	106.13%	424	
Goodwin, Carey	348.00	348.00	100.00%	348.00	328	106.10%	328	
Meyers, Chasity	456.00	456.00	100.00%	456.00	432	105.56%	432	
Houston, Candice	477.50	493.50	96.76%	477.50	456	104.71%	456	
Westerman, Scott	100.00	100.00	100.00%	100.00	96	104.17%	96	
Total	359.44	448.62	80.41%	149,490.15	6256	83.82%	178344	



EmployeeID	Date	Hours	HourlyCost	HourlyRate	TaskID	JobID	Division	PeriodStartDate	PeriodEndDate	TotalHoursBilled
CPERKINS	Sunday, January 13, 2019	8	\$81.51	105	3001TM	NASH000053	3001 Management	Saturday, January 12, 2019	Friday, January 18, 2019	72
CPERKINS	Sunday, January 20, 2019	5	\$81.51	105	3001TM	NASH000053	3001 Management	Saturday, January 19, 2019	Friday, January 25, 2019	10
CPERKINS	Saturday, January 19, 2019	5	\$81.51	105	3001TM	NASH000053	3001 Management	Saturday, January 19, 2019	Friday, January 25, 2019	10
CPERKINS	Friday, January 18, 2019	10	\$81.51	105	3001TM	NASH000053	3001 Management	Saturday, January 12, 2019	Friday, January 18, 2019	72
CPERKINS	Thursday, January 17, 2019	10	\$81.51	105	3001TM	NASH000053	3001 Management	Saturday, January 12, 2019	Friday, January 18, 2019	72
CPERKINS	Wednesday, January 16, 2019	12	\$81.51	105	3001TM	NASH000053	3001 Management	Saturday, January 12, 2019	Friday, January 18, 2019	72
CPERKINS	Tuesday, January 15, 2019	11	\$81.51	105	3001TM	NASH000053	3001 Management	Saturday, January 12, 2019	Friday, January 18, 2019	72
CPERKINS	Monday, January 14, 2019	11	\$81.51	105	3001TM	NASH000053	3001 Management	Saturday, January 12, 2019	Friday, January 18, 2019	72
CPERKINS	Saturday, January 12, 2019	10	\$81.51	105	3001TM	NASH000053	3001 Management	Saturday, January 12, 2019	Friday, January 18, 2019	72
CPERKINS	Friday, January 11, 2019	10	\$81.51	105	3001TM	NASH000053	3001 Management	Saturday, January 5, 2019	Friday, January 11, 2019	58
CPERKINS	Thursday, January 10, 2019	10	\$81.51	105	3001TM	NASH000053	3001 Management	Saturday, January 5, 2019	Friday, January 11, 2019	58
CPERKINS	Wednesday, January 9, 2019	12	\$81.51	105	3001TM	NASH000053	3001 Management	Saturday, January 5, 2019	Friday, January 11, 2019	58
CPERKINS	Tuesday, January 8, 2019	11.5	\$81.51	105	3001TM	NASH000053	3001 Management	Saturday, January 5, 2019	Friday, January 11, 2019	58
CPERKINS	Monday, January 7, 2019	10.5	\$81.51	105	3001TM	NASH000053	3001 Management	Saturday, January 5, 2019	Friday, January 11, 2019	58
CPERKINS	Sunday, January 6, 2019	1	\$81.51	105	3001TM	NASH000053	3001 Management	Saturday, January 5, 2019	Friday, January 11, 2019	58
CPERKINS	Saturday, January 5, 2019	3	\$81.51	105	3001TM	NASH000053	3001 Management	Saturday, January 5, 2019	Friday, January 11, 2019	58
CPERKINS	Friday, January 4, 2019	9	\$81.51	105	3001TM	NASH000053	3001 Management	Saturday, December 29, 2018	Friday, January 4, 2019	31
CPERKINS	Thursday, January 3, 2019	10	\$81.51	105	3001TM	NASH000053	3001 Management	Saturday, December 29, 2018	Friday, January 4, 2019	31
CPERKINS	Wednesday, January 2, 2019	10	\$81.51	105	3001TM	NASH000053	3001 Management	Saturday, December 29, 2018	Friday, January 4, 2019	31



Name	Average of TotalBillableHours	Average of TotalHours	Average of % Utilization	Total Billable Hours	Total Hours by Employee	% Utilization	Total	...
Bond, Noelle		472.00			464	0.00%	464	
Browning, Kristen		55.50			184	0.00%	184	
Cole, Kelly		374.00			424	0.00%	424	
Day, Lora		8.00			8	0.00%	8	
Duncan, Homer		8.00			8	0.00%	8	
Farris, Matt		40.00			40	0.00%	40	
Greer, John		120.00			120	0.00%	120	
Irwin, Rachelle	6.00	6.00	100.00%	6.00	0	0.00%	0	
Johnston, Demetrius		492.00			472	0.00%	472	
Kirby, Armando		200.00			200	0.00%	200	
Nicholson, Gladys		84.50			224	0.00%	224	
Nielsen, Derek	0.00	8.00	0.00%	0.00	32	0.00%	32	
Padilla, Xavier		20.25			32	0.00%	32	
Ramsey, Jorge		24.00			24	0.00%	24	
Rich, Malcom		24.00			24	0.00%	24	
Robbins, Landon		8.00			8	0.00%	8	
Rowland, Freddy		12.00			16	0.00%	16	
Shelton, Stan		472.00			472	0.00%	472	
Smith, Rocket		80.00			80	0.00%	80	
Summers, Serena		36.50			48	0.00%	48	
Todd, Chasity		63.00			96	0.00%	96	
Tran, Sylvester		472.00			472	0.00%	472	
Beasley, Joni	2.00	368.00	0.54%	2.00	368	0.54%	368	
Macdonald, Burton	6.00	272.00	2.21%	6.00	464	1.29%	464	
Hester, Stefan	10.00	411.00	2.43%	10.00	472	2.12%	472	
Graham, Casey	12.00	448.00	2.68%	12.00	448	2.68%	448	
Schaefer, Gustavo	14.50	474.50	3.06%	14.50	472	3.07%	472	
Mckee, Tasha	6.00	152.00	3.95%	6.00	152	3.95%	152	
Beltran, Elvira	22.00	387.00	5.68%	22.00	424	5.19%	424	
Contreras, Louie	10.25	50.25	20.40%	10.25	144	7.12%	144	
Prince, Ronnie	14.00	160.00	8.75%	14.00	160	8.75%	160	
Total	359.44	448.62	80.41%	149,490.15	6256	83.82%	178344	



EmployeeID	Date	Hours	HourlyCost	HourlyRate	TaskID	JobID	Division	PeriodStartDate	PeriodEndDate	TotalHoursBilled
RIRWIN	Sunday, January 13, 2019	6	\$44.484	0	3001FX	CHAR000842	3001 Management	Saturday, January 12, 2019	Friday, January 18, 2019	6

S ⊕ = □ = ∅ ∩

Month	Name	Average of TotalBillableHours	Average of TotalHours	Average of % Utilization	Total Billable Hours	Total Hours by Employee	% Utilization	Total Ho
□ January	Beasley, Joni	2.00	368.00	0.54%		24	0.00%	24
□ February	Bond, Noelle		472.00			128	0.00%	128
■ March	Browning, Kristen		55.50			32	0.00%	32
□ April	Cole, Kelly		374.00			80	0.00%	80
□ May	Farris, Matt		40.00			40	0.00%	40
□ June	Friedman, Kaye	83.75	458.00	18.29%		128	0.00%	128
□ July	Greer, John		120.00			120	0.00%	120
□ August	Guzman, Mariana	60.00	472.00	12.71%		128	0.00%	128
□ September	Johnston, Demetrius		492.00			128	0.00%	128
□ October	Kirby, Armando		200.00			128	0.00%	128
□ November	Massey, Frances	256.00	472.00	54.24%		128	0.00%	128
□ December	Nicholson, Gladys		84.50			40	0.00%	40
	Padilla, Xavier		20.25			32	0.00%	32
	Ramsey, Jorge		24.00			24	0.00%	24
	Rich, Malcom		24.00			8	0.00%	8
	Rowland, Freddy		12.00			16	0.00%	16
	Scott, Miguel	97.00	387.50	25.03%		32	0.00%	32
	Shelton, Stan		472.00			128	0.00%	128
	Smith, Rocket		80.00			80	0.00%	80
	Swanson, Marva	212.00	392.00	54.08%		120	0.00%	120
	Todd, Chasity		63.00			48	0.00%	48
	Tran, Sylvester		472.00			128	0.00%	128
	Wagner, Letitia	224.00	472.00	47.46%		128	0.00%	128
	Schaeter, Gustavo	14.50	474.50	3.06%	0.50	128	0.39%	128
	Caldwell, Edmundo	89.00	485.50	18.33%	1.00	128	0.78%	128
	Case, Alfonzo	89.00	472.00	18.86%	2.00	128	1.56%	128
	Graham, Casey	12.00	448.00	2.68%	4.00	128	3.13%	128
	Huerta, Lavern	30.00	154.00	19.48%	2.00	48	4.17%	48
	Dominguez, Rocky	181.00	472.00	38.35%	6.00	128	4.69%	128
	Macdonald, Burton	6.00	272.00	2.21%	6.00	128	4.69%	128
	Robbins, Della	59.50	59.50	100.00%	8.25	120	6.88%	120
	Beltran, Elvira	22.00	387.00	5.68%	9.00	128	7.03%	128
	Hester, Stefan	10.00	411.00	2.43%	10.00	128	7.81%	128
	Total	354.62	442.66	80.40%	42,115.18	528	85.32%	49360

S ⊕ = □ = ∅ ∩

Month	Name	Average of TotalBillableHours	Average of TotalHours	Average of % Utilization	Total Billable Hours	Total Hours by Employee	% Utilization	Total Hou
<input type="checkbox"/> January	Cox, Rex	317.00	576.50	54.99%	193.50	128	151.17%	128
<input type="checkbox"/> February	Lawson, Malik	494.00	494.00	100.00%	148.00	112	132.14%	112
<input checked="" type="checkbox"/> March	Shea, Bart	380.00	380.00	100.00%	10.00	8	125.00%	8
<input type="checkbox"/> April	Carter, Carolina	480.25	517.75	92.76%	153.75	128	120.12%	128
<input type="checkbox"/> May	Erickson, Darius	490.00	498.00	98.39%	153.00	128	119.53%	128
<input type="checkbox"/> June	Bond, Deidre	481.00	534.50	89.99%	152.00	128	118.75%	128
<input type="checkbox"/> July	Foley, Lucille	440.00	440.00	100.00%	104.00	88	118.18%	88
<input type="checkbox"/> August	Jordan, Darrell	490.00	490.00	100.00%	149.75	128	116.99%	128
<input type="checkbox"/> September	Dominguez, Sheila	524.50	543.00	96.59%	136.00	120	113.33%	120
<input type="checkbox"/> October	Wells, Gail	408.35	409.10	99.82%	144.20	128	112.66%	128
<input type="checkbox"/> November	Ingram, Lavonne	484.00	506.00	95.65%	144.00	128	112.50%	128
<input type="checkbox"/> December	Parker, Concetta	9.00	9.00	100.00%	9.00	8	112.50%	8
	Owen, Theresa	498.00	498.00	100.00%	143.00	128	111.72%	128
	Merritt, Caleb	434.00	486.00	89.30%	134.00	120	111.67%	120
	Lawrence, Charley	502.50	539.50	93.14%	142.25	128	111.13%	128
	Strong, Lela	296.50	296.50	100.00%	80.00	72	111.11%	72
	Whitney, Cletus	457.00	651.00	70.20%	133.00	120	110.83%	120
	Kemp, Irving	486.00	486.00	100.00%	132.00	120	110.00%	120
	Fleming, Joyce	515.50	515.50	100.00%	140.50	128	109.77%	128
	Underwood, Ronda	473.50	484.50	97.73%	140.50	128	109.77%	128
	Savage, Eunice	515.00	523.00	98.47%	140.00	128	109.38%	128
	Sparks, Josue	373.00	539.00	69.20%	140.00	128	109.38%	128
	Brewer, Sammy	484.50	484.50	100.00%	139.50	128	108.98%	128
	Meyers, Chasity	456.00	456.00	100.00%	122.00	112	108.93%	112
	Floyd, Christina	469.00	469.00	100.00%	130.50	120	108.75%	120
	Rivers, Wesley	470.50	482.50	97.51%	138.50	128	108.20%	128
	Riley, Veronica	516.00	516.00	100.00%	138.00	128	107.81%	128
	Schmitt, Leonor	479.50	495.50	96.77%	138.00	128	107.81%	128
	Ayala, Kristi	471.50	471.50	100.00%	128.75	120	107.29%	120
	Sanford, Abram	503.50	511.50	98.44%	137.00	128	107.03%	128
	Dudley, Hank	248.00	288.00	86.11%	136.00	128	106.25%	128
	Huffman, Leon	466.50	466.50	100.00%	127.50	120	106.25%	120
	Pittman, Norman	453.00	455.00	99.56%	136.00	128	106.25%	128
	Total	354.62	442.66	80.40%	42,115.18	528	85.32%	49360

S L = □ = Ø / ∫

Month	Name	Average of TotalBillableHours	Average of TotalHours	Average of % Utilization	Total Billable Hours	Total Hours by Employee	% Utilization	Total Hours
<input type="checkbox"/> January	Beasley, Joni	2.00	368.00	0.54%		24	0.00%	24
<input type="checkbox"/> February	Bond, Noelle		472.00			128	0.00%	128
<input checked="" type="checkbox"/> March	Browning, Kristen		55.50			32	0.00%	32
<input type="checkbox"/> April	Cole, Kelly		374.00			80	0.00%	80
<input type="checkbox"/> May	Farris, Matt		40.00			40	0.00%	40
<input type="checkbox"/> June	Friedman, Kaye	83.75	458.00	18.29%		128	0.00%	128
<input type="checkbox"/> July	Greer, John		120.00			120	0.00%	120
<input type="checkbox"/> August	Guzman, Mariana	60.00	472.00	12.71%		128	0.00%	128
<input type="checkbox"/> September	Johnston, Demetrius		492.00			128	0.00%	128
<input type="checkbox"/> October	Kirby, Armando		200.00			128	0.00%	128
<input type="checkbox"/> November	Massey, Frances	256.00	472.00	54.24%		128	0.00%	128
<input type="checkbox"/> December	Nicholson, Gladys		84.50			40	0.00%	40
	Padilla, Xavier		20.25			32	0.00%	32
	Ramsey, Jorge		24.00			24	0.00%	24
	Rich, Malcom		24.00			8	0.00%	8
	Rowland, Freddy		12.00			16	0.00%	16
	Scott, Miguel	97.00	387.50	25.03%		32	0.00%	32
	Shelton, Stan		472.00			128	0.00%	128
	Smith, Rocket		80.00			80	0.00%	80
	Swanson, Marva	212.00	392.00	54.08%		120	0.00%	120
	Todd, Chasity		63.00			48	0.00%	48
	Tran, Sylvester		472.00			128	0.00%	128
	Wagner, Letitia	224.00	472.00	47.46%		128	0.00%	128
	Schaefer, Gustavo	14.50	474.50	3.06%	0.50	128	0.39%	128
	Caldwell, Edmundo	89.00	485.50	18.33%	1.00	128	0.78%	128
	Case, Alfonzo	89.00	472.00	18.86%	2.00	128	1.56%	128
	Graham, Casey	12.00	448.00	2.68%	4.00	128	3.13%	128
	Huerta, Lavern	30.00	154.00	19.48%	2.00	48	4.17%	48
	Dominguez, Rocky	181.00	472.00	38.35%	6.00	128	4.69%	128
	Macdonald, Burton	6.00	272.00	2.21%	6.00	128	4.69%	128
	Robbins, Della	59.50	59.50	100.00%	8.25	120	6.88%	120
	Beltran, Elvira	22.00	387.00	5.68%	9.00	128	7.03%	128
	Hester, Stefan	10.00	411.00	2.43%	10.00	128	7.81%	128
Total		354.62	442.66	80.40%	42,115.18	528	85.32%	49360

EmployeeType

- ADMINISTRATIVE
- ASSOCIATE
- CONSULTANT
- HOURLY
- SALARY
- SUB-CONTRACTOR

Visualizations

Filters

Field

EmployeeType

Drill through

Cross-report

Keep all filters

Add drill-through fields here

S L = □ = Ø / ∫

Month	EmployeeType	Name	Average of TotalBillableHours	Average of TotalHours	Average of % Utilization	Total Billable Hours	Total Hours by Employee	% Utilization	Total Hours
<input type="checkbox"/> January		Beasley, Joni	2.00	368.00	0.54%		24	0.00%	24
<input type="checkbox"/> February		Bond, Noelle		472.00			128	0.00%	128
<input checked="" type="checkbox"/> March		Browning, Kristen		55.50			32	0.00%	32
		Cole, Kelly		374.00			80	0.00%	80
		Farris, Matt		40.00			40	0.00%	40
		Friedman, Kaye	83.75	458.00	18.29%		128	0.00%	128
		Greer, John		120.00			120	0.00%	120
		Guzman, Mariana	60.00	472.00	12.71%		128	0.00%	128
		Johnston, Demetrius		492.00			128	0.00%	128
		Massey, Frances	256.00	472.00	54.24%		128	0.00%	128
		Nicholson, Gladys		84.50			40	0.00%	40
		Padilla, Xavier		20.25			32	0.00%	32
		Ramsey, Jorge		24.00			24	0.00%	24
		Rich, Malcom		24.00			8	0.00%	8
		Rowland, Freddy		12.00			16	0.00%	16
		Scott, Miguel	97.00	387.50	25.03%		32	0.00%	32
		Shelton, Stan		472.00			128	0.00%	128
		Smith, Rocket		80.00			80	0.00%	80
		Swanson, Marva	212.00	392.00	54.08%		120	0.00%	120
		Todd, Chasity		63.00			48	0.00%	48
		Tran, Sylvester		472.00			128	0.00%	128
		Wagner, Letitia	224.00	472.00	47.46%		128	0.00%	128
		Schaefer, Gustavo	14.50	474.50	3.06%	0.50	128	0.39%	128
		Caldwell, Edmundo	89.00	485.50	18.33%	1.00	128	0.78%	128
		Case, Alfonzo	89.00	472.00	18.86%	2.00	128	1.56%	128
		Graham, Casey	12.00	448.00	2.68%	4.00	128	3.13%	128
		Huerta, Lavern	30.00	154.00	19.48%	2.00	48	4.17%	48
		Dominguez, Rocky	181.00	472.00	38.35%	6.00	128	4.69%	128
		Macdonald, Burton	6.00	272.00	2.21%	6.00	128	4.69%	128
		Robbins, Della	59.50	59.50	100.00%	8.25	120	6.88%	120
		Beltran, Elvira	22.00	387.00	5.68%	9.00	128	7.03%	128
		Hester, Stefan	10.00	411.00	2.43%	10.00	128	7.81%	128
		Prince, Ronnie	14.00	160.00	8.75%	14.00	128	10.94%	128
		Total	354.43	443.17	80.38%	41,995.18	128	85.52%	49104

Filters

Search

Filters on this visual

EmployeeType
is (All)

Add data fields here

Filters on this page

Add data fields here

Filters on all pages

EmployeeType
is HOURLY, SALARY, o...

Filter type

Basic filtering

Search

Select all

ADMINISTRATION 194

ASSOCIATE 125

CONSULTANT 253

HOURLY 281

SALARY 1097

SUB-CONTRACTOR 889

Require single selection

Add data fields here

S L = □ = ∅ / ∕

Month	EmployeeType	Name	Average of TotalBillableHours	Average of TotalHours	Average of % Utilization	Total Billable Hours	Total Hours by Employee	% Utilization	Total Ho
<input type="checkbox"/> January		Lawson, Malik	494.00	494.00	100.00%	148.00	112	132.14%	112
<input type="checkbox"/> February		Shea, Bart	380.00	380.00	100.00%	10.00	8	125.00%	8
<input checked="" type="checkbox"/> March		Foley, Lucille	440.00	440.00	100.00%	104.00	88	118.18%	88
		Jordan, Darrell	490.00	490.00	100.00%	149.75	128	116.99%	128
		Wells, Gail	408.35	409.10	99.82%	144.20	128	112.66%	128
		Strong, Lela	296.50	296.50	100.00%	80.00	72	111.11%	72
		Kemp, Irving	486.00	486.00	100.00%	132.00	120	110.00%	120
		Fleming, Joyce	515.50	515.50	100.00%	140.50	128	109.77%	128
		Riley, Veronica	516.00	516.00	100.00%	138.00	128	107.81%	128
		Ayala, Kristi	471.50	471.50	100.00%	128.75	120	107.29%	120
		Huffman, Leon	466.50	466.50	100.00%	127.50	120	106.25%	120
		Pittman, Norman	453.00	455.00	99.56%	136.00	128	106.25%	128
		Kaufman, Charlotte	479.00	479.00	100.00%	133.00	128	103.91%	128
		Meza, Margaret	469.00	469.00	100.00%	133.00	128	103.91%	128
		Price, Rupert	450.00	450.00	100.00%	133.00	128	103.91%	128
		Reed, Myron	470.00	470.00	100.00%	124.00	120	103.33%	120
		Houston, Candice	477.50	493.50	96.76%	131.50	128	102.73%	128
		Myers, Ricardo	420.25	420.25	100.00%	114.25	112	102.01%	112
		Logan, Rosemarie	497.20	497.20	100.00%	122.10	120	101.75%	120
		Quinn, Wilma	207.00	207.00	100.00%	130.00	128	101.56%	128
		Chambers, Jerri	306.75	306.75	100.00%	129.75	128	101.37%	128
		Atkinson, Annmarie	461.60	461.60	100.00%	129.60	128	101.25%	128
		Sutton, Cathy	385.50	385.50	100.00%	48.50	48	101.04%	48
		Dages, Jeff	80.75	80.75	100.00%	80.75	80	100.94%	80
		Fox, Dwayne	450.75	450.75	100.00%	128.75	128	100.59%	128
		Ruiz, Betsy	440.00	440.00	100.00%	120.50	120	100.42%	120
		Benitez, Flora	464.00	464.00	100.00%	128.00	128	100.00%	128
		Bennett, Wiley	444.00	444.00	100.00%	128.00	128	100.00%	128
		Case, Darell	464.00	464.00	100.00%	128.00	128	100.00%	128
		Compton, Rosendo	397.00	397.00	100.00%	88.00	88	100.00%	88
		Cross, Alyssa	454.00	454.00	100.00%	128.00	128	100.00%	128
		D'Orazio, Brad	80.00	80.00	100.00%	80.00	80	100.00%	80
		Fields, Myles	392.00	392.00	100.00%	128.00	128	100.00%	128
		Total	411.02	406.92	99.49%	6,906.90	128	99.81%	6920

S ⊕ = □ = ∅ ∕

Name Total Hours

Home table Calculations

Format Whole number

Data category Uncategorized

Structure

Formatting

Properties

X ✓

Januar

Februar

March

```
1 Total Hours =
2 SWITCH(
3     MAX('People'[EmployeeType]),
4     "HOURLY", [Total Billable Hours],
5     "SUB-CONTRACTOR", [Total Billable Hours],
6     "SALARY",
7     IF(
8         HASONEVALUE(People[Name]),
9         [Total Hours by Employee],
10        SUMX(
11            SUMMARIZE(
12                'People', People[Name],
13                "_Hours", [Total Hours by Employee]
14            ),
15            [_Hours]
16        )
17    ),
18    BLANK()
19 )
```

GO — P = Ø Ø

Hours by Employee % Utilization Total Hours

Employee	% Utilization	Total Hours
128	100.00%	130
120	100.00%	129
80	100.00%	79
128	100.00%	128
128	100.00%	128
128	100.00%	123
128	100.00%	128
128	100.00%	130
88	100.00%	88
128	100.00%	128
120	100.00%	119
128	100.00%	126
80	100.00%	81
128	100.00%	128
80	100.00%	80
128	100.00%	128
128	100.00%	141
88	100.00%	104
128	100.00%	129
128	100.00%	125
128	100.00%	117
128	100.00%	

Division % Utilization

Division	% Utilization
1001 Technology	100.00%
2001 Accounting	100.00%
3001 Management	100.00%
Marketing	0.00%
Shared Services	0.00%
Total	100.00%

Filters

Search:

Filters on this visual

- % Utilization is (All)
- Division is (All)

Add data fields here

Filters on this page

Add data fields here

Filters on all pages

- EmployeeType** is HOURLY, SALARY, o...

Add data fields here

Visualizations

Build visual

Fields

Search:

Calculations

- % Utilization
- Total Billable H...
- Total Hours
- Total Hours by ...

Budgets and Forecasts

Calendar

Hours

- % Utilization
- Category
- Date
- Division
- EmployeeID
- HourlyCost
- HourlyRate
- Hours
- JobID
- PayType

PeriodEndDate

Drill through

Cross-report

Keep all filters

Add drill-through fields here

GO□—P=∅∅

Name % Utilization Format General Data category Uncategorized

Home table Calculations \$ % , .00 Auto

Structure Formatting Properties

X ✓

February

March

EmployeeType

HOURLY

SALARY

SUB-CONT

```
1 % Utilization =
2     VAR _utilization =
3         DIVIDE([Total Billable Hours],[Total Hours],0)
4     VAR _days = COUNTROWS('Hours')
5 RETURN
6     SWITCH(TRUE(),
7         ISINSCOPE('People'[Name]) && ISBLANK(_days),
8         BLANK(),
9         ISINSCOPE('People'[Name]), _utilization + 0,
10        VAR _tempTable =
11            SUMMARIZE(
12                'People',
13                'People'[Name],
14                "_billableHours", [Total Billable Hours],
15                "_totalHours", [Total Hours]
16            )
17        VAR _billedHours = SUMX(_tempTable, [_billableHours])
18        VAR _hours = SUMX(_tempTable, [_totalHours])
19 RETURN
20         DIVIDE(_billedHours, _hours,0)
21 )|
```



- Segmenting data
- Using report navigating features
- Advanced analysis techniques



- Power BI provides several mechanisms for **segmenting data**. These include the ability to use **groups, hierarchies, and Row-Level Security (RLS)**. Segmenting data allows you to partition or group individual rows of data into logical units that make sense to an organization. This helps ensure that business rules are enforced or that metrics can be shown in a manner that is more easily digestible by business users versus looking at individual rows of data.

Practice_3 - Power BI Desktop

Vien Nguyen

File Home Insert Modeling View Help Format Data / Drill

Cut Copy Format painter Clipboard

Get data v Excel Data SQL Server Enter Dataverse Recent sources v Transform Refresh data v New visual Text box More visuals v New measure Quick Sensitivity v Publish Share

Hours by Month

Hours

60K
40K
20K
0K

January February March

Month

Filters

Search

Filters on this visual

- Hours is (All)
- Month is (All)

Add data fields here

Filters on this page

Add data fields here

Filters on all pages

EmployeeType is HOURLY, SALARY, o...

Add data fields here

Visualizations

Build visual

4

2

3

1

Fields

Search

Calculations

Budgets and Forecasts

Calendar

Date

IsWorkDay

Month

MonthNum

MonthYear

Weekday

WeekdayNum

WeekNum

WorkHours

Year

Hours

% Utilization

Category

Date

Division

EmployeeID

HourlyCost

HourlyRate

Hours

Page 1 Utilization Duplicate of Utilization Page 2 +

Page 4 of 4

73%

The screenshot shows a Power BI desktop interface with a bar chart titled 'Hours by Month' on the left. The chart displays hours worked for January, February, and March. The Y-axis represents 'Hours' from 0K to 60K, and the X-axis represents 'Month'. On the right, there are three main sections: 'Filters', 'Visualizations', and 'Fields'. The 'Filters' section contains dropdowns for 'Hours' (is (All)) and 'Month' (is (All)), along with sections for 'Filters on this page' and 'Filters on all pages' (EmployeeType is HOURLY, SALARY, o...). The 'Visualizations' section shows a 'Build visual' button and a grid of visualization icons. The 'Fields' section lists various data fields like Calculations, Budgets and Forecasts, and a detailed list under 'Hours' including % Utilization, Category, Date, Division, EmployeeID, HourlyCost, HourlyRate, and Hours. Several fields in the 'Fields' section are highlighted with red boxes and numbered 1 through 4. Red box 1 highlights the 'Page 2' button at the bottom. Red box 2 highlights the 'Month' field in the 'Fields' section. Red box 3 highlights the 'Hours' field in the 'Fields' section. Red box 4 highlights the 'Build visual' button in the 'Visualizations' section.

» Fields »

Search

Calculations

Budgets and Forecasts

Calendar

Hours

MonthYears

People

Py Check

Create hierarchy

New measure

New column

New quick measure

Rename

Delete from model

Hide

View hidden

Unhide all

Collapse all **1**

Expand all

New group

Add to filters >

Add to drill through

Groups

Name * EmployeeType (groups)

Field EmployeeType

Group type List

Ungrouped values

ADMINISTRATION
ASSOCIATE
CONSULTANT

Groups and members

- ▲ Billable
 - HOURLY
 - SALARY
 - SUB-CONTRACTOR
- ▲ Other
 - Contains all ungrouped values

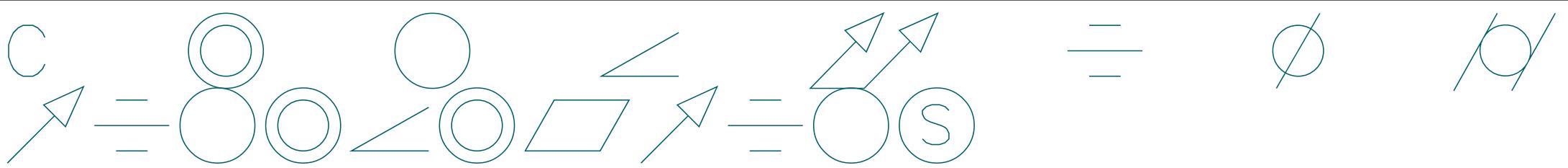
Include Other group ⓘ

OK Cancel

X

The screenshot shows three panels from the Power BI Fields pane:

- Filters:** Contains a search bar and sections for "Filters on this page" and "Filters on all pages". A filter for "EmployeeType" is selected, showing its value as "is HOURLY, SALARY, o...".
 - 1: A red box highlights the "EmployeeType" filter.
- Fields:** Shows a list of fields under the "People" category.
 - 2: A red box highlights the "EmployeeType (gro...)" field, which has a checked checkbox.
- Fields:** Shows a list of fields under the "Hours" category.
 - 3: A red box highlights the "Σ Hours" field, which also has a checked checkbox.



- **Hierarchies** are a powerful feature of Power BI that allows data to be summarized but also allows report viewers to drill down into the data to obtain additional details. Similar to groups, we can create ad hoc hierarchies as well as define hierarchies within our data model. The easiest way to see an ad hoc hierarchy is to use a **Matrix** visualization.

The screenshot shows the Power BI interface with several components:

- Left Panel:** A "Location" dropdown menu with options Charlotte, Cleveland, and Nashville.
- Filters Panel:** A sidebar with sections for "Filters on this visual", "Filters on this page", and "Filters on all pages".
- Visualizations Panel:** A grid of visualization icons. One icon, a grid, is highlighted with a red box and the number 2.
- Fields Panel:** A list of fields under "People": EmployeeType, EmployeeType (grouped), HireDate, ID, Location, Name, TermDate, and Title. The "Location" field is selected and highlighted with a red box and the number 1.

The screenshot shows the "Visualizations" pane with the following settings:

- Format visual:** Buttons for "Grid", "Pencil", and "Search".
- Text:** Font set to "Segoe UI" at size 10, with bold (B), italic (I), and underline (U) options.
- Text color:** Black.
- Background color:** White.
- Banded row color:** On.
- Alignment:** Left, Center, Right.
- Text wrap:** On.
- Icons:** A section with a "Search" button and a "More icons" button (highlighted with a red box and the number 3).

Location	Hours
Charlotte	27,801.50
Cleveland	96,731.50
Nashville	53,464.15
Total	177,997.15

Location	Hours
Charlotte	27,801.50
Cleveland	96,731.50
Nashville	53,464.15
Total	177,997.15

 Filters

 Search

Filters on this visual ...

Division
is (All)

Hours
is (All)

Location
is (All)

Add data fields here

Filters on this page ...

Add data fields here

Filters on all pages ...

Add data fields here

Visualizations

Build visual

Rows **1**

Location **X**

Division **X**

Columns

Add data fields here

Values **2**

Hours **X**

> Fields

Search

> Calculations

> Budgets and Forecasts

> Calendar

> Hours

- % Utilization
- Category
- Date
- Division
- EmployeeID
- \sum HourlyCost
- \sum HourlyRate
- \sum Hours
- JobID
- PayType

> PeriodEndDate

> PeriodStartDate

TaskID

TimesheetBatchID

TimesheetID



- We can also define hierarchies as part of our data model. However, unlike the previous case, where we created an ad hoc hierarchy from columns in separate tables, hierarchies that are created as part of the data model must have all of the columns in the hierarchy come from the same table.

The screenshot shows the Power BI Fields pane with three main sections: Fields, Visualizations, and Fields. The first and third sections show a tree view of fields, while the middle section shows visualization icons and a context menu.

Fields Section:

- Search bar: Search
- Calculations
- Budgets and Forecasts
- Calendar
- Hours** (selected):
 - % Utilization
 - Category
 - Date
 - Division** (selected)
 - Division Hierar...** (highlighted with a red box)
 - Division
 - JobID
 - EmployeeID
 - \sum HourlyCost
 - \sum HourlyRate
 - \sum Hours** (selected)
 - JobID
 - PayType
 - PeriodEndDate
 - PeriodStartDate

Visualizations Section:

- Build visual
- Icons for various visualizations: Bar, Line, Stacked Bar, Area, Scatter, Map, Timeline, Gantt, Circular, Treemap, Radar, Card, Gauge, Number, Date, Time, Percentage, Rating, Progress, and More.

Fields Section (right):

- Search bar: Search
- Calculations
- Budgets and Forecasts
- Calendar
- Hours** (selected):
 - % Utilization
 - Category
 - Date
 - Division** (highlighted with a red box)
 - EmployeeID
 - HourlyCost

Context Menu (Visualizations Section):

- Check
- Create hierarchy
- Values
- Add data file
- Drill through
- Cross-report
- Keep all filters
- Add drill-through
- New measure
- New column
- New quick measure
- Rename
- Delete from model
- Hide
- View hidden
- Unhide all

Division	Hours
1001 Technology	115,613.45
2001 Accounting	45,768.00
3001 Management	15,386.70
8001 Customer Management	726.25
9001 Sales	411.00
Marketing	55.50
Quality	16.00
Shared Services	20.25
Total	177,997.15

↑ ↓ ↻ ⌂ ⌂ ...

Filters

Search

Filters on this visual

- Division is (All)
- Hours is (All)
- JobID is (All)
- Add data fields here

Filters on this page

Add data fields here

Filters on all pages

Add data fields here

Visualizations

Build visual

Rows

Division Hierarchy

- Division
- JobID

Columns

Add data fields here

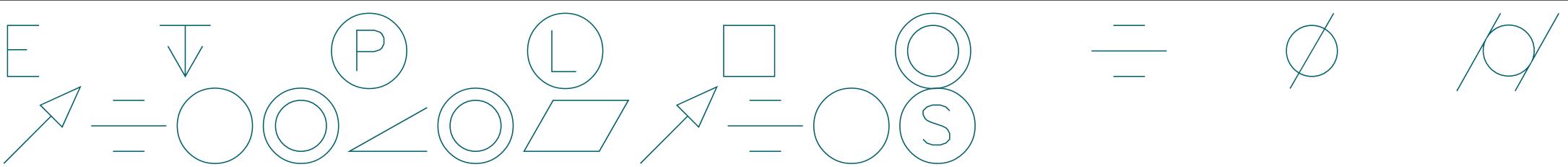
Values

Hours

Fields

Search

- Calculations
- Budgets and Forecasts
- Calendar
- Hours
 - % Utilization
 - Category
 - Date
 - Division
 - Division Hierarchy** (1) (highlighted)
 - Division
 - JobID
 - EmployeeID
 - HourlyCost
 - HourlyRate
 - Hours** (2) (highlighted)
 - JobID
 - PayType
- PeriodEndDate
- PeriodStartDate
- TaskID



Division

Division	Hours
1001 Technology	115,613.45
2001 Accounting	45,768.00
3001 Management	15,386.70
8001 Customer Management	726.25
9001 Sales	411.00
Marketing	55.50
Shared Services	20.25
Quality	16.00
Total	177,997.15

3

2

1

Filters

Search

Filters on this visual

- Division is (All)
- Hours is (All)
- JobID is (All)

Add data fields here

Visualizations

Build visual

1

Fields

Search

- Calculations
- Budgets and Forecasts
- Calendar
- Hours
- MonthYears
- People

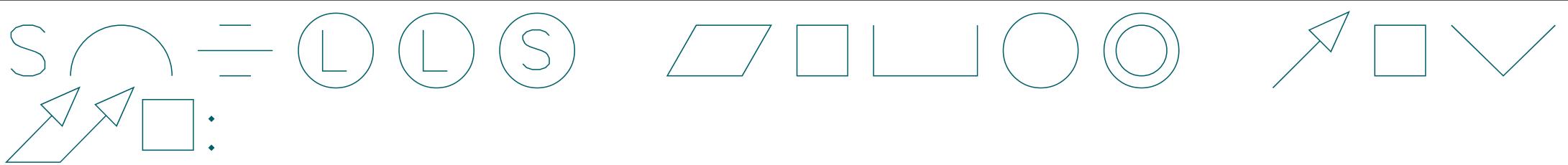
Rows

UΦΩΟΟΣ ΑΦΩΝΑΙ ΡΛΣ

- RLS is used to restrict data access for users viewing reports. Roles are created within Power BI Desktop by using DAX expressions to define filters. These filters restrict the data that's available to that role at the row level of the data. Users are added to roles from within the Power BI Service, but Desktop provides the ability to test role definitions by allowing the report author to view reports as a particular role.

COOKING =/= ØØLOS

- You know that when she creates and publishes the final report, various job functions within the organization will look at the report. Two of these job functions are branch managers and division managers. You want to ensure that each of these managers only sees the information that pertains to their area of management, without having to create separate reports for each job function and each user within those job functions. You can do this by creating roles for each using RLS.



- Apply conditional formatting

- Perform top N analysis

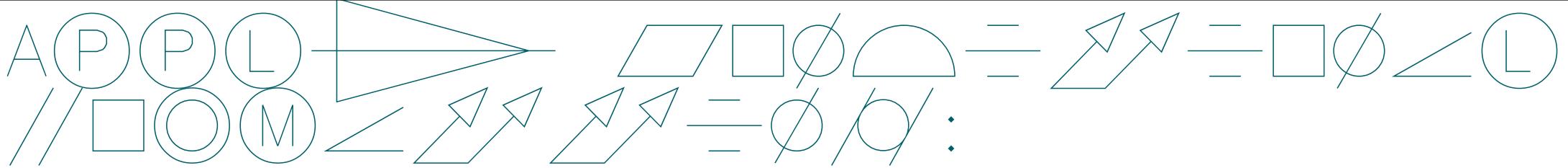
- Explore statistical summary

- **Add a Quick Insights result to a dashboard**

- Create reference lines by using the Analytics pane

- Use the Play Axis feature of a visualization and conduct time-series analysis

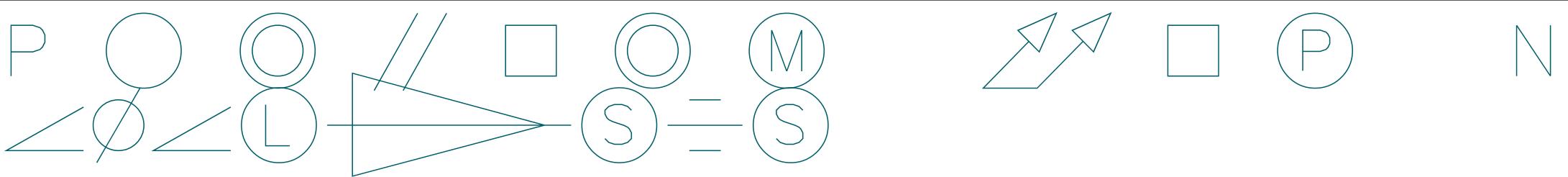
- Personalize visuals



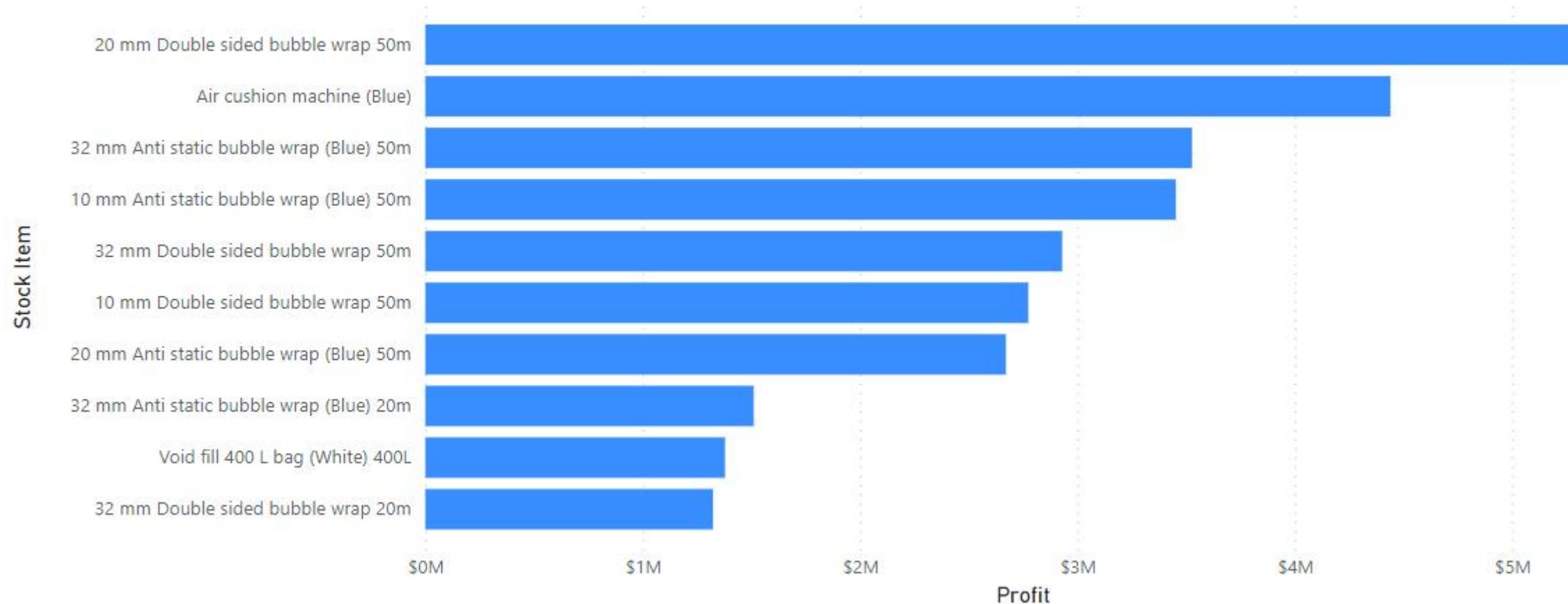
- Tables and matrixes
 - **Background color** Changes the background color of cells
 - **Font color** Changes the font color of values
 - **Data bars** Displays bars in cells
 - **Icons** Displays icons in cells
 - **Web URL** Underlines values and turns cells into hyperlinks; otherwise doesn't change the appearance of values

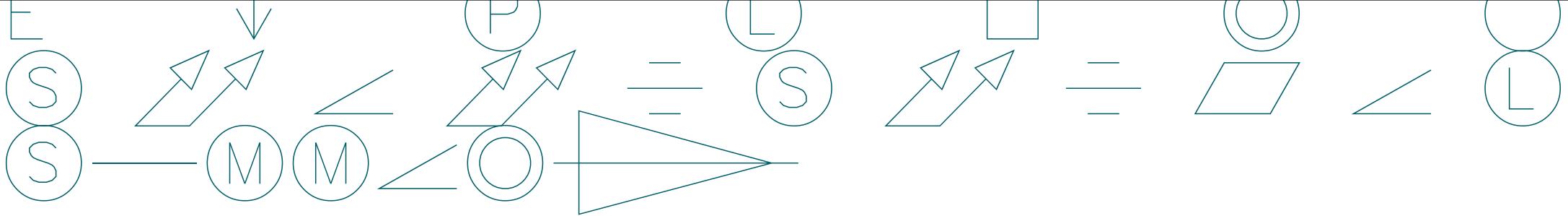
Category	Background color	Font color	Data bars	Icons	
Black	1,101,101	\$30,209,672	\$26,269,280		\$14,390,527
Blue	605,738	\$41,289,464	\$35,903,882		\$17,713,073
Gray	309,847	\$6,823,755	\$5,933,696		\$1,475,749
Light Brown	274,500	\$5,682,150	\$4,941,000		(\$274,500)
N/A	5,745,694	\$89,523,436	\$77,896,013		\$40,792,234
Red	29,033	\$1,335,351	\$4,726,465		\$2,179,948
White	873,153	\$17,365,861	\$15,100,749		\$8,882,849
Yellow	11,562	\$1,000,000	\$1,490,145		\$569,303
Total	8,950,628	\$198,043,439	\$172,261,230		\$85,729,181

Color	Quantity	Total Including Tax	Total Excluding Tax	Profit
Black	1,101,101	\$30,209,672	\$26,269,280	▲ \$14,390,527
Blue	605,738	\$41,289,464	\$35,903,882	▲ \$17,713,073
Gray	309,847	\$6,823,755	\$5,933,700	◆ \$1,475,749
Light Brown	274,500	\$5,682,150	\$4,941,000	◆ (\$274,500)
N/A	5,745,694	\$89,523,436	\$77,896,120	● \$40,792,234
Red	29,033	\$5,121,150	\$4,726,465	◆ \$2,179,948
White	873,153	\$17,365,861	\$15,100,749	◆ \$8,882,849
Yellow	11,562	\$1,490,145	\$1,490,145	◆ \$569,303
Total	8,950,628	\$198,043,439	\$172,261,341	\$85,729,181



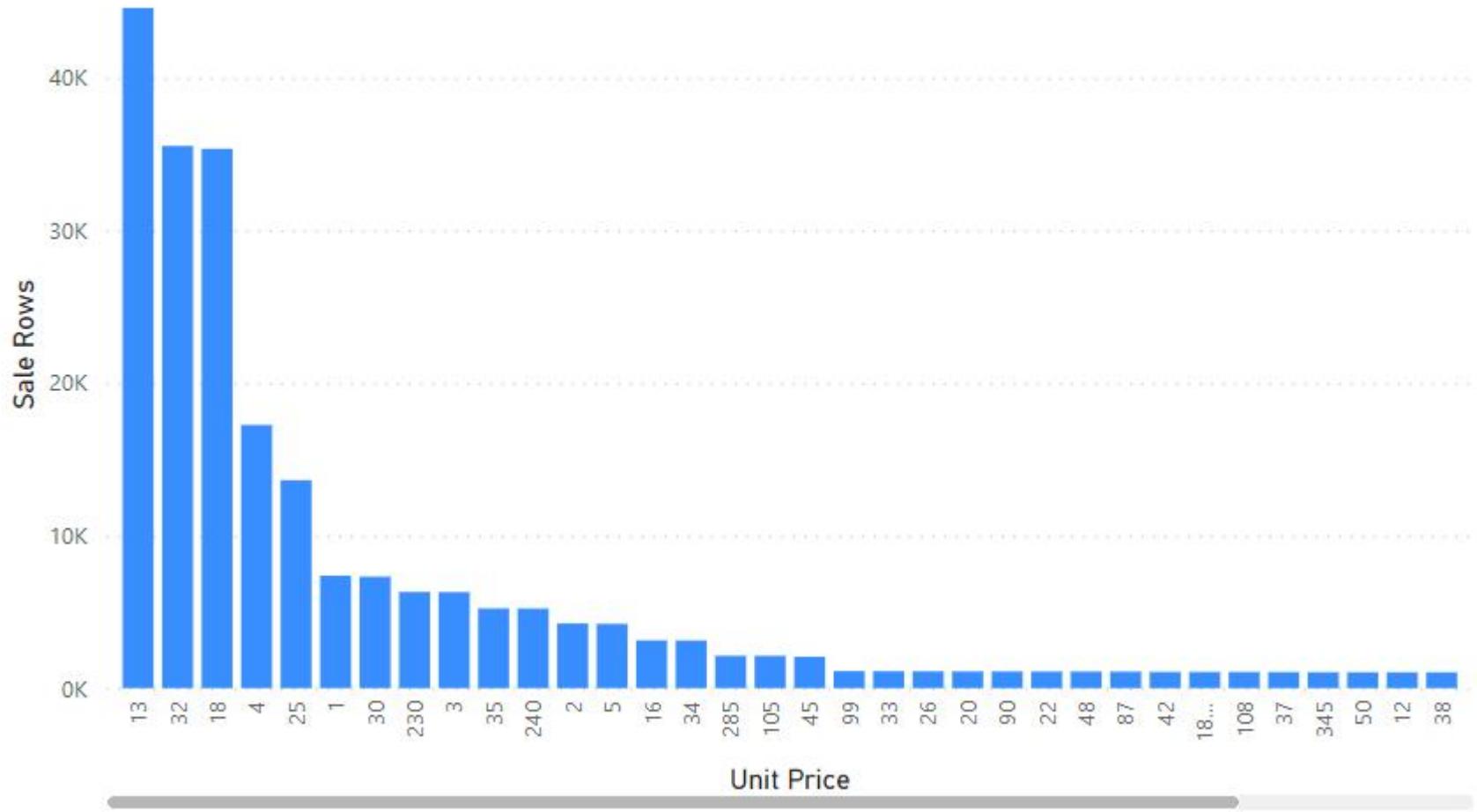
Profit by Stock Item

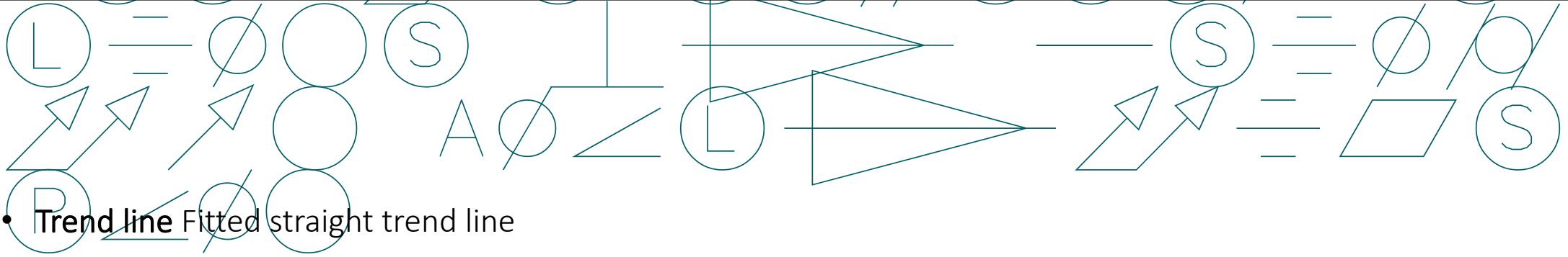




Sale Rows = COUNTROWS(Sale)

Sale Rows by Unit Price

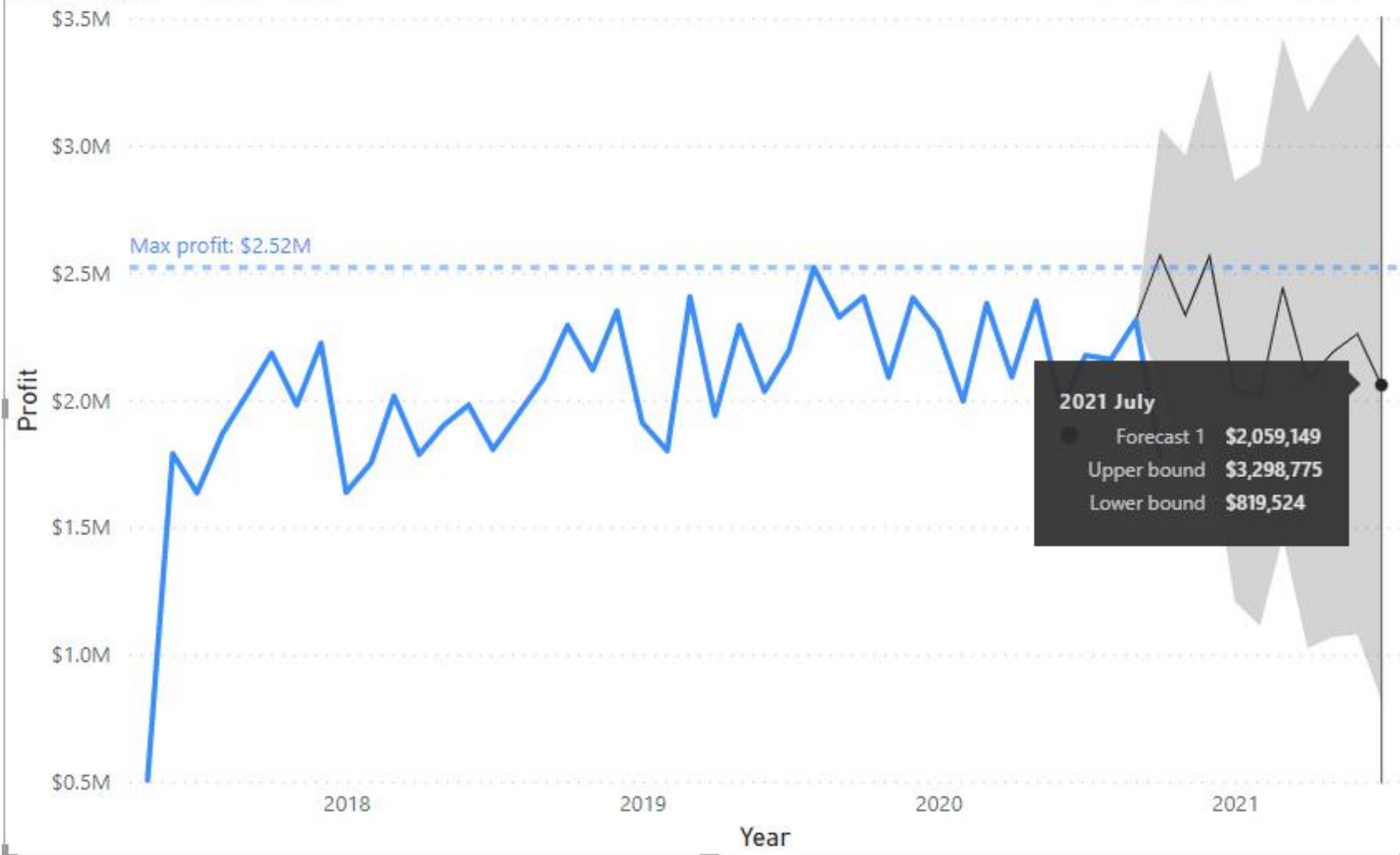


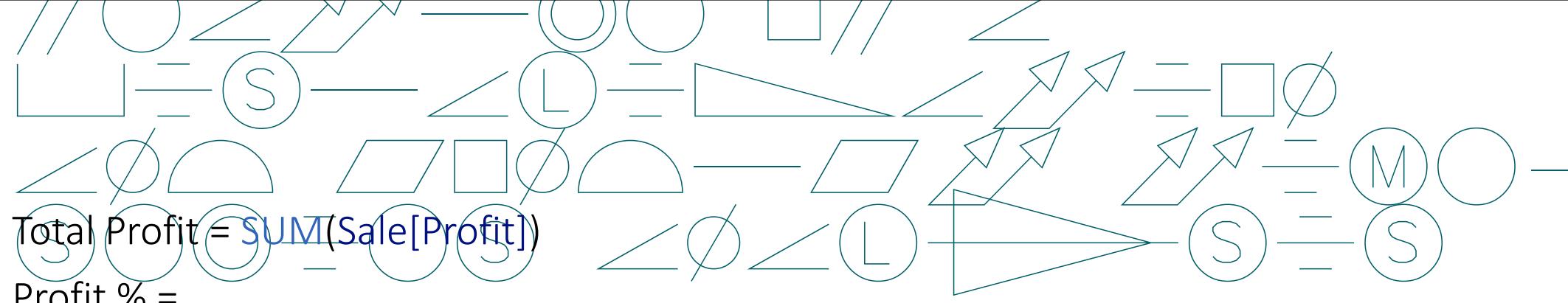


- **Trend line** Fitted straight trend line

- **X-axis constant line** Static vertical line
- **Y-axis constant line** Static horizontal line
- **Min line** Dynamic minimum line
- **Max line** Dynamic maximum line
- **Average line** Dynamic average (mean) line
- **Median line** Dynamic median line
- **Percentile line** Dynamic line for which you need to enter the percentile value
- **Forecast** Forecast line of a set length
- **Symmetry shading** Shades the regions across the symmetry line
- **Ratio line** Ratio of total for all points of Y values over X values

Profit by Year and Month





Total Profit = **SUM**(Sale[Profit])

Profit % =

DIVIDE(

[Total Profit],

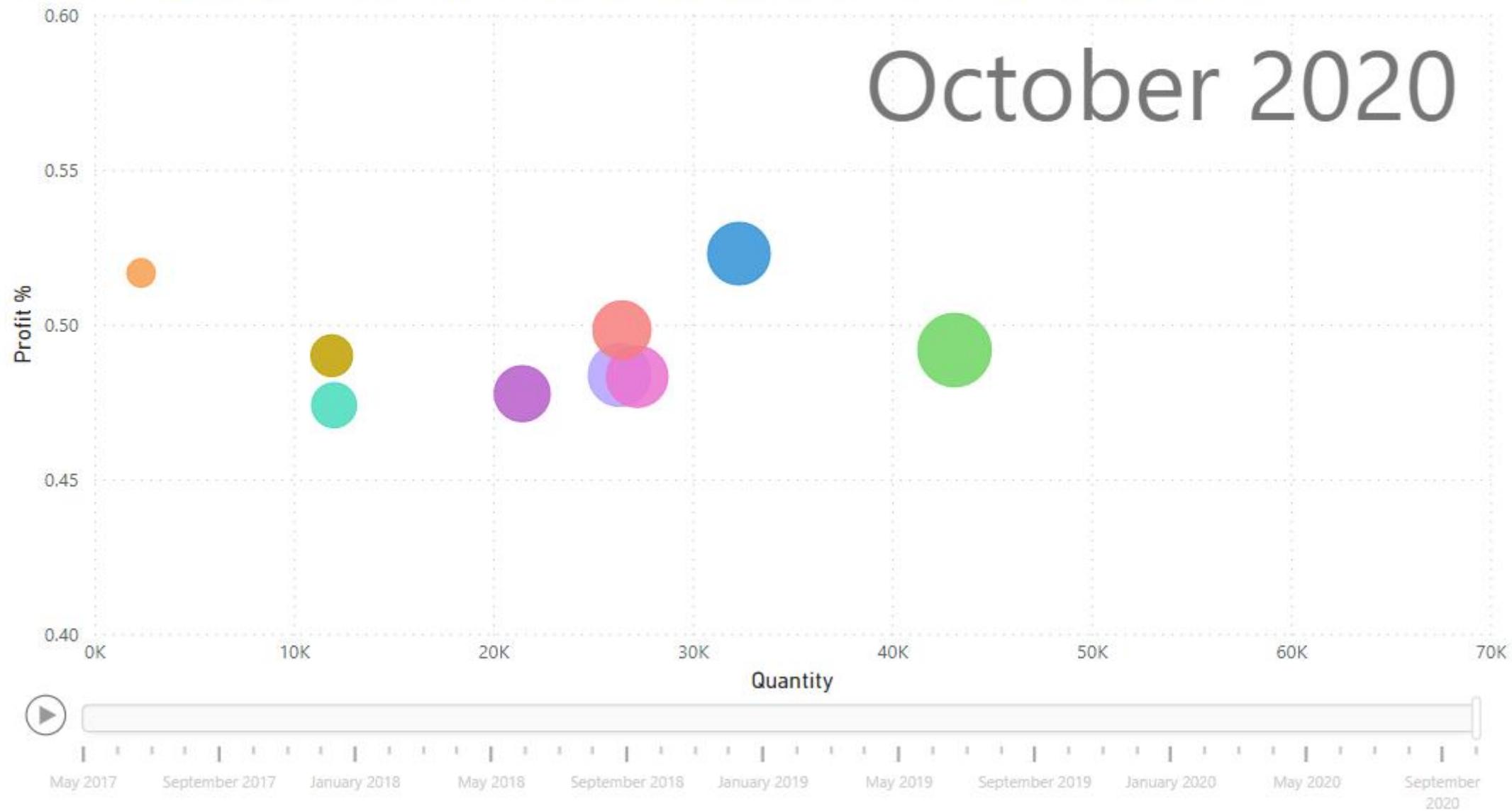
SUM(Sale[Total Excluding Tax])

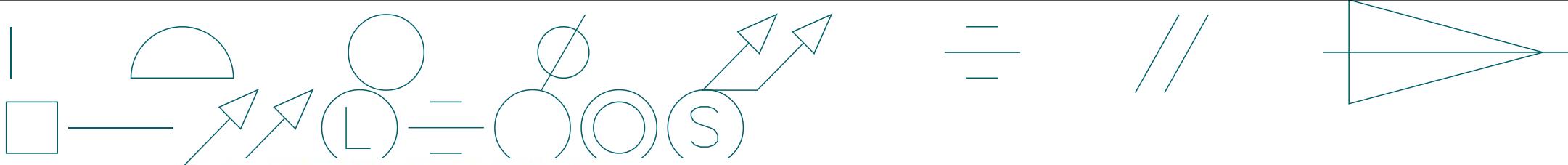
)

Month Year = **EOMONTH**('Date'[Date], 0)

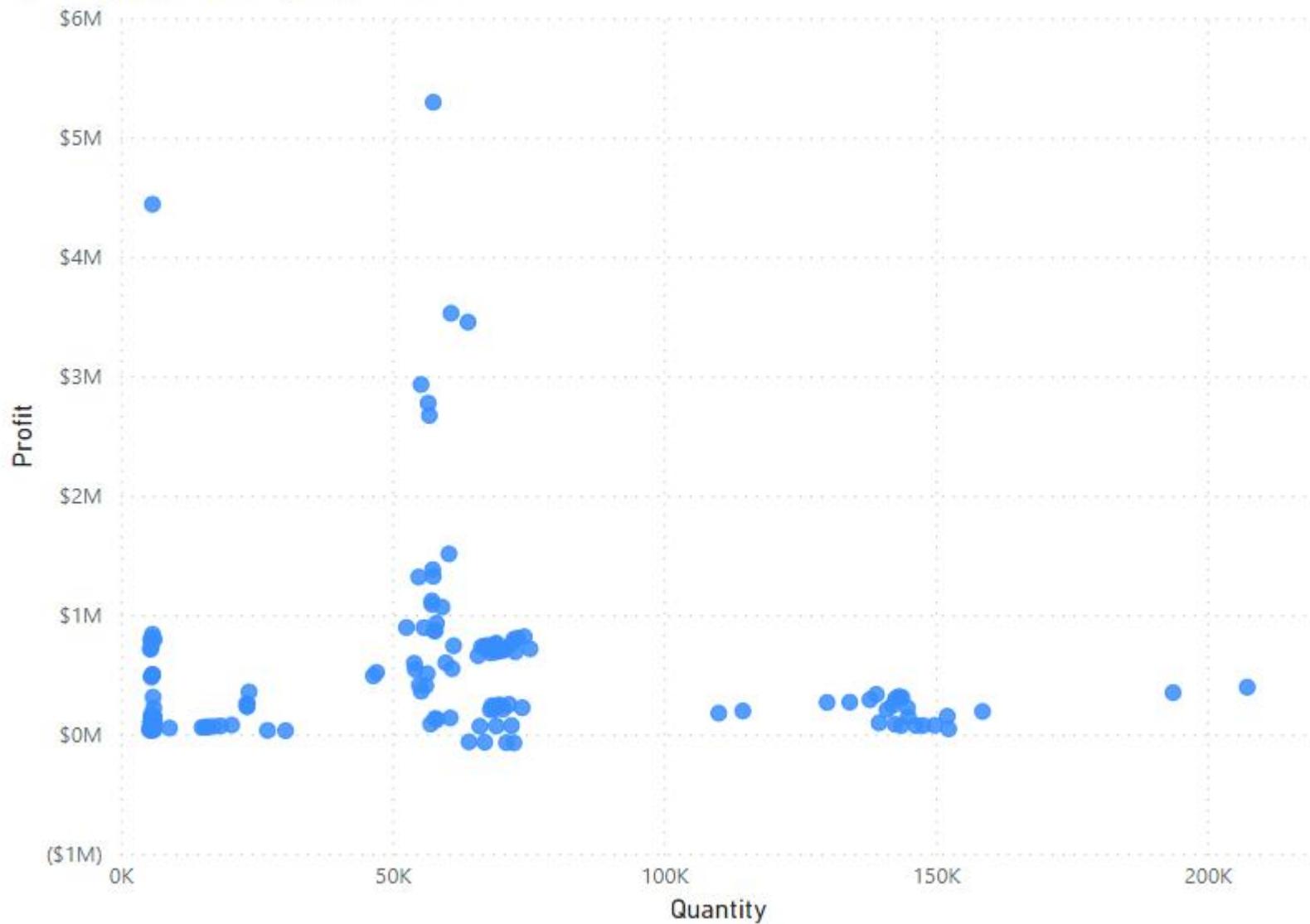
Quantity, Profit % and Sale Rows by Sales Territory and Month Year

Sales Territory ● External ● Far West ● Great Lakes ● Mideast ● New England ● Plains ● Rocky Mountain ● Southeast ● Southwest



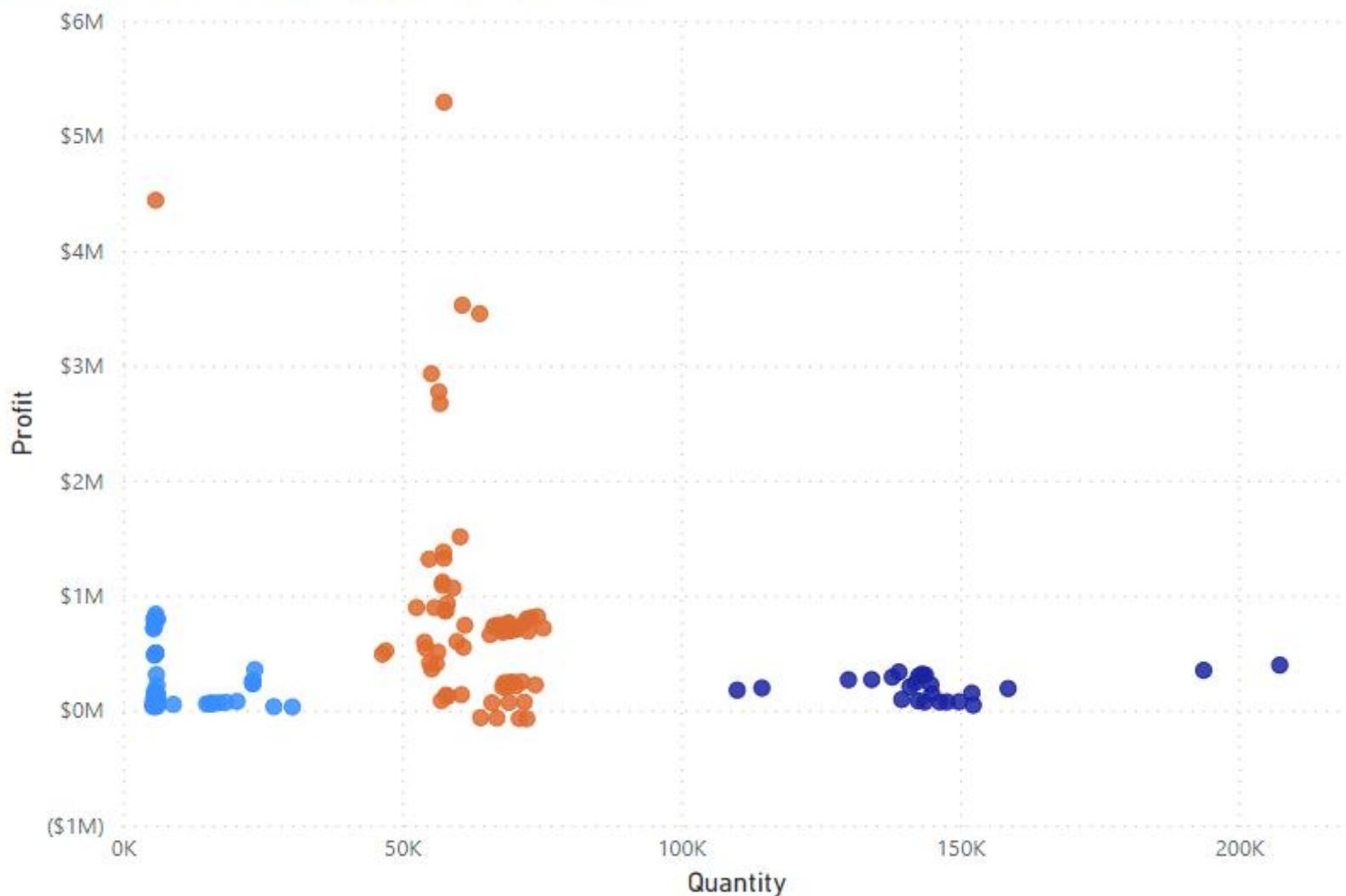


Quantity and Profit by Stock Item



Quantity and Profit by Stock Item and Stock Item (clusters) 3

Stock Item (clusters) 3 ● Cluster1 ● Cluster2 ● Cluster3

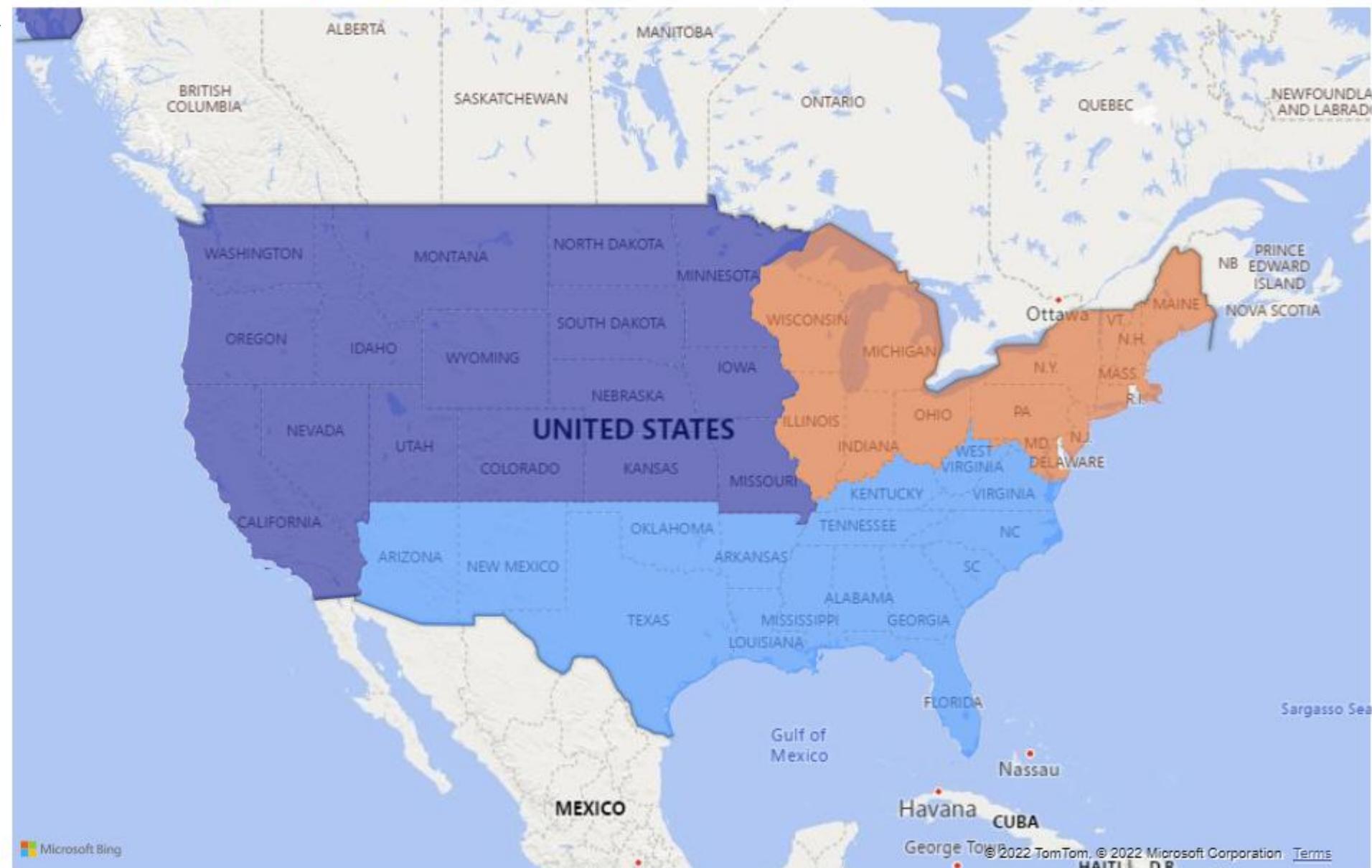


The logo consists of the letters 'G' and 'O' in a bold, black, sans-serif font. The letter 'O' is slightly taller than 'G'. A thick horizontal line extends from the right side of 'O' to the right, ending with a small vertical tick mark. To the right of this line is a circular outline containing the letter 'P'.

Profit by State Province and Sales Region

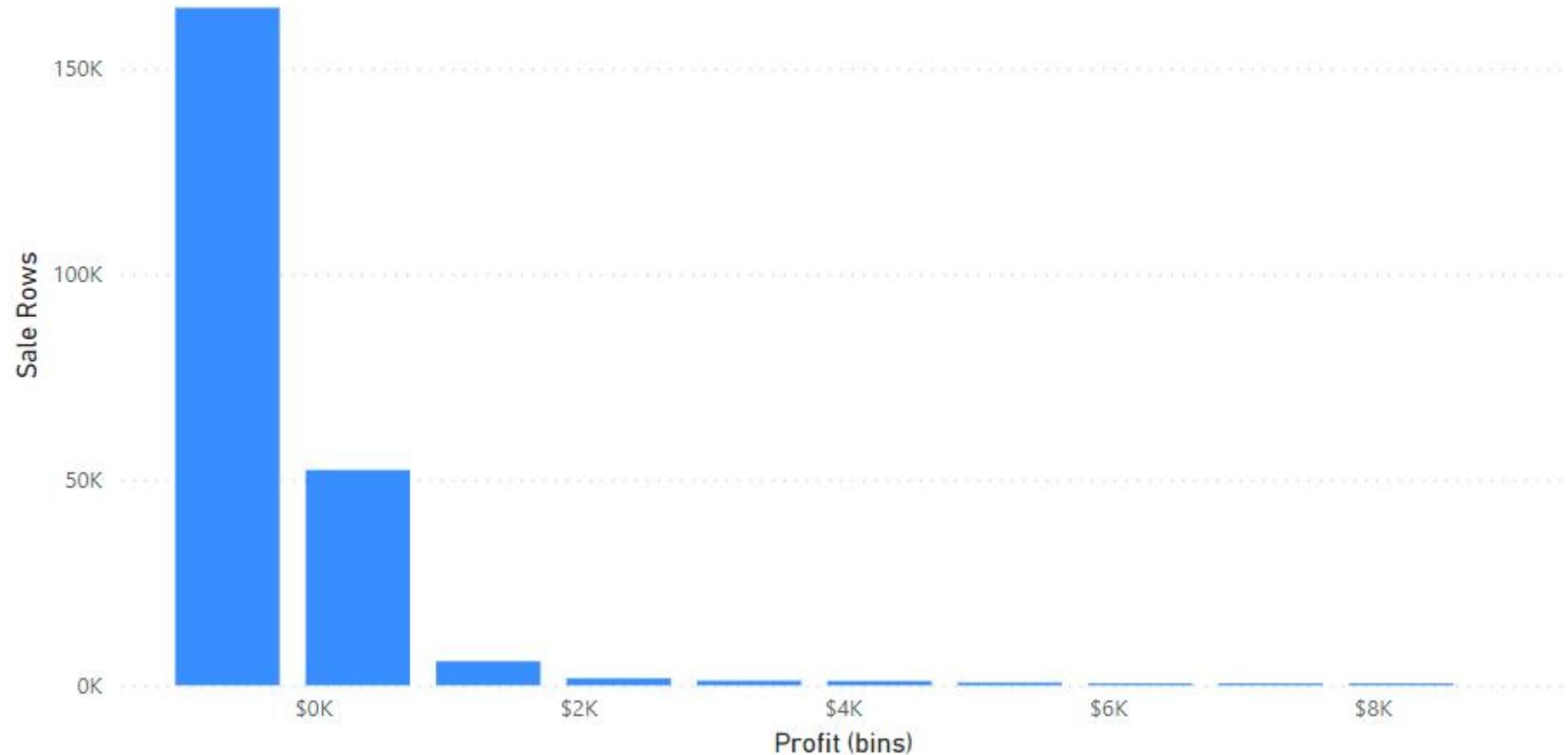
Sales Region ● East ● North ● South ● West

- **West:** Far West, Plains, and Rocky Mountain
 - **East:** Great Lakes, Mideast, and New England
 - **South:** Southeast, Southwest
 - **North:** the others



B = ØØ = ØØS

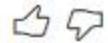
Sale Rows by Profit (bins)





- **Analyze** The metric you want to analyze. It can be continuous or categorical.
 - **Explain by** The factors you want to check in terms of influence.
 - **Expand by** When analyzing a measure or a summarized field, you can increase the detail level without considering these factors as influencers.

Key influencers Top segments



What influences Profit to Increase ▼ ?

When...

...the average of Profit
increases by

| State Province is Texas

\$11.23K

Buying Package is Each

\$8.79K

State Province is
Pennsylvania

\$7.04K

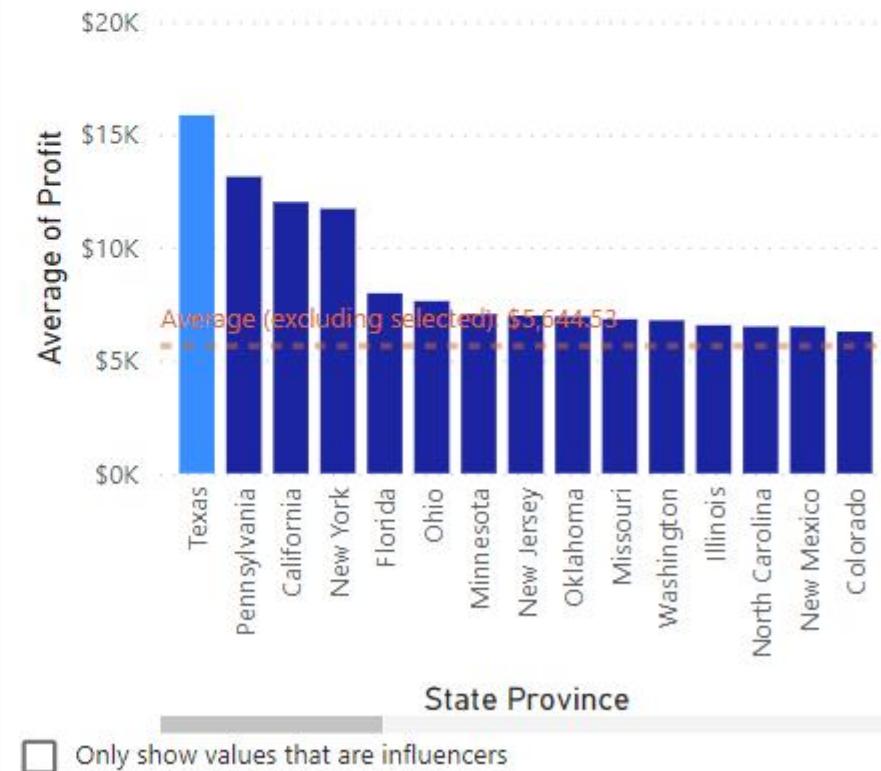
State Province is California

\$6.66K

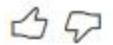
State Province is New York

\$5.91K

← Profit is more likely to increase when State Province is Texas than
otherwise (on average).



Key influencers Top segments



When is Profit more likely to be High ▼ ?

We found 3 segments and ranked them by Average of Profit and population size. Select a segment to see more details.



Segment 1

Average of Profit

\$12.31K

Segment 2

\$10.8K

Segment 3

\$10.51K

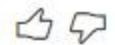
Population count

1946

1909

1934

Key influencers **Top segments**



When is Profit more likely to be **High** ?



Segment 1

Buying Group is N/A

Buying Package is Each

In segment 1, the average Profit is \$12.31K. This is \$6.66K units higher than the overall average, \$5.64K.

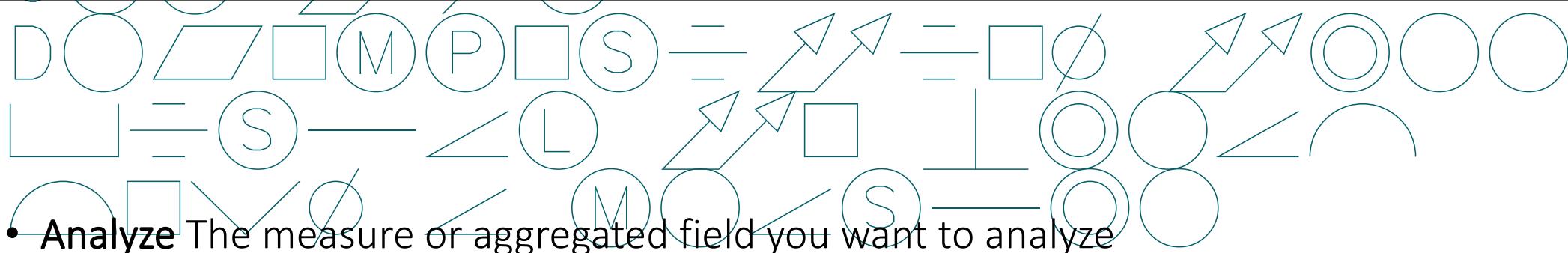
Segment 1 \$12.31K

Overall \$5.64K

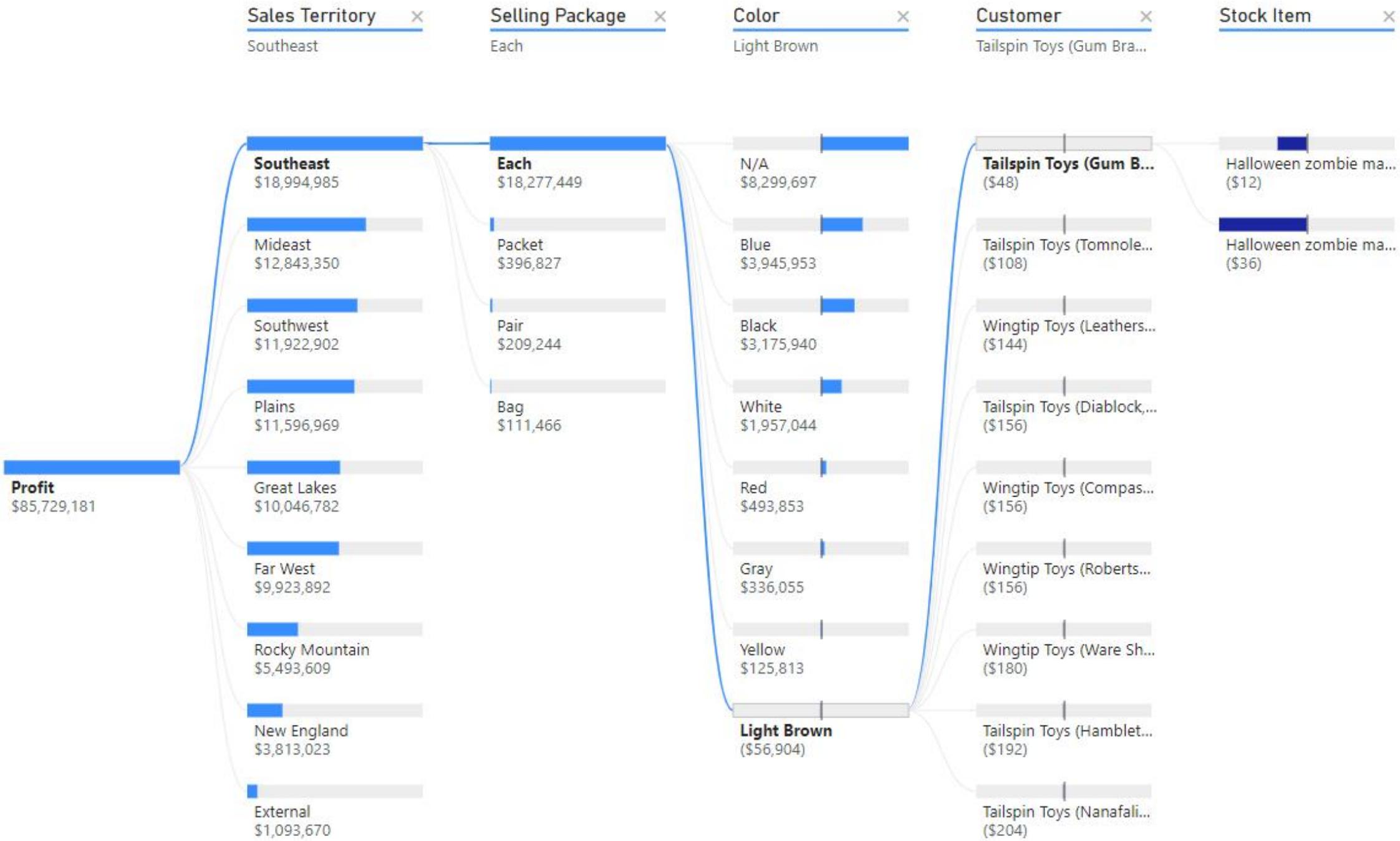
Segment 1 contains 1,946 data points (12.8% of the data).

- Segment 1
- Other



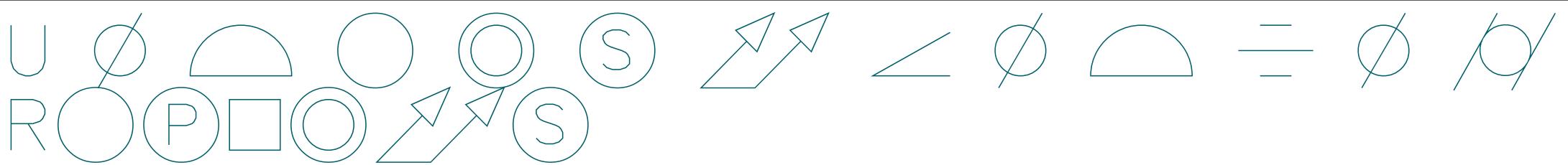


- **Analyze** The measure or aggregated field you want to analyze
 - **Explain by** The dimensions you want to analyze by
 - **Tooltips** The measures or aggregated fields you want to show when hovering over a bar





A ^B _C Stock Item	A ^B _C Extract key phrases	A ^B _C Extract key phrases.KeyPhrase
<ul style="list-style-type: none"> ● Valid 100% ● Error 0% ● Empty 0%  <p>228 distinct, 45 unique</p>	<ul style="list-style-type: none"> ● Valid 99% ● Error 0% ● Empty < 1%  <p>177 distinct, 34 unique</p>	<ul style="list-style-type: none"> ● Valid 99% ● Error 0% ● Empty < 1%  <p>163 distinct, 66 unique</p>
1. Animal with big feet slippers (Brown) XL	Brown, big feet slippers, Animal, XL	Brown
2. Animal with big feet slippers (Brown) XL	Brown, big feet slippers, Animal, XL	big feet slippers
3. Animal with big feet slippers (Brown) XL	Brown, big feet slippers, Animal, XL	Animal
4. Animal with big feet slippers (Brown) XL	Brown, big feet slippers, Animal, XL	XL
5. Animal with big feet slippers (Brown) L	Brown, Animal, big feet slippers	Brown
6. Animal with big feet slippers (Brown) L	Brown, Animal, big feet slippers	Animal
7. Animal with big feet slippers (Brown) L	Brown, Animal, big feet slippers	big feet slippers
8. Animal with big feet slippers (Brown) M	Brown, Animal, big feet slippers	Brown
9. Animal with big feet slippers (Brown) M	Brown, Animal, big feet slippers	Animal
10. Animal with big feet slippers (Brown) M	Brown, Animal, big feet slippers	big feet slippers
11. Animal with big feet slippers (Brown) S	Brown, big feet slippers, Animal	Brown
12. Animal with big feet slippers (Brown) S	Brown, big feet slippers, Animal	big feet slippers
13. Animal with big feet slippers (Brown) S	Brown, big feet slippers, Animal	Animal
14. Ogre battery-powered slippers (Green) XL	Green, Ogre battery-powered slippers, XL	Green
15. Ogre battery-powered slippers (Green) XL	Green, Ogre battery-powered slippers, XL	Ogre battery-powered slippers



- **Power BI native reports** – This report type delivers a highly visual and interactive report that has its roots in Power View. This is the report type I'll mean when I refer to Power BI reports.
- **Excel reports** – Power BI allows you to connect to Excel 2013 (or later) files and view the included table, pivot, and Power View reports.
- **Paginated (Reporting Services) reports** – SSRS is Microsoft's most customizable reporting tool for creating paper-oriented (paginated) reports. As a business user, you can view published paginated reports in Power BI Service.



Pages



File Export Share Chat in Teams Get insights Subscribe Edit ...



Overview

District Monthly Sales

New Stores

District Sales Report

Pages Pane

Context Menu

New Stores Analysis

Filters Pane

Filters

Search

Filters on this page

Chain
is (All)City
is (All)Name
is (All)Open Month
is (All)Store Type
is New Store

This Year Sales by City and Chain



Open Store Count by Open Month and Chain



Sales Per Sq Ft by Name

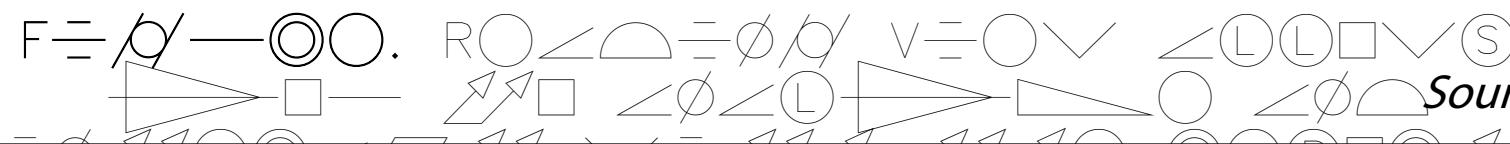


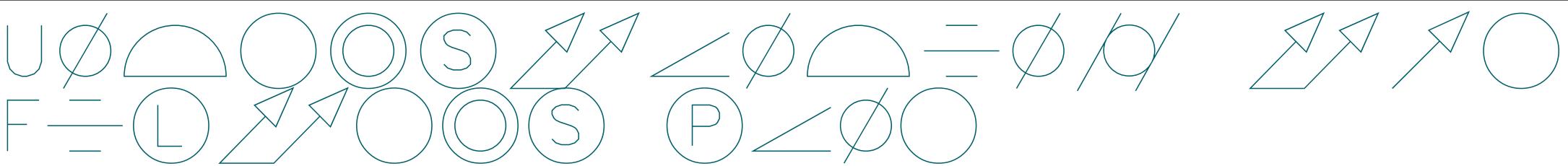
This Year Sales by FiscalMonth



- 5
- Cincinnati 2 Fashions Direct
 - Ft. Oglethorpe Lindseys
 - Knoxville Lindseys
 - Monroeville Fashions Direct
 - Pasadena Lindseys
 - Sharonville Fashions Direct
 - Washington Fashions Direct
 - Wilson Lindseys
 - Winchester Fashions Direct
 - York Fashions Direct

Source: Lachev & Price (2018)





Filters

Sales Per Sq Ft
is (All)

Search

Filters on this page

Filters on this visual

Name
is (All)

Filter type

Basic filtering

Search

Select all

Cincinnati 2 Fashions... 1

Ft. Oglethorpe Linds... 1

Knoxville Lindseys 1

Monroeville Fashion... 1

Pasadena Lindseys 1

Sharonville Fashions ... 1

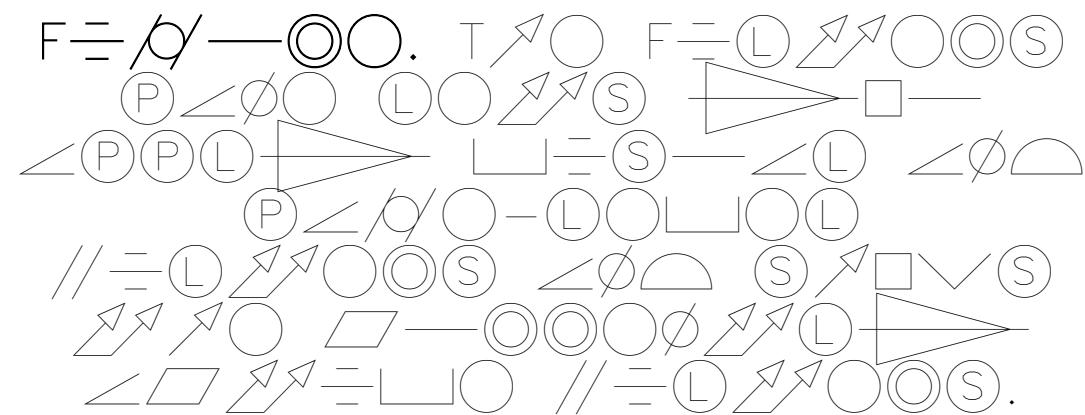
Chain
is (All)

City
is (All)

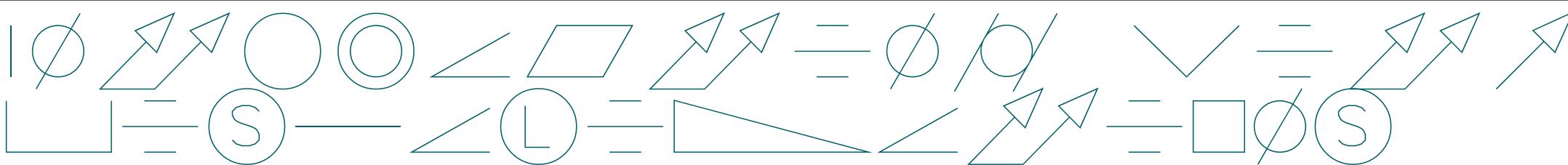
Name
is (All)

Open Month
is (All)

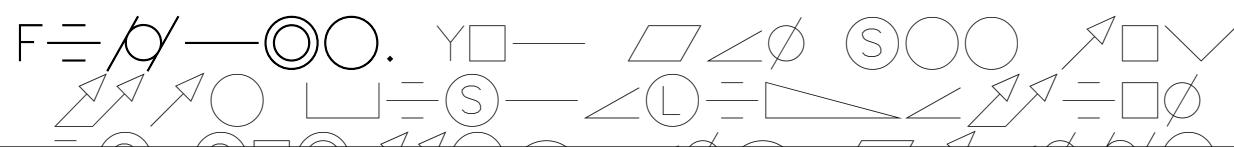
Store Type
is New Store



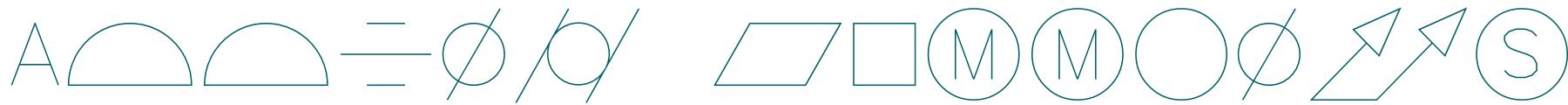
Source: Lachev & Price (2018)



Sales Per Sq Ft by Name

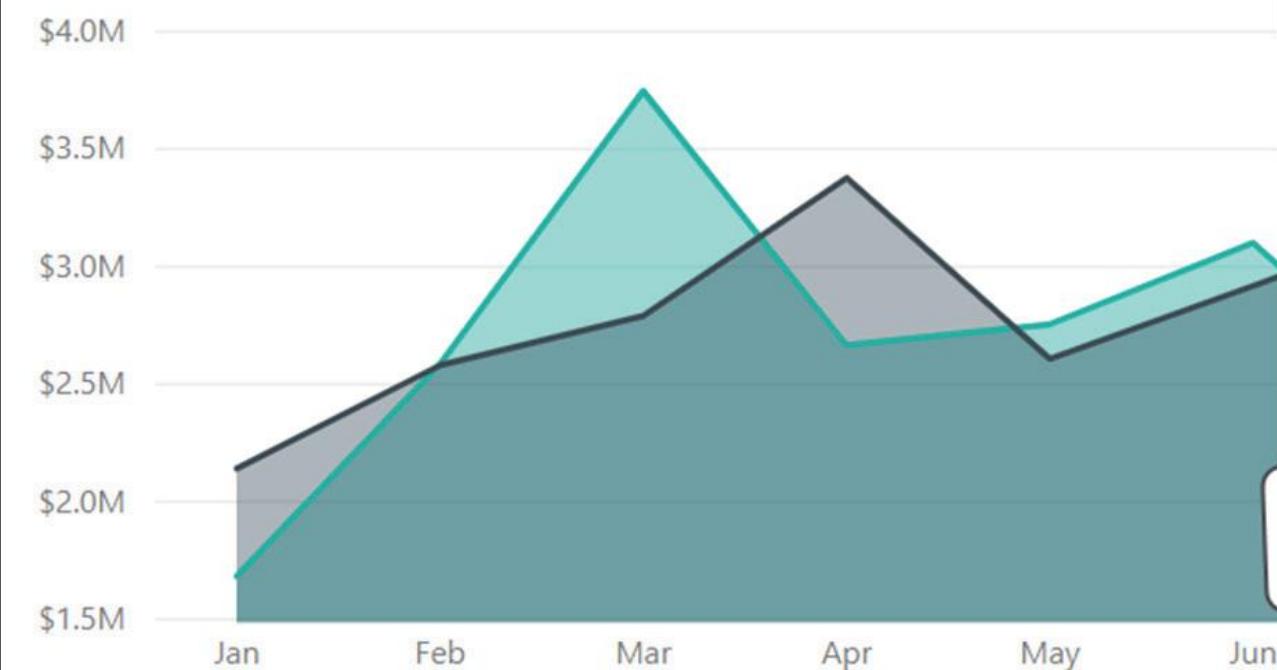


Source: Lachev & Price (2018)



This Year Sales and Last Year Sales by FiscalMonth

This Year Sales Last Year Sales



RETAIL ANALYSIS SAMPLE

All comments

Enter your comments here, and @mention people to grab their attention.

Post



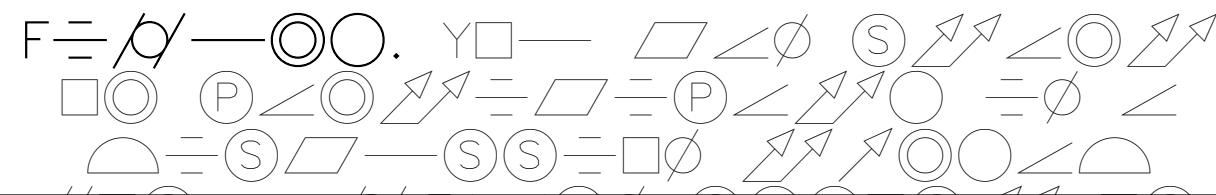
Teo Lachev

now

Great job, @Maya Lachev!

Reply

Visual-level comment



Source: Lachev & Price (2018)

E

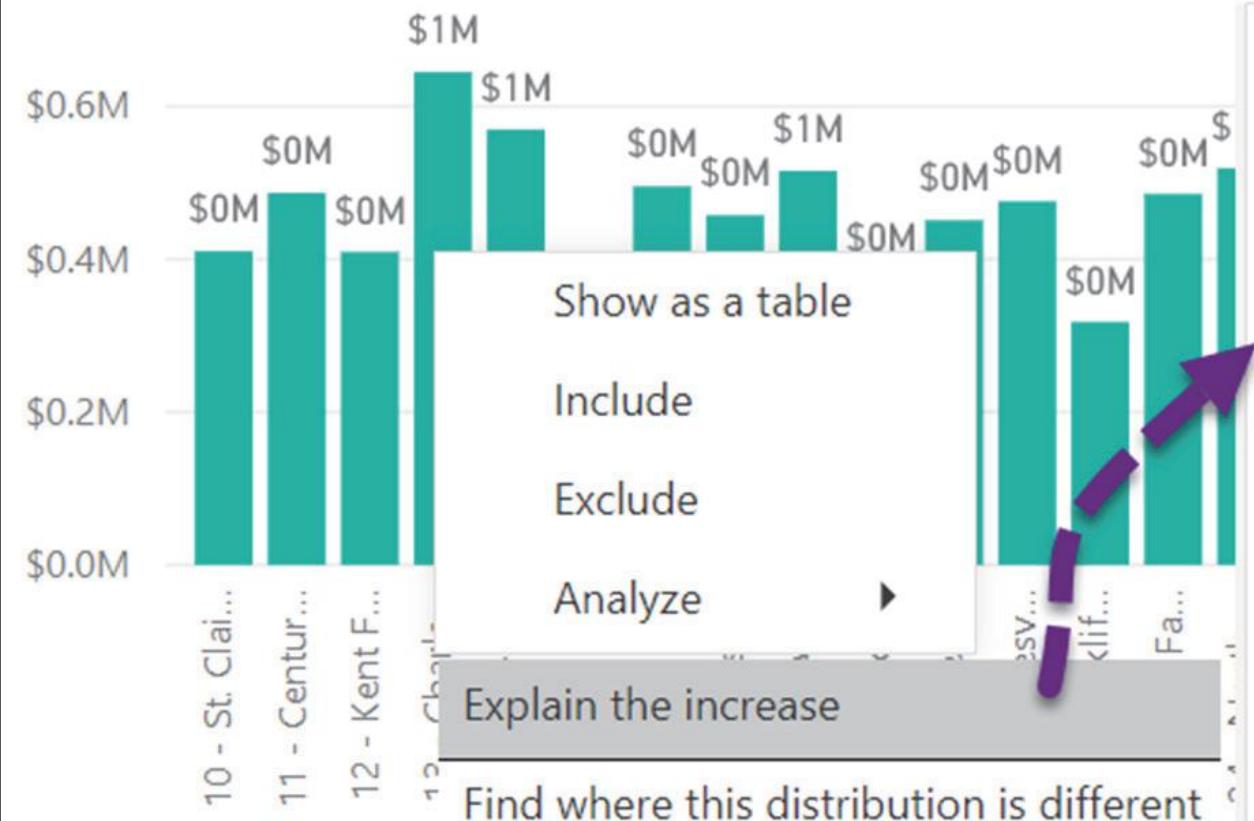
↓

P

L

O

This Year Sales by StoreNumberName



Here's the analysis of the 57.50% increase in This Year Sales between 12 and 13

This Year Sales

BY LOCATIONID AND TERRITORY

'WV' accounted for the majority of the increase among Territory, offsetting the decrease of 'OH'.

● Increase ● Decrease ● Total ● Other



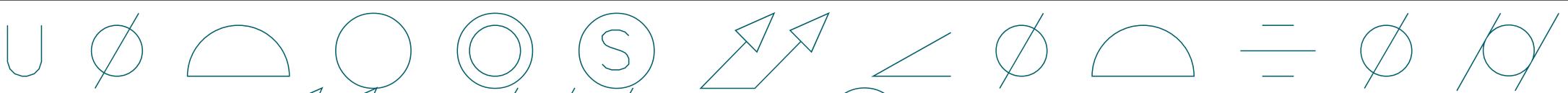
This Year Sales

BY LOCATIONID AND DISTRICT MANAGER

ⓘ This feature is in preview. [Learn more](#)



Source: Lachev & Price (2018)



File ▾ View ▾ Reading view Mobile layout

Ask a question Explore ▾ Text box Refresh Duplicate this page Save Pin to a dashboard Chat in Teams ...

New Stores An

This Year Sales by City and Chain



Open Store Count by Op



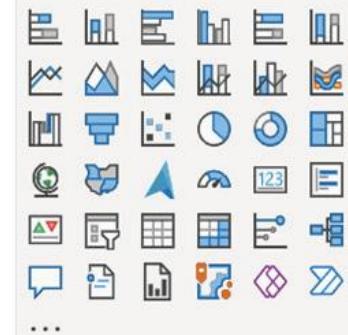
Sales Per Sq Ft by Territory



Name

Cincinnati 2 Fashions Direct

Visualizations



Filters

Values

Add data fields here

Drill through

Cross-report

Off

Keep all filters

On

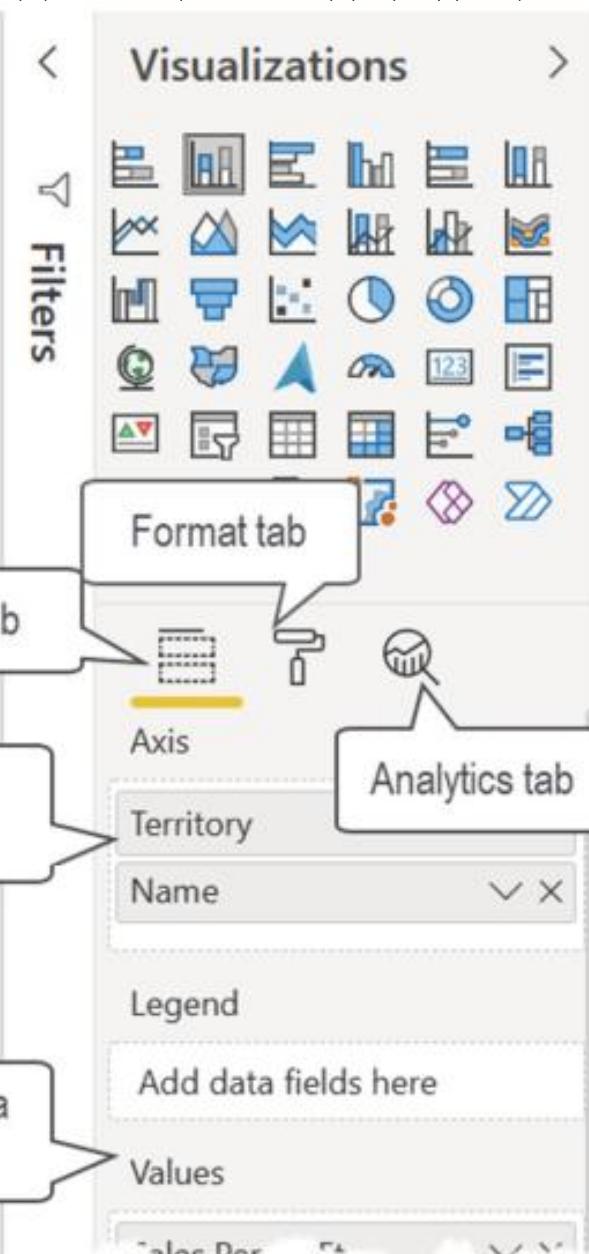
Add drill-through fields here

Fields

Search

- > Sales
- > District
- > Item
- > Store
- > Time

Source: Lachev & Price (2018)



Source: Lachev & Price (2018)

Fields



Search

> Sales

> District

> Item

> Store

Average Selling Ar

Chain

City

Count of OpenDat

Name

New Stores

✓ Check

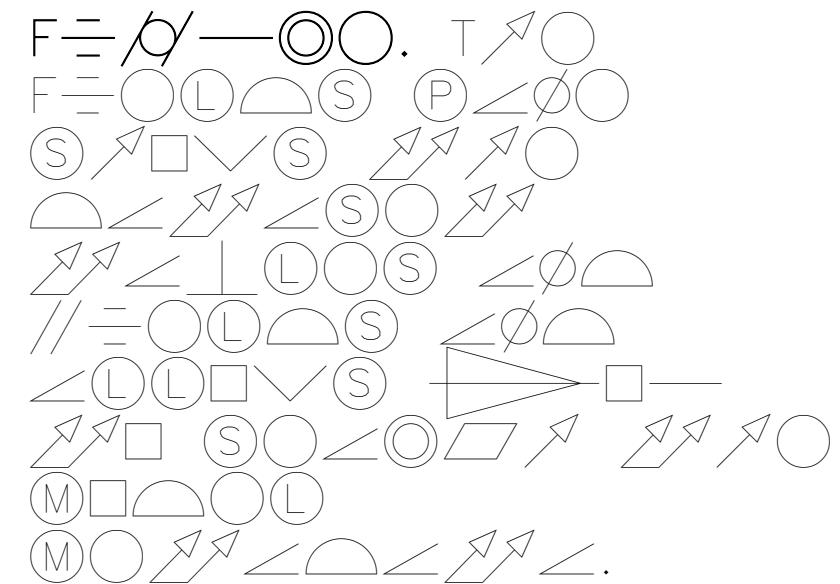
Collapse all

Expand all

Add to filters

Add to drill through

...

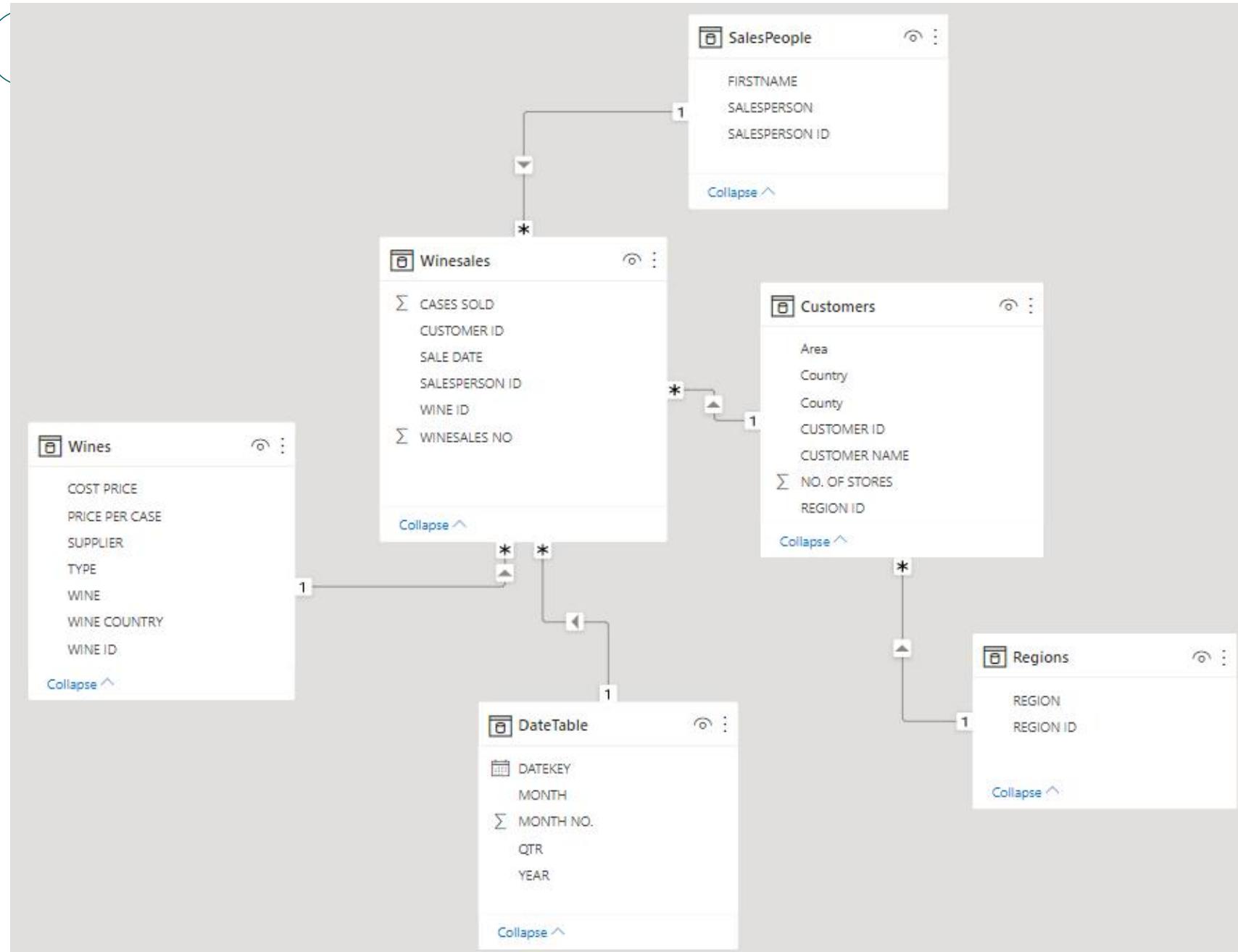


Source: Lachev & Price (2018)



- We've imported six tables into Power BI Desktop as follows:
 - **Winesales** – Records our sales transactions.
 - **Wines** – Records the names and details of the wines we sell.
 - **Customers** – Who we sell our wines to.
 - **SalesPeople** – The people making the sales.
 - **Regions** – Our customers are grouped into these regions.
 - **DateTable** – Records every date, starting from the first day of the month when sales start and ending with the last date in the current financial year, categorizing these dates into year, quarter, and month.

- You will observe that the DateTable, SalesPeople, Customers, and Wines tables are all related to the Winesales table. Notice the “1” and “*” to denote the one side and the many side, respectively. The columns used to create the relationships have the same column names in both tables; for example, WINE ID in the Wines table is related to WINE ID in the Winesales table. The Regions table is the odd one out in that it’s not directly related to the Winesales table but *indirectly* via the Customers table.
- In a Power BI data model, a table should be either one of two types, either a *fact table* or a *dimension* as described in the following sections.





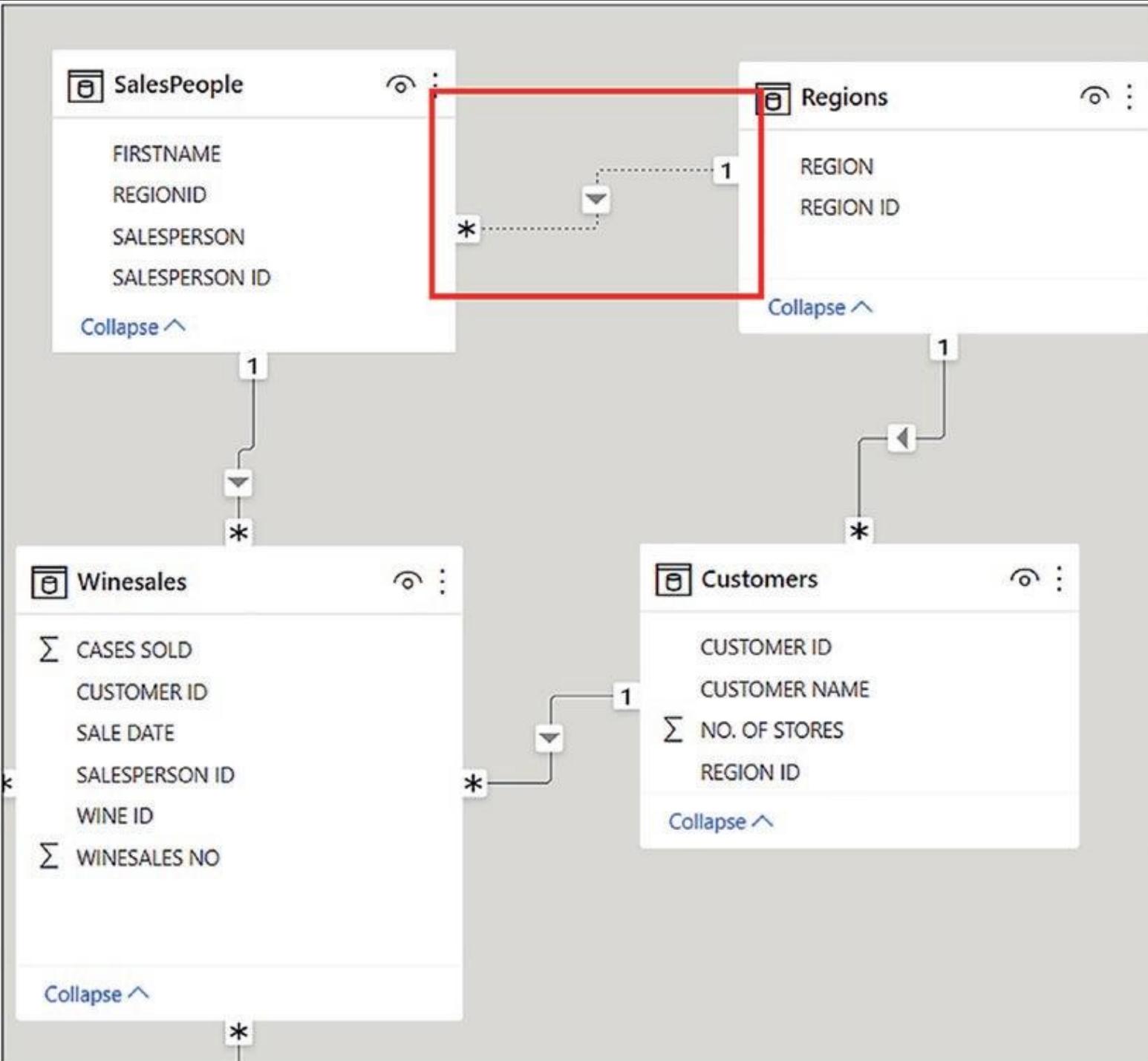
- This type of table stores “events.” The term “event” is used loosely here to describe activities such as sales, orders, or survey results. Fact tables answer the question *what*? That is, what are you analyzing in your report? You can identify the fact table by asking yourself these three questions:
 1. Which table holds the data that you want to analyze in your report?
 2. If you delete this table, will the remaining tables still be related to other tables in the data model?
 3. Which table sits on the many side of all the other relationships?

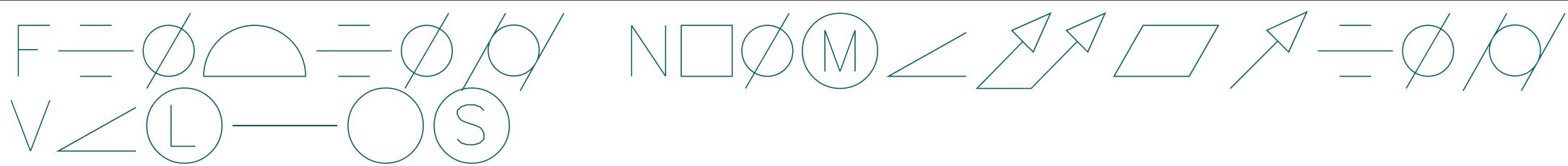
D = M Ø S = □ Ø S

- These tables store **the descriptions of the entities** in your model. Dimensions answer the question **how**? That is, how do you want to analyze your data? In our data model, we can analyze our sales by wines, salespeople, customers, regions, and dates using the data in the columns within these tables. The data in dimensions does not necessarily change regularly, and dimensions tend to have fewer rows than fact tables.
- There's no table property that you set to configure the table type as a dimension or a fact table. It's determined by which side of the relationship the table sits on. Tables that sit on the "one" side are always dimension-type tables, while tables that are *only* related on the "many" side are fact tables.
- The reason it's so important to distinguish between these two different types of tables is that they support two different types of behavior in the data model, as follows:
 - Dimension tables support *grouping* and *filtering*.
 - Fact tables support *summarization*.

T ↗ O S ↗ ↘

- You'll notice in Model view that we have dimensions around the fact table. This is called a star structure, *star schema*. In a perfect schema, there would be no dimensions in the fact table.
- Dimensions that are not directly related to the fact table are described as *snowflake* dimensions. These are created by adding dimensions in chains outward from the fact table.
- One thing that Power BI prevents is having multiple relationships between tables, which filters can propagate. Therefore, this will result in an *inactive* relationship.





- A question that is often asked is what happens when there are missing values in the linking columns used to create relationships. There are two different scenarios here, taking the Wines dimension as our example:
 1. You have values in the WINE ID column in the Wines dimension that don't exist in the WINE ID column in the Winesales fact table.
 2. You have values in the WINE ID column in the Winesales fact table that don't exist in the WINE ID column of the Wines dimension.

Understanding this situation allows us to answer the following question: **Which wines haven't we sold?** When you build a visual that takes a column from a dimension and summarizes a column from the fact table, you will only see items where there's a match for values in the linking columns. By default, all visuals remove items where there is no value to show.



WINE	Total Cases
Bordeaux	54,070
Champagne	49,158
Sauvignon Blanc	47,415
Chardonnay	42,030
Grenache	35,965
Malbec	34,290
Rioja	33,951
Chianti	27,323
Chenin Blanc	24,739
Pinot Grigio	23,449
Merlot	23,084
Shiraz	17,497
Piesporter	10,253
Total	423,224

Columns

WINE

Total Cases

Drill through

Cross-report

Keep all filters

Add drill-through fields here

- Remove field
- Rename for this visual
- Move >
- Conditional formatting >
- Remove conditional formatting
- ✓ Don't summarize
- First
- Last
- Count (Distinct)
- Count
- New quick measure
- Show items with no data
- New group

WINE	Total Cases
Bordeaux	54,070
Champagne	49,158
Sauvignon Blanc	47,415
Chardonnay	42,030
Grenache	35,965
Malbec	34,290
Rioja	33,951
Chianti	27,323
Chenin Blanc	24,739
Pinot Grigio	23,449
Merlot	23,084
Shiraz	17,497
Piesporter	10,253
Lambrusco	
Total	423,224

WINE ID	WINE	SUPPLIER	TYPE	WINE COUNTRY	PRICE PER CASE	COST PRICE
1	Bordeaux	Laithwaites	Red	France	\$75.00	\$25.00
2	Champagne	Laithwaites	White	France	\$150.00	\$100.00
3	Chardonnay	Alliance	White	France	\$100.00	\$75.00
4	Malbec	Laithwaites	Red	Germany	\$85.00	\$40.00
5	Grenache	Redsky	Red	France	\$30.00	\$10.00
6	Piesporter	Redsky	White	Germany	\$135.00	\$50.00
7	Chianti	Redsky	Red	Germany	\$40.00	\$10.00
8	Pinot Grigio	Majestic	White	Italy	\$30.00	\$5.00
9	Merlot	Majestic	Red	France	\$39.00	\$15.00
10	Sauvignon Blanc	Majestic	White	Italy	\$40.00	\$20.00
11	Rioja	Majestic	Red	Italy		
12	Chenin Blanc	Alliance	White	France	12/30/2021	2219
13	Shiraz	Alliance	Red	France	12/30/2021	2219
14	Lambrusco	Alliance	White	Italy	12/30/2021	2219



SALE DATE	WINESALES NO	SALESPERSON ID	CUSTOMER ID	WINE ID	CASES SOLD
12/30/2021	2219	6	25		Sort ascending
12/30/2021	2219	6	25		Sort descending
12/30/2021	2219	6	25		Clear sort
12/27/2021	2216	6	36		Clear filter
12/24/2021	2209	6	80		Clear all filters
12/24/2021	2208	6	24		Number filters
12/13/2021	2182	6	7		
12/4/2021	2212	6	3		
11/27/2021	2153	6	5		
11/26/2021	2150	6	29		
11/26/2021	2151	6	11		
11/25/2021	2148	6	51		
11/23/2021	2147	6	2		
11/17/2021	2141	6	55		
11/13/2021	2132	6	12		
11/12/2021	2129	6	16		
11/4/2021	2119	6	34		
11/4/2021	2118	6	58		
10/28/2021	2114	6	26		
10/24/2021	2106	6	12		
10/24/2021	2105	6	12		
10/24/2021	2107	6	12		

- (Select all)
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13

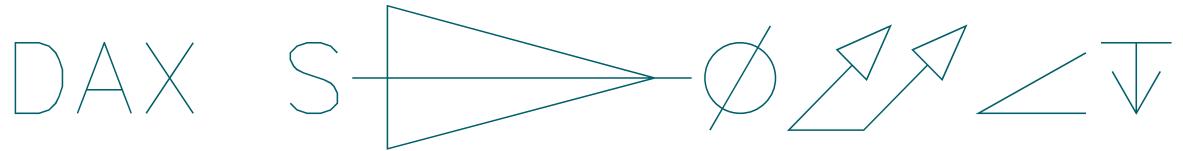
OK

Cancel

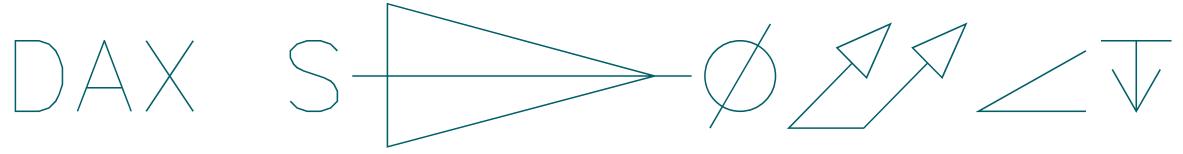
PERSON ID	CUSTOMER ID	WINE ID	CASES SOLD
6	35	100	
3	16	444	
4	20	555	
1	12	100	
2	17	133	
3	45	11	
6	11	7	
2	75	13	

- The “Blank” entry is a result of what we sometimes refer to as “dirty data.” Why are there values in the fact table for which there is no match in the dimension? How are you going to resolve this scenario? This is a question that only the data modeler can answer, and ultimately the solution lies in correcting the data at its source.

WINE	CASES SOLD	WINE
	1,020	<input type="checkbox"/> (Blank)
Bordeaux	54,070	<input type="checkbox"/> Bordeaux
Champagne	49,158	<input type="checkbox"/> Champagne
Chardonnay	41,883	<input type="checkbox"/> Chardonnay
Chenin Blanc	24,739	<input type="checkbox"/> Chenin Blanc
Chianti	27,323	<input type="checkbox"/> Chianti
Grenache	35,895	<input type="checkbox"/> Grenache
Malbec	33,964	<input type="checkbox"/> Lambrusco
Merlot	23,084	<input type="checkbox"/> Malbec
Piesporter	10,253	<input type="checkbox"/> Merlot
Pinot Grigio	23,449	<input type="checkbox"/> Piesporter
Rioja	33,951	<input type="checkbox"/> Pinot Grigio
Sauvignon Blanc	46,938	<input type="checkbox"/> Rioja
Shiraz	17,497	<input type="checkbox"/> Sauvignon Blanc
Total	423,224	<input type="checkbox"/> Shiraz

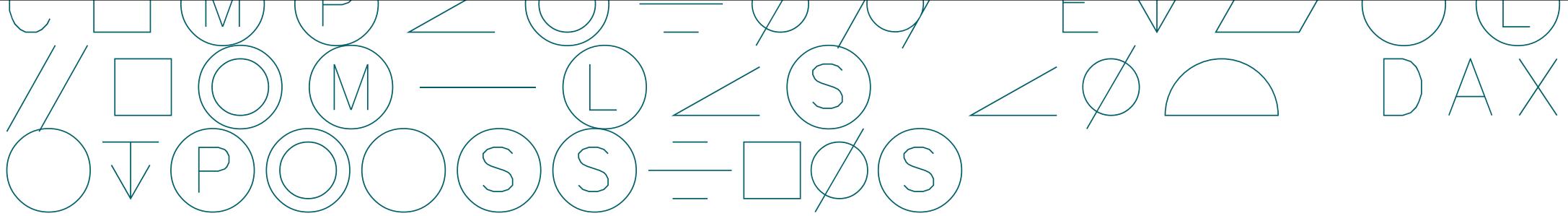


- The formula bar is in effect a code editor and can extend to many lines if the **SHIFT + ENTER** key combination is pressed.
- Also, just like Excel, DAX expressions are *case insensitive*; it makes no difference in what case you type your DAX code.
- However, one of the major differences between DAX and Excel is that in DAX, you can't reference "cells".
- = COUNTROWS (Winesales)
- The DAX IntelliSense list
- Typing spaces is arbitrary as they will be ignored by the DAX editor and can be used wherever you feel they improve the clarity of the expression.



- To reference a column, you surround the column name with **square brackets ([])** and *always precede the column name with the table name*. For example, to sum the CASES SOLD column in the Winesales table, this would be the expression:

```
= SUM ( Winesales[CASES SOLD] )
```



Excel

=IF (B2 > 50 , “Yes” , “No”)

When this formula is copied down, the “B2” will change relatively to “B3, B4, B5 etc.”

= SUM (Winesales[CASES SOLD])

This uses Excel Table syntax where the table is named “Winesales” and the column is named “CASES SOLD”.

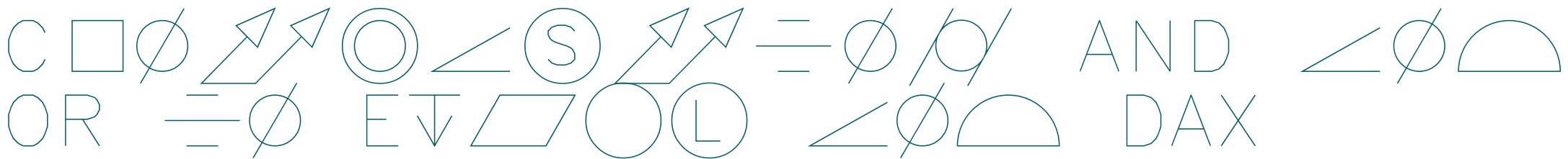
DAX

=IF (Winesales[CASES SOLD] > 50 , “Yes” , “No”)

Used in a calculated column, this expression is automatically applied to the entire column.

= SUM (Winesales[CASES SOLD])

Used in a measure or in a calculated column to find total cases in the CASES SOLD column in the Winesales table.



Excel

AND

```
= IF ( AND ( Winesales[@CASES SOLD] > 50,
Winesales[@CASES SOLD] < 100 ), "Yes" , "No" )
```

Using Excel Table syntax where the table is named "Winesales" and the column is named "CASES SOLD"
Note the use of the "@" to denote "the current row."

OR

```
= IF ( OR ( Winesales[@SALESPERSON ID] = 2 ,
Winesales[@SALESPERSON ID] = 6 ),
"Yes" , "No" )
```

Using Excel Table syntax where the table is named "Winesales" and the column is named "SALESPERSON ID"

DAX

AND

```
= IF ( Winesales[CASES SOLD] > 50
&&
Winesales[CASES SOLD] < 100 ,
"Yes" , "No" )
```

Used in a calculated column.
Using the value in the current row is implicit in calculated columns.

OR

```
= IF ( Winesales[SALESPERSON ID] = 2
|| 
Winesales[SALESPERSON ID] = 6 , "Yes" ,
"No" )
```

Used in a calculated column.

DAX FOOOM <→ = Ø Ø

X ✓ 1 Average Cases for Black Ltd in 2019 = CALCULATE(AVERAGE(Winesales[CASES SOLD]),DateTable[Year]=2019,Customers[CUSTOMER NAME] = "black ltd")
2

X ✓ 1 Average Cases for Black Ltd in 2019 =
2 CALCULATE (
3 AVERAGE (Winesales[CASES SOLD]),
4 DateTable[Year] = 2019,
5 Customers[CUSTOMER NAME] = "black ltd"
6)
7

Comparing unformatted and formatted expressions

DAX FOMO

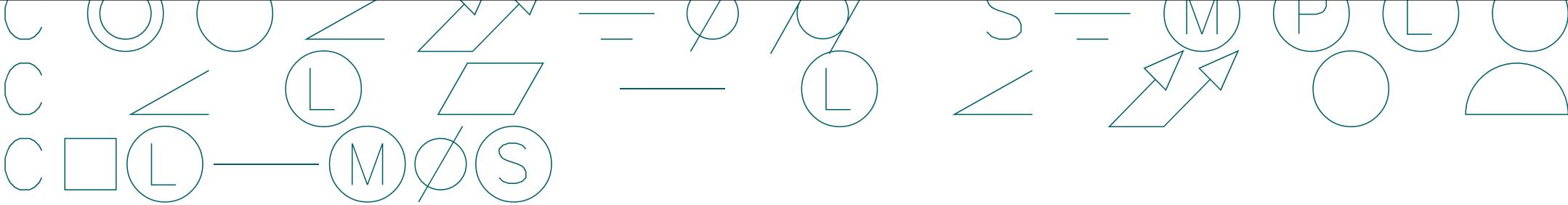
- To add comments to your code, use the following:

-- – Single line comment (double dash)

// – Single line comment (double forward slash)

/* – Start a multiline comment (forward slash and asterisk)

*/ – End a multiline comment (asterisk and forward slash)



- To create our first DAX expression in a calculated column, let's take a very simple calculation and multiply the CASES SOLD values in the Winesales table by 10 percent.

10 PC of Cases = Winesales[CASES SOLD] * 0.1

X ✓ 1 10 PC of Cases = Winesales[CASES SOLD] * 0.1

SALE DATE	WINESALES NO	SALESPERSON ID	CUSTOMER ID	WINE ID	CASES SOLD	ORDER DATE	10 PC of Cases
Saturday, May 23, 2020	1167	6	16	2	500	Saturday, May 16, 2020	50
Wednesday, December 20, 2017	311	6	15	1	498	Wednesday, December 13, 2017	49.8
Sunday, August 13, 2017	186	6	10	2	491	Sunday, August 6, 2017	49.1
Sunday, July 5, 2020	1245	6	21	1	491	Sunday, June 28, 2020	49.1
Sunday, November 8, 2020	1455	6	35	1	485	Sunday, November 1, 2020	48.5
Wednesday, February 21, 2018	366	6	32	1	476	Wednesday, February 14, 2018	47.6
Thursday, July 25, 2019	770	6	29	1	471	Thursday, July 18, 2019	47.1
Wednesday, February 26, 2020	1032	6	23	1	463	Wednesday, February 19, 2020	46.3
Monday, December 2, 2019	940	6	10	1	435	Monday, November 25, 2019	43.5
Saturday, April 6, 2019	677	6	75	1	433	Saturday, March 30, 2019	43.3
Saturday, September 5, 2020	1349	6	35	2	424	Saturday, August 29, 2020	42.4
Sunday, March 21, 2021	1703	6	5	1	421	Sunday, March 14, 2021	42.1
Saturday, April 18, 2020	1100	6	24	1	409	Saturday, April 11, 2020	40.9
Wednesday, July 14, 2021	1923	6	20	2	398	Wednesday, July 7, 2021	39.8
Friday, March 31, 2017	87	6	34	2	383	Friday, March 24, 2017	38.3
Monday, September 7, 2020	1355	6	12	2	383	Monday, August 31, 2020	38.3
Tuesday, January 9, 2018	333	6	28	1	381	Tuesday, January 2, 2018	38.1
Thursday, July 25, 2019	769	6	29	1	381	Thursday, July 18, 2019	38.1
Thursday, July 25, 2019	771	6	29	1	380	Thursday, July 18, 2019	38
Friday, July 21, 2017	175	6	33	2	378	Friday, July 14, 2017	37.8
Tuesday, October 3, 2017	233	6	10	1	376	Tuesday, September 26, 2017	37.6

Fields

Search

> Measures Table

> Customers

> DateTable

> DateTable Compare

> Regions

> SalesPeople

> Wines

> Wines Compare

✓ Winesales

10 PC of Cases

Σ CASES SOLD

CUSTOMER ID

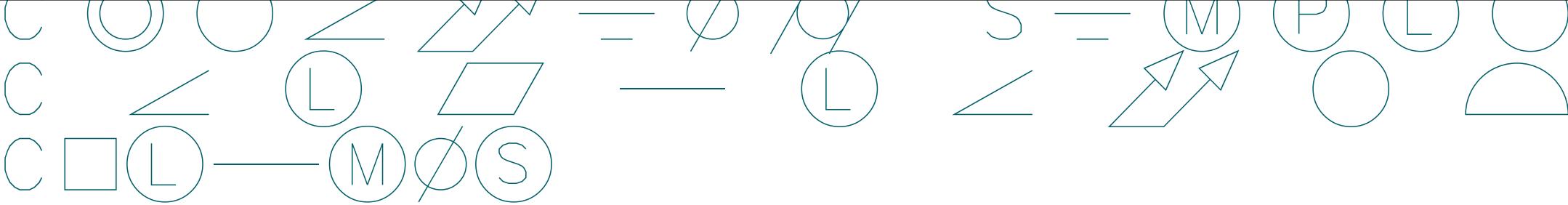
> ORDER DATE

SALE DATE

SALESPERSON ID

WINE ID

Σ WINESALES NO



- Let's now see how we can use the IF function in DAX in a calculated column. We could, for instance, add a column in the Winesales table that's populated with either "Team A" or "Team B." This column will group our salespeople as follows: Salespeople with IDs 1, 3, and 6 are in Team A, and other salespeople are in Team B. We'll call this new column "**Team**".

Team =

```
IF (
    Winesales[SALESPERSON ID] = 1
    || Winesales[SALESPERSON ID] = 3
    || Winesales[SALESPERSON ID] = 6,
    "Team A",
    "Team B"
)
```

Fields

Search	
Measures Table	
Customers	
DateTable	
DataTable Compare	
Regions	
SalesPeople	
Wines	
Wines Compare	
Winesales	10 PC of Cases
CASES SOLD	
CUSTOMER ID	
ORDER DATE	
SALE DATE	
SALESPERSON ID	
Team	
WINE ID	
WINESALES NO	

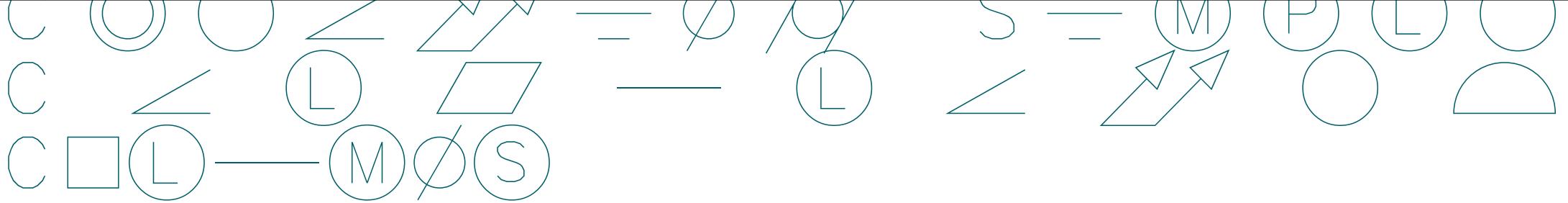
Team =

```

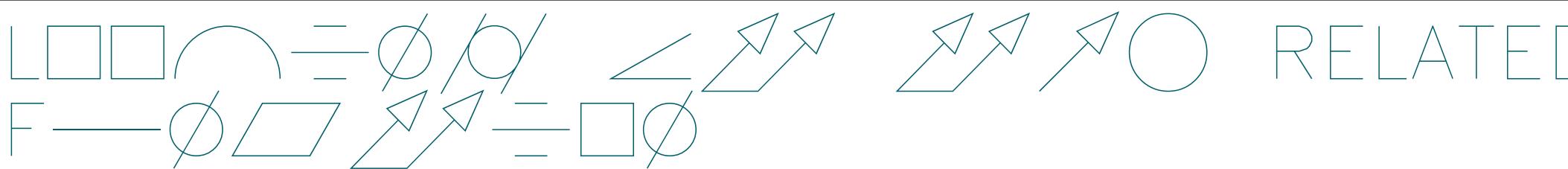
1 Team =
2 IF (
3     Winesales[SALESPERSON ID] = 1
4     || Winesales[SALESPERSON ID] = 3
5     || Winesales[SALESPERSON ID] = 6,
6     "Team A",
7     "Team B"
8 )

```

SALE DATE	WINESALES NO	SALESPERSON ID	CUSTOMER ID	WINE ID	CASES SOLD	ORDER DATE	10 PC of Cases	Team
Saturday, May 23, 2020	1167	6	16	2	500	Saturday, May 16, 2020	50	Team A
Wednesday, December 20, 2017	311	6	15	1	498	Wednesday, December 13, 2017	49.8	Team A
Sunday, August 13, 2017	186	6	10	2	491	Sunday, August 6, 2017	49.1	Team A
Sunday, July 5, 2020	1245	6	21	1	491	Sunday, June 28, 2020	49.1	Team A
Sunday, November 8, 2020	1455	6	35	1	485	Sunday, November 1, 2020	48.5	Team A
Wednesday, February 21, 2018	366	6	32	1	476	Wednesday, February 14, 2018	47.6	Team A
Thursday, July 25, 2019	770	6	29	1	471	Thursday, July 18, 2019	47.1	Team A
Wednesday, February 26, 2020	1032	6	23	1	463	Wednesday, February 19, 2020	46.3	Team A
Monday, December 2, 2019	940	6	10	1	435	Monday, November 25, 2019	43.5	Team A
Saturday, April 6, 2019	677	6	75	1	433	Saturday, March 30, 2019	43.3	Team A
Saturday, September 5, 2020	1349	6	35	2	424	Saturday, August 29, 2020	42.4	Team A
Sunday, March 21, 2021	1703	6	5	1	421	Sunday, March 14, 2021	42.1	Team A
Saturday, April 18, 2020	1100	6	24	1	409	Saturday, April 11, 2020	40.9	Team A
Wednesday, July 14, 2021	1923	6	20	2	398	Wednesday, July 7, 2021	39.8	Team A
Friday, March 31, 2017	87	6	34	2	383	Friday, March 24, 2017	38.3	Team A
Monday, September 7, 2020	1355	6	12	2	383	Monday, August 31, 2020	38.3	Team A



- Similarly, you could group the values in the CASES SOLD column into “High” and “Low” volume where high **volume** is any sales where CASES SOLD is between 50 and 500 by using this DAX expression, noting the use of the double ampersand for “AND” (**Question**).



RELATED

- This function returns a value from a related table and is similar in purpose to the VLOOKUP function in Excel. However, RELATED will only return values from the *one side* of the relationship to the many side. For example, if you want to show the *customer names* related to the **CUSTOMER ID's** in the Winesales table, you could use this DAX expression in a calculated column in the Winesales table:

Customer Name from Customers Table = RELATED (Customers[CUSTOMER NAME])

Customer Name from Customers Table = RELATED (Customers[CUSTOMER NAME])											
SALE DATE	WINESALES NO	SALESPERSON ID	CUSTOMER ID	WINE ID	CASES SOLD	ORDER DATE	10 PC of Cases	Team	Volume	Customer Name from Customers Table	
Saturday, May 23, 2020	1167	6	16	2	500	Saturday, May 16, 2020	50	Team A	Low	Port Hammond Bros	
Wednesday, December 20, 2017	311	6	15	1	498	Wednesday, December 13, 2017	49.8	Team A	Low	Littleton & Sons	
Sunday, August 13, 2017	186	6	10	2	491	Sunday, August 6, 2017	49.1	Team A	Low	Lavender Bay Ltd	
Sunday, July 5, 2020	1245	6	21	1	491	Sunday, June 28, 2020	49.1	Team A	Low	Burningsuit Ltd	
Sunday, November 8, 2020	1455	6	35	1	485	Sunday, November 1, 2020	48.5	Team A	Low	Ellenburg Ltd	
Wednesday, February 21, 2018	366	6	32	1	476	Wednesday, February 14, 2018	47.6	Team A	Low	Sapporo & Co	
Thursday, July 25, 2019	770	6	29	1	471	Thursday, July 18, 2019	47.1	Team A	Low	Spokane Ltd	
Wednesday, February 26, 2020	1032	6	23	1	463	Wednesday, February 19, 2020	46.3	Team A	Low	Villeneuve-d'Ascq Ltd	
Monday, December 2, 2019	940	6	10	1	435	Monday, November 25, 2019	43.5	Team A	Low	Lavender Bay Ltd	
Saturday, April 6, 2019	677	6	75	1	433	Saturday, March 30, 2019	43.3	Team A	Low	Saint Germain en Laye & Co	
Saturday, September 5, 2020	1349	6	35	2	424	Saturday, August 29, 2020	42.4	Team A	Low	Ellenburg Ltd	
Sunday, March 21, 2021	1703	6	5	1	421	Sunday, March 14, 2021	42.1	Team A	Low	Snoqualmie & Sons	
Saturday, April 18, 2020	1100	6	24	1	409	Saturday, April 11, 2020	40.9	Team A	Low	Charleston Ltd	
Wednesday, July 14, 2021	1923	6	20	2	398	Wednesday, July 7, 2021	39.8	Team A	High	Clifton Ltd	
Friday, March 31, 2017	87	6	34	2	383	Friday, March 24, 2017	38.3	Team A	High	Fort Atkinson & Co	
Monday, September 7, 2020	1355	6	12	2	383	Monday, August 31, 2020	38.3	Team A	High	El Cajon & Sons	
Tuesday, January 9, 2018	333	6	28	1	381	Tuesday, January 2, 2018	38.1	Team A	High	Rhodes Ltd	
Thursday, July 25, 2019	769	6	29	1	381	Thursday, July 18, 2019	38.1	Team A	High	Spokane Ltd	
Thursday, July 25, 2019	771	6	29	1	380	Thursday, July 18, 2019	38	Team A	High	Spokane Ltd	
Friday, July 21, 2017	175	6	33	2	378	Friday, July 14, 2017	37.8	Team A	High	Port Orchard & Sons	
Tuesday, October 3, 2017	233	6	10	1	376	Tuesday, September 26, 2017	37.6	Team A	High	Lavender Bay Ltd	
Saturday, February 2, 2019	649	6	24	2	371	Saturday, January 26, 2019	37.1	Team A	High	Charleston Ltd	

Fields

Search

> Measures Table

> Customers

Area

Country

CUSTOMER ID

CUSTOMER NAME

Σ NO OF STORES

REGION ID

> DateTable

> DateTable Compare

> Regions

> SalesPeople

> Wines

> Wines Compare

> Winesales

Σ 10 PC of Cases

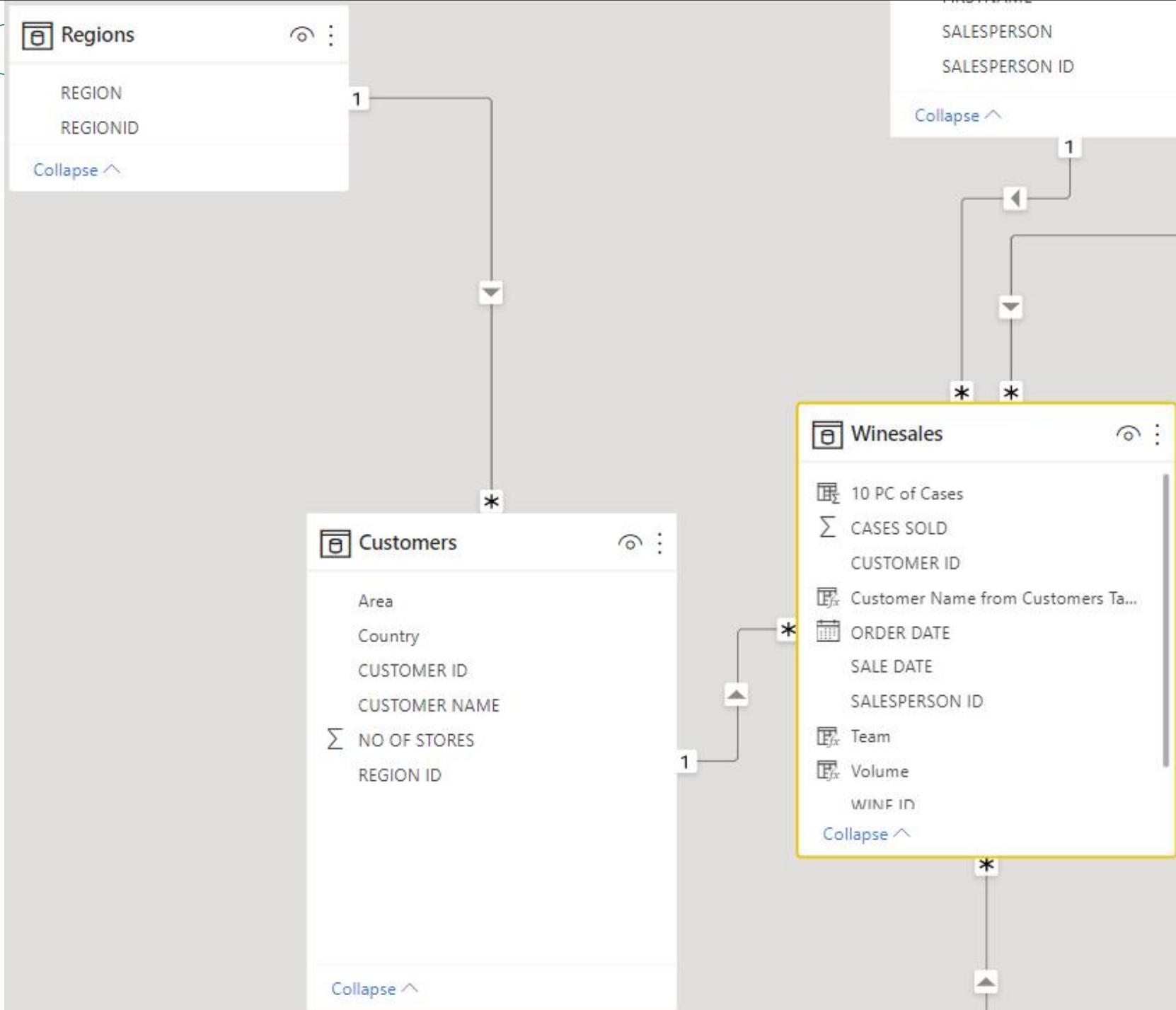
Σ CASES SOLD

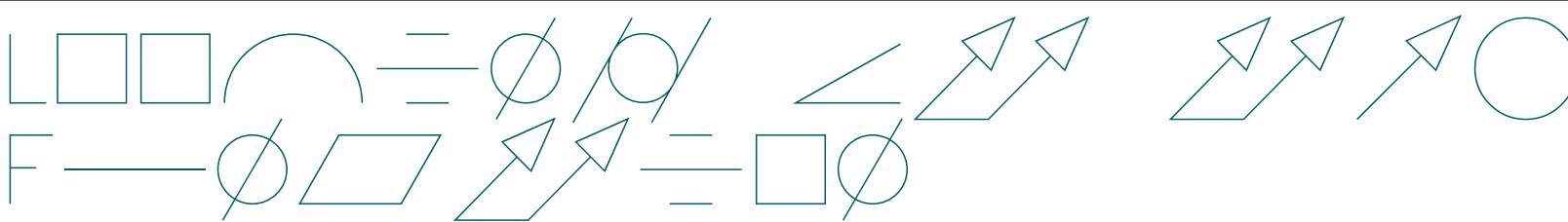
CUSTOMER ID

Customer Name from Customers Table



- You can also use RELATED to pull through values from *indirectly* related tables into the fact table. For example, the Regions table is related to the Customers table that is in turn related to the Winesales table.





RELATED

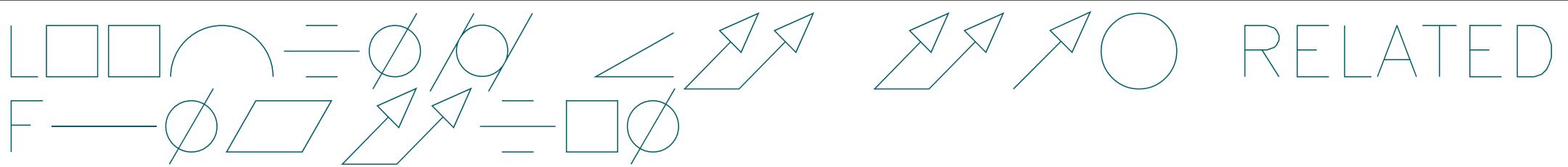
REGION NAME = RELATED (Regions[REGION])

1 REGION NAME = RELATED (Regions[REGION])

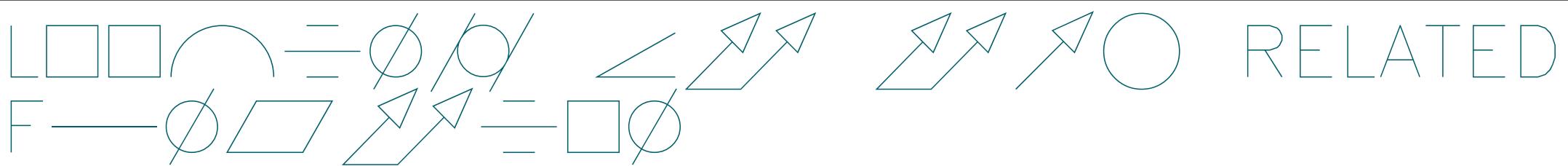
SALE DATE	WINESALES NO	SALESPERSON ID	CUSTOMER ID	WINE ID	CASES SOLD	ORDER DATE	10 PC of Cases	Team	Volume	Customer Name from Customers Table	REGION NAME
Saturday, May 23, 2020	1167	6	16	2	500	Saturday, May 16, 2020	50	Team A	Low	Port Hammond Bros	Italy
Wednesday, December 20, 2017	311	6	15	1	498	Wednesday, December 13, 2017	49.8	Team A	Low	Littleton & Sons	Scotland
Sunday, August 13, 2017	186	6	10	2	491	Sunday, August 6, 2017	49.1	Team A	Low	Lavender Bay Ltd	Czech Republic
Sunday, July 5, 2020	1245	6	21	1	491	Sunday, June 28, 2020	49.1	Team A	Low	Burningsuit Ltd	Wales
Sunday, November 8, 2020	1455	6	35	1	485	Sunday, November 1, 2020	48.5	Team A	Low	Eilenburg Ltd	New Zealand
Wednesday, February 21, 2018	366	6	32	1	476	Wednesday, February 14, 2018	47.6	Team A	Low	Sapporo & Co	France
Thursday, July 25, 2019	770	6	29	1	471	Thursday, July 18, 2019	47.1	Team A	Low	Spokane Ltd	China
Wednesday, February 26, 2020	1032	6	23	1	463	Wednesday, February 19, 2020	46.3	Team A	Low	Villeneuve-d'Ascq Ltd	Germany
Monday, December 2, 2019	940	6	10	1	435	Monday, November 25, 2019	43.5	Team A	Low	Lavender Bay Ltd	Czech Republic
Saturday, April 6, 2019	677	6	75	1	433	Saturday, March 30, 2019	43.3	Team A	Low	Saint Germain en Laye & Co	Australia
Saturday, September 5, 2020	1349	6	35	2	424	Saturday, August 29, 2020	42.4	Team A	Low	Eilenburg Ltd	New Zealand
Sunday, March 21, 2021	1703	6	5	1	421	Sunday, March 14, 2021	42.1	Team A	Low	Snoqualmie & Sons	Wales
Saturday, April 18, 2020	1100	6	24	1	409	Saturday, April 11, 2020	40.9	Team A	Low	Charleston Ltd	United States
Wednesday, July 14, 2021	1923	6	20	2	398	Wednesday, July 7, 2021	39.8	Team A	High	Clifton Ltd	New Zealand
Friday, March 31, 2017	87	6	34	2	383	Friday, March 24, 2017	38.3	Team A	High	Fort Atkinson & Co	Japan
Monday, September 7, 2020	1355	6	12	2	383	Monday, August 31, 2020	38.3	Team A	High	El Cajon & Sons	China
Tuesday, January 9, 2018	333	6	28	1	381	Tuesday, January 2, 2018	38.1	Team A	High	Rhodes Ltd	South Africa
Thursday, July 25, 2019	769	6	29	1	381	Thursday, July 18, 2019	38.1	Team A	High	Spokane Ltd	China
Thursday, July 25, 2019	771	6	29	1	380	Thursday, July 18, 2019	38	Team A	High	Spokane Ltd	China
Friday, July 21, 2017	175	6	33	2	378	Friday, July 14, 2017	37.8	Team A	High	Port Orchard & Sons	Wales
Tuesday, October 3, 2017	233	6	10	1	376	Tuesday, September 26, 2017	37.6	Team A	High	Lavender Bay Ltd	Czech Republic
Saturday, February 2, 2019	649	6	24	2	371	Saturday, January 26, 2019	37.1	Team A	High	Charleston Ltd	United States
Tuesday, October 6, 2020	1411	6	21	1	365	Tuesday, September 29, 2020	36.5	Team A	High	Burningsuit Ltd	Wales
Tuesday, August 4, 2020	1300	6	12	2	365	Tuesday, July 28, 2020	36.5	Team A	High	El Cajon & Sons	China
Monday, June 1, 2020	1182	6	33	1	358	Monday, May 25, 2020	35.8	Team A	High	Port Orchard & Sons	Wales

Fields

- Search
- > Measures Table
- ✓ Customers
 - Area
 - Country
 - CUSTOMER ID
 - CUSTOMER NAME
 - Σ NO OF STORES
 - REGION ID
- > DateTable
- > DateTable Compare
- > Regions
- > SalesPeople
- > Wines
- > Wines Compare
- ✓ Winesales
 - Σ 10 PC of Cases
 - Σ CASES SOLD
 - CUSTOMER ID
 - Customer Name from Customers Table
 - > ORDER DATE
 - > REGION NAME

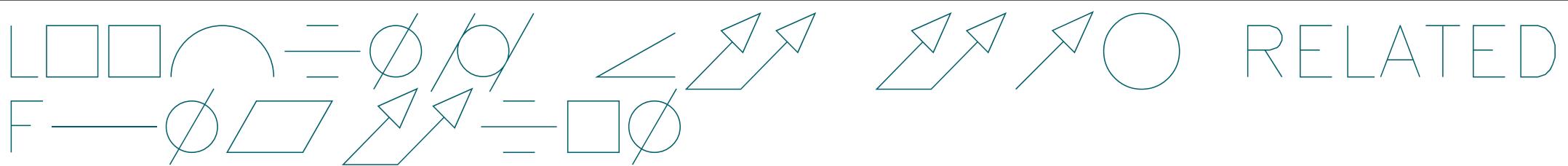


- You should understand that you can only use this function in the following two circumstances:
 1. The tables must be **related**.
 2. Only values from tables on the **one side** of a relationship can be returned to tables on the **many side**.



RELATED

- The act of populating values from tables that sit on the one side of a relationship into tables that sit on the many is called *denormalization*.
- There are at least **three advantages** in doing this:
 1. You now know in which Region each sales transaction was made.
 2. If you need to use the region names in a visual, you can use the calculated column in the Winesales table. Therefore, you no longer need to see the Regions table in Report view. If this is the case, you can hide the Regions table. To hide a table in Report view, right-click the table name in the Fields list in either **Data view** or **Model view**, and select **Hide in report view**.
 3. You can perform a distinct count on the REGION NAME column in the Winesales table to calculate how many *different* Regions we've sold our wines in. We'll do this calculation later, but because the sales transactions must be directly associated with the regions in which they were made, this would be a difficult expression if we left the REGION values in the Regions table.



- Although we have a Winesales table, we have no sales values. However, we can now calculate them. We can multiply the CASES SOLD column in the Winesales table with the PRICE PER CASE column in the Wines table, and because the Winesales table is related to the Wines table in a **many-to-one relationship**, we can use RELATED to do this.
- We're going to look at two different methods of using RELATED to calculate the **Sales** revenue values. **(Question)**

WINE	SALES
Bordeaux	\$4,055,250
Champagne	\$7,373,700
Chardonnay	\$4,203,000
Chenin Blanc	\$1,236,950
Chianti	\$1,092,920
Grenache	\$1,078,950
Malbec	\$2,914,650
Merlot	\$900,276
Piesporter	\$1,384,155
Pinot Grigio	\$703,470
Rioja	\$1,527,795
Sauvignon Blanc	\$1,896,600
Shiraz	\$1,364,766
Total	\$29,732,482

Visualizations

Build visual

Filters

Columns

Drill through

Cross-report

Keep all filters

Add drill-through fields here

Does this sound a very efficient way of doing this calculation?

Probably not?

Give two reasons?

If you shouldn't use a calculated column for the sales calculation, what should you use?

DAX MO \leq S—OOS

- We're now ready to look at the second type of DAX expression, the measure. There are two types of measures that you can use in visuals: *implicit* measures and *explicit* measures.
- What's the **difference** between implicit and explicit measures?



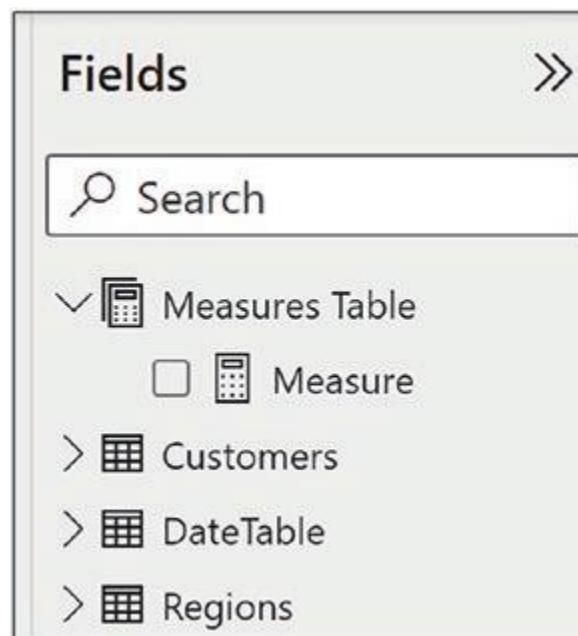
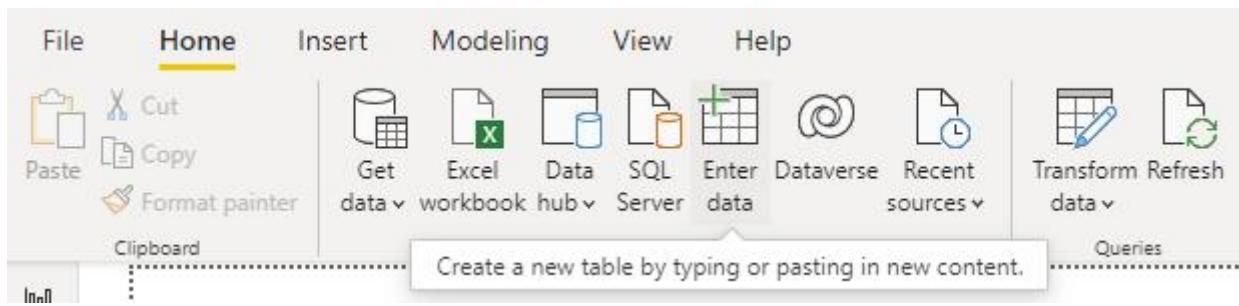
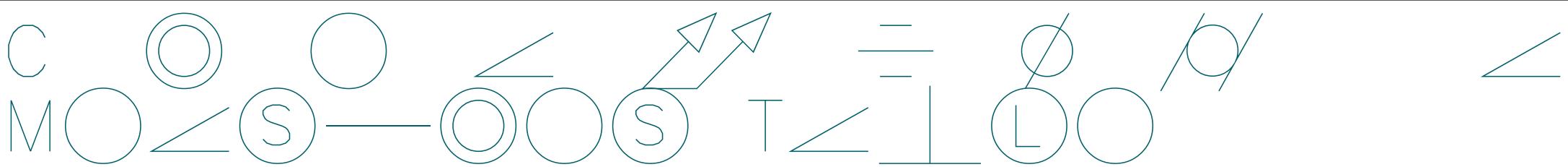
- If you've created any Power BI visual, you've created an implicit measure. Have you ever wondered what the **sigma symbol (Σ)** beside a numeric column in the Fields list means? It has a more precise purpose than signaling a column containing numbers. **The sigma indicates that when you put this column into the Values bucket of a visual, the data in this column will automatically be aggregated.** This is what we mean by an implicit measure.

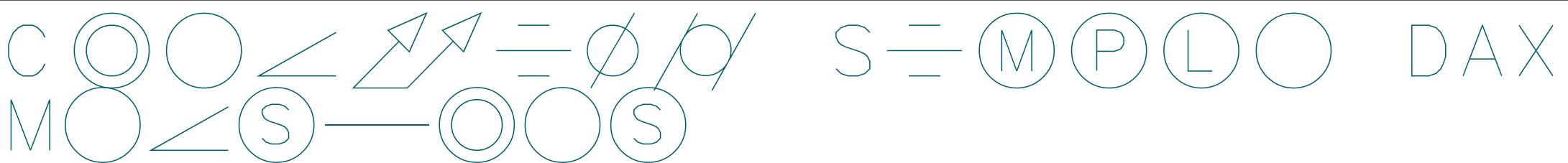
The screenshot shows the Power BI desktop interface. On the left, there's a list of wine names: Bordeaux, Champagne, Chardonnay, Chenin Blanc, Chianti, Grenache, Malbec, Merlot, Piesporter, Pinot Grigio, Rioja, Sauvignon Blanc, Shiraz, and Total. Above this list, the title "WINE" is underlined, and the subtitle "Average of CA" is displayed. To the right, the "Fields" pane is open, showing a search bar and a list of fields categorized under "Wines". The "WINE" field is selected, and its context menu is open, showing options like "Remove field", "Rename for this visual", "Move", "Add a sparkline", "Conditional formatting", "Don't summarize", "Sum", "Average" (which is checked), "Minimum", "Maximum", "Count (Distinct)", "Count", "Standard deviation", "Variance", "Median", "Show value as", and "New quick measure". Other fields listed include COST PRICE, PRICE PER CASE, SUPPLIER, TYPE, and WINE.

WINE	Average of CA
Bordeaux	
Champagne	
Chardonnay	
Chenin Blanc	
Chianti	
Grenache	
Malbec	
Merlot	
Piesporter	
Pinot Grigio	
Rioja	
Sauvignon Blanc	
Shiraz	
Total	



- If you create your own measures rather than relying on implicit measures, these are some of the **benefits**:
 - You'll have more control over the aggregation performed by the measure and be able to name it accordingly.
 - You'll be able to use different numeric formatting for different measures.
 - Explicit measures will become a constituent part of the data model. Your measures will join the Fields list, and you, or people using your data, can use and reuse the measures whenever you need to visualize a particular calculation.
 - By using DAX, you can go far beyond just simple aggregations of your data. You can perform complex calculations to get to the insights you really need.





Visualizations >> Fields >>

Build visual

Values New measure
 New column
 New quick measure
 Refresh data
 Edit query
 Manage relationships
 Incremental refresh
 Manage aggregations
 Add data fields here
 Drill through
 Cross-report On
 Keep all filters On
 Add drill-through fields here
 Collapse all
 Expand all

Search

> Customers
> DateTable
 > Measures Table
 Total Cases ...
> Regions
> SalesPeople
> Wines
> Winesales

X ✓

1 Total Cases =
2 SUM (Winesales[CASES SOLD])

WINE	Total Cases
Bordeaux	54070
Champagne	49158
Chardonnay	42030
Chenin Blanc	24739
Chianti	27323
Grenache	35965
Malbec	34290
Merlot	23084
Piesporter	10253
Pinot Grigio	23449
Rioja	33951
Sauvignon Blanc	47415
Shiraz	17497
Total	423224

Visualizations

Build visual

Filters

Fields

Search

Measures Table

Total Cases

Customers

DateTable

Regions

SalesPeople

Wines

Winesales

Columns

WINE

Total Cases

Drill through

Cross-report

Keep all filters

Add drill-through fields here

File Home Insert Modeling View Help Format

Name Total Cases

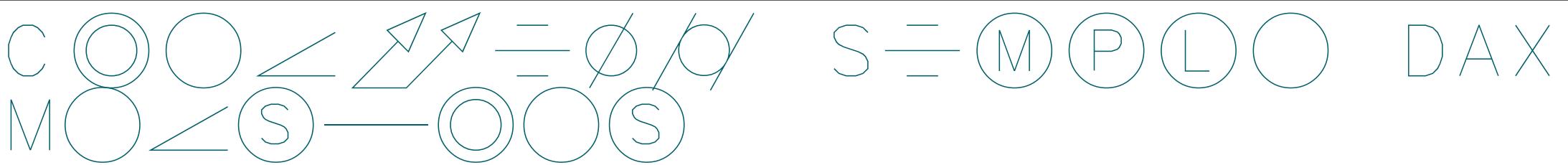
Home table Measures Table

Format Whole number

\$ % , . 0

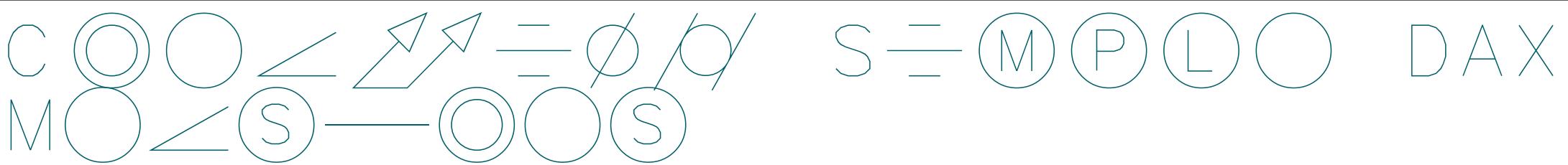
Structure

1 Total Cases =
2 SUM (Winesales[CASES SOLD])



- Avg Cases = calculate the average cases sold.
- No. of Sales = count the number of rows.
- Then re-creating the following figures.

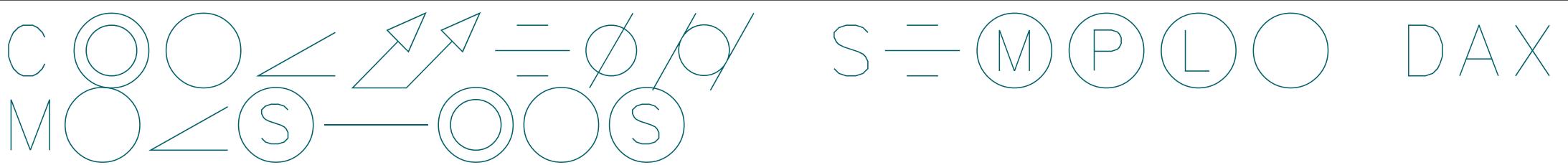
Region	Sales Data			Wine Sales Data		
	Total Cases	Avg Cases	No. of Sales	Total Cases	Avg Cases	No. of Sales
Argentina	18,377	211.23	87	Bordeaux	54,070	300.39
Australia	15,794	183.65	86	Champagne	49,158	372.41
Canada	6,317	180.49	35	Chardonnay	42,030	224.76
China	27,389	195.64	140	Chenin Blanc	24,739	123.70
Czech Republic	33,958	196.29	173	Chianti	27,323	184.61
England	23,080	198.97	116	Grenache	35,965	197.61
France	12,213	187.89	65	Malbec	34,290	201.71
Germany	19,158	193.52	99	Merlot	23,084	147.03
India	34,292	174.07	197	Piesporter	10,253	89.16
Ireland	1,160	290.00	4	Pinot Grigio	23,449	139.58
Italy	35,374	194.36	182	Rioja	33,951	172.34
Japan	22,153	203.24	109	Conventional Blends	17,115	202.22
New Zealand	23,813	177.71	134	Total	423,224	191.76
Northern Ireland	3,489	183.63	19			2207
Russia	1,043	260.75	4	Salesperson Total Sales Data		
Scotland	25,839	198.76	130	Abel	69,871	185.83
South Africa	26,002	192.61	135	Blanchet	65,581	191.20
United Arab Emirates	27,102	193.59	140	Charron	68,137	196.36
United States	14,210	205.94	69	Denis	84,018	193.14
Wales	52,461	185.37	283	Leblanc	69,304	195.22
Total	423,224	191.76	2207	Reyer	66,313	188.93
				Total	423,224	191.76
						2207



- In DAX, we have an aggregate function for this job. Its name is **DISTINCTCOUNT**, and we can simply reference the column required for the analysis. Let's discover how many *different customers* we sold our wines to by authoring this measure:

Distinct Customers = DISTINCTCOUNT (Winesales[CUSTOMER ID])

REGION NAME = RELATED (Regions[REGION])



- We can use this calculated column to create a measure to calculate in how many *different regions* we've sold our wines:

Distinct Regions = DISTINCTCOUNT (Winesales[REGION NAME])

WINE	Distinct Customers	Distinct Regions
Bordeaux	57	18
Champagne	53	19
Chardonnay	59	19
Chenin Blanc	55	19
Chianti	50	18
Grenache	51	17
Malbec	53	20
Merlot	54	18
Piesporter	43	18
Pinot Grigio	51	17
Rioja	59	17
Sauvignon Blanc	58	20
Shiraz	51	18
Total	84	20

« Filters »
Visualizations
Fields »

Build visual

Measures Table

- Avg Cases
- Distinct Customer...
- Distinct Regions
- No. of Sales
- Total Cases

Customers

DateTable

Regions

SalesPeople

Wines

Search

Winesales

Columns

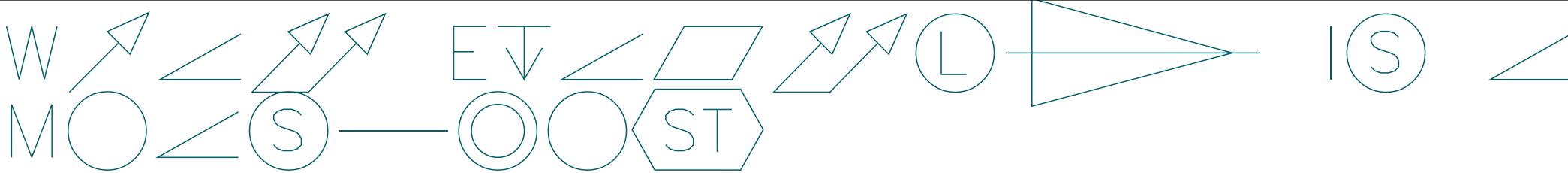
- WINE
- Distinct Customers
- Distinct Regions

Drill through

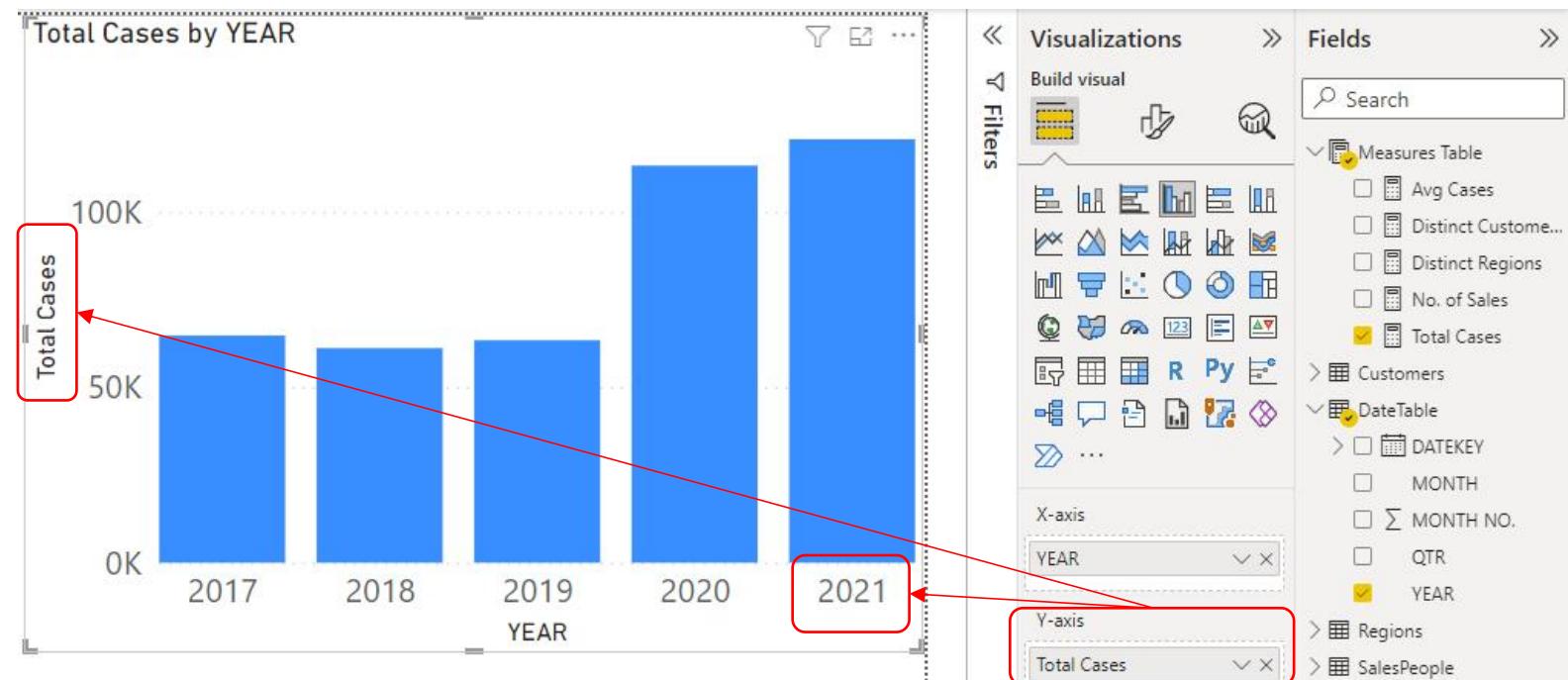
Cross-report Off

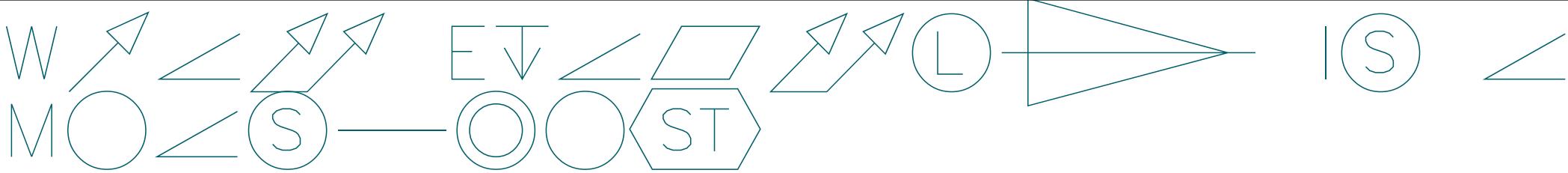
Keep all filters

Add drill-through fields here



- A measure is a DAX expression that is used in a Power BI visual to return a scalar value and is evaluated in a specific filter context.
- In other words, DAX measures *filter the rows of tables* and typically perform an *aggregation* on the filtered data to return a scalar value (which is a *single value*) that is visualized in the report.





- Let's take a closer look at these **three aspects** of the measure:
 1. All visuals on the report canvas use **measures**.
 2. Measures return **scalar** (single) values.
 3. Measures are calculated where a filter has been placed on the data model. This is known as the ***filter context***.

Row Labels	Sum of CASESSOLD
Bordeaux	54070
Champagne	49158
Chardonnay	42030
Chenin Blanc	24739
Chianti	27323
Grenache	35965
Malbec	34290
Merlot	23084
Piesporter	10253
Pinot Grigio	23449
Rioja	33951
Sauvignon Blanc	47415
Shiraz	17497
Grand Total	423224

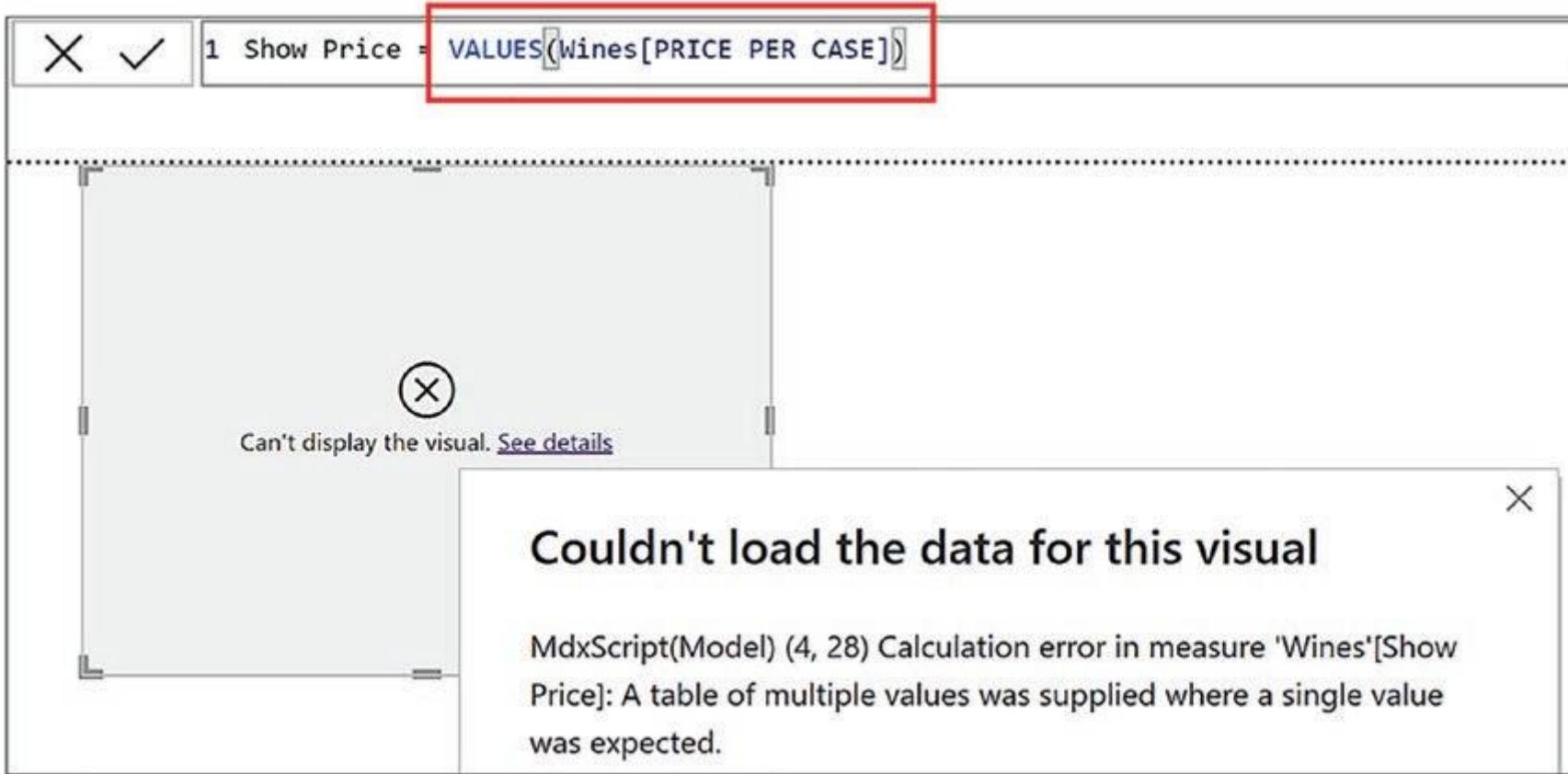


SALEDATE	WINESALESNO	SALESPERSONID	CUSTOMERID	WINEID	CASESSOLD	TOTALSALES	WINE
23/12/2021	2207	3	12	1	290	21750	Bordeaux
14/12/2021	2184	1	34	1	190	14250	Bordeaux
13/12/2021	2181	4	3	1	330	24750	Bordeaux
06/12/2021	2169	5	11	1	188	14100	Bordeaux
20/11/2021	2145	4	44	1	149	11175	Bordeaux
14/11/2021	2134	3	37	1	329	24675	Bordeaux
15/10/2021	2083	3	16	1	197	14775	Bordeaux
07/10/2021	2065	5	39	1	451	33825	Bordeaux
05/10/2021	2060	3	18	1	304	22800	Bordeaux
21/09/2021	2037	3	25	1	240	18000	Bordeaux
19/09/2021	2033	1	17	1	382	28650	Bordeaux
13/09/2021	2022	4	10	1	328	24600	Bordeaux
11/09/2021	2019	5	28	1	405	37125	Bordeaux

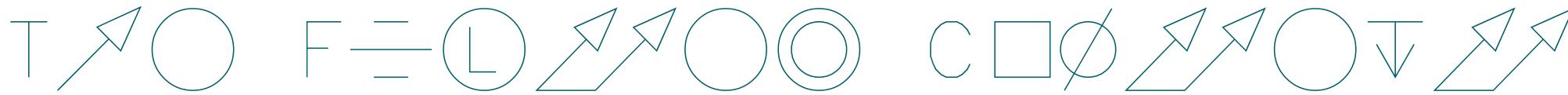
```
1 Total Cases 2 = Winesales[CASES SOLD] * Wines[PRICE PER CASE]
```

! A single value for column 'CASES SOLD' in table 'Winesales' cannot be determined. This c

"A single value for column 'CASES SOLD' in table 'Winesales' cannot be determined. This can happen when a measure formula refers to a column that contains many values without specifying an aggregation such as min, max, count, or sum to get a single result."



"A table of multiple values was supplied where a single value was expected."

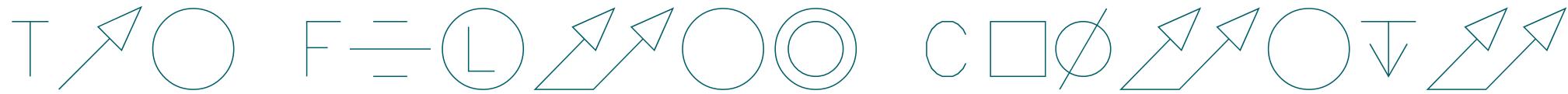


- You learned that measures are report-level calculations and that they must return a scalar value. This brings us to the third and most important aspect of the measure, and that is that all DAX measures are *evaluated in a specific filter context*. To understand what is meant by a “specific filter context,” let’s compare these two different measures:

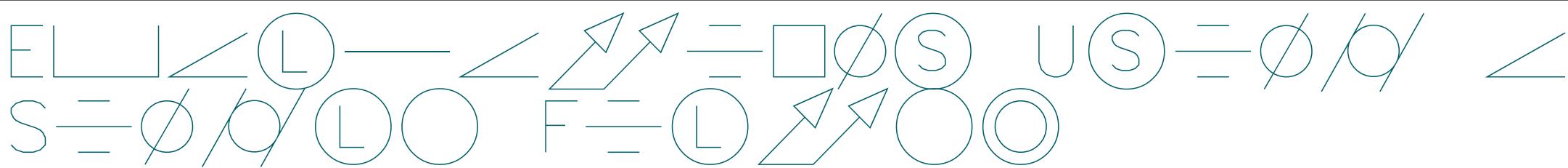
Total Cases = SUM (Winesales[CASES SOLD])

Total Stores = SUM (Customers[NO. OF STORES])

WINE	Total Cases	Total Stores
Bordeaux	54,070	1181
Champagne	49,158	1181
Chardonnay	42,030	1181
Chenin Blanc	24,739	1181
Chianti	27,323	1181
Grenache	35,965	1181
Lambrusco		1181
Malbec	34,290	1181
Merlot	23,084	1181
Piesporter	10,253	1181
Pinot Grigio	23,449	1181
Rioja	33,951	1181
Sauvignon Blanc	47,415	1181
Shiraz	17,497	1181
Total	423,224	1181

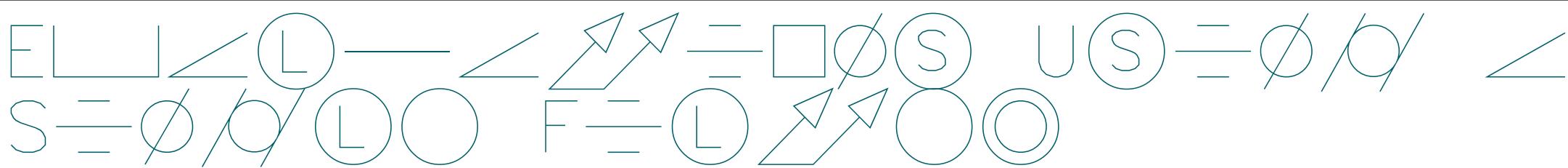


- When a measure is placed into any visual, before the measure is evaluated, the DAX engine *in memory* places filters on tables in the data model depending on three factors:
 1. The column or columns placed *in the visual* that group and categorize the data
 2. The columns *in slicers* that are filtering the data in the visual
 3. Any columns placed in *the Filters pane* that are filtering the data in the visual

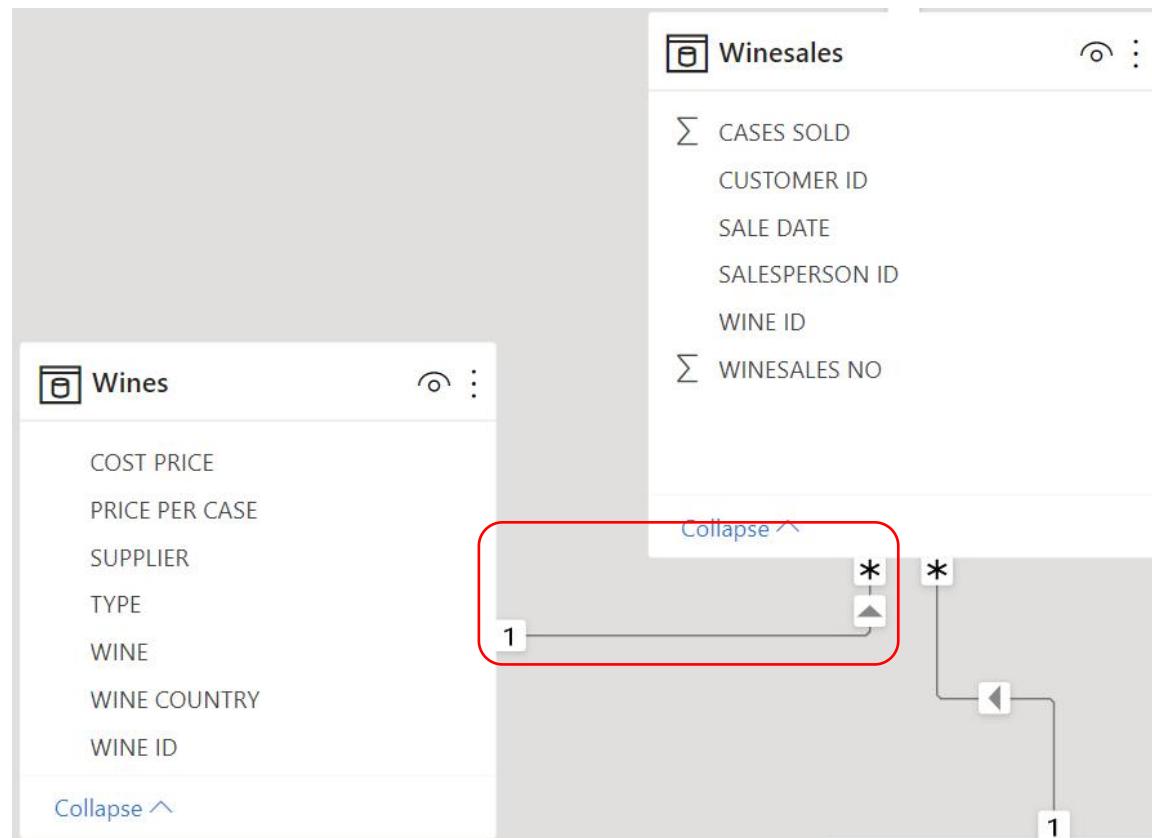


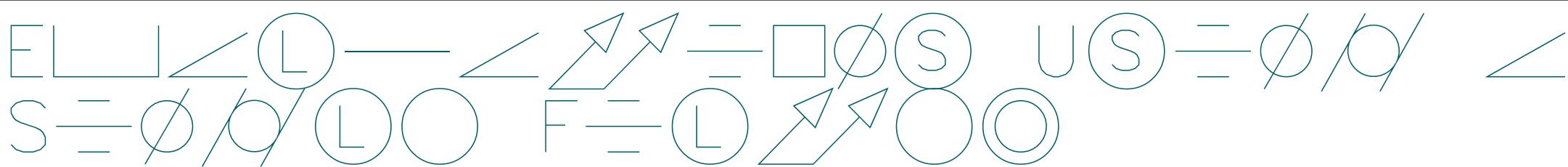
- The column in the visual that's grouping the data is the WINE column from the Wines dimension. The first value in this column to be calculated is the total cases for “Bordeaux” wine.
- Before the “Total Cases” measure calculates the value for “Bordeaux,” a filter is placed in memory on the Wines dimension to filter “Bordeaux” wines.

WINE ID	WINE	SUPPLIER	TYPE	WINE COUNTRY	PRICE PER CASE	COST PRICE
1	Bordeaux	Laithwaites	Red	France	\$75.00	\$25.00



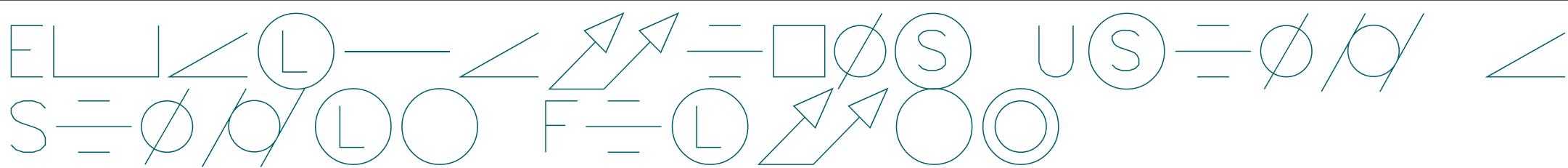
- We can see that the Wines dimension is related to the Winesales fact table in a many-to-one relationship. The arrow tells us that if the Wines dimension is filtered, this filter is *propagated* onward to the Winesales fact table.





- Therefore, the Winesales fact table is now *cross-filtered* to only contain sales for “Bordeaux” wine that has the **WINE ID that equals 1**. Notice there is no filter in the WINE ID column in the Winesales table because the filter on the Winesales table is a cross-filter that is generated only through filter propagation.
- This is the only filter affecting this visual, so the measure now sums the CASES SOLD column for “Bordeaux” wines and returns **54,070**.
- Every evaluation of the “Total Cases” measure is evaluated in a different filter context.*

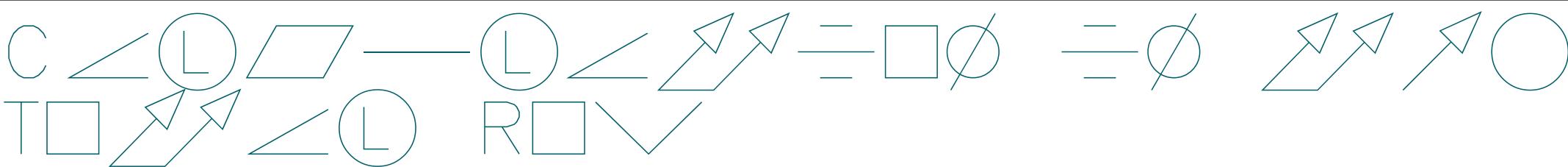
SALE DATE	WINESALES NO	SALESPERSON ID	CUSTOMER ID	WINE ID	CASES SOLD
03/07/2021	1903	6	30	1	323
24/04/2021	1769	6	55	1	208
21/03/2021	1703	6	5	1	421
29/01/2021	1618	6	34	1	236
07/01/2021	1571	6	1	1	234
26/12/2020	1557	6	3	1	322
19/11/2020	1482	6	35	1	249
08/11/2020	1455	6	35	1	485
06/10/2020	1411	6	21	1	365
05/07/2020	1245	6	21	1	491
01/06/2020	1182	6	33	1	358
01/05/2020	1124	6	46	1	280
19/04/2020	1100	6	24	1	470



- There is a way that we can prove that our Wines dimension, in memory, is filtered to one row on the evaluation of a measure that analyzes each wine. We can create this measure that counts the rows of the Wines dimension:

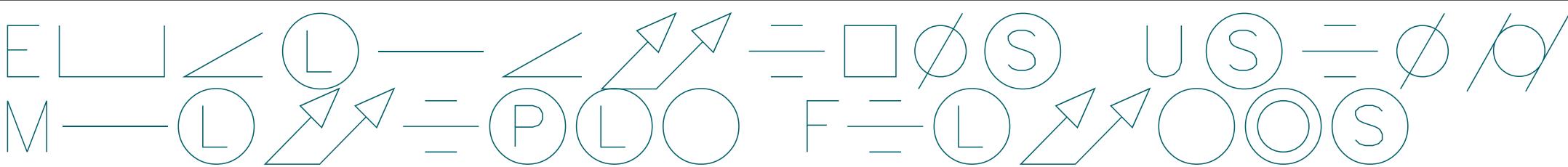
No. of Wines = COUNTROWS (Wines)

WINE	No. of Wines
Bordeaux	1
Champagne	1
Chardonnay	1
Chenin Blanc	1
Chianti	1
Grenache	1
Lambrusco	1
Malbec	1
Merlot	1
Piesporter	1
Pinot Grigio	1
Rioja	1
Sauvignon Blanc	1
Shiraz	1
Total	14



- This value is *not* the sum of the total values for each wine shown in the visual. When the measure is evaluated for the Total row, the filter is removed from the WINE column of the Wines dimension, so the expression is evaluated for *all* wines. In other words, it's our expression “= SUM (Winesales[CASES SOLD])” calculated in yet *another different filter context*.

WINE	Total Cases
Bordeaux	54,070
Champagne	49,158
Sauvignon Blanc	47,415
Chardonnay	42,030
Grenache	35,965
Malbec	34,290
Rioja	33,951
Chianti	27,323
Chenin Blanc	24,739
Pinot Grigio	23,449
Merlot	23,084
Shiraz	17,497
Piesporter	10,253
Total	423,224



- Let's create some more filters that affect the Table visual. For instance, we could include a slicer using the **SALESPERSON** column from the SalesPeople dimension and also have the **REGION** column from the Regions dimension in a page-level filter.

WINE Total Cases Total Stores

WINE	Total Cases	Total Stores
Bordeaux	265	79
Champagne		79
Chardonnay	209	79
Chenin Blanc		79
Chianti	242	79
Grenache		79
Lambrusco		79
Malbec	256	79
Merlot	449	79
Piesporter	254	79
Pinot Grigio	112	79
Rioja	386	79

SALESPERSON

SALESPERSON
Abel
Blanchet
Charron
Denis
Leblanc
Reyer

Search

Filters on this visual

REGION

is Argentina

Filter type ⓘ

Basic filtering

Search

Select all

Argentina 1

Australia 1

Canada 1

China 1

Czech Republic 1

England 1

Require single selection

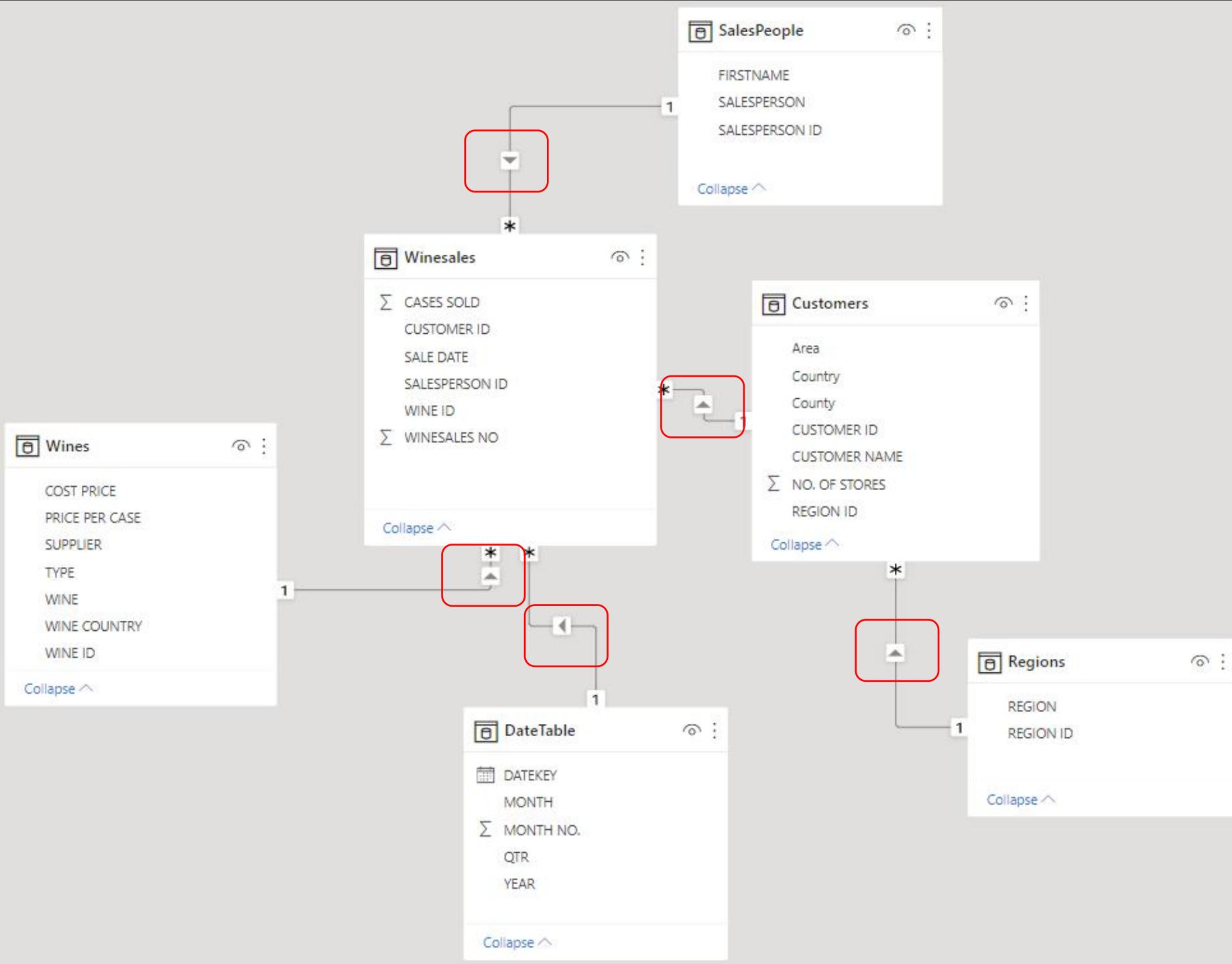
Total Cases is (All)

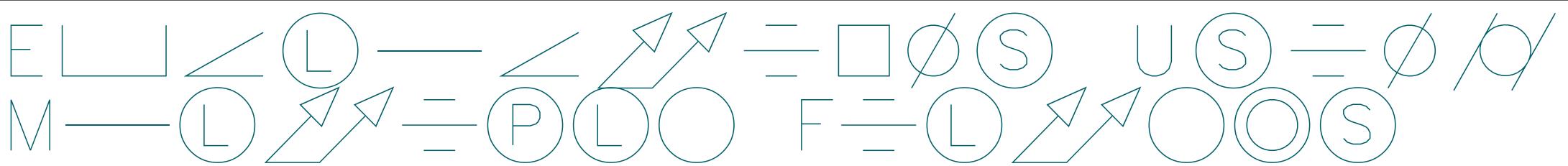
SALESPERSON ID SALESPERSON FIRSTNAME

1	Abel	Claude
---	------	--------

REGION ID REGION

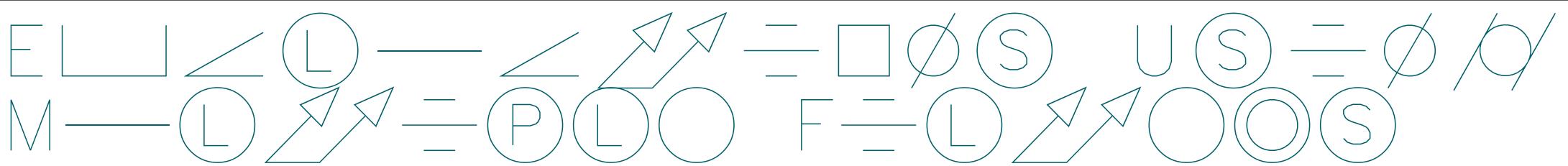
100	Argentina
-----	-----------





- Notice it returns the same value of **1,181** for every wine and also in the Total row. This measure is summing the NO. OF STORES column in the Customers dimension. The Customers dimension has no filter on it when this measure is evaluated. The only filter is on the Wines dimension. Therefore, for the evaluation of every wine, the measure sums the values in the NO. OF STORES column in the Customers table for *all* the customers.





- The filter context underpins all DAX measures and is the reason why it's so important to distinguish between the two different types of table, **dimension tables and fact tables**, because they play two different roles in the evaluation of the measure:
 - The role of dimension tables is to group the data and to propagate filters through the data model into the fact table.
 - The role of fact tables is to summarize subsets of data that have been cross-filtered from dimensions.
- *DAX measures typically summarize data in the fact table that's been cross-filtered by dimension tables.*



- The filter context is not the only evaluation context that DAX uses. There is another evaluation context called the *row context*. Row context is applicable in any DAX expression that *iterates the rows of a table* where the expression is bound to the values in the current row. All calculated columns are evaluated in the row context and this is how they differ from measures, which are always evaluated in the filter context. However, just to make life difficult, some measures will use **both the filter context and the row context** in their evaluation. Also, there are some calculated columns whose row context can be turned into a filter context.
- In an Excel table, the formula is “copied down” where it is calculated for every row in the column. An “@” character is used in the formula to denote using the values in the current row. This is essentially what the row context is in DAX. When using the row context, the DAX expression iterates over every row in the table, and the values used in the expression are *the values sitting in the current row*.

EXCEL

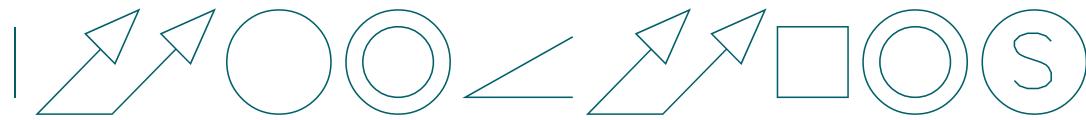
G2 : =[@CASESSOLD]*VLOOKUP([@WINEID],wines,6,0)

	A	B	C	D	E	F	G
1	SALEDATE	WINESALESNO	SALESPERSONID	CUSTOMERID	WINEID	CASESSOLD	SALES
2	01/01/2016	1	3	52	6	77	£10,395
3	01/01/2016	2	5	41	7	82	£3,280
4	02/01/2016	3	2	52	10	89	£3,560
5	02/01/2016	4	2	79	13	107	£8,346
6	02/01/2016	5	6	49	4	131	£11,135
7	02/01/2016	6	6	49	4	131	£11,135
8	03/01/2016	7	4	71	3	171	£17,100

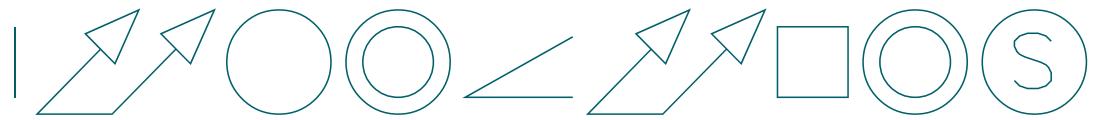
DAX

X ✓ 1 Sales = Winesales[CASES SOLD] * RELATED (Wines[PRICE PER CASE])

SALE DATE	WINESALES NO	SALESPERSON ID	CUSTOMER ID	WINE ID	CASES SOLD	Sales
01 January 2016	1	3	52	6	75	£10,125
01 January 2016	2	5	41	7	102	£7,280
02 January 2016	3	2	52	10	89	£3,560
02 January 2016	4	2	79	13	107	£8,346
02 January 2016	5	6	49	4	131	£11,135
02 January 2016	6	6	49	4	131	£11,135
03 January 2016	7	4	71	3	171	£17,100



- There is a group of functions in DAX that are referred to as *iterators*, and from their name, we can infer that these functions iterate tables in the evaluation of a DAX expression. Any DAX function that ends in an “X” is an iterator, such as the “X” aggregators: SUMX, AVERAGEX, MAXX, MINX, COUNTAX.
- There are also “X” iterating functions that *aren't aggregators* such as CONCATENATEX and RANKX.
- Just to make life even more confusing, there are iterating functions that *don't end in “X”* such as FILTER and ADDCOLUMNS.



- Aggregating iterators have two arguments: **the table to be iterated and the expression** that is to be evaluated for each row of the table, the result of which will then be aggregated. These functions create a row context inside the measure by iterating the table referenced by the function, and each row in the table is “visited” in memory by the measure. Remember that the measure will have generated **a filter context first**, so the table being iterated may have a **filter or cross-filter on it**.

10 PC Increase Total =

SUMX (Winesales, Winesales[CASES SOLD] * 1.1)

AVERAGEX, MAXX, MINX

DAX Measure

1 10 PC Increase Total =
2 SUMX (Winesales,Winesales[CASES SOLD]* 1.1)

SALES NO	SALESPERSON ID	CUSTOMER ID	WINE ID	CASES SOLD	10 Percent Increase
1903	6	30	1	323	355
1769	6	55	1	208	229
1703	6	5	1	421	463
1618				236	260
1571			Bordeaux	54,070	59,477.00
1557			Champagne	49,158	54,073.80
1482			Chardonnay	42,030	46,233.00
1455			Chenin Blanc	24,739	27,212.90
1411			Chianti	27,323	30,055.30
1245			Grenache	35,965	39,561.50
1182			Malbec	34,290	37,719.00
1124			Merlot	23,084	25,392.40
			Piesporter	10,253	11,278.30
			Pinot Grigio	23,449	25,793.90
			Rioja	33,951	37,346.10
			Sauvignon Blanc	47,415	52,156.50
			Shiraz	17,497	19,246.70
			Total	423,224	465,546.40



= SUMX (table, expression)

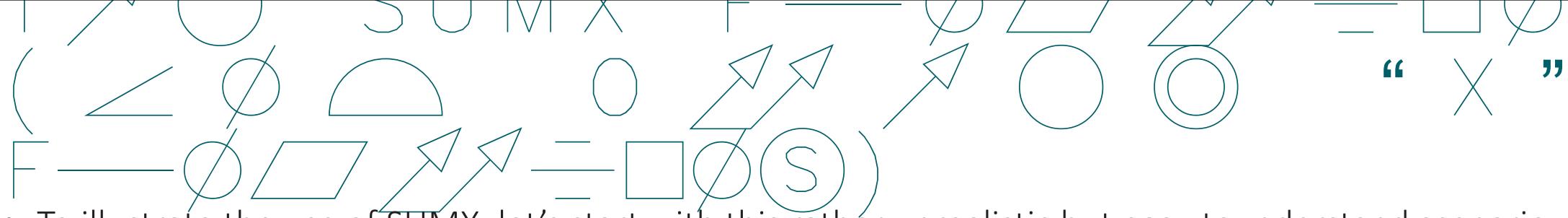
where:

table is the table where you want to perform the calculation.

expression is the calculation you want to be performed for each row in that table.

Here's an example of the SUMX syntax:

= SUMX (Winesales, Winesales[CASES SOLD] * 0.1)



- To illustrate the use of SUMX, let's start with this rather unrealistic but easy-to-understand scenario. We have been asked to find any CASES SOLD value that is greater than 100 and increase this value by 20%; otherwise, we only increase the value by 10%. Perhaps this is some strange way of predicting next year's volume of cases sold, so we'll call this calculation "Next Yr Cases". We then want to see what the "Next Yr Cases" value would be for each of our wines.

Next Yr Cases Measure =

```
SUMX (  
    Winesales,  
    IF ( Winesales[CASES SOLD] > 100,  
        Winesales[CASES SOLD] * 1.2,  
        Winesales[CASES SOLD] * 1.1  
    )  
)
```



```
1 Next Yr Cases =  
2 IF ([Winesales[CASES SOLD] > 100,  
3     Winesales[CASES SOLD] * 1.2,  
4     Winesales[CASES SOLD] * 1.1])
```

DAX Calculated Column

SALE DATE	WINESALES NO	SALESPERSON ID	CUSTOMER ID	WINE ID	CASES SOLD	Next Yr Cases
01/01/2017	2	6	35	10	213	255.6
01/01/2017	1	3	16	4	326	391.2
02/01/2017	3	4	20	5	70	77
03/01/2017	4	1	12	10	264	316.8
07/01/2017	5	2	17	3	147	176.4
08/01/2017	6	3	45	11	155	186
09/01/2017	7	6	11	7	172	207.6

WINE Total Cases Next Yr Cases

Bordeaux	54,070	64,884.00
Champagne	49,158	58,989.60
Chardonnay	42,030	50,436.00
Chenin Blanc	24,739	29,656.80
Chianti	27,323	32,600.20
Grenache	35,965	42,950.10
Malbec	34,290	41,138.00
Merlot	23,084	27,575.90
Piesporter	10,253	11,407.50
Pinot Grigio	23,449	27,906.20
Rioja	33,951	40,741.20
Sauvignon Blanc	47,415	56,898.00
Shiraz	17,497	20,218.70
Total	423,224	505,402.20

Drill through

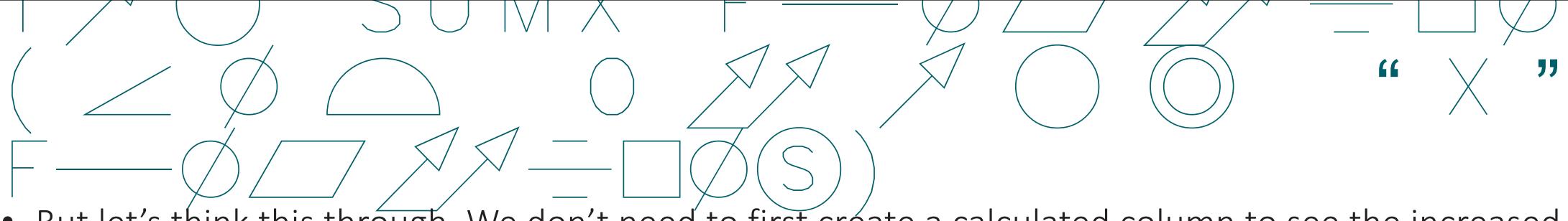
Cross-report

Off

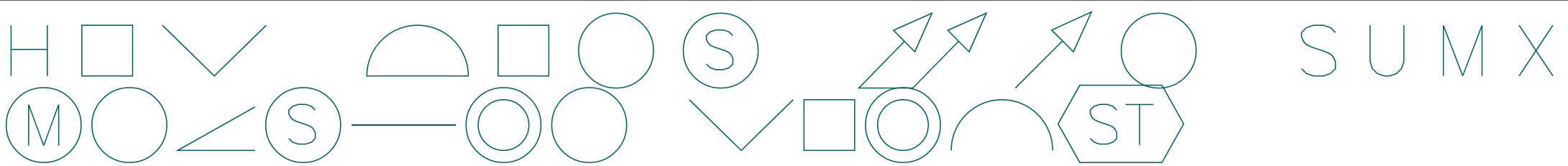
Keep all filters

On

Add drill-through fields here



- But let's think this through. We don't need to first create a calculated column to see the increased value for each row and then in *another step* sum this value for each wine, using an implicit measure. We can do it all in one go using SUMX. If we do this, the requirement for the calculated column is redundant; we can just use the measure. This is the real benefit because measures are always more efficient than calculated columns.



- We know that SUMX sums the expression evaluated for *each row in the table*. The first argument in SUMX references the table where the calculation will be performed, in our case, Winesales. The second argument is the calculation you want to be done in memory *for each row* in this table. This is our expression using IF that SUMX calculates in memory by iterating every row. It then sums the results of this calculation, in this case, for each wine (because that's the current filter context for the evaluation of the measure).



Sales =

Winesales[CASES SOLD] * RELATED (Wines[PRICE PER CASE])

Total Sales =

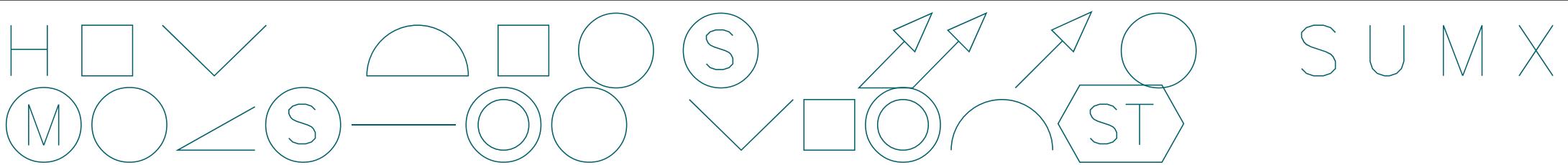
SUMX (

 Winesales,

 Winesales[CASES SOLD] * RELATED (Wines[PRICE PER CASE])

)

WINE	Total Sales	Sales
Bordeaux	\$4,055,250	\$4,055,250
Champagne	\$7,373,700	\$7,373,700
Chardonnay	\$4,203,000	\$4,203,000
Chenin Blanc	\$1,236,950	\$1,236,950
Chianti	\$1,092,920	\$1,092,920
Grenache	\$1,078,950	\$1,078,950
Malbec	\$2,914,650	\$2,914,650
Merlot	\$900,276	\$900,276
Piesporter	\$1,384,155	\$1,384,155
Pinot Grigio	\$703,470	\$703,470
Rioja	\$1,527,795	\$1,527,795
Sauvignon Blanc	\$1,896,600	\$1,896,600
Shiraz	\$1,364,766	\$1,364,766
Total	\$29,732,482	\$29,732,482



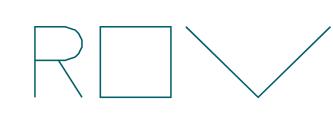
- This is a much cleaner way to calculate our Total Sales. The SUMX function iterates the Winesales table, and for every row in the current filter context, it multiplies the value in the CASES SOLD column with value in the PRICE PER CASE column of the Wines table (using RELATED to find the price of the wine in the current row context). It then sums the results of these row-level calculations for each wine.
- Question:** write DAX measures to find the maximum sales and the average sales.

WINE	Total Sales	Avg Sales	Max Sales
Bordeaux	\$4,055,250	\$22,529.1667	\$37,500
Champagne	\$7,373,700	\$55,861.3636	\$75,000
Chardonnay	\$4,203,000	\$22,475.9358	\$25,000
Chenin Blanc	\$1,236,950	\$6,184.75	\$7,500
Chianti	\$1,092,920	\$7,384.5946	\$11,960
Grenache	\$1,078,950	\$5,928.2967	\$10,500
Malbec	\$2,914,650	\$17,145	\$27,710
Merlot	\$900,276	\$5,734.242	\$7,800
Piesporter	\$1,384,155	\$12,036.1304	\$21,870
Pinot Grigio	\$703,470	\$4,187.3214	\$6,000
Rioja	\$1,527,795	\$7,755.3046	\$9,000
Sauvignon Blanc	\$1,896,600	\$11,289.2857	\$14,000
Shiraz	\$1,364,766	\$6,722.9852	\$11,700
Total	\$29,732,482	\$13,471.8994	\$75,000



- Using AVERAGEX, calculating the average price that our customers have paid for their wines.

CUSTOMER NAME	Average Price
Back River & Co	\$66.20
Ballard & Sons	\$95.00
Barstow Ltd	\$46.71
Beaverton & Co	\$82.14
Black Ltd	\$61.83
Bluffton Bros	\$59.73
Branch Ltd	\$63.40
Brooklyn & Co	\$45.64
Brooklyn Ltd	\$73.00
Brown & Co	\$45.75
Burlington Ltd	\$62.80
Burningsuit Ltd	\$80.00
Busan & Co	\$55.00
Canoga Park Ltd	\$51.00
Cape Canaveral Ltd	\$61.74
Castle Rock Ltd	\$63.34
Chandler & Sons	\$64.85
Charleston Ltd	\$70.04
Charlottesville & Co	\$66.00
Total	\$66.12

T  R  G 

Total Sales Wrong =

`SUM (Winesales[CASES SOLD]) * SUM (Wines[PRICE PER CASE])`

Total Cases =

`SUM (Winesales[CASES SOLD])`

Sum Price =

`SUM (Wines[PRICE PER CASE])`

T □ ↗ L R □ ↘ G O = O //

WINE	Total Sales	Total Sales Wrong	Sum Price	Total Cases
Bordeaux	\$4,055,250	\$4,055,250	\$75	54070
Champagne	\$7,373,700	\$7,373,700	\$150	49158
Chardonnay	\$4,203,000	\$4,203,000	\$100	42030
Chenin Blanc	\$1,236,950	\$1,236,950	\$50	24739
Chianti	\$1,092,920	\$1,092,920	\$40	27323
Grenache	\$1,078,950	\$1,078,950	\$30	35965
Lambrusco			\$20	
Malbec	\$2,914,650	\$2,914,650	\$85	34290
Merlot	\$900,276	\$900,276	\$39	23084
Piesporter	\$1,384,155	\$1,384,155	\$135	10253
Pinot Grigio	\$703,470	\$703,470	\$30	23449
Rioja	\$1,527,795	\$1,527,795	\$45	33951
Sauvignon Blanc	\$1,896,600	\$1,896,600	\$40	47415
Shiraz	\$1,364,766	\$1,364,766	\$78	17497
Total	\$29,732,482	\$388,096,408	\$917	423224

DateKey	Year	Qtr	MonthNo	Month
01 January 2017	2017	Qtr 1		1 Jan
02 January 2017	2017	Qtr 1		1 Jan
03 January 2017	2017	Qtr 1		1 Jan
04 January 2017	2017	Qtr 1		1 Jan
05 January 2017	2017	Qtr 1		1 Jan
06 January 2017	2017	Qtr 1		1 Jan
07 January 2017	2017	Qtr 1		1 Jan
08 January 2017	2017	Qtr 1		1 Jan
09 January 2017	2017	Qtr 1		1 Jan
10 January 2017	2017	Qtr 1		1 Jan
11 January 2017	2017	Qtr 1		1 Jan
12 January 2017	2017	Qtr 1		1 Jan
13 January 2017	2017	Qtr 1		1 Jan

* Winesales

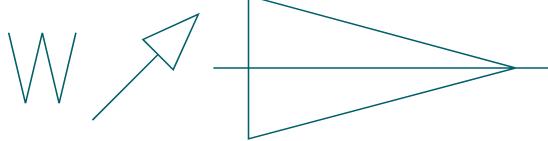
- Σ CASES SOLD
- CUSTOMER ID
- REGION NAME
- SALE DATE
- SALES
- SALESPERSON ID
- WINE ID
- Σ WINESALES NO

[Collapse ^](#)

* DateTable

- DATEKEY
- MONTH
- Σ MONTH NO.
- QTR
- YEAR

[Collapse ^](#)

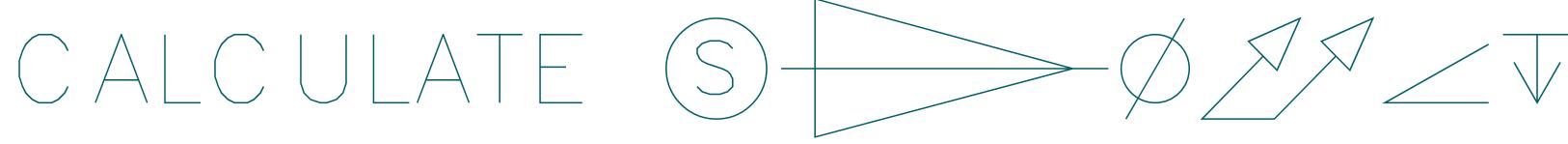
Y N

CALCULATE

WINE	Total Cases	2021 Cases	2021 PC
Bordeaux	54,070	14,940	27.63%
Champagne	49,158	11,461	23.31%
Chardonnay	42,030	11,302	26.89%
Chenin Blanc	24,739	6,952	28.10%
Chianti	27,323	8,535	31.24%
Grenache	35,965	9,702	26.98%
Malbec	34,290	10,543	30.75%
Merlot	23,084	8,158	35.34%
Piesporter	10,253	4,080	39.79%
Pinot Grigio	23,449	6,040	25.76%
Rioja	33,951	10,161	29.93%
Sauvignon Blanc	47,415	14,689	30.98%
Shiraz	17,497	4,137	23.64%
Total	423,224	120,700	28.52%

WINE	Total Cases	2021 Cases	YEAR
Bordeaux	14,940	14,940	2017
Champagne	11,461	11,461	2018
Chardonnay	11,302	11,302	2019
Chenin Blanc	6,952	6,952	2020
Chianti	8,535	8,535	2021
Grenache	9,702	9,702	
Malbec	10,543	10,543	
Merlot	8,158	8,158	
Piesporter	4,080	4,080	
Pinot Grigio	6,040	6,040	
Rioja	10,161	10,161	
Sauvignon Blanc	14,689	14,689	
Shiraz	4,137	4,137	
Total	120,700	120,700	

CALCULATE



- CALCULATE evaluates an expression in a *modified* filter context and has the following syntax:

= **CALCULATE (expression , filter1 , filter2 etc.)**

where:

expression is what you want calculated. This can be a DAX **expression** or a **measure** that defines an expression.

filter1, filter2, etc. is how you want to filter the **expression** or **measure**. You can have multiple filters, and these are combined in an “AND” logical statement.

CALCULATE

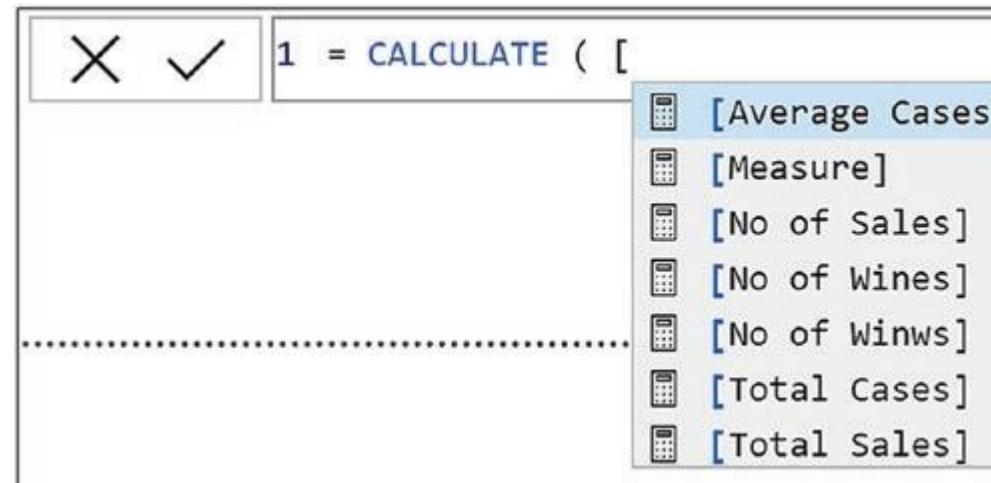


- Here are two examples of the CALCULATE syntax:

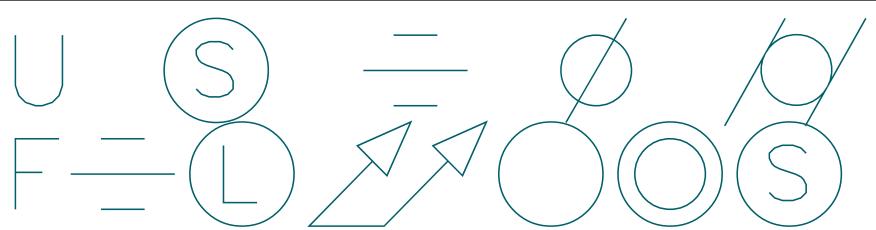
```
= CALCULATE ( SUM ( Winesales[CASES SOLD] ), Wines[WINE] = "Bordeaux" )
```

```
= CALCULATE ( [Total Cases], Wines[WINE] = "Bordeaux" )
```

The first example uses an *expression* in the expression argument, and the second uses a *measure* in the expression argument (highlighted in gray).

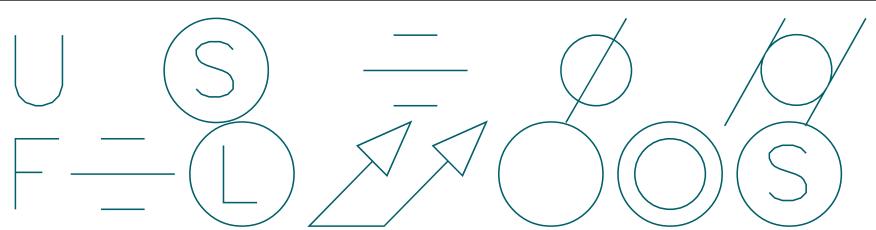


Typing a square bracket "[" into the DAX editor, lists all your measures



- How do we generate a measure to calculate the Total Cases for 2021 while retaining the measure that calculates Total Cases for all years? This is the measure, using CALCULATE that will do the job:

2021 Cases = CALCULATE ([Total Cases], DateTable[Year] = 2021)



- Now we can create the final measure that will calculate the percentage that each wine's total cases for 2021 are of the total for all years and format it as percent:

2021 Percentage = [2021 Cases] / [Total Cases]

OR

2021 Percentage = DIVIDE ([2021 Cases], [Total Cases])

WINE	Total Cases	2021 Cases	2021 Percentage
Bordeaux	54,070	14,940	27.63%
Champagne	49,158	11,461	23.31%
Chardonnay	42,030	11,302	26.89%
Chenin Blanc	24,739	6,952	28.10%
Chianti	27,323	8,535	31.24%
Grenache	35,965	9,702	26.98%
Malbec	34,290	10,543	30.75%
Merlot	23,084	8,158	35.34%
Piesporter	10,253	4,080	39.79%
Pinot Grigio	23,449	6,040	25.76%
Rioja	33,951	10,161	29.93%
Sauvignon Blanc	47,415	14,689	30.98%
Shiraz	17,497	4,137	23.64%
Total	423,224	120,700	28.52%

Visualizations

Build visual

Filters

Fields

Search

Measures Table

- 2021 Cases
- 2021 Percentage
- No. of Sales
- Total Cases
- Total Sales

Customers

DateTable

Region Groups

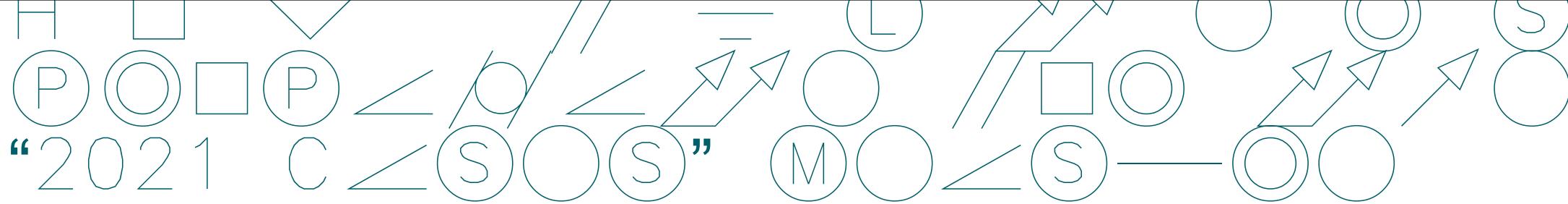
Regions

SalesPeople

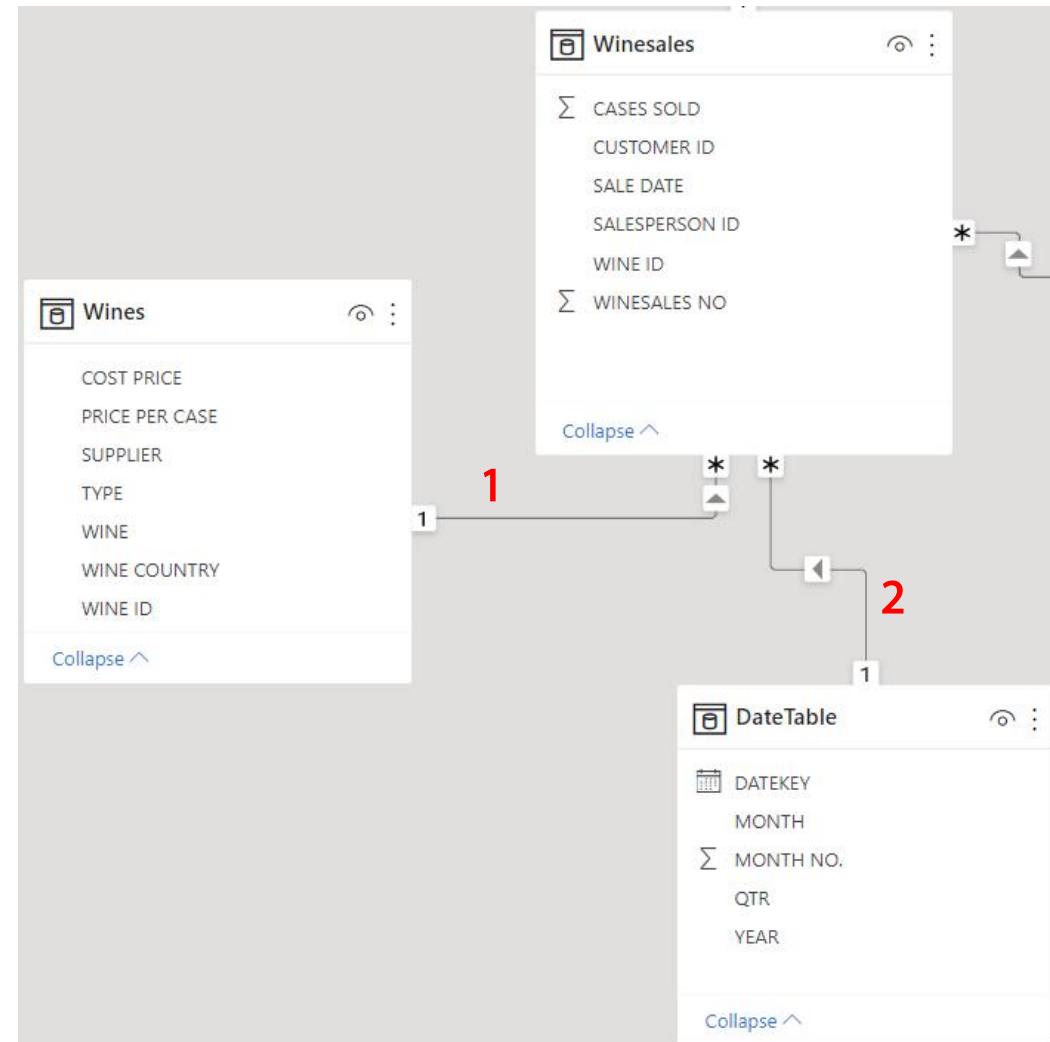
Wines

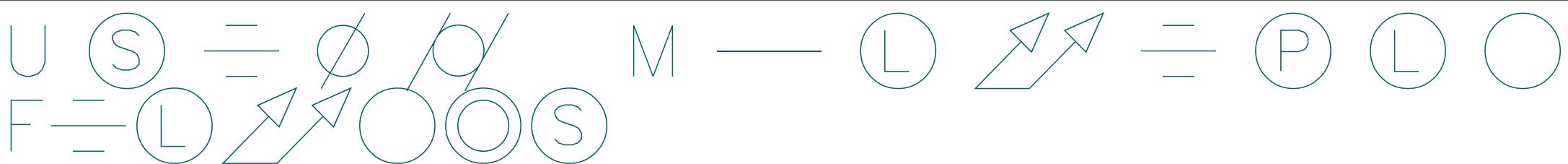
- COST PRICE
- PRICE PER CASE
- SUPPLIER
- TYPE
- WINE
- WINE COUNTRY
- WINE ID

Add drill-through fields here



1. The current filter context filters each WINE in the Wines dimension. This filter is propagated to the Winesales table and cross-filters each wine.
2. The filter provided by CALCULATE programmatically filters the DateTable for the year "2021". This filter is also propagated to the Winesales table and so applies a second cross-filter on Winesales.
3. **Question:** calculate the total sales where the CASES SOLD value is greater than 350.





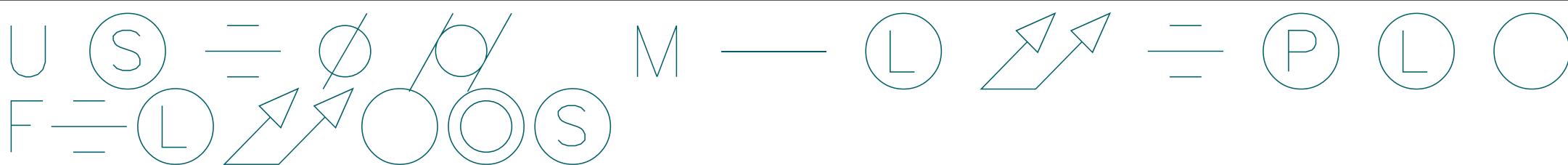
- The following are three more examples of measures that use CALCULATE to modify the filter context. Notice how all the filter arguments to CALCULATE are combined in an “AND”.

```
Total Cases in May 2021 =
CALCULATE (
    [Total Cases],
    DateTable[Year] = 2021,
    DateTable[Month] = "may"
)
```

```
Total Cases for Abel in Argentina =
CALCULATE (
    [Total Cases],
    SalesPeople[SALESPERSON] = "abel",
    Regions[REGION] = "argentina"
)
```

```
Average Cases for Black Ltd in 2021 =
CALCULATE (
    AVERAGE ( Winesales[CASES SOLD] ),
    DateTable[Year] = 2021,
    Customers[CUSTOMER NAME] = "black ltd"
)
```

WINE	Total Cases	Total Cases in May 2021	Total Cases for Abel in Argentina	Average Cases for Black Ltd in 2021
Bordeaux	54,070	1031	265	
Champagne	49,158	313		
Chardonnay	42,030	1390	209	
Chenin Blanc	24,739	128		
Chianti	27,323	1032	242	
Grenache	35,965	645		
Malbec	34,290	1296	256	
Merlot	23,084	1027	449	
Piesporter	10,253	447	254	87.00
Pinot Grigio	23,449	413	112	
Rioja	33,951	657	386	164.00
Sauvignon Blanc	47,415	603	261	
Shiraz	17,497	252	131	
Total	423,224	9234	2565	125.50



- What if you require a filter that uses “OR”, for example, 2021 *OR* 2020. Using CALCULATE, filtering using “OR” on the same column is straightforward. Filtering using “OR” on *different* columns is a little more challenging, and this is where our calculations will get a little trickier. Let’s take the simpler calculations first.
- To use “OR” on the same column, you can use the double pipe (||) operator within the same filter argument, as in these examples:

Total Cases 2020 or 2021 =
 CALCULATE ([Total Cases],
 DateTable[Year] = 2021
 || DateTable[Year] = 2020
)

Average Cases Argentina or Australia =
 CALCULATE (AVERAGE (Winesales[CASES SOLD]),
 Regions[REGION] = "argentina"
 || Regions[REGION] = "australia")

Average Cases Argentina or Australia =
 CALCULATE (AVERAGE (Winesales[CASES SOLD]),
 OR (Regions[REGION] = "argentina",
 Regions[REGION] = "australia")

C□M□P□L□O□T F=□L□↗□O□O□S

- We may want to find Total Sales for red wines OR French wines using the TYPE and WINECOUNTRY columns in the Wines table, respectively, and use this to analyze our salespeople's performance of these wines. This would be the expression:

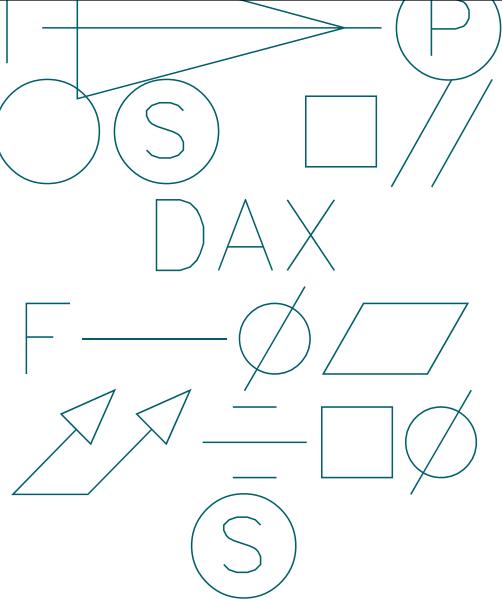
Sales for Red or French #1=

```
CALCULATE (
    [Total Sales],
    Wines[TYPE] = "red"
    || Wines[WINE COUNTRY]
    = "France"
)
```

SALESPERSON	Total Sales	Sales for Red or French #1
Abel	\$5,265,266	\$4,647,576
Blanchet	\$4,860,044	\$4,193,329
Charron	\$5,147,366	\$4,583,416
Denis	\$5,431,390	\$4,518,335
Leblanc	\$4,792,407	\$4,220,947
Reyer	\$4,236,009	\$3,584,654
Total	\$29,732,482	\$25,748,257

DAX TLOO F— ϕ □ ↗ =□∅S

- There is yet another aspect of DAX that is hidden from us, and that therefore must be imagined. That is the generation of **virtual tables**. Much of your DAX code will involve building **in-memory tables** that are used in the evaluation of the measure. We are going to explore this concept, how we create table expressions through the use of table functions, and their purpose in manipulating the data model. In doing so, we will be focusing on the most ubiquitous of the table functions, and that is the **FILTER** function.



<https://learn.microsoft.com/en-us/dax/dax-function-reference>

Function Type	Example	Description
Scalar Functions	Return scalar values. e.g., SUM, COUNTROWS, SUMX, CALCULATE	These functions return a scalar or single value and are used in all measures.
Table Functions	Return virtual tables. e.g., FILTER, VALUES, PREVIOUSMONTH, ALL	Table functions are used to generate “virtual” tables that propagate filters through the data model in the same way as “real” tables. The virtual tables are typically subsets of rows or subsets of columns of the original table, but they can expand the number of rows in the case of the ALL function. Because measures must always return scalar values and not tables, table functions are always nested inside scalar functions.
CALCULATE Modifiers	Modify the filter arguments of CALCULATE. e.g., CROSSFILTER, USERELATIONSHIP, KEEPFILTERS, ALL	We'll meet this type of function later. These functions change the behavior of any filters generated by CALCULATE and so are always nested inside CALCULATE. These functions don't return any value.

FILTER

Article • 06/22/2022 • 2 minutes to read • 2 contributors



Returns a table that represents a subset of another table or expression.

Syntax

DAX

 Copy

```
FILTER(<table>,<filter>)
```

Parameters

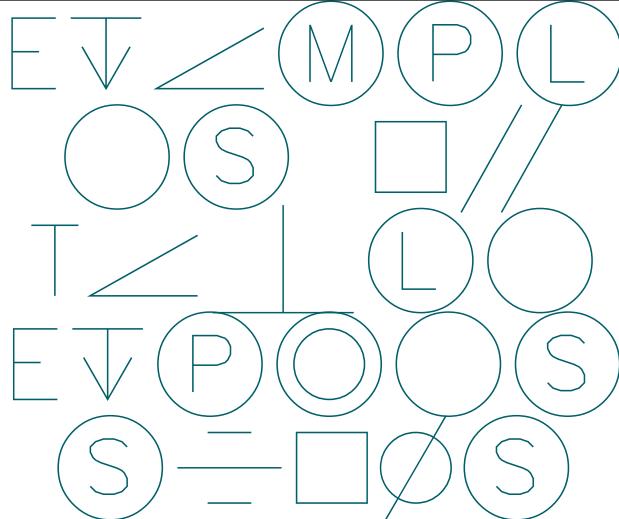
Term	Definition
table	The table to be filtered. The table can also be an expression that results in a table.
filter	A Boolean expression that is to be evaluated for each row of the table. For example, [Amount] > 0 or [Region] = "France"

Return value

A table containing only the filtered rows.

T L O F — = S

- Table functions create *table expressions* and can be used for two purposes:
 1. To generate additional tables in your data model using the **New Table** button. These are referred to as *calculated tables*. If this is your requirement, the recommendation is that new tables are generated using Power Query, not DAX.
 2. To generate in-memory virtual tables as part of the evaluation of measures.



- You need to understand that it's a *table expression* being used as the "table" argument or as the "filter" argument inside CALCULATE.

Home table Measures Table \$.. % .00 Auto

COUNTROWS([Table])

Structure X ✓

1 Counts the number of rows in a table.

2 COUNTROWS(VALUES(Wines[WINE]))

Home table Measures Table \$.. % .00 Auto

SUMX(Table, Expression)

Structure X ✓

1 Returns the sum of an expression evaluated for each row in a table.

2 SUMX(VALUES(Wines), Wines[PRICE PER CASE])

Home table Measures Table \$.. % .00 Auto

CALCULATE(Expression, [Filter1], ...)

Structure X ✓

1 Evaluates an expression in a context modified by filters.

2 CALCULATE([Total Cases], VALUES(Wines))



- Nested inside any other function other than CALCULATE, table expressions supply the “table” argument and often create subsets of the original table, either subsets of rows or subsets of columns. For example, FILTER nested inside SUMX will normally generate a table with fewer rows for SUMX to iterate. Some table functions are also used to generate “hybrid” tables that comprise combinations of columns from different tables.
- On the other hand, as filter arguments inside CALCULATE, table expressions generate virtual tables that are used as *filters*.

T O FILTER F — =

- The FILTER function returns a *table* that is a subset of another table and has the following syntax:

= FILTER (*table* , *filter*)

where:

table is the table that you want to filter. The table can also be supplied by another table function.

filter is the filter you want to apply to the **table** as a Boolean expression, for example, "Wines[TYPE]= "red".

Ex: = COUNTROWS (FILTER (Wines, Wines[TYPE]= "red"))

1. FILTER ROWS

- We could calculate the number of high-volume sales where high volume is any transaction where the CASES SOLD value is greater than 300. To do this, we can use FILTER nested inside COUNTROWS to count the rows of the filtered Winesales table as in the following expression:

No. of High Volume Sales =

```
COUNTROWS ( FILTER ( Winesales, Winesales[CASES SOLD] > 300 ) )
```

WINE	No. of High Volume Sales
Bordeaux	89
Champagne	100
Grenache	32
Malbec	1
Sauvignon Blanc	64
Total	286

1. FILTER US

- FILTER can also be **nested inside SUMX**, whereby the number of rows in the table iterated by SUMX will be reduced by FILTER. For example, the “Total Sales” measure could be extended to filter the sales where the volume of cases is greater than 300.

Cases GT 300 =

SUMX (

FILTER (Winesales, Winesales[CASES SOLD] > 300),

Winesales[CASES SOLD] * RELATED (Wines [PRICE PER CASE])

WINE	Cases GT 300
Bordeaux	\$2,658,150
Champagne	\$6,075,150
Grenache	\$310,530
Malbec	\$27,710
Sauvignon Blanc	\$835,280
Total	\$9,906,820



- If FILTER is used in a filter argument of CALCULATE, FILTER generates an in-memory table that is used to filter the data model, just as dimensions filter the data model.
- Before March 2021, *it was a requirement* to use the FILTER function inside CALCULATE in the following two situations:
 1. When the filter includes *more than one column* from the same table
 2. When the filter includes *an expression*



CALCULATE

- However, it is now possible to omit the FILTER function when filtering two or more columns in the same table, but depending on slicer selections, the measure can still fail. It is also now possible to omit FILTER if the expression is a simple Boolean test using an aggregate function, such as AVERAGE, but using any other expression in the filter argument still requires the use of FILTER.
- With this in mind, let's return to our "Sales for Red or French #1" measure we authored when exploring the CALCULATE function. This was the measure:

Sales for Red or French #1 =

```
CALCULATE (
    [Total Sales],
    Wines[TYPE] = "red"
    || Wines[WINE COUNTRY] = "France" )
```

WINE	Sales for Red or French #1
Bordeaux	\$4,055,250
Champagne	\$7,373,700
Chardonnay	\$4,203,000
Chenin Blanc	\$1,236,950
Chianti	\$1,092,920
Grenache	\$1,078,950
Malbec	\$2,914,650
Merlot	\$900,276
Rioja	\$1,527,795
Shiraz	\$1,364,766
Total	\$25,748,257

FILTER AOK — MOO

SALESPERSON	Total Sales	Sales for Red or French #1
Abel	\$2,050,276	\$4,647,576
Blanchet	\$1,734,279	\$4,193,329
Charron	\$2,029,616	\$4,583,416
Denis	\$2,711,085	\$4,518,335
Leblanc	\$2,232,097	\$4,220,947
Reyer	\$2,177,254	\$3,584,654
Total	\$12,934,607	\$25,748,257

TYPE
 Red
 White

Omitting FILTER can return incorrect results



- We established that the root of the problem lies in the fact that we're using two *different* columns in our filter and indeed in earlier days, we were prevented from authoring such code. To resolve this problem, we need to use the table function FILTER inside CALCULATE. So let's now get to grips with how we can use FILTER in this context and use it to author the correct version of the measure, "Sales for Red or French #2":

Sales for Red or French #2=

`CALCULATE (`

`[Total Sales],`

`FILTER (Wines, Wines[TYPE] = "red"`

`|| Wines[WINE COUNTRY] = "France"`

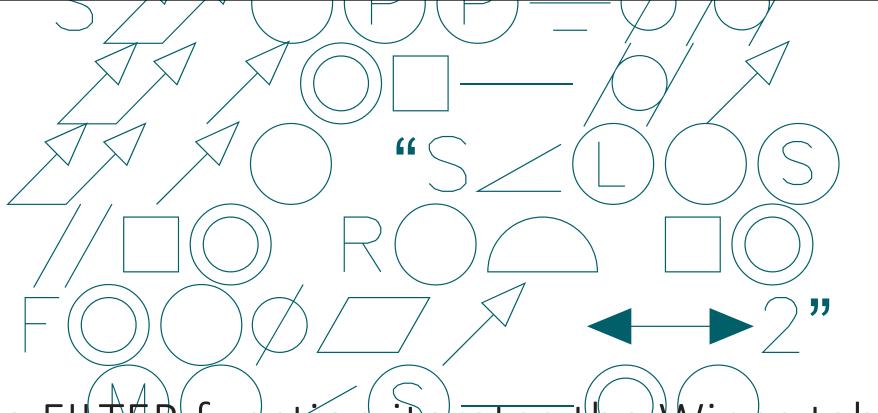
`)`

SALESPERSON Total Sales Sales for Red or French #1 Sales for Red or French #2

SALESPERSON	Total Sales	Sales for Red or French #1	Sales for Red or French #2
Abel	\$1,244,481	\$4,647,576	\$1,244,481
Blanchet	\$967,929	\$4,193,329	\$967,929
Charron	\$1,116,846	\$4,583,416	\$1,116,846
Denis	\$1,573,080	\$4,518,335	\$1,573,080
Leblanc	\$1,224,252	\$4,220,947	\$1,224,252
Reyer	\$1,272,654	\$3,584,654	\$1,272,654
Total	\$7,399,242	\$25,748,257	\$7,399,242

TYPE
█ Red
█ White

WINE COUNTRY
█ France
█ Germany
█ Italy



1. The FILTER function iterates the Wines table in memory and filters any rows where TYPE = "red" or WINECOUNTRY = "France".
2. FILTER generates an in-memory virtual table containing only those rows where the test is true.
3. The virtual table generated by FILTER is used as the filter argument to CALCULATE to filter the Winesales table.

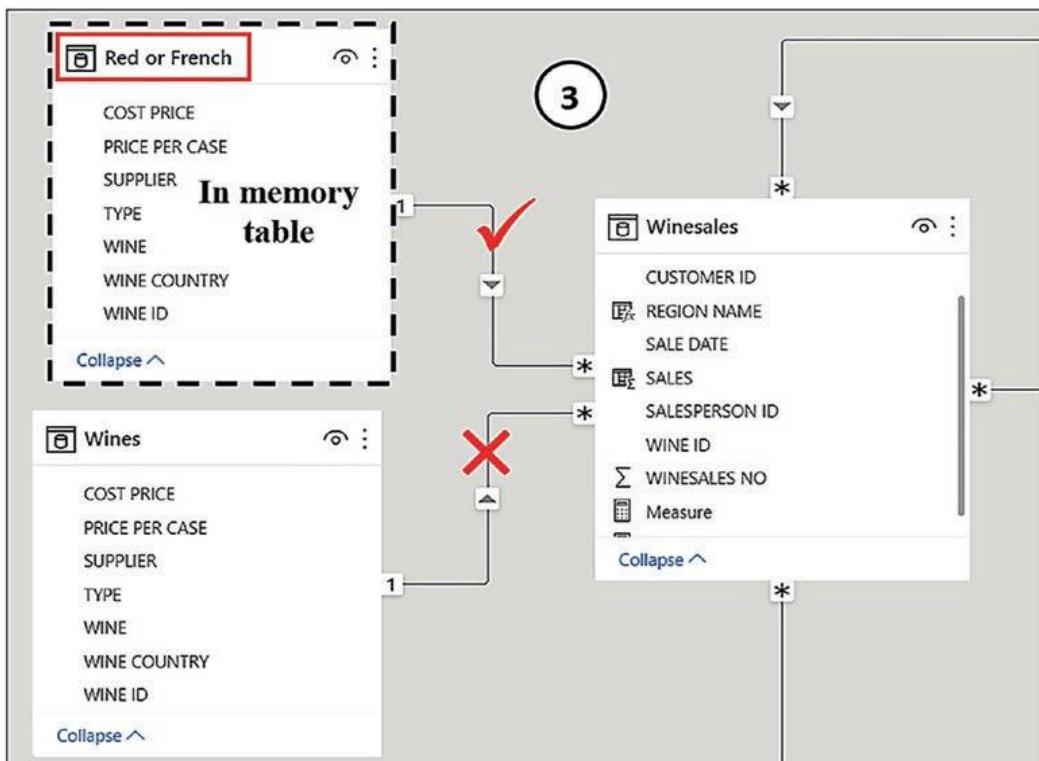
1 Wines dimension

WINE ID	WINE	SUPPLIER	TYPE	WINE COUNTRY	PRICE PER CASE	COST PRICE
1	Bordeaux	Laithwaites	Red	France	£75.00	£25
2	Champagne	Laithwaites	White	France	£150.00	£100
3	Chardonnay	Alliance	White	France	£100.00	£75
4	Malbec	Laithwaites	Red	Germany	£85.00	£40
5	Grenache	Redsky	Red	France	£30.00	£10
6	Piesporter	Redsky	White	Germany	£135.00	£50
7	Chianti	Redsky	Red	Germany	£40.00	£10
8	Pinot Grigio	Majestic	White	Italy	£30.00	£5
9	Merlot	Majestic	Red	France	£39.00	£15
10	Sauvignon Blanc	Majestic	White	Italy	£40.00	£20
11	Rioja	Majestic	Red	Italy	£45.00	£15
12	Chenin Blanc	Alliance	White	France	£50.00	£10
13	Shiraz	Alliance	Red	France	£78.00	£30
14	Lambrusco	Alliance	White	Italy	£20.00	£15

2 FILTER function filters the Wines dimension

WINE ID	WINE	SUPPLIER	TYPE	WINE COUNTRY	PRICE PER CASE	COST PRICE
1	Bordeaux	Laithwaites	Red	France	£75	£25
2	Champagne	Laithwaites	White	France	£150	£100
3	Chardonnay	Alliance	White	France	£100	£75
4	Malbec	Laithwaites	Red	Germany	£85	£40
5	Grenache	Redsky	Red	France	£30	£10
7	Chianti	Redsky	Red	Germany	£40	£10
9	Merlot	Majestic	Red	France	£39	£15
11	Rioja	Majestic	Red	Italy	£45	£15
12	Chenin Blanc	Alliance	White	France	£50	£10
13	Shiraz	Alliance	Red	France	£78	£30

Red or French
in memory
table





- You want to find **the number of sales** (i.e., the number of rows in the Winesales table) for **high profit wines**. High profit wines are where wines have a price that is three times the cost price. This test involves two columns in the Wines dimension, PRICE PER CASE and COST PRICE, and therefore, it's recommended that you use FILTER. These are the measures you can use:

No. of Sales =

COUNTROWS (Winesales)

No. of Sales of High profit Wines =

CALCULATE (

[No. of Sales],

FILTER (Wines, Wines[PRICE PER CASE] >= Wines[COST PRICE] * 3)

)

SALESPERSON	No. of Sales	No. of Sales of High profit Wines	WINE	No. of Sales	No. of Sales of High profit Wines
Abel	376	173	Bordeaux	180	180
Blanchet	343	160	Champagne	132	
Charron	347	168	Chardonnay	187	
Denis	435	228	Chenin Blanc	200	200
Leblanc	355	166	Chianti	148	148
Reyer	351	180	Grenache	182	182
Total	2,207	1,075	Malbec	170	
			Merlot	157	
			Piesporter	115	
			Pinot Grigio	168	168
			Rioja	197	197
			Sauvignon Blanc	168	
			Shiraz	203	
			Total	2,207	1,075



- You will notice again that FILTER can be omitted here because expressions using different columns from the same table are now valid. However, take note that if you had a filter on either the PRICE PER CASE column or the COST PRICE column, you would not see correct values being returned. Therefore, it is recommended that you use FILTER nested inside CALCULATE whenever more than one column is being referenced.
- However, we also need FILTER whenever we need to use an *expression* in the filter argument in CALCULATE. We've set out two examples of this requirement where we are calculating the following:
 1. The number of sales where the total sales values are greater than 20,000. We are using the "Total Sales" measure in the filter test.
 2. The number of sales that are greater than the average sales value. To calculate the average sales, we are using the AVERAGEX expression.



Sales Greater than 20K Wrong =

CALCULATE ([No. of Sales], [Total Sales] > 20000)

Sales Greater than 20K =

CALCULATE ([No. of Sales],
FILTER (Winesales, [Total Sales] > 20000))



Sales Greater than Avg Wrong =

```
CALCULATE (
    [No. of Sales],
    [Total Sales]
    > AVERAGEX (
        Winesales,
        Winesales[CASES SOLD] *
        RELATED ( Wines[PRICE PER CASE] )
    )
)
```

Sales Greater than Avg =

```
CALCULATE (
    [No. of Sales],
    FILTER (
        Winesales,
        [Total Sales]
        > AVERAGEX (
            Winesales,
            Winesales[CASES SOLD] *
            RELATED ( Wines[PRICE PER CASE] )
        )
    )
)
```



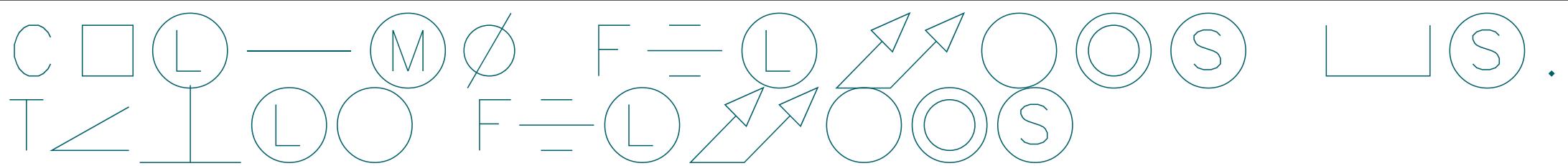
- If the requirement is to calculate the number of sales that are greater than the average cases sold, this expression does not require FILTER because it's using the simple aggregate function AVERAGE.

Cases GT Avg =

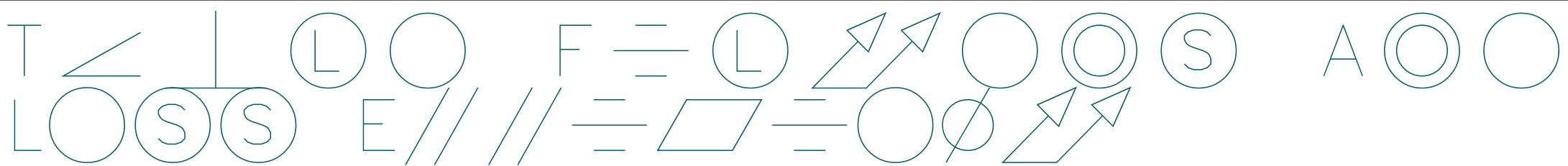
```
CALCULATE (
    [No. of Sales],
    Winesales[CASES SOLD] > AVERAGE ( Winesales[CASES SOLD] )
)
```

Cases GT Avg =

```
CALCULATE (
    [No. of Sales],
    FILTER ( Winesales, Winesales[CASES SOLD]
        > AVERAGE ( Winesales[CASES SOLD] ) )
)
```



- Why do we need to distinguish between table filters and column filters? There are essentially two reasons why this difference is important:
 1. Because the DAX engine has to generate the virtual tables, table filters take longer to process.
 2. Your measure may return a different result depending on the filter type.



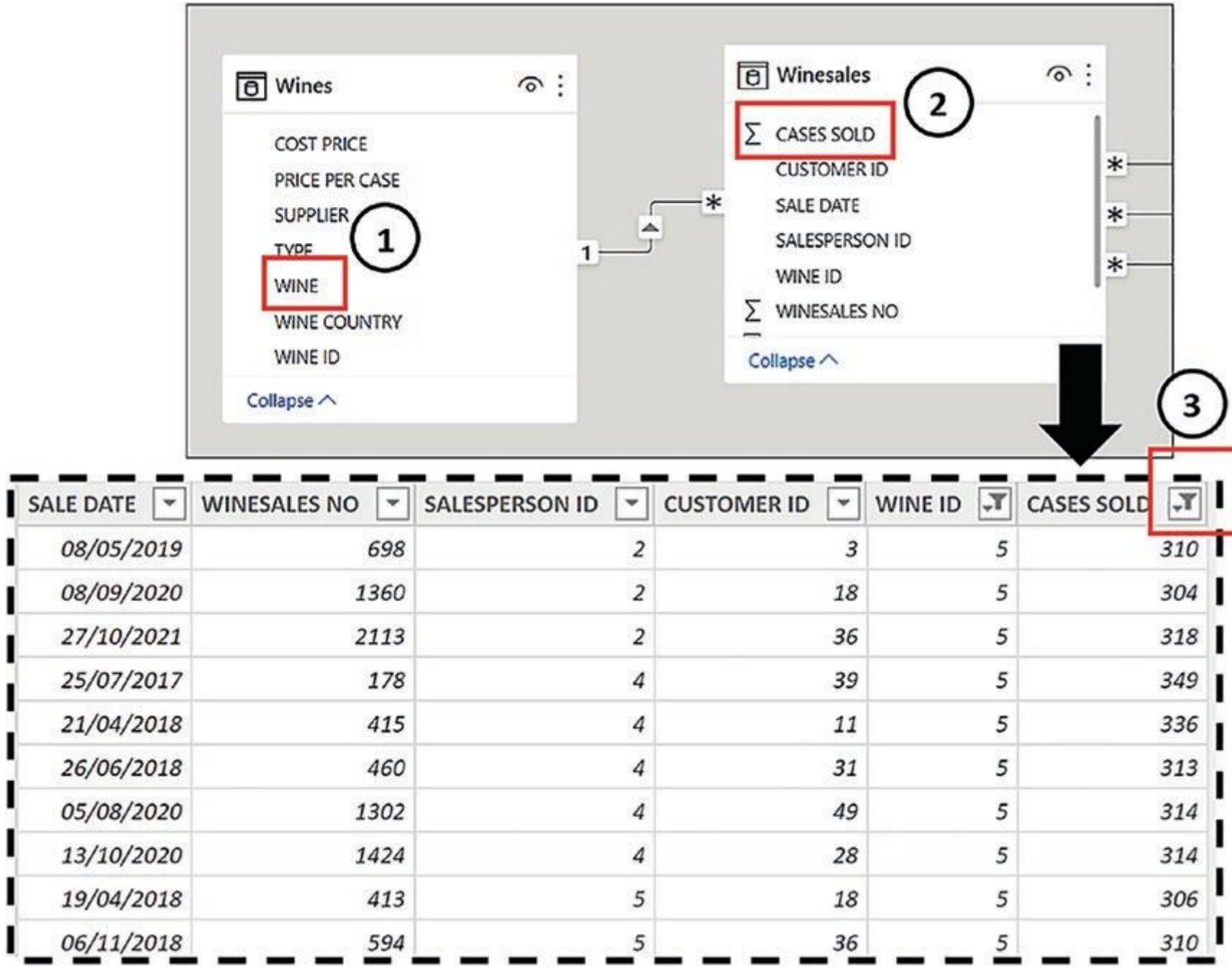
Cases GT 300 #1 =

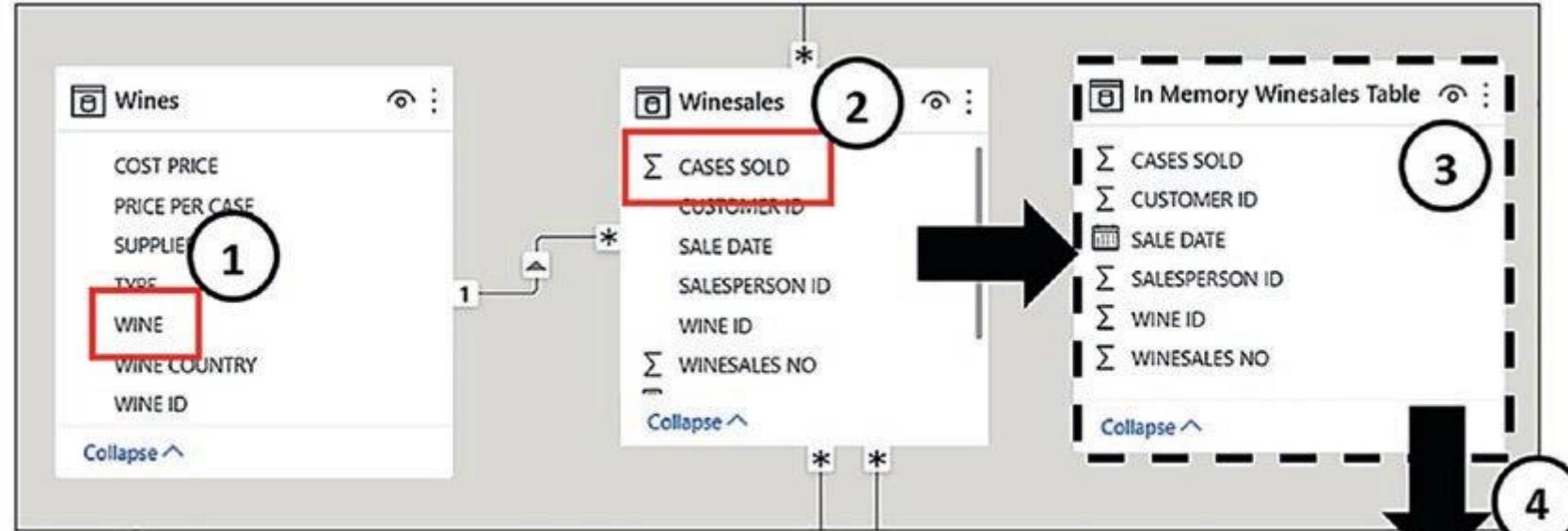
```
CALCULATE ([Total Sales], Winesales[CASES SOLD] > 300 )
```

Cases GT 300 #2 =

```
CALCULATE (
    [Total Sales],
    FILTER ( Winesales, Winesales[CASES SOLD] > 300 )
)
```

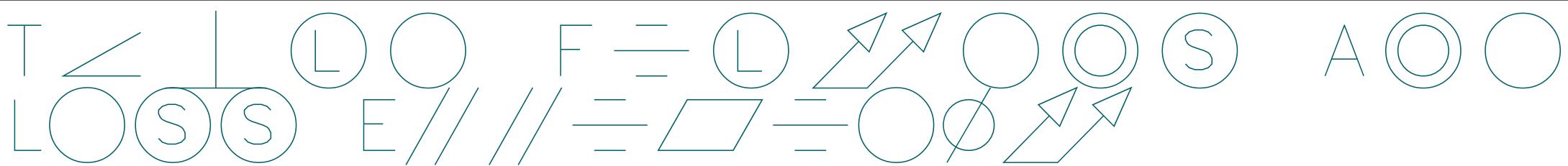
WINE	Cases GT 300 #1	Cases GT 300 #2
Bordeaux	\$2,658,150	\$2,658,150
Champagne	\$6,075,150	\$6,075,150
Grenache	\$310,530	\$310,530
Malbec	\$27,710	\$27,710
Sauvignon Blanc	\$835,280	\$835,280
Total	\$9,906,820	\$9,906,820





A data grid displaying sales information:

SALE DATE	WINESALES NO	SALESPERSON ID	CUSTOMER ID	WINE ID	CASES SOLD
08/08/2021	1966	6	37	5	305
05/03/2021	1678	6	7	5	307
30/04/2020	1123	6	36	5	327
02/04/2020	1069	6	12	5	314
06/12/2018	617	6	29	5	338
21/11/2018	601	6	4	5	341
03/10/2017	234	6	40	5	347
14/09/2017	209	6	18	5	302
20/06/2020	1221	1	14	5	334
08/04/2020	1079	1	17	5	304

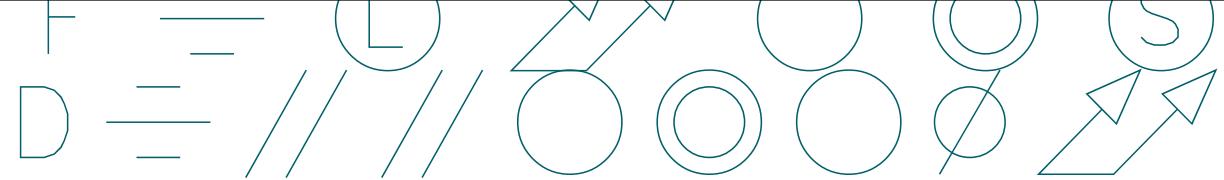
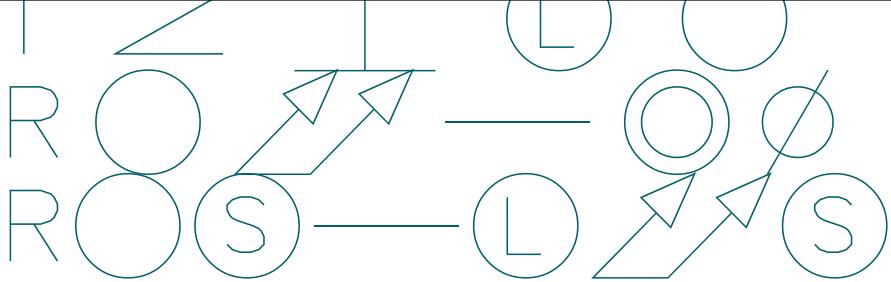


- If your fact table contains many millions of rows, the FILTER function must iterate these rows to build the virtual table. You pay a heavy processing price if you use table filters rather than column filters.
 - *“A side effect of a table filter is that it requires a large materialization to the storage engine to enable the formula engine to compute the result.”*

Cases GT 300 #3 =

SUMX (FILTER (Winesales, Winesales[CASES SOLD] > 300),
[No. of Sales])

- First, FILTER iterates the fact table to generate a table containing the rows to be considered. Then SUMX iterates the table generated by FILTER. That's a lot of iterations!



Bordeaux Wines #1 =

```

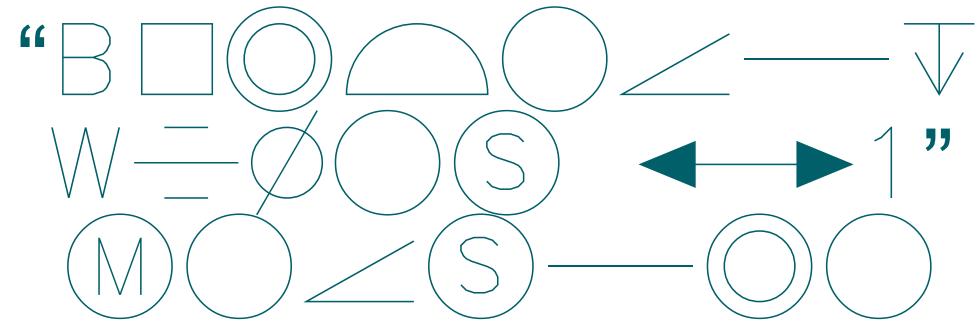
CALCULATE (
  SUM ( Winesales[CASES SOLD] ),
  Wines[WINE] = "Bordeaux" )
  
```

Bordeaux Wines #2 =

```

CALCULATE (
  SUM ( Winesales[CASES SOLD] ),
  FILTER ( Wines, Wines[WINE] = "Bordeaux" )
)
  
```

WINE	Bordeaux Wines #1	Bordeaux Wines #2
Bordeaux	54070	54070
Champagne	54070	
Chardonnay	54070	
Chenin Blanc	54070	
Chianti	54070	
Grenache	54070	
Lambrusco	54070	
Malbec	54070	
Merlot	54070	
Piesporter	54070	
Pinot Grigio	54070	
Rioja	54070	
Sauvignon Blanc	54070	
Shiraz	54070	
Total	54070	54070



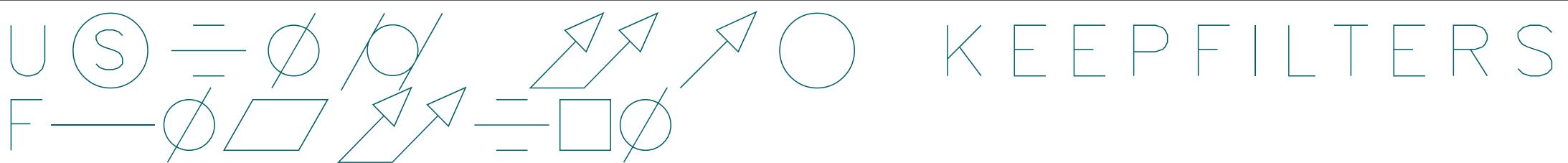
The diagram illustrates a sequence of shapes: a circle with 'B' and 'M', a square, a circle with a smaller concentric circle, a semi-circle, a circle, a triangle pointing left, a horizontal line, a downward arrow, a circle with a diagonal line, a circle, and a circle with 'S'. This sequence is followed by a double-headed arrow labeled '2'.

In Memory Wines Dimension

WINE ID	WINE	SUPPLIE
1	Bordeaux	Laithwaite
2	Champagne	Laithwaite
3	Chardonnay	Alliance

Virtual Wines table generated by FILTER

WINE ID	WINE	SUPPLIE
1	Bordeaux	Laithwa



Bordeaux Wines #1 =

```
CALCULATE (
    SUM ( Winesales[CASES SOLD] ),
    KEEPFILTERS ( Wines[WINE] = "Bordeaux" )
)
```

WINE	Bordeaux Wines #1	Bordeaux Wines #2
Bordeaux	54070	54070
Total	54070	54070

Always use column filters where you can. Only use table filters where necessary.



- The ALL function has the following syntaxes:

= ALL (**table**)

where:

table is the table from where you want to clear the filters.

Here is an example of the ALL function syntax, referencing a table:

= ALL (**Winesales**)

- Or you can reference a column:

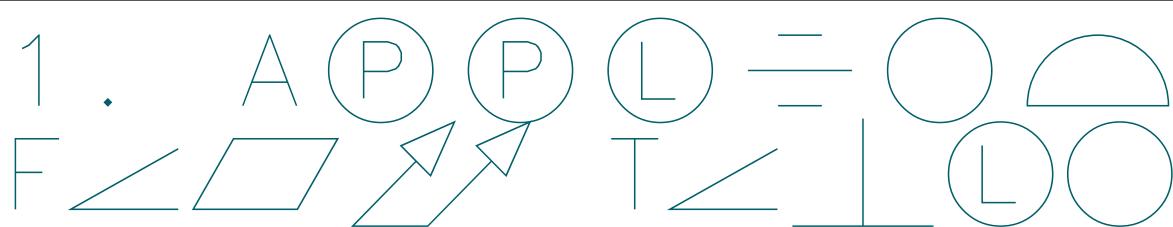
= ALL (**column 1, column 2, etc.**)

where:

column(s) is the column or columns from where you want to clear the filters.

Here is an example of the ALL function syntax referencing a column:

= ALL (**Wines[TYPE]**)



No. of Sales =

`COUNTROWS (Winesales)`

Grand Total No. of Sales =

`COUNTROWS (ALL (Winesales))`

No. of Sales as Percent of Grand Total =

`DIVIDE ([No. of Sales] , [Grand Total No. of Sales])`

WINE	No. of Sales	Grand Total No. of Sales	No. of Sales as Percent of Grand Total
Bordeaux	180	2207	0.08
Champagne	132	2207	0.06
Chardonnay	187	2207	0.08
Chenin Blanc	200	2207	0.09
Chianti	148	2207	0.07
Grenache	182	2207	0.08
Lambrusco		2207	
Malbec	170	2207	0.08
Merlot	157	2207	0.07
Piesporter	115	2207	0.05
Pinot Grigio	168	2207	0.08
Rioja	197	2207	0.09
Sauvignon Blanc	168	2207	0.08
Shiraz	203	2207	0.09
Total	2207	2207	1.00

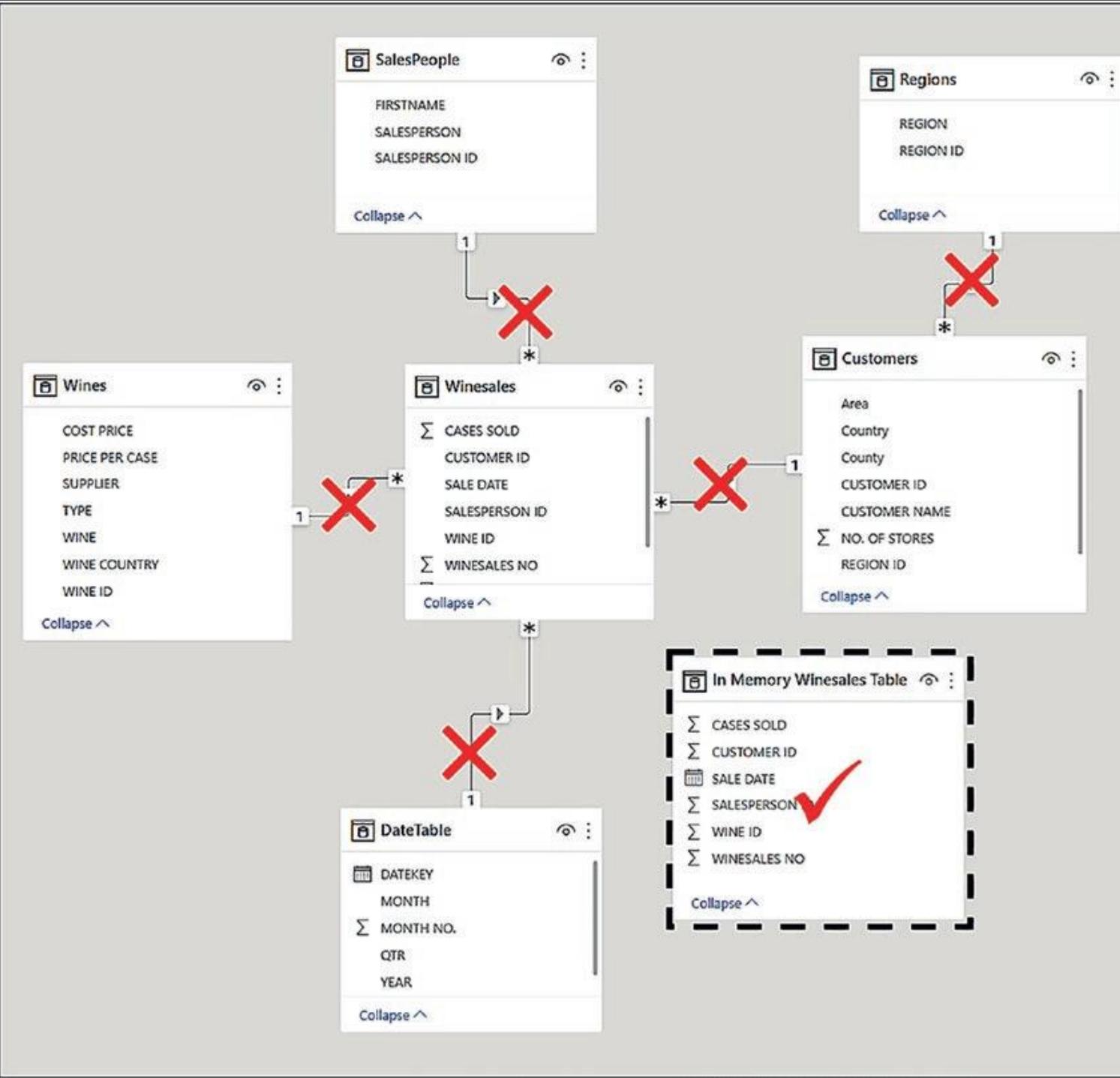
SALESPERSON

 Abel Blanchet Charron Denis Leblanc Reyer

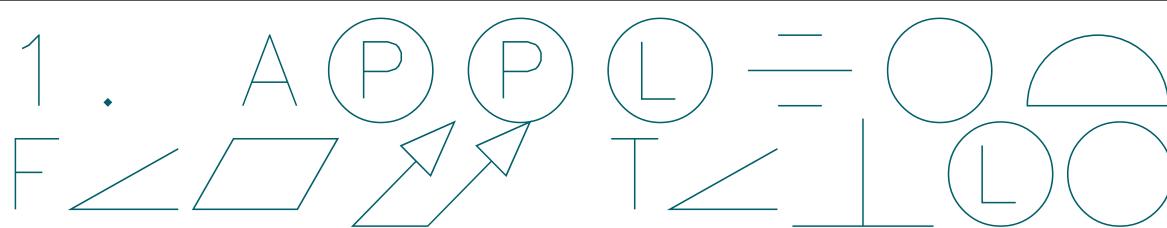
WINE

No. of
SalesGrand Total
No. of SalesNo. of Sales as
Percent of
Grand Total

		No. of Sales	Grand Total No. of Sales	No. of Sales as Percent of Grand Total
	Bordeaux	30	2207	0.01
	Champagne	29	2207	0.01
	Chardonnay	36	2207	0.02
	Chenin Blanc	25	2207	0.01
	Chianti	24	2207	0.01
	Grenache	30	2207	0.01
	Lambrusco		2207	
	Malbec	25	2207	0.01
	Merlot	31	2207	0.01
	Piesporter	25	2207	0.01
	Pinot Grigio	30	2207	0.01
	Rioja	34	2207	0.02
	Sauvignon Blanc	19	2207	0.01
	Shiraz	38	2207	0.02
	Total	376	2207	0.17



The ALL function passed to the fact table generates a virtual fact table that is used for all evaluations, and any filters from dimensions are ignored

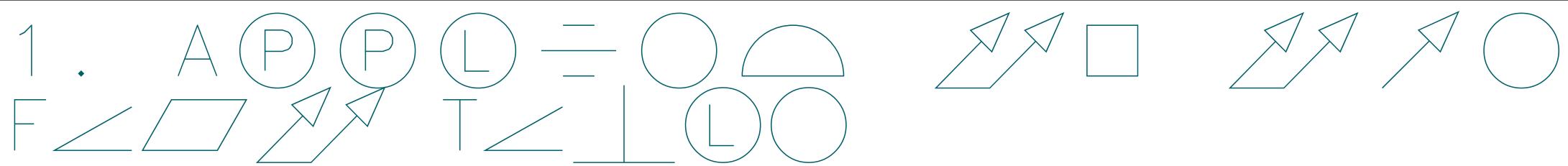


Total Cases All Winesales =

`CALCULATE ([Total Cases], ALL(Winesales))`

*The ALL function nested inside
CALCULATE to find the grand total of
cases sold*

WINE	Total Cases	Total Cases All Winesales
Bordeaux	54070	423224
Champagne	49158	423224
Chardonnay	42030	423224
Chenin Blanc	24739	423224
Chianti	27323	423224
Grenache	35965	423224
Lambrusco		423224
Malbec	34290	423224
Merlot	23084	423224
Piesporter	10253	423224
Pinot Grigio	23449	423224
Rioja	33951	423224
Sauvignon Blanc	47415	423224
Shiraz	17497	423224
Total	423224	423224



Avg Cases All Winesales =

```
CALCULATE( AVERAGE ( Winesales[CASES SOLD] ), ALL ( Winesales ) )
```

No. of Sales Where Cases is GT Avg All Wines =

```
CALCULATE (
    [No. of Sales],
    FILTER ( Winesales, Winesales[CASES SOLD]
        >= [Avg Cases All Winesales] )
)
```

WINE	Total Cases Winesales	Total Cases All Winesales	Avg Cases All Winesales	Avg Cases Where Cases is GT	No. of Sales All Wines
Bordeaux	54070	423224	300.39	191.76	145
Champagne	49158	423224	372.41	191.76	131
Chardonnay	42030	423224	224.76	191.76	185
Chenin Blanc	24739	423224	123.70	191.76	
Chianti	27323	423224	184.61	191.76	70
Grenache	35965	423224	197.61	191.76	91
Lambrusco		423224		191.76	
Malbec	34290	423224	201.71	191.76	88
Merlot	23084	423224	147.03	191.76	12
Piesporter	10253	423224	89.16	191.76	
Pinot Grigio	23449	423224	139.58	191.76	14
Rioja	33951	423224	172.34	191.76	28
Sauvignon Blanc	47415	423224	282.23	191.76	168
Shiraz	17497	423224	86.19	191.76	
Total	423224	423224	191.76	191.76	932

No. of Sales All Wines Wrong =

COUNTROWS (ALL (Wines))

No. of Sales =

COUNTROWS (Winesales)

No. of Sales All Wines =

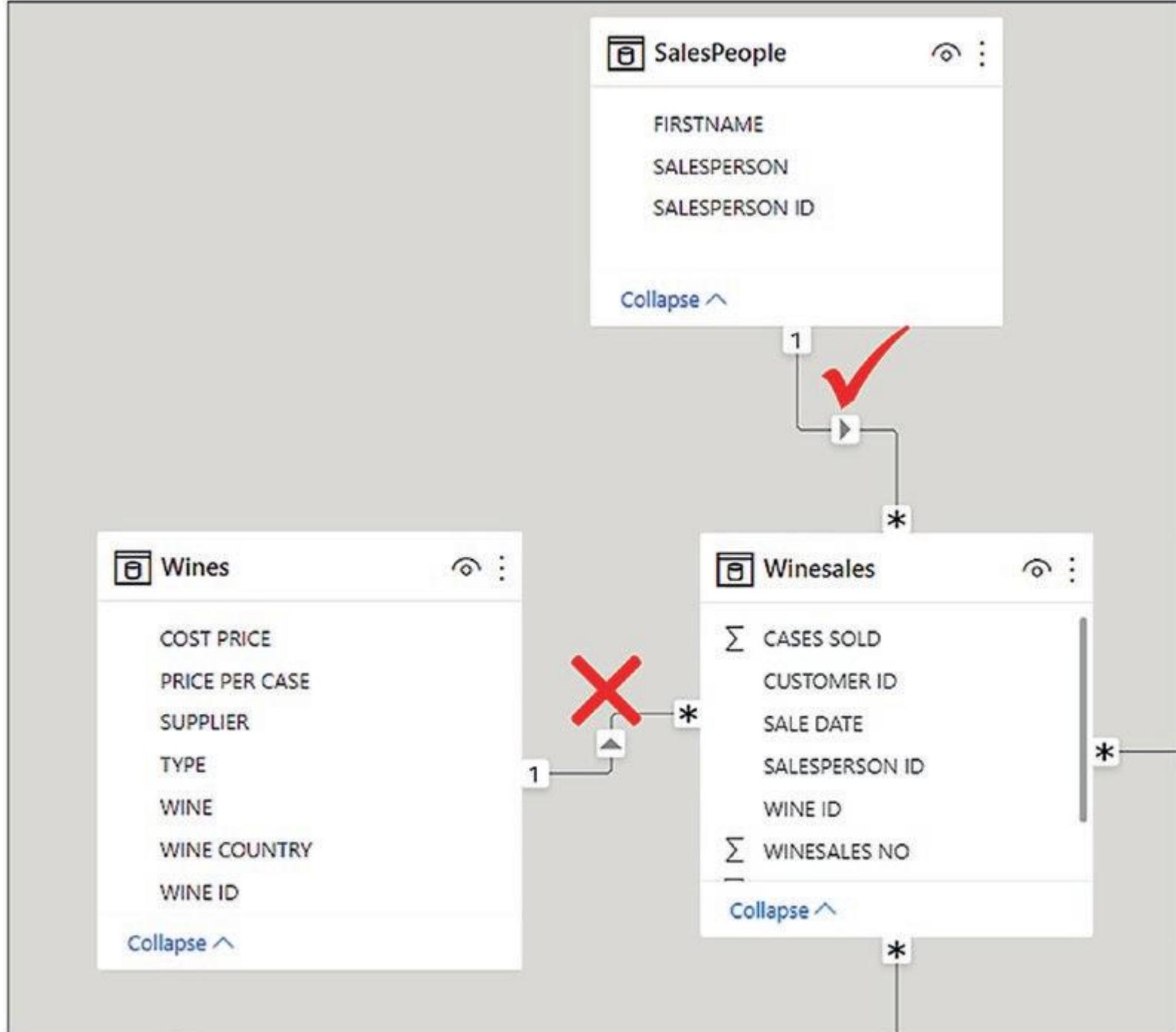
CALCULATE ([No. of Sales], ALL (Wines))

No. of Sales as Percent of Filtered Value =

DIVIDE ([No. of Sales] , [No. of Sales All Wines])

SALESPERSON	WINE	No. of Sales	No. of Sales All Wines	No. of Sales as Percent of Filtered Value
<input checked="" type="checkbox"/> Abel	Bordeaux	30	376	7.98%
<input type="checkbox"/> Blanchet	Champagne	29	376	7.71%
<input type="checkbox"/> Charron	Chardonnay	36	376	9.57%
<input type="checkbox"/> Denis	Chenin Blanc	25	376	6.65%
<input type="checkbox"/> Leblanc	Chianti	24	376	6.38%
<input type="checkbox"/> Reyer	Grenache	30	376	7.98%
	Lambrusco		376	
	Malbec	25	376	6.65%
	Merlot	31	376	8.24%
	Piesporter	25	376	6.65%
	Pinot Grigio	30	376	7.98%
	Rioja	34	376	9.04%
	Sauvignon Blanc	19	376	5.05%
	Shiraz	38	376	10.11%
	Total	376	376	100.00%

The ALL function removes filters from the Wines dimension, but other filters are preserved



3. $\text{US} = \emptyset / \emptyset$ ALL $\square \emptyset \angle C \square L \rightarrow M \emptyset$

SALESPERSON	WINE	No. of Sales	No. of Sales All Wines	No. of Sales as Percent of Filtered Value
<input checked="" type="checkbox"/> Abel	Chardonnay	36	376	9.57%
<input type="checkbox"/> Blanchet	Chenin Blanc	25	376	6.65%
<input type="checkbox"/> Charron	Lambrusco		376	
<input type="checkbox"/> Denis	Shiraz	38	376	10.11%
<input type="checkbox"/> Leblanc	Total	99	376	26.33%
<input type="checkbox"/> Reyer				
SUPPLIER				
<input checked="" type="checkbox"/> Alliance				
<input type="checkbox"/> Laithwaites				
<input type="checkbox"/> Majestic				
<input type="checkbox"/> Redsky				

ALL that references a table will remove filters from all columns in a table, which may be incorrect

3. $\text{US} = \emptyset / \forall \text{ ALL } \square \emptyset \angle \text{ CL} \rightarrow \text{M} \emptyset$

WINE ID	WINE	SUPPLIER	TYPE	WINE COUNTRY	PRICE PER CASE	COST PRICE
3	Chardonnay	Alliance	White	France	\$100.00	\$75.00

Filters are on both the WINE column and the SUPPLIER column

3. US = Ø Ø ALL Ø Ø < C Ø L — M Ø

No. of Sales All Wines #2 =

CALCULATE ([No. of Sales] , ALL (Wines[WINE]))

WINE ID	WINE	SUPPLIER	TYPE	WINE COUNTRY	PRICE PER CASE	COST PRICE
14	Lambrusco	Alliance	White	Italy	\$20.00	\$15.00
13	Shiraz	Alliance	Red	France	\$78.00	\$30.00
12	Chenin Blanc	Alliance	White	France	\$50.00	\$10.00
3	Chardonnay	Alliance	White	France	\$100.00	\$75.00

Using ALL on a column removes the filter from that column only

3. $\text{US} = \phi / \phi$ ALL $\square \phi \angle \text{C} \square \text{L} \rightarrow \text{M} \phi$

No. of Sales as Percent of Filtered Value #2 =

DIVIDE ([No. of Sales] , [No. of Sales All Wines #2])

SALESPERSON	WINE	No. of Sales	No. of Wines #2	No. of Sales All Wines #2	No. of Sales as Percent of Filtered Value #2
■ Abel	Chardonnay	36	99	99	36.36%
□ Blanchet	Chenin Blanc	25	99	99	25.25%
□ Charron	Lambrusco		99	99	
□ Denis	Shiraz	38	99	99	38.38%
□ Leblanc					
□ Reyer	Total	99	99	99	100.00%
SUPPLIER					
■ Alliance					
□ Laithwaites					
□ Majestic					
□ Redsky					

The correct percentage for sales for "Abel" for "Alliance" supplier

3. US = Ø / ALL Ø < C Ø L — M Ø

All Wines Type =

CALCULATE ([Total Cases], ALL (Wines[TYPE]))

Percentage of Wine Country =

DIVIDE ([Total Cases] , [All Wines Type])

*Calculating
percentages across
grouped data*

WINE COUNTRY	Total Cases	All Wines Type	Percentage of Wine Country
France	246,543	246,543	100.00%
Red	130,616	246,543	52.98%
White	115,927	246,543	47.02%
Germany	71,866	71,866	100.00%
Red	61,613	71,866	85.73%
White	10,253	71,866	14.27%
Italy	104,815	104,815	100.00%
Red	33,951	104,815	32.39%
White	70,864	104,815	67.61%
Total	423,224	423,224	100.00%

Total Cases and Percentage of Wine Country by WINE COUNTRY and TYPE



TYPE • Red • White

300K

250K

200K

150K

100K

50K

Total Cases

0K



WINE COUNTRY France

TYPE White

Total Cases 115,927

Percentage of Wine Country 47.02%

France

Italy

Germany

WINE COUNTRY

*Stacked column chart showing the percentage breakdown across
WINE COUNTRY in the Tooltip*

WINE ID	WINE	SUPPLIER	TYPE	WINE COUNTRY	PRICE PER CASE	COST PRICE
13	Shiraz	Alliance	Red	France	\$78.00	\$30.00
9	Merlot	Majestic	Red	France	\$39.00	\$15.00
5	Grenache	Redsky	Red	France	\$30.00	\$10.00
1	Bordeaux	Laithwaites	Red	France	\$75.00	\$25.00

WINE ID	WINE	SUPPLIER	TYPE	WINE COUNTRY	PRICE PER CASE	COST PRICE
13	Shiraz	Alliance	Red	France	\$78.00	\$30.00
12	Chenin Blanc	Alliance	White	France	\$50.00	\$10.00
9	Merlot	Majestic	Red	France	\$39.00	\$15.00
5	Grenache	Redsky	Red	France	\$30.00	\$10.00
3	Chardonnay	Alliance	White	France	\$100.00	\$75.00
2	Champagne	Laithwaites	White	France	\$150.00	\$100.00
1	Bordeaux	Laithwaites	Red	France	\$75.00	\$25.00

Using ALL on the TYPE column removes the filter from only that column

3. $\text{US} = \phi / \text{ALL} \quad \square \phi \angle \text{CL} - \text{M} \phi$

All Wines Type, Supplier & Wine =

CALCULATE ([Total Cases],
ALL (Wines[TYPE], Wines[SUPPLIER], Wines[WINE])
)

All Except Wine Country =

CALCULATE ([Total Cases],
ALLEXCEPT (Wines, Wines[WINE COUNTRY])
)

Percentage of Wine Country #2=

DIVIDE ([Total Cases] , [All Except Wine Country])

WINE COUNTRY	Total Cases	All Wines Type, Supplier & Wine	Percentage of Wine Country #2
France	246,543	246,543	100.00%
Red	130,616	246,543	52.98%
Laithwaites	54,070	246,543	21.93%
Bordeaux	54,070	246,543	21.93%
Redsky	35,965	246,543	14.59%
Grenache	35,965	246,543	14.59%
Majestic	23,084	246,543	9.36%
Merlot	23,084	246,543	9.36%
Alliance	17,497	246,543	7.10%
Shiraz	17,497	246,543	7.10%
White	115,927	246,543	47.02%
Alliance	66,769	246,543	27.08%
Chardonnay	42,030	246,543	17.05%
Chenin Blanc	24,739	246,543	10.03%
Laithwaites	49,158	246,543	19.94%
Champagne	49,158	246,543	19.94%
Germany	71,866	71,866	100.00%
Red	61,613	71,866	85.73%
Laithwaites	34,290	71,866	47.71%
Malbec	34,290	71,866	47.71%
Total	423,224	423,224	100.00%

T O ALLEXCEPT F — =

- ALLEXCEPT removes all filters in a table except filters that are applied to the columns you specify. This can be used for situations in which you want to remove the filters on many but not all of the columns in a table.
- The ALLEXCEPT function has the following syntax:

= ALLEXCEPT (**table**, **column1**, **column2**, etc.)

where:

table is the table where you want to clear the filters from *except* the filters on the columns specified in the next arguments.

column1, **column2** are the columns where you want filters preserved.

- Here is an example of the ALLEXCEPT syntax:

= ALLEXCEPT (**Wines**, **Wines[WINE COUNTRY]**)

T↗ O ALLEXCEPT F—∅□△↗=□∅

The “Grand Total No. of Sales” measure is not the total for the selected wines in the slicer

WINE	No. of Sales	Grand Total No. of Sales	WINE
Bordeaux	180	2207	Bordeaux
Champagne	132	2207	Champagne
Chardonnay	187	2207	Chardonnay
Chenin Blanc	200	2207	Chenin Blanc
Total	699	2207	Chianti
			Grenache
			Lambrusco
			Malbec
			Merlot
			Piesporter
			Pinot Grigio
			Rioja
			Sauvignon Blanc
			Shiraz

We need to find a function that specifically finds grand totals *for the items that have been filtered in the visual.* The function we need is called **ALLSELECTED**.

T O ALLSELECTED F —

- The syntax for the ALLSELECTED function is the same as for the ALL function:

= ALLSELECTED (table)

or

= ALLSELECTED (Column 1, Column 2, etc.)

“ALLSELECTED removes context filters from columns and rows in the current query, while retaining all other context filters or explicit filters. The ALLSELECTED function gets the context that represents all rows and columns in the query, while keeping explicit filters and contexts other than row and column filters. This function can be used to obtain visual totals in queries.”

(DAX Function Library)

T ↗ O ALLSELECTED F — ⊖ □ ⊞ ⊞ = □ ⊖

Grand Total No. of Sales for Selected Wines =

CALCULATE ([No. of Sales], ALLSELECTED (Wines[WINE]))

WINE	No. of Sales	Grand Total No. of Sales for Selected Wines	WINE
Bordeaux	180	699	Bordeaux
Champagne	132	699	Champagne
Chardonnay	187	699	Chardonnay
Chenin Blanc	200	699	Chenin Blanc
Total	699	699	Chianti
			Grenache
			Lambrusco
			Malbec
			Merlot
			Piesporter
			Pinot Grigio
			Rioja
			Sauvignon Blanc
			Shiraz

The screenshot illustrates the use of the ALLSELECTED function in Power BI. It shows two tables and a slicer. The top table has one row with Bordeaux wine. The bottom table has four rows: Chenin Blanc, Chardonnay, Champagne, and Bordeaux. Red boxes highlight the filter icons in the Wine column of both tables. A large black arrow points from the Wine column of the bottom table to a slicer on the right. The slicer shows a list of wine types: Bordeaux, Champagne, Chardonnay, Chenin Blanc, Chianti, Grenache, Lambrusco, and Malbec. The first four are selected and highlighted with a red border.

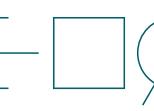
WINE ID	WINE	SUPPLIER	TYPE	WINE COU
1	Bordeaux	Laithwaites	Red	France

WINE ID	WINE	SUPPLIER	TYPE	WINE COU
12	Chenin Blanc	Alliance	White	France
3	Chardonnay			
2	Champagne	Laithwaites	White	France
1	Bordeaux	Laithwaites	Red	France

WINE

- Bordeaux
- Champagne
- Chardonnay
- Chenin Blanc
- Chianti
- Grenache
- Lambrusco
- Malbec

The ALLSELECTED function replaces the filter to reflect the slicer selections

T  ALLSELECTED F —     

- Up to now, we've been using the ALL function (and its variations) while not considering whether it's being used as a *table function* or is being used as a *modifier* to CALCULATE. The "ALL" functions seem to be doing their job, and we're thankful for that.
- We know that ALL removes filters whether by removing filters from tables and columns or by generating virtual tables containing all the rows. However, we are now going to focus our attention on the difference between ALL as a table function and ALL as a modifier to CALCULATE.

ALL < S < M □ ⚹ = // = ○ ○ ↗ □

No. of Wines #1 =

COUNTROWS (ALL (Wines[WINE]))

No. of Wines #2 =

CALCULATE (COUNTROWS (Wines), ALL (Wines[WINE]))

WINE	No. of Wines #1	No. of Wines #2	SUPPLIER
Chardonnay	14	4	Alliance
Chenin Blanc	14	4	Laithwaites
Lambrusco	14	4	Majestic
Shiraz	14	4	Redsky
Total	14	4	

WINE
Bordeaux
Champagne
Chardonnay
Malbec
Grenache
Piesporter
Chianti
Pinot Grigio
Merlot
Sauvignon Blanc
Rioja
Chenin Blanc
Shiraz
Lambrusco

*ALL as a table function
generates a virtual table of
distinct values*

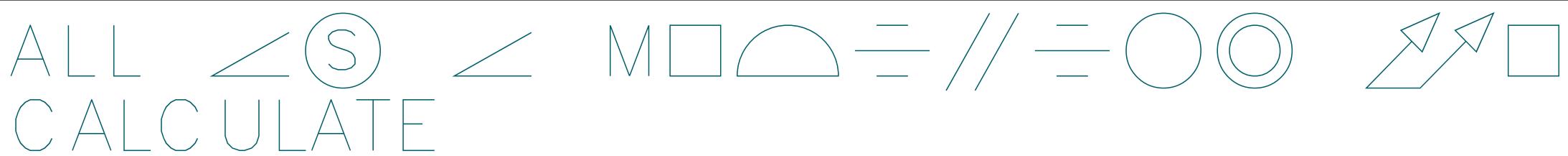
14 Rows

*The ALL function as a CALCULATE modifier
removes the filter on the WINE column*

WINE ID	WINE	SUPPLIER	TYPE	WINE C
3	Chardonnay	Alliance	White	France

WINE ID	WINE	SUPPLIER	TYPE	WINE C
14	Lambrusco	Alliance	White	Italy
13	Shiraz	Alliance	Red	France
12	Chenin Blanc	Alliance	White	France
3	Chardonnay	Alliance	White	France

4 Rows



No. of Sales Where Cases GT 300 #1 =
CALCULATE ([No. of Sales],

ALL (Winesales),
 Winesales[CASES SOLD] > 300
)

No. of Sales Where Cases GT 300 #2 =
CALCULATE (

 [No. of Sales],
 FILTER (
 ALL (Winesales), Winesales[CASES SOLD] >
 300)
)

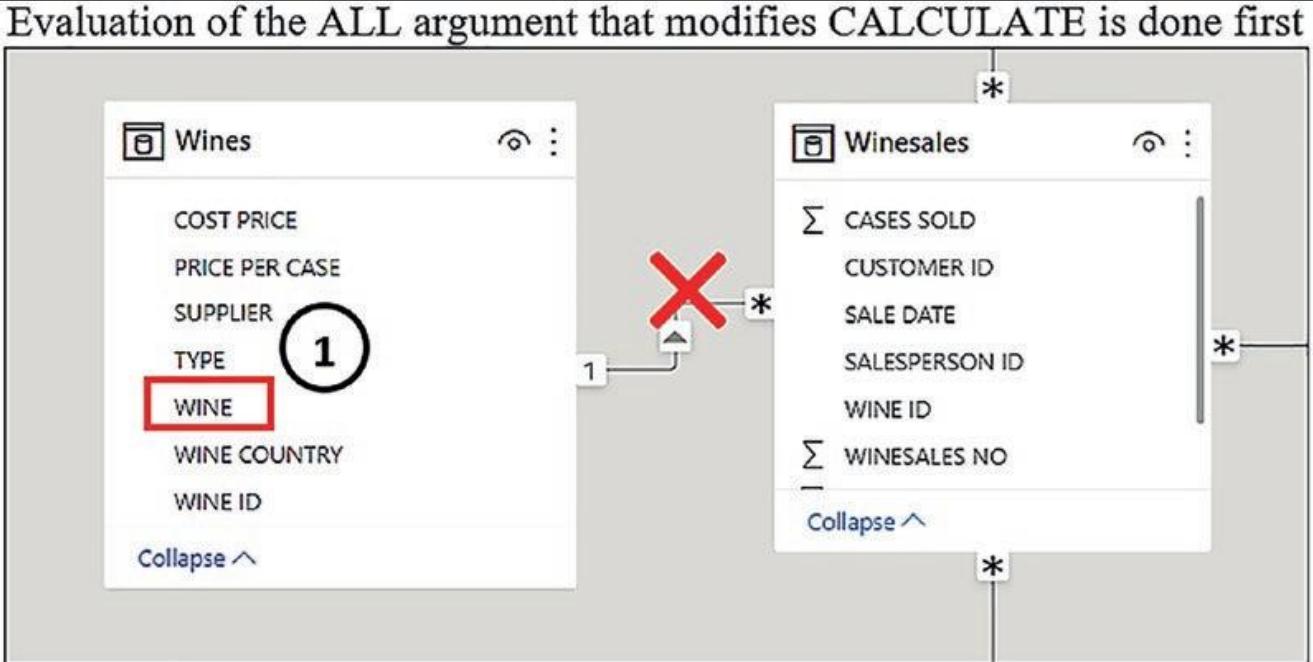
No. of Sales Where Cases GT 300 #3 =
CALCULATE (

 [No. of Sales],
 ALL (Winesales),
 FILTER (Winesales, Winesales[CASES SOLD] >300)
)

WINE	No. of Sales	No. of Sales	No. of Sales
	Where Cases	Where Cases	Where Cases
	GT 300 #1	GT 300 #2	GT 300 #3
Bordeaux	286	286	89
Champagne	286	286	100
Chardonnay	286	286	
Chenin Blanc	286	286	
Chianti	286	286	
Grenache	286	286	32
Lambrusco	286	286	
Malbec	286	286	1
Merlot	286	286	
Piesporter	286	286	
Pinot Grigio	286	286	
Rioja	286	286	
Sauvignon Blanc	286	286	64
Shiraz	286	286	
Total	286	286	286

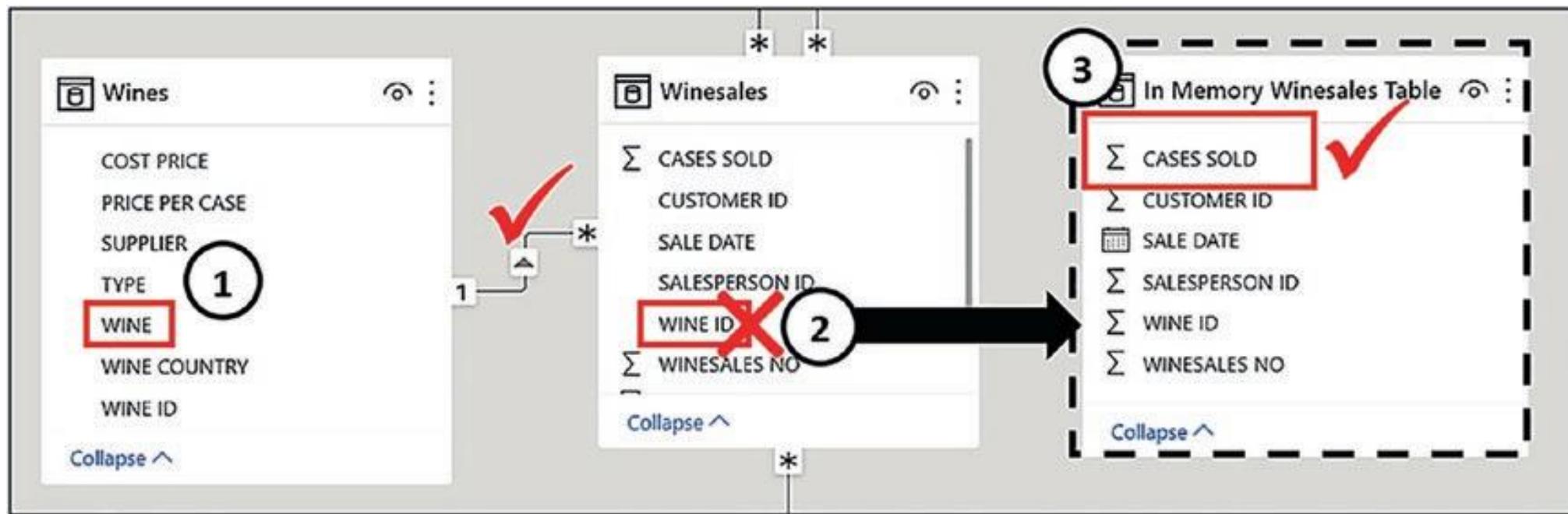
Similar expressions can return different results

The evaluation of "No. of Sales Where Cases GT 300 #1"



Evaluation of the filter argument is done next

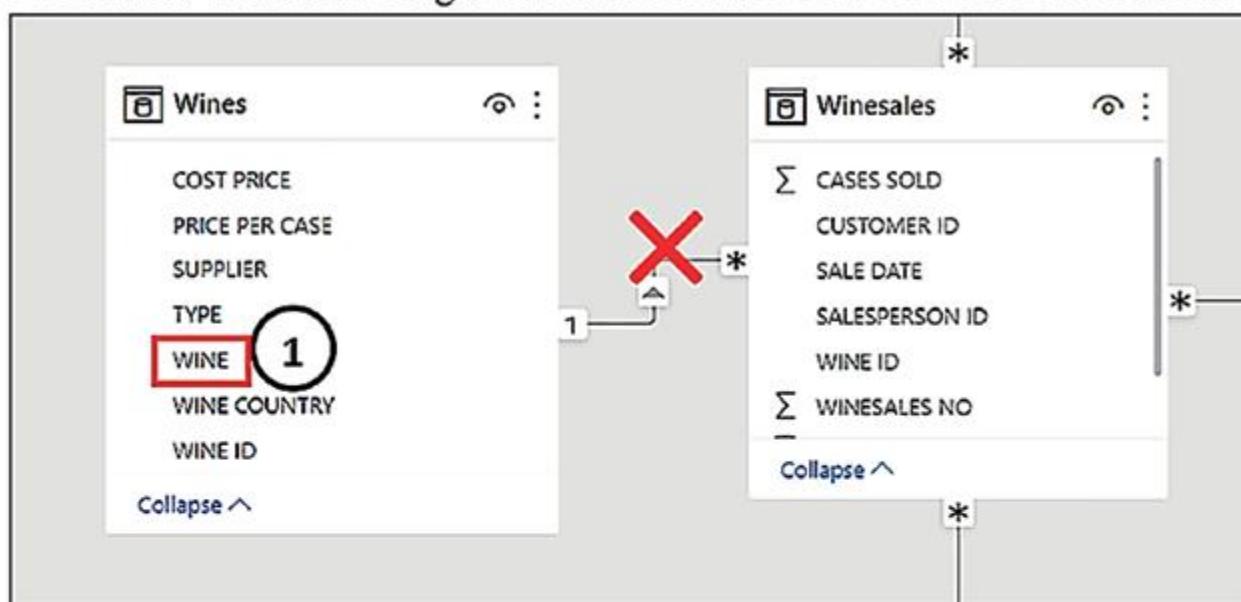




The evaluation of "No. of Sales Where Cases GT 300 #2"

*The evaluation of "No.
of Sales Where Cases
GT 300 #3"*

Evaluation of the ALL argument that modifies CALCULATE is done first

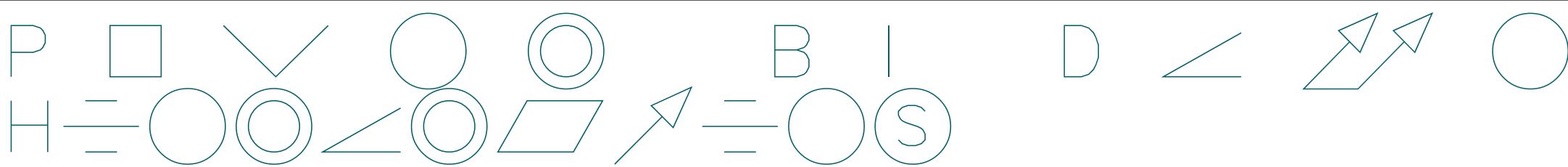


Evaluation of the filter argument using FILTER is done next

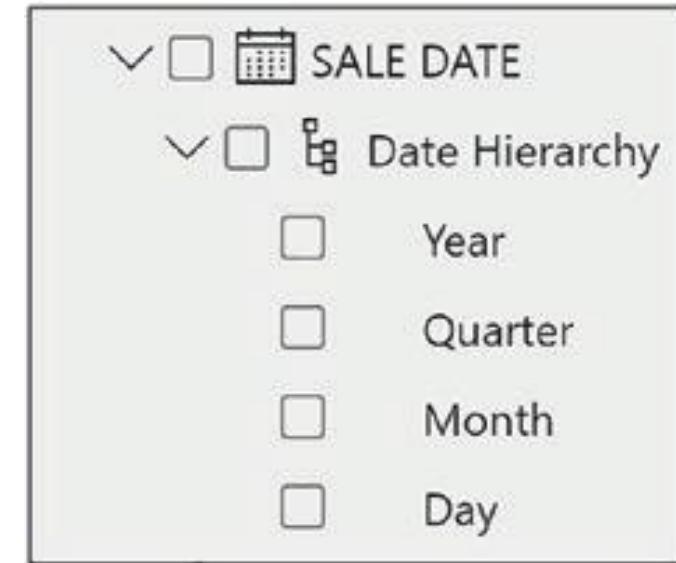


T O ALL // — =

- The ALL function as a *table* function generates a virtual table containing all the rows from a table or all the distinct rows of a column or columns. This virtual table containing all the rows can be *refiltered* by FILTER, and this will then propagate filters through the model as in measure #2 before.
- The ALL function as a modifier to CALCULATE is evaluated first before any filter arguments inside CALCULATE. ALL as a modifier *removes* any filters from a table or a column and generates an empty filter. Any other filter arguments of CALCULATE are then evaluated and generate the new filter context as in measures #1 and #3 before.
- Because ALL has a different behavior when used as a top-level argument to CALCULATE, users believed it should have a different name when used in this context. As a result, in 2019, a new function was introduced into the DAX Function Library, REMOVEFILTERS. This function is synonymous with ALL, but it can be used only as a CALCULATE modifier and not as a table expression like ALL.



- In the absence of a date dimension in your model, if you have columns of a date data type in any tables, for every one of these columns, Power BI will generate an in-memory date table for you that also contains a date hierarchy. We have removed the DateTable dimension from our data model, and so the SALE DATE column is now expressed as a date hierarchy.



Options

GLOBAL

- Data Load
- Power Query Editor
- DirectQuery
- R scripting
- Python scripting
- Security
- Privacy
- Regional Settings
- Updates
- Usage Data
- Diagnostics
- Preview features
- Auto recovery
- Report settings

CURRENT FILE

- Data Load
- Regional Settings
- Privacy
- Auto recovery

Type Detection

- Always detect column types and headers for unstructured sources
- Detect column types and headers for unstructured sources according to each file's setting
- Never detect column types and headers for unstructured sources

Background Data

- Always allow data previews to download in the background
- Allow data previews to download in the background according to each file's setting
- Never allow data previews to download in the background

Parallel loading of tables

When you load data into Power BI (via import or DirectQuery), each data table is backed by a Power Query query. These queries are evaluated simultaneously instead of one-by-one, which can speed up the process. In certain situations, you might want to adjust the default number of simultaneous query evaluations and memory used. [Learn more](#)

8



Maximum number of simultaneous evaluations

432



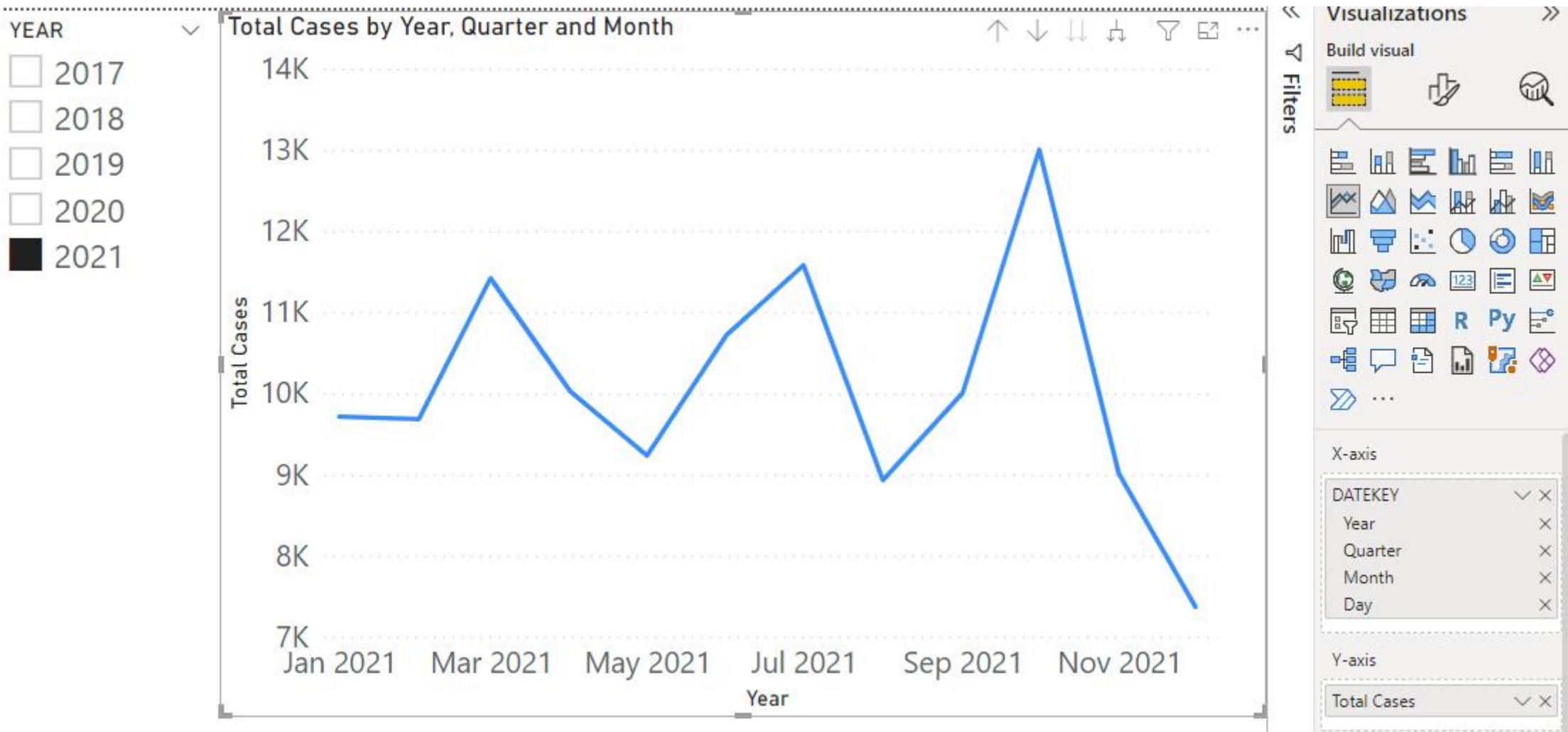
Maximum memory used per simultaneous evaluation (MB)

Time intelligence

- Auto date/time for new files [Learn more](#)

OK

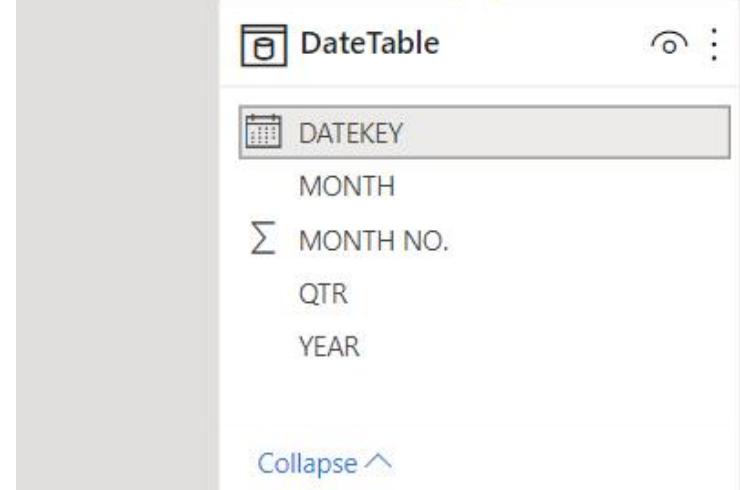
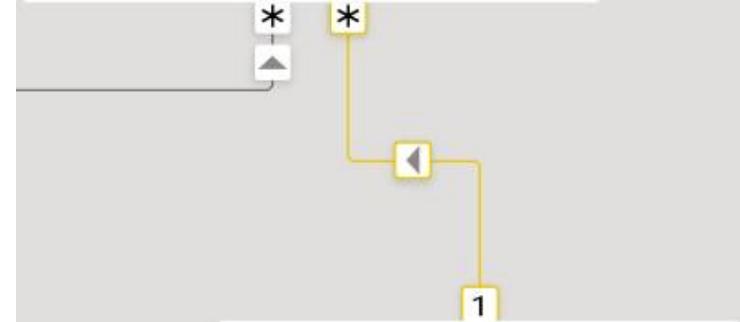
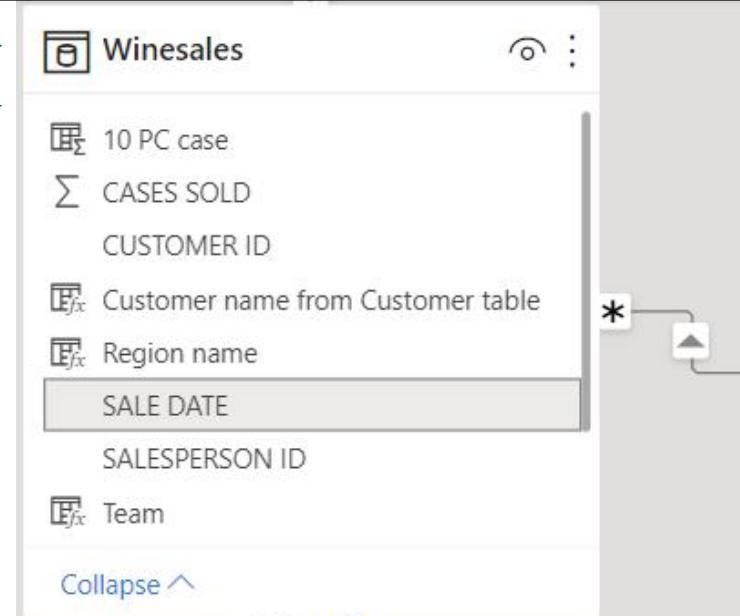
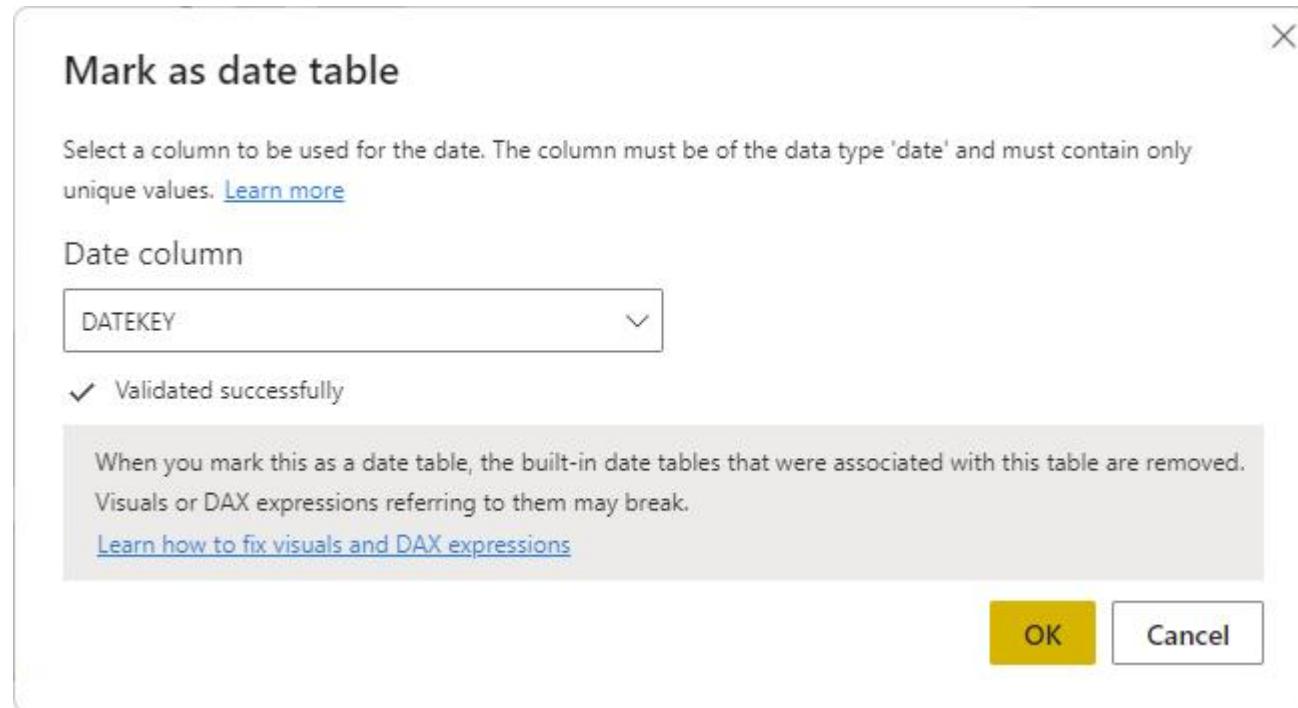
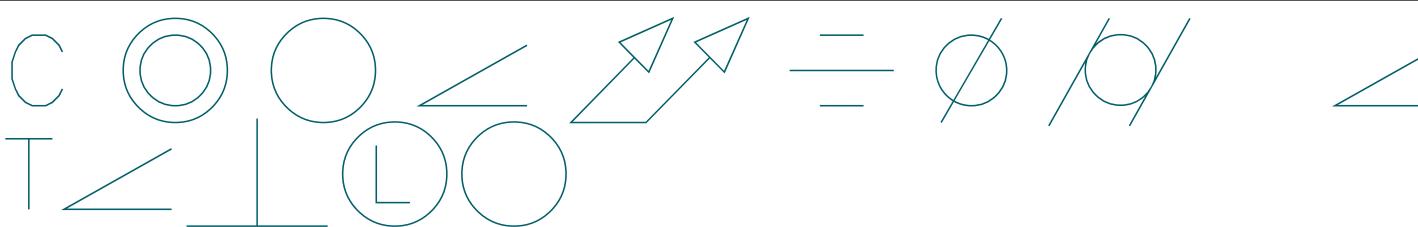
Cancel

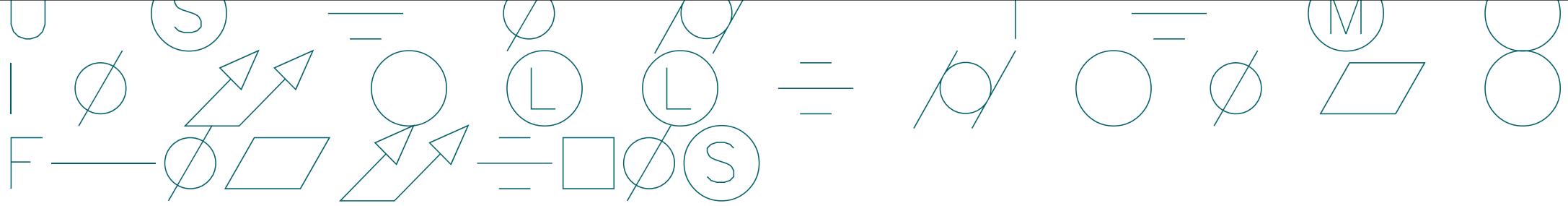




- However, there are a number of drawbacks to using these hierarchies:
 - What if your financial year doesn't start in January?
 - What if you want to analyze sales by week granularity? How would you add week numbers?
 - What if you want to compare sales in 2020 with sales in 2021 in a clustered column chart?

All the preceding problems present a real challenge if you're using built-in date hierarchies, but if you have a date table dimension in your model, life becomes a lot easier as far as date calculations go. Therefore, the first step is generating your date dimension table and integrating it into your data model.

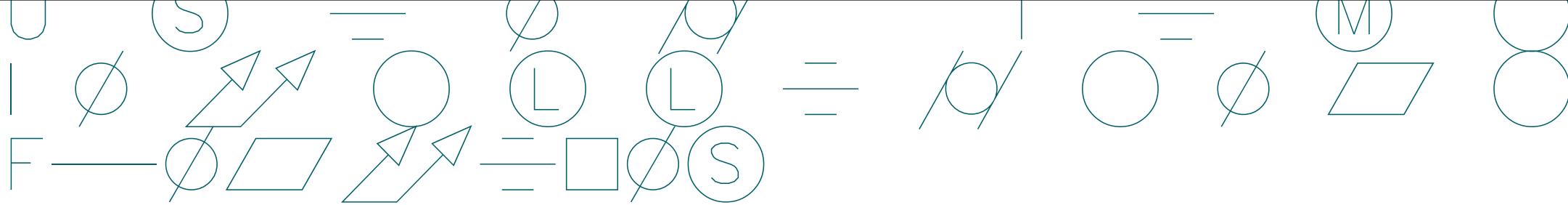




- Time intelligence functions use a *base date* from which to perform the required calculation. This base date is supplied by the current filter context. For example, the terms “previous month” and “same period last year” are relative terms, relative, that is, to the date that is in the current filter context. Therefore, with most of these functions, you must have a specific date filtered (a year, a quarter, a month, or a day) either by using slicers, by using the Filters pane, or by having dates in the visual. For example, if you want to find the previous month’s sales, you must have a current month filtered in the visual or in a slicer.

YEAR	MONTH	Total Cases	Previous Month's Total Cases		WINE	Total Cases	Previous Month's Total Cases		MONTH	YEAR
2017	Jan	6,657			Bordeaux	998	478		Jan	2017
2017	Feb	5,705	6,657		Champagne	251	1,038		Feb	2018
2017	Mar	5,544	5,705		Chardonnay	430			Mar	2019
2017	Apr	5,364	5,544		Chenin Blanc	483	586		Apr	2020
2017	May	4,757	5,364		Chianti	123	619		May	2021
2017	Jun	3,011	4,757		Grenache	916	893		Jun	
2017	Jul	5,079	3,011		Malbec	801	963		Jul	
2017	Aug	3,182	5,079		Piesporter		484		Aug	
2017	Sep	7,279	3,182		Pinot Grigio	338	487		Sep	
2017	Oct	5,602	7,279		Rioja	658	858		Oct	
2017	Nov	5,045	5,602		Sauvignon Blanc	1,465	1,721		Nov	
2017	Dec	7,521	5,045		Shiraz	512	98		Dec	
2018	Jan	6,247	7,521		Total	6,975	8,225			
	Total	421,281								

The “base date” is supplied by the filter context which can be through columns in the visual or by year and month slicers



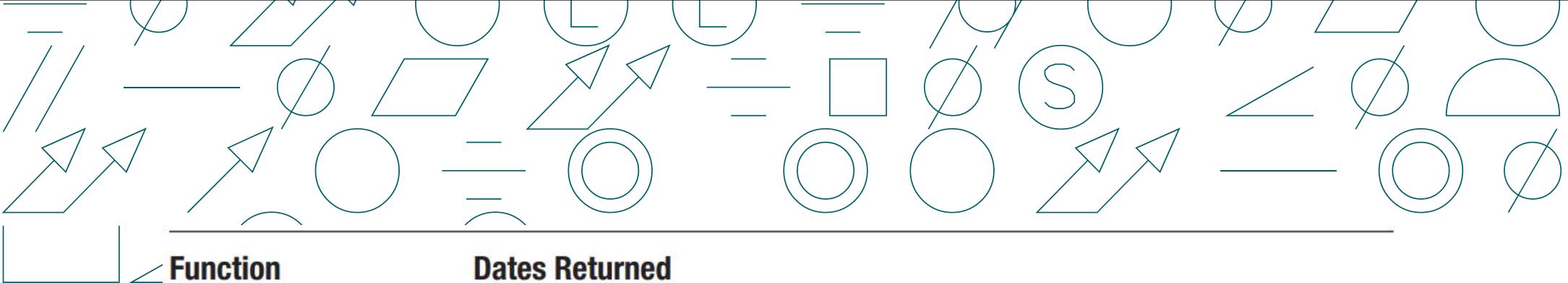
- All time intelligence functions (except LASTNONBLANK and LASTNONBLANKVALUE) have an argument that requires specifying a column of dates to be used in the calculation. In most cases, in this argument, you supply the name of the column in your date table that holds the list of unique dates, for example, the DATEKEY column in our data.

PREVIOUSMONTH(Dates)

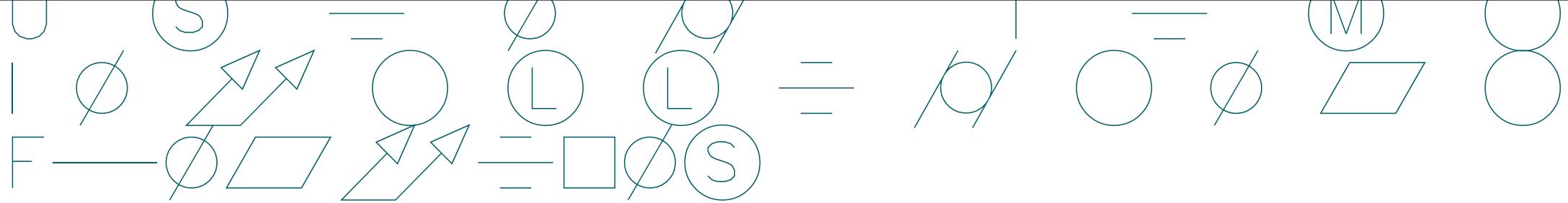
Returns a previous month.

1 Previous Month = CALCULATE([Total Sales],PREVIOUSMONTH(dateTable[DATEKEY]))

The “Dates” argument normally requires referencing the column that holds the list of unique dates



Function	Dates Returned
PREVIOUSMONTH	The previous month from the month in the current filter context.
SAMEPERIODLASTYEAR	The same period last year from the month in the current filter context.
DATEADD	Prior (or future) years, quarters, months, or days from the current filter context.
DATESYTD	The year up to the date in the current filter context.
DATESBETWEEN	Between two dates.
DATESINPERIOD	Starting with a date and then going back (or forward) by any number of years, quarters, months, or days from the current filter context.
LASTDATE	The last date in the current filter context.
LASTNONBLANK	The last date in a column where the expression is nonblank in the current filter context.
LASTNONBLANKVALUE	The last value in a column where the expression is nonblank in the current filter context.



- Typically, time intelligence functions generate a virtual one-column table containing filtered dates from the DATEKEY column in the date dimension (or whatever you've named this column). This virtual table is used as a table filter inside CALCULATE to filter the dates in the fact table.
 - However, DAX time intelligence functions either can be *table* functions that are nested inside CALCULATE as the filter argument or can return *scalar* values. The reason for this is that if a table function returns a one-column, one-row table, this virtual table is converted into a scalar value by the DAX engine.

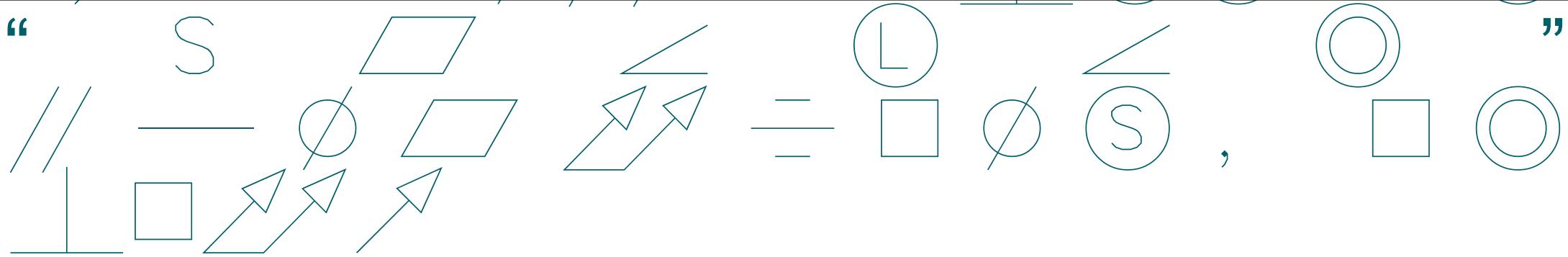


Table	Table or Scalar	Scalar
DATEADD	LASTDATE	LASTNONBLANKVALUE
DATESBETWEEN	LASTNONBLANK	
DATESINPERIOD		
DATESYTD		
PREVIOUSMONTH		
SAMEPERIODLASTYEAR		



LastDate Example #1 =

`LASTDATE(Winesales[SALE DATE])`

LastDate Example #2 =

`CALCULATE([Total Sales],LASTDATE(Winesales[SALE DATE]))`

LastDate Example #3 =

`CALCULATE(LASTDATE(Winesales[SALE DATE]),DateTable[YEAR]=2020)`

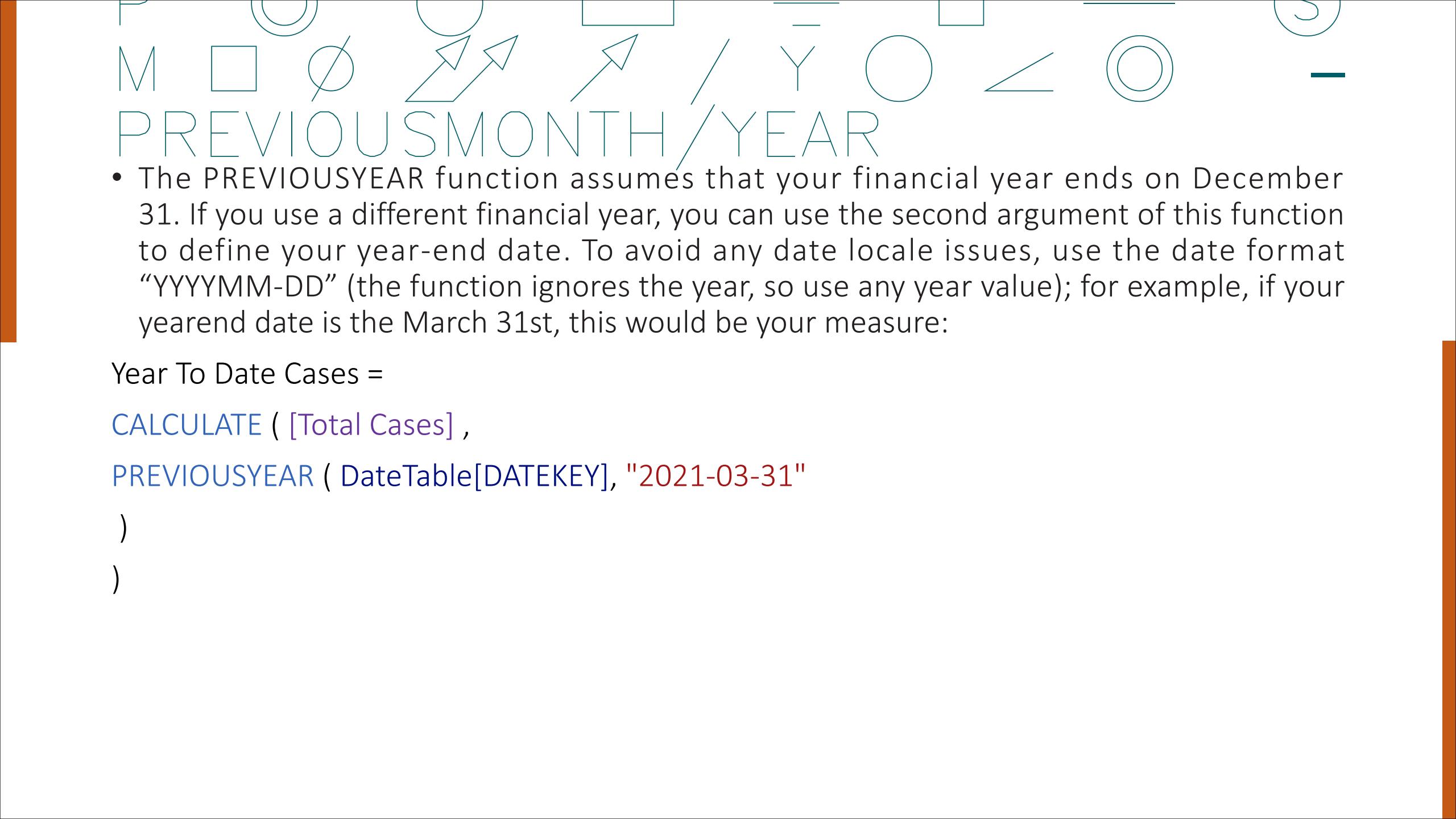
M \square \emptyset PREVIOUSMONTH / YEAR

Previous Month Total Cases =

```
CALCULATE ( [Total Cases],  
    PREVIOUSMONTH ( DateTable[DATEKEY] )  
)
```

Previous Year Total Cases =

```
CALCULATE ( [Total Cases],  
    PREVIOUSYEAR ( DateTable[DATEKEY] )  
)
```



PREVIOUSMONTH / YEAR

- The PREVIOUSYEAR function assumes that your financial year ends on December 31. If you use a different financial year, you can use the second argument of this function to define your year-end date. To avoid any date locale issues, use the date format "YYYYMM-DD" (the function ignores the year, so use any year value); for example, if your yearend date is the March 31st, this would be your measure:

Year To Date Cases =

```
CALCULATE ( [Total Cases] ,  
PREVIOUSYEAR ( DateTable[DATEKEY], "2021-03-31"  
)  
)
```

S < M O O P O O = □ ▱ L < S
Y O O < O - SAMEPERIODLASTYEAR

Same Period Last Year Cases =

```
CALCULATE ( [Total Cases],  
    SAMEPERIODLASTYEAR ( DateTable[DATEKEY] )  
)
```



6 Months Ago Cases =

```
CALCULATE ( [Total Cases],  
            DATEADD ( DateTable[DATEKEY], -6, MONTH )  
        )
```

30 Days Ago Cases =

```
CALCULATE ( [Total Cases] ,  
            DATEADD ( DateTable[DATEKEY], -30, DAY )  
        )
```

YO   D  - DATESYTD

Year To Date Cases =

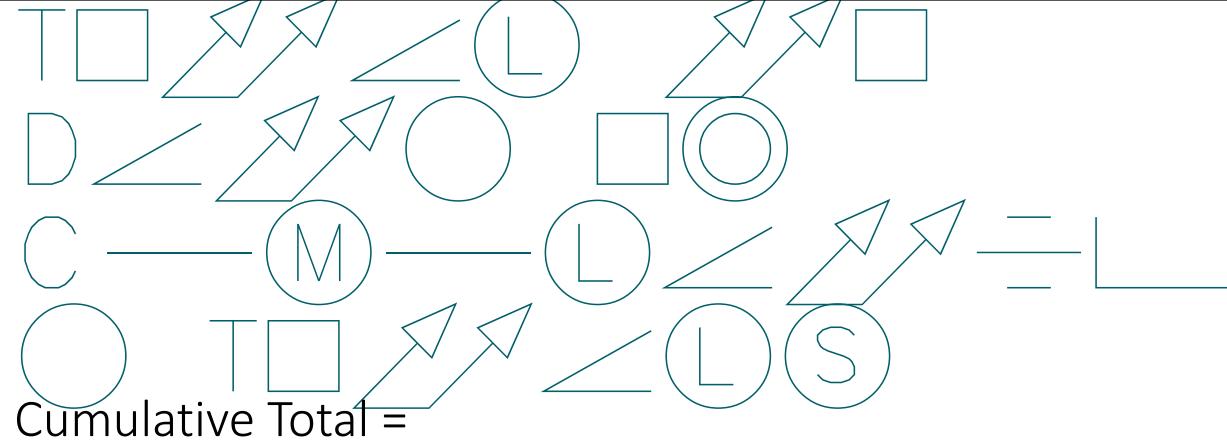
```
CALCULATE ( [Total Cases] ,  
    DATESYTD ( DateTable[DATEKEY] )  
)
```

The DATESYTD function, like PREVIOUSYEAR, assumes that your financial year ends in December, and just like PREVIOUSYEAR, you can use the second argument of this function to define your year-end date, using the format “YYYY-MM-DD” to avoid date locale issues, as follows:

Year To Date Cases =

```
CALCULATE ( [Total Cases] ,  
    DATESYTD ( DateTable[DATEKEY], "2021-03-31")  
)
```

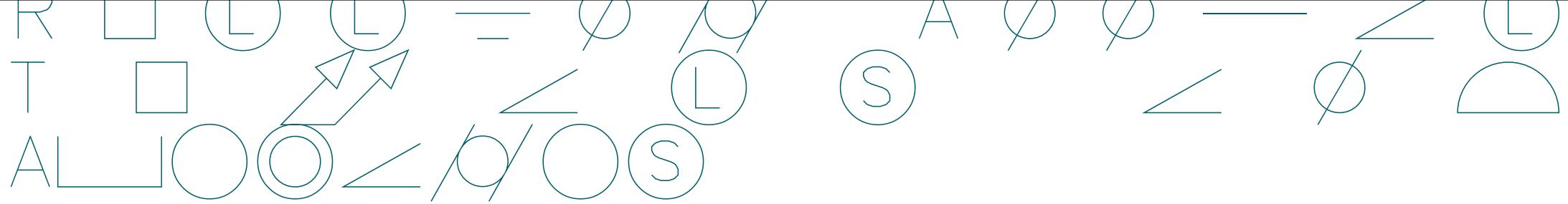
YEAR	WINE	Total	Previous	Same	6 Months	Year To
		Cases	Month Total	Period Last	Ago Cases	Date Cases
<input type="checkbox"/> 2017	Bordeaux	998	951	563	139	18,514
<input type="checkbox"/> 2018	Champagne	251	1,038	1,825	1,089	12,164
<input type="checkbox"/> 2019	Chardonnay	430	218	887	940	12,671
<input type="checkbox"/> 2020	Chenin Blanc	595	586	360	869	8,206
<input checked="" type="checkbox"/> 2021	Chianti	123	619	1,279	1,996	8,837
MONTH	Grenache	916	893	189	1,240	10,293
	Malbec	801	963	914	769	11,082
	Merlot		98	343	573	6,378
	Piesporter		484	440	681	3,736
	Pinot Grigio	338	487	902	752	6,400
	Rioja	833	858	1,416	821	9,193
	Sauvignon Blanc	1,465	1,721	929	335	12,247
	Shiraz	616	98	301	516	4,675
	Total	7,366	9,014	10,348	10,720	124,396
	Dec					



CALCULATE ([Total Sales] ,
 DATESBETWEEN (DateTable[DATEKEY], 0 ,
 LASTDATE (DateTable[DATEKEY])

)
)

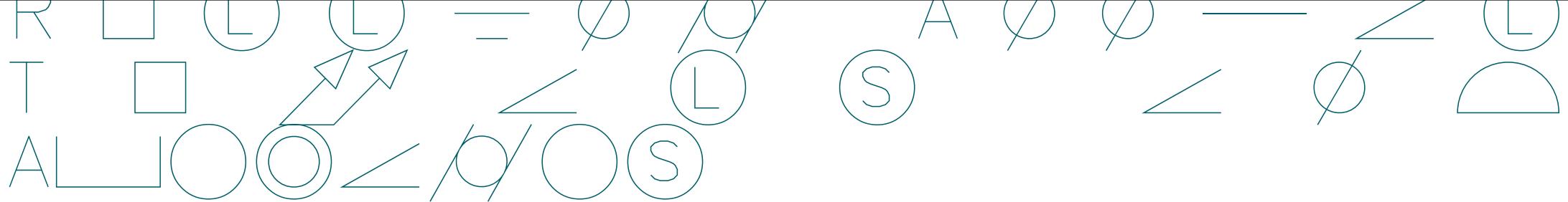
YEAR	Total Sales	Cumulative Total
2017	\$4,649,285	\$4,649,285
Jan	\$451,887	\$451,887
Feb	\$385,299	\$837,186
Mar	\$400,977	\$1,238,163
Apr	\$327,070	\$1,565,233
May	\$353,073	\$1,918,306
Jun	\$241,419	\$2,159,725
Jul	\$410,507	\$2,570,232
Aug	\$194,755	\$2,764,987
Sep	\$559,821	\$3,324,808
Oct	\$438,513	\$3,763,321
Nov	\$301,695	\$4,065,016
Dec	\$584,269	\$4,649,285
2018	\$4,207,871	\$8,857,156
Jan	\$407,812	\$5,057,097
Feb	\$299,495	\$5,356,592
Mar	\$232,473	\$5,589,065
Apr	\$484,275	\$6,073,340
Total	\$29,732,482	\$29,732,482



Rolling Annual Total Sales =

```
CALCULATE ( [Total Sales],  
    DATESINPERIOD ( DateTable[DATEKEY],  
        LASTDATE ( DateTable[DATEKEY] ) , -1 , YEAR ) )
```

The LASTDATE function in this measure finds the last date in the current filter context (i.e., the last date of the month sitting in any row of the Table visual, in a slicer, or in the Filters pane). The DATESINPERIOD function calculates the total sales, starting with this last date and going back by 1 year.



Rolling Annual Average Total Sales =

CALCULATE (

[Total Sales] / COUNTROWS (VALUES (DateTable[MONTH])),

DATESINPERIOD (

DateTable[DATEKEY],

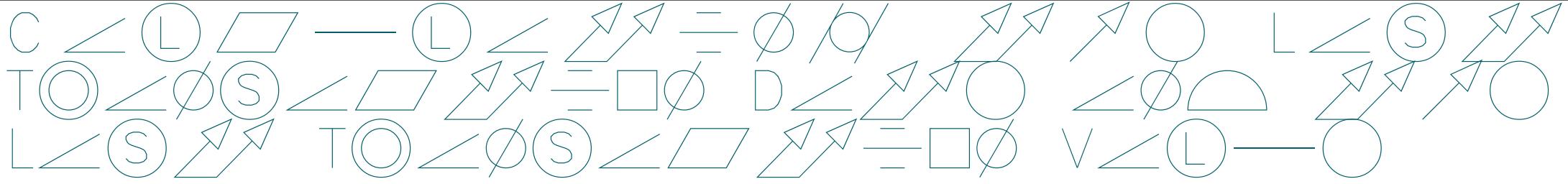
LASTDATE (DateTable[DATEKEY]), -1, YEAR

)

)

The expression for the rolling annual average does much the same as the expression for the rolling annual total. However, we need to find the average monthly total for each rolling year. If we divided the “Total Sales” measure by 12, this would not be correct for the *first year* because in January, only one month is rolling; in February, only two months are rolling; in March, only three months; etc. This is why we need to use the COUNTROWS and VALUES functions to calculate the correct number of rolling months for the denominator and not simply divide by 12.

YEAR	Total Sales	Rolling Annual Total Sales	Rolling Annual Average Total Sales
2017	\$4,649,285	\$4,649,285	\$387,440
Jan	\$451,887	\$451,887	\$451,887
Feb	\$385,299	\$837,186	\$418,593
Mar	\$400,977	\$1,238,163	\$412,721
Apr	\$327,070	\$1,565,233	\$391,308
May	\$353,073	\$1,918,306	\$383,661
Jun	\$241,419	\$2,159,725	\$359,954
Jul	\$410,507	\$2,570,232	\$367,176
Aug	\$194,755	\$2,764,987	\$345,623
Sep	\$559,821	\$3,324,808	\$369,423
Oct	\$438,513	\$3,763,321	\$376,332
Nov	\$301,695	\$4,065,016	\$369,547
Dec	\$584,269	\$4,649,285	\$387,440
2018	\$4,207,871	\$4,207,871	\$350,656
Jan	\$407,812	\$4,605,210	\$383,768
Feb	\$299,495	\$4,519,406	\$376,617
Mar	\$222,172	\$4,250,000	\$362,575
Total	\$29,732,482	\$8,263,718	\$688,643



Date of Last Transaction =

`LASTNONBLANK (DateTable[DATEKEY], [Total Sales])`

Date of First Transaction =

`FIRSTNONBLANK (DateTable[DATEKEY], [Total Sales])`

Value of First Transaction =

`FIRSTNONBLANKVALUE (DateTable[DATEKEY], [Total Sales])`

Value of Last Transaction =

`LASTNONBLANKVALUE (DateTable[DATEKEY], [Total Sales])`

WINE	Date of First Transaction	Date of Last Transaction	Value of First Transaction	Value of Last Transaction
Bordeaux	18/01/2017	23/12/2021	\$24,525	\$21,750
Champagne	22/01/2017	11/12/2021	\$52,050	\$37,650
Chardonnay	07/01/2017	07/12/2021	\$14,700	\$22,500
Chenin Blanc	15/01/2017	22/12/2021	\$7,350	\$5,150
Chianti	09/01/2017	27/12/2021	\$6,920	\$4,920
Grenache	02/01/2017	30/12/2021	\$2,100	\$15,120
Malbec	01/01/2017	14/12/2021	\$27,710	\$22,865
Merlot	14/01/2017	20/11/2021	\$4,680	\$3,822
Piesporter	15/01/2017	19/11/2021	\$9,450	\$9,045
Pinot Grigio	19/01/2017	23/12/2021	\$9,720	\$5,730
Rioja	08/01/2017	26/12/2021	\$6,975	\$7,065
Sauvignon Blanc	01/01/2017	24/12/2021	\$8,520	\$13,240
Shiraz	10/01/2017	30/12/2021	\$8,268	\$15,912
Total	01/01/2017	30/12/2021	\$36,230	\$31,032



Previous Sales Date =

```
CALCULATE (  
    LASTNONBLANK ( DateTable[DATEKEY], [Total Sales] ),  
    DateTable[DATEKEY] < MAX (DateTable[DATEKEY] )  
)
```

Previous Sales Value =

```
CALCULATE (  
    LASTNONBLANKVALUE ( DateTable[DATEKEY], [Total Sales] ),  
    DateTable[DATEKEY] < MAX ( DateTable[DATEKEY] )  
)
```

Sales Difference =

```
[Total Sales] - [Previous Sales Value]
```

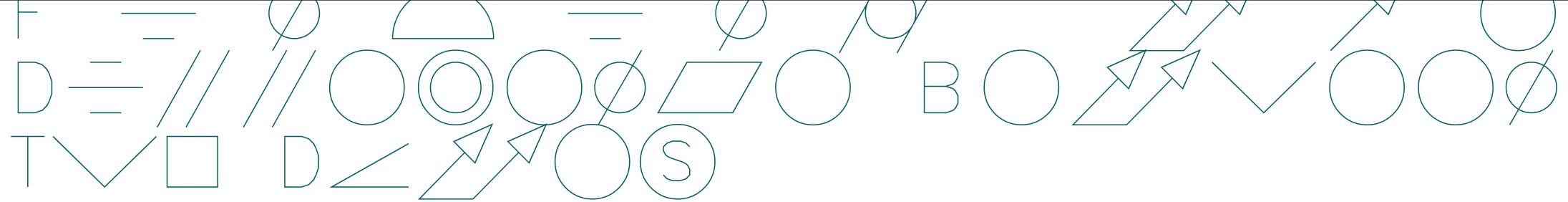
DATEKEY	Previous Sales Date	Previous Sales Value	Total Sales	Sales Difference
1 January, 2017			\$36,230	\$36,230
2 January, 2017	01/01/2017	\$36,230	\$2,100	(\$34,130)
3 January, 2017	02/01/2017	\$2,100	\$10,560	\$8,460
4 January, 2017	03/01/2017	\$10,560		(\$10,560)
5 January, 2017	03/01/2017	\$10,560		(\$10,560)
6 January, 2017	03/01/2017	\$10,560		(\$10,560)
7 January, 2017	03/01/2017	\$10,560	\$14,700	\$4,140
8 January, 2017	07/01/2017	\$14,700	\$6,975	(\$7,725)
9 January, 2017	08/01/2017	\$6,975	\$6,920	(\$55)
10 January, 2017	09/01/2017	\$6,920	\$8,268	\$1,348
11 January, 2017	10/01/2017	\$8,268		(\$8,268)
12 January, 2017	10/01/2017	\$8,268	\$22,152	\$13,884
13 January, 2017	12/01/2017	\$22,152	\$22,800	\$648
14 January, 2017	13/01/2017	\$22,800	\$4,680	(\$18,120)
15 January, 2017	14/01/2017	\$4,680	\$44,949	\$40,269
16 January, 2017	15/01/2017	\$44,949		(\$44,949)
17 January, 2017	15/01/2017	\$44,949	\$7,410	(\$37,539)
Total	30/12/2021	\$31,032	\$29,732,482	\$29,701,450



- Because we are using the DATEKEY column from the date dimension in the Table visual, the expressions using LASTNONBLANK and LASTNONBLANKVALUE will be evaluated for every date in this column, regardless of whether each date has a transaction in the Winesales table. When you then populate the “Total Sales” measure into the Table visual, you will see blank values for dates where there are no transactions. To resolve this, use a visual-level filter and filter the “Total Sales” measure to exclude blank values.
- The important factor in the evaluation of these expressions is the use of CALCULATE to modify the filter context in which the LASTNONBLANK and LASTNONBLANKVALUE are evaluated. The expression “`MAX (DateTable[DATEKEY])`” returns the date value in the current filter context, for example, **7 January 2017**. The MAX function is used to return a scalar value. As there is only a single date in the current filter context, we could equally use MIN or SUM. The filter argument of CALCULATE therefore is saying “find the date in the DATEKEY column of the DateTable that is before the date returned by ‘`MAX (DateTable[DATEKEY])`’ but only if it has a sales value and is not blank.” The LASTNONBLANK function returns this date, that is, **3 January 2017**. The LASTNONBLANKVALUE function returns the sales value associated with this date, **\$10,560**.

DATEKEY	Previous Sales Date	Previous Sales Value	Total Sales	Sales Difference
1 January, 2017			\$36,230	\$36,230
2 January, 2017	01/01/2017	\$36,230	\$2,100	(\$34,130)
3 January, 2017	02/01/2017	\$2,100	\$10,560	\$8,460
4 January, 2017	03/01/2017	\$10,560		(\$10,560)
5 January, 2017	03/01/2017	\$10,560		(\$10,560)
6 January, 2017	03/01/2017	\$10,560		(\$10,560)
7 January, 2017	03/01/2017	\$10,560	\$14,700	\$4,140
8 January, 2017	07/01/2017	\$14,700	\$6,975	(\$7,725)
9 January, 2017	08/01/2017	\$6,975	\$6,920	(\$55)

*Focusing on an evaluation of the LASTNONBLANK and
LASTNONBLANKVALUE expressions*



Days Difference =

`INT ([Date of Last Transaction]) - INT ([Date of First Transaction])`

Months Difference =

`DATEDIFF ([Date of First Transaction], [Date of Last Transaction], MONTH)`

WINE	Date of First Transaction	Date of Last Transaction	Days Difference	Months Difference
Bordeaux	18/01/2017	23/12/2021	1,800	59
Champagne	22/01/2017	11/12/2021	1,784	59
Chardonnay	07/01/2017	07/12/2021	1,795	59
Chenin Blanc	15/01/2017	22/12/2021	1,802	59
Chianti	09/01/2017	27/12/2021	1,813	59
Grenache	02/01/2017	30/12/2021	1,823	59
Malbec	01/01/2017	14/12/2021	1,808	59
Merlot	14/01/2017	20/11/2021	1,771	58
Piesporter	15/01/2017	19/11/2021	1,769	58
Pinot Grigio	19/01/2017	23/12/2021	1,799	59
Rioja	08/01/2017	26/12/2021	1,813	59
Sauvignon Blanc	01/01/2017	24/12/2021	1,818	59
Shiraz	10/01/2017	30/12/2021	1,815	59
Total	01/01/2017	30/12/2021	1,824	59

T ↗ O BLANK() F — Ø □ ↗ ↘ = □ Ø

- In DAX, there is a special way to identify null or empty values, and that's by using a value called "blank." To return blank values, we can use the BLANK() function as shown in a calculated column created in the Winesales table:

10 Percent =

```
IF ( Winesales[CASES SOLD] > 100,
    Winesales[CASES SOLD] * 0.1, BLANK () )
```

10 Percent =
 $\text{IF} (\text{Winesales}[\text{CASES SOLD}] > 100,$
 $\text{Winesales}[\text{CASES SOLD}] * 0.1)$

X ✓

```
1 10 Percent =
2 IF ( Winesales[CASES SOLD] > 100,
3 Winesales[CASES SOLD] * 0.1, BLANK () )
```

Sale Date	Winesales No	Salesperson ID	Customer ID	Wine ID	Cases Sold	10 PC case	Team	Volume level	Customer name from Customer table	Region name	10 Percent
6/21/2021	1870			10	240	24.0	Team A	High	Lavender Day Ltd	Czech Republic	24.0
6/19/2021	1869	6	16	13	98	9.8	Team A	High	Port Hammond Bros	Italy	
6/12/2021	1861	6	23	6	74	7.4	Team A	High	Villeneuve-d'Ascq Ltd	Germany	
6/12/2021	1860	6	23	6	90	9	Team A	High	Villeneuve-d'Ascq Ltd	Germany	
6/11/2021	1856	6	24	6	99	9.9	Team A	High	Charleston Ltd	United States	
6/11/2021	1855	6	24	6	90	9	Team A	High	Charleston Ltd	United States	
6/11/2021	1857	6	24	6	78	7.8	Team A	High	Charleston Ltd	United States	
6/10/2021	1852	6	28	8	140	14	Team A	High	Rhodes Ltd	South Africa	14
6/8/2021	1848	6	13	4	283	28.3	Team A	High	Sedro Woolley Ltd	England	28.3
6/7/2021	1847	6	32	12	107	10.7	Team A	High	Sapporo & Co	France	10.7

T ↗ O BLANK() F —— Ø □ ↗ ↘ = □ Ø

Blank? =

IF (Winesales[CASES SOLD] = BLANK(), "Blank", "Other")

Zero? =

IF (Winesales[CASES SOLD] = 0, "Zero", "Other")

X ✓

1 Zero? =
2 IF (Winesales[CASES SOLD] = 0, "Zero", "Other")

SALE DATE	WINESALES NO	SALESPERSON ID	CUSTOMER ID	WINE ID	CASES SOLD	10 PC case	Team	Volume level	Customer name from Customer table	Region name	10 Percent	Blank?	Zero?
12/30/2021	2219	6	25	5	168	16.8	Team A	High	Chatou & Co	India	16.8	Other	Other
12/30/2021	2219	6	25	5	168	16.8	Team A	High	Chatou & Co	India	16.8	Other	Other
12/30/2021	2219	6	25	5	168	16.8	Team A	High	Chatou & Co	India	16.8	Other	Other
12/27/2021	2216	6	36	7	123	12.3	Team A	High	Castle Rock Ltd	Wales	12.3	Other	Other
12/24/2021	2209	6	80	13	115	11.5	Team A	High	Germantown & Co	Argentina	11.5	Other	Other
12/24/2021	2208	6	24	10	331	33.1	Team A	High	Charleston Ltd	United States	33.1	Other	Other
12/13/2021	2182	6	7	11	150	15	Team A	High	Newcastle upon Tyne & Sons	United States	15	Other	Other
12/4/2021	2212	6	3	11	175	17.5	Team A	High	Cape Canaveral Ltd	India	17.5	Other	Other

T ISBLANK F — =

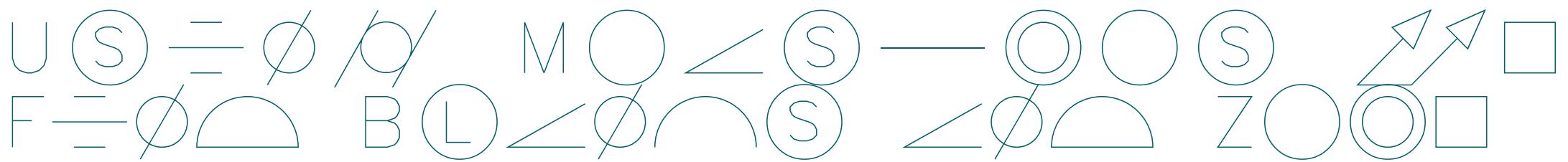
Blank or Zero? =

```
IF (
    ISBLANK ( Winesales[CASES SOLD] ),
    "Blank",
    IF ( Winesales[CASES SOLD] = 0, "Zero", "Other" )
)
```

TOS  =  /  //   Z  

Zero? =

```
IF (  
    NOT ( ISBLANK ( Winesales[CASES SOLD] ) )  
    && Winesales[CASES SOLD] = 0,  
    "Zero",  
    "Other"  
)
```



No. of Customers with No Sales =

COUNTROWS (FILTER (Customers, ISBLANK ([Total Sales])))

No. of Customers with Zero or No Sales =

COUNTROWS (FILTER (Customers, [Total Sales] = 0))

No. of Customers with Zero sales =

COUNTROWS (

FILTER (Customers, NOT (ISBLANK ([Total Sales])))

&& [Total Sales] = 0)

)

CUSTOMER NAME

Total Sales

Back River & Co	\$0
Ballard & Sons	\$138,545
Barstow Ltd	\$78,576
Beaverton & Co	\$99,495
Black Ltd	\$231,205
Bluffton Bros	\$198,624
Branch Ltd	\$127,286
Brooklyn & Co	\$85,742
Brooklyn Ltd	\$27,018
Brown & Co	\$25,542
Burlington Ltd	\$41,552
Burningsuit Ltd	\$893,230
Busan & Co	\$117,105
Canoga Park Ltd	\$37,310
Cape Canaveral Ltd	\$530,254
Castle Rock Ltd	\$815,730
Chandler & Sons	\$725,413
Charleston Ltd	\$667,446
Charlottesville & Co	\$209,340
Total	\$29,732,482

5

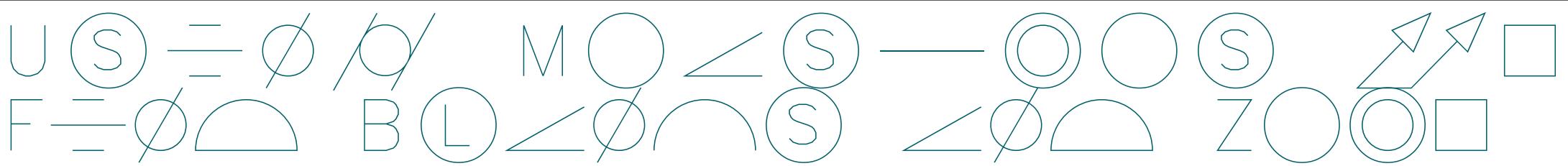
No. of Customers with No Sales

6

No. of Customers with Zero or No Sales

1

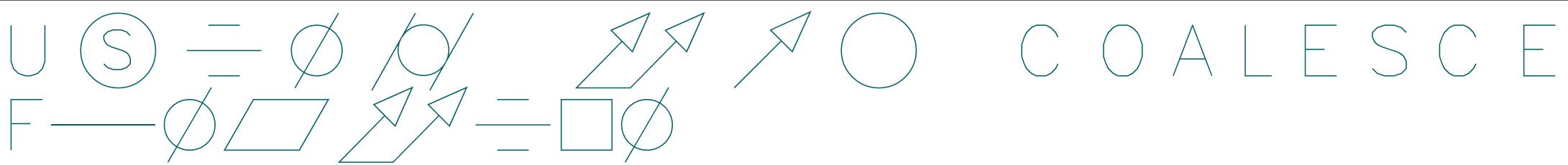
No. of Customers with Zero sales



- We can conclude, therefore, that we must be careful using the following expression:

= IF ([expression] = 0)

because it will include blank values as well as zero values.



If Blank Return Zero =

If (ISBLANK ([Total Sales]), 0, [Total Sales])

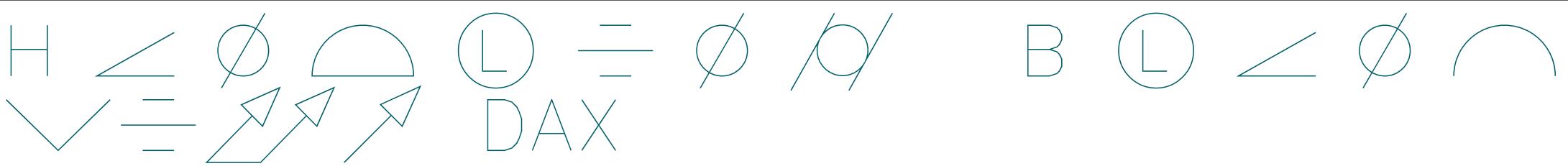
If Blank Return Zero =

COALESCE([Total Sales],0)

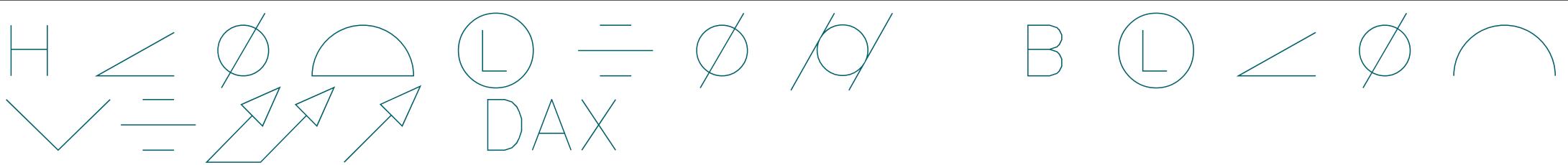
If Blank Return No Sales =

COALESCE([Total Sales],"No Sales")

Customer Name	Total Sales	If Blank Return	If Blank Return
		Zero	No Sales
Acme & Sons		\$0	No Sales
Bloxon Bros.		\$0	No Sales
Jones Ltd		\$0	No Sales
Sainsbury's		\$0	No Sales
Smith & Co		\$0	No Sales
Back River & Co	\$0	\$0	\$0
Palo Alto Ltd	\$14,836	\$14,836	\$14,836
St. Leonards Ltd	\$16,965	\$16,965	\$16,965
Victoria Ltd	\$24,710	\$24,710	\$24,710
Brown & Co	\$25,542	\$25,542	\$25,542
Brooklyn Ltd	\$27,018	\$27,018	\$27,018
Canoga Park Ltd	\$37,310	\$37,310	\$37,310
Loveland & Co	\$38,098	\$38,098	\$38,098
Burlington Ltd	\$41,552	\$41,552	\$41,552
Kennebunkport & Co	\$48,150	\$48,150	\$48,150
Liverpool & Sons	\$49,843	\$49,843	\$49,843
Colombes & Co	\$53,513	\$53,513	\$53,513
Waterbury Ltd	\$55,377	\$55,377	\$55,377
Total	\$29,732,482	\$29,732,482	\$29,732,482

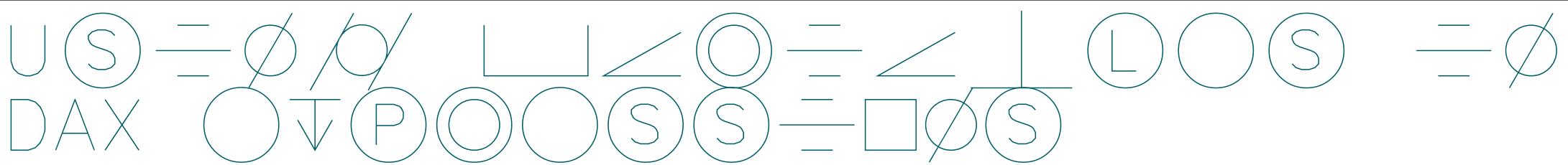


- Checks a condition, and returns one value when it's TRUE, otherwise it returns a second value.
- `IF(<logical_test>, <value_if_true>[, <value_if_false>])`
- Return value: Either **value_if_true**, **value_if_false**, or BLANK.



- Returns the first expression that does not evaluate to BLANK. If all expressions evaluate to BLANK, BLANK is returned.
- COALESCE(<expression>, <expression>[, <expression>]...)
- Return value: A scalar value coming from one of the expressions or BLANK if all expressions evaluate to BLANK.
- Examples:
 - EVALUATE { COALESCE(BLANK(), 10, DATE(2008, 3, 3)) }
 - = COALESCE(SUM(FactInternetSales[SalesAmount]), 0)

E ∇ \angle M P L O



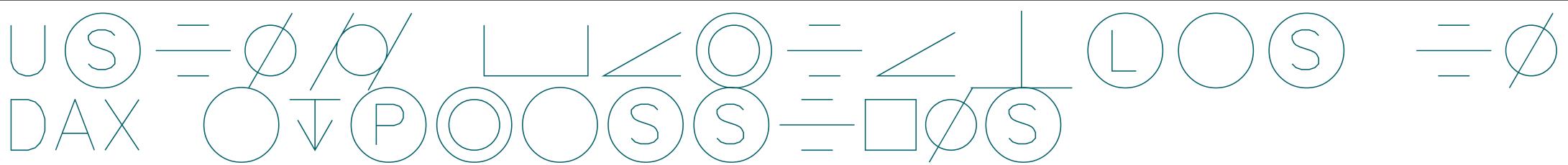
- There are three major advantages gained by using variables:
 1. Improved performance
 2. Improved readability
 3. Reduced complexity
- To include variables in your code, use the keyword **VAR** followed by the name of the variable and then the definition of the variable. The keyword **RETURN** is then used at the end of the code to return the expression to be evaluated:

Example Measure =

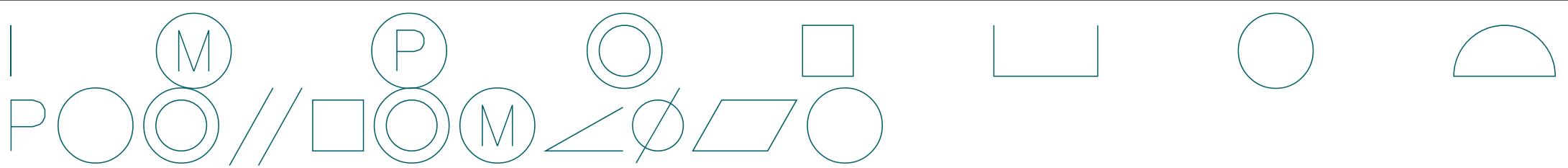
```
VAR MyVariable = SUM (Winesales[CASES SOLD])
```

```
RETURN
```

```
MyVariable * 1.1
```

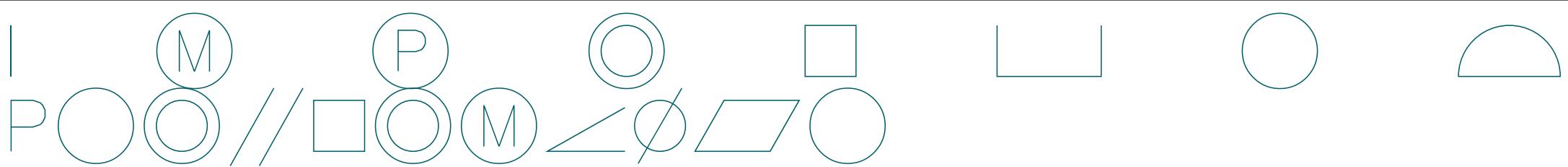


- Variables can be used in both measures and calculated columns to harvest the values generated by:
 - Expressions, for example, SUM (Winesales[CASES SOLD])
 - Measures, for example, [Total Cases]
 - Tables, for example, FILTER (Winesales, Winesales[CASES SOLD] >300)
 - Values, for example, 0.1, 10, 20



10 PC or 5 PC =

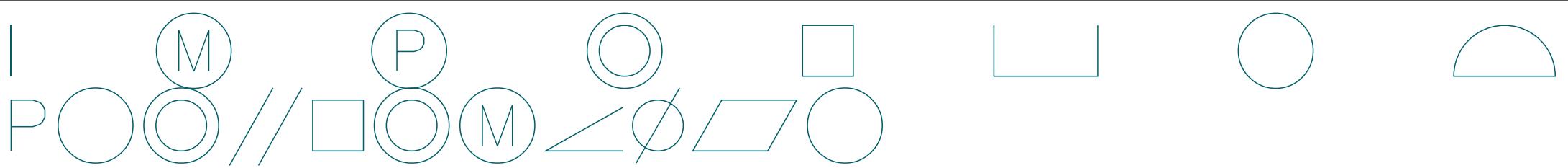
```
IF (
    SUM ( Winesales[CASES SOLD] ) > 20000,
    SUM ( Winesales[CASES SOLD] ) * 0.1,
    IF (
        SUM ( Winesales[CASES SOLD] ) > 15000,
        SUM ( Winesales[CASES SOLD] ) * 0.5,
        SUM ( Winesales[CASES SOLD] )
    )
)
```



- The problem with this expression, especially as far as performance goes, is that there are five repetitions of the SUM function, forcing the evaluation of these expressions five times. Also, the use of the nested IF is rather cumbersome. Using the SWITCH function in place of the nested IF is a small improvement:

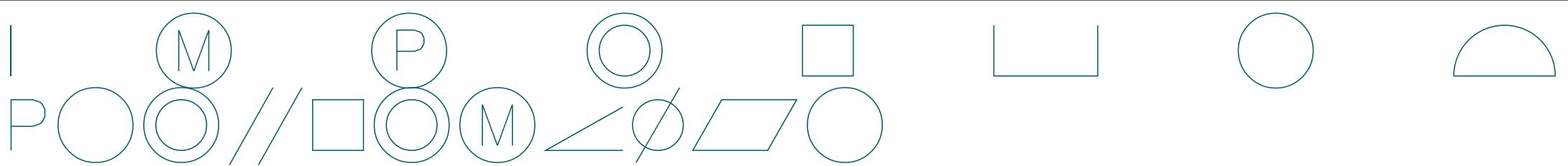
10 PC or 5 PC #2 =

```
SWITCH (
    TRUE (),
    SUM ( Winesales[CASES SOLD] ) > 20000,
        SUM ( Winesales[CASES SOLD] ) * 0.1,
    SUM ( Winesales[CASES SOLD] ) > 15000,
        SUM ( Winesales[CASES SOLD] ) * 0.5,
    SUM ( Winesales[CASES SOLD] )
)
```



=SWITCH (expression, value1, result1, value2, result2 etc...else)

- Notice that inside SWITCH, the function TRUE() is used as the expression to be evaluated and then Boolean statements are listed, followed by the value to be returned if the statements are true. The final argument is the “else” expression.



10 PC or 5 PC #3 =

VAR TotalCasesValue =

SUM (Winesales[CASES SOLD])

RETURN

SWITCH (

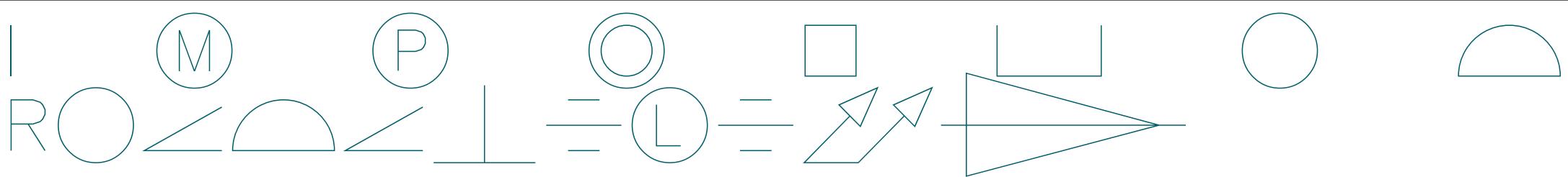
TRUE (),

TotalCasesValue > 20000, TotalCasesValue * 0.1,

TotalCasesValue > 15000, TotalCasesValue * 0.5,

TotalCasesValue

)



- Variables can also help to clarify expressions that use nested measures or nested expressions where the readability of the expressions gets more convoluted. For example, consider the following expression that calculates growth percentage. Notice that the first variable defines a *measure* and the second variable defines an *expression*. The use of the variables and the RETURN statement result in the expression much simpler to understand:

Growth % =

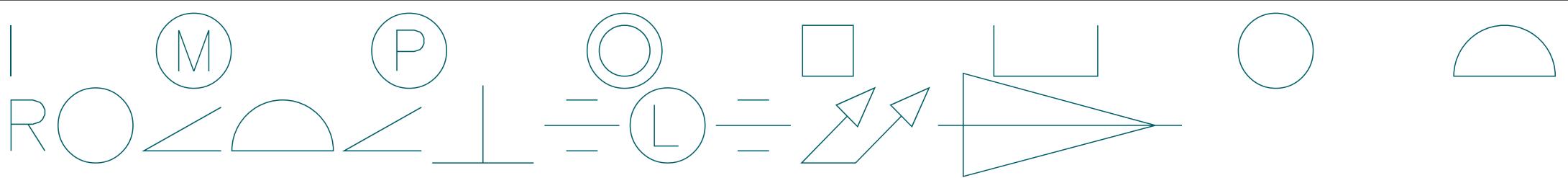
```
VAR CurrentCases = [Total Cases]
```

```
VAR LastYrCases =
```

```
CALCULATE ( [Total Cases], PREVIOUSYEAR ( DateTable[DateKey] ) )
```

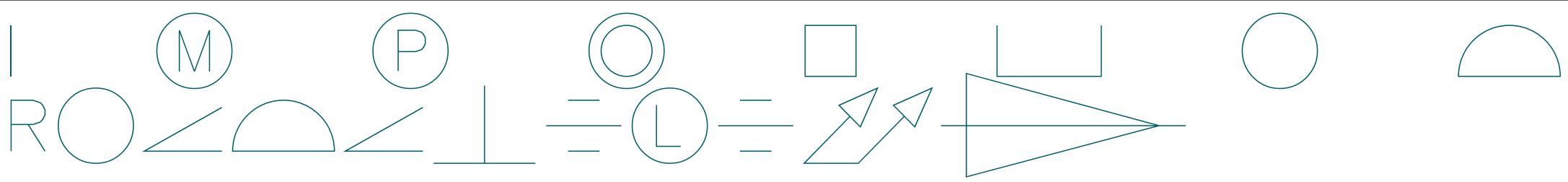
```
RETURN
```

```
DIVIDE ( CurrentCases - LastYrCases, LastYrCases )
```



High-profit Wines =

```
CALCULATE ( [No. of Sales],  
FILTER ( Wines, Wines[PRICE PER CASE] >=  
Wines[COST PRICE] * 3 ))
```



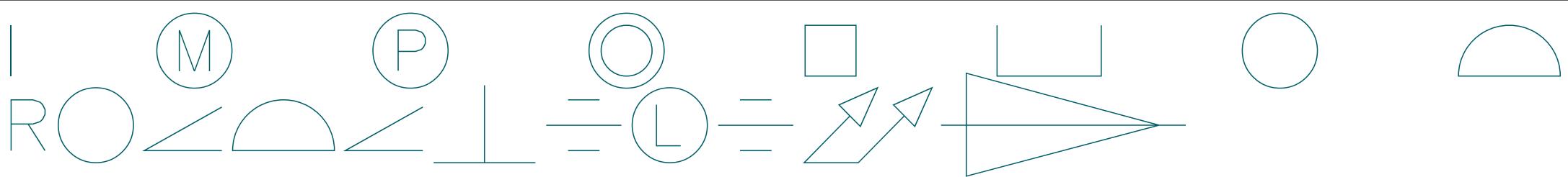
High-profit Wines #1 =

VAR TableOfWines =

```
FILTER ( Wines, Wines[PRICE PER CASE] >=  
        Wines[COST PRICE] * 3 )
```

RETURN

```
CALCULATE ( [No. of Sales], TableOfWines )
```



We can use variables in calculated columns too, for instance, within the arguments of IF:

Cases Sold Increase =

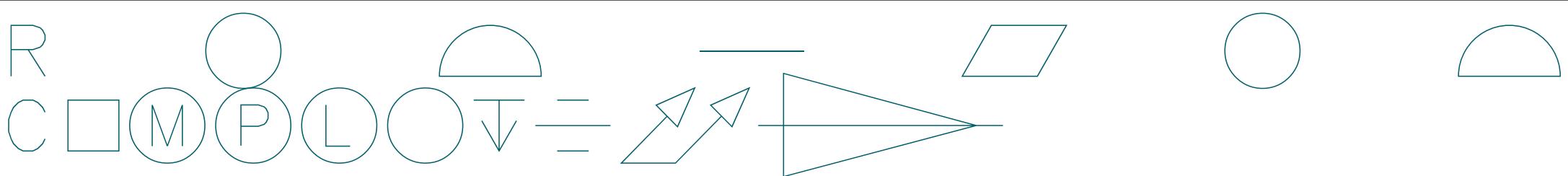
```
VAR CasesSold = Winesales[CASES SOLD]
```

```
VAR MyValue1 = 1.1
```

```
VAR MyValue2 = 1.2
```

```
RETURN
```

```
IF(CasesSold > 100, CasesSold * MyValue1, CasesSold * MyValue2)
```



- We calculated the number of sales where the value in the CASES SOLD column was above the average cases for all wines. This was the measure:

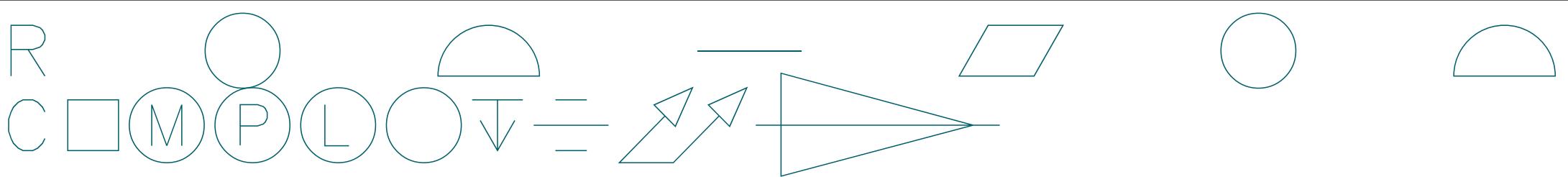
No. of Sales Where Cases is GT Avg All Wines =

```
CALCULATE([No. of Sales],
```

```
    FILTER (Winesales,
```

```
        Winesales[CASES SOLD] >= [Avg Cases All Winesales] ) )
```

- The problem with this code is that because it nests the measure “Avg Cases All Winesales” within the expression, this measure must already exist in our model, as would any measures we use in this context. We may be required to continually locate such measures in the Fields list in order to edit or debug them, leading to frustration and annoyance.



No. of Sales Where Cases is GT Avg All Wines #2 =

VAR AvgAllWines =

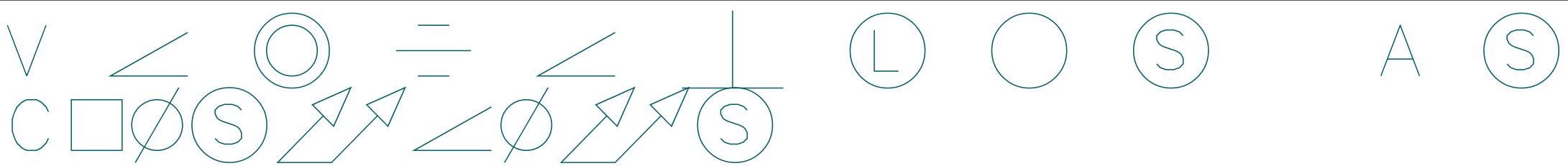
```
CALCULATE( AVERAGE ( Winesales[CASES SOLD] ) ,ALL ( Winesales ) )
```

VAR FilterAvgAll =

```
FILTER ( Winesales, Winesales[CASES SOLD] >= AvgAllWines )
```

RETURN

```
CALCULATE ( [No. of Sales], FilterAvgAll )
```



- There is one last important point to make regarding variables, and that is the term “variable” can be misleading. Perhaps if we called DAX variables “constants,” this might be a more accurate description because that’s what they really are. Consider the following expression:

Sales for Abel =

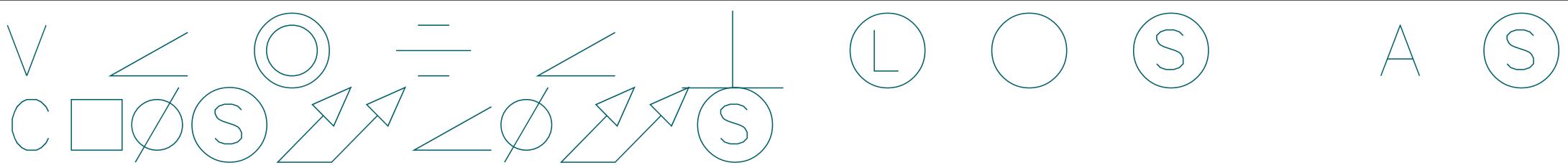
```
VAR MyAmount = [Total Sales]
```

```
RETURN
```

```
CALCULATE ( MyAmount, SalesPeople[SALESPERSON] = "abel" )
```

WINE	Total Sales	Sales for Abel
Bordeaux	\$4,055,250	\$4,055,250
Champagne	\$7,373,700	\$7,373,700
Chardonnay	\$4,203,000	\$4,203,000
Chenin Blanc	\$1,236,950	\$1,236,950
Chianti	\$1,092,920	\$1,092,920
Grenache	\$1,078,950	\$1,078,950
Malbec	\$2,914,650	\$2,914,650
Merlot	\$900,276	\$900,276
Piesporter	\$1,384,155	\$1,384,155
Pinot Grigio	\$703,470	\$703,470
Rioja	\$1,527,795	\$1,527,795
Sauvignon Blanc	\$1,896,600	\$1,896,600
Shiraz	\$1,364,766	\$1,364,766
Total	\$29,732,482	\$29,732,482

Variables behave as constants and can't be modified by CALCULATE



- The reason for this is that the variable “MyAmount” is calculated where it is declared, in this case, before any other code. It then *does not* and *cannot* change by using CALCULATE to modify the filter. This is where we must use a measure such as “Total Sales” inside CALCULATE instead.
- However, the immutable nature of variables is also their strength. For instance, consider the scenario where you want to identify the months where you’ve had exceptionally high sales. You’ve identified exceptionally high sales as those transactions where the sales value is greater than 5% of the total sales for that month.

No of Sales GT 5% Wrong =

CALCULATE (

[No. of Sales],

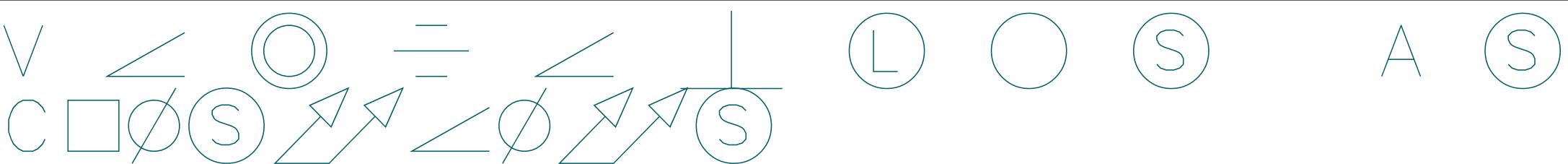
FILTER (

Winesales,

[Total Sales] > [Total Sales] * 0.05

)

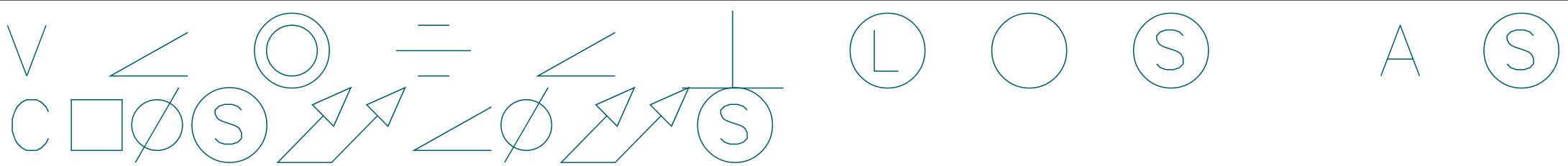
•)



- However, this measure does not return the correct result. The value of the “Total Sales” measure when used inside an iterator such as the FILTER function calculates the total sales for *each* row in the Winesales table, not the total sales for each month. Therefore, the measure “Total Sales GT 5% Wrong” calculates the number of sales where the sales value is greater than 5% of the sales value on each row (i.e., each transaction) and so returns the number of sales.

The “No of Sales GT 5% Wrong” measure returns the number of sales

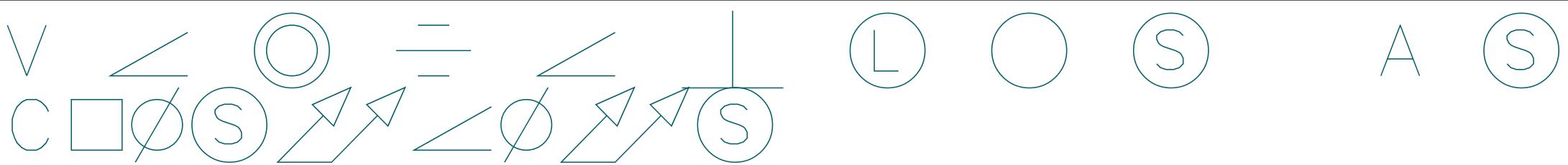
YEAR	Total Sales	No of Sales GT 5% Wrong	No. of Sales
2017	\$4,649,285	320	320
Jan	\$451,887	32	32
Feb	\$385,299	29	29
Mar	\$400,977	27	27
Apr	\$327,070	28	28
May	\$353,073	27	27
Jun	\$241,419	14	14
Jul	\$410,507	23	23
Aug	\$194,755	16	16
Sep	\$559,821	36	36
Oct	\$438,513	25	25
Nov	\$301,695	28	28
Dec	\$584,269	35	35
2018	\$4,207,871	312	313
Jan	\$407,812	30	30
Feb	\$299,495	21	21
Mar	\$232,473	20	21
Apr	\$484,275	32	32
Total	\$29,732,482	2197	2207



- The correct expression must calculate the total sales in the current filter context, which is the total sales for each month, that has been lost by the iteration of FILTER. To reapply this filter, CALCULATE can use the filter that is placed on the Winesales table, the code for which would be a challenge even to experienced DAX users:

No of Sales GT 5% Difficult =

```
CALCULATE (
    [No. of Sales],
    FILTER (
        Winesales,
        [Total Sales] > CALCULATE ( [Total Sales], Winesales ) * 0.05
    )
)
```



- This measure has been labelled as the “difficult” expression because it uses two challenging DAX concepts that we’ve yet to meet: context transition and table expansion. However, you may be relieved to know that you don’t need this advanced knowledge to arrive at the correct calculation. You can use variables instead, and this will render the expression very easy:

No of Sales GT 5% Easy =

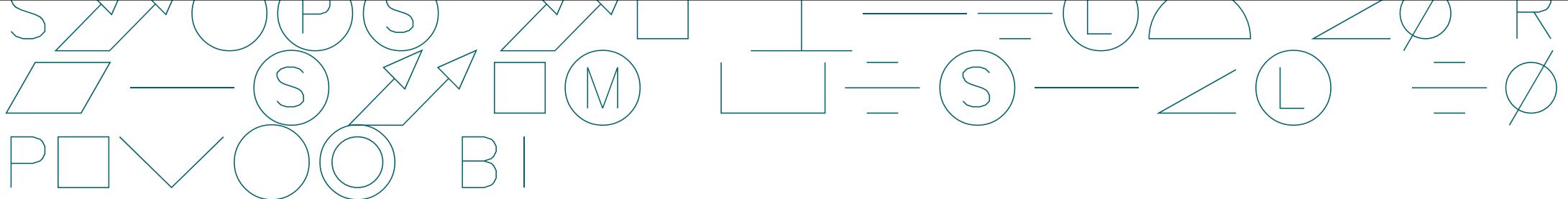
```
VAR PerCentToFind = [Total Sales] * 0.05
```

```
RETURN
```

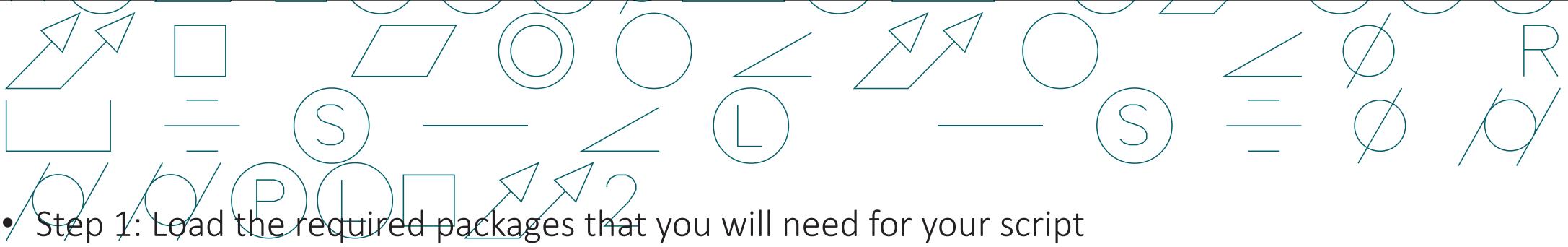
```
CALCULATE ( [No. of Sales],  
FILTER ( Winesales, [Total Sales] > PerCentToFind ) )
```

S Y — T ≡ Y — K
DAX

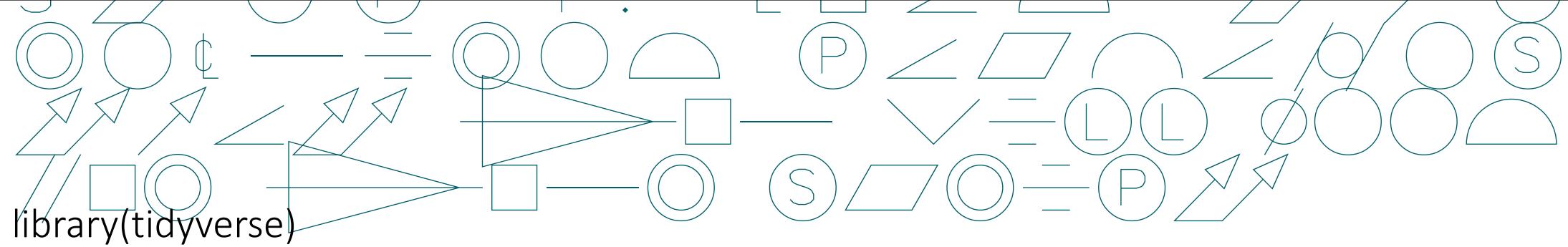
<https://community.powerbi.com/t5/Data-Stories-Gallery/So-You-Think-You-Know-DAX/td-p/325132>



- Step 1: Configure Power BI
- Step 2: Drag the “R custom visual” icon to the Power BI canvas
- Step 3: Define the data set
- Step 4: Develop the visual in your default R IDE
- Step 5: Use the template to develop your visual
- Step 6: Make the script functional

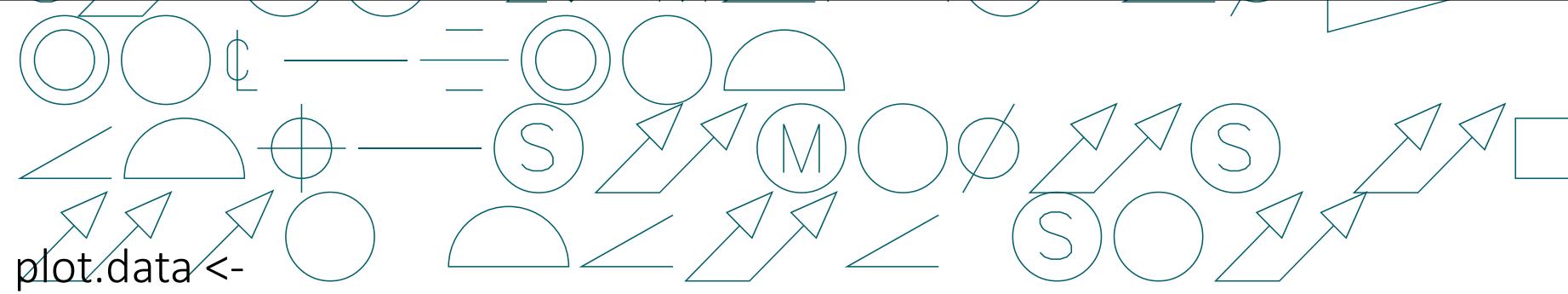


- Step 1: Load the required packages that you will need for your script
- Step 2: Make any required adjustments to the data set
- Step 3: Initiate the creation of the visualization with the ggplot() function
- Step 4: Add desired geom(s)
- Step 5: Define your titles, subtitles, and caption
- Step 6: Make any necessary changes to the x and y axis
- Step 7: Apply themes if needed
- Step 8: Use the theme() function to change any specific non-data elements
- Bonus step: Specifying specific colors for your points in your scatter plot



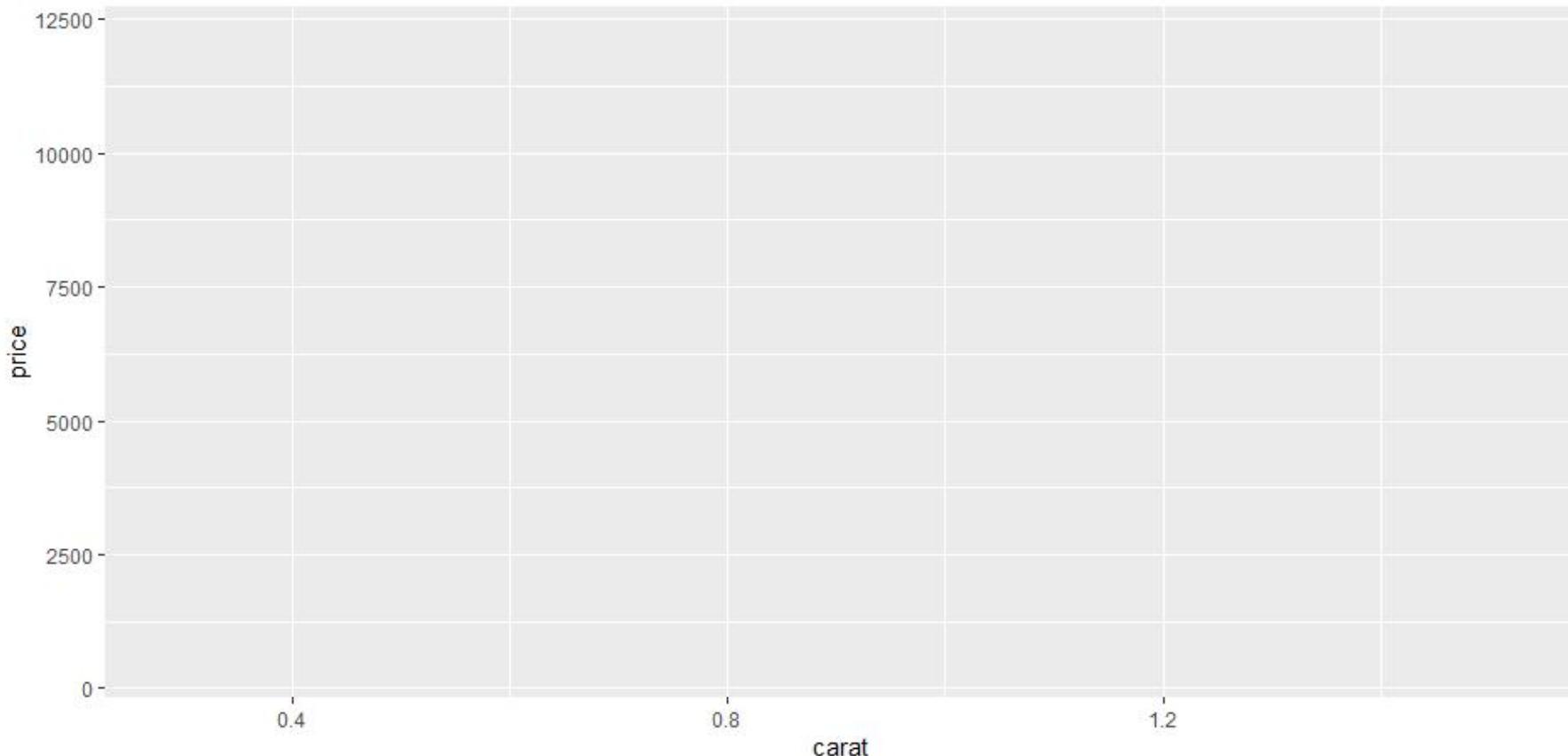
library(tidyverse)

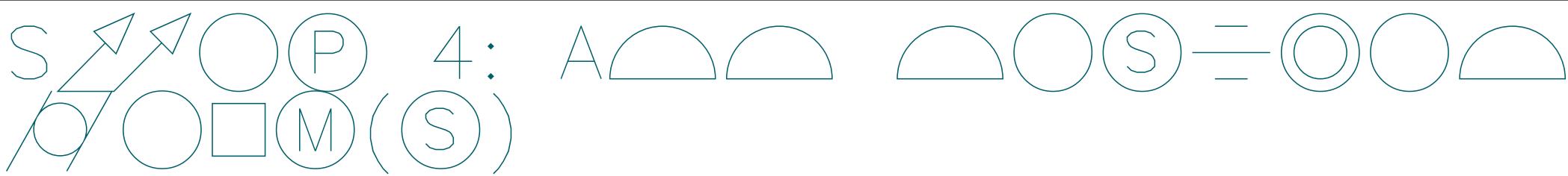
library(scales)



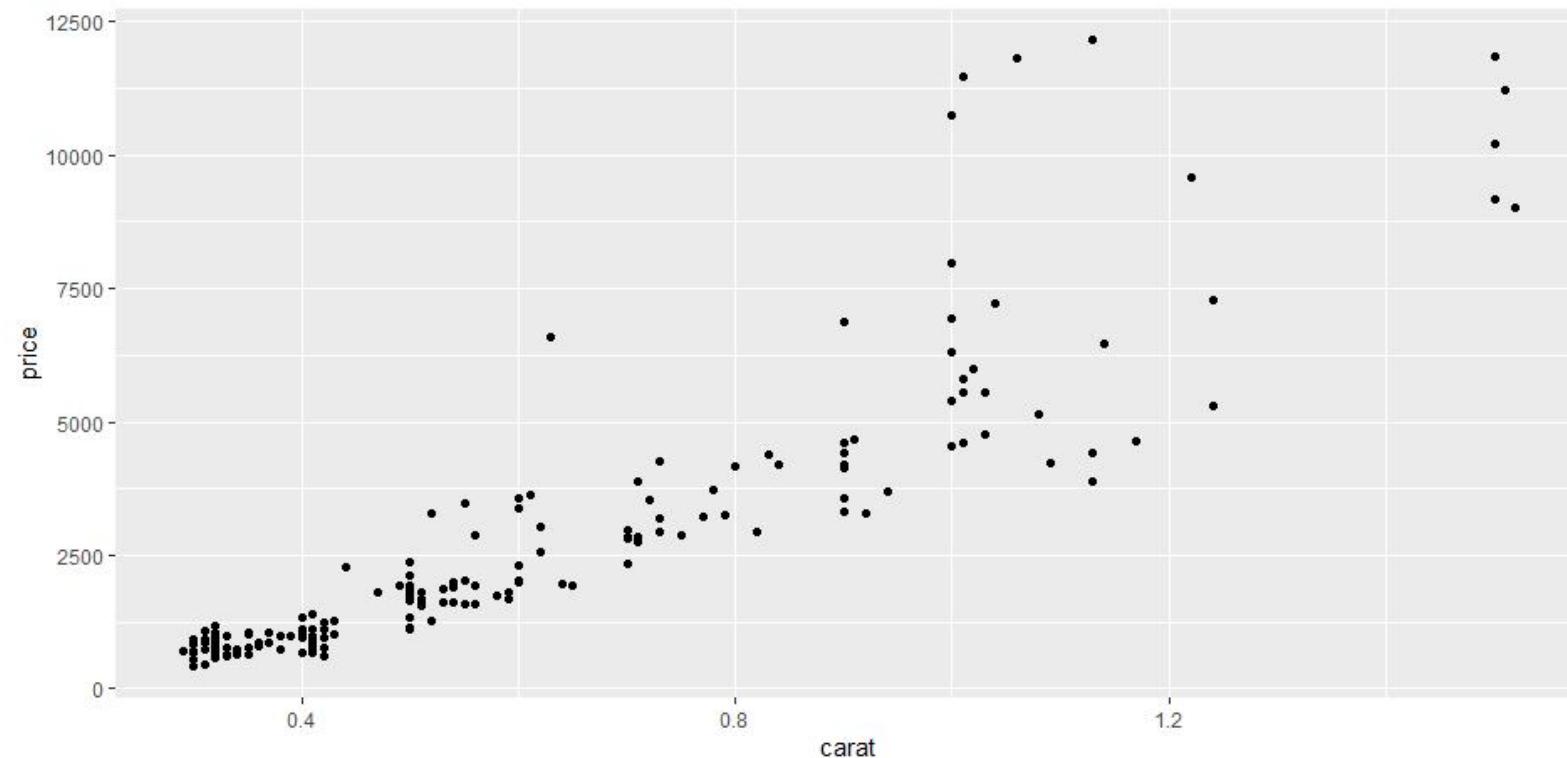
```
plot.data <-  
  diamonds %>%  
  filter(color == "D") %>%  
  sample_n(200)
```

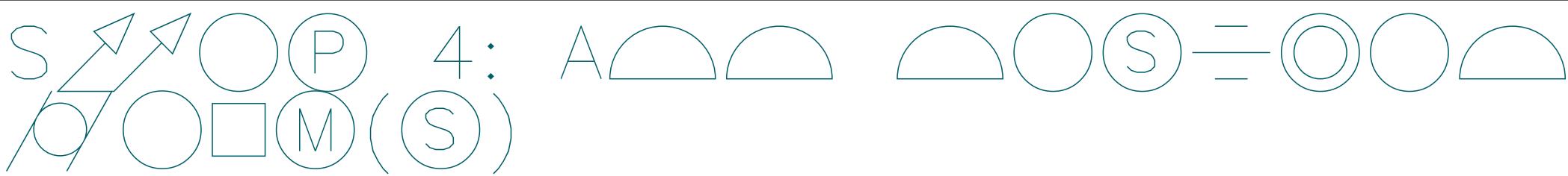
ggplot(data = plot.data, aes(x = carat, y = price))



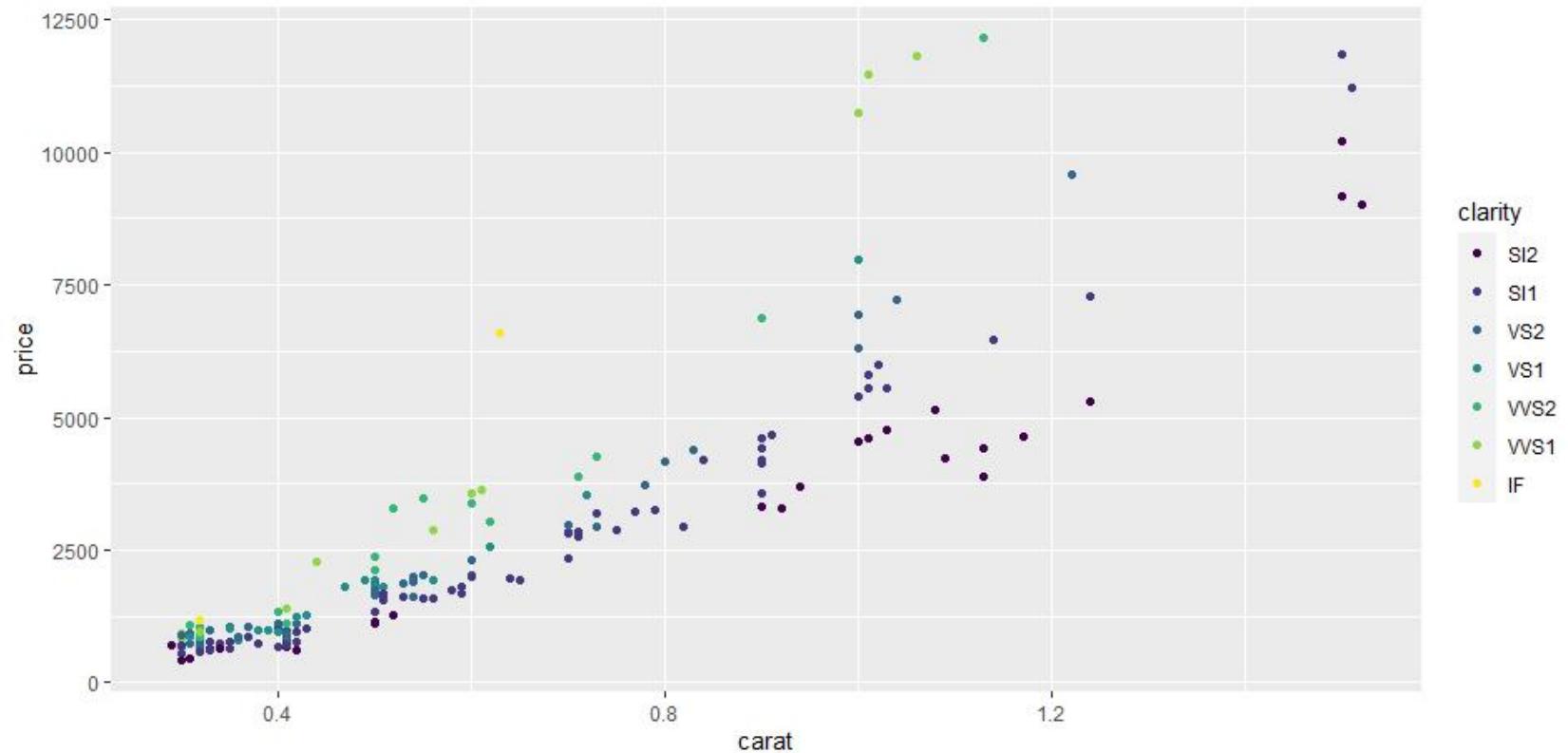


```
ggplot(data = plot.data, aes(x = carat, y = price)) +  
  geom_point()
```

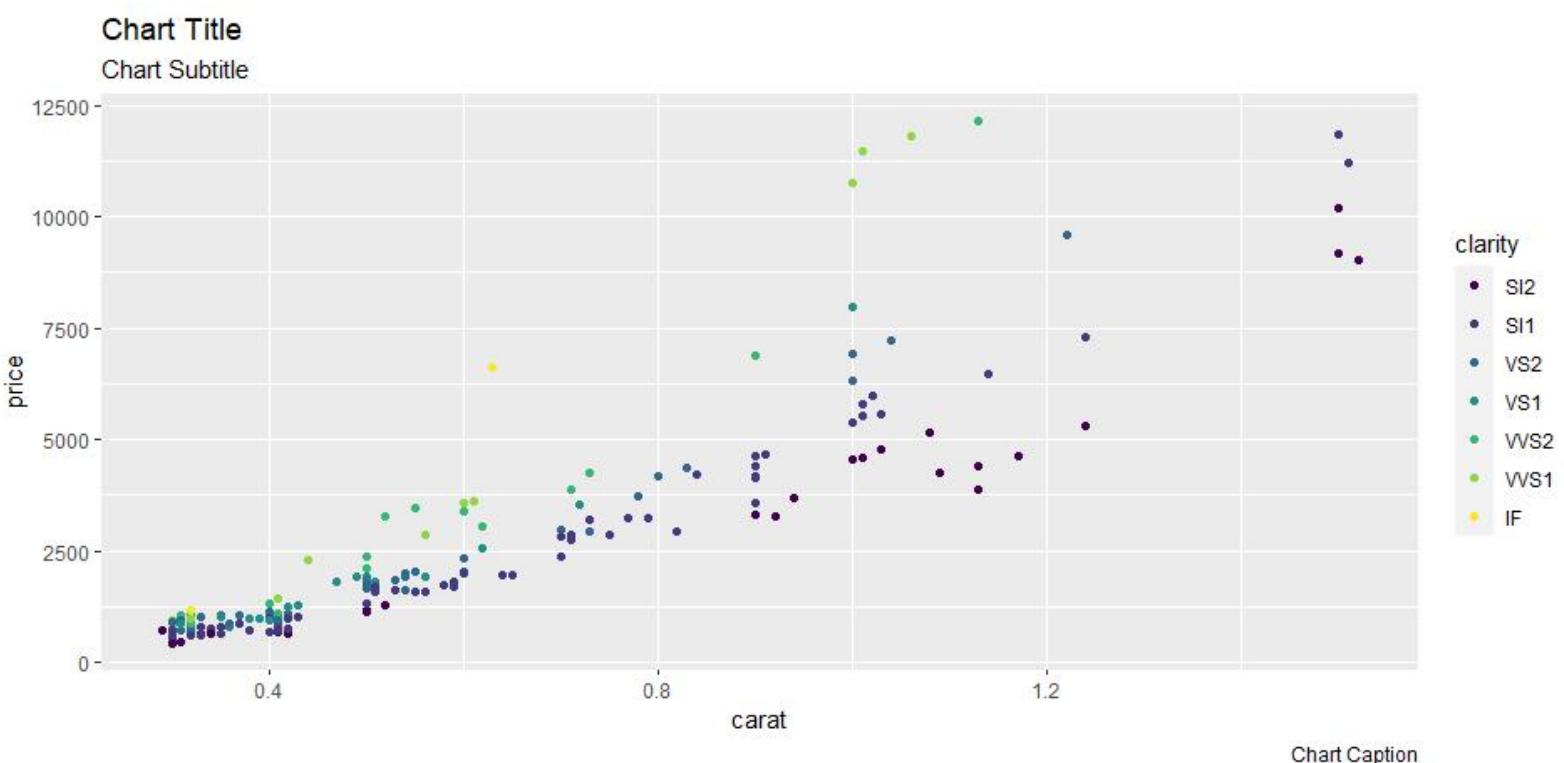


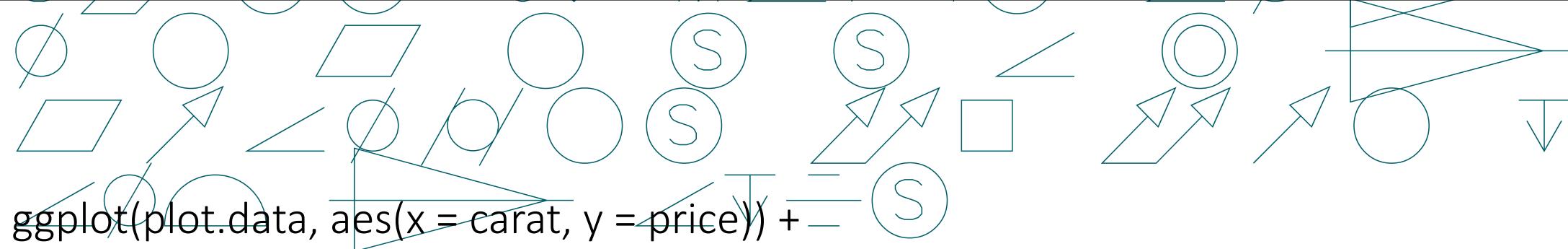


```
ggplot(data = plot.data, aes(x = carat, y = price)) +  
  geom_point(aes(color = clarity))
```

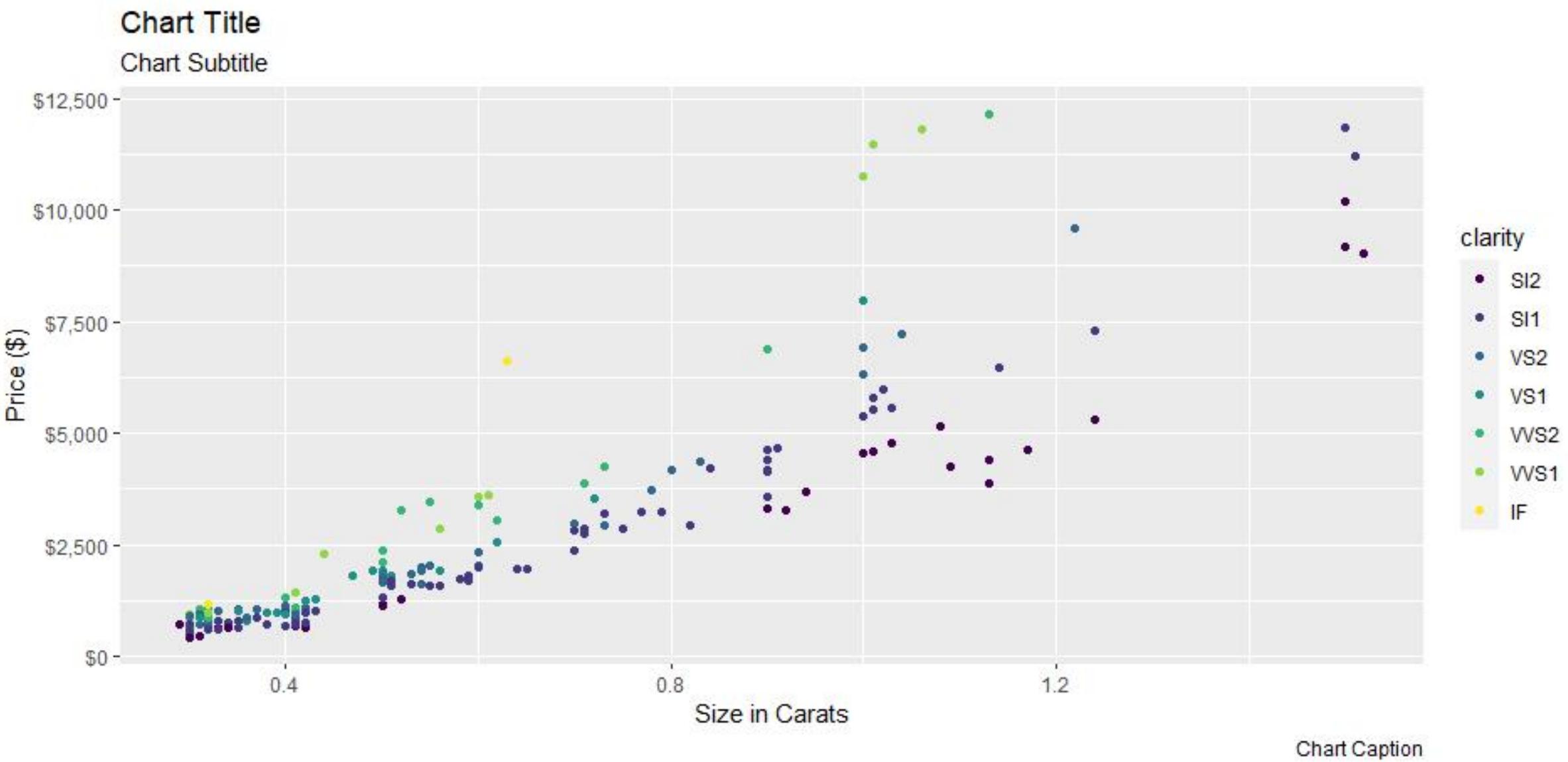


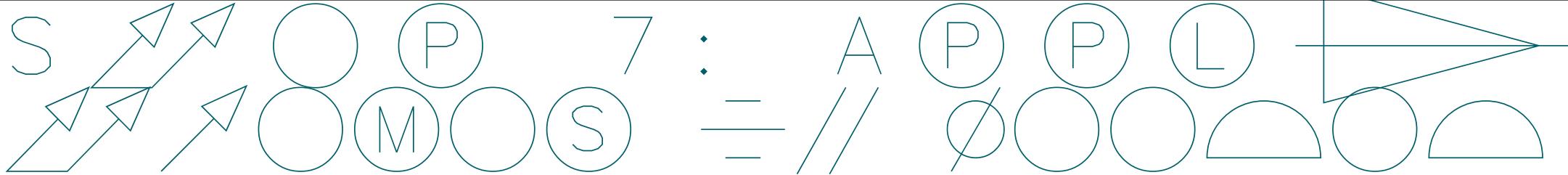
```
ggplot(plot.data, aes(x = carat, y = price)) +  
  geom_point(aes(color = clarity)) +  
  labs(  
    title = "Chart Title",  
    subtitle = "Chart Subtitle",  
    caption = "Chart Caption"  
)
```





```
ggplot(plot.data, aes(x = carat, y = price)) +  
  geom_point(aes(color = clarity)) +  
  labs(  
    title = "Chart Title",  
    subtitle = "Chart Subtitle",  
    caption = "Chart Caption"  
) +  
  scale_x_continuous(name = "Size in Carats") +  
  scale_y_continuous(  
    name = "Price ($)",  
    labels = dollar_format())  
)
```

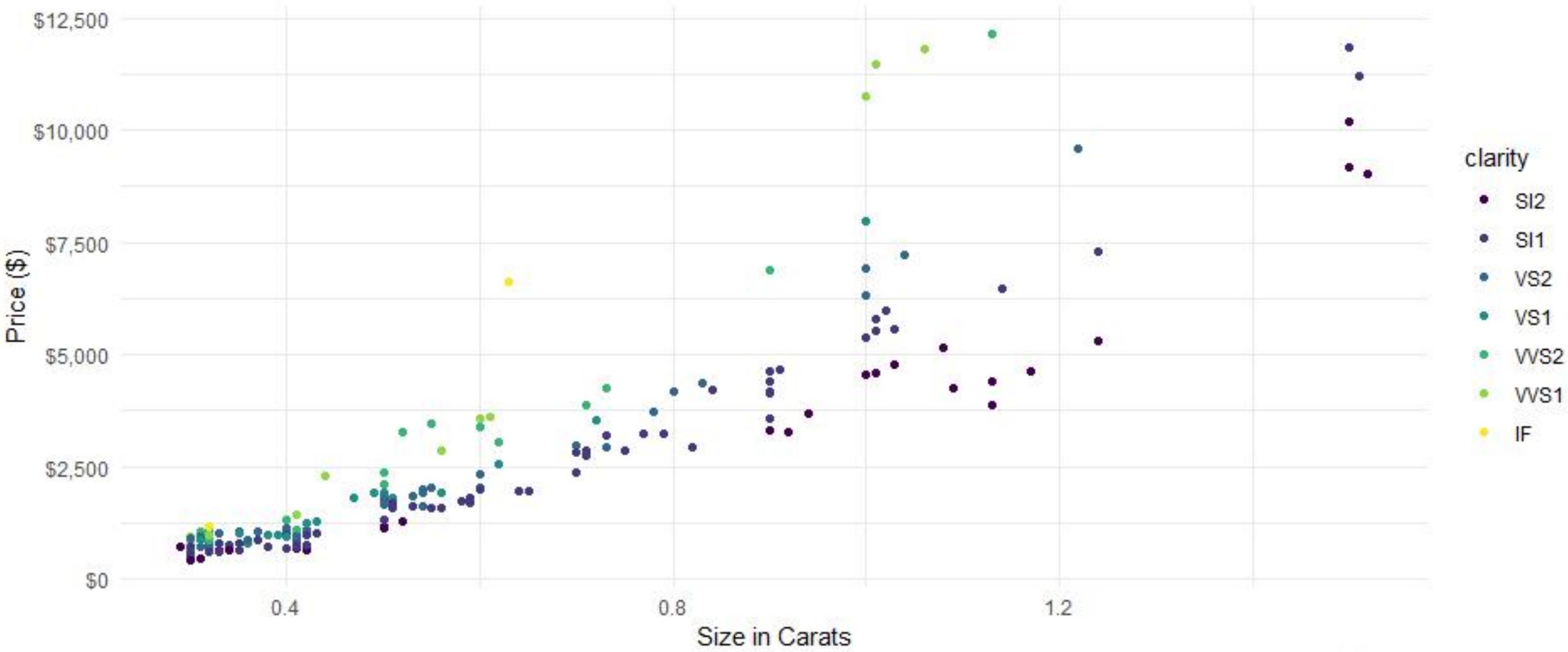


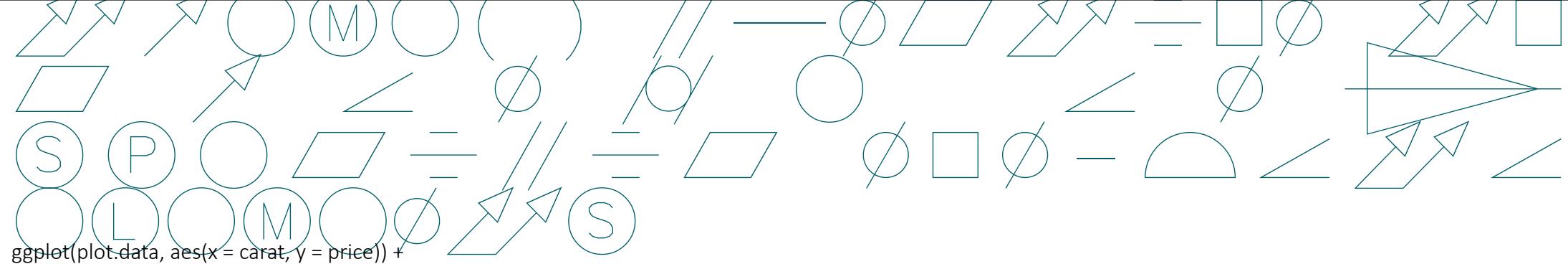


```
ggplot(plot.data, aes(x = carat, y = price)) +  
  geom_point(aes(color = clarity)) +  
  labs(  
    title = "Chart Title",  
    subtitle = "Chart Subtitle",  
    caption = "Chart Caption"  
) +  
  scale_x_continuous(name = "Size in Carats") +  
  scale_y_continuous(  
    name = "Price ($)",  
    labels = dollar_format())  
) +  
  theme_minimal()
```

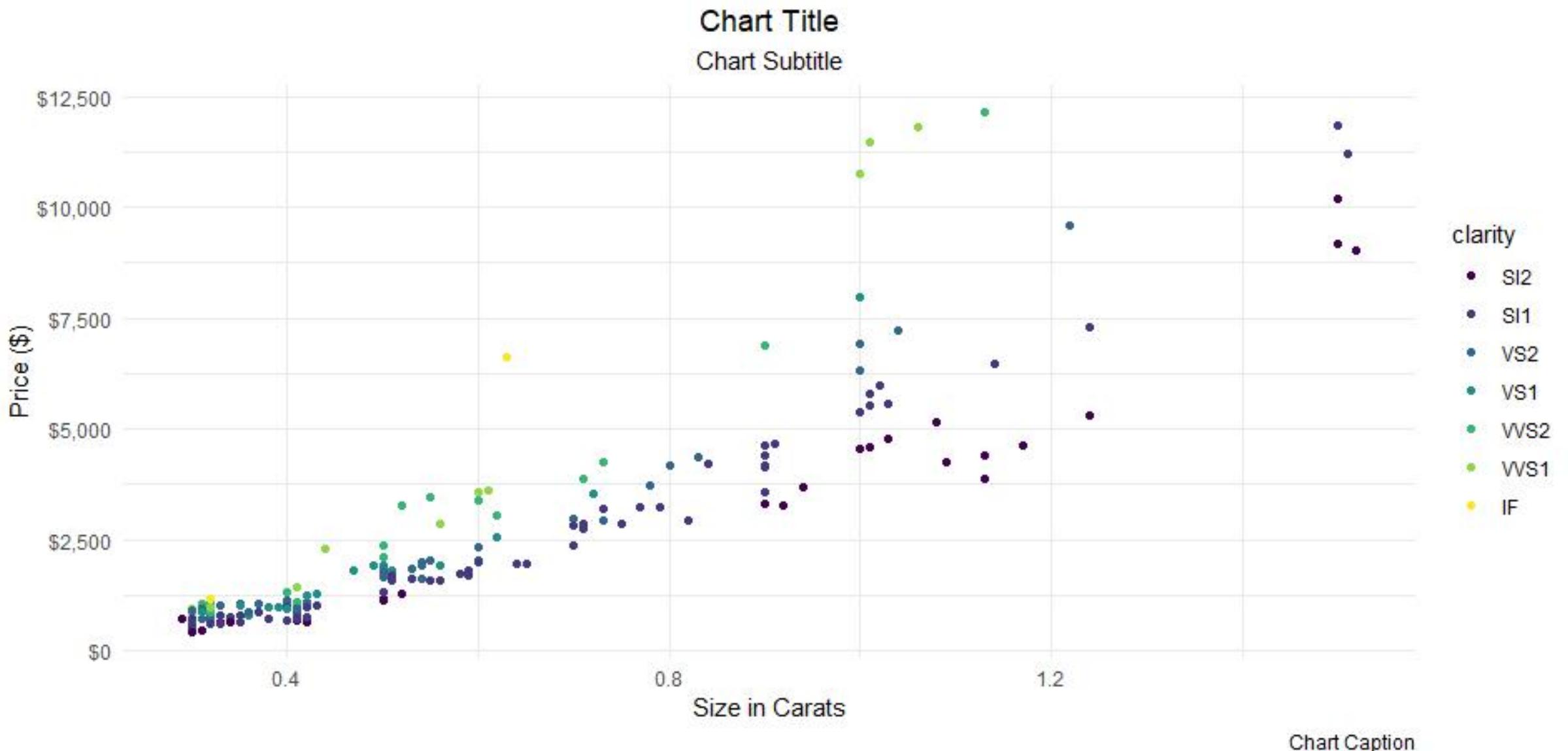
Chart Title

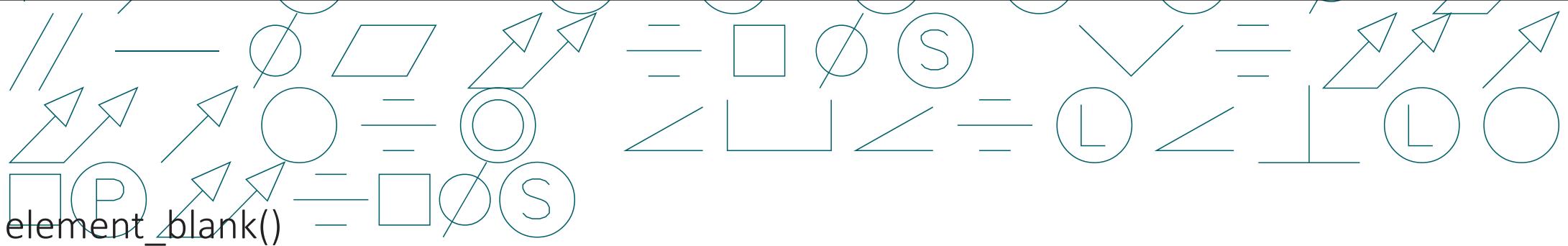
Chart Subtitle





```
ggplot(plot.data, aes(x = carat, y = price)) +  
  geom_point(aes(color = clarity)) +  
  labs(  
    title = "Chart Title",  
    subtitle = "Chart Subtitle",  
    caption = "Chart Caption"  
) +  
  scale_x_continuous(name = "Size in Carats") +  
  scale_y_continuous(  
    name = "Price ($)",  
    labels = dollar_format()  
) +  
  theme_minimal() +  
  theme(  
    plot.title = element_text(hjust = 0.5),  
    plot.subtitle = element_text(hjust = 0.5),  
    plot.caption = element_text(hjust = 1)  
)
```



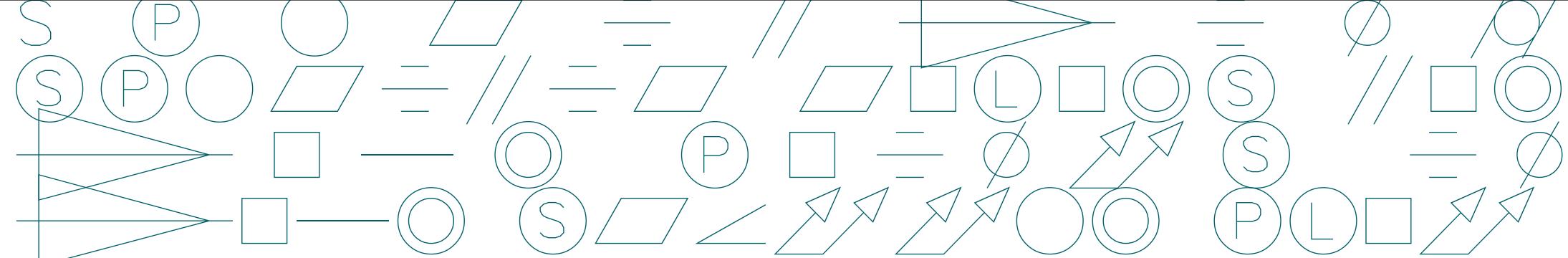


`element_rect(fill = NULL, colour = NULL, size = NULL, linetype = NULL, color = NULL,
inherit.blank = FALSE)`

`element_line(colour = NULL, size = NULL, linetype = NULL, lineend = NULL, color = NULL,
arrow = NULL, inherit.blank = FALSE)`

`element_text(family = NULL, face = NULL, colour = NULL, size = NULL, hjust = NULL, vjust =
NULL, angle = NULL, lineheight = NULL, color = NULL, margin = NULL, debug = NULL,
inherit.blank = FALSE)`

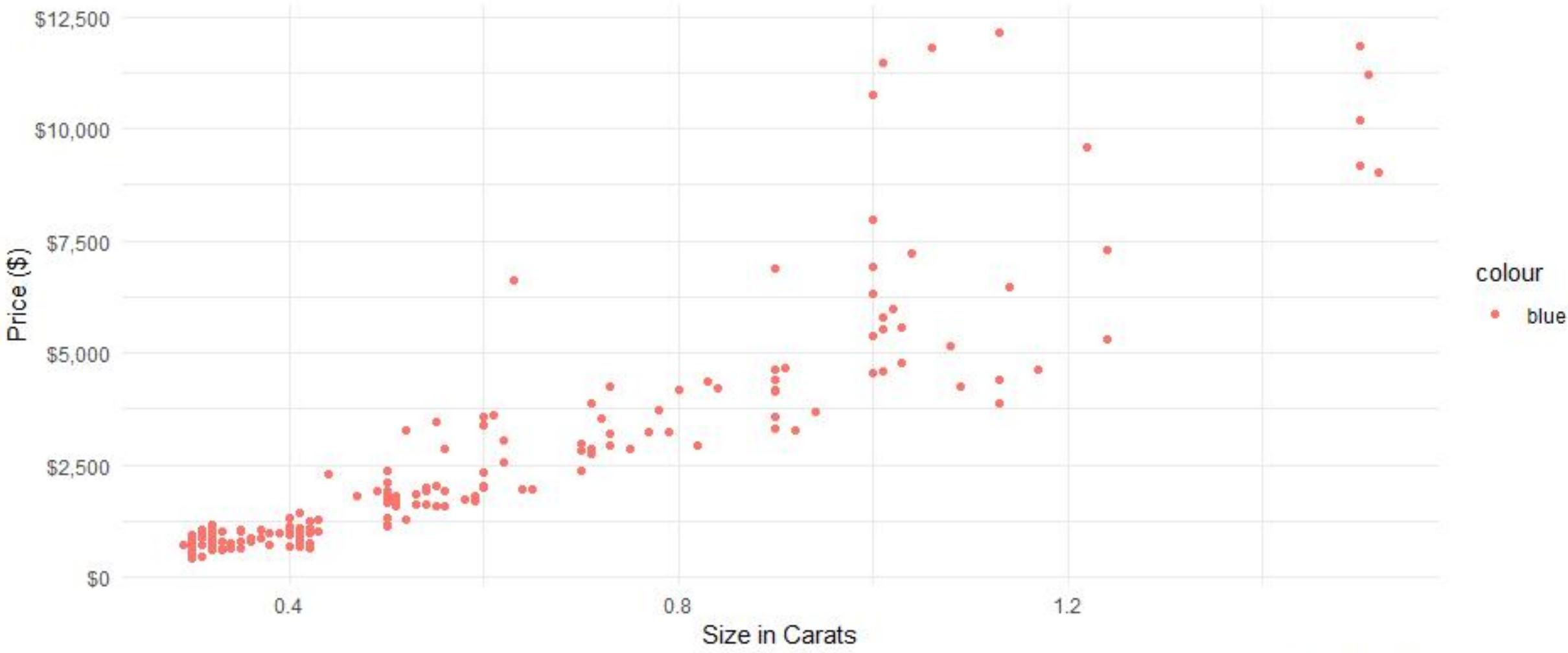
<https://ggplot2.tidyverse.org/reference/element.html>



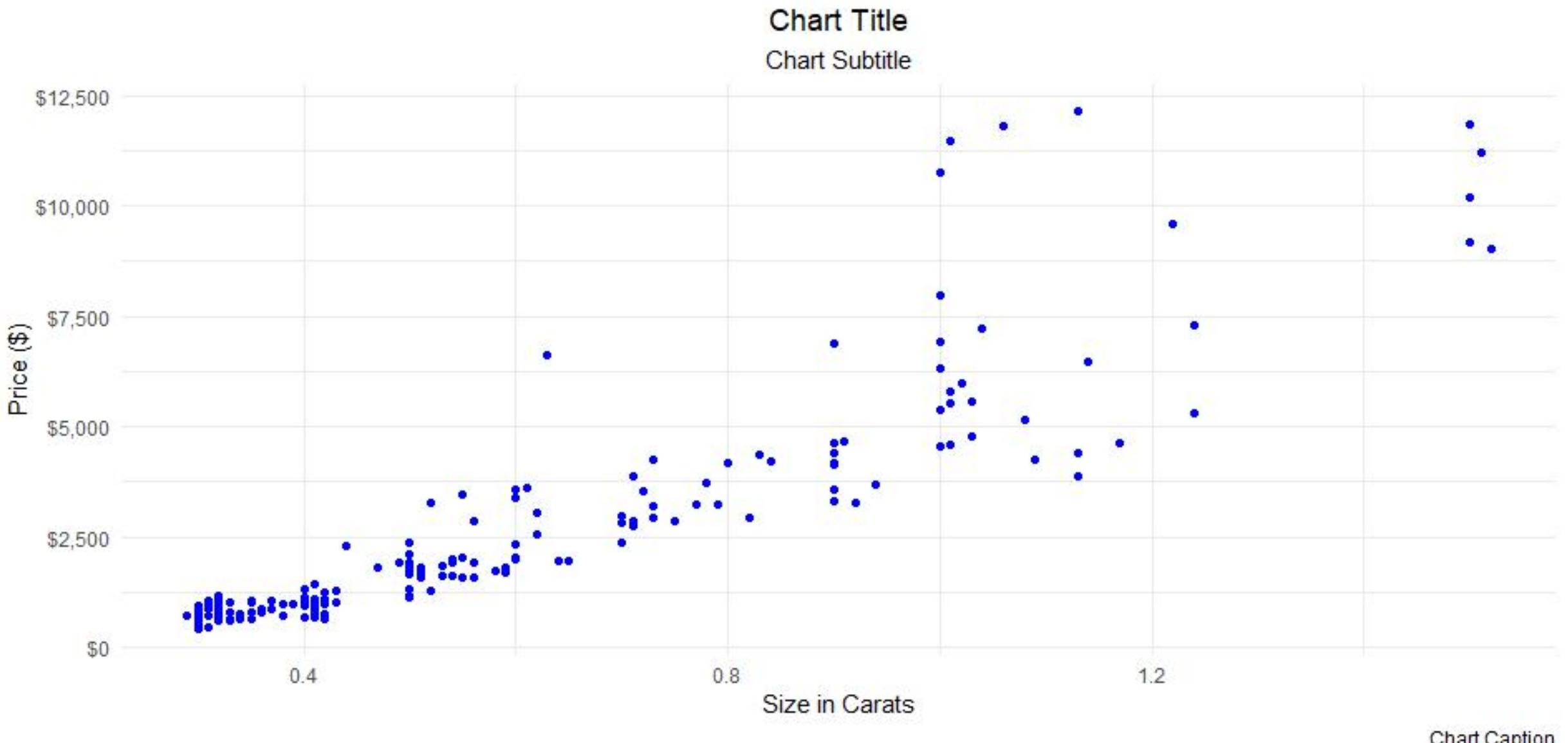
```
ggplot(plot.data, aes(x = carat, y = price)) +  
  geom_point(aes(color = "blue")) +  
  labs(  
    title = "Chart Title",  
    subtitle = "Chart Subtitle",  
    caption = "Chart Caption"  
) +  
  scale_x_continuous(name = "Size in Carats") +  
  scale_y_continuous(  
    name = "Price ($)",  
    labels = dollar_format()  
) +  
  theme_minimal() +  
  theme(  
    plot.title = element_text(hjust = 0.5),  
    plot.subtitle = element_text(hjust = 0.5),  
    plot.caption = element_text(hjust = 1)  
)
```

Chart Title

Chart Subtitle



```
ggplot(plot.data, aes(x = carat, y = price)) +  
  geom_point(color = "blue") +  
  labs(  
    title = "Chart Title",  
    subtitle = "Chart Subtitle",  
    caption = "Chart Caption"  
) +  
  scale_x_continuous(name = "Size in Carats") +  
  scale_y_continuous(  
    name = "Price ($)",  
    labels = dollar_format()  
) +  
  theme_minimal() +  
  theme(  
    plot.title = element_text(hjust = 0.5),  
    plot.subtitle = element_text(hjust = 0.5),  
    plot.caption = element_text(hjust = 1)  
)
```



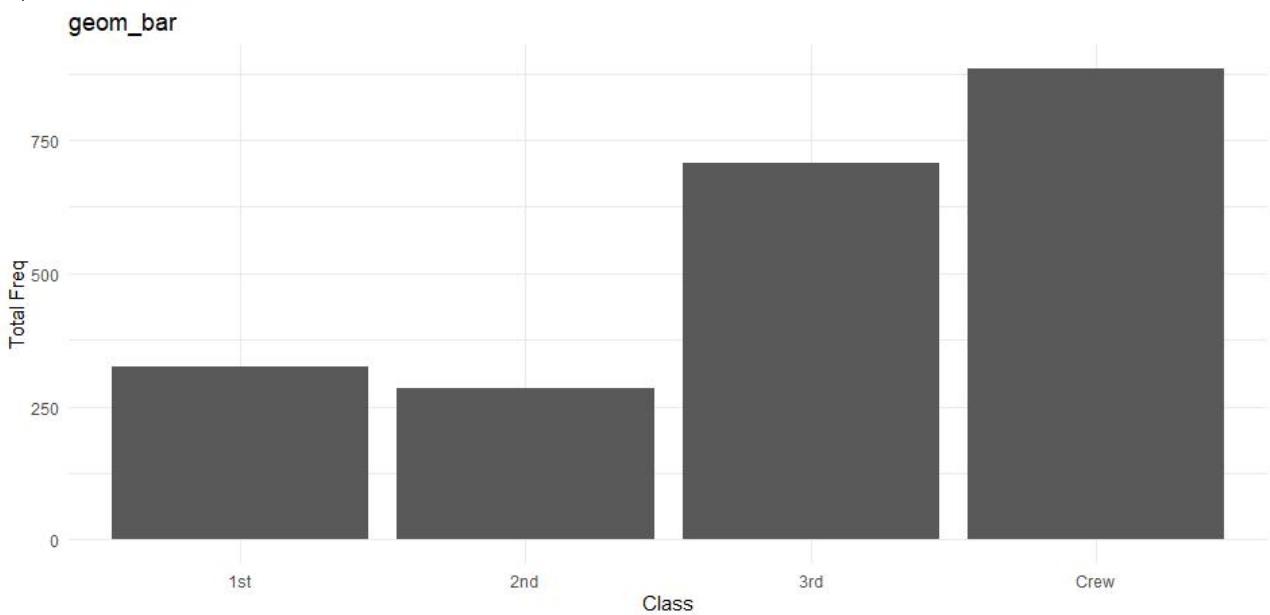
P□P—L<O ↗ O□M S

- *geom_bar()*: column charts and horizontal bar charts.
- *geom_histogram()*: histogram chart.
- *geom_line()*: line charts
- *geom_point()* and *geom_smooth()*: scatter plot with regression line.

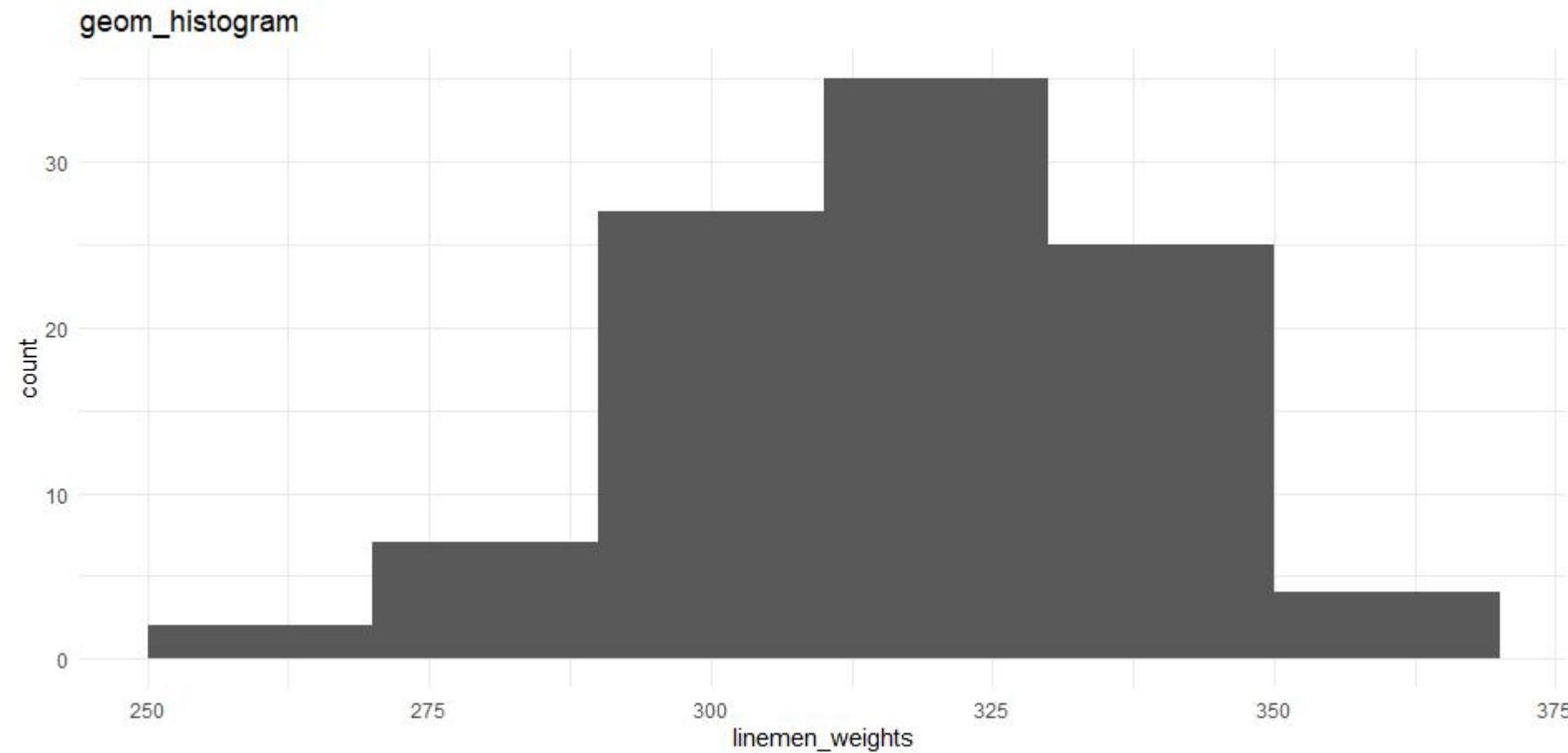


```
plot.data <-
  data.frame(Titanic) %>%
  group_by(Class) %>%
  summarize(`Total Freq` = sum(Freq))

ggplot(plot.data, aes(x = Class, y = `Total Freq`)) +
  geom_bar(stat = "identity") +
  labs(title = "geom_bar") +
  theme_minimal()
```



h○□M → ≡ S ▲ □ h○∠M()





```
library(tidyverse)
```

```
library(lubridate)
```

```
players <- c("Kobe Bryant", "Pau Gasol", "Lamar Odom")
```

```
plot.data <- lakers %>%
```

```
  filter(result == "made" & team == "LAL" & player %in% players) %>%
```

```
  mutate(date = ymd(date)) %>%
```

```
  group_by(player, date) %>%
```

```
  summarize(points = sum(points))
```

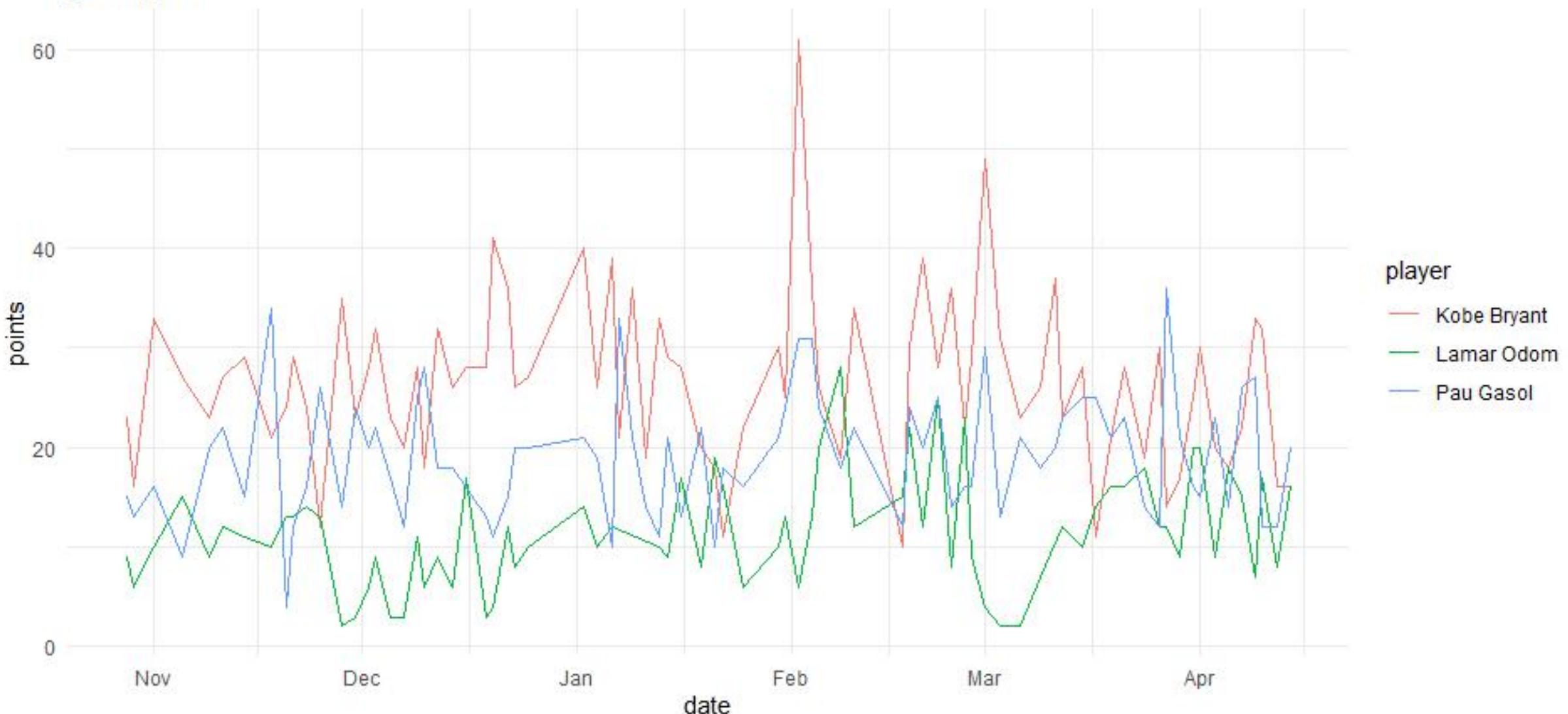
```
ggplot(plot.data, aes(x=date,y=points,color=player)) +
```

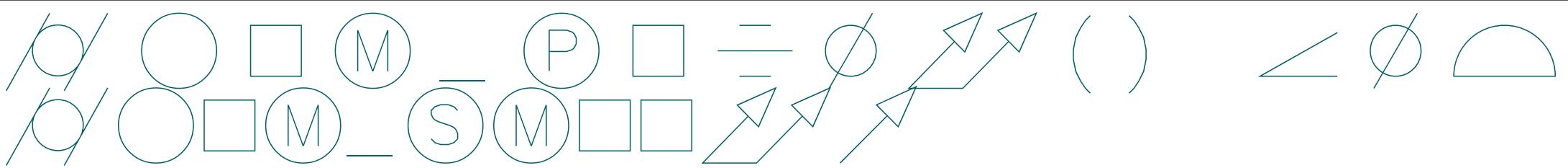
```
  geom_line() +
```

```
  labs(title = "geom_line") +
```

```
  theme_minimal()
```

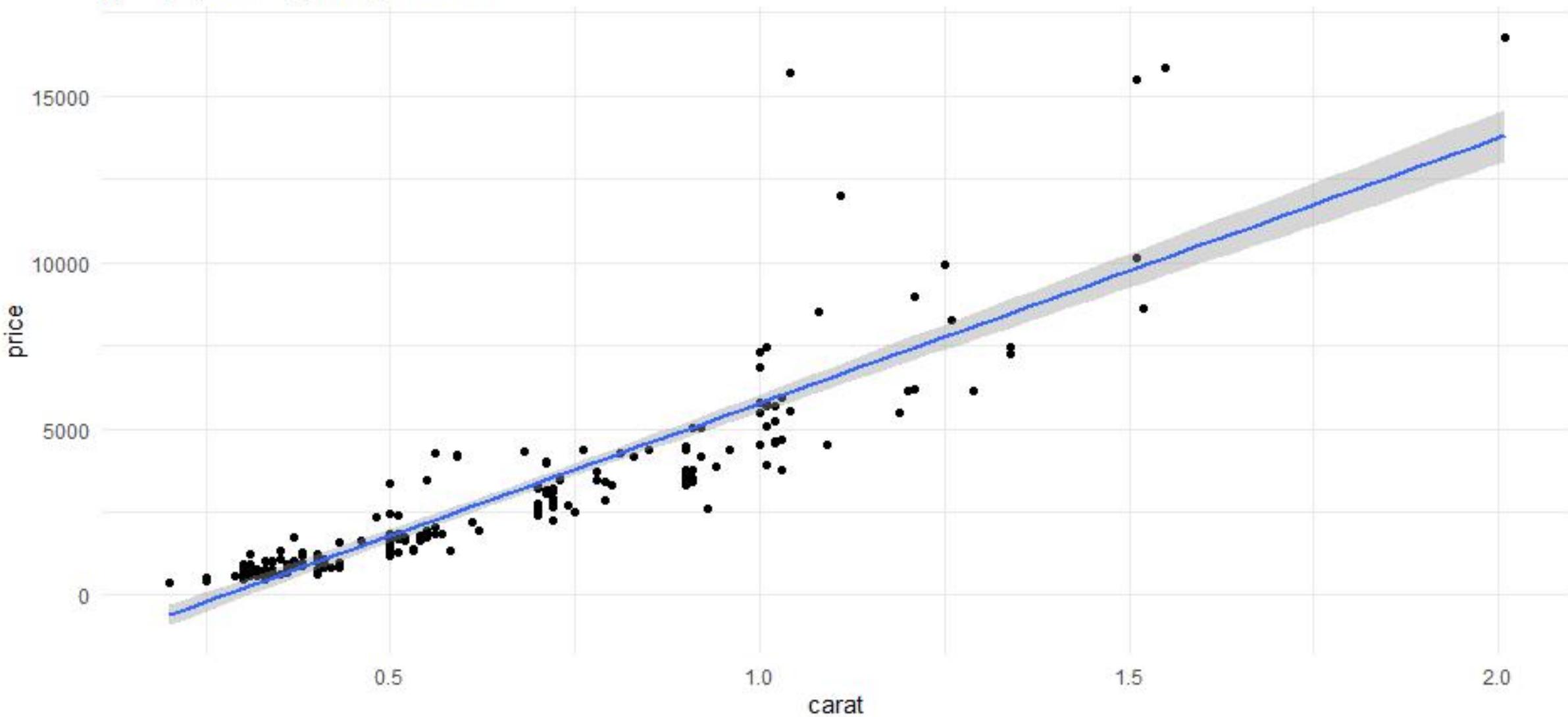
geom_line



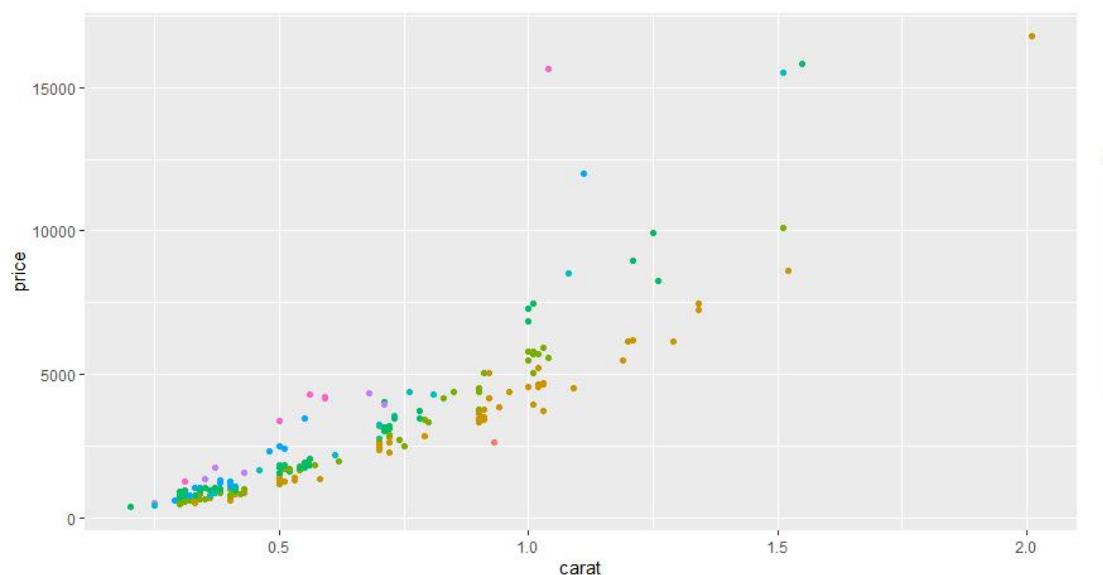
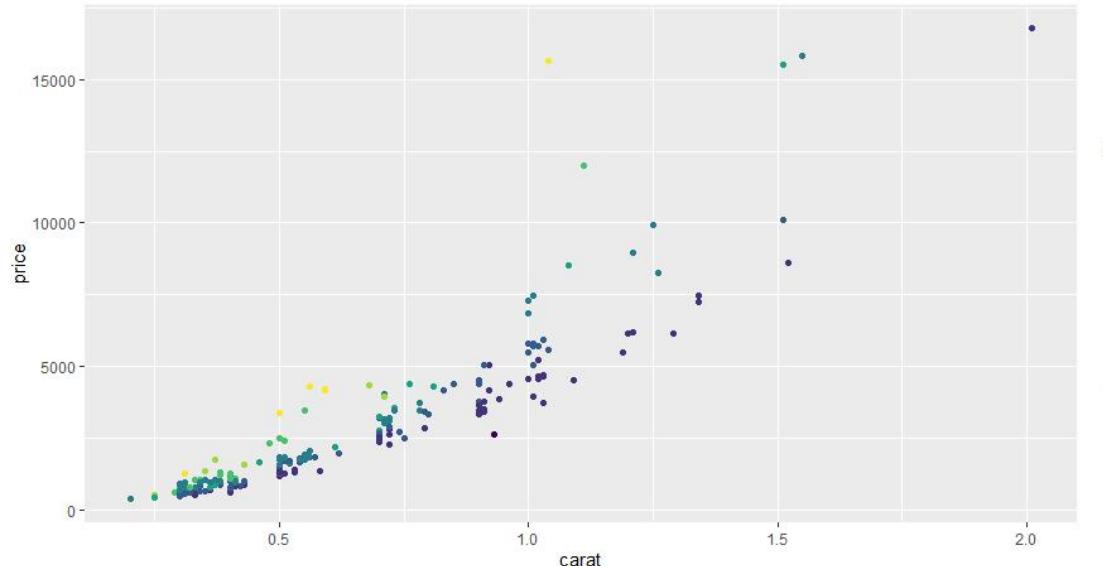


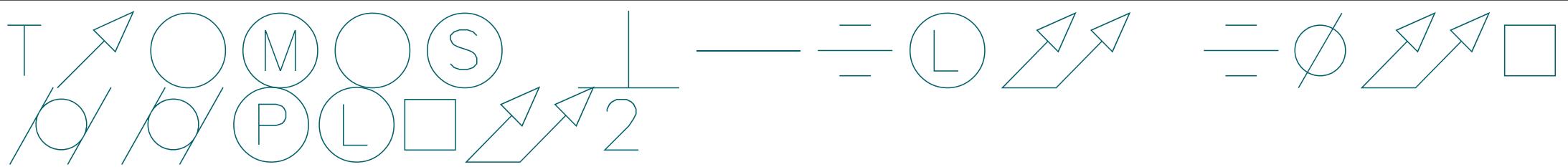
```
library(tidyverse)
set.seed(1)
plot.data <-
  diamonds %>%
  filter(color == "D") %>%
  sample_n(200)
ggplot(plot.data, aes(x=carat, y=price)) +
  geom_point() +
  geom_smooth(method ="lm") +
  labs(title = "geom_point & geom_smooth") +
  theme_minimal()
```

geom_point & geom_smooth

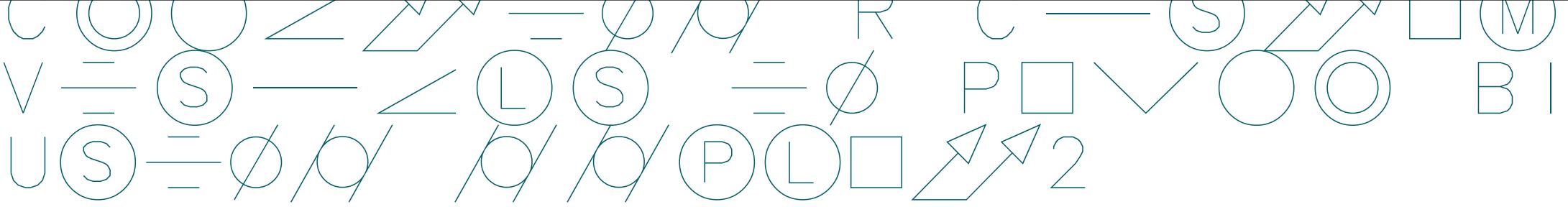


```
ggplot(plot.data, aes(x = carat, y = price)) +  
  geom_point(aes(color = clarity))  
  
ggplot(plot.data, aes(x = carat, y = price)) +  
  geom_point(aes(color = clarity)) +  
  scale_x_continuous() +  
  scale_y_continuous() +  
  scale_color_discrete()
```





- *theme_bw()*: A variation on *theme_gray()* that uses a white background and thin gray grid lines
- *theme_linedraw()*: A theme with only black lines of various widths on white backgrounds, reminiscent of a linedrawing
- *theme_light()*: Similar to *theme_linedraw()* but with light gray lines and axes, to direct more attention toward the data
- *theme_dark()*: The dark cousin of *theme_light()*, with similar line sizes but a dark background. Useful to make thin colored lines pop out
- *theme_minimal()*: A minimalistic theme with no background annotations
- *theme_classic()*: A classic-looking theme, with x and y axis lines and no gridlines
- *theme_void()*: A completely empty theme

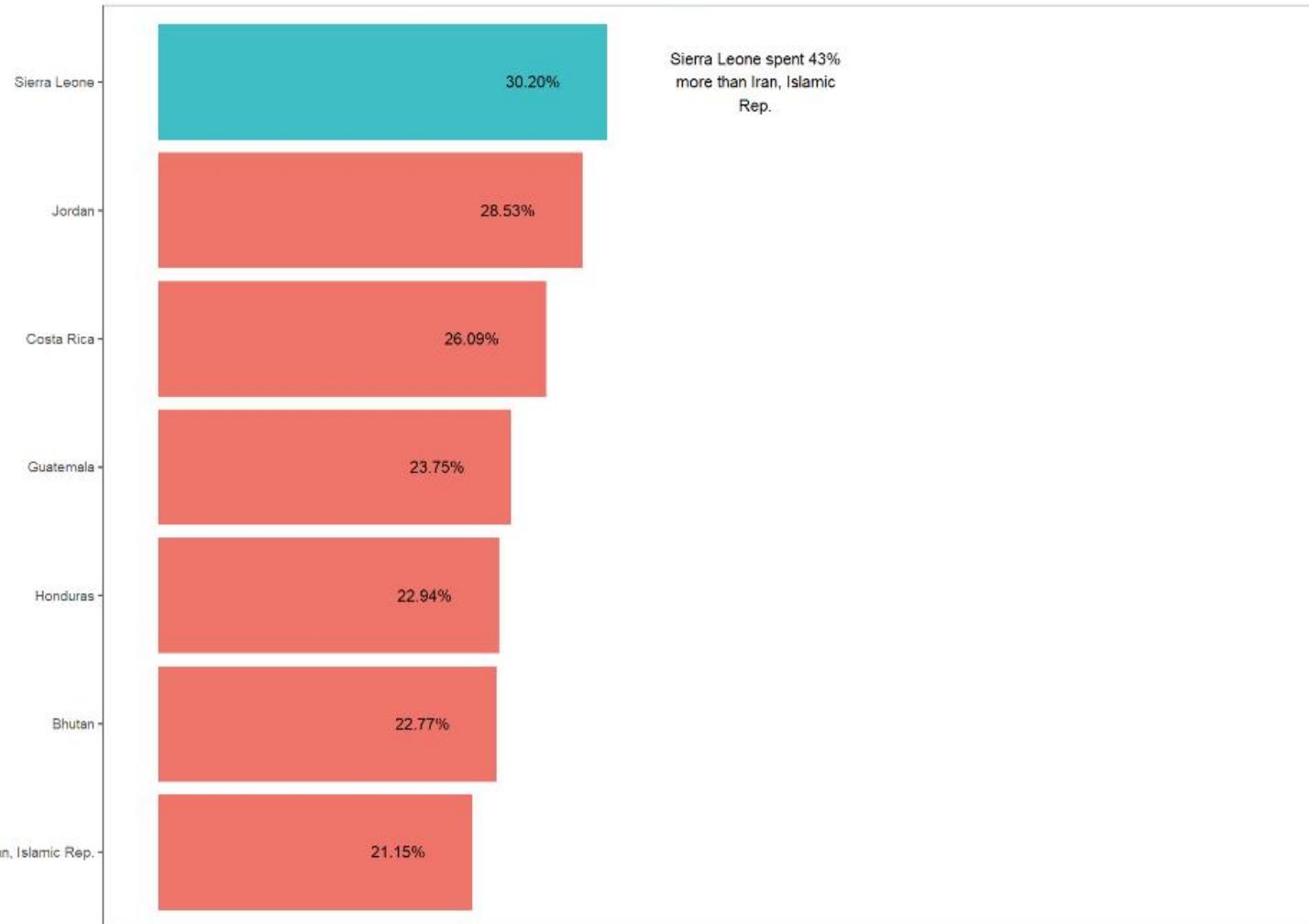


- A *callout chart* with dynamic titles and annotations
- A *bubble chart* with dynamic titles that displays five dimensions of data
- A detailed *forecast visualization* that forecasts the Wikipedia page views for Lamar Jackson and Deshaun Watson
- A *line chart* that displays GDP by Year with a *background shade* based on the political party in power for the point in time
- A *map visualization* that displays the state selected and colors the counties in that state based on the population quintile the county is in
- A *quad chart* that compares LA Lakers basketball players based on total points scored and rebounds made in the 2008–2009 season
- A *scatter plot* of the average weight by height of US women that is overlaid with a *regression line*

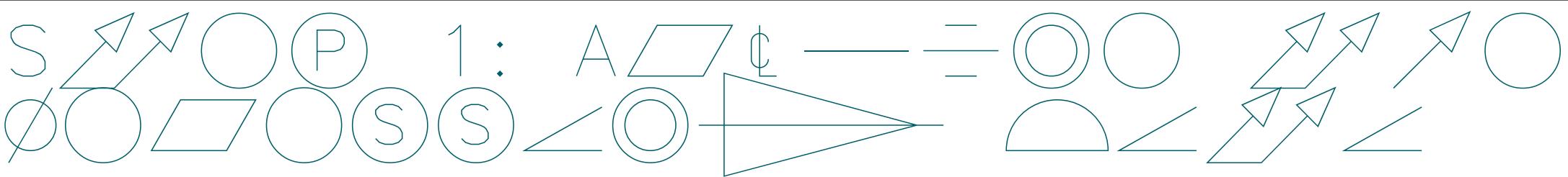


% of Government Expenditure on Education

The top 7 ranked countries in 2018

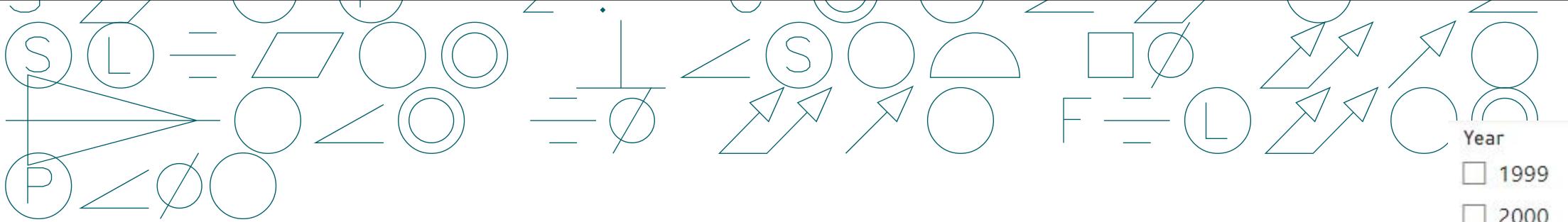


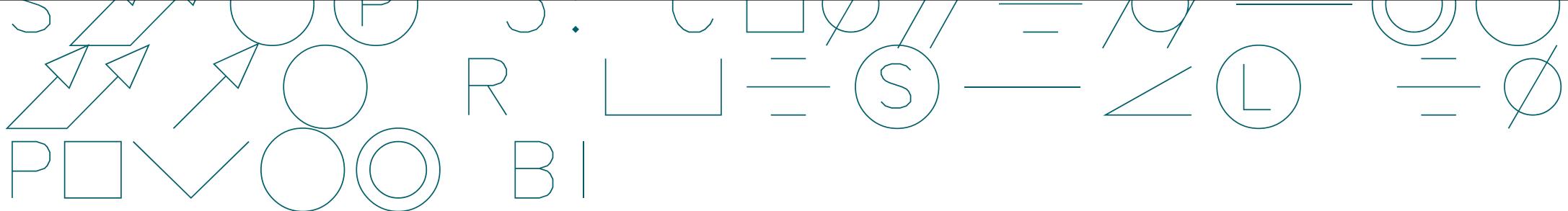
Source: <https://datacatalog.worldbank.org/source/world-development-indicators>



- Import “calloutChartData.csv” data file.

Column Name	Data Type
Country	Text
Country Code	Text
Year	Whole number
Total Expend %	Decimal number





Visualizations >>

Build visual

Values

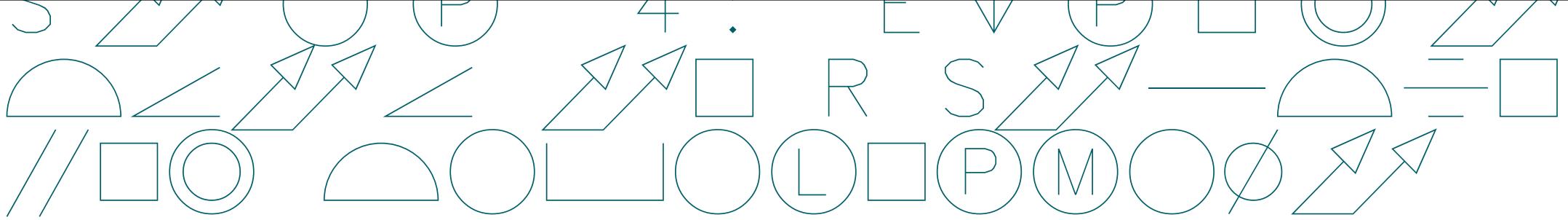
Country	▼	X
Total Expend %	▼	X
Year	▼	X

Drill through

Cross-report Off

Keep all filters On

Add drill-through fields here



R script editor

⚠ Duplicate rows will be removed from the data.

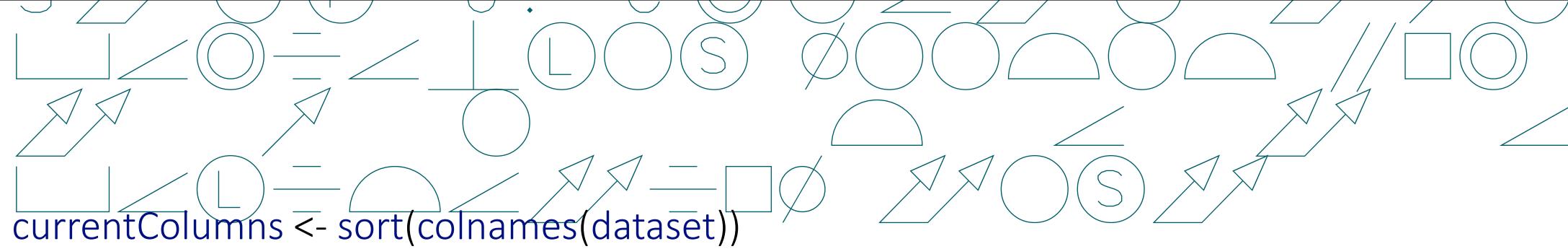
```
1 # The following code to create a data frame and remove duplicated rows is always executed and acts as a preamble for your script:  
2  
3 # dataset <- data.frame(Country, Total Expend %, Year)  
4 # dataset <- unique(dataset)  
5  
6 # Paste or type your script code here:
```

P J . L Z = O S

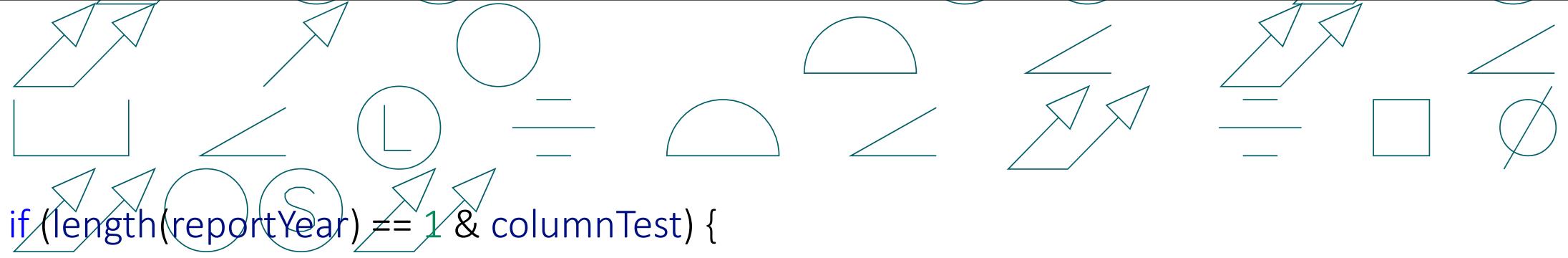
library(tidyverse)

library(scales)

library(ggthemes)



```
currentColumns <- sort(colnames(dataset))  
requiredColumns <- c("Country", "Total Expend %", "Year")  
columnTest <- isTRUE(all.equal(currentColumns, requiredColumns))  
reportYear <- unique(dataset$Year)
```



```
if (length(reportYear) == 1 & columnTest) {
```

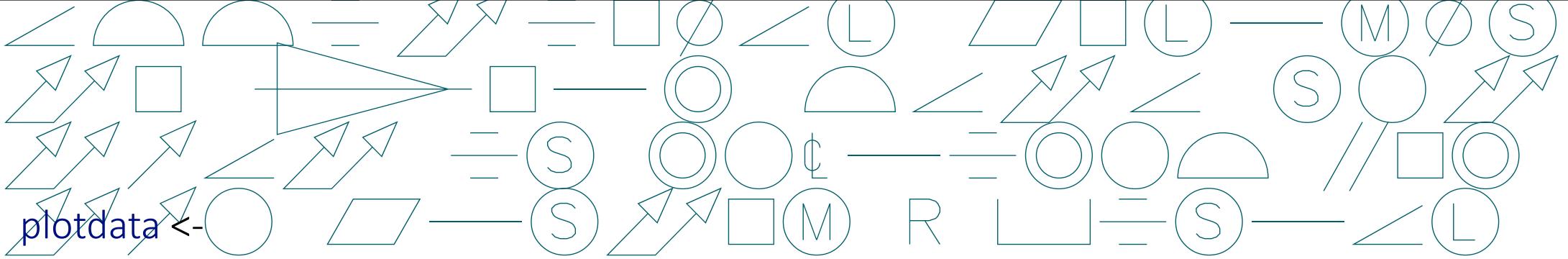
```
    <code to produce visual>
```

```
} else {
```

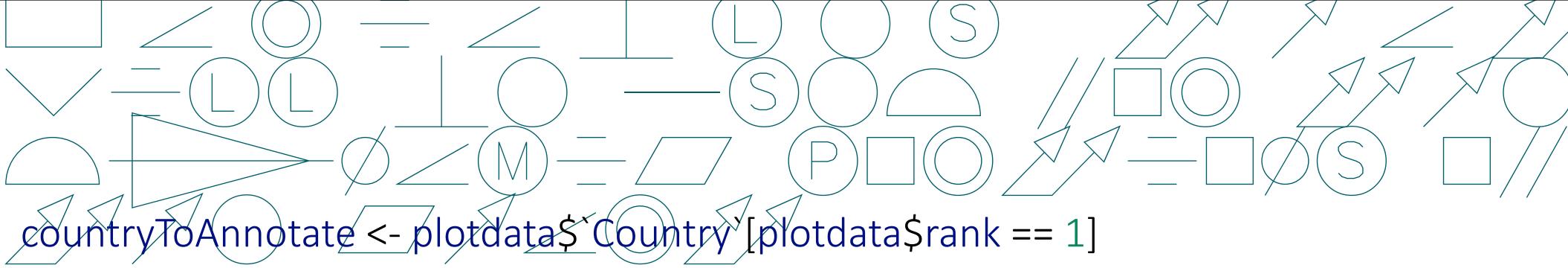
```
    plot.new()
```

```
    title("The data supplied did not meet the requirements of the chart.")
```

```
}
```



```
plotdata <- dataset %>%
  mutate(
    rank = dense_rank(desc(`Total Expend %`)),
    `Country` = fct_reorder(`Country`, rank, .desc = TRUE),
    callout = ifelse(rank == 1, TRUE, FALSE)
  ) %>%
  filter(rank <= 7)
```



```
countryToAnnotate <- plotdata$Country[plotdata$rank == 1]
```

```
minExpenditure <- min(plotdata`Total Expend %`)
```

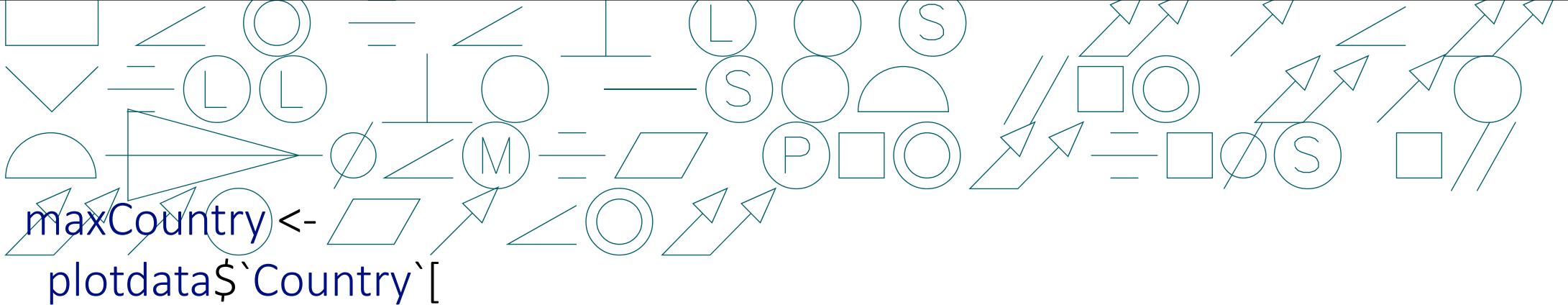
```
maxExpenditure <- max(plotdata`Total Expend %`)
```

```
minCountry <-
```

```
plotdata$Country[
```

```
which(plotdata`Total Expend %` == minExpenditure)]
```

```
minCountry <- paste(minCountry, collapse = " & ")
```



```
maxCountry <- paste(maxCountry, collapse = " & ")
```

```
mainTitle = "% of Government Expenditure on Education"
```

```
subTitle =
```

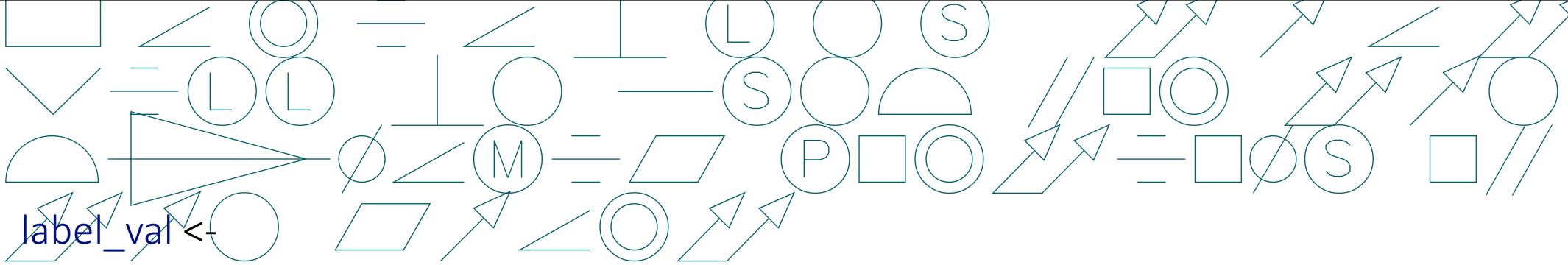
```
paste(
```

"The top 7 ranked countries in",

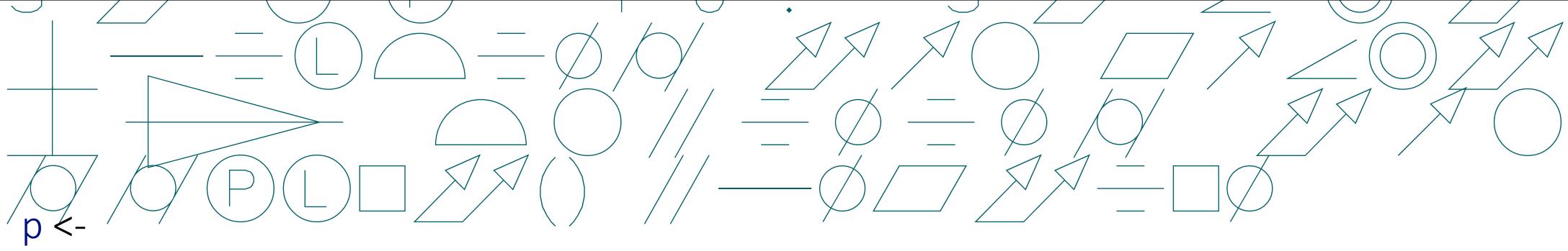
```
reportYear,
```

```
sep = " ")
```

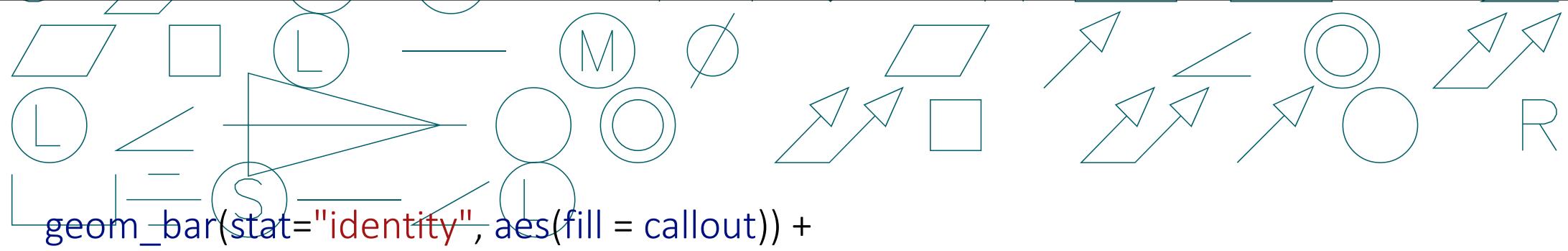
caption = "Source: <https://databank.worldbank.org/source/world-development-indicators>"

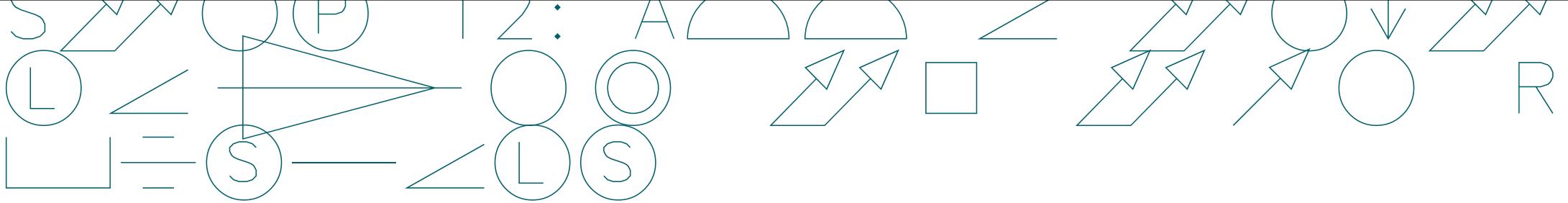


```
label_val <- str_wrap(  
  paste(maxCountry, "spent",  
    percent((maxExpenditure/minExpenditure-1)),  
    "more than", minCountry, sep = " "),  
  width = 25  
)
```

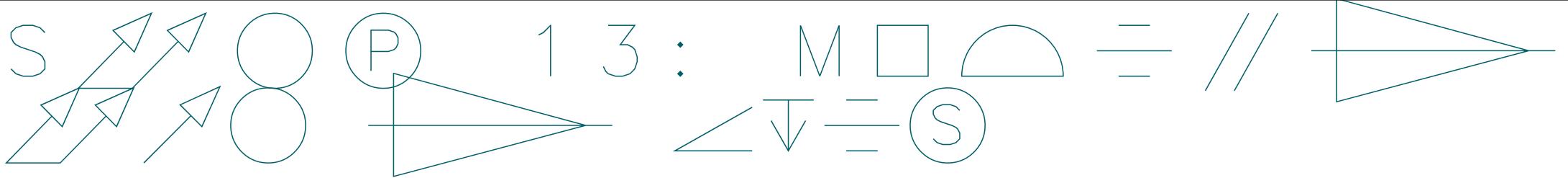


```
ggplot(  
  data = plotdata,  
  aes(x = `Country`, y = `Total Expend %`, fill = callout,  
      label = percent(`Total Expend %`))  
) +
```



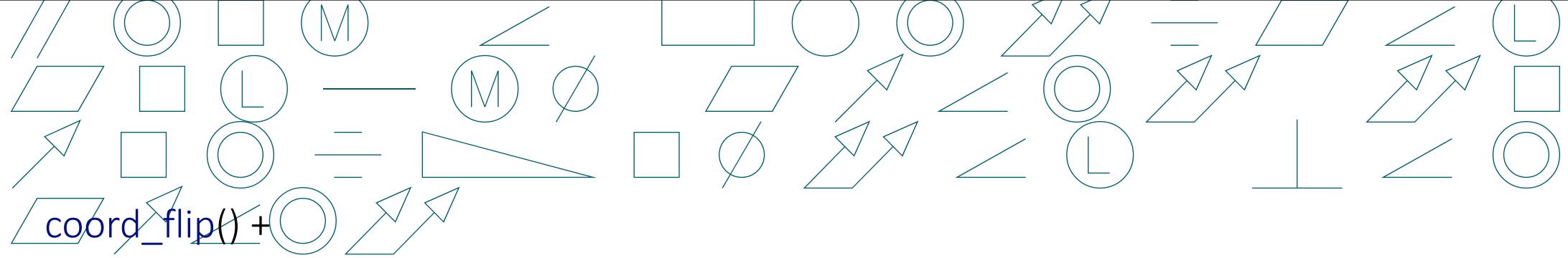


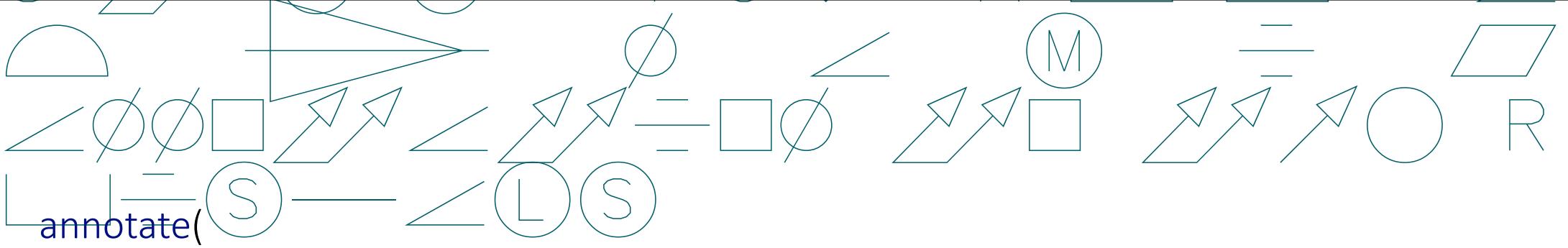
geom_text(nudge_y = -0.05) +



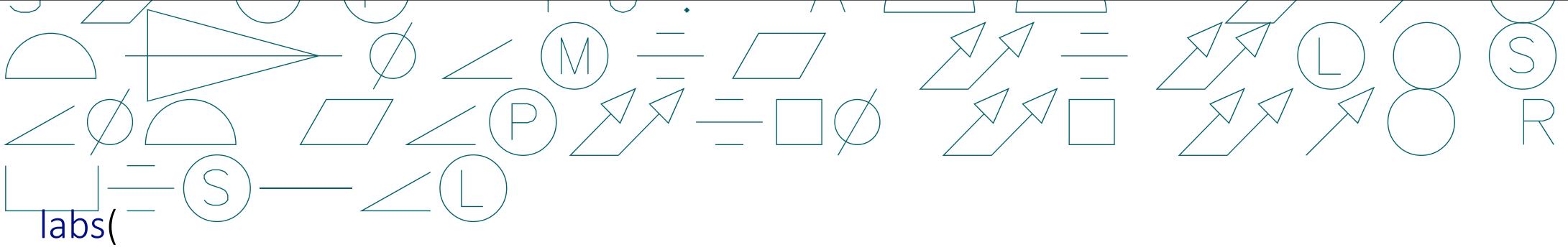
scale_y_continuous(

limits = c(0,0.75), labels = NULL, breaks = NULL) +





```
"text",  
label = label_val,  
x = countryToAnnotate[1],  
y = maxExpenditure + 0.1) +
```



title = mainTitle,

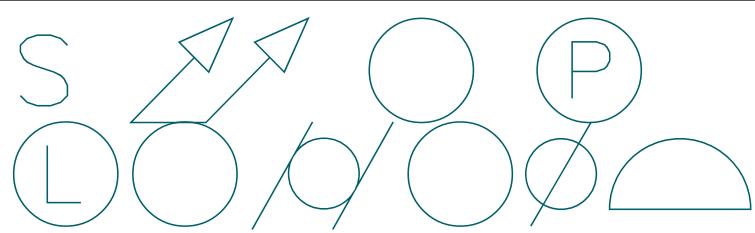
subtitle = subTitle,

caption = caption

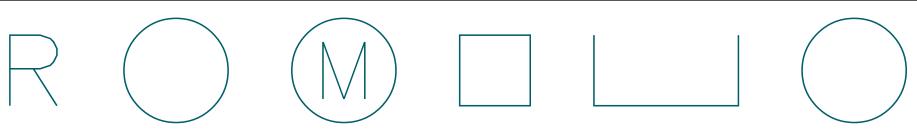
) +

S *Z* *T* *P* *R* *M*
L *O* *L* *S* *O* *D*
Q *H* *M* *V* *E* *S*
∅ *U* *—* *—* *—* *S*

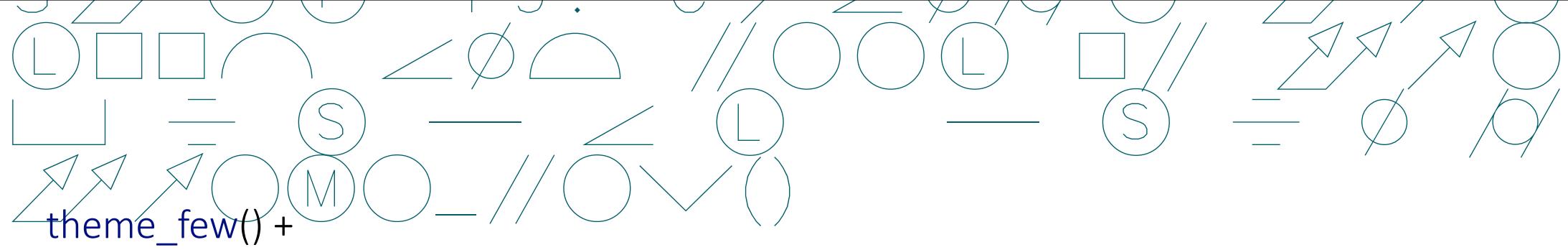
xlab(label = **NULL**) +
ylab(label = **NULL**) +



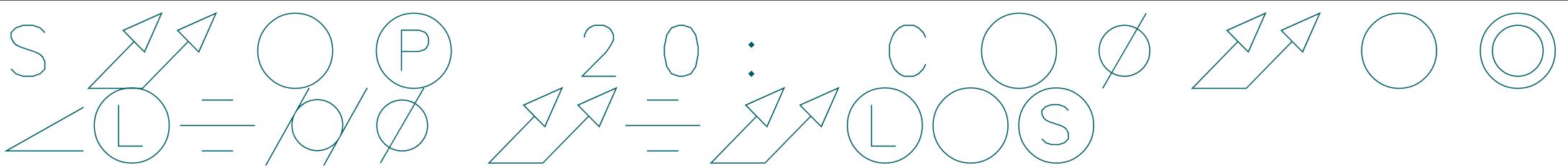
1 8 :



guides(fill=FALSE) +



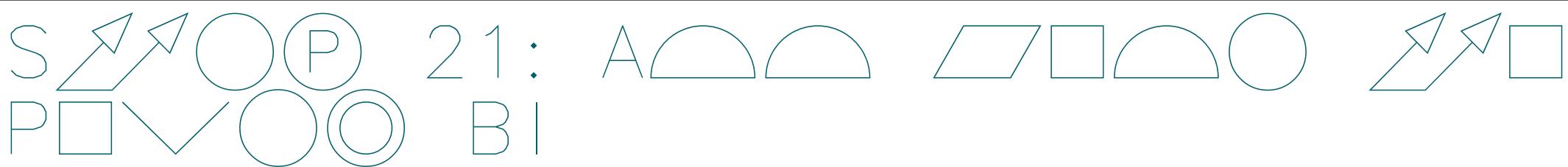
theme_few() +



theme(

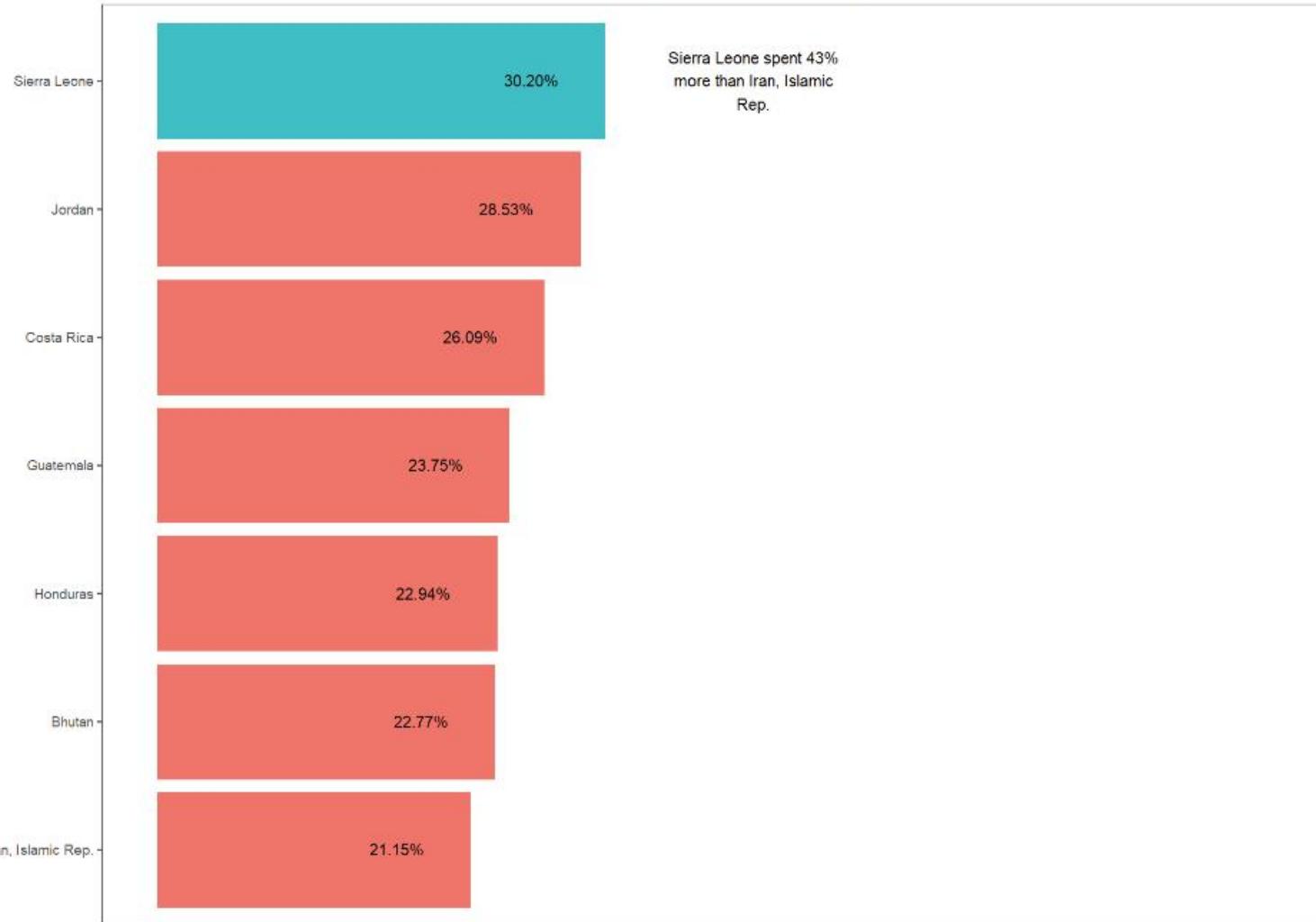
```
  plot.title = element_text(hjust = 0.5),  
  plot.subtitle = element_text(hjust = 0.5))
```

p



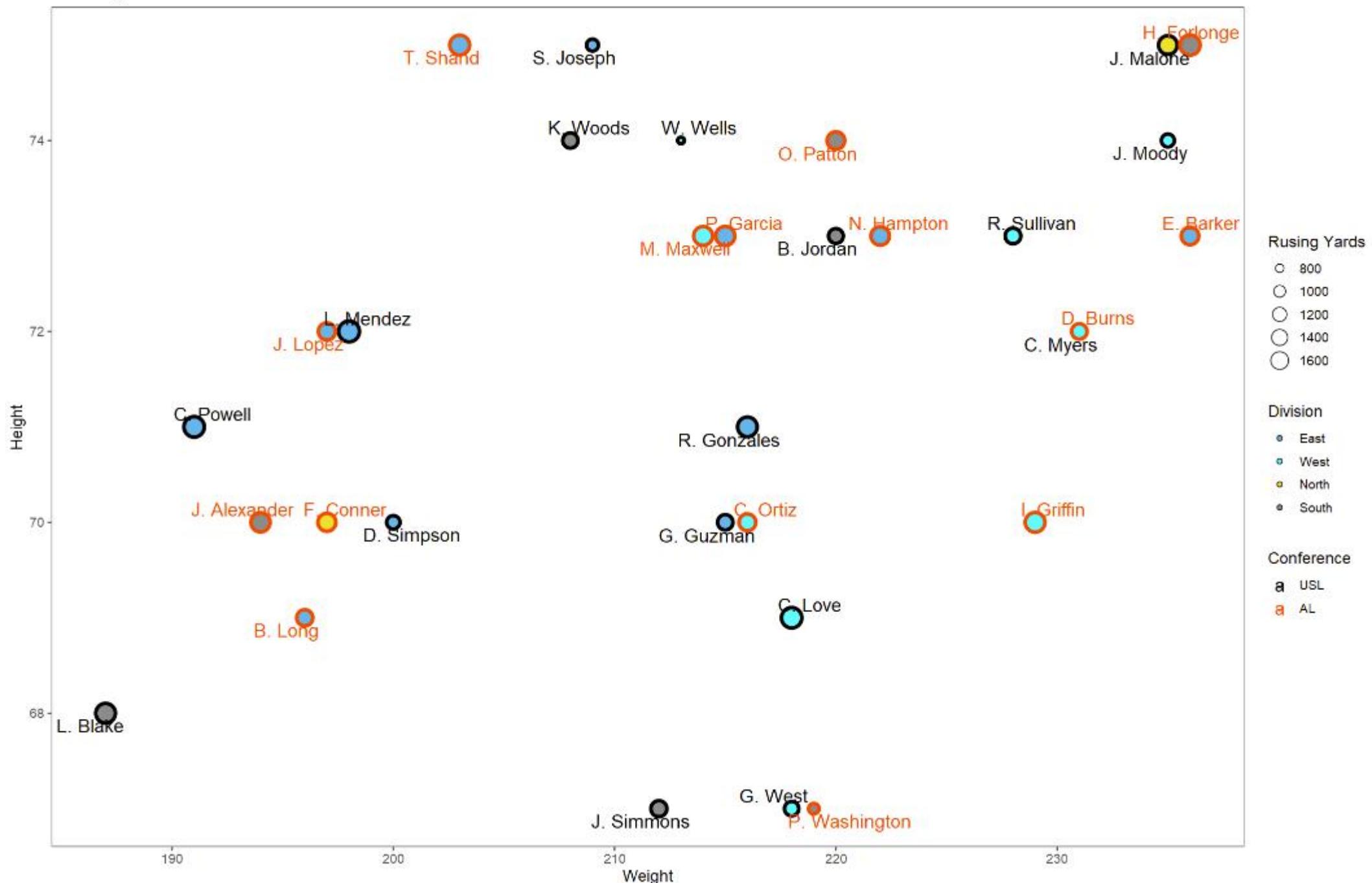
% of Government Expenditure on Education

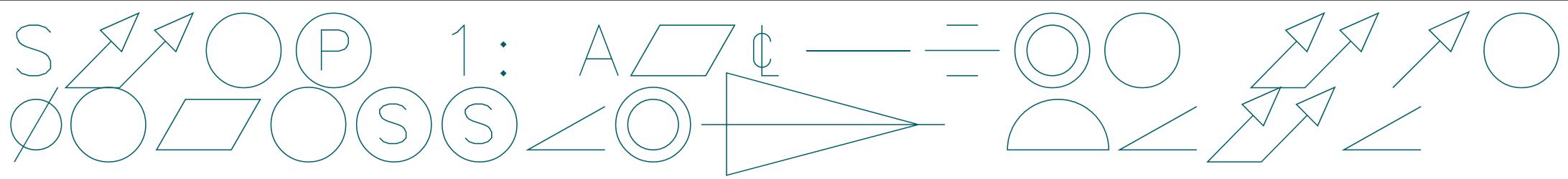
The top 7 ranked countries in 2018



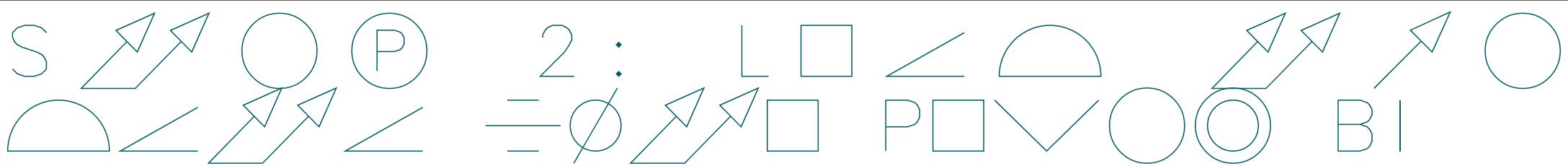
Source: <https://databank.worldbank.org/source/world-development-indicators>

Runningback Quad Chart for 2019

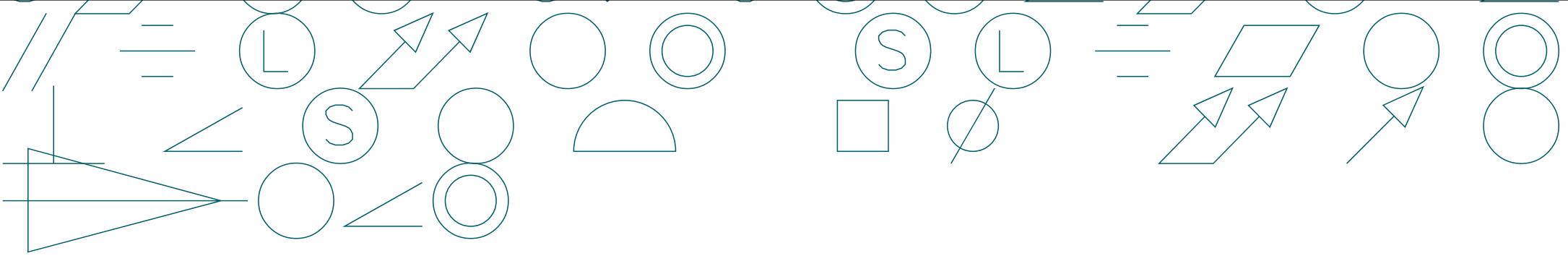




- Import “FakeFootballLeagueData.csv” data file.



Column Name	Column Data Type
ID	Whole Number
Year	Whole Number
Key	Whole Number
FN	Text
LN	Text
Weight	Whole Number
Height	Whole Number
Rushing Yards	Whole Number
Division	Text
Conference	Text



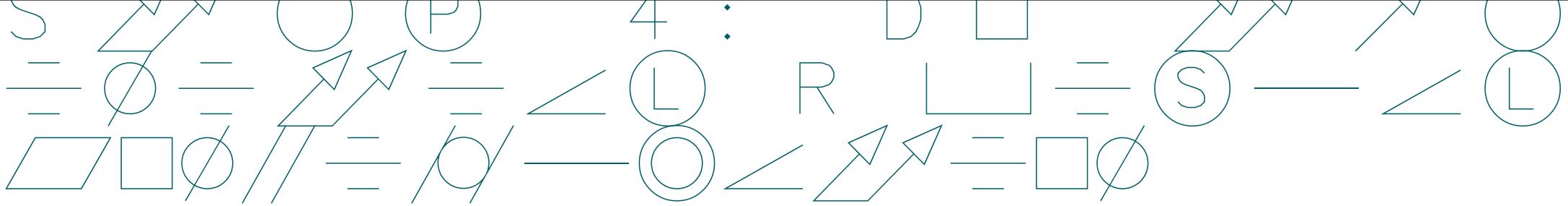
Year ▾

2016

2017

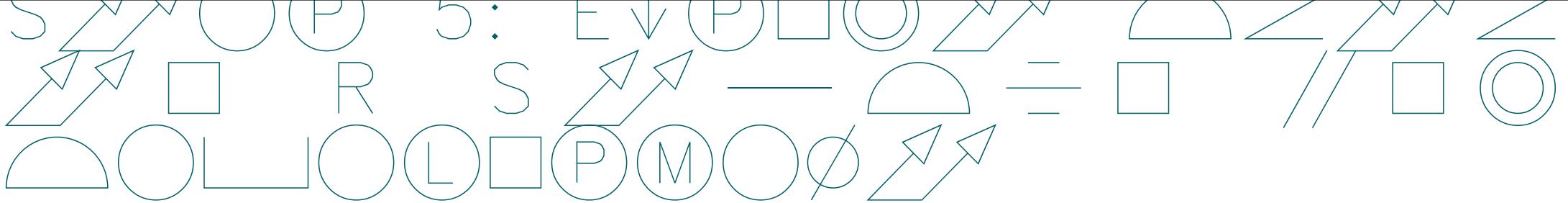
2018

2019



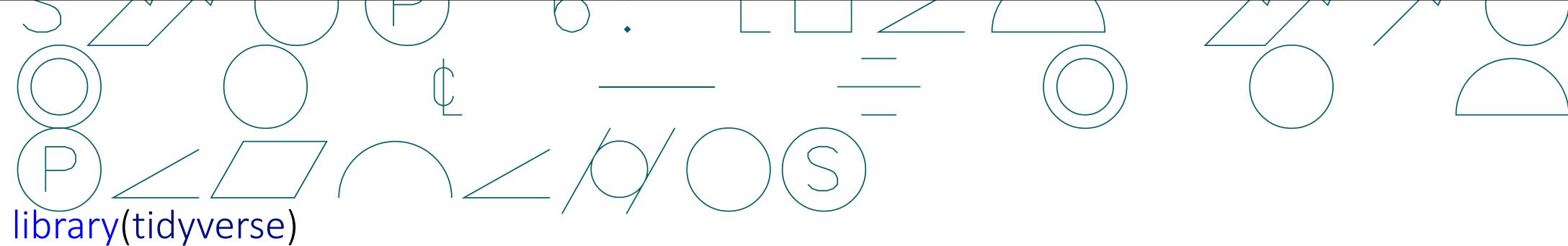
The screenshot shows a software interface with a toolbar at the top containing icons for file operations, data entry, and other functions. Below the toolbar is a section titled "Values" which lists several fields with checkboxes next to them. The fields and their status are:

Field	Status
ID	✓ X
Year	✓ X
FN	✓ X
LN	✓ X
Weight	✓ X
Height	✓ X
Rushing Yards	✓ X
Division	✓ X
Conference	✓ X



R script editor

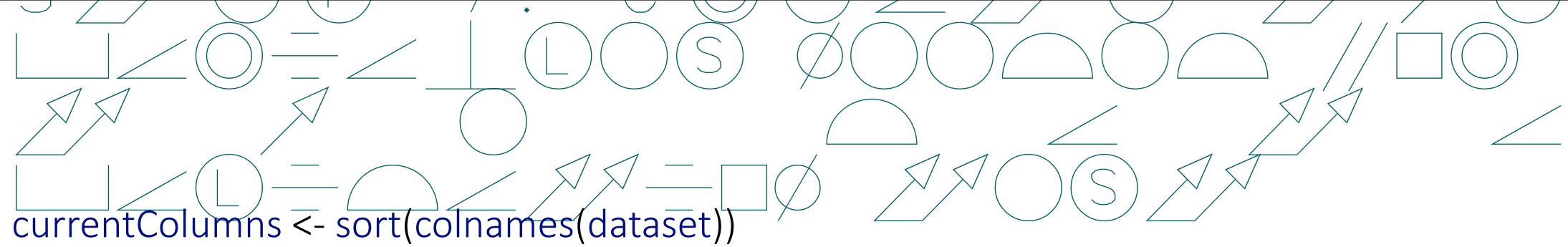
```
1 # The following code to create a dataframe and remove duplicated rows is always executed and acts as a preamble for your script:  
2  
3 # dataset <- data.frame(ID, Year, FN, LN, Weight, Height, Rusing Yards, Division, Conference)  
4 # dataset <- unique(dataset)  
5  
6 # Paste or type your script code here:
```



```
library(tidyverse)
```

```
library(ggrepel)
```

```
library(ggthemes)
```

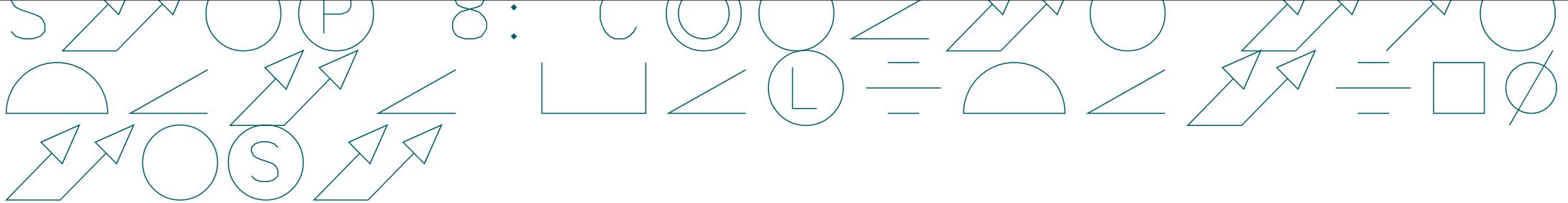


```
currentColumns <- sort(colnames(dataset))

requiredColumns <- c("Conference", "Division", "FN", "Height", "ID", "LN", "Rusing Yards",
"Weight", "Year")

columnTest <- isTRUE(all.equal(currentColumns, requiredColumns))

reportYear <- unique(dataset$Year)
```

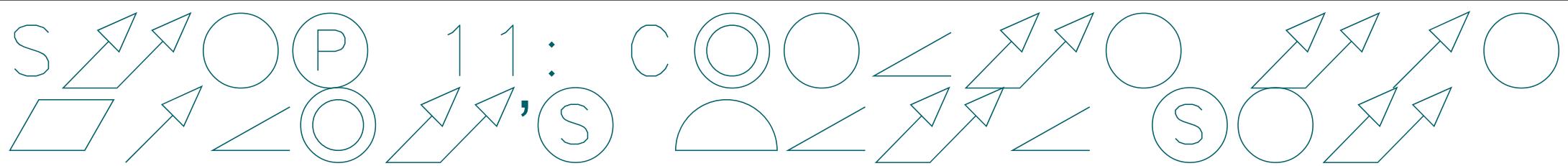


```
if(length(reportYear) == 1 & columnTest) {  
  <code to create the visual>  
} else{  
  
  plot.new()  
  title("The data supplied did not meet the requirements of the chart.")  
}
```

```
divisionColors <- c("East"="#56B4E9", "West"="#33FAFF", "North"#FOE442,
"South"="#8B8D8D")
conferenceColors <- c("USL"="#000000", "AL"="#FC4E07")
```

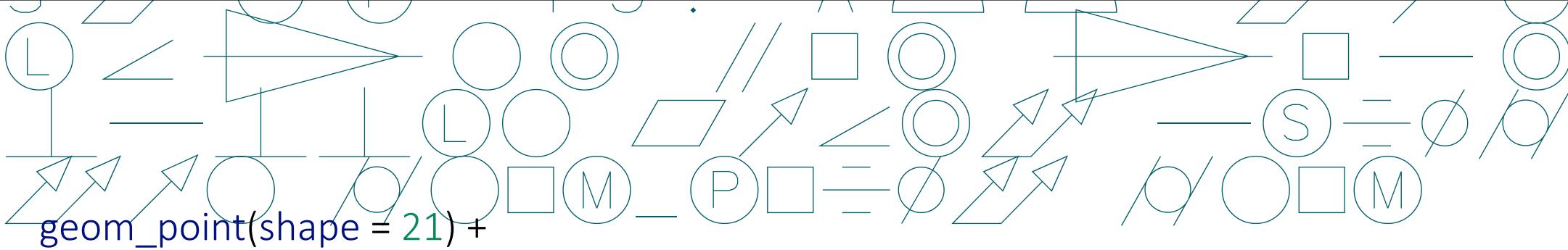
<https://www.hexcolortool.com/#010328>

```
chartTitle<- paste("Runningback Quad Chart for", reportYear, sep = " ")
```

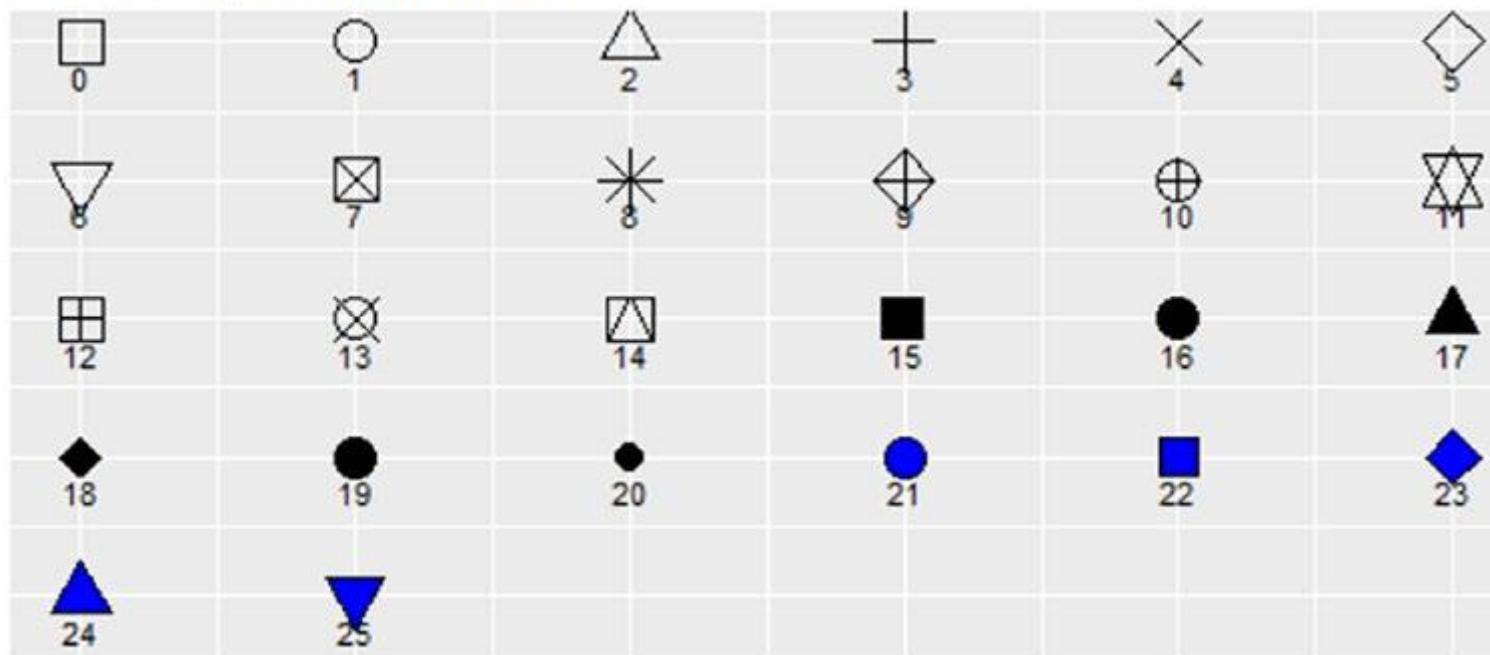


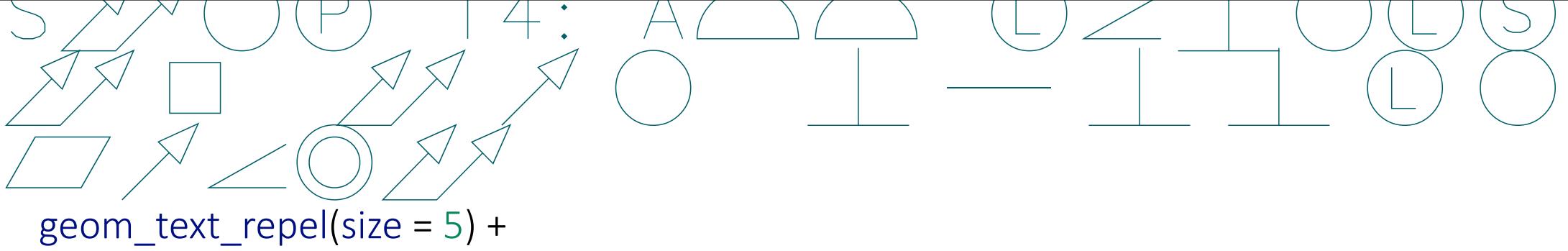
```
chartData <-  
dataset %>%  
mutate(Name = paste0(substr(FN,1,1),". ", LN))
```

```
chartData,  
aes(  
  x = Weight, y = Height, size = `Rusing Yards`,  
  color = Conference, fill = Division, stroke = 2,  
  label = Name  
)  
) +
```



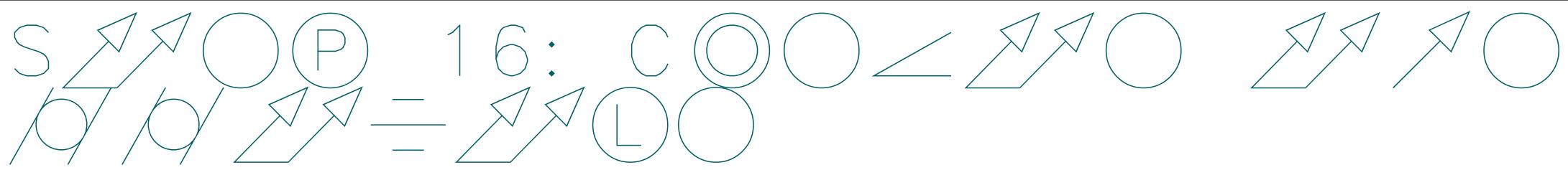
Point shapes available in R



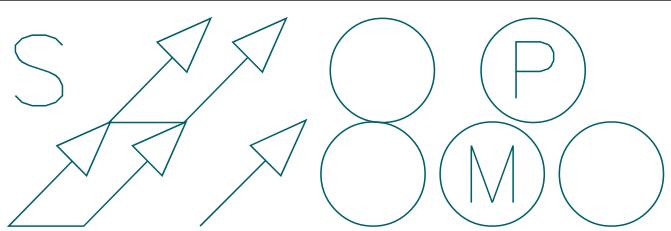


A network graph visualization featuring various nodes and edges. Nodes include circles, squares, diamonds, and arrows pointing in different directions. Some nodes contain letters like 'L', 'S', and 'C'. The graph is overlaid on a grid of horizontal and vertical lines. Below the graph, there is R code:

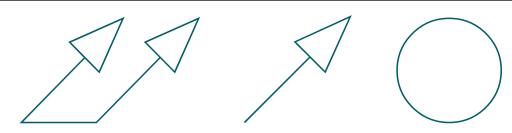
```
scale_color_manual(values = conferenceColors) +  
  scale_fill_manual(values = divisionColors) +
```



ggttitle(chartTitle) +

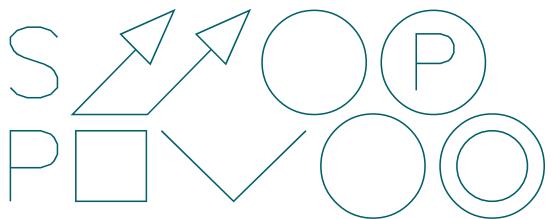


17 : SO

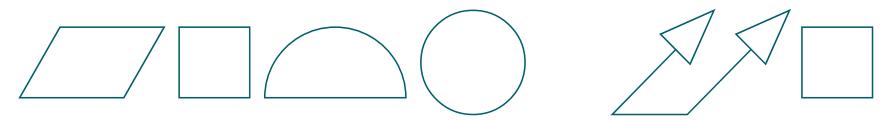


theme_few()

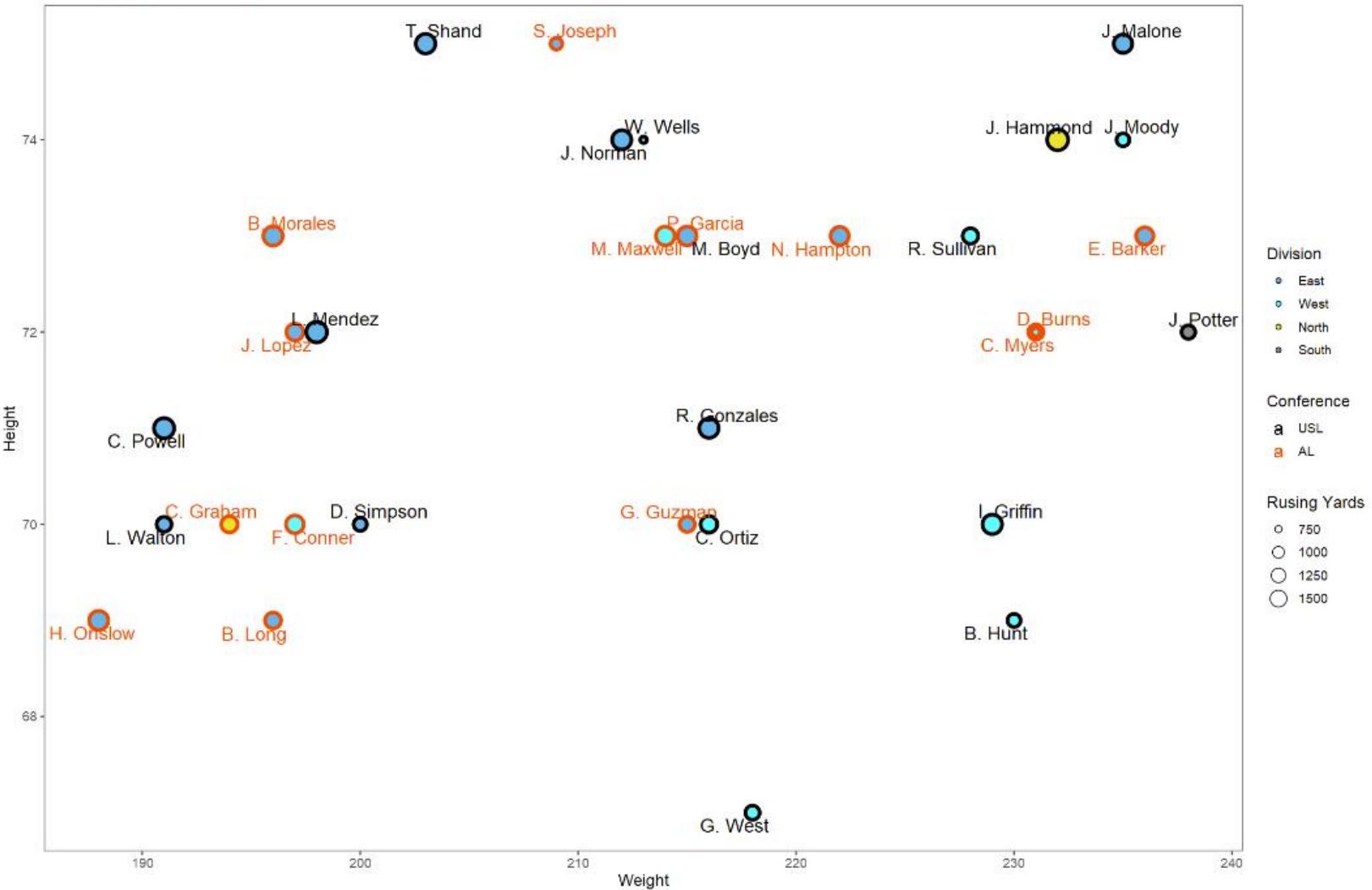
p



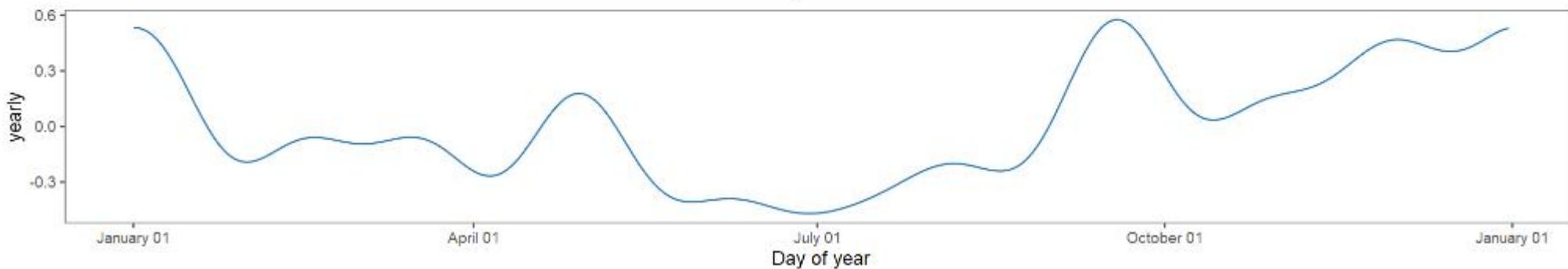
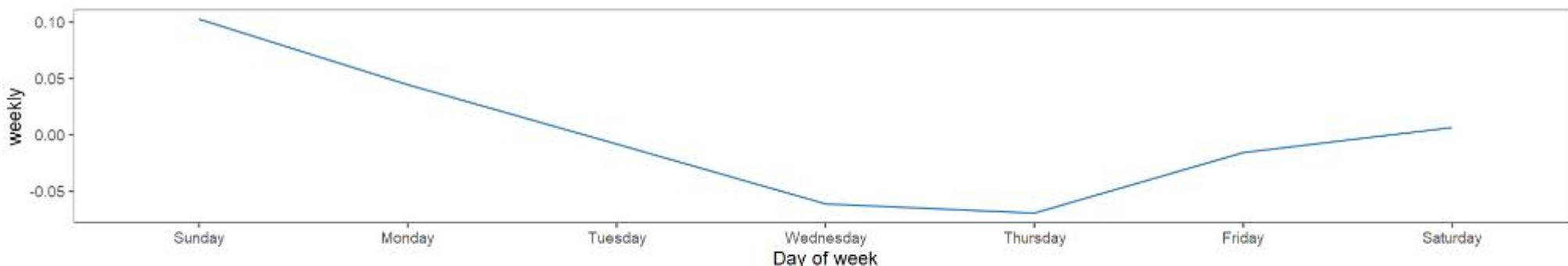
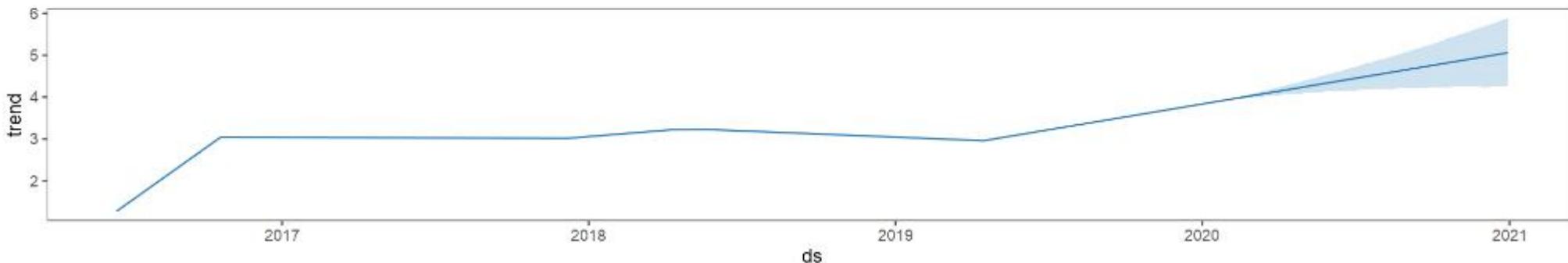
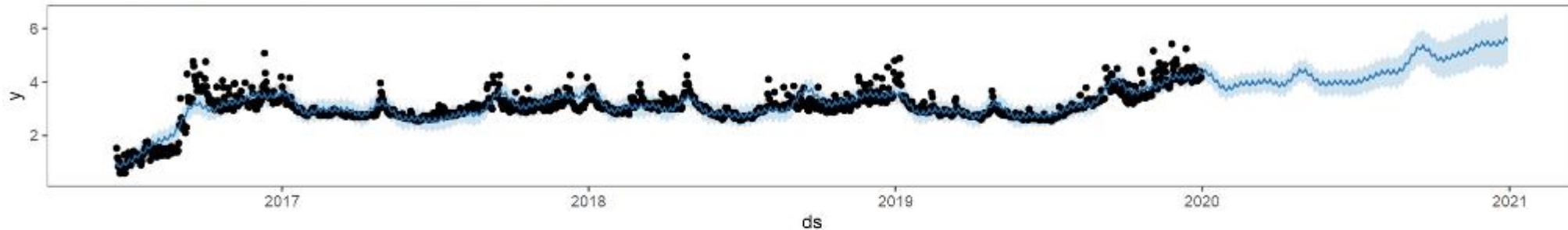
18: A | B |

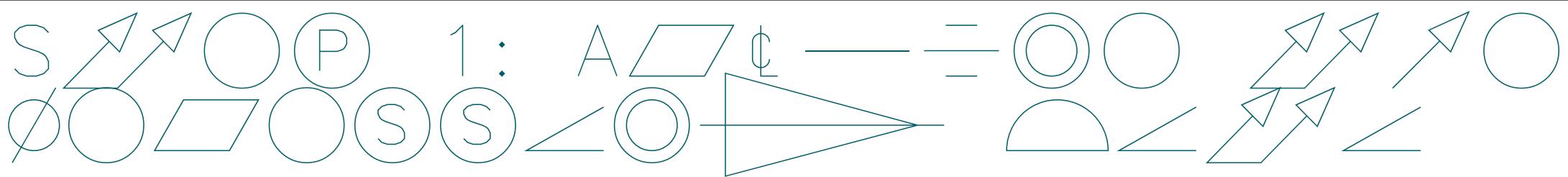


Runningback Quad Chart for 2018



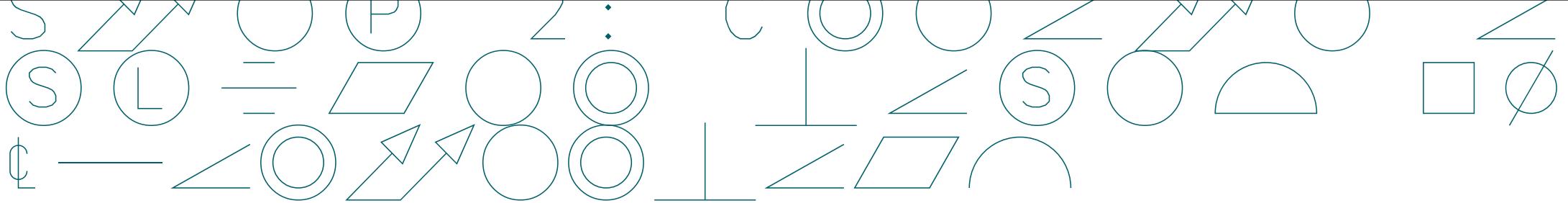
Jackson's Wikipedia page views forecast analysis





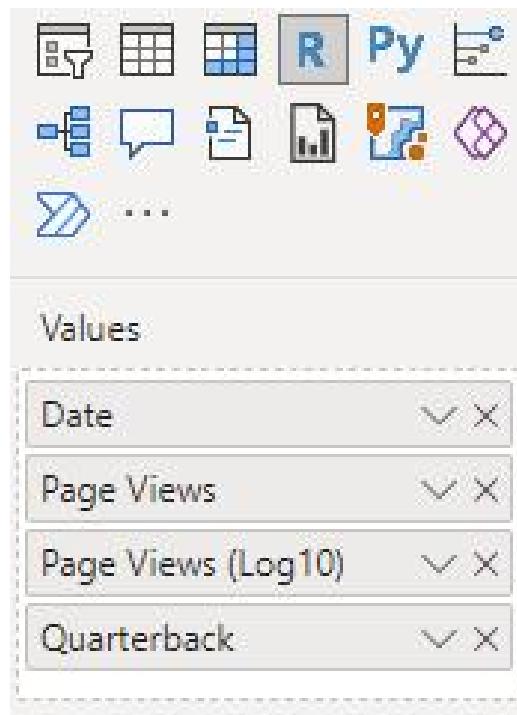
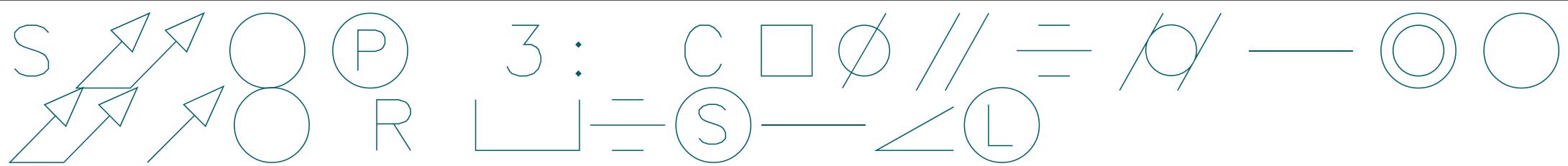
- Import “input.csv” data file.
- Hint: Date.ToString([Date],"yyyyMMdd")

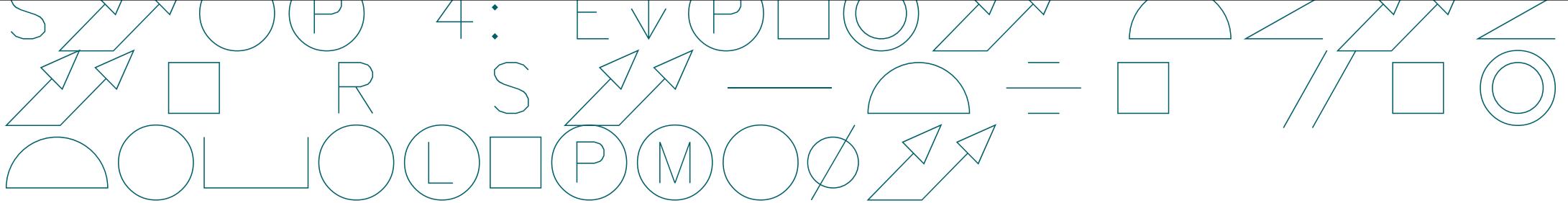
Column Name	Data Type
Date	Text
Page Views	Whole number
Quarterback	Text
Page Views (Log10)	Decimal number



Quarterback

Jackson

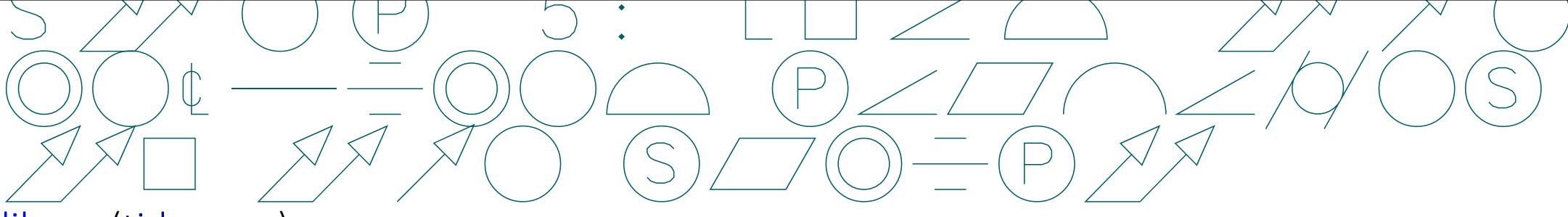




R script editor

⚠ Duplicate rows will be removed from the data.

```
1 # The following code to create a data frame and remove duplicated rows is always executed and acts as a preamble for your script:  
2  
3 # dataset <- data.frame(Date, Page Views, Page Views (Log10), Quarterback)  
4 # dataset <- unique(dataset)  
5  
6 # Paste or type your script code here:
```



library(tidyverse)

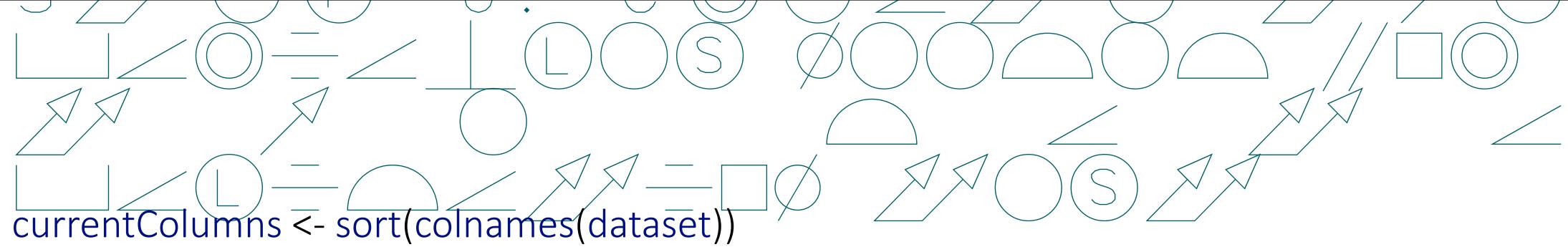
library(prophet)

library(ggthemes)

library(gridExtra)

library(lubridate)

library(rlang)

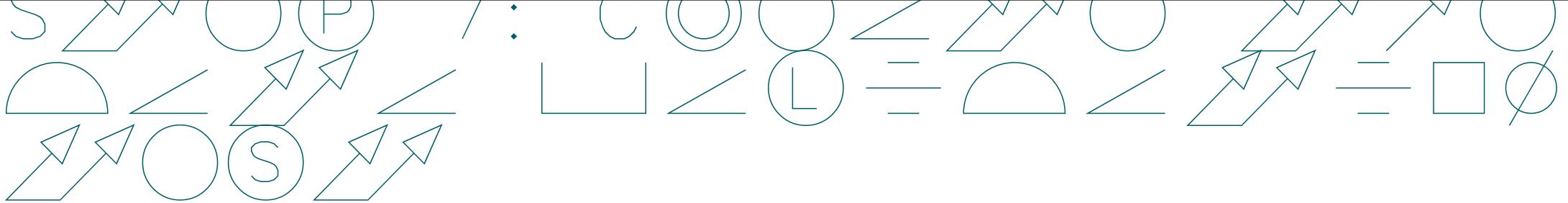


```
currentColumns <- sort(colnames(dataset))
```

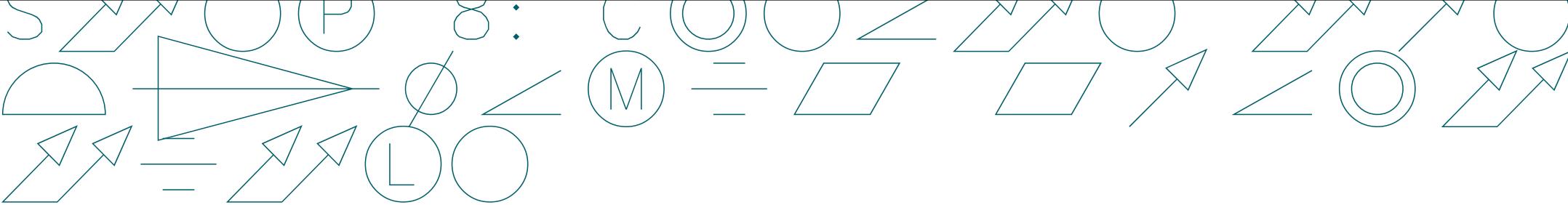
```
requiredColumns <- c("Date", "Page Views", "Page Views (Log10)", "Quarterback")
```

```
columnTest <- isTRUE(all.equal(currentColumns, requiredColumns))
```

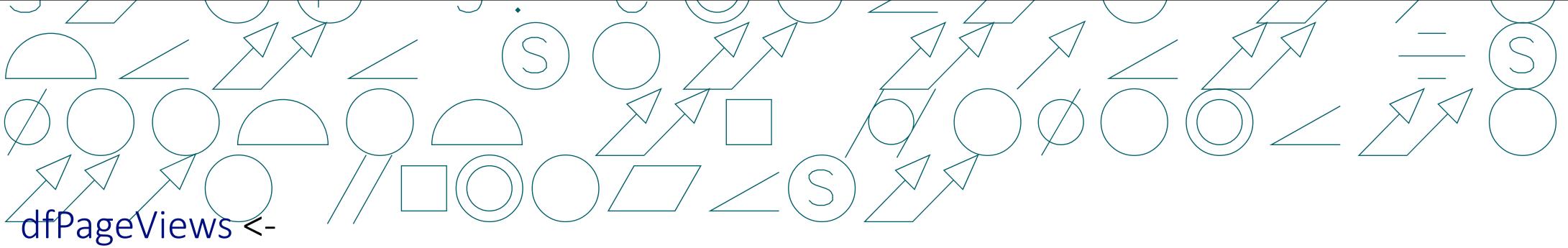
```
qb <- unique(dataset$Quarterback)
```



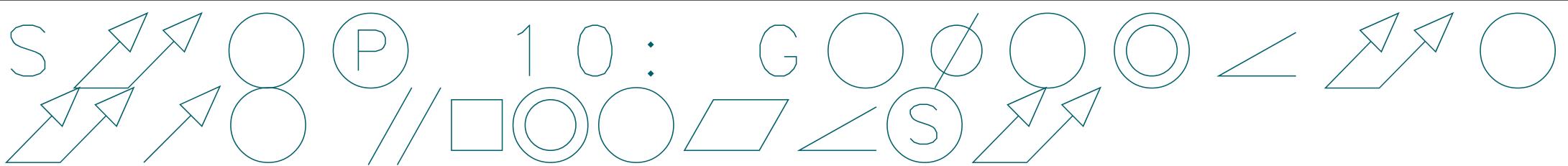
```
if (length(qb) == 1 & columnTest) {  
  <code to produce visual>  
} else {  
  
  plot.new()  
  title("The data supplied did not meet the requirements of the chart.")  
}
```



```
chartTitle <- paste0(qb, "'s Wikipedia page views forecast analysis")
```



```
dataset %>%  
  mutate(Date = ymd(Date)) %>%  
  rename(ds = Date, y = `Page Views (Log10)`)
```



```
m <- prophet(dfPageViews, yearly.seasonality=TRUE)
```

```
future <- make_future_dataframe(m, periods = 365)
```

```
forecast <- predict(m, future)
```



```
p1 <- plot(m, forecast) + ggtitle(chartTitle) + theme_few()
```

```
p <- prophet_plot_components(m, forecast)
```

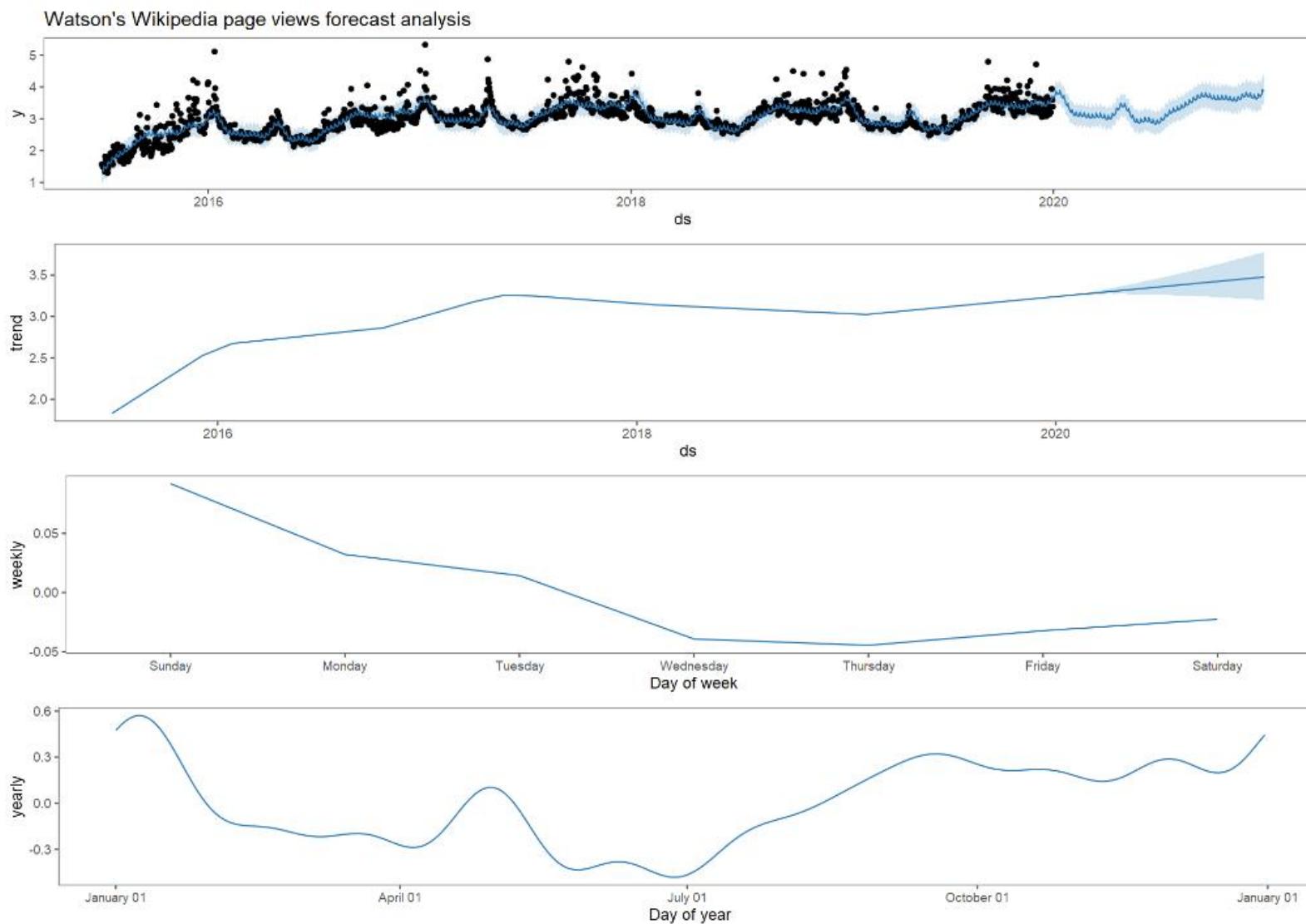
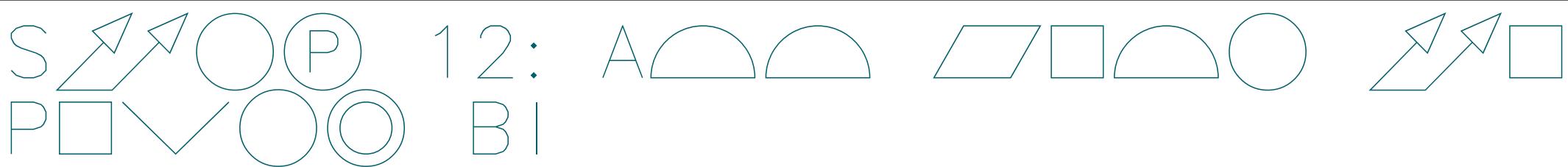
```
p2 <- p[[1]] + theme_few()
```

```
p3 <- p[[2]] + theme_few()
```

```
p4 <- p[[3]] + theme_few()
```

```
p5 <- grid.arrange(p1, p2, p3, p4, nrow =4)
```

```
p5
```



Political View

House

Overall

President

Senate

Stat

Actual GDP

GDP % Change

Year, GetPoliticalLevel, GetPoliticalLevelName, GetGDPStat and GetGDPStatName

Actual GDP Analysis

Overall view

Party Republican Democrat Tie

20,000

15,000

10,000

5,000

0

Actual GDP (in Billions)

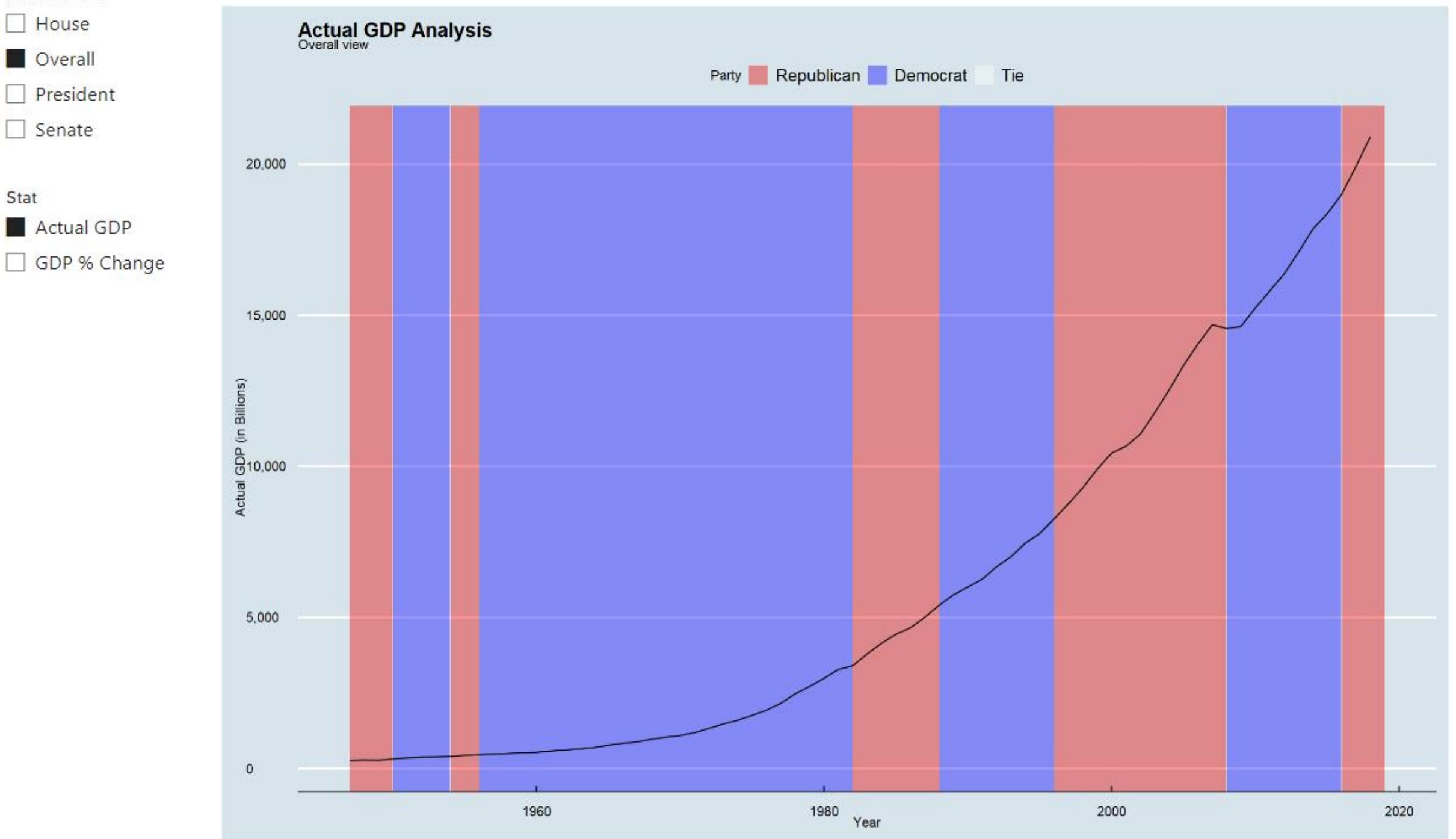
1960

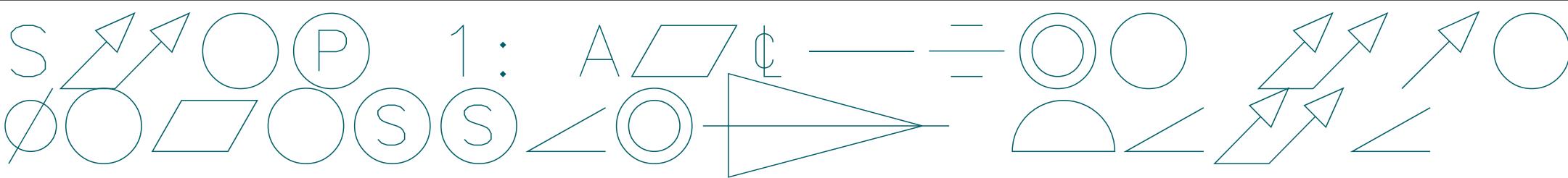
1980

2000

2020

Year

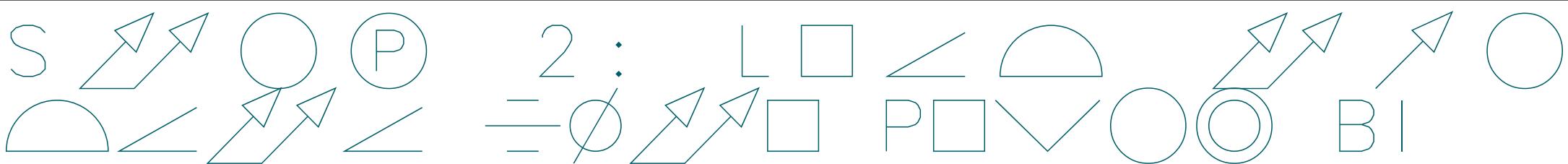




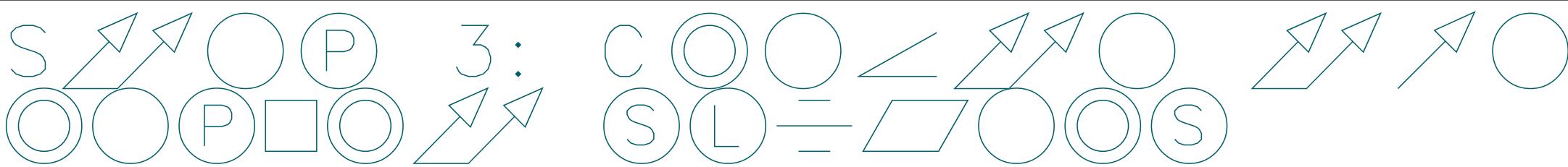
- Import “*PoliticalInfoWithGDP.csv*” data file.
- Create two new tables.
- =


```
Table.FromRows(Json.Document(Binary.Decompress(Binary.FromText("i45W8sgvLU5VitWJ
VgpOzUssgTADilKLM1NS80rAPP+y1KLEnByl2FgA", BinaryEncoding.Base64),
Compression.Deflate)), let _t = ((type text) meta [Serialized.Text = true]) in type table
[#"Political View" = _t])
```
- =


```
Table.FromRows(Json.Document(Binary.Decompress(Binary.FromText("i45WckwuKU3MUX
B3CVCK1YIWAtIKqgrOGYI56alKsbEA", BinaryEncoding.Base64), Compression.Deflate)), let
_t = ((type text) meta [Serialized.Text = true]) in type table [Stat = _t])
```



Column	Data Type
Index	Whole Number
Year	Whole Number
GDP	Decimal Number
% GDP Change	Decimal Number
Senate Majority	Text
House Majority	Text
President's Majority	Text
Overall Majority	Text



Political View

House

Overall

President

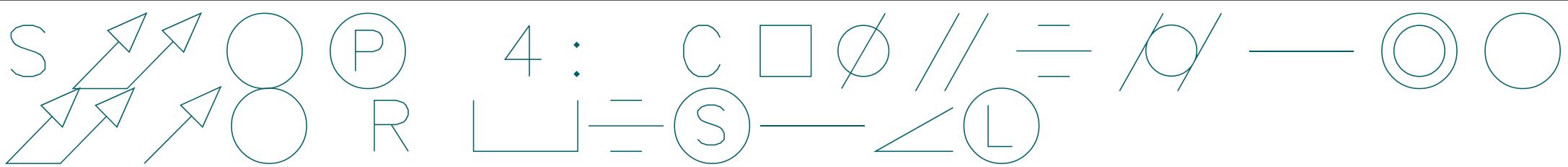
Senate

Stat



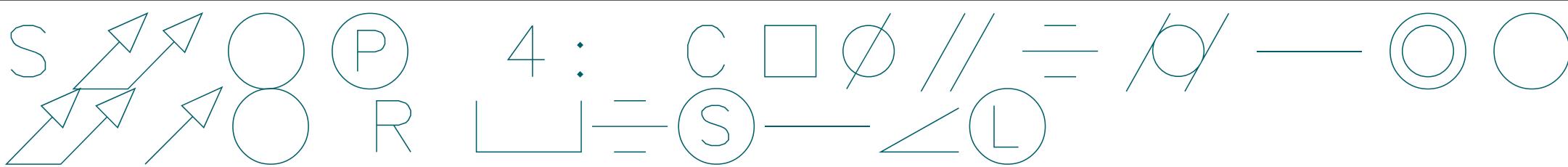
Actual GDP

GDP % Change



GetPoliticalLevel =

```
SWITCH (  
    SELECTEDVALUE ( 'Political View'[Political View] ),  
    "House", MIN ( PoliticalInfoWithGDP[House Majority] ),  
    "Overall", MIN ( PoliticalInfoWithGDP[Overall Majority] ),  
    "President", MIN ( PoliticalInfoWithGDP[President's Party] ),  
    "Senate", MIN ( PoliticalInfoWithGDP[Senate Majority] )  
)
```



GetPoliticalLevelName = SELECTEDVALUE ('Political View'[Political View])

GetGDPStat =

SWITCH (

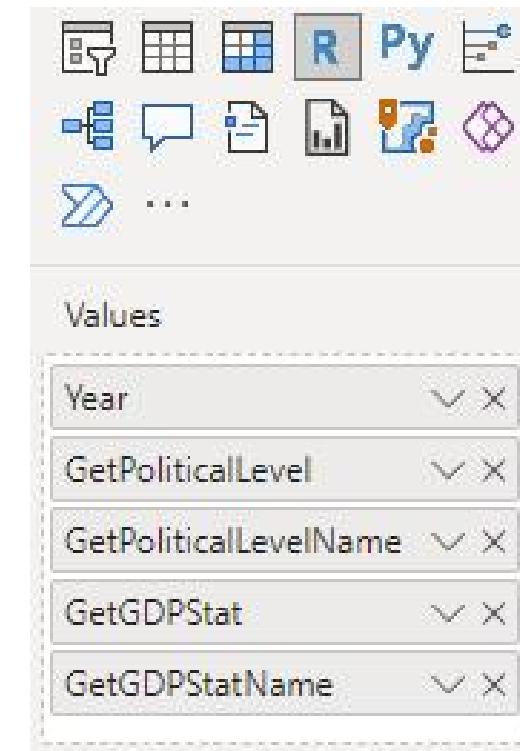
SELECTEDVALUE ('Stat'[Stat]),

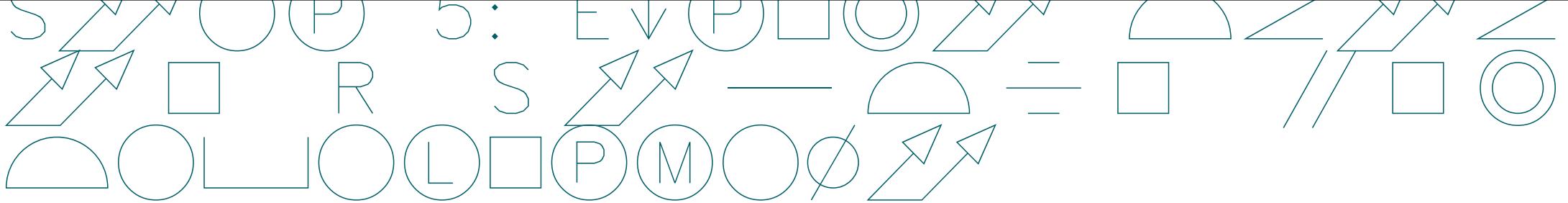
"Actual GDP", MIN (PoliticalInfoWithGDP[GDP]),

"GDP % Change", MIN (PoliticalInfoWithGDP[% GDP Change])

)

GetGDPStatName = SELECTEDVALUE ('Stat'[Stat])

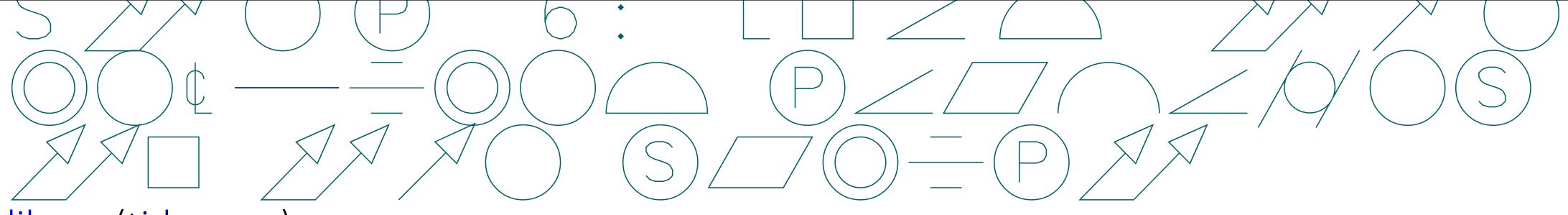




R script editor

⚠ Duplicate rows will be removed from the data. X

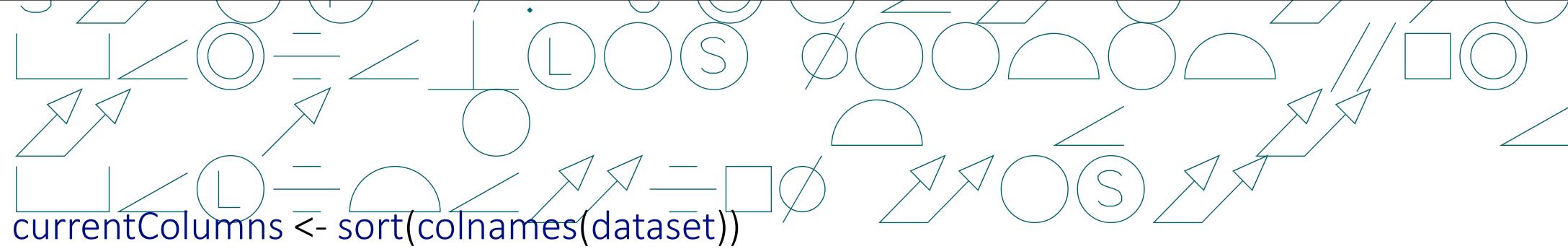
```
1 # The following code to create a dataframe and remove duplicated rows is always executed and acts as a preamble for your script:  
2  
3 # dataset <- data.frame(Year, GetPoliticalLevel, GetPoliticalLevelName, GetGDPStat, GetGDPStatName)  
4 # dataset <- unique(dataset)  
5  
6 # Paste or type your script code here:
```



`library(tidyverse)`

`library(ggthemes)`

`library(scales)`

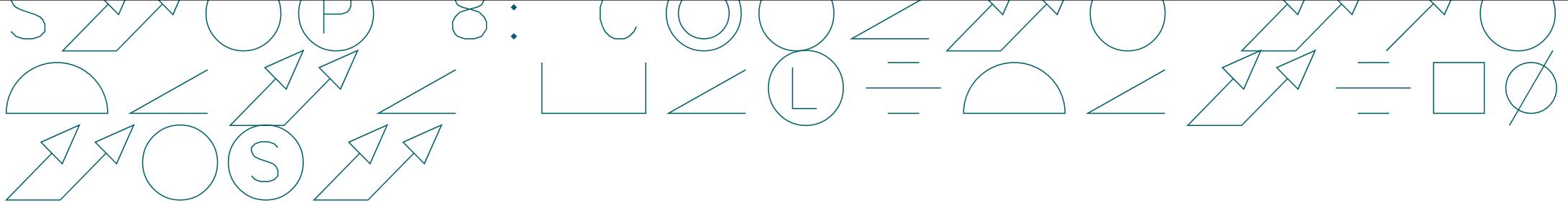


```
currentColumns <- sort(colnames(dataset))  
requiredColumns <- c("GetGDPStat", "GetGDPStatName", "GetPoliticalLevel",  
"GetPoliticalLevelName", "Year")
```

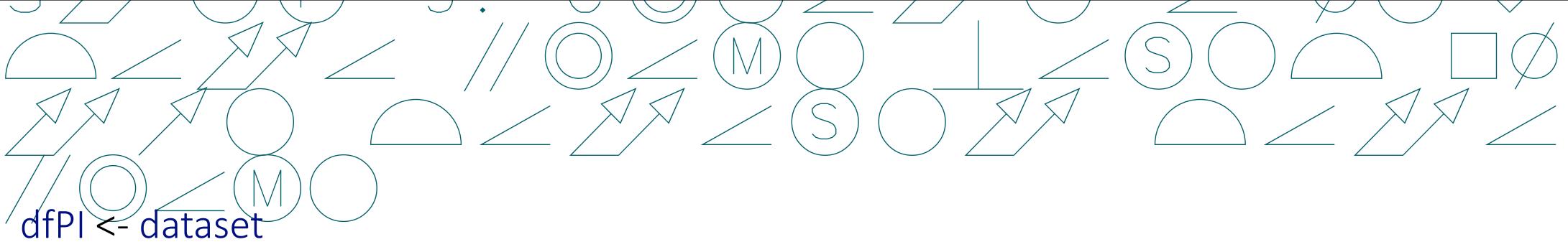
```
columnTest <- isTRUE(all.equal(currentColumns, requiredColumns))
```

```
politicalLevelName <- unique(dataset$GetPoliticalLevelName)
```

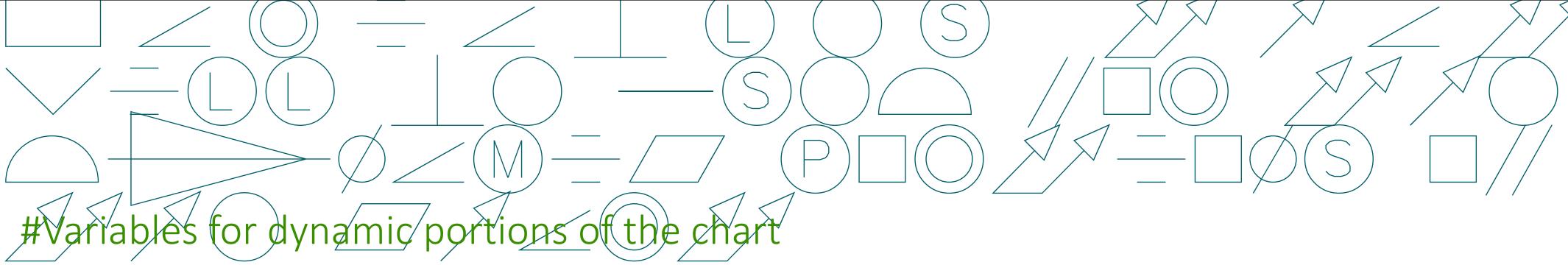
```
gdpStatName <- unique(dataset$GetGDPStatName)
```



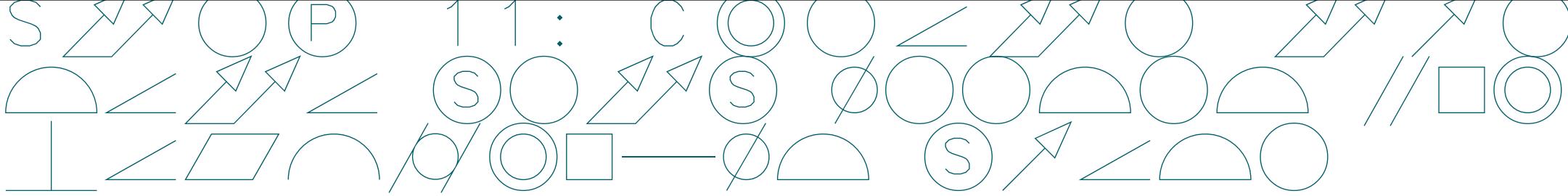
```
if(length(politicalLevelName) == 1 & length(gdpStatName) == 1 & columnTest) {  
  <code to produce visual>  
} else {  
  
  plot.new()  
  title("The data supplied did not meet the requirements of the chart.")  
  
}
```



```
dfPI <- dataset
```

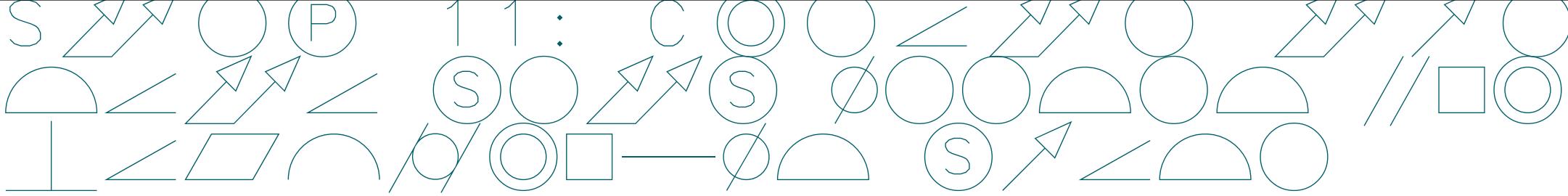


```
politicalLevelName <- unique(dfPI$GetPoliticalLevelName)  
gdpStatName <- unique(dfPI$GetGDPStatName)  
yAxisName <- paste(gdpStatName, ifelse(gdpStatName == "Actual GDP", "(in Billions)", ""),  
sep = " ")  
chartTitle <- paste(gdpStatName, "Analysis", sep = " ")  
chartSubtitle <- paste(politicalLevelName, "view", sep = " ")
```



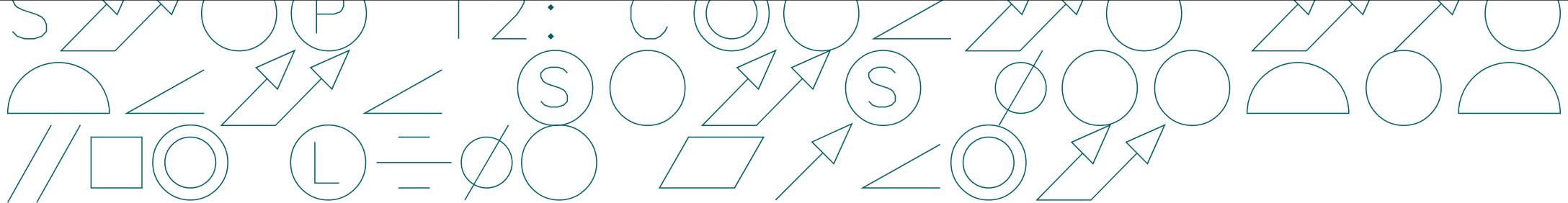
#Get the number of

```
dfPI$GetPoliticalLevel <- as.character(dfPI$GetPoliticalLevel)  
runs <- rle(dfPI$GetPoliticalLevel)  
group_id <- rep(seq_along(runs$lengths), runs$lengths)
```



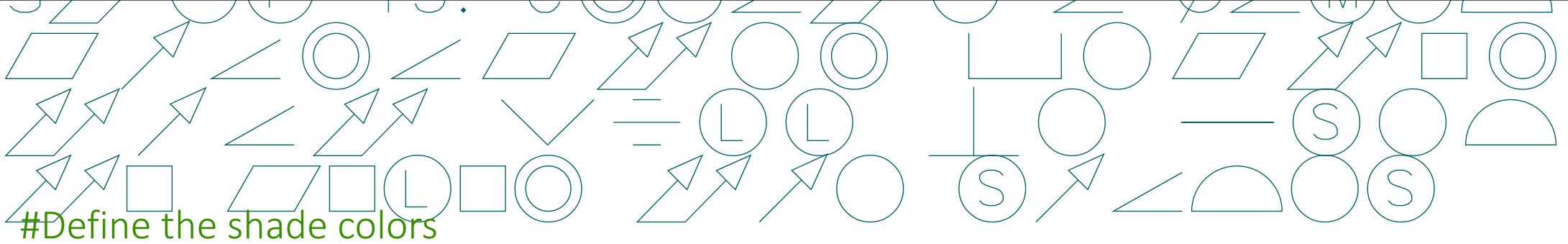
#Create the data set for shade layer

```
dfShadeInfo <-  
  cbind(dfPI, group_id) %>%  
  transmute(group_id, Year, Party = GetPoliticalLevel) %>%  
  group_by(group_id, Party) %>%  
  summarize(start = min(Year), end = max(Year)+0.99) %>%  
  ungroup()
```



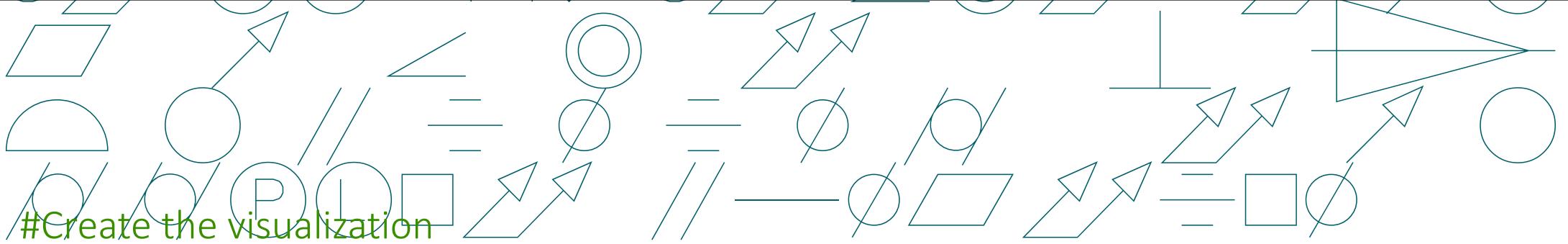
#Create the data set for line chart layer

```
dfLineChartInfo <-  
  dfPI %>%  
  transmute(Year, GetGDPStat)
```



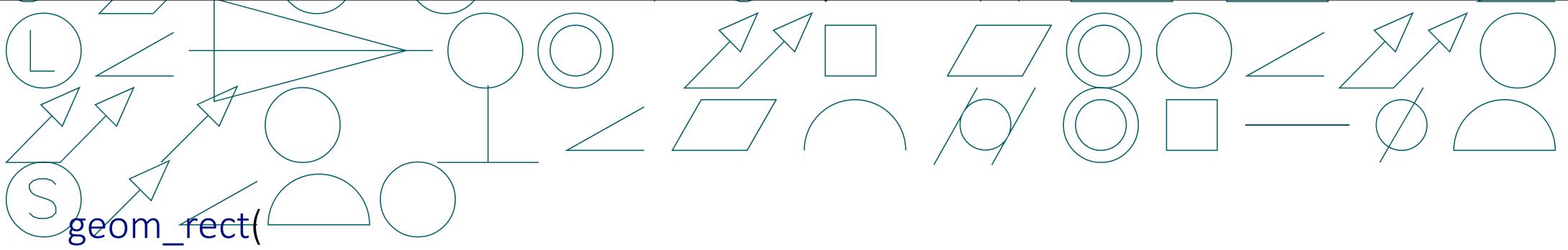
#Define the shade colors

```
partyColors = c("Republican"="red", "Democrat"="blue", "Tie" = "white")
```

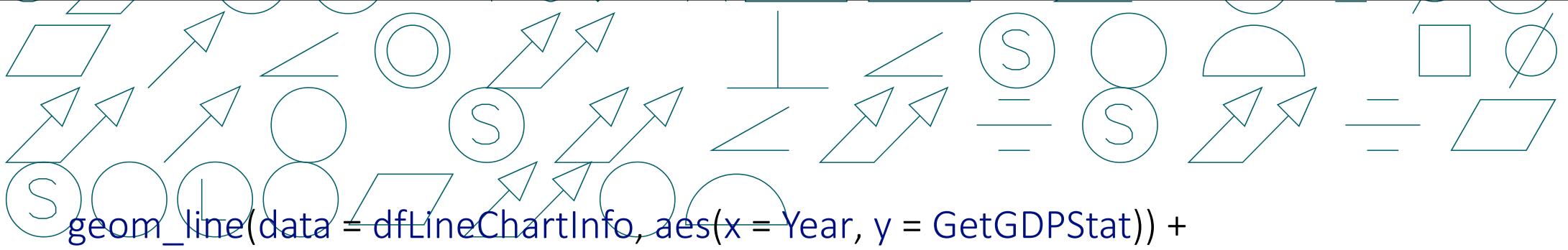


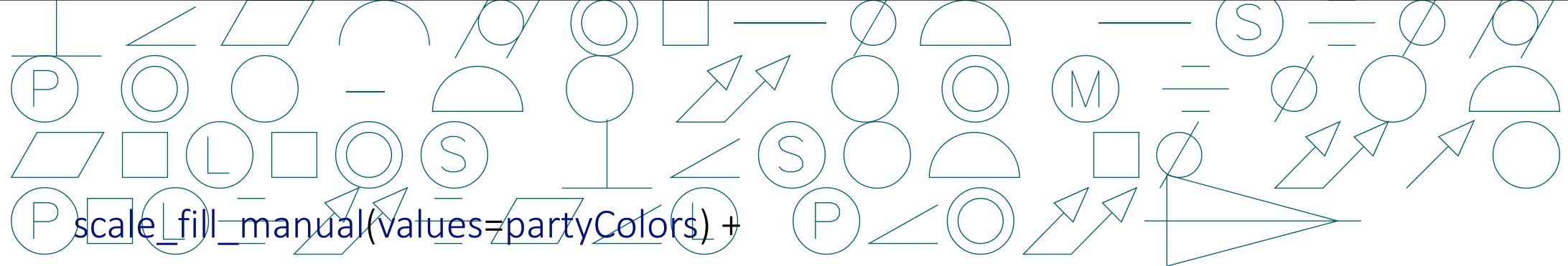
#Create the visualization

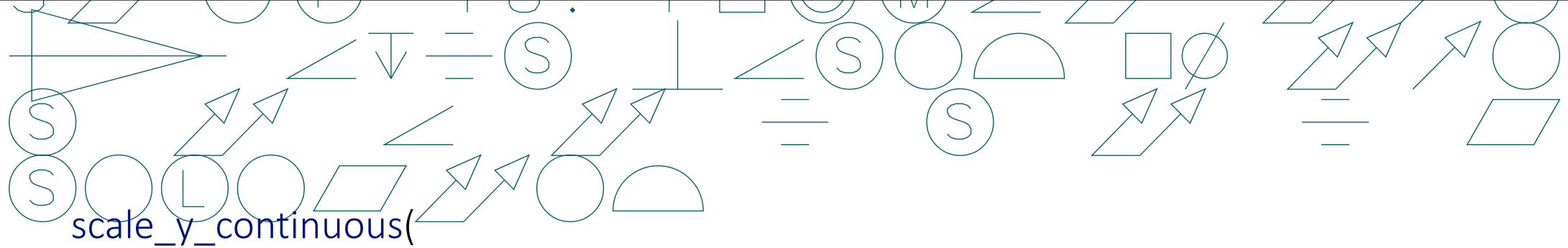
```
p <- ggplot() +
```



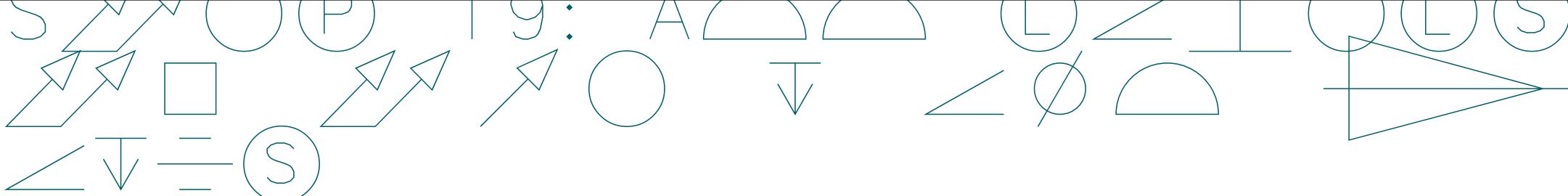
```
geom_rect(  
  data = dfShadeInfo,  
  aes(xmin = start, xmax = end, fill = Party),  
  ymin = -Inf, ymax = Inf, alpha = 0.4, color = NA  
) +
```







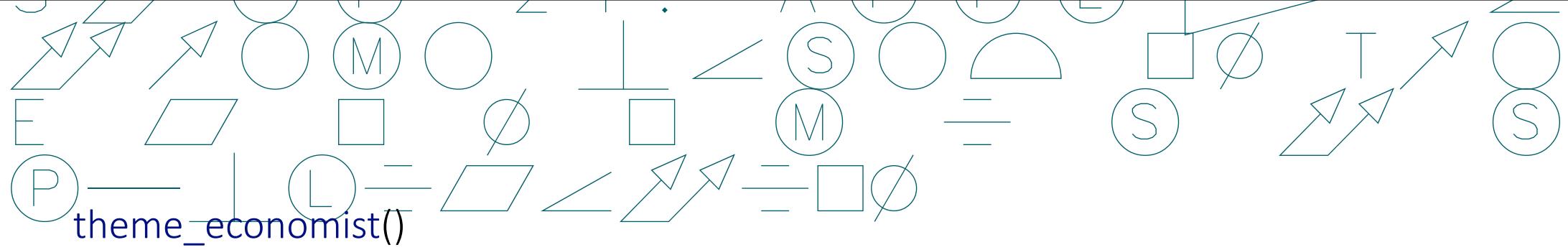
```
scale_y_continuous(  
  labels = ifelse(gdpStatName == "Actual GDP", comma_format(), percent_format())  
) +
```



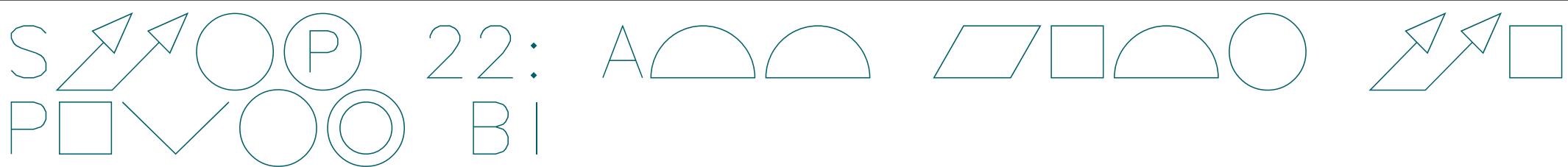
ylab(yAxisName) +

xlab("Year") +

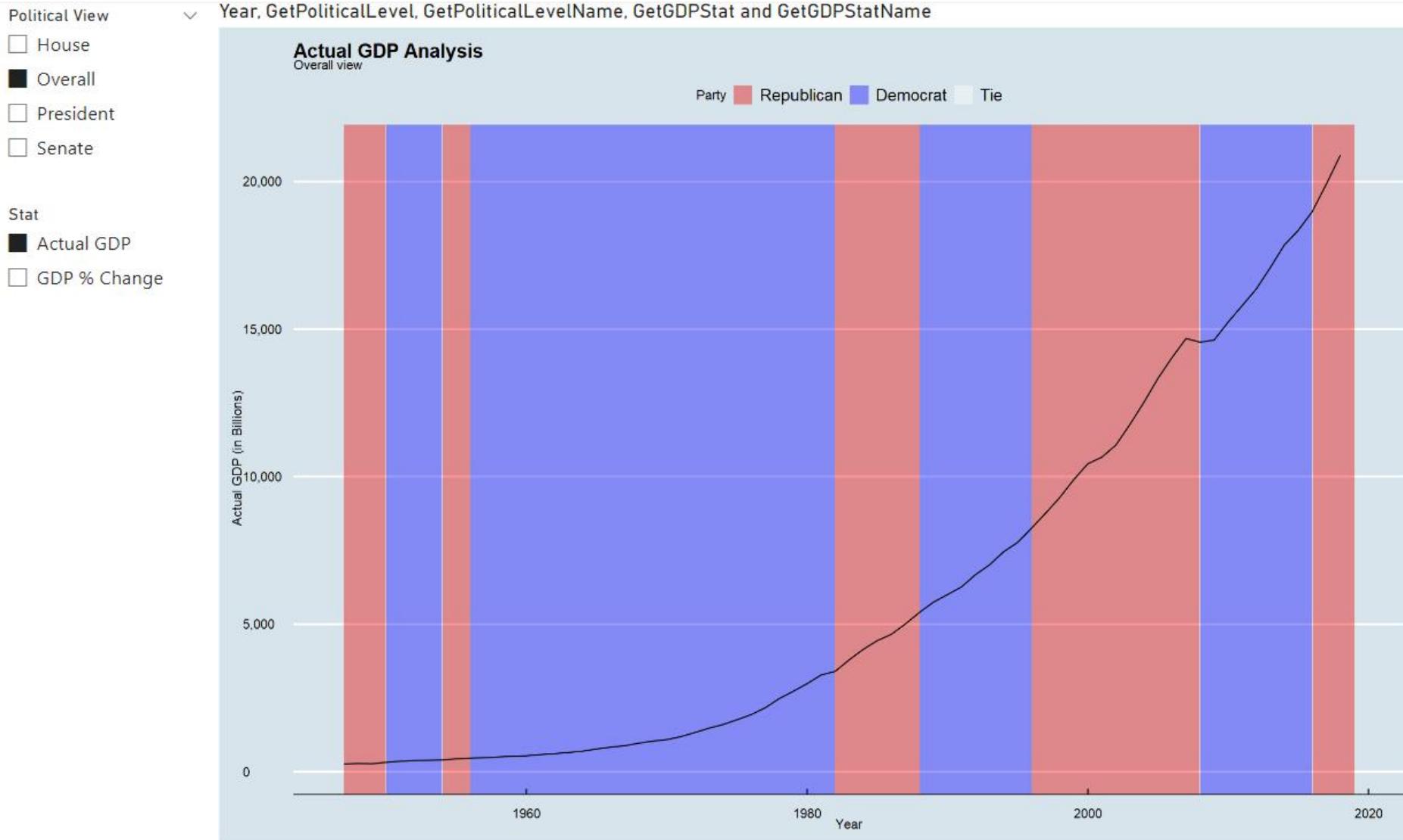
```
ggttitle(chartTitle, subtitle = chartSubtitle) +
```



p

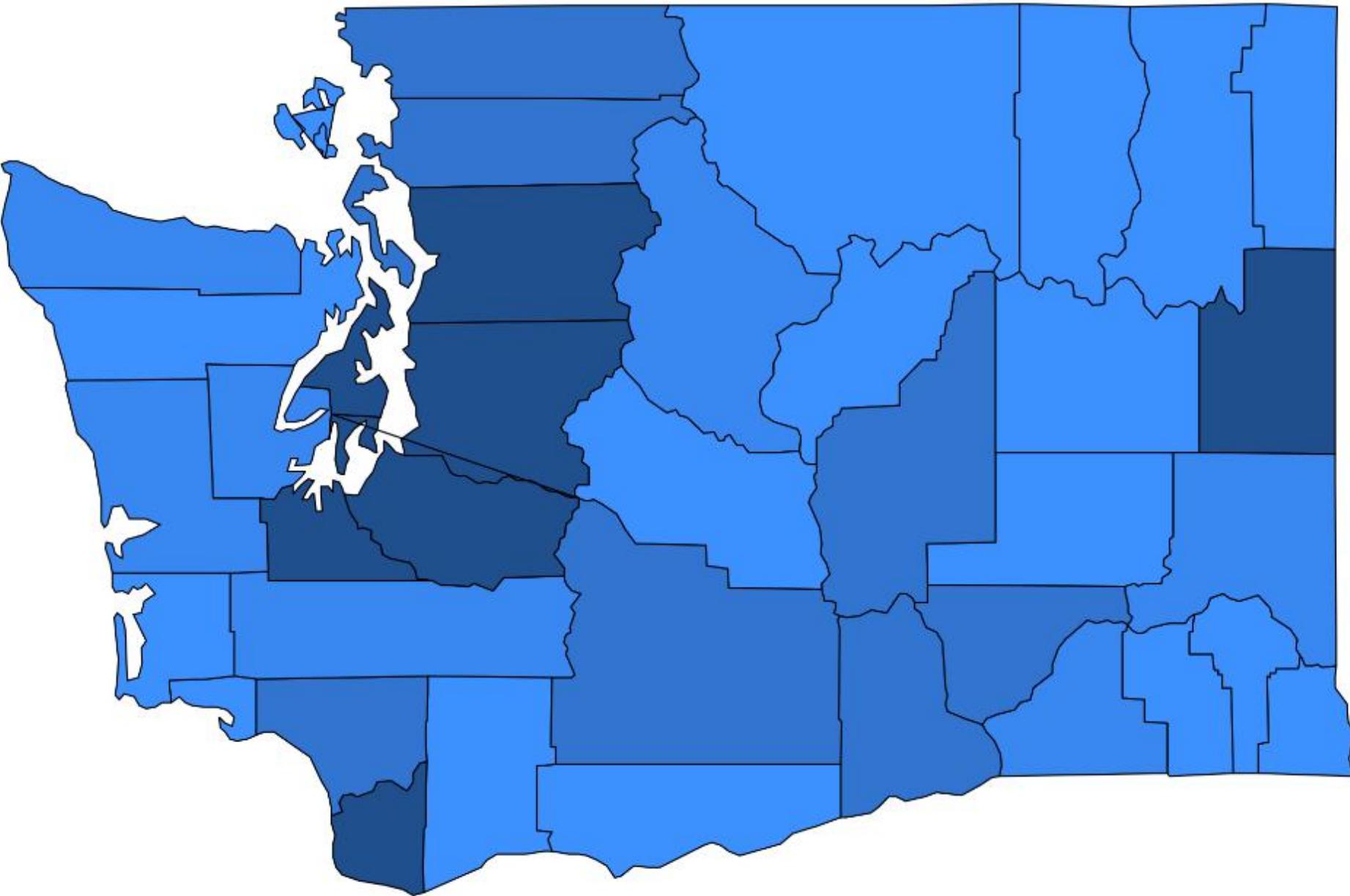


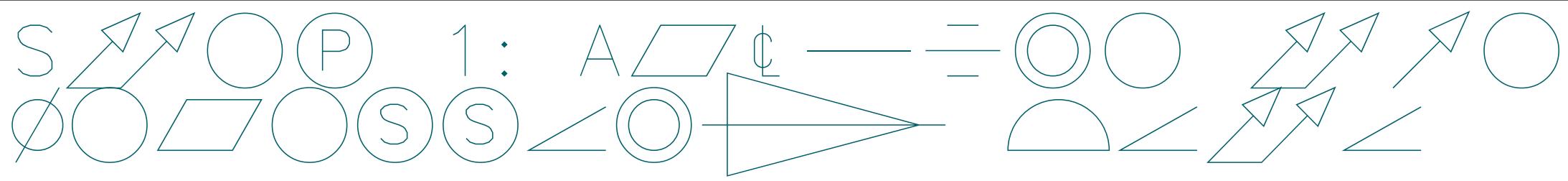
22: A
B |



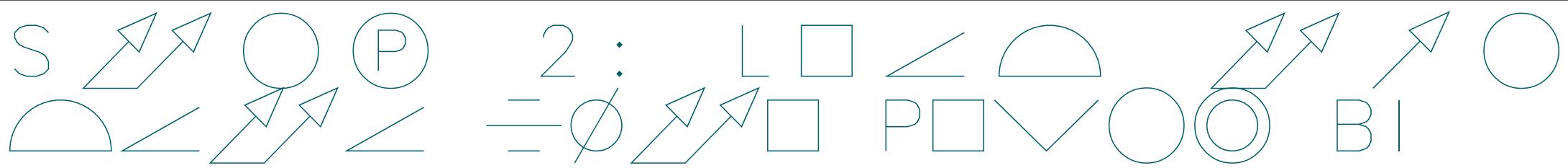
WA

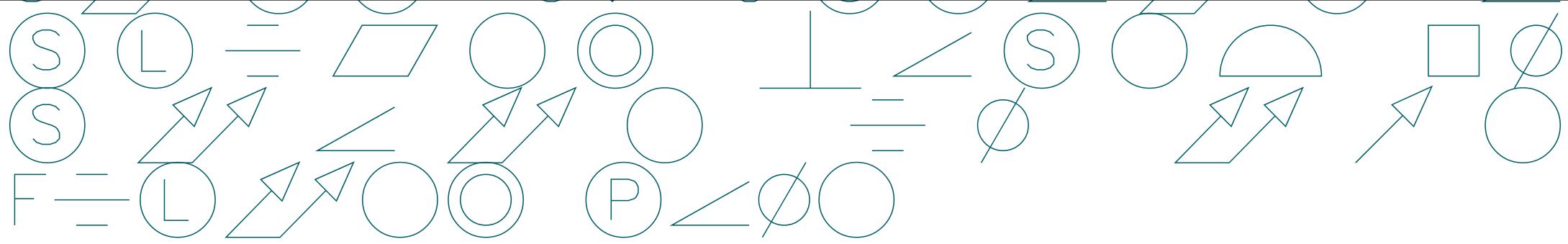
Washington's County Population Analysis
(the darker shades the higher the population)





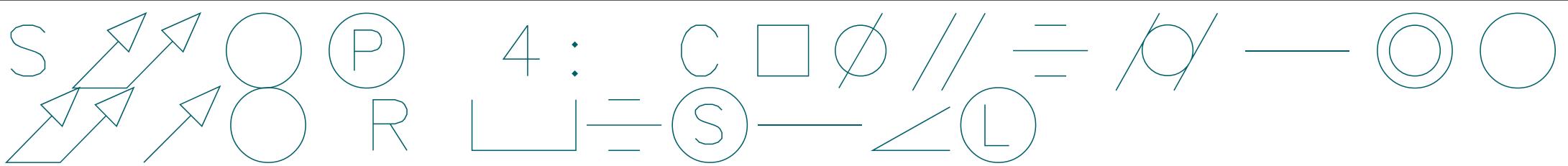
- Import “*chartdata.csv*” data file.





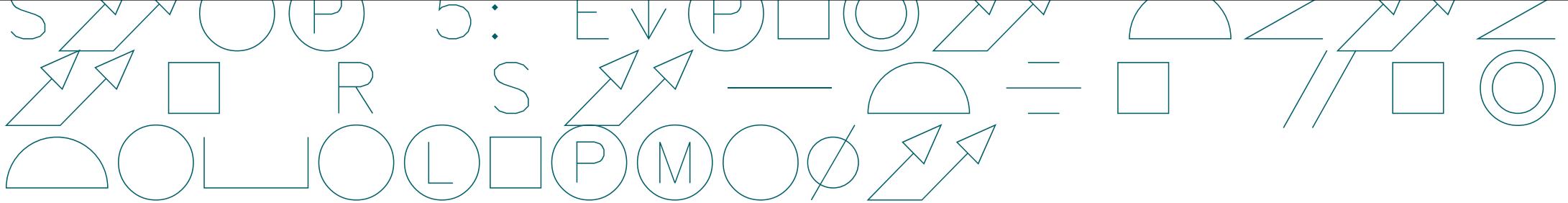
State Abbreviation

CA



A screenshot of a data visualization application's interface. At the top, there is a toolbar with various icons: a magnifying glass, a grid, a blue square labeled 'R', a blue square labeled 'Py', a line chart, a speech bubble, a document, a bar chart, a map, and a diamond. Below the toolbar is a section titled 'Values' which contains a list of variables:

Value	Control
Index	▼ X
lat	▼ X
long	▼ X
County	▼ X
Total Population	▼ X
State	▼ X



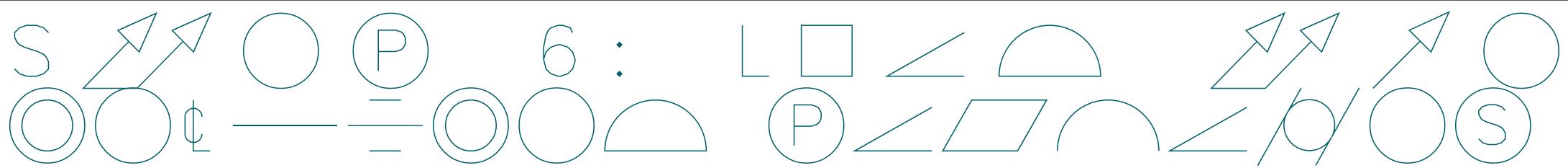
R script editor

⚠ Duplicate rows will be removed from the data.



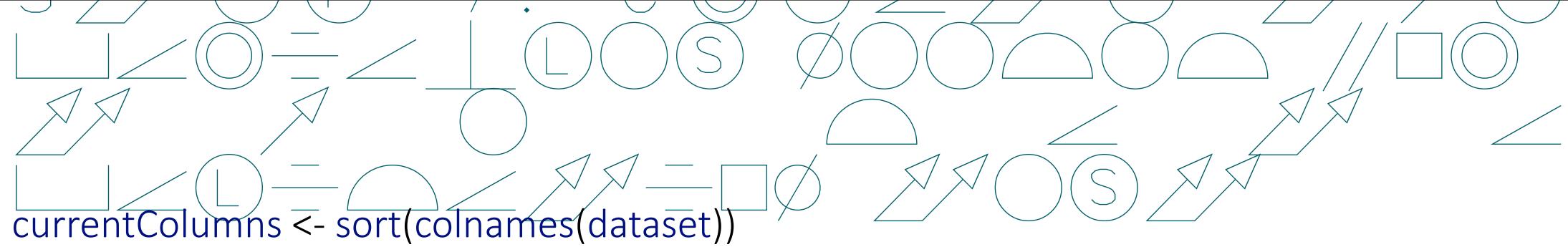
X

```
1 # The following code to create a dataframe and remove duplicated rows is always executed and acts as a preamble for your script:  
2  
3 # dataset <- data.frame(Index, lat, long, County, Total Population, State)  
4 # dataset <- unique(dataset)  
5  
6 # Paste or type your script code here:
```



```
library(tidyverse)
```

```
library(ggthemes)
```

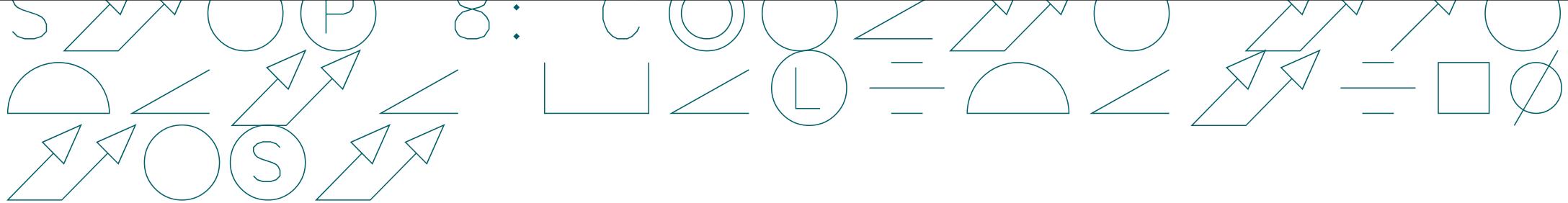


```
currentColumns <- sort(colnames(dataset))
```

```
requiredColumns <- c("County", "Index", "lat", "long", "State", "Total Population")
```

```
columnTest <- isTRUE(all.equal(currentColumns, requiredColumns))
```

```
state <- str_to_title(unique(dataset$State))
```



```
if (length(state) == 1 & columnTest)
```

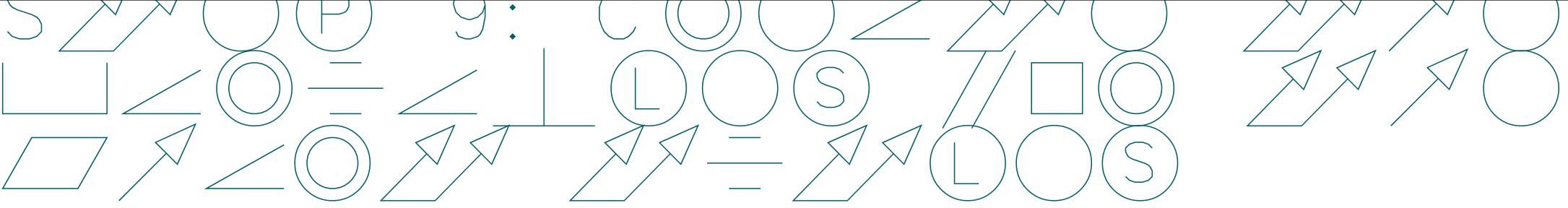
<code to build visual>

```
 } else {
```

`plot.new()`

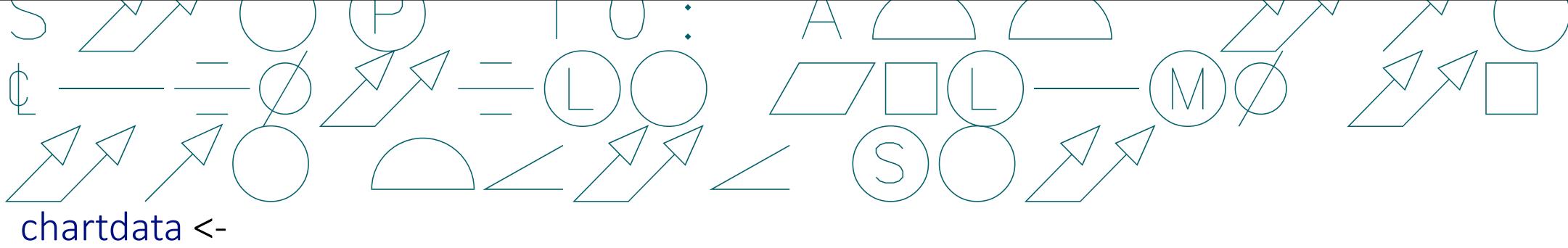
```
title("The data supplied did not meet the requirements of the chart.")
```

}



```
chartTitle <- paste0(state, "'s County Population Analysis")
```

```
subTitle <- "(the darker shades the higher the population)"
```



chartdata <-

dataset %>%

```
mutate(quintile = factor(ntile(`Total Population`, 5)))
```



<http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf>

```
quintileColors <-
```

```
c(
```

```
  "1" = "dodgerblue",
```

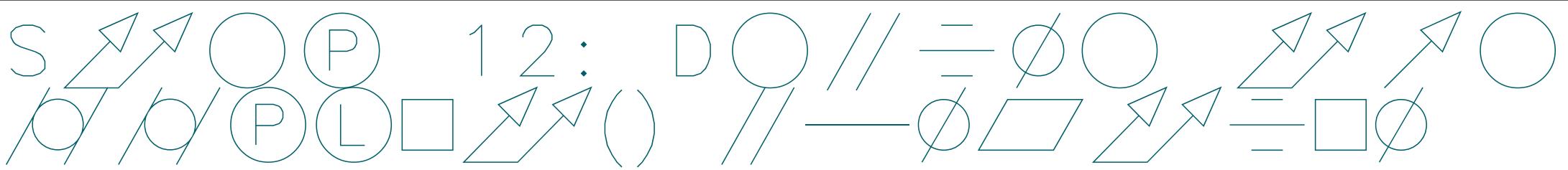
```
  "2" = "dodgerblue1",
```

```
  "3" = "dodgerblue2",
```

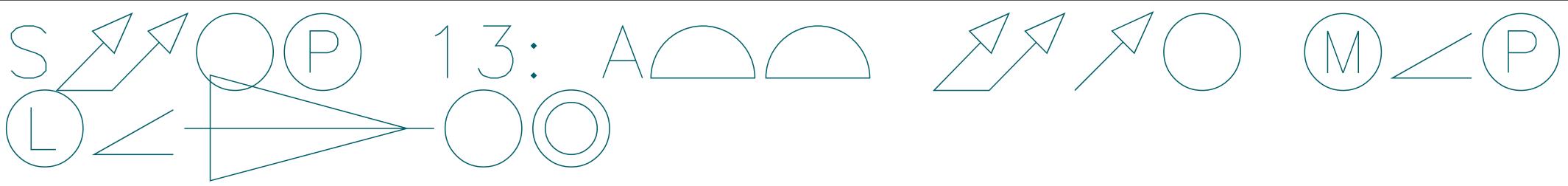
```
  "4" = "dodgerblue3",
```

```
  "5" = "dodgerblue4"
```

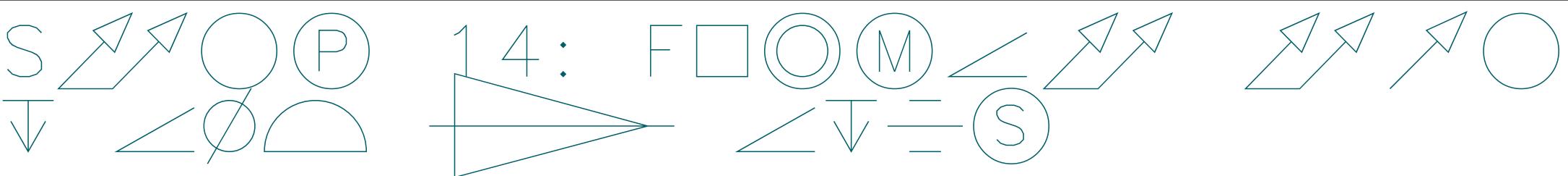
```
)
```



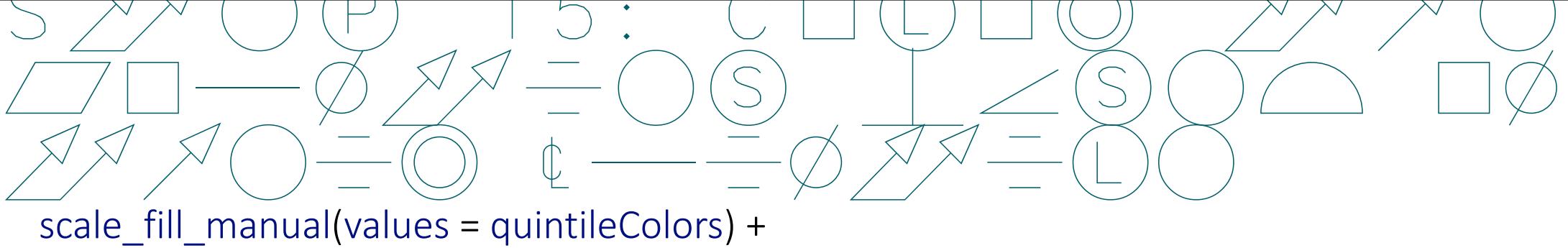
```
ggplot(chartdata, aes(long, lat, group = County, fill = quintile)) +
```

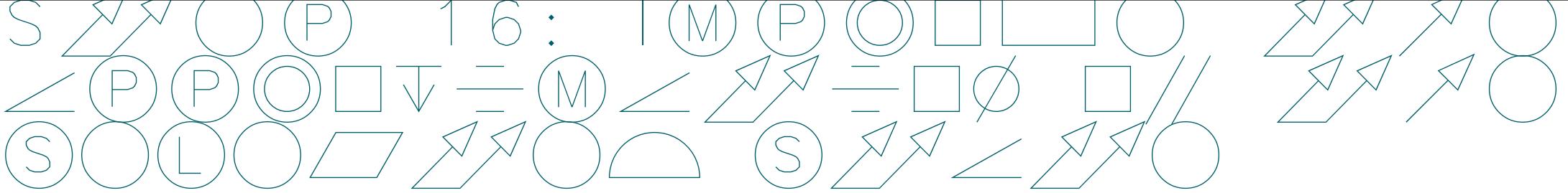


```
geom_polygon(show.legend = FALSE, color = "black") +
```



```
scale_x_continuous(name = NULL, labels = NULL, breaks = NULL) +  
scale_y_continuous(name = NULL, labels = NULL, breaks = NULL) +
```



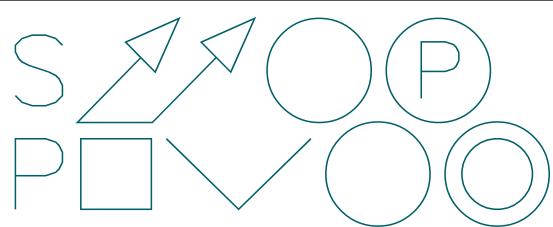


coord_quickmap() +

```
ggttitle(chartTitle, subtitle = subTitle) +
```



theme_map()



19: A
B |

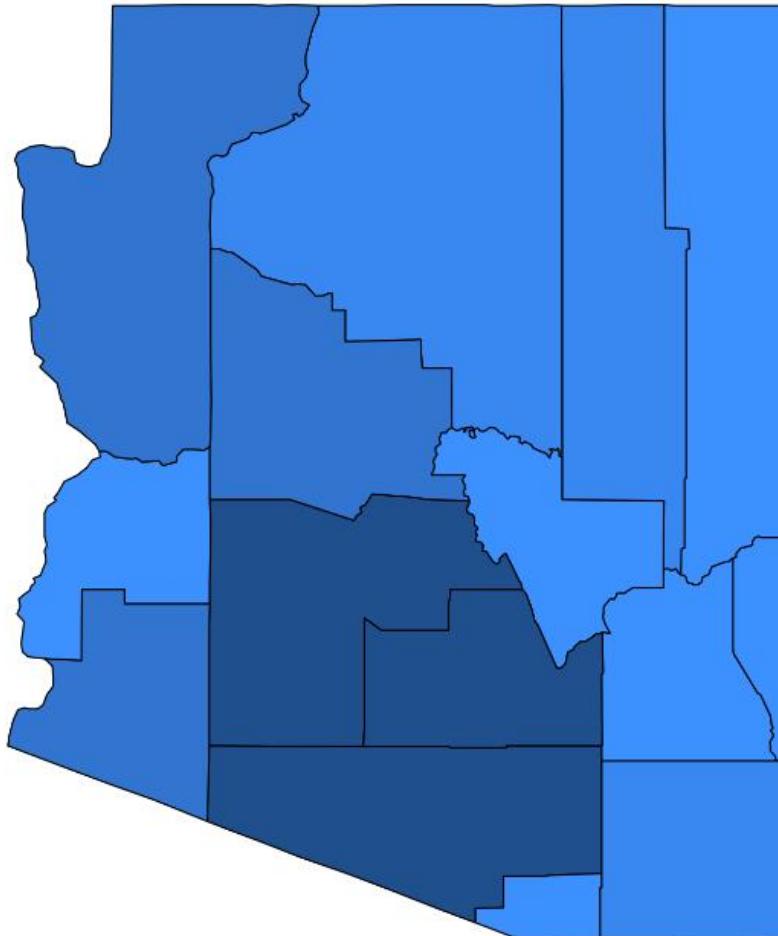


State Abbreviation

AZ

Index, lat, long, County, Total Population and State

Arizona's County Population Analysis
(the darker shades the higher the population)



Game Type

Player, Rebounds and Total Points

 away home

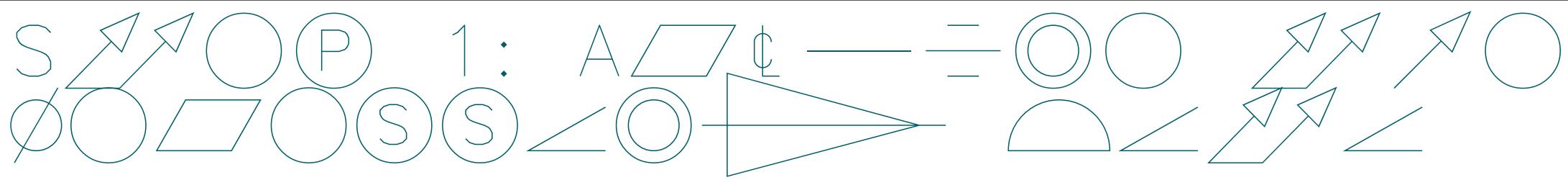
Period

 1 2 3 4 5

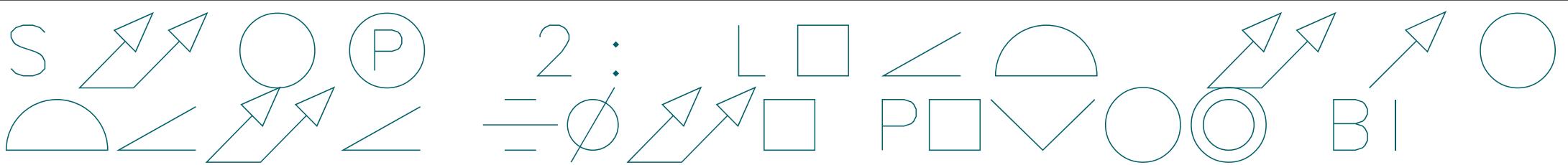
LA Lakers player comparison

(2008 - 2009 Season)

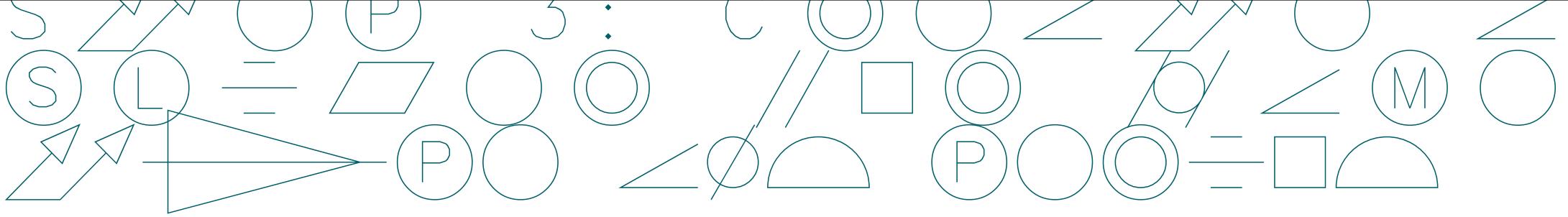


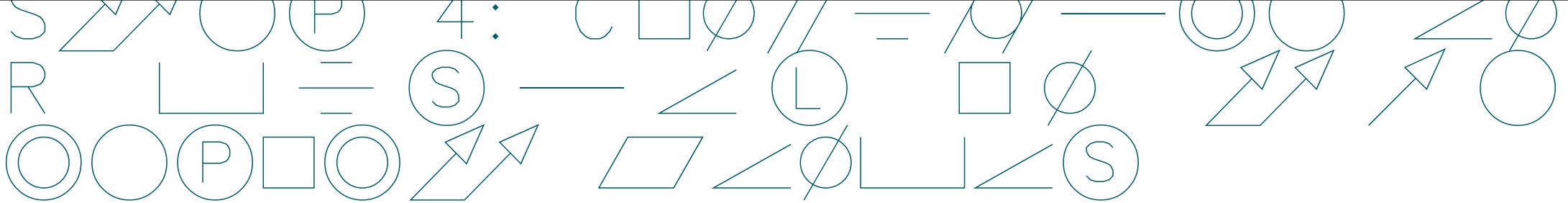


- Import “chartData.csv” data file.



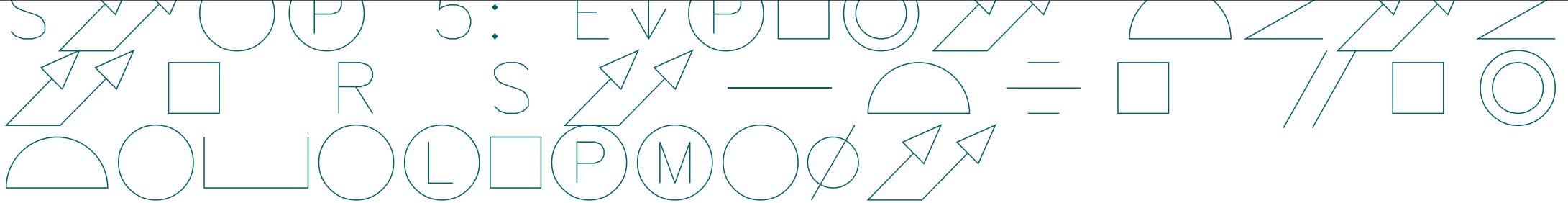
Column Name	Data Type
Game Type	Text
Period	Whole Number
Player	Text
Total Points	Whole Number
Rebounds	Whole Number





UI elements for a data visualization tool:

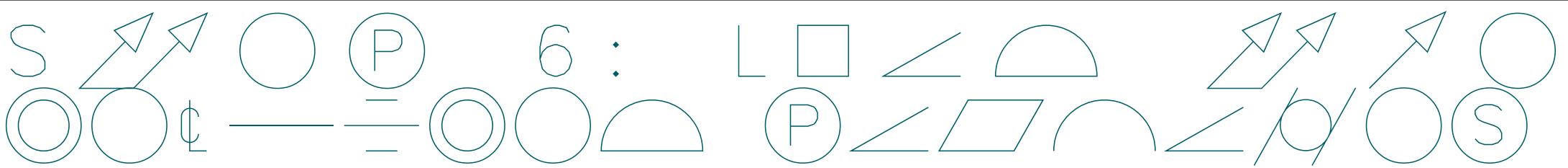
- Icon bar:
 - Eye icon
 - Table icon
 - Grid icon
 - Blue square icon (highlighted)
 - R icon
 - Py icon
 - Line chart icon
- Icon bar:
 - Database icon
 - Message icon
 - File icon
 - Bar chart icon
 - Blue square icon (highlighted)
 - Flame icon
 - Wrench icon
- More options icon: three horizontal dots
- Values section:
 - Player
 - Rebounds
 - Total Points



R script editor

⚠ Duplicate rows will be removed from the data. X

```
1 # The following code to create a dataframe and remove duplicated rows is always executed and acts as a preamble for your script:  
2  
3 # dataset <- data.frame(Player, Rebounds, Total Points)  
4 # dataset <- unique(dataset)  
5  
6 # Paste or type your script code here:
```



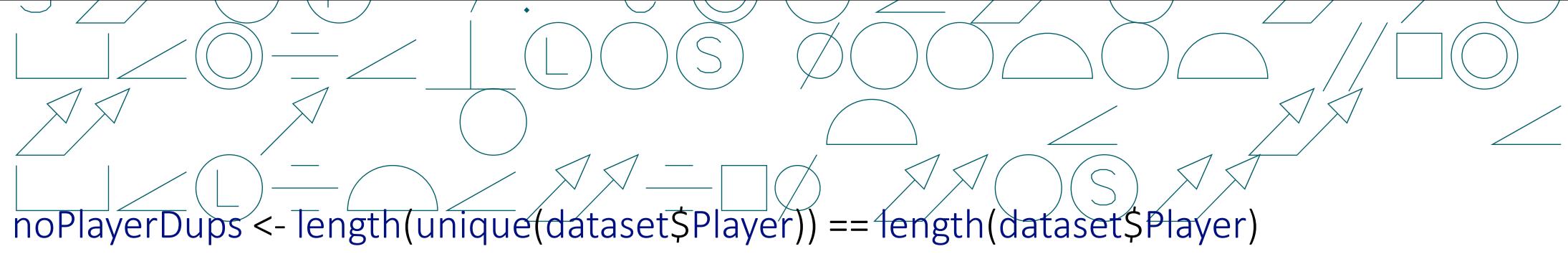
```
library(tidyverse)
```

```
library(ggrepel)
```

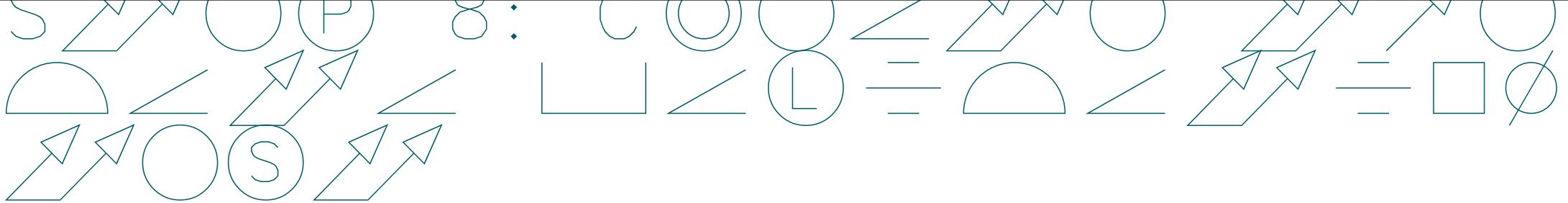
```
library(ggthemes)
```

```
library(scales)
```

```
library(stringi)
```



```
noPlayerDups <- length(unique(dataset$Player)) == length(dataset$Player)  
currentColumns <- sort(colnames(dataset))  
requiredColumns <- c("Player", "Rebounds", "Total Points")  
columnTest <- isTRUE(all.equal(currentColumns, requiredColumns))
```



```
if (noPlayerDups & columnTest) {  
    <code to produce visual>  
} else {  
  
    plot.new()  
    title("The data supplied did not meet the requirements of the chart.")  
}
```

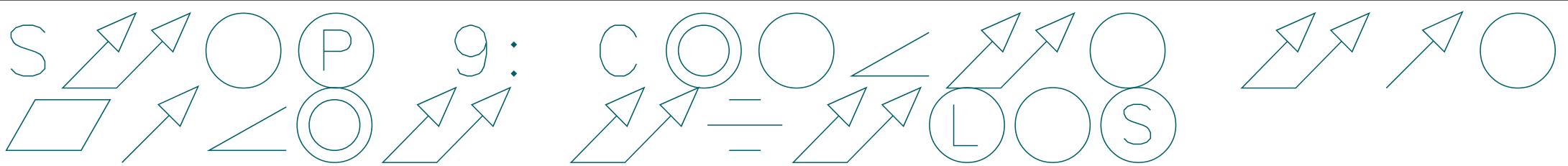
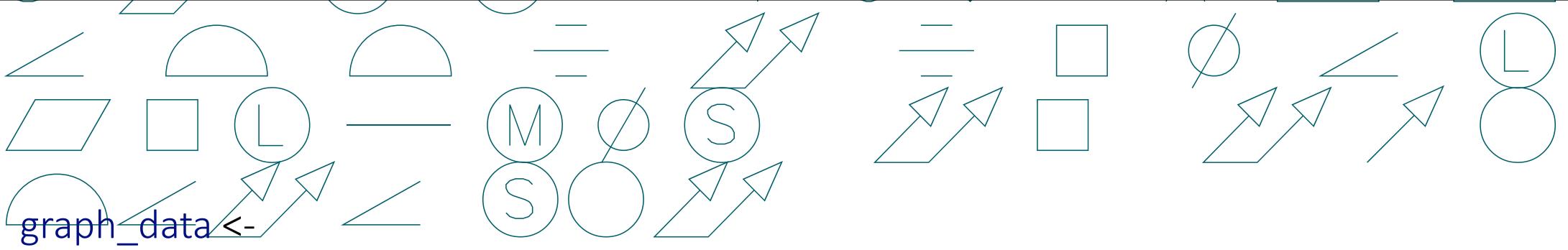


chart.title <- "LA Lakers player comparison"

chartsubtitle <- "(2008 - 2009 Season)"

chart_source <- "Source: lubridate package"



```
graph_data <-  
dataset %>%  
mutate(  
  Scaled.Rebounds =  
    round(rescale(Rebounds, to = c(-10, 10)), 1)  
  ,Scaled.TotalPoints =  
    round(rescale(`Total Points`, to = c(-10, 10)), 1)  
)
```

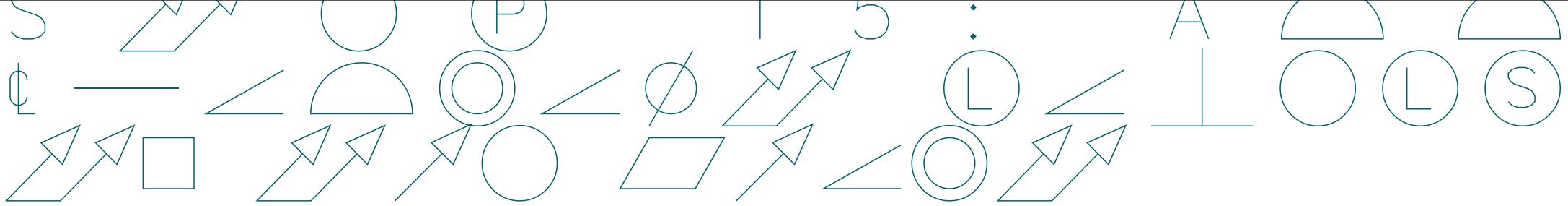
A network graph visualization featuring various nodes represented by different shapes: circles, squares, and triangles. Some nodes contain letters such as 'P', 'I', and 'O'. Directed edges connect the nodes, forming a complex web of relationships. The background is white, and the graph is rendered in a light blue color scheme.

```
p <- ggplot(graph_data, aes(x = Scaled.Rebounds, y = Scaled.TotalPoints)) +
```

geom_point() +

The diagram consists of several clusters of nodes connected by arrows. One cluster on the left contains nodes P, L, M, and various symbols (square, circle, parentheses). Another cluster in the center contains nodes P, L, and symbols. A third cluster on the right contains nodes M, S, and symbols. A large triangle points to a node labeled S.

```
geom_label_repel(aes(label = Player), size = 4, show.legend = FALSE) +
```

```
# quad labels
```

```
annotate("text", x = -5, y = -11, label = "Average", alpha = 0.2, size = 6) +  
annotate("text", x = -5, y = 11, label = "High Scorer", alpha = 0.2, size = 6) +  
annotate("text", x = 5, y = -11, label = "High Rebounder", alpha = 0.2, size = 6) +  
annotate("text", x = 5, y = 11, label = "Beast Mode", alpha = 0.2, size = 6) +
```

```
# Squares for quad labels
```

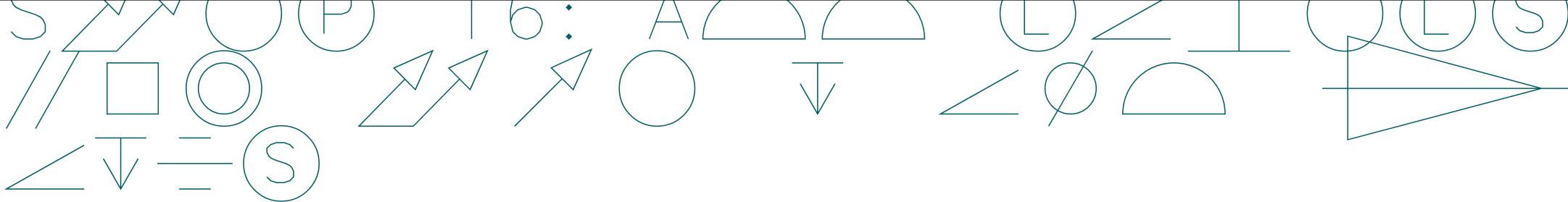
```
annotate("rect", xmin = -3.5, xmax = -6.5, ymin = -11.5, ymax = -10.5, alpha = .2) +  
annotate("rect", xmin = -3.5, xmax = -6.5, ymin = 10.5, ymax = 11.5, alpha = .2) +  
annotate("rect", xmin = 3.5, xmax = 6.5, ymin = -11.5, ymax = -10.5, alpha = .2) +  
annotate("rect", xmin = 3.5, xmax = 6.5, ymin = 10.5, ymax = 11.5, alpha = .2) +
```

```
# Shade lower left quadrant
```

```
annotate("rect", xmin = -Inf, xmax = 0.0, ymin = -Inf, ymax = 0, alpha = 0.1, fill = "lightskyblue") +
```

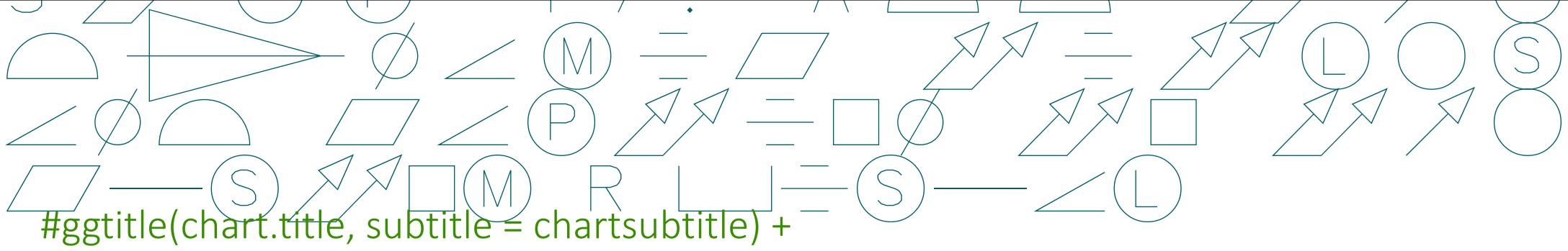
```
# Shade upper right quadrant
```

```
annotate("rect", xmin = 0.0, xmax = Inf, ymin = 0.0, ymax = Inf, alpha = 0.1, fill = "lightskyblue") +
```

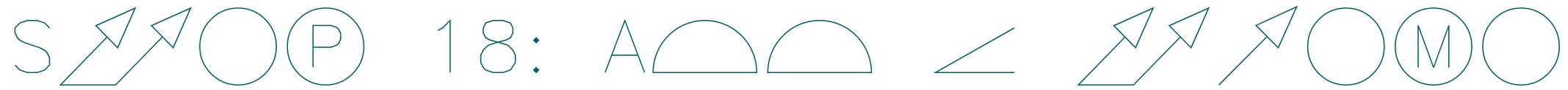


Titles

```
xlab(bquote("Rebounding" ~ symbol('\256'))) +  
ylab(bquote("Scoring" ~ symbol('\256'))) +
```

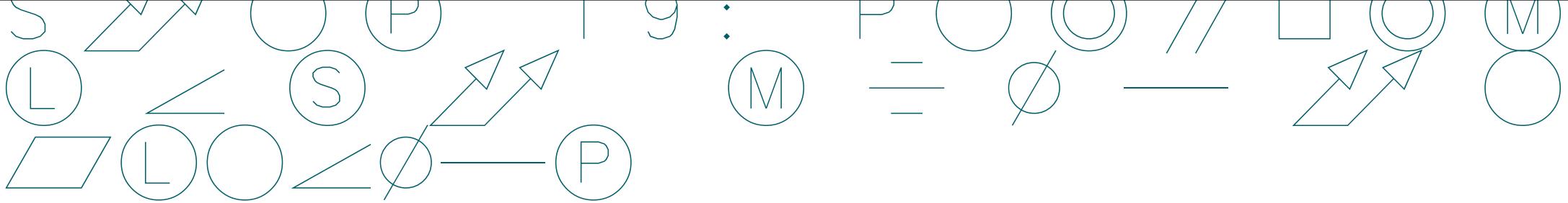


```
labs(title = chart.title, subtitle = chart.subtitle, caption = chart_source) +
```



Prettying things up

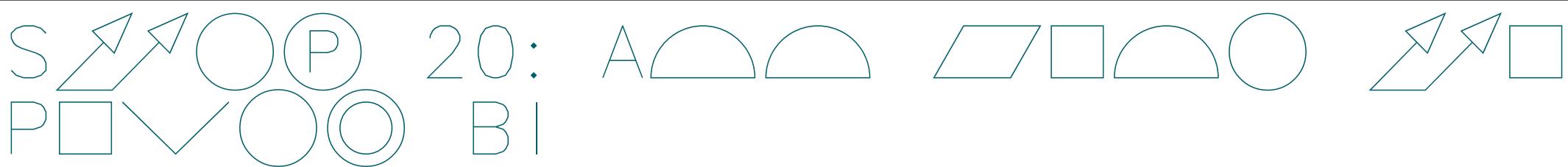
```
theme_tufte() +
```



theme(

```
  plot.title = element_text(hjust = 0.5, size = 25)
  , plot.subtitle = element_text(hjust = 0.5, size = 20)
  , panel.border = element_rect(colour = "black", size = 2, fill = NA)
  , axis.title.x = element_text(hjust = 0.1, size = 18)
  , axis.title.y = element_text(hjust = 0.1, size = 18)
) +
scale_x_continuous(labels = NULL, breaks = NULL) +
scale_y_continuous(labels = NULL, breaks = NULL)
```

p



Game Type ▾ Player, Rebounds and Total Points

away

home

LA Lakers player comparison (2008 - 2009 Season)

Period

1

2

3

4

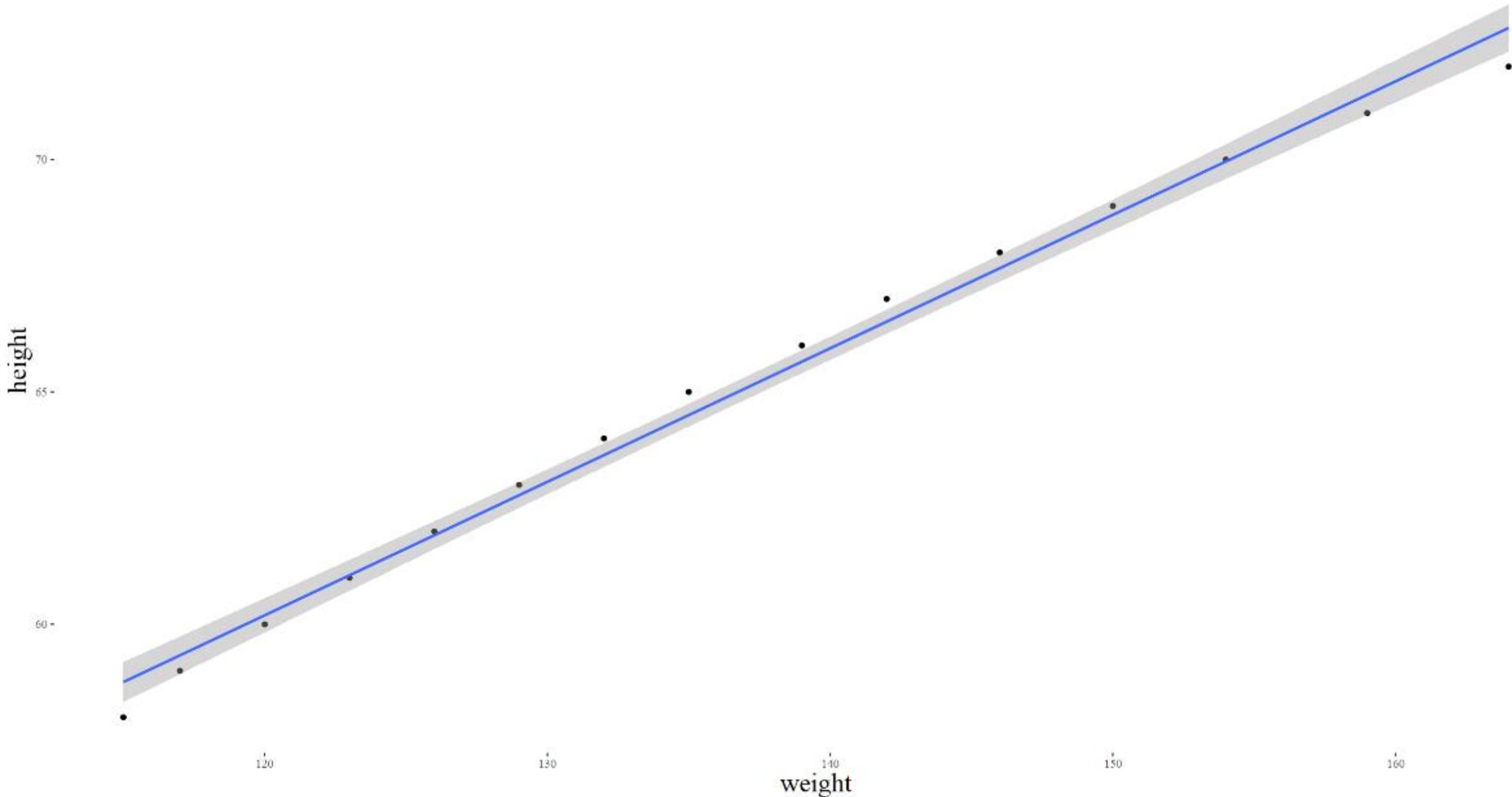
5

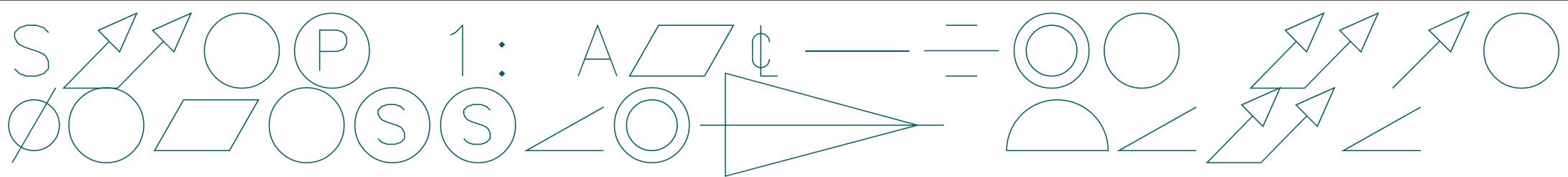


Source: Lubridate package

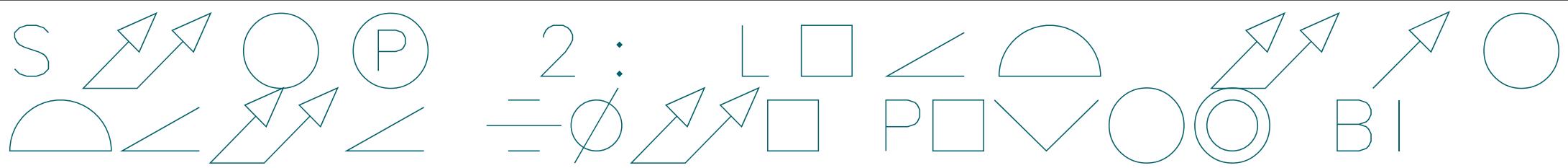
Average Weight by Heights for American Women

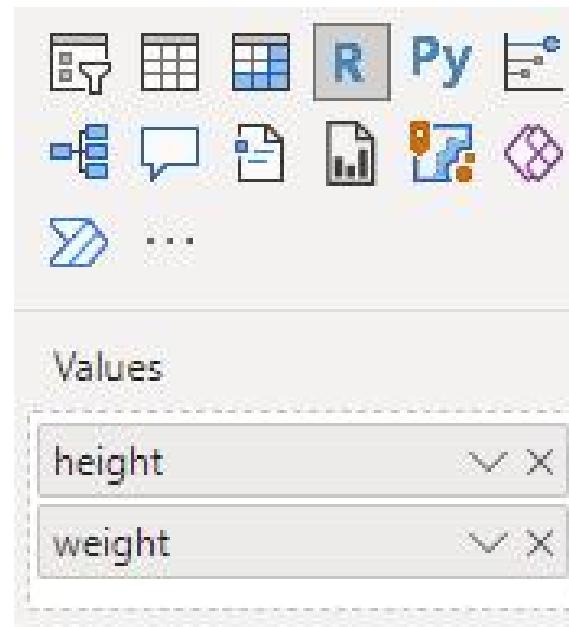
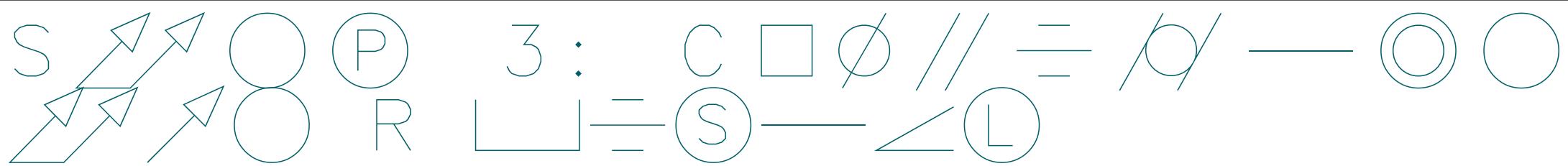
(with added regression line and confidence interval)

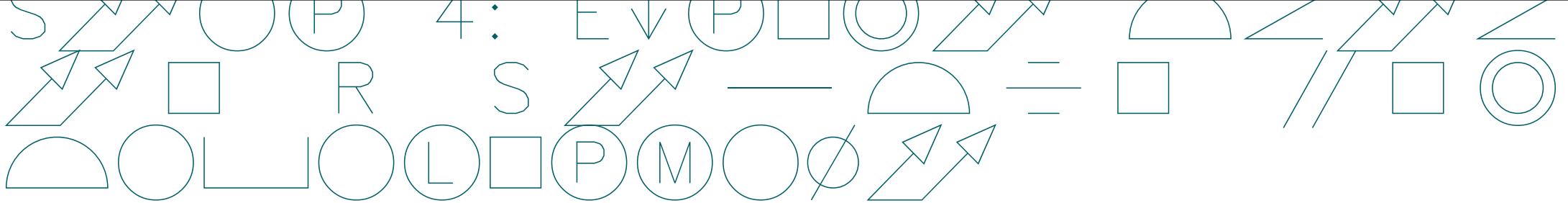




- Import “*women.csv*” data file.
- library(tidyverse)
- data(women)
- setwd("<path to folder where the file is located>")
- write_csv(women, "women.csv")



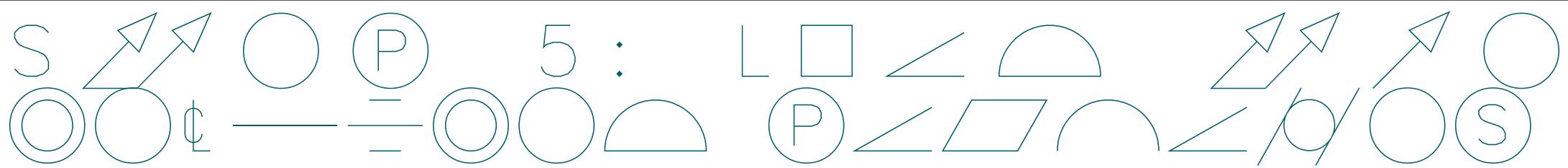




R script editor

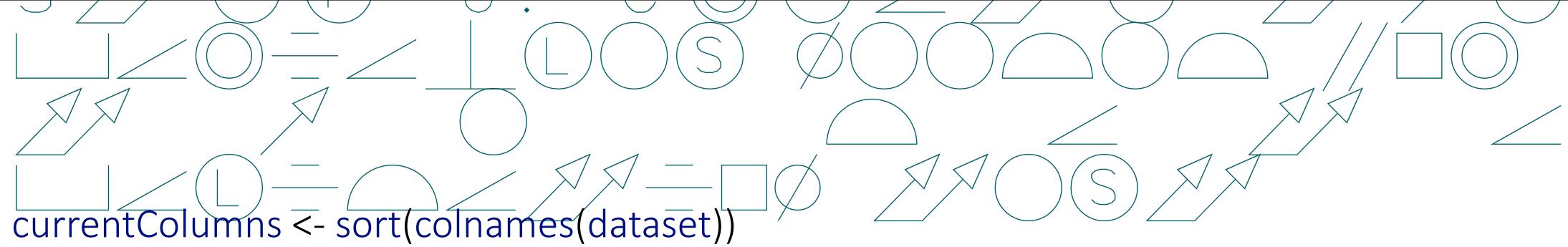
⚠ Duplicate rows will be removed from the data. X

```
1 # The following code to create a dataframe and remove duplicated rows is always executed and acts as a preamble for your script:  
2  
3 # dataset <- data.frame(height, weight)  
4 # dataset <- unique(dataset)  
5  
6 # Paste or type your script code here:
```



```
library(tidyverse)
```

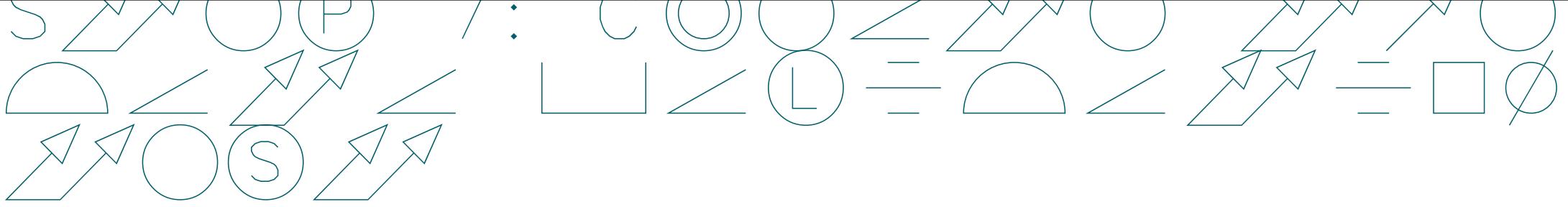
```
library(ggthemes)
```



```
currentColumns <- sort(colnames(dataset))
```

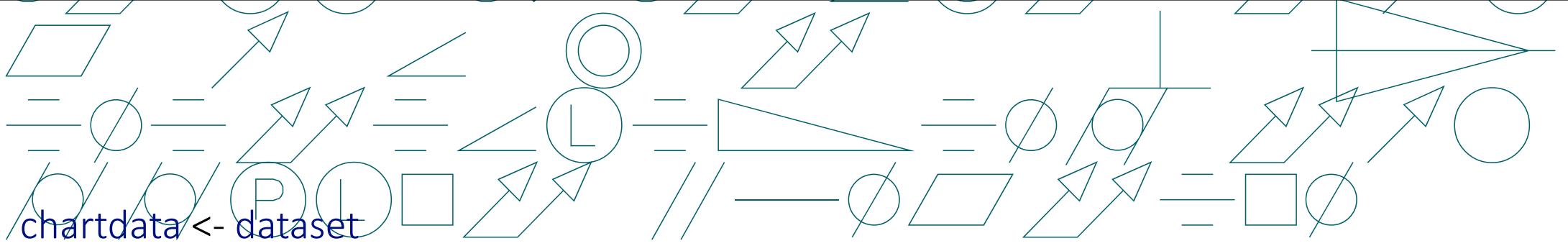
```
requiredColumns <- c("height", "weight")
```

```
columnTest <- isTRUE(all.equal(currentColumns, requiredColumns))
```

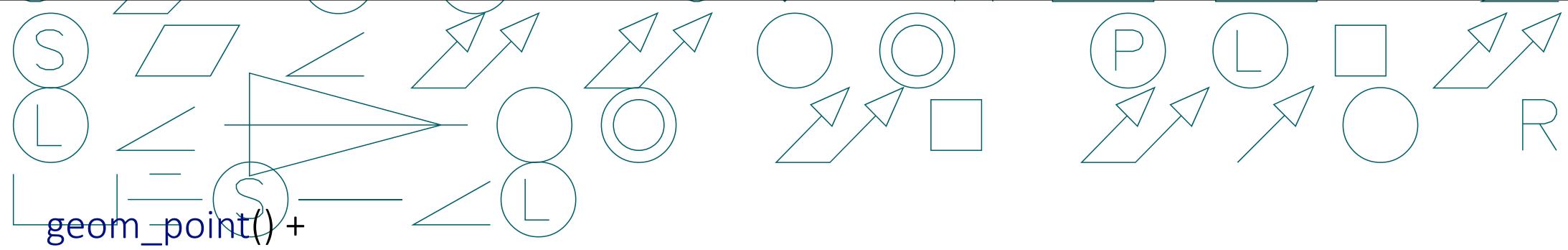


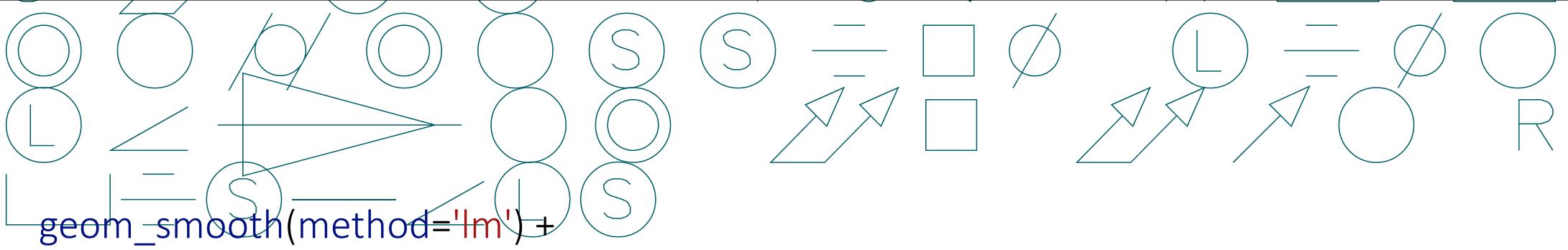
```
if (columnTest) {  
    <code to produce visual>  
} else {
```

```
    plot.new()  
    title("The data supplied did not meet the requirements of the chart.")  
}
```

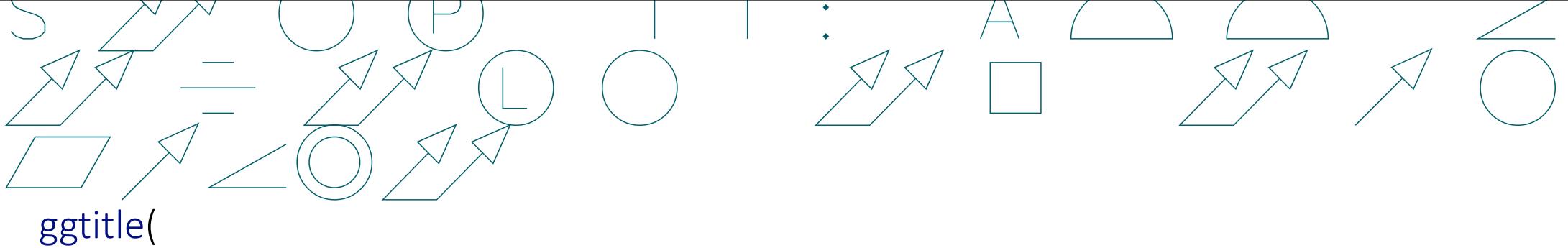


```
ggplot(chartdata, aes(x = weight, y = height)) +
```





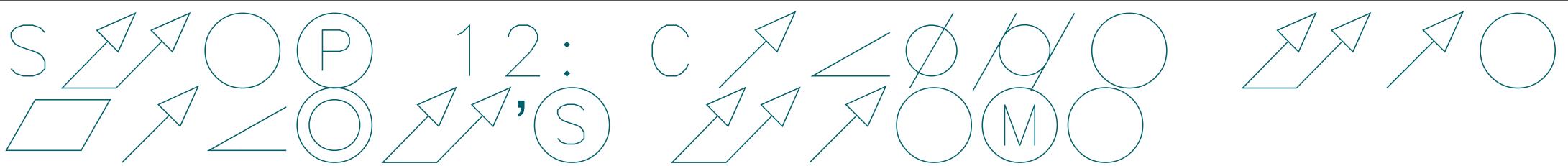
```
geom_smooth(method='lm') +
```



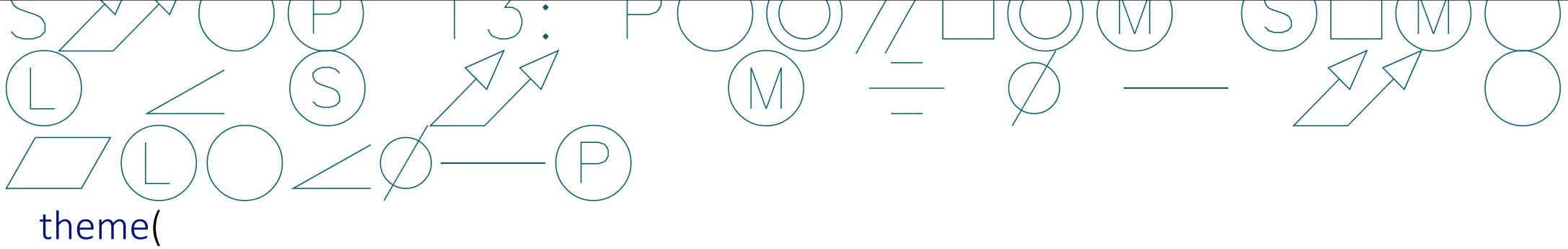
ggttitle(

"Average Weight by Heights for American Women",
subtitle = "(with added regression line and confidence interval)"

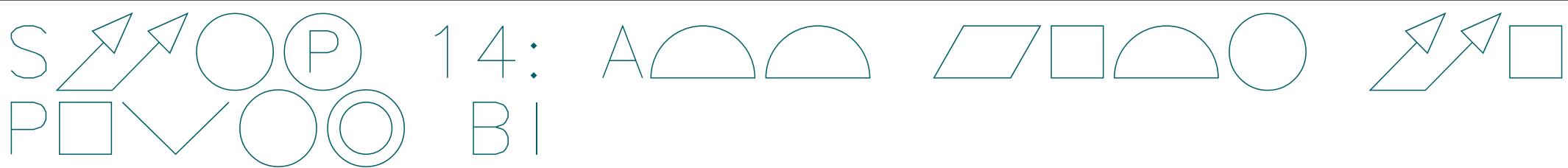
) +



theme_tufte() +



```
theme(  
  axis.title = element_text(size = 20),  
  plot.title = element_text(size = 30),  
  plot.subtitle = element_text(size = 20)  
)
```



14: A
B |

Average Weight by Heights for American Women (with added regression line and confidence interval)

