**Scenario:** A product-based software company handles customer queries through a Customer Service portal. All the users raise Incidents if they encounter any problem with the product. So, the company wants to know the statistics and reports through a dashboard to understand their employee performance and identify areas for improvement.

**Problem Statement**: The problem is that the company needs a database to create a dashboard. So, the company hires a data engineer to create a database management system in SSMS, handle the data orchestration with the existing data first, and then make a visualization dashboard through Power BI and draw required insights.

The formulation of a dashboard via Power BI entails the following procedural steps:

* **SSMS and SQL Server Installation:** Download and Install SQL Server Management Studio and SQL server. Set up SQL server Connection in SSMS using credentials or windows authentication (Skip this step if already installed).
* **Database Creation:** Create a new Database named [Incident\_Data]
* **Tables Creation:** Create tables in the following order using below create queries.
  + *Tables Names -* [Department, Category, Incident\_Status, Employee, Incident]
  + *Department:* [Id (*Primary Key*), Department\_Name]

CREATE TABLE Department (

Id INT PRIMARY KEY,

Department\_Name VARCHAR(255) NOT NULL

);

* + *Category:* [Id (*Primary Key*), Issue\_Type]

CREATE TABLE Category (

Id INT PRIMARY KEY,

Issue\_Type VARCHAR(255) NOT NULL

);

* + *Incident\_Status:* [Id (*Primary Key*), Status]

CREATE TABLE Incident\_Status (

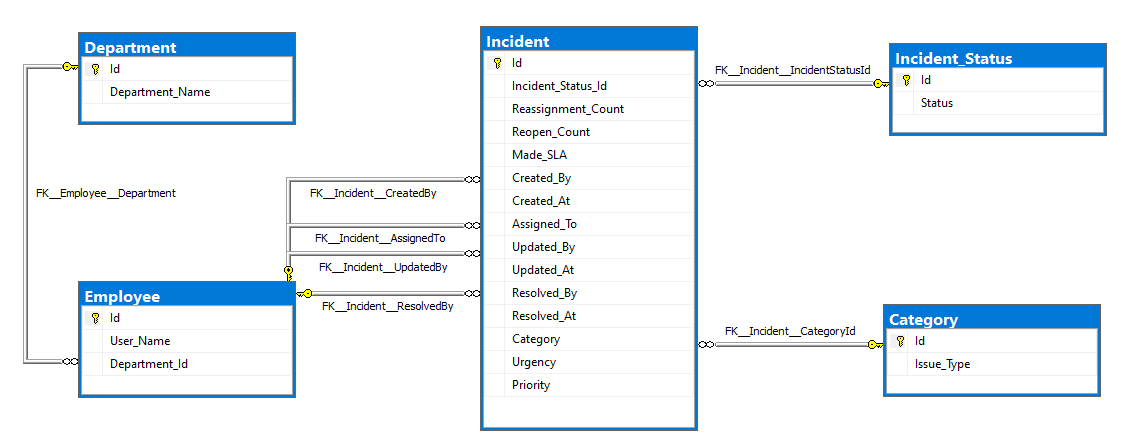
Id INT PRIMARY KEY,

Status VARCHAR(255) NOT NULL

);

* + *Employee:* [Id (*Primary Key*), User\_Name, Department\_Id]
    - ​​Employee[Department\_Id] -> Department[Id]

* + CREATE TABLE Employee (
  + Id INT PRIMARY KEY,
  + User\_Name VARCHAR(255) NOT NULL,
  + Department\_Id INT NOT NULL,
  + FOREIGN KEY (Department\_Id) REFERENCES Department(Id)
  + );
  + *Incident:* [Id (*Primary Key*), Incident\_Status\_Id, Reassignment\_Count, Reopen\_Count, Made\_SLA, Created\_By, Created\_At, Assigned\_To, Updated\_By, Updated\_At, Resolved\_By, Resolved\_At, Category, Urgency, Priority]
    - Incident[Created\_By] -> Employee[Id]
    - Incident[Assigned\_To] -> Employee[Id]
    - Incident[Updated\_By] -> Employee[Id]
    - Incident[Resolved\_By] -> Employee[Id]
    - Incident[Category] -> Category[Id]
    - Incident[Incident\_Status\_Id] -> Incident\_Status[Id]
  + CREATE TABLE Incident (
  + Id VARCHAR(255) PRIMARY KEY NOT NULL,
  + Incident\_Status\_Id INT NOT NULL,
  + Reassignment\_Count INT,
  + Reopen\_Count INT,
  + Made\_SLA VARCHAR(255),
  + Created\_By INT NOT NULL,
  + Created\_At VARCHAR(255) NOT NULL,
  + Assigned\_To INT,
  + Updated\_By INT,
  + Updated\_At VARCHAR(255),
  + Resolved\_By INT NULL,
  + Resolved\_At VARCHAR(255) NULL,
  + Category INT NOT NULL,
  + Urgency VARCHAR(255) NOT NULL,
  + Priority VARCHAR(255) NOT NULL,
  + FOREIGN KEY (Incident\_Status\_Id) REFERENCES Incident\_Status(Id),
  + FOREIGN KEY (Created\_By) REFERENCES Employee(Id),
  + FOREIGN KEY (Assigned\_To) REFERENCES Employee(Id),
  + FOREIGN KEY (Updated\_By) REFERENCES Employee(Id),
  + FOREIGN KEY (Resolved\_By) REFERENCES Employee(Id),
  + FOREIGN KEY (Category) REFERENCES Category(Id)
  + );
* **ER Diagram:** Design an ER diagram in SSMS using ‘Database diagrams’



* **Data Insertion:** Insert provided .csv files using ‘Import Wizard’ in SSMS with ‘Flat File’ as Data Source and ‘Microsoft OLE DB Provider for SQL Server’ as Destination Source, then verify the data insertion.



* **Data Manipulation:**
  + First Update all rows of ‘Resolved\_At’ to ‘NULL’ in the ‘Incident’ Table where ‘Resolved\_By’ value is ‘0’.

Update Incident set Resolved\_At = NULL where Resolved\_By = 0

* + Next, Update all rows of ‘Resolved\_By’ to ‘NULL’ in the ‘Incident’ Table where ‘Resolved\_By’ value is ‘0’.

Update Incident set Resolved\_By = NULL where Resolved\_By = 0

* + Now, Insert (3142, ‘Jane Foster’, 3) details into the ‘Employee’ table.

Insert into Employee(Id, User\_Name, Department\_Id) Values (3142,'Jane Foster', 3)

* + Verify Insertion using select query.

select \* from Employee where id=3142

* + Delete that record from the table.

Delete from Employee where Id=3142

* **Data Analysis:** Examine different metrics and trends within the dataset to gain insights and identify patterns. This involves calculating key performance indicators, analyzing data based on various categories, and identifying areas for improvement. The goal is to extract valuable insights that can inform decision-making and drive operational efficiency.

|  |
| --- |
| --Total Incident Count: Count of Distinct Incidents SELECT Count(DISTINCT Id) AS 'No of Incidents' FROM incident    --Total Active Incident Count: Count of Distinct Incidents which are Currently not resolved SELECT Count(DISTINCT Id) AS 'No of Active Incidents' FROM incident WHERE incident\_status\_id <> 6    --Total resolved Incident Count: Count of Distinct Incidents which are resolved SELECT Count(DISTINCT Id) AS 'No of Resolved Incidents' FROM incident WHERE incident\_status\_id = 6    --Total Incident Count Priority: Count of Distinct Incidents categorized by priority. SELECT priority,  Count(DISTINCT Id) AS 'No of Incidents' FROM incident GROUP BY priority    --Total Incident Count Status: Count of Distinct Incidents categorized by Status. SELECT i\_s.status,  Count(DISTINCT i.Id) AS 'No of Incidents' FROM incident i  INNER JOIN incident\_status i\_s  ON i\_s.id = i.incident\_status\_id GROUP BY i\_s.status    --Total resolved Incident Count Issue Type: Count of Distinct Incidents categorized by Issue Type. SELECT c.issue\_type,  Count(DISTINCT i.Id) AS 'No of Incidents' FROM incident i  INNER JOIN category c  ON c.id = i.incident\_status\_id GROUP BY c.issue\_type    --Total resolved Incident Count Department: Count of Distinct Incidents categorized by Department. SELECT d.department\_name,  Count(DISTINCT i.Id) AS 'No of Incidents' FROM incident i  INNER JOIN employee e  ON e.id = i.created\_by  INNER JOIN department d  ON d.id = e.department\_id WHERE Incident\_Status\_Id = 6 GROUP BY d.department\_name    --Department wise Category Incident Count: Break down Distinct incident counts by 'department' and 'type of issue'. SELECT d.department\_name,  c.issue\_type,  Count(DISTINCT i.Id) AS 'No of Incidents' FROM incident i  INNER JOIN employee e  ON e.id = i.created\_by  INNER JOIN category c  ON c.id = i.incident\_status\_id  INNER JOIN department d  ON d.id = e.department\_id GROUP BY d.department\_name,  c.issue\_type ORDER BY d.department\_name    --Average Resolution Time for each Type of Priority: Calculate the average time taken to resolve incidents based on priority levels. SELECT priority,  Avg(Datediff(day, Created\_At, Resolved\_At)) AS  'Average Days to Resolve' FROM incident  WHERE incident\_status\_id = 6 Group By priority    --Closed Incidents without Proper Resolution: Identify incidents that were closed without a proper resolution. SELECT Id, Reopen\_Count FROM incident WHERE reopen\_count > 0 ORDER BY reopen\_count DESC    --Employee Leaderboard: Rank employees based on their incident count and display Employee Names with Rank, Incident count, Average resolution time in days. Select Top 15 e.User\_Name 'User Name',  Rank()  OVER (  ORDER BY Count(i.ID) DESC) AS 'Rank',  count(i.Id) 'No of Resolved Incidents', Avg(Datediff(day, i.Created\_At, i.Resolved\_At)) AS 'Avg Resolution Time Days' from Incident i left join Employee e on e.id = i.Resolved\_By where Incident\_Status\_Id = 6 group by e.User\_Name    --View for Incident Statistics: Create a view that includes detailed statistics for each resolved incident, such as the Id, Name of employee resolved, opened time, resolved time, time taken to resolve, priority, and average time taken for that particular priority type of that incident. CREATE VIEW Incident\_Statistics AS Select i.Id Id, e.User\_Name 'User Name', i.Created\_At, i.Resolved\_At, Datediff(day, i.Created\_At, i.Resolved\_At) as Resolved\_In, i.Priority, pa.[Avg Resolution Time Days] from Incident i Inner join Employee e on e.id = i.Resolved\_By Inner Join (Select Priority,  Avg(Datediff(day, Created\_At, Resolved\_At)) AS 'Avg Resolution Time Days' from Incident where Incident\_Status\_Id = 6 group by Priority) Pa on pa.Priority = i.Priority where Incident\_Status\_Id = 6 group by i.Id, e.User\_Name, i.Created\_At, i.Resolved\_At, Datediff(day, i.Created\_At, i.Resolved\_At), i.Priority, pa.[Avg Resolution Time Days]  SELECT \* FROM Incident\_Statistics    --Stored Procedure for Employee Employee Leaderboard on Priority(Input Parameter): Develop a stored procedure that generates a leaderboard of employees based on incident count with priority as input parameter. Display Employee Names with Rank, Incident count, Priority, Average resolution time in days. Alter Procedure GetEmployeeLeaderboardByPriority  @Priority NVARCHAR(50) AS BEGIN SELECT  e.User\_Name, i.Priority,  Rank()  OVER (  Partition by i.Priority ORDER BY Count(i.ID) DESC) AS 'Rank',  COUNT(i.Id) AS Incident\_Count,  AVG(DATEDIFF(Day, i.Created\_At, i.Resolved\_At)) AS Avg\_Resolution\_Time\_Days  FROM  Incident i  INNER JOIN  Employee e ON i.Resolved\_By = e.Id  WHERE  i.Priority = @Priority AND   i.Resolved\_At IS NOT NULL  GROUP BY  e.User\_Name,i.Priority END;  EXEC GetEmployeeLeaderboardByPriority '1 - Critical' |

* Connect to SQL server and import data from SQL Server to Power BI
* *Data Visualization Power BI Dashboard:*
  + Card Charts for KPI’s of Total Incident Count and Active Incident Count
  + Pie Chart for Incident Count on Status basis
  + Donut Chart for Category Incident Count
  + Bar Chart for Department Incident Count
  + Import ‘Incident\_Statistics’ view data from SQL server to Power BI to create a table. Now apply conditional formatting on ‘time taken to resolve’ and ‘average time taken for that particular priority type of that incident’ columns to create heatmap visualization.
  + Add Slicers for Status, Category, Urgency, Priority to control the charts
  + Remove Interaction of ‘Status’ slicer from Heatmap, and KPI’s
* Upload the Report (.pbix file) in the PowerBI workspace from your local machine.
* Review the report to ensure everything looks as expected. Check for any errors or issues that need to be addressed. Publish the report when ready.

