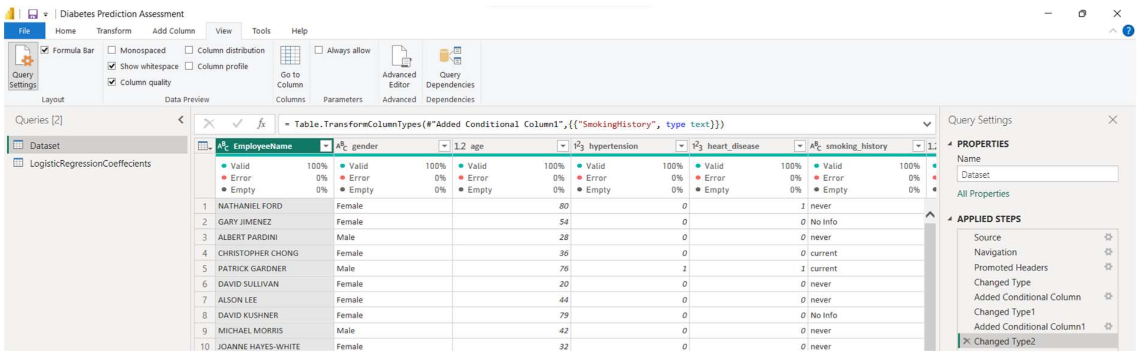
Diabetes Prediction Assessment Answers

1. Data Cleansing: What steps would you take to clean the data, such as handling missing values or outliers?

After loading the data, we need to clean and transform the data. We need to ensure that the data is consistent, doesn't contain null, NaN or empty values, all the columns are of the right data type, there are no unnecessary columns and lastly no outliers.

We can use the **column quality tool** to check for percentage of valid values, values with errors and values which are empty.

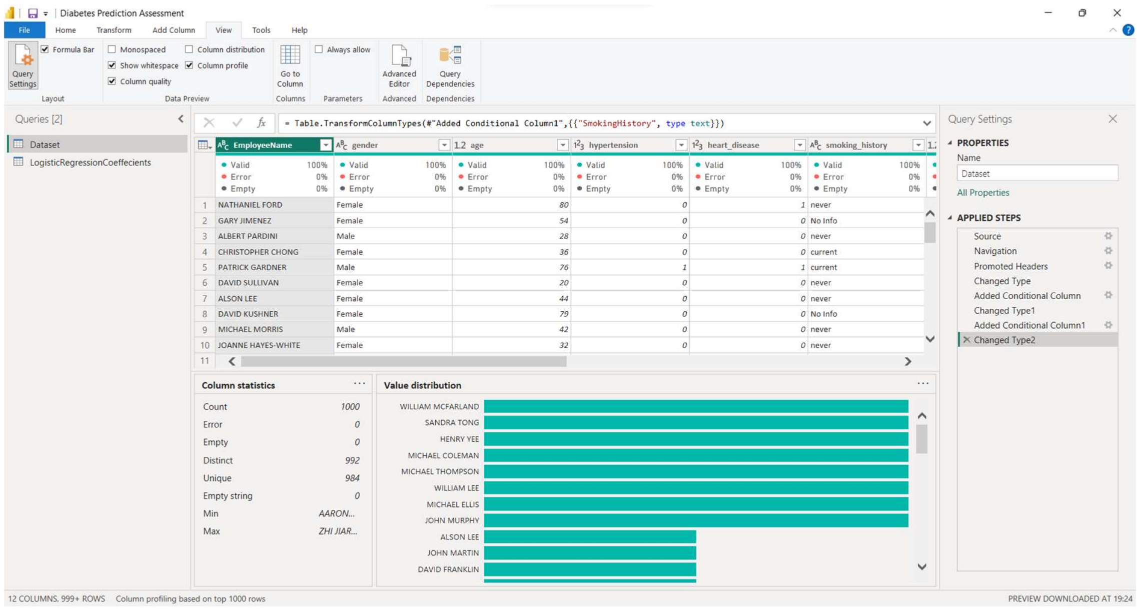


```
= Table.TransformColumnTypes(#"Added Conditional Column1",{{"SmokingHistory", type text}})
```

	EmployeeName	gender	age	hypertension	heart_disease	smoking_history
	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%
1	NATHANIEL FORD	Female	80	0	1	never
2	GARY JIMENEZ	Female	54	0	0	No Info
3	ALBERT PARDINI	Male	28	0	0	never
4	CHRISTOPHER CHONG	Female	36	0	0	current
5	PATRICK GARDNER	Male	76	1	1	current
6	DAVID SULLIVAN	Female	20	0	0	never
7	ALSON LEE	Female	44	0	0	never
8	DAVID KUSHNER	Female	79	0	0	No Info
9	MICHAEL MORRIS	Male	42	0	0	never
10	JOANNE HAYES-WHITE	Female	32	0	0	never

APPLIED STEPS:
- Source
- Navigation
- Promoted Headers
- Changed Type
- Added Conditional Column
- Changed Type1
- Added Conditional Column1
- Changed Type2

We can then use the **column profile tool** to check the value distribution and check key column statistics like Min, Max, Average and Standard Deviation to eliminate the outliers if any.



```
= Table.TransformColumnTypes(#"Added Conditional Column1",{{"SmokingHistory", type text}})
```

	EmployeeName	gender	age	hypertension	heart_disease	smoking_history
	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%	Valid 100% Error 0% Empty 0%
1	NATHANIEL FORD	Female	80	0	1	never
2	GARY JIMENEZ	Female	54	0	0	No Info
3	ALBERT PARDINI	Male	28	0	0	never
4	CHRISTOPHER CHONG	Female	36	0	0	current
5	PATRICK GARDNER	Male	76	1	1	current
6	DAVID SULLIVAN	Female	20	0	0	never
7	ALSON LEE	Female	44	0	0	never
8	DAVID KUSHNER	Female	79	0	0	No Info
9	MICHAEL MORRIS	Male	42	0	0	never
10	JOANNE HAYES-WHITE	Female	32	0	0	never
11						

APPLIED STEPS:
- Source
- Navigation
- Promoted Headers
- Changed Type
- Added Conditional Column
- Changed Type1
- Added Conditional Column1
- Changed Type2

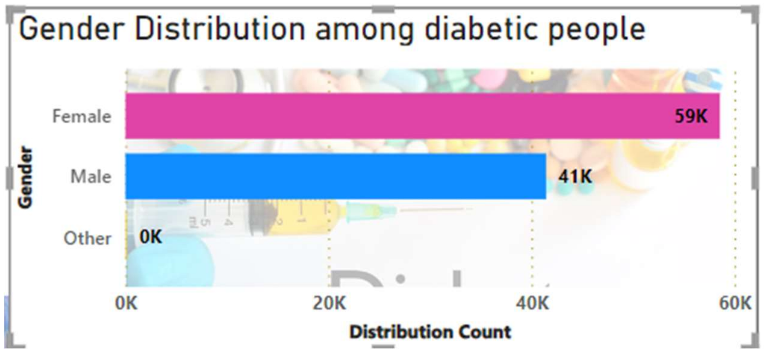
Column statistics

Count	1000
Error	0
Empty	0
Distinct	992
Unique	984
Empty string	0
Min	AARON...
Max	ZHI JIAR...

Value distribution

12 COLUMNS, 999+ ROWS   Column profiling based on top 1000 rows   PREVIEW DOWNLOADED AT 19:24

2. Basic Visualization: Create a simple bar chart to show the distribution of genders in the dataset.



3. DAX Introduction: What is DAX, and why is it used in Power BI?

   Formula language DAX (Data Analysis Expression) is utilized in Power BI to perform tailored computations and consolidations.

   Key characteristics of DAX include:

   Tabular Data Model:

   DAX is optimized for use in tabular data models, which are columnar databases used by tools like Power BI. It operates on columns of data rather than individual cells.

   Formula Language:

   DAX is a formula language, similar to Excel formulas, but tailored for working with tables and columns of data in a database-like structure.

   Aggregation and Calculation:

   DAX is often used to create calculated columns, calculated tables, and measures. Measures, in particular, are dynamic aggregations that allow users to perform calculations on data based on the context of a report or visualization.

   Filter Context:

   DAX formulas automatically adapt to the filter context in which they are used. This means that calculations respond to filters applied in a report, making them dynamic and context-aware.

   Time Intelligence:

   DAX includes functions that are specifically designed for time-related calculations, which is crucial for tasks like year-to-date (YTD), quarter-to-date (QTD), and month-to-date (MTD) aggregations.

   As for why DAX is used in Power BI, Power BI is a business analytics service by Microsoft that provides interactive visualizations and business intelligence capabilities. DAX plays a crucial role in Power BI for the following reasons:

Data Modeling: DAX is used to create relationships between tables, define calculated columns, and establish hierarchies in the data model.

Calculations: DAX enables users to create custom calculations and aggregations, allowing for more advanced and flexible analysis of data.

Dynamic Reporting: Measures written in DAX respond dynamically to changes in filters and selections, providing users with a more interactive and responsive reporting experience.

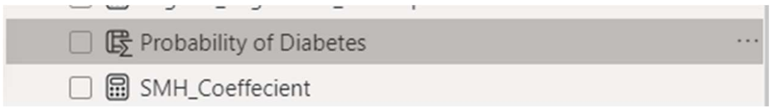
Overall, DAX is a powerful tool in Power BI that empowers users to build sophisticated data models and perform complex calculations, contributing to the creation of insightful and interactive reports and dashboards.

4. Calculated Columns: How can you create a calculated column in Power BI, and why might you need one?

   To create a calculated column in Power BI, select the table in which you need to create the calculated column, right click on the table, select new column, enter the DAX formula in the formula bar and press enter and the new calculated column is ready.

   Alternatively, we can select the new column option from the modelling tab and enter the DAX formula and press enter.

   We need a calculated column when we need to perform a calculation and show the result for each row of the data set. It is represented by a Summation icon as shown below whereas a measure is represented by a calculator icon.

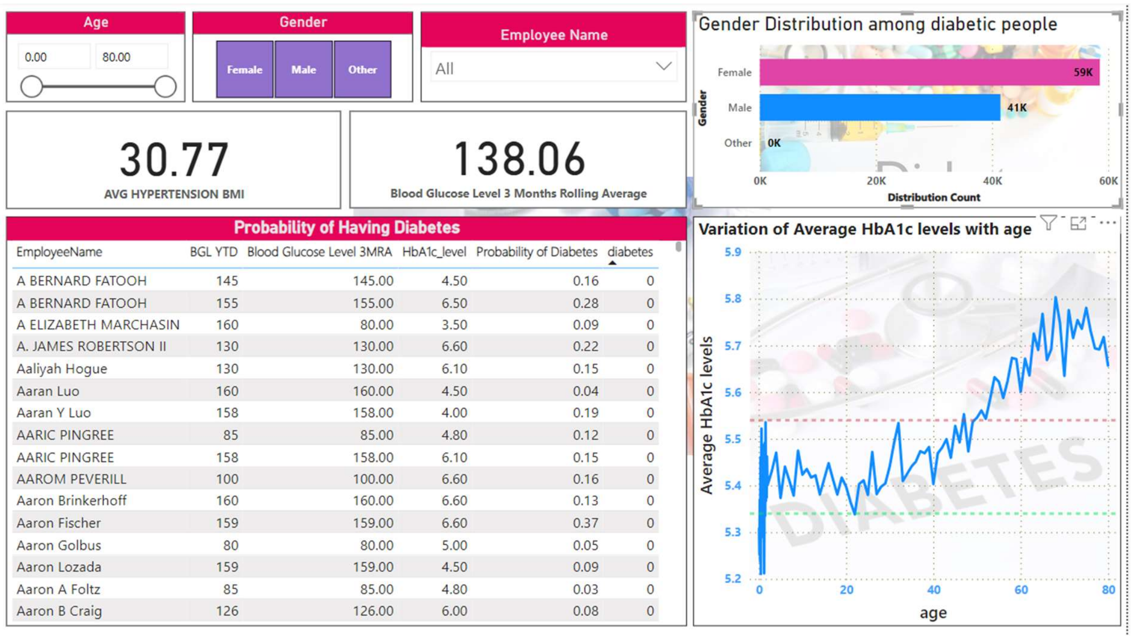


5. Filtering Data: Explain how to filter data to display information for employees aged between 30 and 40.

   To filter data to display info for employees aged between 30 & 40, we can use a page level filter / visual level filter / report level filter from the Filter Pane, select advanced filtering and apply a condition which is age >= 30 and age less than 40 as shown below.

   I have used a page level filter.

   Tab Before filtering



Age

0.00 80.00

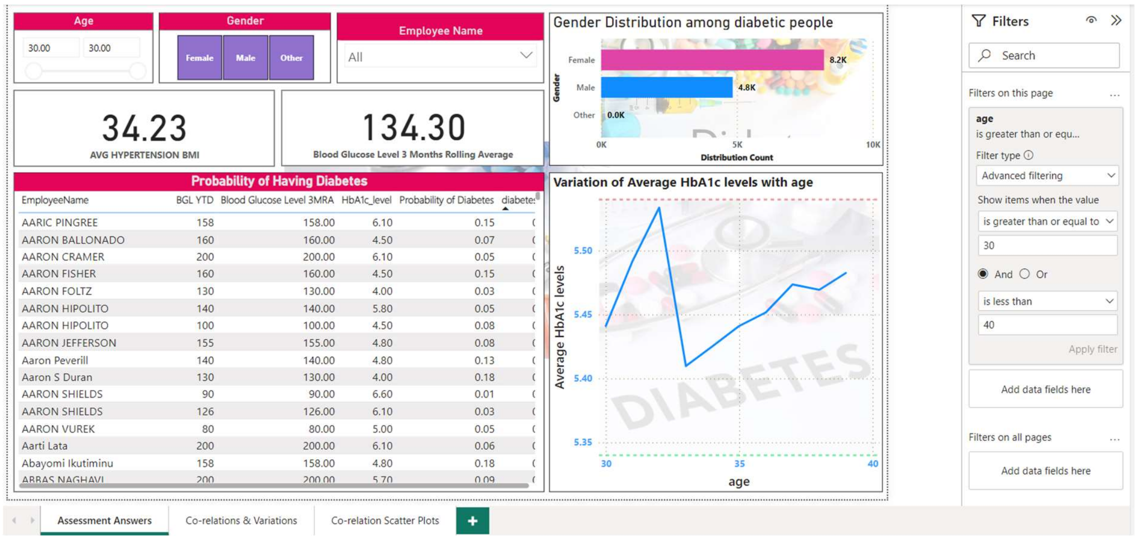
30.77
AVG HYPERTENSION BMI

138.06
Blood Glucose Level 3 Months Rolling Average

Probability of Having Diabetes

EmployeeName	BGL YTD	Blood Glucose Level 3MRA	HbA1c_level	Probability of Diabetes	diabetes
A BERNARD FATOOH	145	145.00	4.50	0.16	0
A BERNARD FATOOH	155	155.00	6.50	0.28	0
A ELIZABETH MARCHASIN	160	80.00	3.50	0.09	0
A. JAMES ROBERTSON II	130	130.00	6.60	0.22	0
Aaliyah Hogue	130	130.00	6.10	0.15	0
Aaran Luo	160	160.00	4.50	0.04	0
Aaran Y Luo	158	158.00	4.00	0.19	0
AARIC PINGREE	85	85.00	4.80	0.12	0
AARIC PINGREE	158	158.00	6.10	0.15	0
AAROM PEVERILL	100	100.00	6.60	0.16	0
Aaron Brinkerhoff	160	160.00	6.60	0.13	0
Aaron Fischer	159	159.00	6.60	0.37	0
Aaron Golbus	80	80.00	5.00	0.05	0
Aaron Lozada	159	159.00	4.50	0.09	0
Aaron A Foltz	85	85.00	4.80	0.03	0
Aaron B Craig	126	126.00	6.00	0.08	0

After filtering



Age

30.00 30.00

34.23
AVG HYPERTENSION BMI

134.30
Blood Glucose Level 3 Months Rolling Average

Probability of Having Diabetes

EmployeeName	BGL YTD	Blood Glucose Level 3MRA	HbA1c_level	Probability of Diabetes	diabetes
AARIC PINGREE	158	158.00	6.10	0.15	(
AARON BALLONADO	160	160.00	4.50	0.07	(
AARON CRAMER	200	200.00	6.10	0.05	(
AARON FISHER	160	160.00	4.50	0.15	(
AARON FOLTZ	130	130.00	4.00	0.03	(
AARON HIPOLITO	140	140.00	5.80	0.05	(
AARON HIPOLITO	100	100.00	4.50	0.08	(
AARON JEFFERSON	155	155.00	4.80	0.08	(
Aaron Peverill	140	140.00	4.80	0.13	(
Aaron S Duran	130	130.00	4.00	0.18	(
AARON SHIELDS	90	90.00	6.60	0.01	(
AARON SHIELDS	126	126.00	6.10	0.03	(
AARON VUREK	80	80.00	5.00	0.05	(
Aarti Lata	200	200.00	6.10	0.06	(
Abayomi Ikutiminu	158	158.00	4.80	0.18	(
ABBAS NAGHAVI	200	200.00	5.70	0.09	(

Filters

Search

Filters on this page

age
is greater than or equ...

Filter type: Advanced filtering

Show items when the value
is greater than or equal to
30

● And ○ Or

is less than
40

Apply filter

Add data fields here

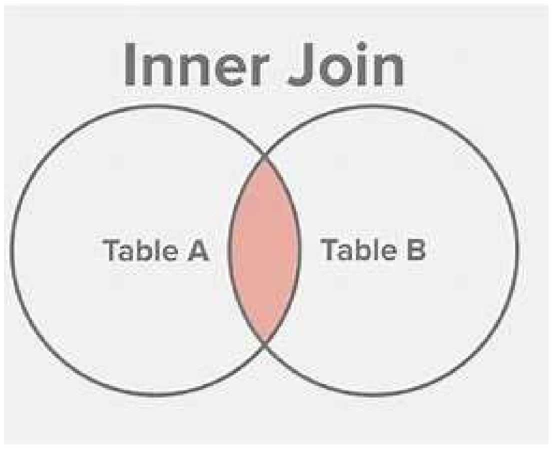
Filters on all pages

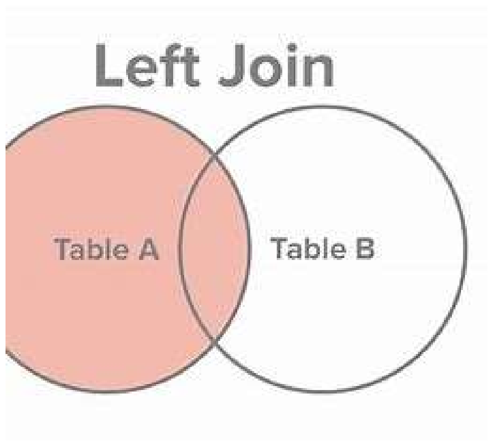
Add data fields here

6. Joins: What is the difference between inner join and left join, and when would you use each in Power BI?

   An inner join combines all the records from both the tables which satisfy the join condition, whereas a left join merges records such that it keeps all records from the left table and only the matching records from the right table which satisfy the join condition.

   We would use an inner join when we need matching records from both tables whereas we would use a left join when we need all records from the left table and only matching records from the right table.





7. Data Modelling: Describe the importance of data modelling in Power BI and how it impacts your visualizations.

Data modelling is a crucial aspect of Power BI that significantly impacts the effectiveness and efficiency of your visualizations. Here are key aspects of the importance of data modelling in Power BI and its impact on visualizations:

Relationships between Tables:

Importance: Data modelling involves establishing relationships between tables. These relationships define how tables are related and enable the creation of meaningful connections between different sets of data.

Impact on Visualizations: Relationships allow for the creation of more complex and insightful visualizations by allowing users to drill down into related data across different tables.

Calculated Columns and Measures:

Importance: Data modelling involves the creation of calculated columns and measures. Calculated columns add new columns to a table, and measures are dynamic calculations based on the data in the model.

Impact on Visualizations: Calculated columns and measures provide the foundation for creating custom calculations, aggregations, and key performance indicators (KPIs) that are crucial for building meaningful visualizations.

Hierarchies and Grouping:

Importance: Data modelling allows the creation of hierarchies and grouping within tables. Hierarchies provide a structured way to navigate data, and grouping helps organize data for analysis.

Impact on Visualizations: Hierarchies and grouping enhance the user experience by enabling users to explore data at different levels of granularity. They play a vital role in creating interactive and drillable visualizations.

Data Transformation and Cleaning:

Importance: Data modelling involves data transformation and cleaning processes, such as handling missing values, removing duplicates, and transforming data types.

Impact on Visualizations: Clean and transformed data ensures the accuracy and reliability of visualizations. Proper data preparation enhances the quality of insights derived from visualizations.

Performance Optimization:

Importance: Efficient data modelling includes considerations for optimizing performance, such as managing table relationships, reducing unnecessary columns, and using appropriate data types.

Impact on Visualizations: Well-optimized data models contribute to faster query response times, resulting in more responsive and interactive visualizations.

Integration with External Data Sources:

Importance: Power BI allows integration with a variety of external data sources. Data modelling includes connecting to and integrating data from these external sources.

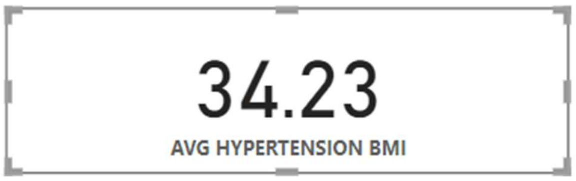
Impact on Visualizations: Integrated external data sources provide a comprehensive view of the business landscape, allowing for more holistic and insightful visualizations.

In summary, effective data modelling in Power BI lays the foundation for creating powerful and meaningful visualizations. It enables users to explore, analyze, and derive insights from complex datasets in an intuitive and interactive manner. A well-designed data model ensures that visualizations accurately represent the underlying data, leading to more informed decision-making processes.

The data which has been used in this report consists of just one table and hence no modelling was required.

8. Measures in DAX: Create a DAX measure to calculate the average BMI for employees with hypertension.

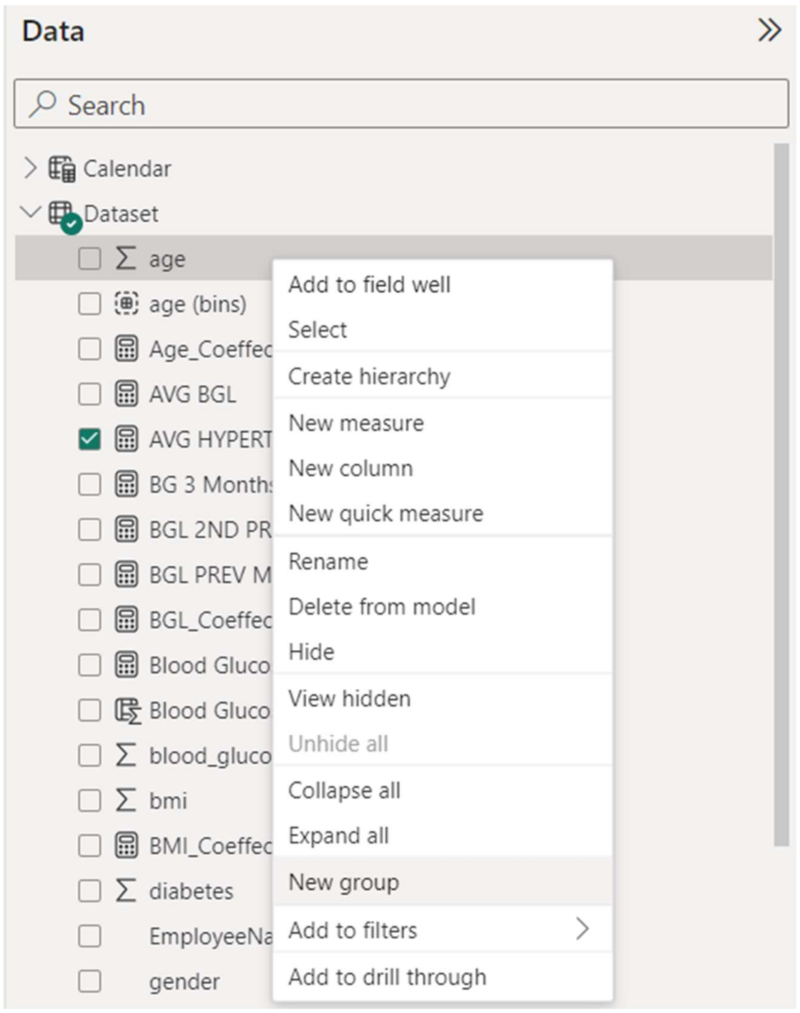
```
AVG HYPERTENSION BMI = CALCULATE(AVERAGE('Dataset'[bmi]), 'Dataset'[hypertension] == 1)
```


34.23

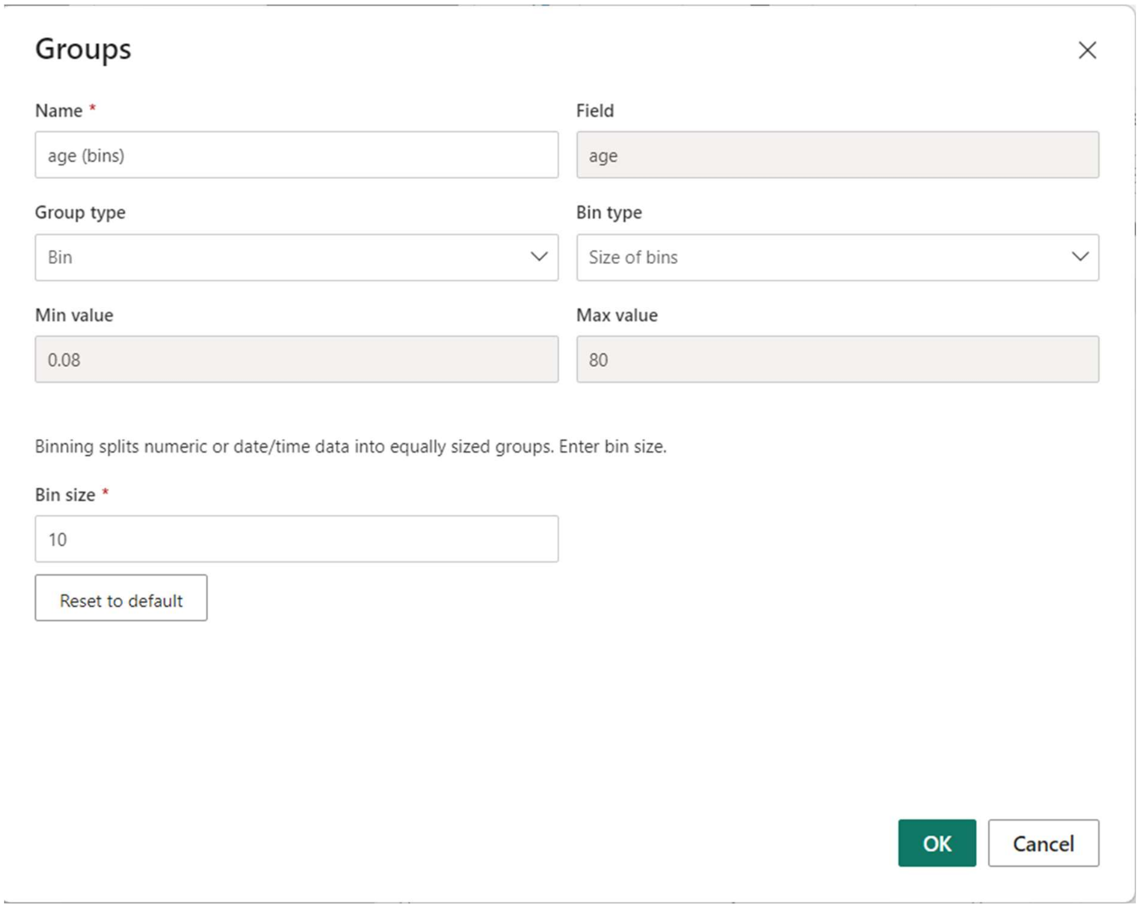
AVG HYPERTENSION BMI

9. Advanced Filtering: How can you create a slicer that allows users to filter data based on age ranges (e.g., 20-30, 30-40, 40-50)?

   Firstly, we need to create an age group. This can be achieved by right clicking on the age field and selecting new group.



   Then we need to select the bin size to appropriately create age groups.



Groups

Name *: age (bins)

Field: age

Group type: Bin

Bin type: Size of bins

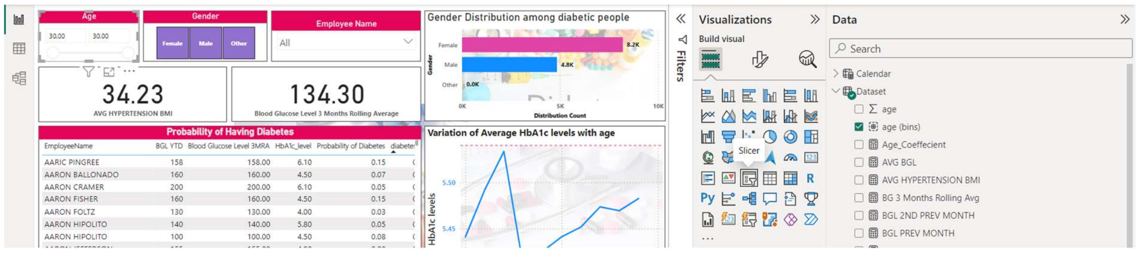
Min value: 0.08

Max value: 80

Binning splits numeric or date/time data into equally sized groups. Enter bin size.

Bin size *: 10

After this we can select the slicer from the visualization pane and add the age group bin we had created into the field of the slicer.



Probability of Having Diabetes

EmployeeName	BGL YTD	Blood Glucose Level 3MRA	HbA1c_level	Probability of Diabetes
AARIC PINGREE	158	158.00	6.10	0.15
AARON BALLONADO	160	160.00	4.50	0.07
AARON CRAMER	200	200.00	6.10	0.05
AARON FISHER	160	160.00	4.50	0.15
AARON FOLTZ	130	130.00	4.00	0.03
AARON HIPOLITO	140	140.00	5.80	0.05
AARON HIPOLITO	100	100.00	4.50	0.08

Data
- Calendar
- Dataset
  - [ ] age
  - [✓] age (bins)
  - [ ] Age_Coeffecient
  - [ ] AVG BGL
  - [ ] AVG HYPERTENSION BMI
  - [ ] BG 3 Months Rolling Avg
  - [ ] BGL 2ND PREV MONTH
  - [ ] BGL PREV MONTH

10. Time Intelligence: Explain how to use DAX to calculate the year-to-date total for blood glucose levels.

```
Blood Glucose YTD =

 VAR SelectedRow = MAX ('Dataset'[EmployeeName] )
RETURN CALCULATE (SUM('Dataset'[blood_glucose_level]), FILTER(ALL(
'Dataset'[EmployeeName] ), 'Dataset'[EmployeeName] <= SelectedRow ))
```

Explanation:

VAR (Variable):

VAR SelectedRow = MAX('Dataset'[EmployeeName]): This line defines a variable named SelectedRow that stores the maximum value of the 'EmployeeName' column in the 'Dataset' table. It seems to be aiming to capture the maximum employee name, but note that this might not be the most appropriate approach for YTD calculations.

RETURN Statement:

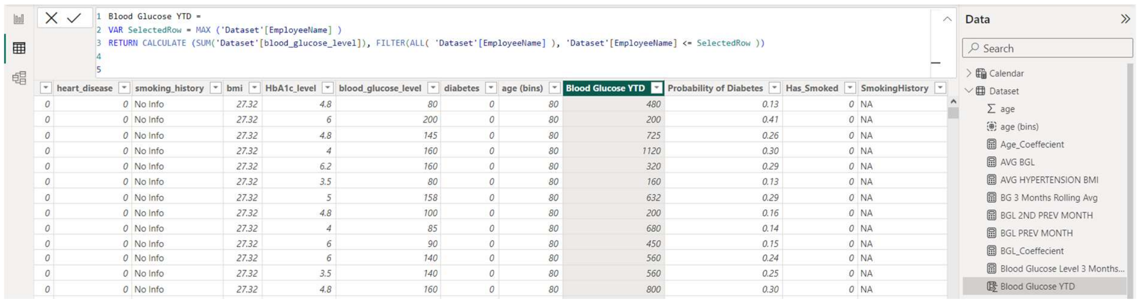
RETURN: This keyword indicates the start of the expression that the measure will return.

CALCULATE Function:

CALCULATE(SUM('Dataset'[blood_glucose_level]), ...): This is the main calculation. It uses the CALCULATE function to calculate the sum of the 'blood_glucose_level' column in the 'Dataset' table.

FILTER Function:

FILTER(ALL('Dataset'[EmployeeName]), 'Dataset'[EmployeeName] <= SelectedRow): This part filters the 'Dataset' table. It uses the ALL function to remove any filters applied to 'EmployeeName' and then filters the table to include only rows where 'EmployeeName' is less than or equal to the SelectedRow. This part aims to filter data up to the selected employee, but it may not be an appropriate approach for YTD calculations.



```
Blood Glucose YTD =
VAR SelectedRow = MAX ('Dataset'[EmployeeName] )
RETURN CALCULATE (SUM('Dataset'[blood_glucose_level]), FILTER(ALL( 'Dataset'[EmployeeName] ), 'Dataset'[EmployeeName] <= SelectedRow ))
```

	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes	age (bins)	Blood Glucose YTD	Probability of Diabetes	Has_Smoked	SmokingHistory
0	0	No Info	27.32	4.8	80	0	80	480	0.13	0	NA
0	0	No Info	27.32	6	200	0	80	200	0.41	0	NA
0	0	No Info	27.32	4.8	145	0	80	725	0.26	0	NA
0	0	No Info	27.32	4	160	0	80	1120	0.30	0	NA
0	0	No Info	27.32	6.2	160	0	80	320	0.29	0	NA
0	0	No Info	27.32	3.5	80	0	80	160	0.13	0	NA
0	0	No Info	27.32	5	158	0	80	632	0.29	0	NA
0	0	No Info	27.32	4.8	100	0	80	200	0.16	0	NA
0	0	No Info	27.32	4	85	0	80	680	0.14	0	NA
0	0	No Info	27.32	6	90	0	80	450	0.15	0	NA
0	0	No Info	27.32	6	140	0	80	560	0.24	0	NA
0	0	No Info	27.32	3.5	140	0	80	560	0.25	0	NA
0	0	No Info	27.32	4.8	160	0	80	800	0.30	0	NA

11. Complex Joins: Combine data from multiple tables, including employee data and diabetes data, using appropriate joins.

    Since the data set contains only 1 table and there was no need to normalize this table into multiple tables, there was no need to use any join.

12. Data Aggregation: What is the purpose of SUMMARIZE in DAX, and how can you use it to aggregate data?

The SUMMARIZE function in DAX (Data Analysis Expressions) is used to create summary tables and aggregations based on specified grouping columns. It allows you to generate a table with summarized or aggregated data, often used for reporting or analysis.

Here's the basic syntax of the SUMMARIZE function:

```
SUMMARIZE (

    <Table>,

    [<Grouping_Column1>, <Grouping_Column2>, ...],

    [<Name_Column1>, <Name_Column2>, ...],

    [<Aggregate_Column1>, <Aggregate_Column2>, ...]

)
```

<Table>: The name of the table for which you want to create a summary.

<Grouping_Column1>, <Grouping_Column2>, ...: Columns used for grouping and creating unique combinations.

<Name_Column1>, <Name_Column2>, ...: Columns whose unique values will become the names of the new table columns.

<Aggregate_Column1>, <Aggregate_Column2>, ...: Columns for which you want to perform aggregation (e.g., sum, average).

Here's an example of how we might use SUMMARIZE:

```
Gender wise BMI Summary =

SUMMARIZE (

    Dataset,

    Dataset[Gender],

    "AVG BMI", AVEARGE(Dataset[bmi])

)
```

In this example:

Dataset is the base table.

Dataset[Gender] is the grouping column,

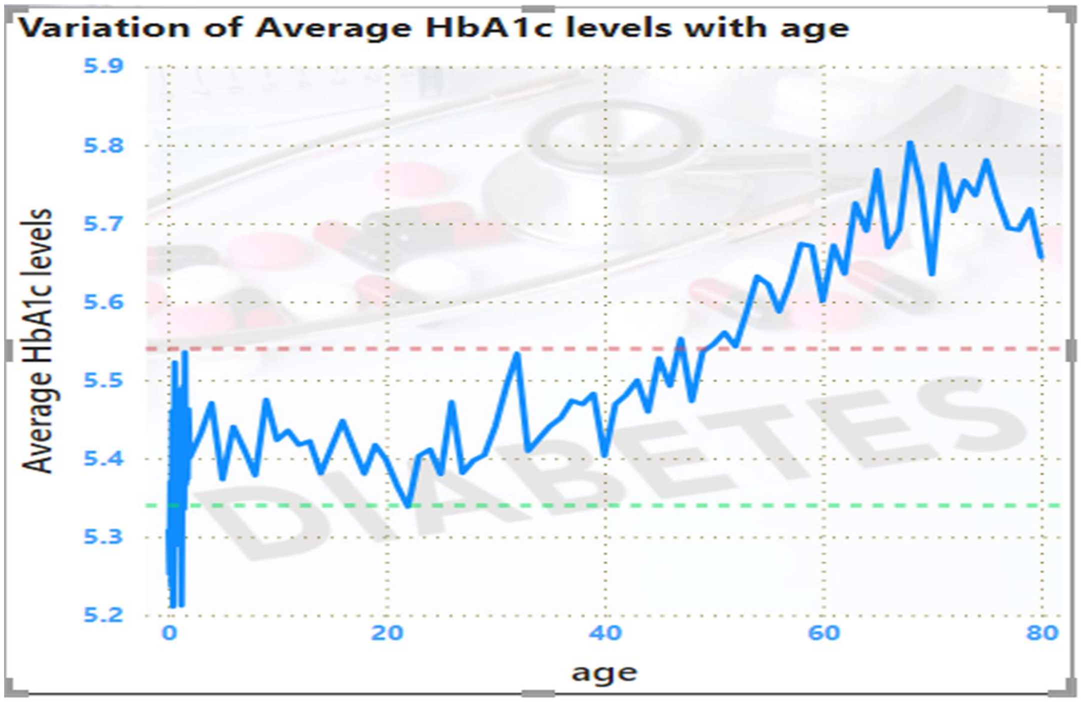
"AVG BMI" is the name for the new column in the summary table.

AVERAGE(Sales[Quantity]) is the aggregations performed on the specified column.

The resulting Gender wise BMI Summary table will have unique rows for each gender category, along with average BMI for each category.

The SUMMARIZE function is powerful for creating aggregated tables that can be used in Power BI visuals or for further analysis. It's especially useful when you want to group and summarize data based on certain dimensions or attributes in your dataset.

13. Advanced Visualization: Create a line chart that shows the trend of HbA1c levels over time, with proper axis formatting.



14. Performance Optimization: Discuss techniques for improving the performance of a Power BI report, especially when dealing with large datasets.

Optimizing the performance of a Power BI report, particularly when dealing with large datasets, is crucial for providing a smooth user experience. Here are several techniques to improve performance:

Data Model Optimization:

Use Import Mode: When feasible, use Import mode rather than DirectQuery. Importing data into the Power BI model can significantly improve query performance.

Limit Columns: Only load the columns you need for analysis. Reducing the number of columns in your tables can speed up data refresh and decrease the model size.

Remove Unnecessary Tables: If certain tables are not essential for your analysis, consider removing them from the data model.

Data Transformations:

Optimize Queries: In Power Query, optimize your queries by removing unnecessary steps, avoiding excessive data transformations, and filtering data early in the process.

Use Query Folding: Leverage query folding when possible. This allows Power BI to push operations back to the data source, reducing the amount of data brought into the Power BI environment.

Indexing and Sorting:

Sort Columns: Sort columns in your data model, especially those used for relationships, as this can improve compression and query performance.

Create Indexes: In the data source (e.g., SQL Server), ensure that columns used for filtering or grouping are indexed. This can significantly speed up query performance.

Aggregations and Summary Tables:

Use Summary Tables: Create pre-aggregated summary tables to store aggregated values for commonly used metrics. This can reduce the need for recalculating aggregations during report rendering.

Leverage Aggregations in Power BI Premium: Power BI Premium offers the capability to use aggregations to store pre-aggregated data for better performance.

Visual Best Practices:

Limit Visuals per Page: Limit the number of visuals on a report page to reduce the workload on the rendering engine.

Use Card Visuals for Single Values: For single values or key metrics, consider using card visuals instead of tables or charts.

Filtering and Slicers:

Use Slicers Wisely: Limit the number of slicers on a page, and be cautious with slicers that have many unique values, as they can impact performance.

Implement Smart Filtering: Use cross-filtering and relationships efficiently to filter data at the model level before it reaches the report layer.

Partitioning:

Partition Large Tables: If your data source supports it, consider partitioning large tables. This allows Power BI to read and refresh only the necessary partitions, improving performance.

Use Power BI Performance Analyzer:

Performance Analyzer Tool: Utilize the built-in Performance Analyzer tool in Power BI Desktop to identify bottlenecks and performance issues. It helps analyze query durations and resource usage.

Power BI Premium:

Consider Power BI Premium: If your organization deals with large datasets and requires enhanced performance, consider investing in Power BI Premium, which offers dedicated cloud resources for better scalability and performance.

Implementing these techniques requires a balance between the specific needs of your report and the constraints of your data source. Regular testing and monitoring are essential to ensure ongoing optimal performance.

14. Row-Level Security: How can you implement row-level security in Power BI to restrict access to sensitive employee data?

Row-level security (RLS) in Power BI allows you to restrict data access at the row level based on user roles. This is particularly useful when you need to control access to sensitive data, such as employee information. Here's a general guide on how to implement row-level security in Power BI:

Define Roles:

In Power BI Desktop, go to the "Model" view.

Under the "Model" view, select "Manage Roles" from the "Model" tab.

Click on "Create" to add a new role.

Create Roles:

Define roles based on your security requirements. For example, you might create roles like "Managers," "HR," and "Employees."

Assign Users to Roles:

In the "Manage Roles" window, assign Power BI users to the roles you've created.

Users can be added individually or imported from a security group if using Power BI service.

Implement DAX Filters:

Once roles are created, use DAX filters to restrict data at the row level based on the user's role.

For example, you might use the USERNAME() or USERPRINCIPALNAME() DAX functions to create filters like:

DAX

```
EmployeeTable[EmployeeID] IN VALUES('SecurityTable'[EmployeeID])
```

This filter ensures that users can only see rows in the "EmployeeTable" where their "EmployeeID" matches the "EmployeeID" values in the "SecurityTable."

Test Row-Level Security:

In Power BI Desktop, use the "View as Roles" feature to test row-level security for different roles.

This allows you to preview the data as it would be seen by users in each role.

Publish to Power BI Service:

Publish your Power BI report to the Power BI service.

Ensure that the users accessing the report in the Power BI service are assigned to the appropriate roles.

Manage Security in Power BI Service:

In the Power BI service, go to the dataset settings.

Under the "Security" tab, manage roles and assign users to roles.

Monitor and Update:

Regularly monitor and update row-level security as needed.

Adjust roles and filters based on changes in your organization's security requirements.

Remember to be cautious with the use of dynamic security filters and DAX expressions, especially if they involve sensitive data. Always test thoroughly and consider the implications of each security configuration. Row-level security is effective when configured correctly but requires ongoing management to ensure it meets the evolving needs of your organization.

15. Advanced DAX: Write DAX code to calculate the rolling average of blood glucose levels over a 3-month period.

```
    Blood Glucose Level 3 Months Rolling Average =

VAR PeriodToUse = DATESINPERIOD ('Calendar' [Date], LASTDATE ('Calendar' [Date]), -3,
MONTH)

VAR Result = CALCULATE (AVERAGE ( 'Dataset'[blood_glucose_level] ), PeriodToUse)

VAR NMonthsPeriodBlank = if ( [AVG BGL] = BLANK (),0,1) + if ( [BGL PREV MONTH] = BLANK
(),0,1) + if ( [BGL 2ND PREV MONTH] = BLANK (),0,1)

RETURN IF (NMonthsPeriodBlank < 3, BLANK (), Result)
```

The DAX formula you provided appears to be calculating a 3-month rolling average for blood glucose levels in a Power BI model. Let's break down the key components of the formula:

Let's break down the code step by step:

VAR PeriodToUse:

This variable uses the DATESINPERIOD function to define a date range for the last 3 months based on the 'Calendar' table's [Date] column.

VAR Result:

This variable uses the CALCULATE function to calculate the average of the 'Dataset'[blood_glucose_level] column within the specified period.

VAR NMonthsPeriodBlank:

This variable calculates the number of months within the specified period that have blank values for blood glucose levels. It adds up the blanks for the current month ([AVG BGL]), the previous month ([BGL PREV MONTH]), and the second previous month ([BGL 2ND PREV MONTH]).

RETURN:

The RETURN statement uses an IF statement to check if the number of months with blank values is less than 3. If so, it returns BLANK(); otherwise, it returns the calculated result.

Overall, the formula calculates a 3-month rolling average for blood glucose levels, taking into account blank values for each month. If there are more than 2 months with blank values within the rolling period, the result is set to BLANK(). This could be useful for handling scenarios where there might be missing data in the blood glucose levels for some months.

16. Custom Visuals: Explain the process of importing and using custom visuals in Power BI.

Importing and using custom visuals in Power BI allows you to enhance your reports and dashboards with additional visualization options beyond the built-in visuals. Here's a step-by-step process for importing and using custom visuals in Power BI:

Importing Custom Visuals:

Access the Power BI Marketplace:

Open Power BI Desktop or go to the Power BI service.

If using Power BI Desktop, go to the "Home" tab, and select "Get Data" to start a new report.

Open AppSource:

In Power BI Desktop, navigate to the "Visualizations" pane on the right.

Click on the "Import from Marketplace" option (icon resembling a puzzle piece).

Browse or Search for Custom Visuals:

The Power BI Marketplace (AppSource) will open.

Browse or search for the custom visuals you want to use. These can include visuals created by Microsoft or third-party developers.

Select and Import:

Click on a custom visual to view details.

Click the "Add" button to add it to your Power BI Desktop or service.

Confirm Import:

Confirm the import by clicking "Add" or "Import."

Using Custom Visuals in Power BI Desktop:

Access Custom Visuals Pane:

In Power BI Desktop, navigate to the "Visualizations" pane on the right.

Find Imported Visuals:

Look for the imported custom visuals in the "Custom" section of the "Visualizations" pane.

Drag and Drop:

Drag the desired custom visual from the "Custom" section onto the report canvas.

Configure and Customize:

Customize the visual by dragging fields to the appropriate well (e.g., Values, Axis, Legend) in the "Fields" pane.

Configure any specific settings provided by the custom visual.

Using Custom Visuals in Power BI Service:

Publish Report to Power BI Service:

If using Power BI Desktop, save your report and publish it to the Power BI service.

Open Report in Power BI Service:

Open the report in the Power BI service.

Access Custom Visuals:

In the Power BI service, navigate to the "Visualizations" pane on the right.

Find and Add Custom Visuals:

Look for the imported custom visuals in the "Custom" section of the "Visualizations" pane.

Drag the custom visual onto the report canvas.

Configure and Interact:

Configure and interact with the custom visual as needed.

Save and share the report with others.

Notes:

Some custom visuals may require additional setup or configuration. Refer to documentation provided by the visual's creator.

Be cautious when using third-party visuals. Ensure they come from reputable sources, as they have access to your data.

Custom visuals are not supported in all Power BI deployment scenarios (e.g., Power BI Report Server). Verify compatibility based on your environment.

By following these steps, you can easily import and use custom visuals to enhance your Power BI reports and gain additional insights from your data.

17. Cross-Filtering: Describe the concept of cross-filtering and its impact on report interactivity.

Cross-filtering is a concept in Power BI that refers to the ability of one table or visualization to affect the data displayed in another table or visualization. It is a key mechanism that enhances the interactivity of Power BI reports, allowing users to explore and analyze data dynamically. Cross-filtering establishes relationships between tables and enables the propagation of filters from one table to another.

Here's how cross-filtering works and its impact on report interactivity:

How Cross-Filtering Works:

Establish Relationships:

In Power BI, you define relationships between tables based on common columns. For example, you might have a relationship between a 'Sales' table and a 'Customers' table based on the 'CustomerID' column.

Filter Propagation:

When you apply a filter in one visualization or table (e.g., select a specific customer), the filter is propagated through the established relationships to other related tables.

Filter Direction:

Filters can have different directions: one-way or two-way.

One-Way Filter (Default): The filter flows from the "one" side of the relationship to the "many" side.

Two-Way Filter: The filter flows in both directions, allowing bidirectional filtering.

Interactive Selections:

As users interact with one visualization or table (e.g., click on a data point in a chart), the related visuals are automatically updated to show relevant data based on the applied filter.

Impact on Report Interactivity:

Dynamic Exploration:

Cross-filtering enables users to dynamically explore data by interacting with different visualizations and tables. Users can click on data points, select values, or use slicers to filter data, and the entire report adjusts accordingly.

Drill-Down and Drill-Up:

Users can drill down into more detailed information or drill up to see higher-level summaries. For example, clicking on a specific category in a chart might drill down to show details for that category in other visuals.

Contextual Analysis:

Cross-filtering provides context-aware analysis. Filters applied in one part of the report contextually impact related parts, allowing users to understand how specific selections affect the overall data.

Synchronized Visuals:

Visualizations that share relationships automatically stay synchronized. For instance, selecting a specific product in a slicer could update charts, tables, or other visuals related to sales, revenue, or other metrics associated with that product.

Enhanced User Experience:

Overall, cross-filtering enhances the user experience by providing a more interactive and dynamic environment. Users can answer ad-hoc questions, explore trends, and gain insights without the need for predefined static reports.

In summary, cross-filtering is a powerful feature in Power BI that promotes interactive data exploration and analysis. It allows users to seamlessly navigate through data, gain insights, and make informed decisions by interactively applying filters and exploring relationships between tables and visuals.

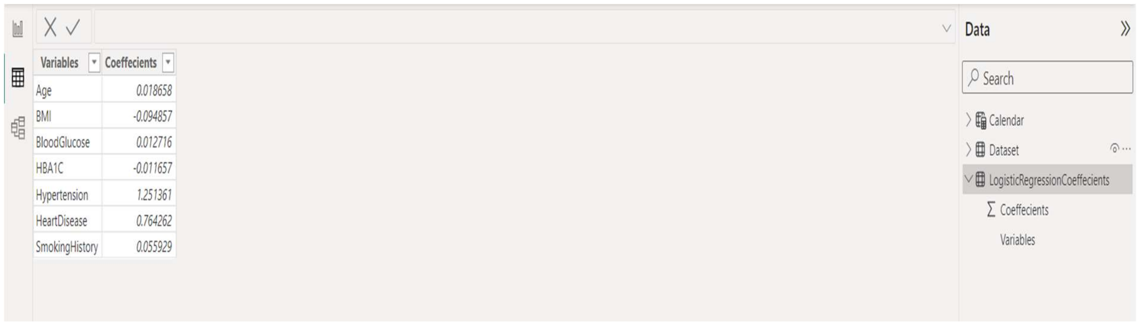
18. Advanced Calculations: Create a DAX measure that calculates the probability of an employee having diabetes based on their age, gender, and other factors.

    This was a very tricky and advanced DAX formula which was suggested by Co-pilot as, I was not able recall the probability formulae and was finding myself stuck. It was wonderful being able to learn this from Co-pilot and execute this formula using a Logistic Regression model using Python Script on Jupyter notebook.

```
Probability of Diabetes =
VAR Intercept = [Logistic_Regression_Intercept]
VAR LinearPredictor =
Intercept +
'Dataset'[Age_Coeffecient] * 'Dataset'[age] +
'Dataset'[Hypertension_Coeffecient] * 'Dataset'[hypertension] +
'Dataset'[HD_Coeffecient] * 'Dataset'[heart_disease] +
'Dataset'[SMH_Coeffecient] * 'Dataset'[Has_Smoked] +
'Dataset'[BMI_Coeffecient] * 'Dataset'[bmi] +
'Dataset'[BGL_Coeffecient] *
'Dataset'[blood_glucose_level]+
                                             'Dataset'[HBA1C_Coeffecient] *
'Dataset'[HbA1c_level]
RETURN
 1 / (1 + EXP(-LinearPredictor))
```

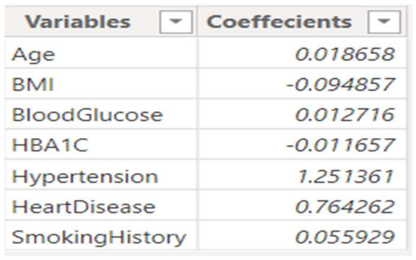
```
Where,

  Logistic_Regression_Intercept = -1.738933225384652

  Logistic_Regression_Coefficients = CONCATENATEX(LogisticRegressionCoeffecients,
FORMAT('LogisticRegressionCoeffecients'[Coeffecients], "#.###"), ", ")
```

I have manually entered data in a table to store the variables and their coefficients which I have obtained using a Logistic Regression ML Model in Python with help from Microsoft's Co-pilot.



Variables	Coeffecients
Age	0.018658
BMI	-0.094857
BloodGlucose	0.012716
HBA1C	-0.011657
Hypertension	1.251361
HeartDisease	0.764262
SmokingHistory	0.055929

Variables and their coeffecients



Variables	Coeffecients
Age	0.018658
BMI	-0.094857
BloodGlucose	0.012716
HBA1C	-0.011657
Hypertension	1.251361
HeartDisease	0.764262
SmokingHistory	0.055929

Please refer the Python Script in the same folder as this document for more clarity.

Here is a link to the related LinkedIn post which also has a link to my presentation of the report.

https://www.linkedin.com/posts/nik995_psyliq-powerbi-dataanalysis-activity-7146017302398894082-lp_3?utm_source=share&utm_medium=member_desktop