DATA ANALYST INTERNSHIP

Task 1. Diabetes Prediction Analysis

By- Mansi Pawar

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

Data cleansing is an essential step to ensure the quality and reliability of your analysis. Here are some steps to clean the data:

1. Handle Missing Values:

Replace missing values with a reasonable estimate, such as the mean, median, or mode of the column. Remove rows or columns with a high proportion of missing values if they are not critical for analysis. Replace missing values based on domain knowledge or business rules.

2. Address Outliers:

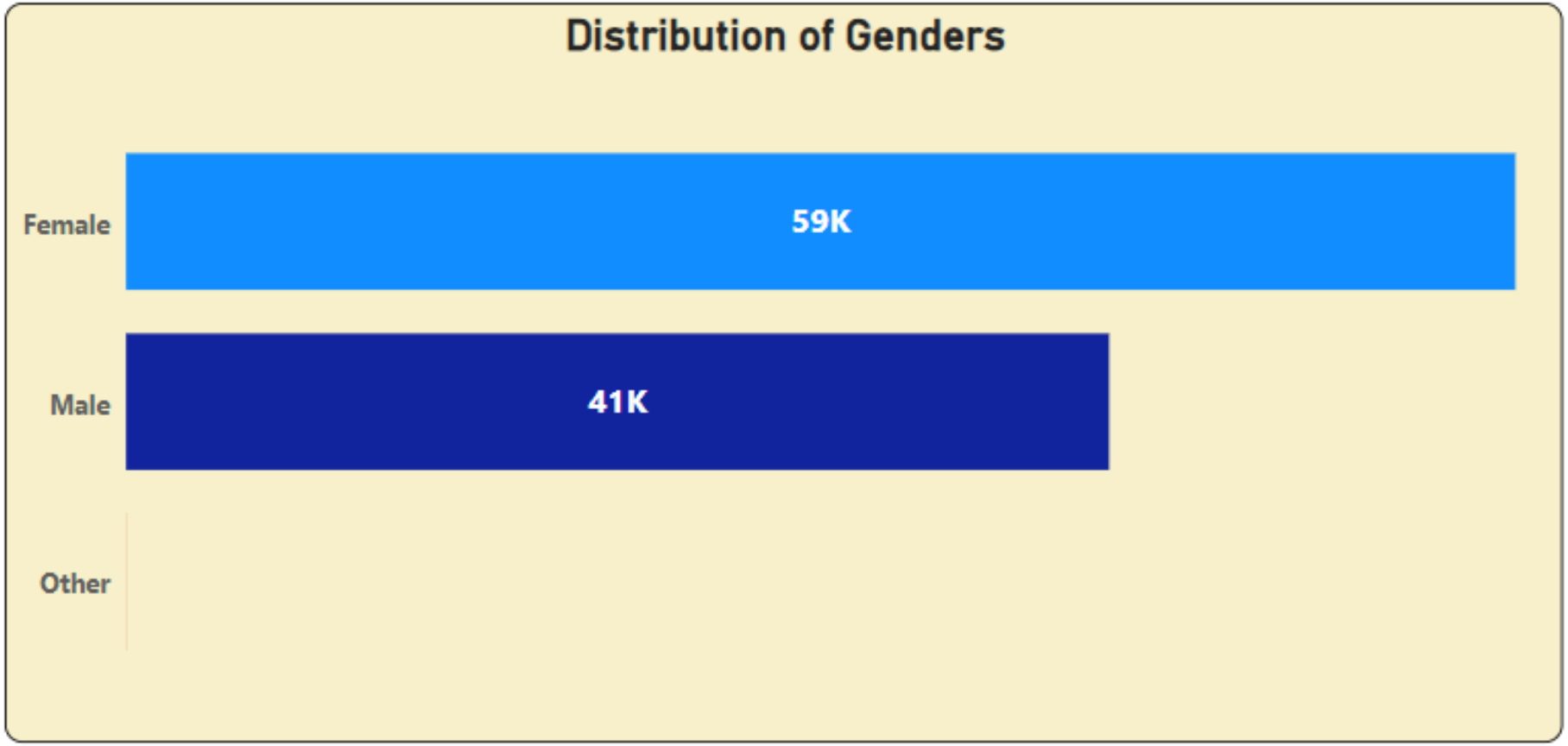
Replace extreme values with less extreme values (e.g., replacing values above the 95th percentile with the value at the 95th percentile). Apply mathematical transformations (e.g., logarithmic transformation) to make the distribution more symmetrical. Remove extreme values if they are likely to be errors or anomalies and not representative of the underlying data distribution.

3. Data Type Conversion:
Ensure that each column has the appropriate data type. For example, numeric columns should be stored as numbers, dates as dates, and categorical variables as text or integers. Use Power BI's data type detection feature to automatically assign data types, but verify and adjust as needed.

4. Handle Inconsistent Data:
Check for inconsistencies in categorical variables, such as different spellings or capitalizations of the same category.

5. Address Data Integrity Issues:
Check for duplicate records and remove them if necessary. Verify referential integrity if the dataset includes relationships between tables.

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



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

DAX stands for Data Analysis Expressions.

It's a formula language used in Power BI, as well as in Excel Power Pivot and SQL Server Analysis Services (SSAS). DAX is designed to perform calculations, manipulate data, and create custom measures and columns within Power BI reports and dashboards.

Here's why it's used:

1. DAX provides a wide range of functions that enable users to perform complex calculations and analysis on their data. These calculations can include aggregations, comparisons, logical operations, and much more.

2. With DAX, users can create custom measures and calculated columns based on their specific business requirements. These measures and columns can be tailored to calculate metrics, ratios, trends, or any other analytical insights necessary for decision-making.

3. DAX is relationship-aware, meaning it understands the relationships between tables in your data model. This allows for seamless integration of calculations across related tables, enabling comprehensive analysis even with complex data structures.

4. DAX expressions can be used to drive dynamic visualizations and interactive reports in Power BI. For example, DAX measures can power KPIs, trend lines, or conditional formatting within visuals, providing users with real-time insights.

5. DAX is optimized for performance within the Power BI environment. It leverages in-memory processing and compression techniques to deliver fast query response times, even when dealing with large datasets.

6. DAX is particularly well-suited for business intelligence (BI) applications, where users need to perform sophisticated calculations and analysis on their data to derive actionable insights and make informed decisions.

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

In Power BI, you can create calculated columns using the DAX (Data Analysis Expressions) language. A calculated column is a column that you add to a table in your data model, where each row's value is calculated based on a DAX formula.

Why a Calculated Column?

1. Calculated columns allow you to derive new information from existing data. For example, you might calculate a total cost column by multiplying quantity and unit price.

2. You can create custom metrics or key performance indicators (KPIs) based on your business requirements.

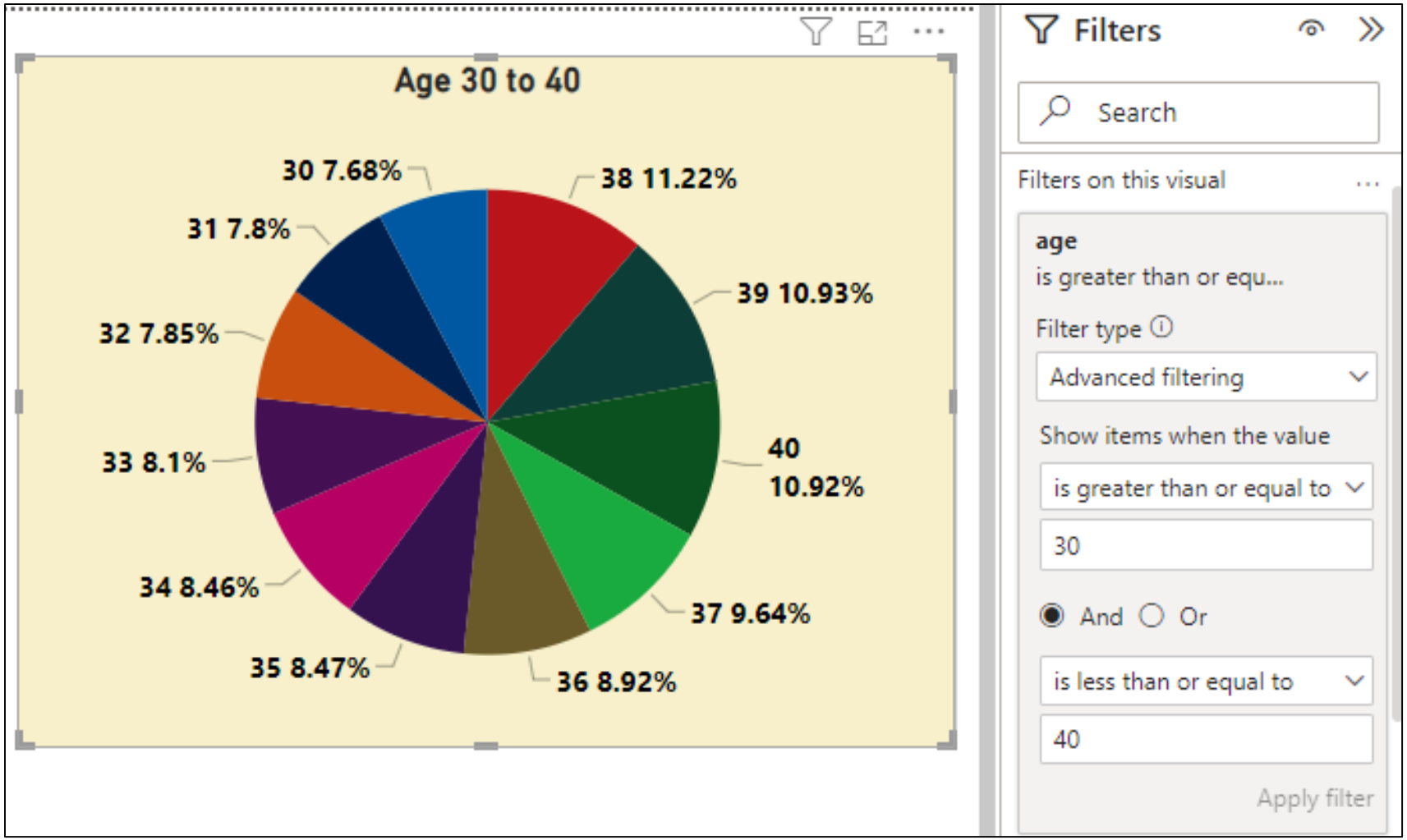
3. Calculated columns can be used for data transformation tasks such as text manipulation, date calculations, or conditional logic.

4. Calculated columns can facilitate filtering and slicing data in visuals and reports.

5. Calculated columns can be used as inputs for visuals, allowing you to visualize calculated metrics or perform further analysis.

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

In the Filters pane →"Advanced filtering" → apply conditions →"Apply filter"



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

- Inner Join:

An inner join returns only the rows where there is a match between the columns being joined from both tables. If there is no match for a row in one of the tables, that row will not be included in the result set.

Use an inner join when you want to retrieve only the records that have matching values in both tables. It's typically used when you're only interested in data that exists in both tables.

- Left Join (or Left Outer Join):

A left join returns all the rows from the left table (the first table specified) and the matched rows from the right table (the second table specified). If there is no match for a row in the right table, NULL values are returned for the columns from the right table.

Use a left join when you want to retrieve all the records from the left table regardless of whether there is a match in the right table. It's useful for situations where you want to include all the data from one table and only matching data from the other table.

Data Modeling: Describe the importance of data modeling in Power BI and how it impacts your visualizations.

1. Data modeling allows you to integrate and combine data from multiple sources into a single coherent model. By defining relationships between tables, you can bring together related datasets and create a unified view of your data, making it easier to analyze and visualize.

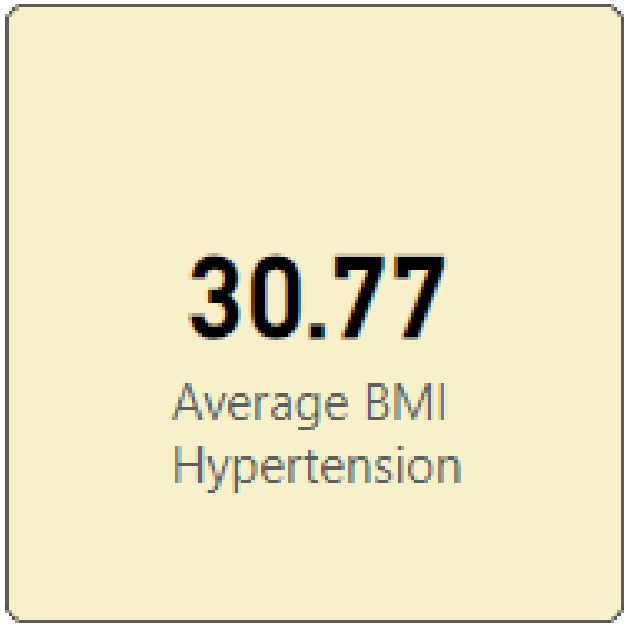
2. In the data modeling process, you can perform data cleansing and transformation operations to ensure that your data is accurate, consistent, and formatted correctly. This may involve tasks such as removing duplicates, handling missing values, standardizing data formats, and creating calculated columns or measures.

3. Effective data modeling can improve the performance of your Power BI reports and dashboards. By optimizing the structure of your data model, including managing relationships and defining appropriate data types and indexing strategies, you can reduce query response times and enhance overall performance, especially when dealing with large datasets.

4. A well-structured and properly modeled dataset provides the foundation for creating clear, accurate, and meaningful visualizations. It ensures that your visualizations accurately represent the underlying data and enable users to interpret and understand the insights effectively

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

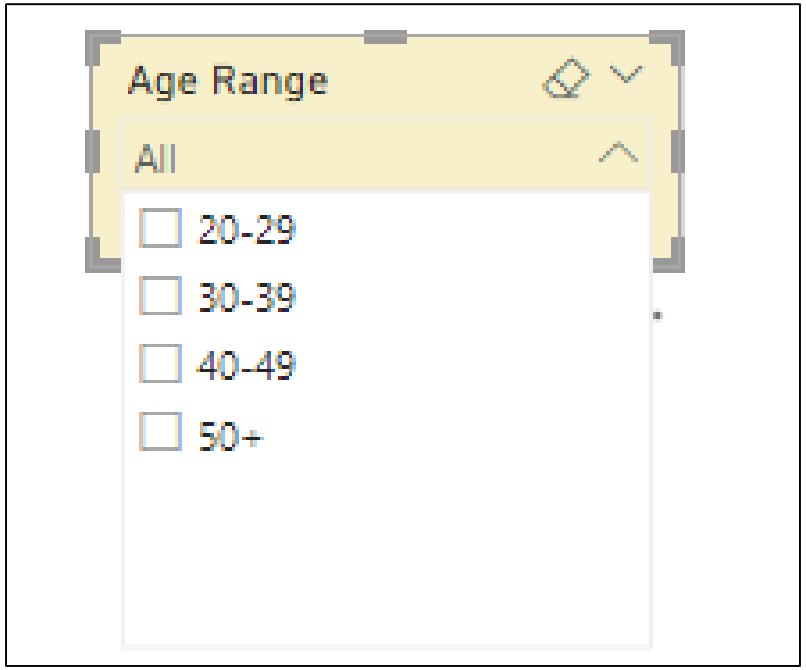
```
Average BMI Hypertension =

AVERAGEX( FILTER(  'Dataset',  'Dataset'[hypertension] = 1 ),  'Dataset'[bmi] )
```



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)?

Create New Column using below DAX

```
Age Range =  SWITCH(  TRUE(),  'Dataset'[age] >= 20 && 'Dataset'[age] < 30, "20-29",  'Dataset'[age] >= 30 && 'Dataset'[age] < 40, "30-39",  'Dataset'[age] >= 40 && 'Dataset'[age] < 50, "40-49",  "50+"  )
```



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

DAX Formula

YTD_Blood_Glucose_Total = CALCULATE(SUM('Dataset [blood_glucose_level]), DATESYTD('Date'[Date]))

Explanation

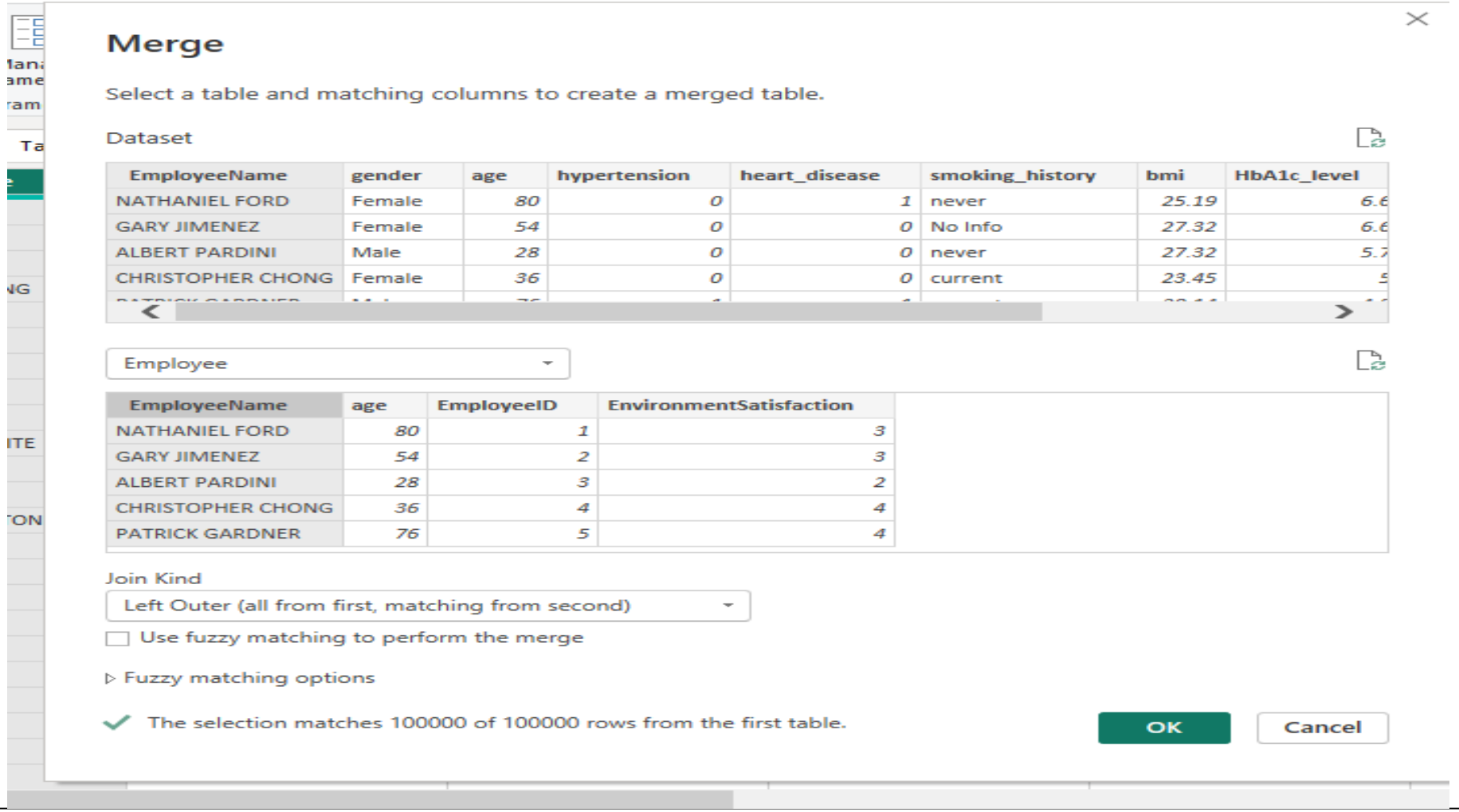
CALCULATE: This function evaluates an expression in a context modified by filters.

SUM(Dataset'[blood_glucose_level]): This calculates the sum of blood glucose levels from the ‘Dataset' table.

DATESYTD('Date'[Date]): This function returns a table of dates for the year-to-date period based on the Date column in the date table. It filters the data to include only dates from the beginning of the current year up to and including the current date.

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

Power Query Editor → Home tab → Merge Query →used left join



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

Purpose of SUMMARIZE:

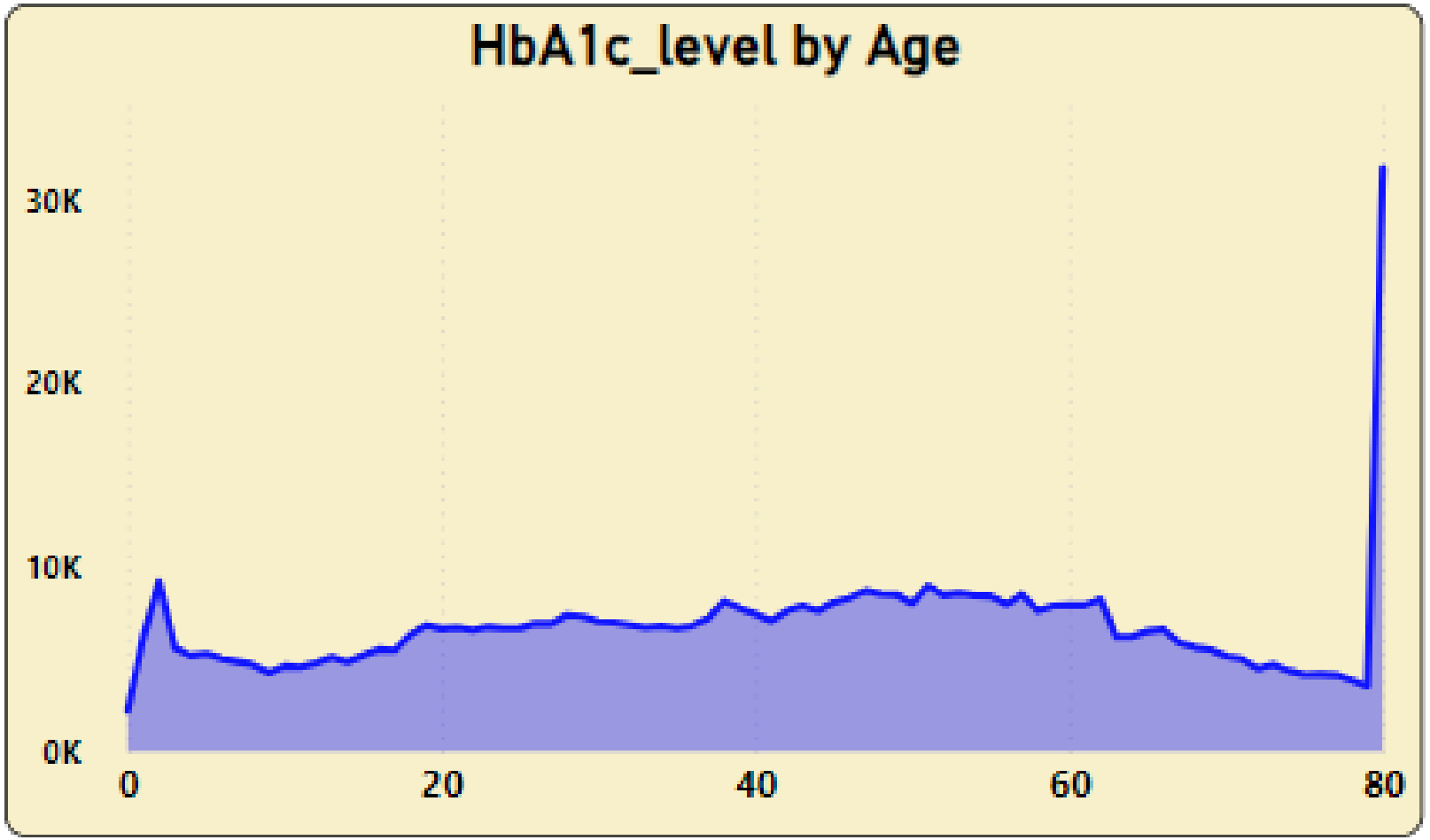
The main purpose of the SUMMARIZE function is to generate summary tables that consolidate data based on grouping criteria. It allows you to summarize and aggregate data from one or more tables based on specified column combinations.

SUMMARIZE is commonly used in DAX to create summary tables for reporting purposes, allowing users to analyze data at different levels of detail and granularity.

How to Use SUMMARIZE to Aggregate Data:

SUMMARIZE (<table>, <group_by_column1>, <group_by_column2>, ..., <expression>)

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



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

1. Data Model Optimization:

Simplify the data model by removing unnecessary tables, columns, and relationships. Reduce the cardinality of relationships by optimizing the granularity of tables. Use calculated columns and measures instead of importing unnecessary columns into the data model.

2. Query Optimization:

Minimize the number of queries sent to the data source by combining multiple queries into one or using query folding. Use direct query or live connection mode for large datasets stored in external databases, instead of importing all data into Power BI. Apply query filters to limit the amount of data retrieved from the data source. Optimize query performance by avoiding complex calculations or transformations in Power Query Editor.

3. Data Loading and Refreshing:

Schedule data refreshes during off-peak hours to reduce the impact on performance. Use incremental data loading techniques to load only new or changed data, rather than reloading the entire dataset every time.

4. Data Reduction Techniques:

Apply data reduction techniques such as aggregation, summarization, or filtering to reduce the volume of data loaded into Power BI. Implement server-side filtering or slicing to offload filtering operations to the data source whenever possible.

5. Use of Indexes and Data Source Optimization:

Ensure that indexes are properly configured on database tables to improve query performance. Optimize the performance of the underlying data source, such as optimizing SQL queries, partitioning tables, and tuning database server settings.

6. Visualization Optimization:

Limit the number of visuals and data points displayed on each page to reduce rendering time. Use slicers, filters, and drill-down functionality to allow users to interactively explore data without loading unnecessary details upfront.

7. Hardware and Infrastructure Optimization:

Ensure that Power BI Desktop and Power BI Service are running on adequately configured hardware with sufficient memory and processing power. Distribute workloads across multiple Power BI reports or datasets to avoid overloading resources.

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

1. Identify the different roles or groups of users who will access the Power BI report. Determine the level of access each role should have to the sensitive employee data.

2. Create one or more tables in your data model that define the security rules for each role. These tables typically contain columns for user IDs or roles and filters to restrict data access based on specific criteria.

3. Establish relationships between the security tables and the main tables containing sensitive employee data. Ensure that these relationships are based on columns that uniquely identify users or roles.

4. Write DAX expressions to define the row-level security filters based on the security tables. Use functions such as USERNAME(), USERPRINCIPALNAME(), or ISINROLE() to identify the current user and determine their access rights. Use functions like FILTER() or CALCULATETABLE() to apply row-level security filters to the main tables based on user permissions.

5. In Power BI Desktop, navigate to the "Modeling" tab and click on "Manage Roles.“ Create a new role for each security role defined earlier. Write DAX expressions to define the filters for each role, based on the security rules established in the security tables.

6. Test row-level security by publishing the report to the Power BI Service and assigning users to their respective security roles. Verify that users can only see the data they are authorized to access based on their roles.

7.Regularly review and update security roles as needed, especially when user access requirements change. Monitor user activity and access patterns to ensure compliance with security policies.

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

```
RollingAverageBloodGlucose =

VAR CurrentDate = 'Date'[Date]

RETURN AVERAGEX(

    FILTER( ALL('Date'),

      'Date'[Date] <= CurrentDate &&

          'Date'[Date] > DATEADD(CurrentDate, -3, MONTH)

              ),

       CALCULATE(AVERAGE('Dataset [blood_glucose_level]) )

   )
```

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

- Explore the Power BI Marketplace and download the custom visual file (usually in .pbiviz format).
- Open your Power BI Desktop → "Visualizations" pane →three dots (...) at the bottom → "Import a custom visual.“ → browse .pbiviz file → Import.
- Once it is visible in Visualization Pane used it in your Dashboard.

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

Cross-filtering is a fundamental concept in Power BI that refers to the behavior where filtering one visual element (such as a chart or table) affects the data displayed in other related visual elements on the same report page. It enables users to interactively explore and analyze data across different visualizations by applying filters dynamically based on their selections.

Cross-filtering impact on report interactivity:

1. If a user selects a specific category in a bar chart representing product sales, other visuals such as a table showing sales details or a line chart displaying sales trends will update to reflect only data related to that selected category.

2. Cross-filtering in Power BI supports bi-directional filtering, meaning that filters applied in one visual can also influence filters applied in other related visuals. Like, if a user applies a filter in a table to show sales data for a particular region, a bar chart showing sales by product category will update to show only data related to that selected region. Similarly, if the user then selects a specific product category in the bar chart, the table will update to show sales data for that category within the selected region.

3. Cross-filtering enables dynamic interactivity within Power BI reports, allowing users to explore and analyze data in real-time by interacting with different visualizations. Users can drill down into specific data points, filter data based on various attributes, and gain deeper insights by visually exploring the relationships between different aspects of the data.

4. By leveraging cross-filtering capabilities, users can quickly identify trends, patterns, and relationships within their data. It facilitates ad-hoc analysis and empowers users to ask and answer questions on the fly, leading to more informed decision-making and actionable insights.

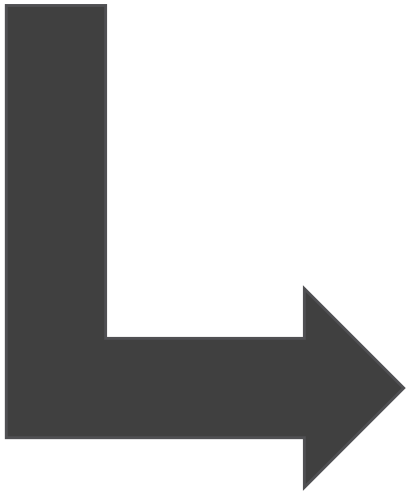
5. Power BI provides options to customize cross-filtering behavior and control how filters propagate between visual elements.

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

Dax Formula

```
DiabetesProbability =

1 / (1+EXP(-(

0.5 * 'Dataset'[Age] +

0.3 * ('Dataset'[Gender] = "Male") +

0.2 * ('Dataset'[BMI]>25) +

0.1 * ('Dataset'[Diabetes] = 1)

)))
```

DIABETES PREDICTION ANALYSIS DASHBOARD



Diabetes Prediction Analysis

Age Range
All

Female | Male | Other

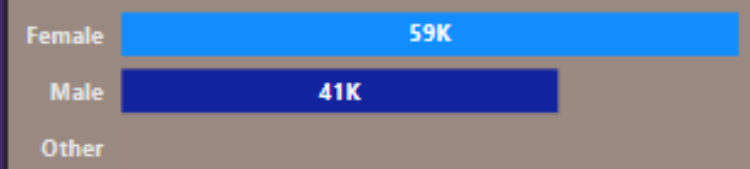
8500
Diabetes Count

31
Avg BMI Hypertensi...

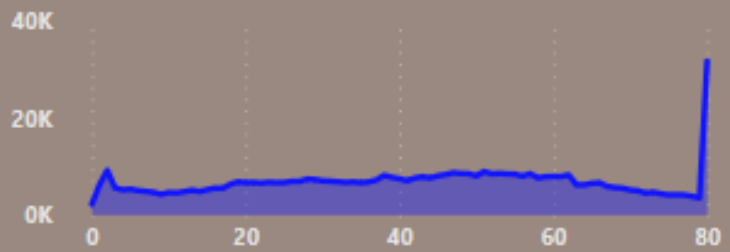
100K
Total Employees

42
Average Age

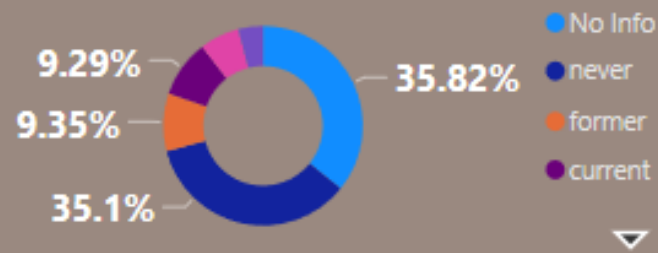
Distribution of Genders



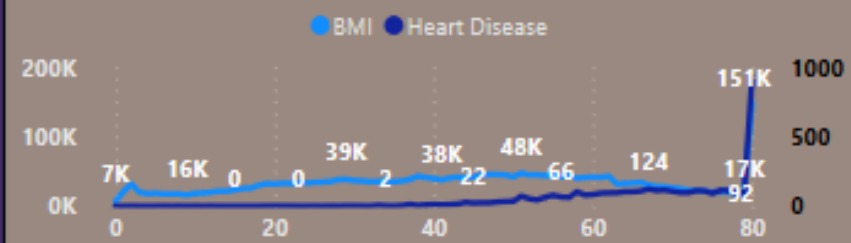
HbA1c_level by Age



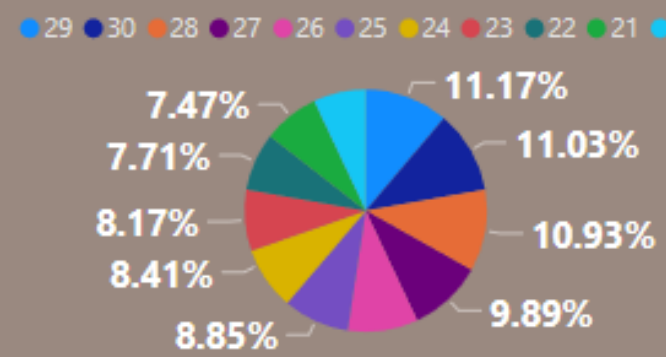
Smoking History



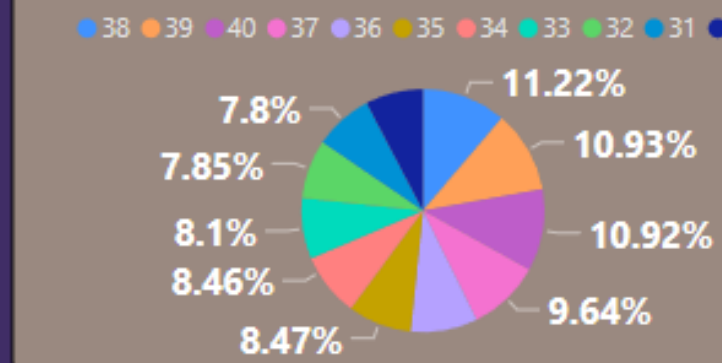
BMI & Heart Disease by Age



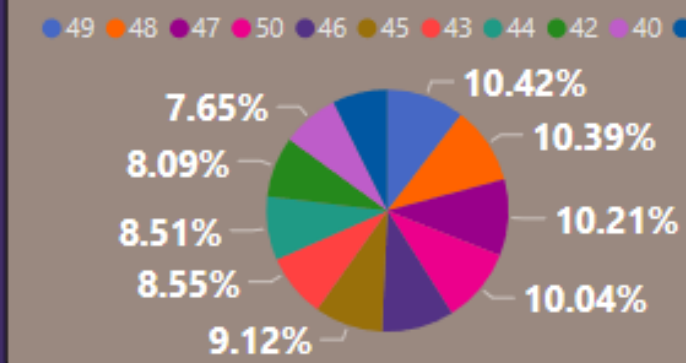
Age 20 to 30



Age 30 to 40



Age 40+



THANK YOU!!!