DIABETES PREDICTION ASSESSMENT

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

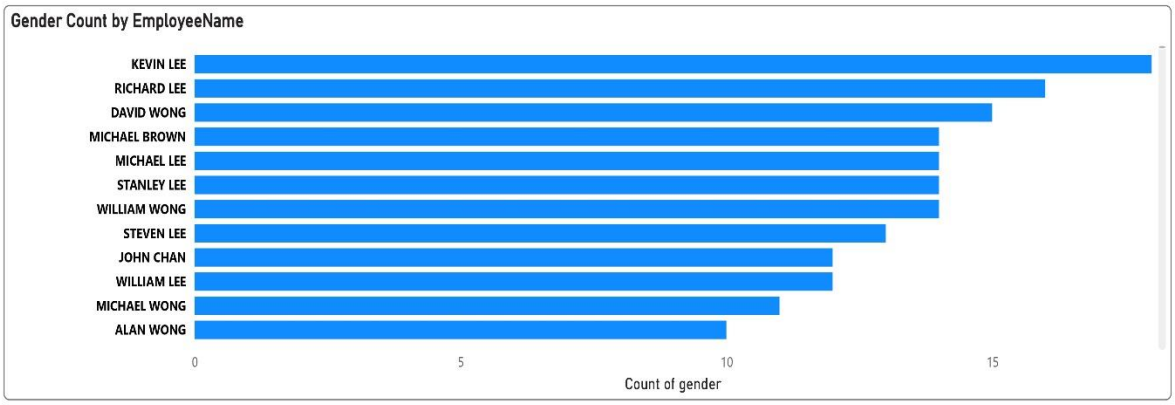
Ans→ Handling missing values or outliers:-

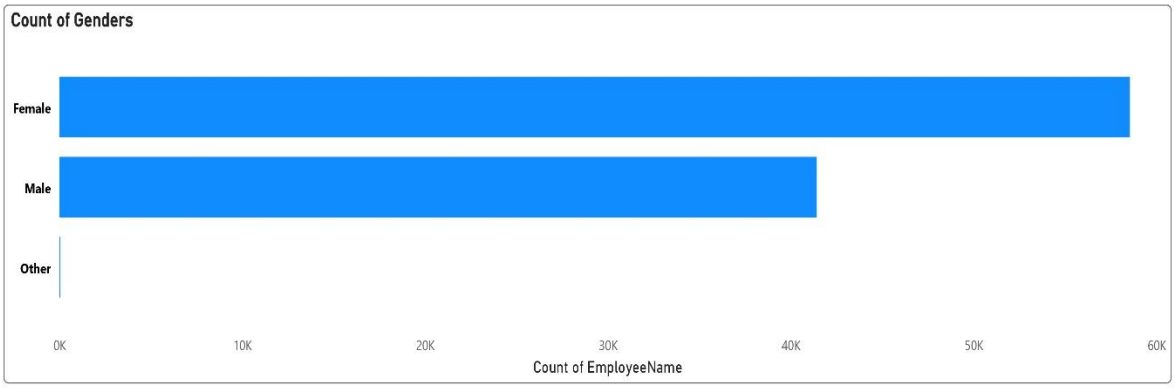
In power, BI uses power query for data transformations to clean the data quickly

- First I will identify missing values by clicking the drop-down for each column to see the missing values
- There are two ways we can handle missing values either we remove rows or attribute values based on context.
- Identify outliers using statistical methods such as the Z score method in Power BI using DAX query to filter out outliers based on a threshold.
    - Z_Score = (give_act_Column - AVERAGE(actual_Column_name)) / stdev.p(YourColumn)
- You can use Box Plots visualizations that can help you visually identify outliers The box plot provides a clear representation of the distribution of data, including potential outliers.
- Remove duplicates, check the spelling of fields and give between the names if needed, and check the whole data it should correct values and fields if something is missing ask your stakeholder
- Filter out the data or use the filter tab, remove unnecessary columns and rows, Most Importantly Check all data types are correct like the date column
- For data cleaning few columns are important Date/Time col, ID col, Target/label col for machine learning, Numeric col, text, col
- Overall consistency should be there if needed create new columns communicate with read the data carefully & data will communicate with you

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

   Ans→ Go to visualization tab → select stacked bar char → put Emp_name col in y-axis and gender col in x-axis customized it based on your need here I have created 2 charts the first gender count by emp_name and the second is the count of genders





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

   Ans→ DAX (Data Analysis Expression):-

   Dax is a formula language used in Power BI, Excel and PowerPoint to create custom calculations for business intelligence and data analysis

   DAX is designed for creating custom calculations and aggregations on data, allowing users to perform complex data analysis and build business intelligence solutions. DAX is crucial for defining relationships between tables, enabling users to create meaningful connections and associations in their data models.

   Key Features:-

Formulas:

- DAX allows the creation of formulas to perform calculations on data, similar to Excel formulas but more powerful in the context of data modeling.
- Time Intelligence:

- DAX includes functions specifically designed for time-based calculations, making it well-suited for handling scenarios involving dates and time.
- Aggregate Functions:

- DAX offers various aggregate functions like SUM, AVERAGE, MIN, MAX, etc., enabling users to summarize and analyze data effectively.
- Filter and Row Context:

- DAX considers both filter context (applied by slicers, and filters) and row context (the current row in a table), allowing for dynamic and context-aware calculations.

Use Cases:

Business Intelligence:

- DAX is extensively used in Power BI for creating interactive reports and dashboards that provide insights into business performance.

- Financial Modelling:- DAX is valuable for financial analysts, allowing them to create complex financial models, calculate key performance indicators (KPIs), and perform scenario analysis.

- Data Modelling:- DAX plays a central role in defining relationships, calculated columns, and measures in data models, ensuring accurate and meaningful representation of data.

Why DAX in Power BI:

- Integration:- DAX is tightly integrated into Power BI, allowing seamless creation of calculations and measures within the Power BI environment.

- Customization:- Users can create customized calculations and metrics using DAX to meet specific business requirements, providing flexibility in reporting and analysis.

- Dynamic Analysis:- DAX enables dynamic analysis based on user interactions, slicers, and filters, making it easier to explore and understand data.

- Consistency Across Platforms:- Since DAX is used in various Microsoft products like Power BI, Excel Power Pivot, and SSAS, users can achieve consistency in calculations and data models across different platforms.

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

Ans→ Go to the "Modeling" tab on the ribbon → Click on "New Column" in the "Calculations" group.

Formula Bar:

- In the formula bar, enter the DAX formula for your calculated column.
- For example: **Total Sales = Sales[Quantity] * Sales[Price]**

Enter Key:

- Press Enter to create the calculated column.

Why You Might Need Calculated Columns:

1. **Derived Data:**
   - Calculated columns allow you to create new columns based on calculations using existing data.
   - Example: Creating a "Total Sales" column by multiplying quantity and price.

2. **Business Logic:**
    - Implement business logic at the column level.
    - Example: Categorizing customers based on their purchasing behavior.
3. **Statistical Calculations:**
    - Perform statistical calculations on existing data.
    - Example: Calculating averages, percentages, or growth rates.
4. **Date Calculations:**
    - Easily create columns for date-related calculations.
    - Example: Extracting the month or year from a date column.
5. **Filters and Slicers:**
    - Calculated columns can be used in filters and slicers.
    - Example: Filtering data based on a calculated category column.
6. **Consistency Across Reports:**
    - Calculated columns ensure consistency in calculations across different visualizations and reports.
7. **Aggregations:**
    - Pre-calculate aggregations to improve performance.
    - Example: Creating a column for cumulative sales to avoid repetitive calculations.
8. **Complex Transformations:**
    - Handle complex transformations that might be challenging with standard queries.
    - Example: Concatenating text fields or creating conditional columns.

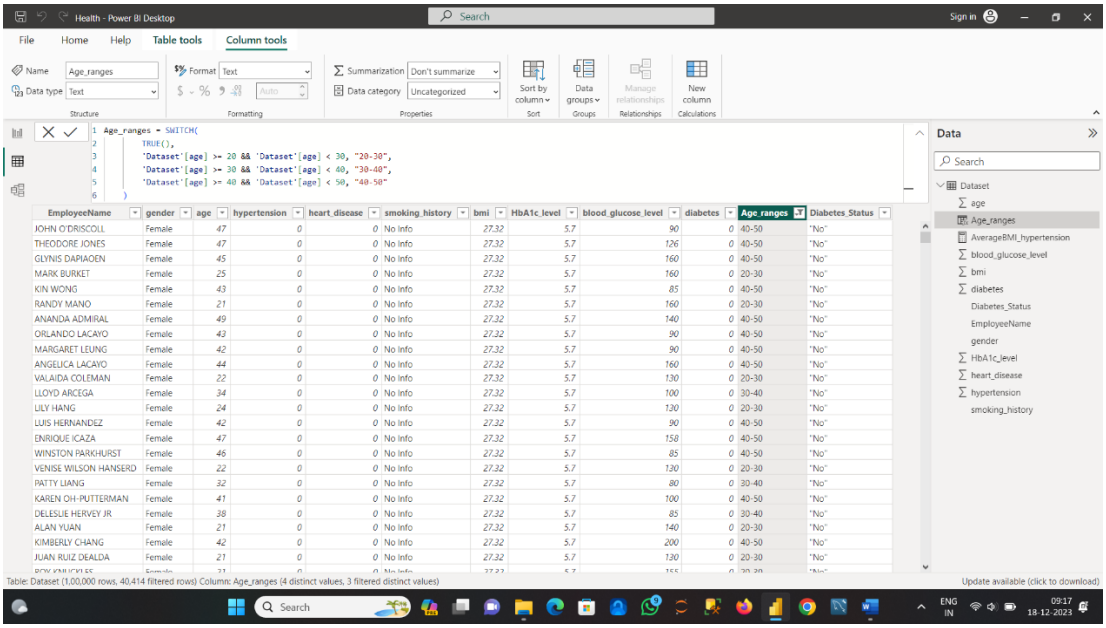
5. Filtering Data: Explain how to filter data to display information for employees aged between 30 and 40.

Ans→ Firstly We Ensure that your data is presented in a table format within Power BI. If you already have a table, you can use it; otherwise, you may need to create one.

1. **Add a Filter:**
    - Click on the table or chart where you want to apply the age filter.
    - On the right side of the Power BI window, you'll find the "Fields" pane. Locate the "age" field (or the field representing employee age).
2. **Apply a Filter:**

- Click on the "age" field, and a dropdown menu will appear.
- Choose "Filter."

3. **Set the Filter Range:**
    - In the filter pane, you can set up your age range. For example, you can set the filter to include values between 30 and 40.
4. **Apply the Filter:**
    - After setting up the filter range, click the "Apply Filter" button.



Power BI will now filter the data based on the age range you specified, and your table or chart will update accordingly. This approach allows you to visually explore the data and create dynamic visualizations that adjust based on the applied filters.

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

Ans→ In Power BI, as in many relational database systems, inner join and left join are types of joins used to combine data from two or more tables. Here's an explanation of the key differences between the inner join and the left join.

1. **Inner Join:**

    An inner join returns only the rows where there is a match in both tables based on the specified join condition.

    - **Result:** The result set includes only the common rows between the tables.

- **Use Case:** Use an inner join when you want to retrieve only the records that have matching values in both tables. It is useful for situations where you're interested in the intersection of data from the participating tables.
  SELECT * FROM Table1 INNER JOIN Table2 ON Table1.ID=Table2.ID;

2. **Left Join (or Left Outer Join):**

   A left join returns all the rows from the left table and the matched rows from the right table. If there is no match, the result will contain NULL values for columns from the right table.

- **Result:** The result set includes all rows from the left table and matching rows from the right table. If there's no match in the right table, NULL values will be filled in.
- **Use Case:** Use a left join when you want to retrieve all records from the left table, regardless of whether there is a match in the right table. It is useful when you want to include unmatched records from the left table in the result set.

  SELECT * FROM Table1 LEFT JOIN Table2 ON Table1.ID = Table2.ID;

When to Use Each:

- Use an **inner join** when you want to focus on the intersection of data and exclude unmatched records.
- Use a **left join** when you want to include all records from the left table, regardless of whether there is a match in the right table.

In Power BI, you perform these joins during the data modeling stage by defining relationships between tables using common fields. The choice between inner join and left join depends on the specific analysis or reporting requirements for your data.

7. **Data Modeling: Describe the importance of data modeling in Power BI and how it impacts your visualizations.**
   Ans→ Data modeling is a crucial aspect of Power BI that significantly impacts the effectiveness and accuracy of your visualizations. Here are key points

   highlighting the importance of data modeling in Power BI:

1. **Data Integration:**
    - Power BI allows you to import data from various sources. Data modeling enables you to integrate and combine data from different tables or sources into a unified view, making it easier to analyze and visualize.
2. **Relationships:**
    - Data modeling involves establishing relationships between tables. These relationships define how the data in one table relates to the data in another. Properly defined relationships ensure accurate and meaningful insights when creating visualizations.
3. **Cardinality and Cross Filter Direction:**
    - Understanding and configuring cardinality (one-to-one, one-to-many, many-to-one) and cross-filter direction (single, both) in relationships is essential. These settings determine how data is filtered and aggregated across tables, influencing the accuracy of your visualizations.
4. **Column and Measure Creation:**
    - Data modeling allows you to create calculated columns and measures. Calculated columns are computed based on data in the table, while measures are calculations that aggregate data. These custom columns and measures enhance the depth and specificity of your analysis.
5. **Hierarchies and Categories:**
    - You can define hierarchies and categories in data modeling. This is particularly useful for organizing and drilling down into data. For example, you can create a date hierarchy to view data at different time levels (year, quarter, month).
6. **Optimizing Performance:**
    - Well-designed data models can significantly improve report performance. Avoiding unnecessary data duplication, optimizing relationships, and using appropriate column types contribute to faster query performance and a more responsive user experience.
7. **Data Cleansing and Transformation:**
    - Power BI provides tools for data cleansing and transformation. Data modeling involves shaping and cleaning the data to ensure accuracy. You can handle missing values, transform data types,

and apply various transformations to prepare the data for analysis.

8. **Security:**
    - Data modeling plays a role in implementing security measures. Row-level security (RLS) can be applied based on user roles and relationships defined in the data model. This ensures that users only see the data relevant to their role.
9. **Scalability and Flexibility:**
    - A well-designed data model is scalable and adaptable. As your data grows or your reporting requirements change, a well-structured model allows for easy expansion and modification without compromising performance.

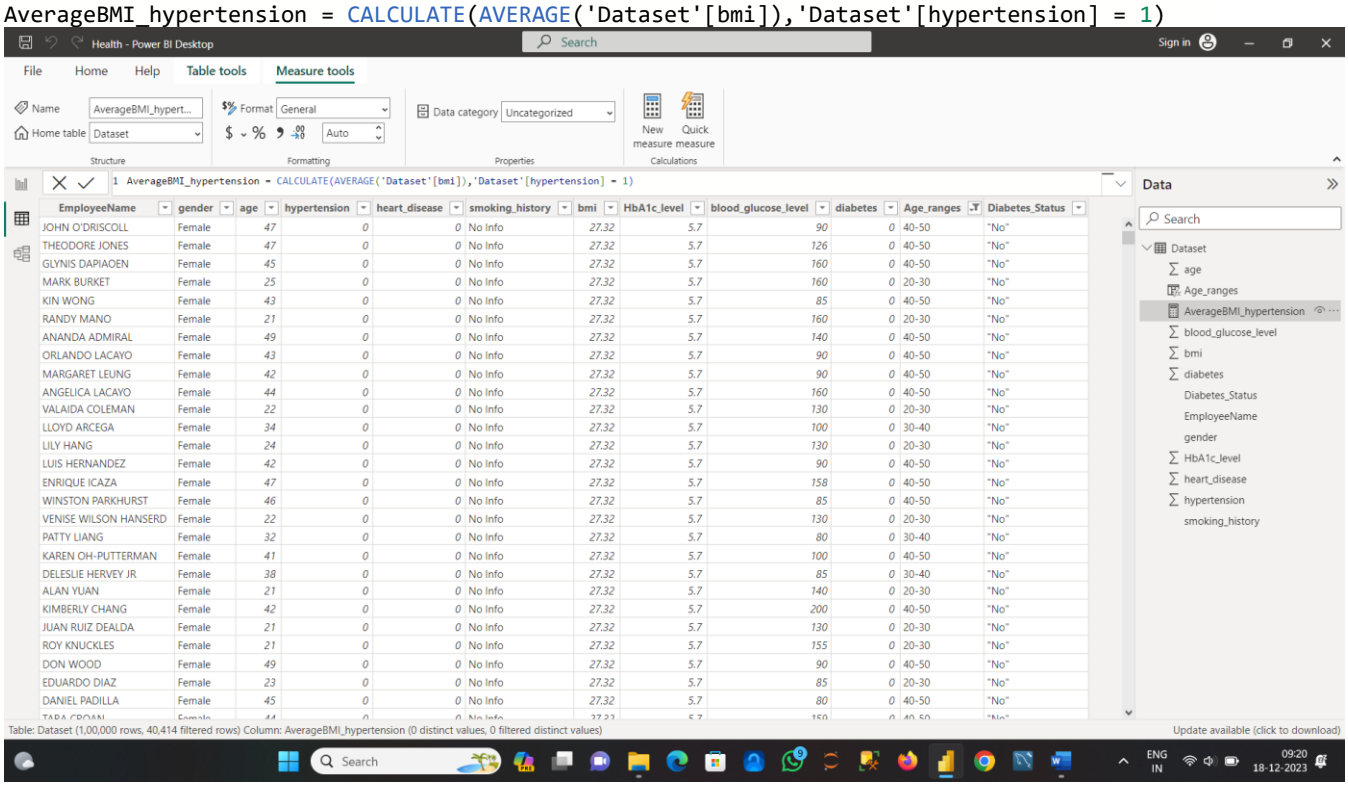
8. Measures in DAX: Create a DAX measure to calculate the average BMI for employees with hypertension.

Ans→ In the Home tab Go to → calculation group → select new measure

Give your name of measure and the formula

To calculate the average BMI for employees with hypertension

```
AverageBMI_hypertension = CALCULATE(AVERAGE('Dataset'[bmi]),'Dataset'[hypertension] = 1)
```

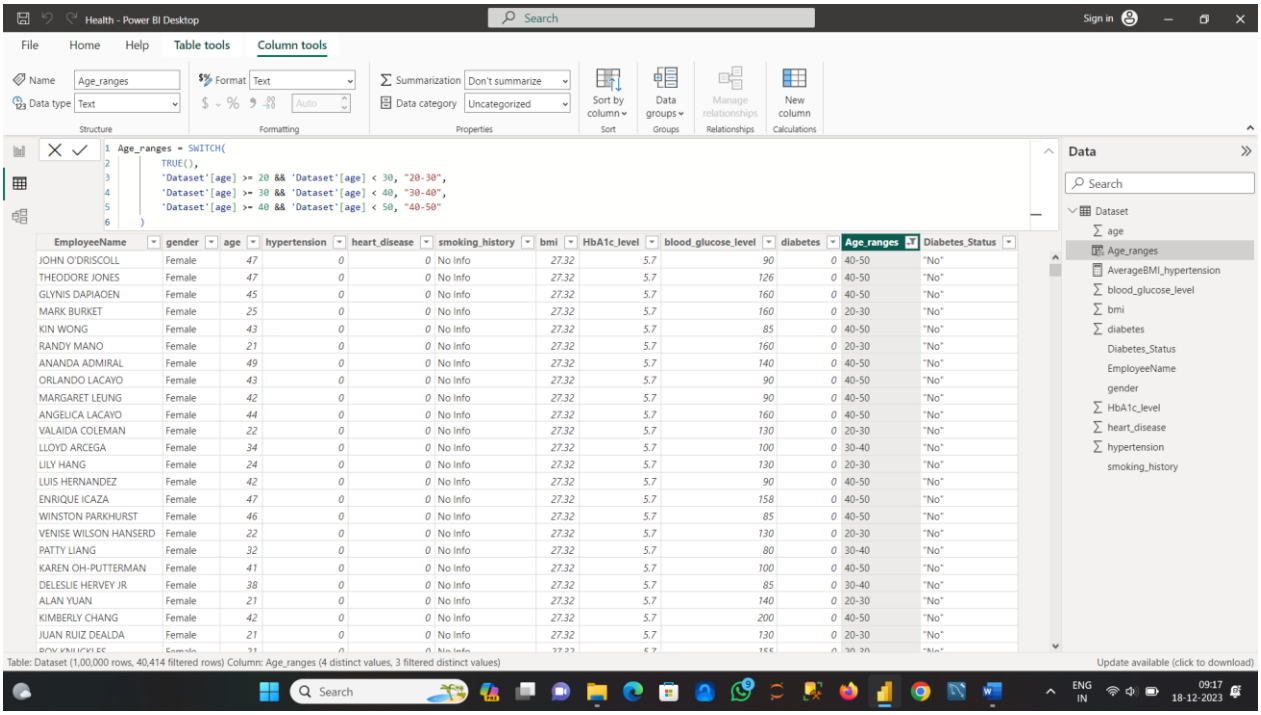


EmployeeName	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes	Age_ranges	Diabetes_Status
JOHN O'DRISCOLL	Female	47	0	0	No Info	27.32	5.7	90	0	40-50	"No"
THEODORE JONES	Female	47	0	0	No Info	27.32	5.7	126	0	40-50	"No"
GLYNIS DAPIAOEN	Female	45	0	0	No Info	27.32	5.7	160	0	40-50	"No"
MARK BURKET	Female	25	0	0	No Info	27.32	5.7	160	0	20-30	"No"
KIN WONG	Female	43	0	0	No Info	27.32	5.7	85	0	40-50	"No"
RANDY MANO	Female	21	0	0	No Info	27.32	5.7	160	0	20-30	"No"
ANANDA ADMIRAL	Female	49	0	0	No Info	27.32	5.7	140	0	40-50	"No"
ORLANDO LACAYO	Female	43	0	0	No Info	27.32	5.7	90	0	40-50	"No"
MARGARET LEUNG	Female	42	0	0	No Info	27.32	5.7	90	0	40-50	"No"
ANGELICA LACAYO	Female	44	0	0	No Info	27.32	5.7	160	0	40-50	"No"
VALAIDA COLEMAN	Female	22	0	0	No Info	27.32	5.7	130	0	20-30	"No"
LLOYD ARCEGA	Female	34	0	0	No Info	27.32	5.7	100	0	30-40	"No"
LILY HANG	Female	24	0	0	No Info	27.32	5.7	130	0	20-30	"No"
LUIS HERNANDEZ	Female	42	0	0	No Info	27.32	5.7	90	0	40-50	"No"
ENRIQUE ICAZA	Female	47	0	0	No Info	27.32	5.7	158	0	40-50	"No"
WINSTON PARKHURST	Female	46	0	0	No Info	27.32	5.7	85	0	40-50	"No"
VENISE WILSON HANSERD	Female	22	0	0	No Info	27.32	5.7	130	0	20-30	"No"
PATTY LIANG	Female	32	0	0	No Info	27.32	5.7	80	0	30-40	"No"
KAREN OH-PUTTERMAN	Female	41	0	0	No Info	27.32	5.7	100	0	40-50	"No"
DELESLIE HERVEY JR	Female	38	0	0	No Info	27.32	5.7	85	0	30-40	"No"
ALAN YUAN	Female	21	0	0	No Info	27.32	5.7	140	0	20-30	"No"
KIMBERLY CHANG	Female	42	0	0	No Info	27.32	5.7	200	0	40-50	"No"
JUAN RUIZ DEALDA	Female	21	0	0	No Info	27.32	5.7	130	0	20-30	"No"
ROY KNUCKLES	Female	21	0	0	No Info	27.32	5.7	155	0	20-30	"No"
DON WOOD	Female	49	0	0	No Info	27.32	5.7	90	0	40-50	"No"
EDUARDO DIAZ	Female	23	0	0	No Info	27.32	5.7	85	0	20-30	"No"
DANIEL PADILLA	Female	45	0	0	No Info	27.32	5.7	80	0	40-50	"No"

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)?**

   Ans→ Create new Age ranges column → give name to the column →Tell your condition in the Dax query bar

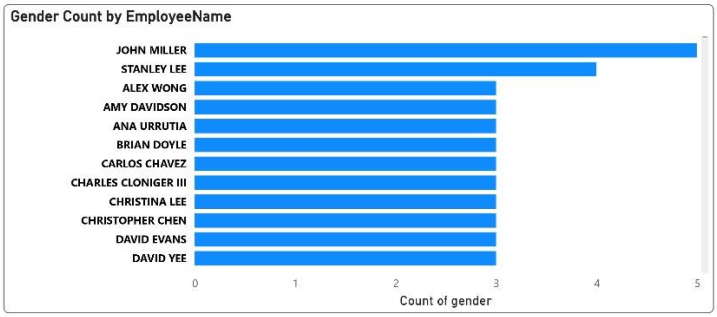
```
Age_ranges = SWITCH(
        TRUE(),
        'Dataset'[age] >= 20 && 'Dataset'[age] < 30, "20-30",
        'Dataset'[age] >= 30 && 'Dataset'[age] < 40, "30-40",
        'Dataset'[age] >= 40 && 'Dataset'[age] < 50, "40-50"
    )
```

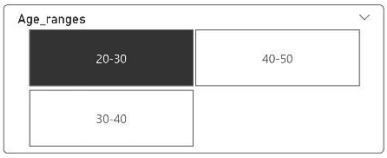


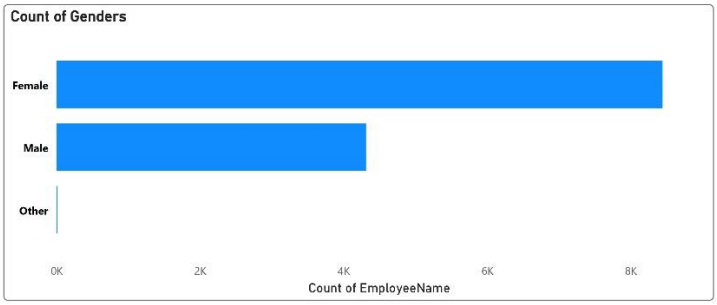
→ Now click on Age_ranges filter Drop down icon & filter it out or in the filter tab

➔ Drag the slicer chart in the canvas → put age_ranges col in the field

➔ Adjust the setting in the slicer setting







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

Ans→

1. **Create a Date Table (if not already available):**
   - It's good practice to have a separate date table in your Power BI model. If you don't have one, create a date table that includes a column for the date (**Date**) and a column for the year (**Year**).
2. **Create a Relationship:**
   - Ensure that there is a relationship between the date column in your main data table (containing blood glucose levels) and the date column in the date table.
3. **Create a YTD Total Measure:**
   - In Power BI Desktop, go to the "Data" view.
   - Click on "New Measure" in the Modeling tab.
   - Use the following DAX formula to create a YTD total measure:

```
YTD Blood Glucose Total = TOTALYTD( SUM('Dataset'[blood_glucose_level]),
    'Dataset'[Date] )
```

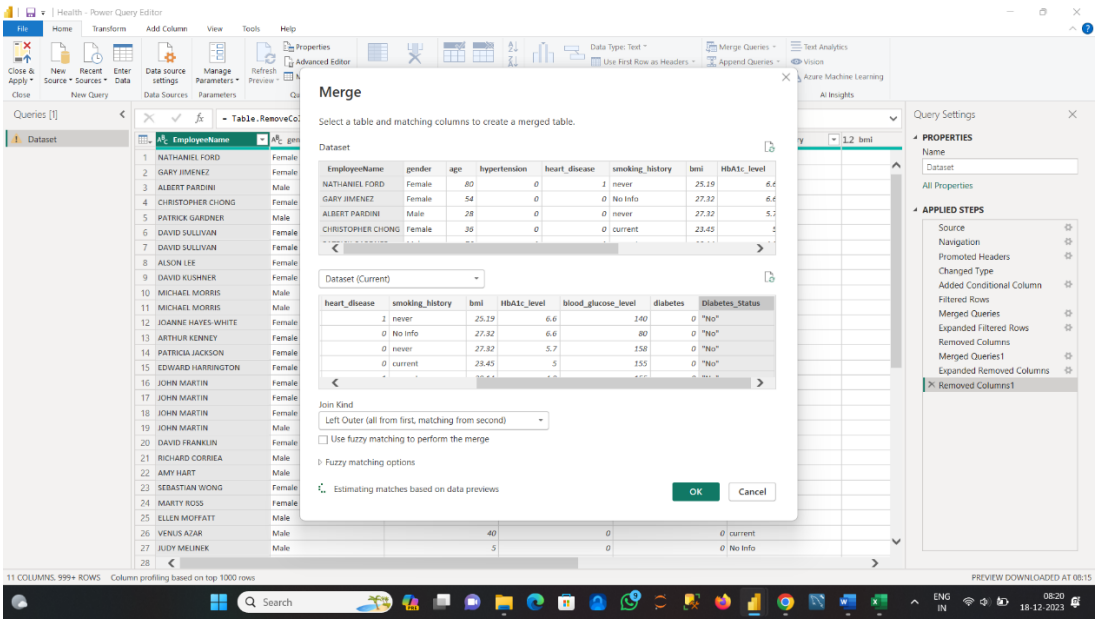
4. **Display the YTD Total in a Visual:**

- Go to the "Report" view in Power BI.
- Drag the newly created measure, **YTD Blood Glucose Total**, to a table, card, or any other visual where you want to display the YTD total.

Now, the measure **YTD Blood Glucose Total** will dynamically calculate the year-to-date total for blood glucose levels based on the date selection in your report.

This approach uses the **TOTALYTD** function, which aggregates the blood glucose levels from the beginning of the year up to the selected date. The relationship with the date table ensures that the calculation is performed correctly based on the data context in your report.

11. Complex Joins: Combine data from multiple tables, including employee data and diabetes data, using appropriate joins.
    Ans→ Go to Power Query Editor → In Home Tab → Click on “Merge Queries” → Choose the tables employee data and diabetes
    I have created a new column named Diabetes Status to merge



    Here I have used Left join because I want all records from the left column and only matching records on the right

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

Ans→ The **SUMMARIZE** function in DAX (Data Analysis Expressions) is used for data summarization and aggregation. Its primary purpose is to create a summary table or virtual table based on specified grouping criteria and aggregate functions. This function is commonly used in Power BI and other tools that support DAX for building meaningful reports and visualizations.

purpose of **SUMMARIZE** and how you can use it to aggregate data:

1. **Create Summary Tables:**
    - **SUMMARIZE** is used to create summary tables that provide aggregated information based on specified columns. This is particularly useful when you want to analyze data at a higher level of granularity than the original data.

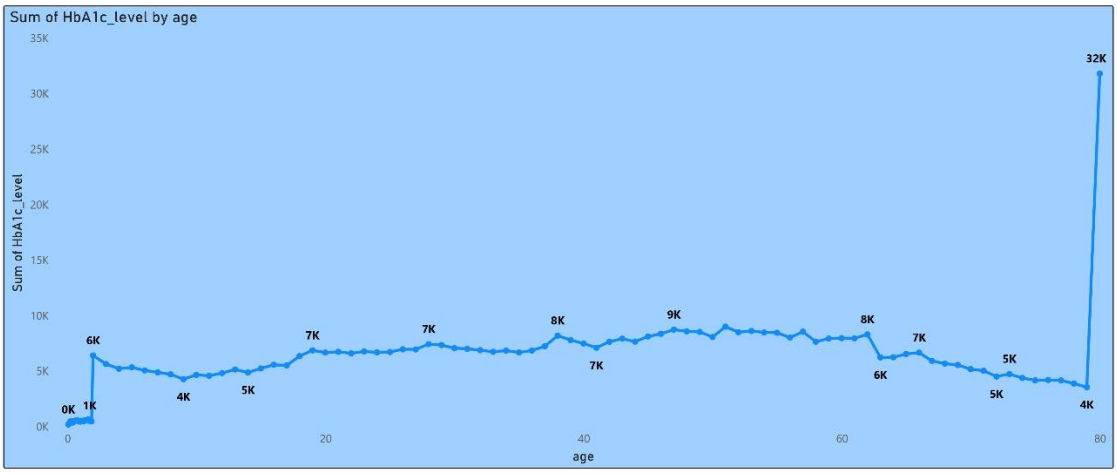
2. **Grouping Data:**
    - You can use **SUMMARIZE** to group your data based on one or more columns. These columns define the criteria for summarization.
3. **Aggregating Data:**
    - **SUMMARIZE** allows you to include aggregate functions (e.g., **SUM**, **COUNT**, **MIN**, **MAX**, **AVERAGE**, etc.) to perform calculations on the grouped data. These aggregate functions operate on the columns specified in the **SUMMARIZE** function.
4. **Creating Summary Tables:**
    - The result of the **SUMMARIZE** function is a virtual table that contains unique combinations of the grouping columns and the calculated values based on the aggregate functions. This summary table is useful for creating concise and informative reports.

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

    Ans→ Go to visualization tab → select line chart → put HbA1c in the y-axis

    → Put age in the x-axis

    No time/Date column exists that's why I put age in the axis



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

Ans→

Performance optimization in Power BI is crucial for handling large datasets. Here are some key techniques:

1. **Data Model Optimization:**
    - Simplify and optimize your data model by removing unnecessary columns and tables.
    - Use relationships wisely, and consider hiding unnecessary fields.
2. **Data Import Settings:**
    - Load only the necessary data into Power BI to reduce memory usage.
    - Utilize filters in Power Query to import a subset of data.
3. **Data Types and Cardinality:**
    - Choose appropriate data types to minimize memory consumption.
    - Optimize cardinality in relationships to reduce unnecessary duplication.

4. **Indexing:**
    - In the data source, ensure that your database tables have proper indexing.
    - Indexing can significantly improve query performance.
5. **Query Folding:**
    - Leverage query folding in Power Query to push operations back to the data source. This reduces the amount of data transferred to Power BI.
6. **Aggregations:**
    - Create pre-aggregated tables to speed up queries.
    - Use the Aggregations feature to define summary tables for large datasets.
7. **DAX Optimization:**
    - Write efficient DAX measures.
    - Avoid unnecessary iterations and use functions like CALCULATE and FILTER judiciously.
8. **Use of Calculated Columns vs. Measures:**
    - Prefer measures over calculated columns for aggregations.
    - Calculated columns consume more memory.
9. **Data Compression:**
    - Compress your data to reduce the amount of memory required.
    - Utilize the column and rowar-level compression capabilities.
10. **Report Design:**
    - Limit visuals on a single report page to reduce rendering time.
    - Optimize visuals for the right level of detail based on user needs.
11. **Incremental Data Refresh:**
    - Use incremental data refresh for large datasets to update only the new or changed data.

- This reduces the refresh time.

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

Ans→ Row-Level Security (RLS) in Power BI is a feature that allows you to control data access here you can implement row-level security:-

1. **Define Roles:**- Identify different user roles, such as managers, employees, or executives.
2. **Create RLS Rules:**- Define rules for each role to filter data based on specific criteria.
    - For example, a manager may have access to all data, while an employee sees only their team's data.
3. **Set Up Roles in Power BI Desktop:**- In Power BI Desktop, go to "Modeling" and click on "Manage Roles."
    - Create roles and define rules using DAX expressions.
4. **Publish to Power BI Service:**
    - Save and publish the Power BI report to the Power BI service.
5. **Configure Security in Power BI Service:**- In the Power BI service, go to the dataset settings.
    - Under "Security," add users or groups to specific roles.
6. **Test RLS:**- Sign in with different user accounts to verify that RLS is working as expected.
    - Users will only see data allowed by their assigned role.

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

Ans→ Create a new Column → write your Dax Query

```
RollingAverageBloodGlucose =
CALCULATE(
    AVERAGE('Dataset'[blood_glucose_level]),
    FILTER(
        ALL('Dataset'),
        'Dataset'[Date] > EARLIER('Dataset'[Date]) - 90 &&
        'Dataset'[Date] <= EARLIER('Dataset'[Date])
    )
)
```

Here there was no date column, so I just wrote the formula

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

Ans→ **Importing Custom Visuals:**

1. **Download the Custom Visual File:**
    - Custom visuals are typically distributed as **.pbiviz** files.
    - Download the custom visual file from a reliable source, such as the Microsoft AppSource or a trusted community repository.
2. **Open Power BI Desktop:**
    - Launch Power BI Desktop on your computer.
3. **Navigate to the "Visualizations" Pane:**
    - In Power BI Desktop, look for the "Visualizations" pane on the right side of the screen.
4. **Click on "Import a Custom Visual":**
    - Click on the ellipsis (...) in the "Visualizations" pane to open the options menu.
    - Select "Import a custom visual."
5. **Choose the Custom Visual File:**

- Browse to the location where you downloaded the custom visual file (.pbiviz).
- Select the file and click "Open" to import it.

6. **Confirm Import:**
    - Power BI will import the custom visual, and you'll see it listed in the "Visualizations" pane.

Using Custom Visuals in Your Report:

1. **Drag the Custom Visual to the Report Canvas:**
    - Locate the imported custom visual in the "Visualizations" pane.
    - Drag and drop it onto the report canvas.
2. **Configure the Custom Visual:**
    - Customize the visual by configuring its fields, formatting options, and other settings.
    - Use the "Fields" pane to associate data fields with the visual.
3. **Adjust Visual Properties:**
    - Depending on the custom visual, you may have additional formatting and customization options available in the "Visualizations" pane.
4. **Interact with the Visual:**
    - If the custom visual supports interactions, you can click, hover, or perform other actions based on its capabilities.
5. **Refresh Data:**
    - After configuring the visual, ensure your report's data is up-to-date by refreshing the data.
6. **Save the Report:**
    - Save your Power BI report to retain the custom visual configurations.

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

Ans→ **Cross-Filtering in Power BI:-**

Cross-filtering is a concept in Power BI that defines the way filters applied to one visual element affect the data displayed in other related visual elements. It enables interactivity between visuals on a report page.

Impact on Report Interactivity:

1. **Bi-Directional Filtering:**
    - Cross-filtering supports both single-direction and bi-directional filtering.
    - In single-direction, filtering in one visual affects another, but not vice versa.
    - In bi-directional, the filter flows in both directions, affecting each other.
2. **Enhanced Exploration:**
    - Users can interactively explore and analyze data by clicking on elements in one visual, and related visuals respond accordingly.
    - For example, clicking on a bar in a chart can filter a table to show details related to that specific bar.
3. **Dynamic Drill through:**
    - Cross-filtering facilitates dynamic drill-through.
    - Users can drill through from summary visuals to detailed visuals, and the detailed visuals automatically adjust based on the selected data point.
4. **Contextual Insights:**
    - Visuals become context-aware as filters applied in one visual provide context to others.
    - This helps users gain insights within the specific context of their analysis.
5. **Enhanced User Experience:**

- Cross-filtering enhances the overall user experience by providing a more interactive and dynamic exploration of data.
- It enables users to answer ad-hoc questions and perform on-the-fly analysis.

6. **Complex Relationships:**
    - Cross-filtering works in conjunction with relationships defined in the data model.
    - It allows users to leverage complex relationships to filter data across different tables.
7. **Custom Interactivity:**
    - Users can create custom interactions between visuals by defining specific cross-filtering behaviors.
    - This allows for tailored and intuitive report interactions.

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

Ans→ Go to tools table → select new measure

```
DiabetesProbability =
VAR AgeCoef = 0.02
VAR GenderCoef =
    IF (
        VALUES('dataset'[gender]) = "Male",
        0.5,
        0
    )
VAR BMI_Coef = 0.03
VAR HypertensionCoef =
    IF (
        VALUES('dataset'[hypertension]) = 1,
        0.8,
        0
    )
VAR HeartDiseaseCoef =
    IF (
        VALUES('dataset'[heart_disease]) = 1,
        1,
        0
```

```
    )
VAR SmokingCoef =
    SWITCH (
        TRUE (),
        VALUES('dataset'[smoking_history]) = "current", 0.9,
        VALUES('dataset'[smoking_history]) = "former", 0.5,
        VALUES('dataset'[smoking_history]) = "ever", 0.7,
        0
    )
VAR Intercept = -5
VAR LinearPredictor =
    AgeCoef * VALUES('dataset'[age]) +
    GenderCoef +
    BMI_Coef * VALUES('dataset'[bmi]) +
    HypertensionCoef +
    HeartDiseaseCoef +
    SmokingCoef +
    Intercept
VAR Odds = EXP (LinearPredictor)
RETURN
    Odds / (1 + Odds)
```