**Project Title**

Employee Engagement Survey 2023

**Project Description**

This project analyzes employee engagement survey 2023 results - a single-year analysis that focus on detailed insights, patterns & key drivers

**Tools used**

SQL (SQL Server in SSMS)

**Dataset**

1. EmployeeData\_2023 (DataYear, EmployeeID, DivisionName, DepartmentName, Gender, Year, Month)
2. SurveyResponse\_2023 (DataYear, EmployeeID, QuestionID, LikertScore, AverageScore)
3. SurveyQuestion\_MarketScore\_2023 (DataYear, QuestionID, Metric, Category, QuestionText, Theme, MarketScore)
4. MetricMapping (Metric, MetricOrder)
5. CategoryMapping (Metric, Category, CategoryOrder)
6. QuestionMapping (QuestionID, QuestionText, QuestionOrder)

**Metrics**

1. Total Employees Surveyed - to ensure enough employees to make valid conclusions
2. Overall Engagement Score - to check for engagement “health check”
3. Participation Rate
4. Headcount by Gender - show % of total
5. Male Headcount
6. Female Headcount
7. Engagement by Gender
8. Male Engagement
9. Female Engagement - do male/female employees experience the company culture differently
10. Headcount by Division
11. Engagement by Division
12. Headcount by Department
13. Engagement by Department
14. Engagement Level for Division
15. Engagement Level for Department - to identify which areas are thriving or struggling

**SQL - I have imported the data into SQL Server & I am using SSMS to query it**

/\*

tell SQL Server to create a new database named EAC\_EES2023, short & meaningful

\*/

CREATE DATABASE EAC\_EES2023;

/\*

tell SQL Server to use the EAC\_EES2023 database for all subsequent commands, this is to ensure that we choose the correct database to CREATE TABLE + BULK INSERT later

\*/

USE EAC\_EES2023;

**To store data permanently in SQL Server**

1. CREATE DATABASE

2. CREATE TABLE

3. BULK INSERT + FROM + WITH (for CSV file)

I set the PKs & FKs, to establish logical relationships between Tables

This is to ensure data integrity by preventing orphan records & enforcing referential integrity

**1st Table: EmployeeData\_2023**

CREATE TABLE EmployeeData\_2023

(

DataYear INT NOT NULL,

EmployeeID INT NOT NULL,

Gender CHAR(1) NOT NULL,

DivisionName NVARCHAR(30) NOT NULL,

DepartmentName NVARCHAR(32) NOT NULL,

YearsEmployed INT NOT NULL,

MonthsEmployed INT NOT NULL,

CONSTRAINT PK\_EmployeeData\_2023 PRIMARY KEY (EmployeeID)

);

BULK INSERT EmployeeData\_2023 -- data will be inserted in the Table here

FROM 'C:\Users\izziw\Downloads\EmployeeData\_2023.csv' -- file location

WITH

(

FORMAT = 'CSV', -- file type

FIRSTROW = 2, -- skip the 1st row (header)

FIELDTERMINATOR = ',', -- CSV column separator

ROWTERMINATOR = '\n', -- each new line is new row of data

TABLOCK -- lock the Table while inserting for better speed

);

**2nd Table: SurveyQuestion\_MarketScore\_2023**

CREATE TABLE SurveyQuestion\_MarketScore\_2023

(

DataYear INT NOT NULL,

QuestionID NVARCHAR(3) NOT NULL,

Metric NVARCHAR(5) NOT NULL,

Category NVARCHAR(10) NOT NULL,

QuestionText NVARCHAR(104) NOT NULL,

Theme NVARCHAR(25) NOT NULL,

MarketScore DECIMAL(3,2) NOT NULL, -- Total 3 digits, 2 after decimal with range: -9.99 to 9.99

CONSTRAINT PK\_SurveyQuestion\_MarketScore\_2023 PRIMARY KEY (QuestionID)

);

BULK INSERT SurveyQuestion\_MarketScore\_2023

FROM 'C:\Users\izziw\Downloads\SurveyQuestion\_MarketScore\_2023.csv'

WITH

(

FORMAT = 'CSV',

FIRSTROW = 2,

FIELDTERMINATOR = ',',

ROWTERMINATOR ='\n',

TABLOCK

);

**3rd Table: SurveyResponse\_2023**

CREATE TABLE SurveyResponse\_2023

(

DataYear INT NOT NULL,

EmployeeID INT NOT NULL,

QuestionID NVARCHAR(3) NOT NULL,

LikertScore TINYINT NOT NULL,

AverageScore DECIMAL (10,9) NOT NULL,

CONSTRAINT PK\_SurveyResponse\_2023 PRIMARY KEY (EmployeeID, QuestionID),

CONSTRAINT FK\_SurveyResponse\_2023\_EmployeeID FOREIGN KEY (EmployeeID)

REFERENCES EmployeeData\_2023(EmployeeID) ON DELETE CASCADE,

CONSTRAINT FK\_SurveyResponse\_2023\_QuestionID FOREIGN KEY (QuestionID)

REFERENCES SurveyQuestion\_MarketScore\_2023(QuestionID) ON DELETE CASCADE

);

/\* TINYINT (1 byte, 0–255), INT (4 bytes), save space & boost speed

add CPK to prevent duplicate responses & keep data structured, the BEST practice

add ON DELETE CASCADE for EmployeeID & QuestionID, prevent orphaned records \*/

BULK INSERT SurveyResponse\_2023

FROM 'C:\Users\izziw\Downloads\SurveyResponse\_2023.csv'

WITH

(

FORMAT = 'CSV',

FIRSTROW = 2,

FIELDTERMINATOR = ',',

ROWTERMINATOR ='\n',

TABLOCK

);

**Tables for SurveyMapping**

**1st Table: MetricMapping**

CREATE TABLE MetricMapping

(

Metric NVARCHAR(5) NOT NULL,

MetricOrder INT NOT NULL

);

INSERT INTO

MetricMapping (Metric, MetricOrder)

VALUES

('Core', 1),

('Self', 2),

('Group', 3);

**2nd Table: CategoryMapping**

CREATE TABLE CategoryMapping

(

Metric NVARCHAR(5) NOT NULL,

Category NVARCHAR(10) NOT NULL,

Metric\_Category NVARCHAR(15) NOT NULL,

CategoryOrder INT NOT NULL

);

INSERT INTO

CategoryMapping (Metric, Category, Metric\_Category, CategoryOrder)

VALUES

('Core', 'Leadership', 'Core-Leadership', 1),

('Core', 'Culture', 'Core-Culture', 2),

('Core', 'Initiative', 'Core-Initiative', 3),

('Self', 'Heart', 'Self-Heart', 1),

('Self', 'Mind', 'Self-Mind', 2),

('Self', 'Soul', 'Self-Soul', 3),

('Group', 'Think', 'Group-Think', 1),

('Group', 'Feel', 'Group-Feel', 2),

('Group', 'Do', 'Group-Do', 3);

**3rd Table: QuestionMapping**

CREATE TABLE QuestionMapping

(

QuestionID NVARCHAR(3) NOT NULL,

QuestionText NVARCHAR(68) NOT NULL,

QuestionID\_QuestionText NVARCHAR(72) NOT NULL,

QuestionOrder INT NOT NULL

);

BULK INSERT QuestionMapping

FROM 'C:\Users\izziw\Downloads\QuestionMapping.csv'

WITH

(

FORMAT = 'CSV',

FIRSTROW = 2,

FIELDTERMINATOR = ',',

ROWTERMINATOR ='\n',

TABLOCK

);

/\*

Total Employees Surveyed - number of employee responded

SQL concepts used - FROM, SELECT, COUNT()

\*/

SELECT

COUNT(EmployeeID) AS TotalEmployeesSurveyed

FROM

EmployeeData\_2023;

/\*

Overall Engagement Score - average of engagement responses

SQL concepts used - FROM, SELECT, CAST(), AVG()

\*/

SELECT

CAST(AVG(CAST(LikertScore AS DECIMAL(10,2))) / 5 \* 100 AS DECIMAL (3,0)) AS

OverallEngagementScore

FROM

SurveyResponse\_2023;

/\*

Participation Rate - employees who completed the survey

SQL concepts used - FROM, LEFT JOIN, SELECT, CAST(), COUNT(), CASE

\*/

SELECT

CAST(COUNT(CASE WHEN sr.EmployeeID IS NOT NULL THEN 1 END) \* 100 / COUNT(\*) AS

DECIMAL(3,0)) AS ParticipationRate

FROM

EmployeeData\_2023 AS ed

LEFT JOIN

SurveyResponse\_2023 AS sr

ON

ed.EmployeeID = sr.EmployeeID;

/\*

Headcount by Gender - count employees grouped by gender

SQL concepts used - FROM, GROUP BY, SELECT, COUNT()

\*/

SELECT

Gender,

COUNT(EmployeeID) AS Headcount

FROM

EmployeeData\_2023

GROUP BY

Gender;

/\*

Engagement by Gender - average of engagement responses by gender

SQL concepts used: FROM, INNER JOIN, GROUP BY, SELECT, CAST(), AVG()

\*/

SELECT

ed.Gender,

CAST(AVG(CAST(sr.LikertScore AS DECIMAL(10,2))) / 5 \* 100 AS DECIMAL (3,0)) AS

EngagementScore

FROM

EmployeeData\_2023 AS ed

INNER JOIN

SurveyResponse\_2023 AS sr

ON

ed.EmployeeID = sr.EmployeeID

GROUP BY

ed.Gender;

/\*

Headcount by Division sorted in DESC order

SQL concepts used - FROM, GROUP BY, SELECT, COUNT(), ORDER BY

\*/

SELECT

DivisionName,

COUNT(EmployeeID) AS Headcount

FROM

EmployeeData\_2023

GROUP BY

DivisionName

ORDER BY

DivisionName DESC;

/\*

Engagement by Division, sorted by Engagement Score in DESC order

SQL concepts used - FROM, INNER JOIN, GROUP BY, SELECT, CAST(), AVG(), ORDER BY

\*/

SELECT

ed.DivisionName,

CAST(AVG(CAST(sr.LikertScore AS DECIMAL(10,2))) / 5 \* 100 AS DECIMAL (3,0)) AS

EngagementScore

FROM

EmployeeData\_2023 AS ed

INNER JOIN

SurveyResponse\_2023 AS sr

ON

ed.EmployeeID = sr.EmployeeID

GROUP BY

ed.DivisionName

ORDER BY

EngagementScore DESC;

/\*

Headcount by Department, sorted by Headcount in DESC order

SQL concepts used - FROM, GROUP BY, SELECT, COUNT(), ORDER BY

\*/

SELECT

DepartmentName,

COUNT(EmployeeID) AS Headcount

FROM

EmployeeData\_2023

GROUP BY

DepartmentName

ORDER BY

Headcount DESC;

/\*

Engagement by Department, sorted by Engagement Score in DESC order

SQL concepts used - FROM, INNER JOIN, GROUP BY, SELECT, CAST(), AVG(), ORDER BY

\*/

SELECT

ed.DepartmentName,

CAST(AVG(CAST(sr.LikertScore AS DECIMAL(10,2))) / 5 \* 100 AS DECIMAL (3,0)) AS EngagementScore

FROM

EmployeeData\_2023 AS ed

INNER JOIN

SurveyResponse\_2023 AS sr

ON

ed.EmployeeID = sr.EmployeeID

GROUP BY

DepartmentName

ORDER BY

EngagementScore DESC;

/\*

Headcount, Engagement Score & Engagement Level by Division, sorted by Engagement Score in DESC order

SQL concepts used - FROM, INNER JOIN, GROUP BY, SELECT, COUNT(DISTINCT), AVG(), CAST(), CASE, ORDER BY

\*/

SELECT

ed.DivisionName,

COUNT(DISTINCT ed.EmployeeID) AS Headcount,

CAST(AVG(CAST(sr.LikertScore AS DECIMAL(10,2))) / 5 \* 100 AS DECIMAL (3,0)) AS EngagementScore,

CASE

WHEN AVG(CAST(sr.LikertScore AS DECIMAL(10,2))) / 5 \* 100 >= 80 THEN 'Highly Engaged'

WHEN AVG(CAST(sr.LikertScore AS DECIMAL(10,2))) / 5 \* 100 >= 65 THEN 'Engaged'

WHEN AVG(CAST(sr.LikertScore AS DECIMAL(10,2))) / 5 \* 100 >= 50 THEN 'Neutral'

WHEN AVG(CAST(sr.LikertScore AS DECIMAL(10,2))) / 5 \* 100 >= 25 THEN 'Disengaged'

ELSE 'Actively Disengaged'

END AS 'EngagementLevel'

FROM

EmployeeData\_2023 AS ed

INNER JOIN

SurveyResponse\_2023 AS sr

ON

ed.EmployeeID = sr.EmployeeID

GROUP BY

ed.DivisionName

ORDER BY

EngagementScore DESC;

/\*

Headcount, Engagement Score & Engagement Level by Division, sorted by Engagement Score in DESC order

using CTE

\*/

WITH CombinedData AS (

SELECT

ed.DivisionName,

COUNT(DISTINCT ed.EmployeeID) AS Headcount,

CAST(AVG(CAST(sr.LikertScore AS DECIMAL(10,2))) / 5 \* 100 AS DECIMAL (3,0)) AS EngagementScore,

CASE

WHEN AVG(CAST(sr.LikertScore AS DECIMAL(10,2))) / 5 \* 100 >= 80 THEN 'Highly Engaged'

WHEN AVG(CAST(sr.LikertScore AS DECIMAL(10,2))) / 5 \* 100 >= 65 THEN 'Engaged'

WHEN AVG(CAST(sr.LikertScore AS DECIMAL(10,2))) / 5 \* 100 >= 50 THEN 'Neutral'

WHEN AVG(CAST(sr.LikertScore AS DECIMAL(10,2))) / 5 \* 100 >= 25 THEN 'Disengaged'

ELSE 'Actively Disengaged'

END AS 'EngagementLevel'

FROM

EmployeeData\_2023 AS ed

INNER JOIN

SurveyResponse\_2023 AS sr

ON

ed.EmployeeID = sr.EmployeeID

GROUP BY

ed.DivisionName)

SELECT

DivisionName,

Headcount,

EngagementScore,

EngagementLevel

FROM

CombinedData

ORDER BY

EngagementScore DESC;

/\*

Headcount, Engagement Score & Engagement Level by Department, sorted by Engagement Score in DESC order

using CTE

\*/

WITH CombinedData AS (

SELECT

ed.DepartmentName,

COUNT(DISTINCT ed.EmployeeID) AS Headcount,

CAST(AVG(CAST(sr.LikertScore AS DECIMAL(10,2))) / 5 \* 100 AS DECIMAL (3,0)) AS EngagementScore,

CASE

WHEN AVG(CAST(sr.LikertScore AS DECIMAL(10,2))) / 5 \* 100 >= 80 THEN 'Highly Engaged'

WHEN AVG(CAST(sr.LikertScore AS DECIMAL(10,2))) / 5 \* 100 >= 65 THEN 'Engaged'

WHEN AVG(CAST(sr.LikertScore AS DECIMAL(10,2))) / 5 \* 100 >= 50 THEN 'Neutral'

WHEN AVG(CAST(sr.LikertScore AS DECIMAL(10,2))) / 5 \* 100 >= 25 THEN 'Disengaged'

ELSE 'Actively Disengaged'

END AS 'EngagementLevel'

FROM

EmployeeData\_2023 AS ed

INNER JOIN

SurveyResponse\_2023 AS sr

ON

ed.EmployeeID = sr.EmployeeID

GROUP BY

ed.DepartmentName)

SELECT

DepartmentName,

Headcount,

EngagementScore,

EngagementLevel

FROM

CombinedData

ORDER BY

EngagementScore DESC;