IST659 Project Deliverable

Developing a One Source of Truth Relational Database Management System

Quinn Knudsen

Syracuse University

Table of Contents

[**Summary** 3](#_Toc27316066)

[**Business Case** 3](#_Toc27316067)

[**Business Rules** 3](#_Toc27316068)

[**Stakeholders** 3](#_Toc27316069)

[**Selected Sample of Descriptive Data Questions** 3](#_Toc27316070)

[**Expectations** 4](#_Toc27316071)

[‘One Source of Truth’ Conceptualization 4](#_Toc27316072)

[KNIME Automation Example 5](#_Toc27316073)

[**Data Dictionary** 6](#_Toc27316074)

[**Conceptual Model** 8](#_Toc27316075)

[**Logical Model** 9](#_Toc27316076)

[**Normalized Model** 9](#_Toc27316077)

[**Implementation** 10](#_Toc27316078)

[Completed Tables 10](#_Toc27316079)

[Business Question Answers 13](#_Toc27316080)

[GUI Prototype 16](#_Toc27316081)

[Employee Core Data Entry Screen 16](#_Toc27316082)

[Employee Listing 17](#_Toc27316083)

[Pivot Table of Compensation for Active Employees by Employee 17](#_Toc27316084)

[**Reflection** 17](#_Toc27316085)

[**Final Summary** 20](#_Toc27316086)

[**Appendix Code** 20](#_Toc27316087)

# **Summary**

# **Business Case**

It is essential for the success and development of advanced analytics within HR to design and implement a cohesive ‘one source of truth’ database to eliminate the nuances, risk of human error, and sustainability of disparate systems that do not ‘talk to’ one another. This project overviews a conceptual and logical model for this ‘one source of truth’ structure and outlines the necessary relationships needed for successful data integration. The value-add aspects of this project are in the time saved by having systems aligned for day to day reporting and the elimination of extensive preprocess data preparation and manual data amalgamation. Furthermore, the sustainability and cohesion of this model allows for greater depth of analytics and provides the structure necessary for a solid foundation of historical variables that enable accurate descriptive analytics as well as predictive analytics (such as machine learning). The time savings ROI assuming 3 hours saved a week for all analysts approximates to roughly $900 a week in time spent marrying datasets or searching for answers. Additionally, the ROI on predictive analytics through preemptive the determination of flight risk in key talents exceeds $200,000 per employee.

# **Business Rules**

1. Each employee must have at least one active status; reactivated employees retain their original employee ID.
2. Each business unit has multiple employees, each employee has one active business unit.
3. An employee may have multiple job roles, only one job role can be active.
4. Data deemed to be sensitive by governance will have restricted access given only to certain approved users.

# **Stakeholders**

Key stakeholders: HR leadership team, data governance, HR analytics

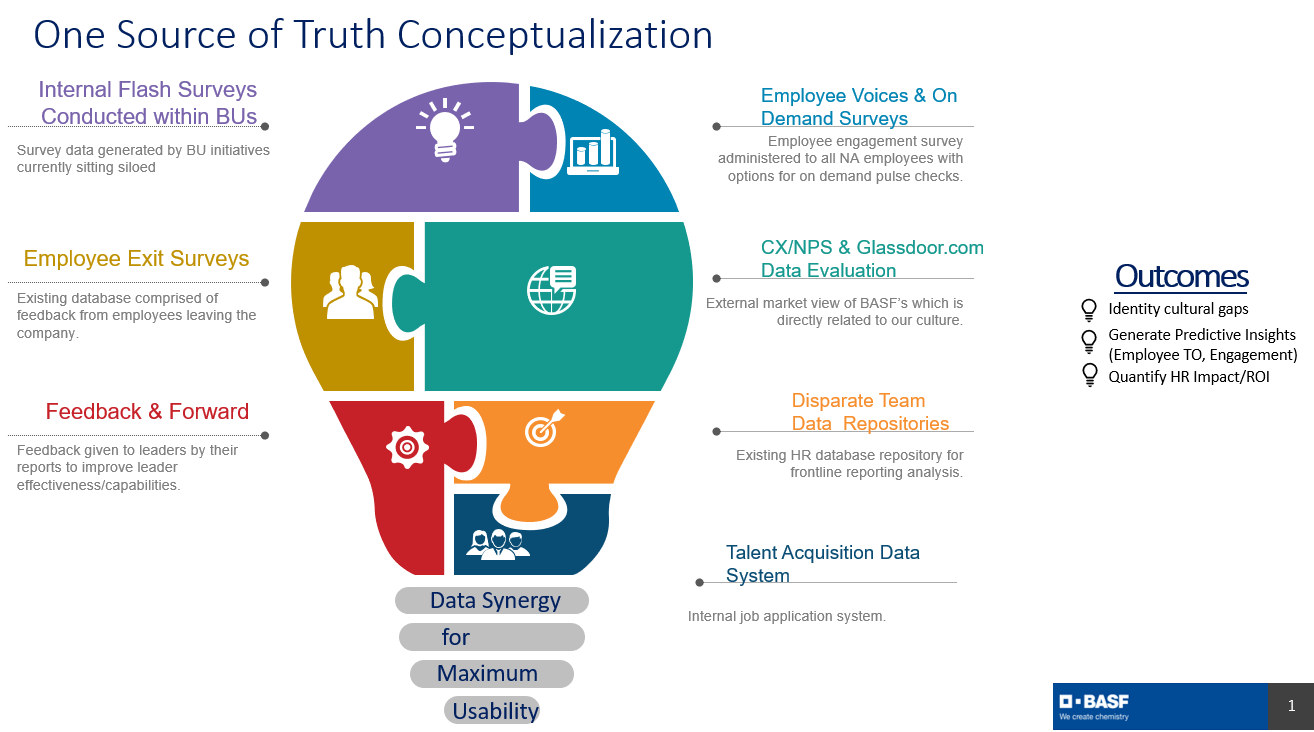
# **Selected Sample of Descriptive Data Questions**

1. Which business units have the highest:
   1. Number of employees
   2. Resignation
   3. Engagement
   4. NPS
   5. Span of Control
2. How many employees are mobile?
   1. What percentage of employees are mobile?
3. How many employees are HiPos?
   1. Which HiPos are mobile?
4. What percentage of employees are HiPos?
5. How many recognition points have top employees earned?
6. Which job families has the most employees?

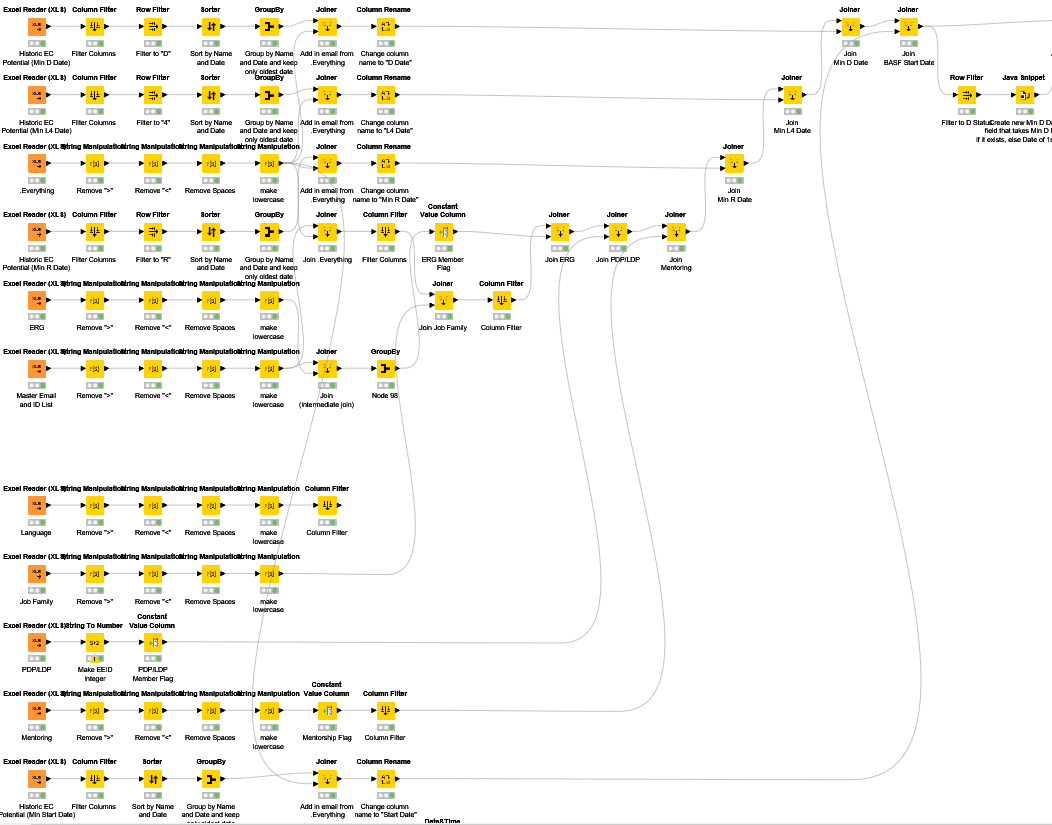
# **Expectations**

The result of this project will be a ‘one source of truth’ data warehouse to enable current and future analytics within HR. Expectations are for required parties to devote 40 working hours over the next two months to train on the required software and integrate the systems into SQL.

## ‘One Source of Truth’ Conceptualization



## KNIME Automation Example



*Note.* Snippet from one use case of current operational model utilizing KNIME without ‘one source of truth’ relational database management system integration. Process development took weeks of work.

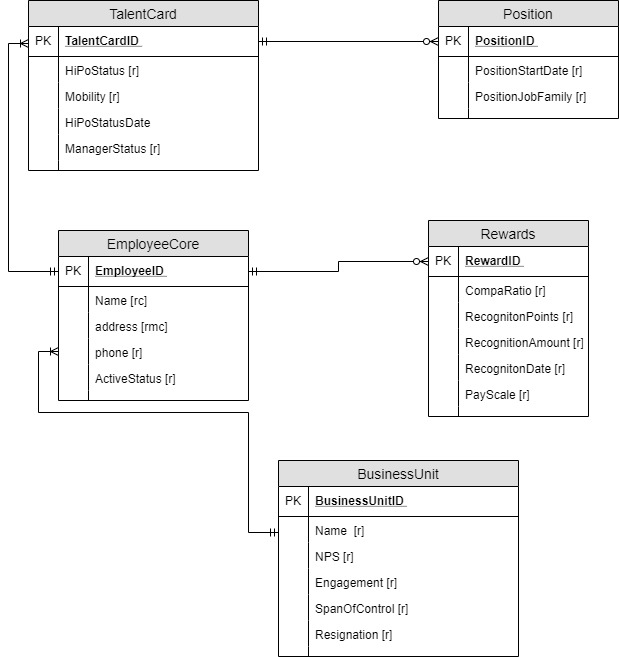
# **Data Dictionary**

Reviewing the document and working with the volunteers, we have identified the following list of entities and their attributes.

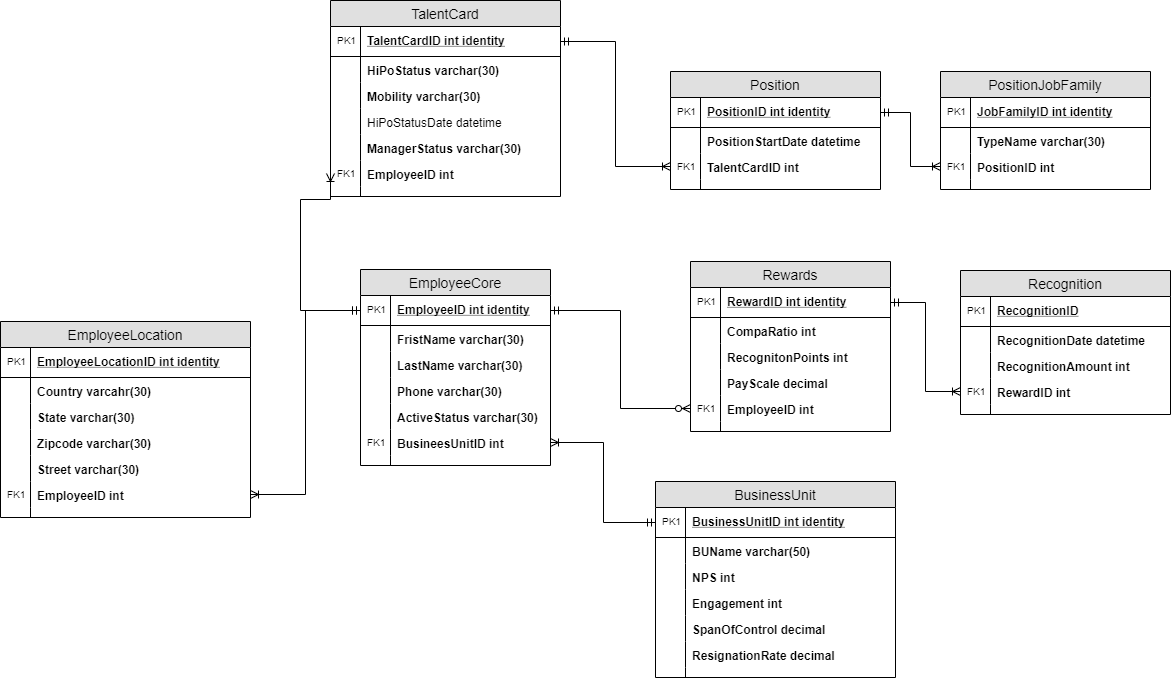
| **Entity** | **Attribute** | **Properties** |
| --- | --- | --- |
| EmployeeCore | Name | Required. Composite of Forename and Surname. |
| Address | Required. Multivalued. |
| Phone | Required. |
| BusinessUnit | BUName | Required. |
| NPS | Required. Customer feedback score. |
| Engagement | Required. |
| SpanOfControl | Required. Number of direct reports per manager. |
| ResignationRate | Required. |
| TalentCard | HiPoStatus | Required. High potential designation. |
| Mobility | Required. Freedom to move around the globe. |
| HiPoStatusDate | Nomination date. |
| ManagerStatus | Required. |
| Position | PositionStartDate | Required. |
| PositionJobFamily | Required. |
| Rewards | RecognitionPoints | Required. Points awarded for excellent work. |
| PayScale | Required. Employee salary designation. |
| CompaRatio | Required. Employee pay compared to market value. |

# **Conceptual Model**

When connecting the data relationships, and entity relationship diagram (ERD) is produced. After consulting key stakeholders, the ERD model below was developed to help conceptualize the linkages that will occur within the database.



# **Logical Model**



# **Normalized Model**

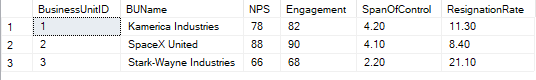
The effort to create the logical model resulted in relations in at least third-normal form, so no further normalization is required.

# **Implementation**

For SQL commands please reference the code in the Appendix. The following section outlines the completed tables, business question answers, and command outputs.

# Completed Tables

BusinessUnit Table



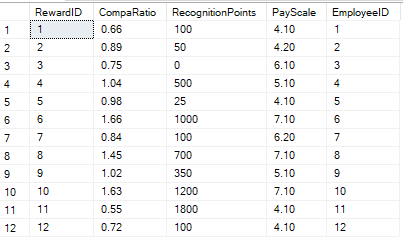
EmployeeCore Table



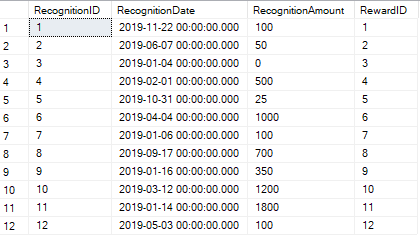
EmployeeLocation Table



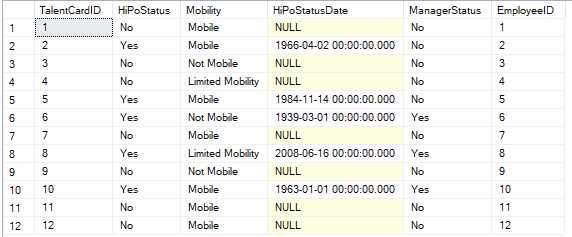
Rewards Table



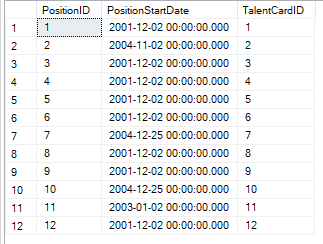
Recognition Table



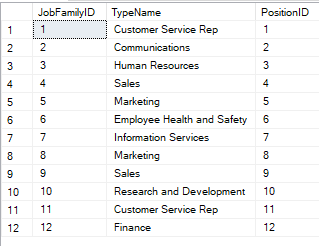
TalentCard Table



Position Table

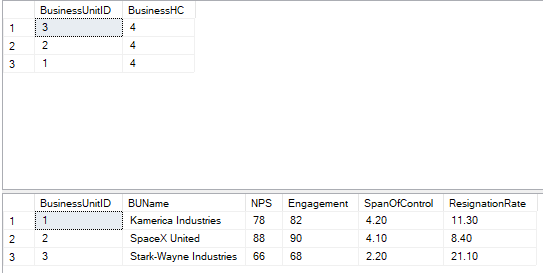


PositionJobFamily Table



# Business Question Answers

Highest Number of Employees



Highest Resignation Rate



Highest Engagement



Highest NPS



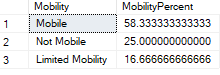
Highest Span of Control



Total Number of Fully Mobile Employees



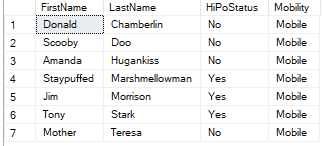
Percent Breakdown by ‘Mobility Status’ for Employees



Count of High Potentials



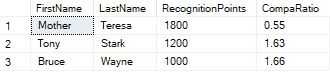
Which fully mobile employees have a high potential?



What percentage of employees are high potentials?



What is the total number of recognition points received for top recognition receivers? Who are they and how does their recognition compare to their compa ratio?



Which job family has the most employees?



Procedure: How many times was employee 6 recognized?



BusinessUnit Descriptives



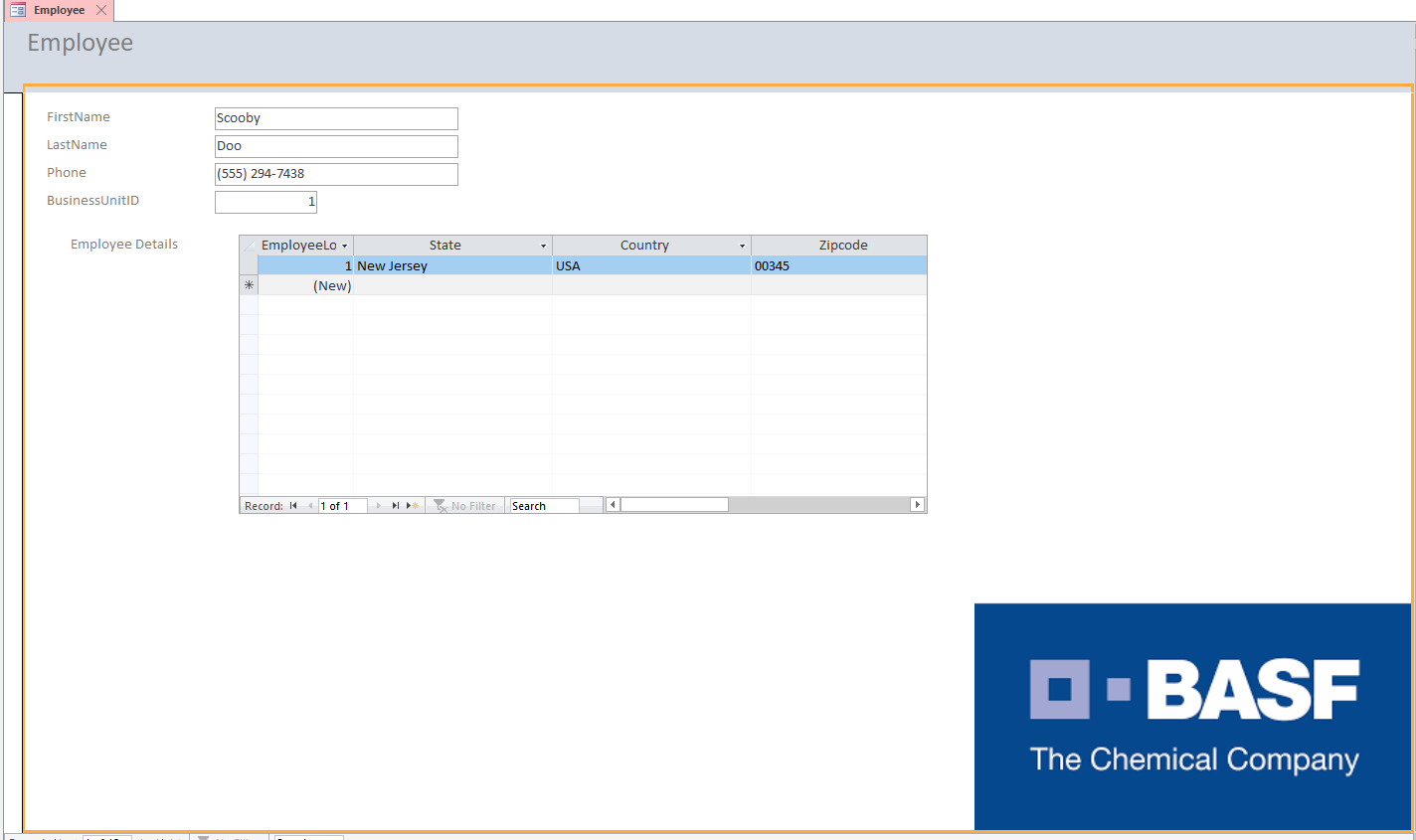
Function: Look up for which BUID Elon Musk belongs to:

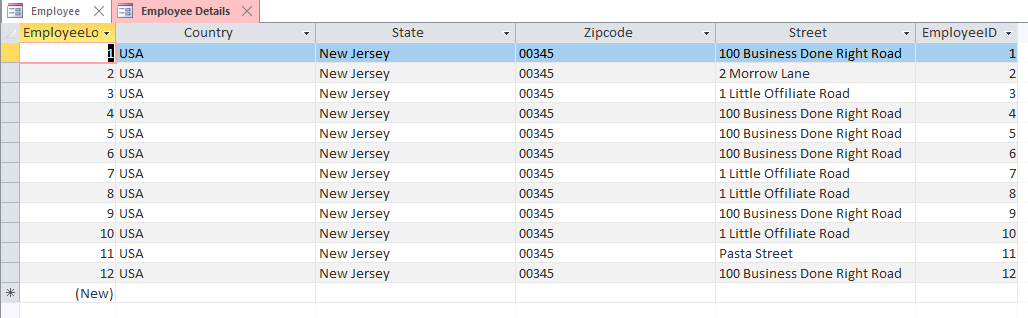


### GUI Prototype

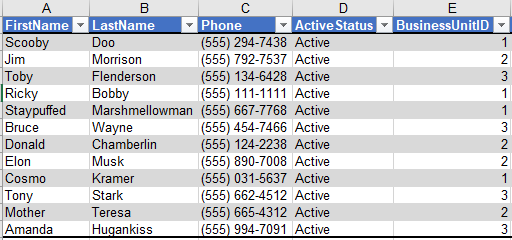
Access data entry screen and selected excel view and pivot table.

### Employee Core Data Entry Screen

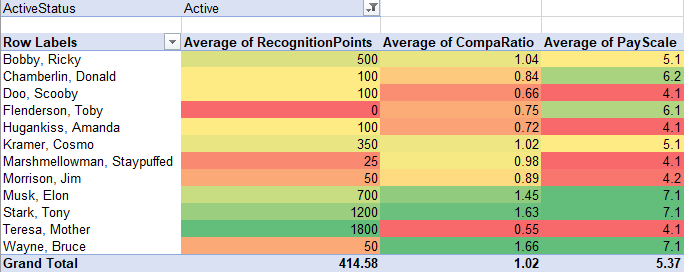




# Employee Listing



# Pivot Table of Compensation for Active Employees by Employee

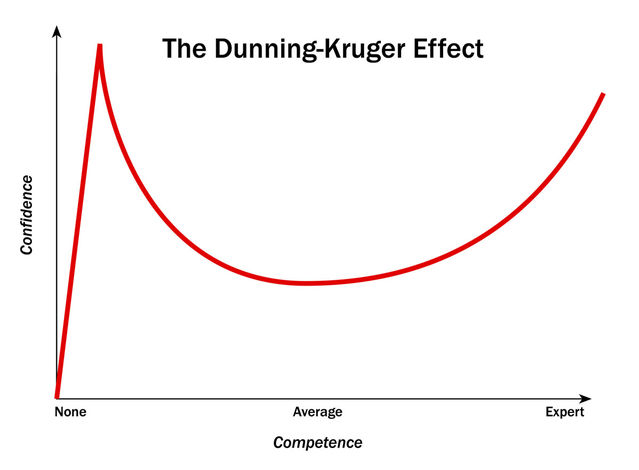


# **Reflection**

1. What assumptions did you have at the start of your project that changed by the end? Think in terms of both your own problem domain as well as your knowledge of the process.

At the outset of this project, I operated under a much more simplistic understanding and assumption of the intricacies that go into designing and implementing that the backend of database design, migration, user interface, and maintenance. Before this course, I would have never anticipated the level of planning that goes into a database design to achieve a normalized model. Another assumption that this project demonstrates is that you cannot assume that the SQL/database design expert has a fundamental understanding of the business questions the database is hoping to help answer. Without clear communication from all parties it is easy to have misses and missed opportunities for what can be achieved. Without the business leader communicating what they want to achieve from the database, the database designer will not properly know how design it. Furthermore, without the database designer not successfully communicating the full functionality of the database software, a business leader may not fully understand what to ask for.

With that being said, the database we created is tabular and incredibly small in comparison to many of the databases that exist in the world which is, if nothing else, humbling in terms of the complexity. Our data set was built on low volume, variety, and velocity with a decent veracity which are not always the case in the business world. This made understanding the concepts much easier, but reinforced that there many tiers in terms of SQL and database design expertise. One of my favorite psychological phenomena is the Dunning-Kruger effect (below) in which people tend to over assess their capacity to do something. It is related to the cognitive bias of illusory superiority and comes from the inability of people to recognize their lack of ability. This holds very true with learning how complex designing a relational database structure can be and the extent to which you can utilize SQL commands to manage that process.



1. The next time you do this, what will be different?

Having gone through this process once, I understand that it is an iterative endeavor and that experience with both the technical coding and conceptual design will enhance the efficiencies and output. I would liken this to the first two paragraph essay that was written in 2nd grade. The exertion of effort necessary to provide a narrative far surpasses the tangible results. However, each time you go through the process of writing an essay, the nuances become more routine, the vocabulary multifarious, and the outcome standards are enhanced. Similarly, conceptually designing and implementing the code necessary to produce a working database suitable to answer the chosen business questions is exhaustive when the process is new. The next time I do this I will have completed my concepts and the depth and breadth of coding will naturally develop based upon familiarity, demand, and natural curiously.

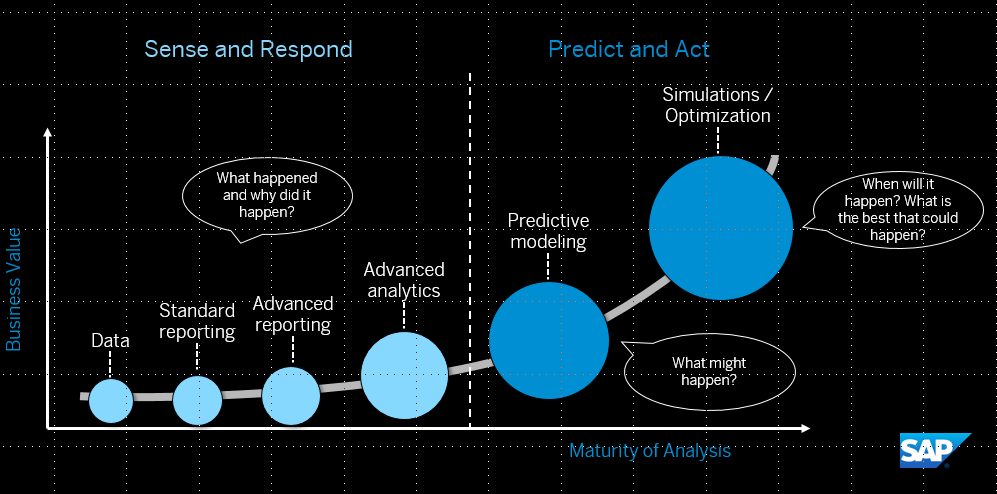
Moving from conceptual and tangible much in the way this course is designed, palpable changes I would make would be to eliminate the use of insert statements to reduce time in favor of an import. Existing datasets would become mind-numbingly tedious to hand code insert statements for the many rows on columns and already exist in excel format. I also may have to loosen some of the mandatory fields suggested in the design above such as mobility or be savvy and extend the entry choices to include ‘undefined’ or something of that nature for employees who are very new to the company and may not have had a conversation around their career mobility.

1. Regardless of whether you go through these steps again, how do you think it will inform your approach to data as an information professional?

This process was fantastic for understanding how to better communicate with someone designing a database, or if need be developing the process myself. A lesson was learned regardless of activity to do extensive preplanning (deliberate planning), but also include contingency planning and reactive adjustment to deal with the obstacles that always arise in a project which has been linked to task performance (DeChurch & Haas, 2008). The initial design of a database is vulnerable to change based upon variable business questions and things that arise throughout the implementation process.

As a data professional, I typically operate as the recipient of a dataset with finite warehousing interaction. Although the only holder of the HR master datasets and KNIME project leader an integrating HR and non-HR data, we do not have a system in place that houses that raw data. My operating space in terms of database construction is through manipulating and combining reports external to the database altogether. Through this course I have leaned some invaluable database skills and the SQL coding language with enough knowledge to be dangerous enough in the space to bring this business case to light in my organization. As a catalyst for the adoption of a data lake or similar, this will vastly improve the extent of projects I am able to undertake from a data analysis perspective and open the door to others that are unobtainable without a proper skillset or infostructure in this area.

Finally, this course redefined how I think about data. Previously, data storage and the ‘behind the scenes’ side of data was something that I did not think deeply about. Typically, as a user or recipient of data reports, aside from integrating datasets or cleaning datasets before analysis, I had little insight into the operations behind the created reports. With this enriched understanding, it will help me better design our ‘One source of truth’ HR database that is highly necessary for progressing our analytics along the data maturity curve from predominantly standard reporting to predictive modeling and optimizations.



# **Final Summary**

I answered the data questions by creating a series of views and added additional stored procedures and functions that would enable answering future business questions. Additional analysis could be performed by extracting the data into Python, R, or simply in Excel as visualized in the pivot table. At the outset of the project or any database creation it is helpful to have guiding business questions. However, it is impossible to know the precise extent of the questions that the data will be used to solve. Proper UI will enable the users to extract and analyze the data in the future without being burdened with needing to know extensive SQL code. This is important to keep in mind when designing the views and UI.

The UI was kept very simplistic to accelerate the learning curve of the end user and included the company branded logo for optics and legitimacy. Although it is a working protype and increasingly sophisticated UI can and should be developed, this was a starting point. Access is a bit more user friendly than asking the end user to supply SQL code to enter or retrieve the data they wish to interact with.

# **Appendix Code**

--Final Project Part 2--

--SQL DDL--

/\*

SQL DDL Statements for 'One HR Source of Truth Concept'

Author: Quinn Knudsen

December 2019

IST 659 Final Project

\*/

/\*

Drop the tables first, if they exist

Dropped in reverse order of creation to avoid any problems with

foreign key references

Using SQL Server 2016 DROP method

\*/

DROP TABLE dbo.EmployeeLocation

DROP TABLE dbo.PositionJobFamily

DROP TABLE dbo.Recognition

DROP TABLE dbo.Position

DROP TABLE dbo.Rewards

DROP TABLE dbo.TalentCard

DROP TABLE dbo.EmployeeCore

DROP TABLE dbo.BusinessUnit

/\*

Create the tables. Start by creating strong tables first

to simplify constraint creation.

\*/

CREATE TABLE BusinessUnit (

BusinessUnitID int identity primary key

, BUName varchar(50) not null

, NPS int not null

, Engagement int not null

, SpanOfControl decimal(4,2) not null

, ResignationRate decimal(4,2) not null

)

CREATE TABLE EmployeeCore (

EmployeeID int identity primary key

, FirstName varchar(30) not null

, LastName varchar(30) not null

, Phone varchar(30) not null

, ActiveStatus varchar(30) not null

, BusinessUnitID int not null

CONSTRAINT FK1\_EmployeeCore FOREIGN KEY (BusinessUnitID) REFERENCES BusinessUnit(BusinessUnitID)

)

CREATE TABLE TalentCard (

TalentCardID int identity primary key

, HiPoStatus varchar(30) not null

, Mobility varchar(30) not null

, HiPoStatusDate datetime

, ManagerStatus varchar(30) not null

, EmployeeID int not null FOREIGN KEY REFERENCES EmployeeCore(EmployeeID)

)

CREATE TABLE Rewards (

RewardID int identity primary key

, CompaRatio decimal(4,2) not null

, RecognitionPoints int not null

, PayScale decimal(4,2) not null

, EmployeeID int not null FOREIGN KEY REFERENCES EmployeeCore(EmployeeID)

)

CREATE TABLE Position (

PositionID int identity primary key

, PositionStartDate datetime not null

, TalentCardID int not null FOREIGN KEY REFERENCES TalentCard(TalentCardID)

)

CREATE TABLE Recognition (

RecognitionID int identity primary key

, RecognitionDate datetime not null

, RecognitionAmount int not null

, RewardID int not null FOREIGN KEY REFERENCES Rewards(RewardID)

)

CREATE TABLE PositionJobFamily (

JobFamilyID int identity primary key

, TypeName varchar(30) not null

, PositionID int not null FOREIGN KEY REFERENCES Position(PositionID)

)

CREATE TABLE EmployeeLocation (

EmployeeLocationID int identity primary key

, Country varchar(30) not null

, State varchar(30) not null

, Zipcode varchar(30) not null

, Street varchar(30) not null

, EmployeeID int not null FOREIGN KEY REFERENCES EmployeeCore(EmployeeID)

)

--SQL DML--

--Note: Several rows would be entered in each table. These are just representative examples of what would be added.

--BusinessUnit Table

INSERT INTO BusinessUnit([BUName], [NPS],[Engagement], [SpanOfControl], [ResignationRate]) VALUES('Kamerica Industries',78, 82, 4.2, 11.3);

INSERT INTO BusinessUnit([BUName],[NPS],[Engagement], [SpanOfControl], [ResignationRate]) VALUES('SpaceX United',88, 90, 4.1, 8.4);

INSERT INTO BusinessUnit([BUName], [NPS],[Engagement], [SpanOfControl], [ResignationRate]) VALUES('Stark-Wayne Industries',66, 68, 2.2, 21.1);

SELECT \* FROM BusinessUnit

--Employee Core Table--

INSERT INTO EmployeeCore(FirstName,LastName,Phone, ActiveStatus, BusinessUnitID) VALUES('Scooby','Doo','(555) 294-7438','Active', 1);

INSERT INTO EmployeeCore([FirstName],[LastName],[Phone], [ActiveStatus], [BusinessUnitID]) VALUES('Jim','Morrison','(555) 792-7537','Active',2);

INSERT INTO EmployeeCore([FirstName],[LastName],[Phone], [ActiveStatus], [BusinessUnitID]) VALUES('Toby','Flenderson','(555) 134-6428','Active',3);

INSERT INTO EmployeeCore([FirstName],[LastName],[Phone], [ActiveStatus], [BusinessUnitID]) VALUES('Ricky','Bobby','(555) 111-1111','Active',1);

INSERT INTO EmployeeCore([FirstName],[LastName],[Phone], [ActiveStatus], [BusinessUnitID]) VALUES('Staypuffed','Marshmellowman','(555) 667-7768','Active',1);

INSERT INTO EmployeeCore([FirstName],[LastName],[Phone], [ActiveStatus], [BusinessUnitID]) VALUES('Bruce','Wayne','(555) 454-7466','Active',3);

INSERT INTO EmployeeCore([FirstName],[LastName],[Phone], [ActiveStatus], [BusinessUnitID]) VALUES('Donald','Chamberlin','(555) 124-2238','Active',2);

INSERT INTO EmployeeCore([FirstName],[LastName],[Phone], [ActiveStatus], [BusinessUnitID]) VALUES('Elon','Musk','(555) 890-7008','Active',2);

INSERT INTO EmployeeCore([FirstName],[LastName],[Phone], [ActiveStatus], [BusinessUnitID]) VALUES('Cosmo','Kramer','(555) 031-5637','Active',1);

INSERT INTO EmployeeCore([FirstName],[LastName],[Phone], [ActiveStatus], [BusinessUnitID]) VALUES('Tony','Stark','(555) 662-4512','Active',3);

INSERT INTO EmployeeCore([FirstName],[LastName],[Phone], [ActiveStatus], [BusinessUnitID]) VALUES('Mother','Teresa','(555) 665-4312','Active',2);

INSERT INTO EmployeeCore([FirstName],[LastName],[Phone], [ActiveStatus], [BusinessUnitID]) VALUES('Amanda','Hugankiss','(555) 994-7091','Active',3);

SELECT \* FROM EmployeeCore

--EmployeeLocation Table

INSERT INTO EmployeeLocation([Country],[State],[Zipcode], [Street], [EmployeeID]) VALUES('USA','New Jersey','00345','100 Business Done Right Road',1);

INSERT INTO EmployeeLocation([Country],[State],[Zipcode], [Street], [EmployeeID]) VALUES('USA','New Jersey','00345','2 Morrow Lane',2);

INSERT INTO EmployeeLocation([Country],[State],[Zipcode], [Street], [EmployeeID]) VALUES('USA','New Jersey','00345','1 Little Offiliate Road',3);

INSERT INTO EmployeeLocation([Country],[State],[Zipcode], [Street], [EmployeeID]) VALUES('USA','New Jersey','00345','100 Business Done Right Road',4);

INSERT INTO EmployeeLocation([Country],[State],[Zipcode], [Street], [EmployeeID]) VALUES('USA','New Jersey','00345','100 Business Done Right Road',5);

INSERT INTO EmployeeLocation([Country],[State],[Zipcode], [Street], [EmployeeID]) VALUES('USA','New Jersey','00345','100 Business Done Right Road',6);

INSERT INTO EmployeeLocation([Country],[State],[Zipcode], [Street], [EmployeeID]) VALUES('USA','New Jersey','00345','1 Little Offiliate Road',7);

INSERT INTO EmployeeLocation([Country],[State],[Zipcode], [Street], [EmployeeID]) VALUES('USA','New Jersey','00345','1 Little Offiliate Road',8);

INSERT INTO EmployeeLocation([Country],[State],[Zipcode], [Street], [EmployeeID]) VALUES('USA','New Jersey','00345','100 Business Done Right Road',9);

INSERT INTO EmployeeLocation([Country],[State],[Zipcode], [Street], [EmployeeID]) VALUES('USA','New Jersey','00345','1 Little Offiliate Road',10);

INSERT INTO EmployeeLocation([Country],[State],[Zipcode], [Street], [EmployeeID]) VALUES('USA','New Jersey','00345','Pasta Street',11);

INSERT INTO EmployeeLocation([Country],[State],[Zipcode], [Street], [EmployeeID]) VALUES('USA','New Jersey','00345','100 Business Done Right Road',12);

Select \* FROM EmployeeLocation

--Rewards Table

INSERT INTO Rewards([CompaRatio],[RecognitionPoints],[PayScale],[EmployeeID]) VALUES(0.66, 100, 4.1,1);

INSERT INTO Rewards([CompaRatio],[RecognitionPoints],[PayScale],[EmployeeID]) VALUES(0.89, 50, 4.2,2);

INSERT INTO Rewards([CompaRatio],[RecognitionPoints],[PayScale],[EmployeeID]) VALUES(0.75, 0, 6.1,3);

INSERT INTO Rewards([CompaRatio],[RecognitionPoints],[PayScale],[EmployeeID]) VALUES(1.04, 500, 5.1,4);

INSERT INTO Rewards([CompaRatio],[RecognitionPoints],[PayScale],[EmployeeID]) VALUES(0.98, 25, 4.1,5);

INSERT INTO Rewards([CompaRatio],[RecognitionPoints],[PayScale],[EmployeeID]) VALUES(1.66, 1000, 7.1,6);

INSERT INTO Rewards([CompaRatio],[RecognitionPoints],[PayScale],[EmployeeID]) VALUES(0.84, 100, 6.2,7);

INSERT INTO Rewards([CompaRatio],[RecognitionPoints],[PayScale],[EmployeeID]) VALUES(1.45, 700, 7.1,8);

INSERT INTO Rewards([CompaRatio],[RecognitionPoints],[PayScale],[EmployeeID]) VALUES(1.02, 350, 5.1,9);

INSERT INTO Rewards([CompaRatio],[RecognitionPoints],[PayScale],[EmployeeID]) VALUES(1.63, 1200, 7.1,10);

INSERT INTO Rewards([CompaRatio],[RecognitionPoints],[PayScale],[EmployeeID]) VALUES(0.55, 1800, 4.1,11);

INSERT INTO Rewards([CompaRatio],[RecognitionPoints],[PayScale],[EmployeeID]) VALUES(0.72, 100, 4.1,12);

SELECT \* FROM Rewards

--Recognition Table

INSERT INTO Recognition([RecognitionDate],[RecognitionAmount],[RewardID]) VALUES('11/22/2019', 100,1);

INSERT INTO Recognition([RecognitionDate],[RecognitionAmount],[RewardID]) VALUES('6/7/2019', 50,2);

INSERT INTO Recognition([RecognitionDate],[RecognitionAmount],[RewardID]) VALUES('1/4/2019', 0,3);

INSERT INTO Recognition([RecognitionDate],[RecognitionAmount],[RewardID]) VALUES('2/1/2019', 500,4);

INSERT INTO Recognition([RecognitionDate],[RecognitionAmount],[RewardID]) VALUES('10/31/2019', 25,5);

INSERT INTO Recognition([RecognitionDate],[RecognitionAmount],[RewardID]) VALUES('4/4/2019', 1000,6);

INSERT INTO Recognition([RecognitionDate],[RecognitionAmount],[RewardID]) VALUES('1/6/2019', 100,7);

INSERT INTO Recognition([RecognitionDate],[RecognitionAmount],[RewardID]) VALUES('9/17/2019', 700,8);

INSERT INTO Recognition([RecognitionDate],[RecognitionAmount],[RewardID]) VALUES('1/16/2019', 350,9);

INSERT INTO Recognition([RecognitionDate],[RecognitionAmount],[RewardID]) VALUES('3/12/2019', 1200,10);

INSERT INTO Recognition([RecognitionDate],[RecognitionAmount],[RewardID]) VALUES('1/14/2019', 1800,11);

INSERT INTO Recognition([RecognitionDate],[RecognitionAmount],[RewardID]) VALUES('5/3/2019', 100,12);

SELECT \* FROM Recognition

--TalentCard Table

INSERT INTO TalentCard([HiPoStatus],[Mobility], [ManagerStatus], [EmployeeID]) VALUES ('No', 'Mobile', 'No', 1);

INSERT INTO TalentCard([HiPoStatus],[Mobility],[HiPoStatusDate], [ManagerStatus], [EmployeeID]) VALUES ('Yes', 'Mobile', '4/2/1966', 'No', 2);

INSERT INTO TalentCard([HiPoStatus],[Mobility], [ManagerStatus], [EmployeeID]) VALUES ('No', 'Not Mobile','No', 3);

INSERT INTO TalentCard([HiPoStatus],[Mobility], [ManagerStatus], [EmployeeID]) VALUES ('No', 'Limited Mobility', 'No', 4);

INSERT INTO TalentCard([HiPoStatus],[Mobility],[HiPoStatusDate], [ManagerStatus], [EmployeeID]) VALUES ('Yes', 'Mobile', '11/14/1984', 'No', 5);

INSERT INTO TalentCard([HiPoStatus],[Mobility],[HiPoStatusDate], [ManagerStatus], [EmployeeID]) VALUES ('Yes', 'Not Mobile', '3/1/1939', 'Yes', 6);

INSERT INTO TalentCard([HiPoStatus],[Mobility], [ManagerStatus], [EmployeeID]) VALUES ('No', 'Mobile', 'No', 7);

INSERT INTO TalentCard([HiPoStatus],[Mobility],[HiPoStatusDate], [ManagerStatus], [EmployeeID]) VALUES ('Yes', 'Limited Mobility', '6/16/2008', 'Yes', 8);

INSERT INTO TalentCard([HiPoStatus],[Mobility], [ManagerStatus], [EmployeeID]) VALUES ('No', 'Not Mobile', 'No', 9);

INSERT INTO TalentCard([HiPoStatus],[Mobility],[HiPoStatusDate], [ManagerStatus], [EmployeeID]) VALUES ('Yes', 'Mobile', '1/1/1963', 'Yes', 10);

INSERT INTO TalentCard([HiPoStatus],[Mobility], [ManagerStatus], [EmployeeID]) VALUES ('No', 'Mobile', 'No', 11);

INSERT INTO TalentCard([HiPoStatus],[Mobility], [ManagerStatus], [EmployeeID]) VALUES ('No', 'Mobile', 'No', 12);

SELECT \* FROM TalentCard

--Position Table

INSERT INTO Position([PositionStartDate],[TalentCardID]) VALUES ('12/2/2001', 1);

INSERT INTO Position([PositionStartDate],[TalentCardID]) VALUES ('11/2/2004', 2);

INSERT INTO Position([PositionStartDate],[TalentCardID]) VALUES ('12/2/2001', 3);

INSERT INTO Position([PositionStartDate],[TalentCardID]) VALUES ('12/2/2001', 4);

INSERT INTO Position([PositionStartDate],[TalentCardID]) VALUES ('12/2/2001', 5);

INSERT INTO Position([PositionStartDate],[TalentCardID]) VALUES ('12/2/2001', 6);

INSERT INTO Position([PositionStartDate],[TalentCardID]) VALUES ('12/25/2004', 7);

INSERT INTO Position([PositionStartDate],[TalentCardID]) VALUES ('12/2/2001', 8);

INSERT INTO Position([PositionStartDate],[TalentCardID]) VALUES ('12/2/2001', 9);

INSERT INTO Position([PositionStartDate],[TalentCardID]) VALUES ('12/25/2004', 10);

INSERT INTO Position([PositionStartDate],[TalentCardID]) VALUES ('1/2/2003', 11);

INSERT INTO Position([PositionStartDate],[TalentCardID]) VALUES ('12/2/2001', 12);

SELECT \* FROM Position

--PositionJobFamily Table

INSERT INTO PositionJobFamily([TypeName],[PositionID]) VALUES ('Customer Service Rep', 1);

INSERT INTO PositionJobFamily([TypeName],[PositionID]) VALUES ('Communications', 2);

INSERT INTO PositionJobFamily([TypeName],[PositionID]) VALUES ('Human Resources', 3);

INSERT INTO PositionJobFamily([TypeName],[PositionID]) VALUES ('Sales', 4);

INSERT INTO PositionJobFamily([TypeName],[PositionID]) VALUES ('Marketing', 5);

INSERT INTO PositionJobFamily([TypeName],[PositionID]) VALUES ('Employee Health and Safety', 6);

INSERT INTO PositionJobFamily([TypeName],[PositionID]) VALUES ('Information Services', 7);

INSERT INTO PositionJobFamily([TypeName],[PositionID]) VALUES ('Marketing', 8);

INSERT INTO PositionJobFamily([TypeName],[PositionID]) VALUES ('Sales', 9);

INSERT INTO PositionJobFamily([TypeName],[PositionID]) VALUES ('Research and Development', 10);

INSERT INTO PositionJobFamily([TypeName],[PositionID]) VALUES ('Customer Service Rep', 11);

INSERT INTO PositionJobFamily([TypeName],[PositionID]) VALUES ('Finance', 12);

SELECT \* FROM PositionJobFamily

/\*

Answers to Data Questions (SQL SELECT Statements)

The following statements are views that will answer the data questions posed by the stakeholders.

\*/

/\*

1. Which business units have the highest:

a. Number of employees

b. Resignation

c. Engagement

d. NPS

e. Span of Control

\*/

--Highest # of employees

SELECT TOP 3

EmployeeCore.BusinessUnitID,

COUNT(EmployeeCore.BusinessUnitID) AS BusinessHC

FROM

EmployeeCore

INNER JOIN BusinessUnit ON EmployeeCore.BusinessUnitID = BusinessUnit.BusinessUnitID

GROUP BY

EmployeeCore.BusinessUnitID

ORDER BY

COUNT(EmployeeCore.BusinessUnitID) DESC;

GO

SELECT \* FROM BusinessUnit

--Highest resignation rate

--Relevant to write at 'top 3' in the case of acquisitions for company expansion

DROP VIEW HighestResignationRate

GO

CREATE VIEW HighestResignationRate AS

SELECT TOP 3

BusinessUnit.BusinessUnitID,

BusinessUnit.BUName,

BusinessUnit.ResignationRate AS ResignationRateRank

FROM

BusinessUnit

GO

SELECT \* FROM HighestResignationRate

--Highest engagement

DROP VIEW HighestEngagement

GO

CREATE VIEW HighestEngagement AS

SELECT TOP 3

BusinessUnit.BusinessUnitID,

BusinessUnit.BUName,

BusinessUnit.Engagement AS EngagementRank

FROM

BusinessUnit

ORDER BY (BusinessUnit.Engagement) DESC;

GO

--Highest NPS

DROP VIEW HighestNPS

GO

CREATE VIEW HighestNPS AS

SELECT TOP 3

BusinessUnit.BusinessUnitID,

BusinessUnit.BUName,

BusinessUnit.NPS AS NPS\_Rank

FROM

BusinessUnit

ORDER BY (BusinessUnit.NPS) DESC;

GO

--Highest Span of Control

DROP VIEW HighestSpanofControl

GO

CREATE VIEW HighestSpanofControl AS

SELECT TOP 3

BusinessUnit.BusinessUnitID,

BusinessUnit.BUName,

BusinessUnit.SpanOfControl AS SpanOfControlRank

FROM

BusinessUnit

ORDER BY (BusinessUnit.SpanOfControl) DESC;

GO

/\*2. How many employees are mobile?

a. How many employees are fully mobile?

\*/

DROP VIEW MobileCount

GO

CREATE VIEW MobileCount AS

SELECT

COUNT(distinct TalentCard.TalentCardID) AS FullyMobileCount

From

TalentCard

WHERE TalentCard.Mobility = 'Mobile'

GO

--Percent breakdown by Mobility for HC

DROP VIEW MobilityPercent

GO

CREATE VIEW MobilityPercent AS

SELECT TOP 3 TalentCard.Mobility, count(\*) \* 100.0 / sum(count(\*)) over() AS MobilityPercent

From TalentCard

Group By TalentCard.Mobility

Order BY (MobilityPercent) DESC;

GO

/\*

3. How many employees are HiPos?

\*/

DROP VIEW HiPoCount

GO

CREATE VIEW HiPoCount AS

SELECT

COUNT(distinct TalentCard.TalentCardID) AS HiPoCount

From

TalentCard

WHERE TalentCard.HiPoStatus = 'Yes'

GO

--Who are my Mobile employees? Are any HiPos?

DROP VIEW HiPoMobility

GO

CREATE VIEW HiPoMobility AS

SELECT TOP 100

EmployeeCore.FirstName,

EmployeeCore.LastName,

TalentCard.HiPoStatus,

TalentCard.Mobility

From

TalentCard

JOIN EmployeeCore on EmployeeCore.EmployeeID = TalentCard.EmployeeID

Where TalentCard.Mobility = 'Mobile'

Order By LastName, FirstName DESC

GO

--4. What percentage of employees are HiPos?

DROP VIEW HiPoPercent

GO

CREATE VIEW HiPoPercent AS

select TOP 2 TalentCard.HiPoStatus, count(\*) \* 100.0 / sum(count(\*)) over() AS HiPoPercent

from TalentCard

group by TalentCard.HiPoStatus

Order BY (HiPoPercent) DESC;

GO

--5. How many recognition points have top employees earned? What does their compensation look like?

DROP VIEW RecognitionPointsLeaders

GO

CREATE VIEW RecognitionPointsLeaders AS

SELECT Top 3

EmployeeCore.FirstName,

EmployeeCore.LastName,

Rewards.RecognitionPoints,

Rewards.CompaRatio

FROM Rewards

JOIN EmployeeCore on EmployeeCore.EmployeeID = Rewards.EmployeeID

GROUP BY

EmployeeCore.LastName,

EmployeeCore.Firstname,

Rewards.CompaRatio,

Rewards.RecognitionPoints

ORDER BY Rewards.RecognitionPoints DESC, EmployeeCore.LastName

GO

/\*

6. Which job family has the most employees?

\*/

DROP VIEW TopJobFamily

GO

CREATE VIEW TopJobFamily AS

SELECT

Max(distinct PositionJobFamily.TypeName) AS TopJobFamily

FROM PositionJobFamily

GO

--Create a procedure to update a employee's mobility

Drop Procedure Employee\_ChangeMobility

GO

CREATE PROCEDURE Employee\_ChangeMobility(@employeeid int, @newMobility varchar(30))

AS

BEGIN

UPDATE TalentCard SET Mobility = @newMobility

WHERE EmployeeID = @employeeid

END

GO

DROP Procedure Employee\_ChangeCompa

GO

CREATE PROCEDURE Employee\_ChangeCompa(@employeeid int, @newCompa decimal)

AS

BEGIN

UPDATE Rewards SET CompaRatio = @newCompa

WHERE EmployeeID = @employeeid

END

GO

Drop Function dbo.RewardsReceivedCount

GO

CREATE FUNCTION dbo.RewardsReceivedCount(@RecognitionID int)

RETURNS int AS --COUNT() is an integer, so return as an int

BEGIN

DECLARE @returnValue int --matches the functions return type

SELECT @returnValue = COUNT(RecognitionID) FROM Recognition

WHERE Recognition.RecognitionID = @RecognitionID

--Return @returnValue to the calling code.

RETURN @returnVALUE

END

GO

Select dbo.RewardsReceivedCount(6)

SELECT \* FROM Recognition

DROP Procedure CompanyDescriptives

GO

CREATE Procedure CompanyDescriptives AS

SELECT

Count(BusinessUnit.BusinessUnitID) as NumberOfBUs,

AVG(BusinessUnit.NPS) as Average\_NPS,

MIN(BusinessUnit.Engagement) as MinEngagement,

AVG(BusinessUnit.Engagement) as AvgEngagement,

MAX(BusinessUnit.SpanOfControl) as MaxSpanOfControl,

AVG(ResignationRate) as Average\_ResignationRate

FROM BusinessUnit

--View Descriptives Call

SELECT

Count(BusinessUnit.BusinessUnitID) as NumberOfBUs,

AVG(BusinessUnit.NPS) as Average\_NPS,

MIN(BusinessUnit.Engagement) as MinEngagement,

AVG(BusinessUnit.Engagement) as AvgEngagement,

MAX(BusinessUnit.SpanOfControl) as MaxSpanOfControl,

AVG(ResignationRate) as Average\_ResignationRate

FROM BusinessUnit

DROP Function dbo.BULookup

GO

CREATE FUNCTION dbo.BULookup(@EmployeeLastName varchar(30))

RETURNS int AS

BEGIN

DECLARE @returnValue int --Matches the function's return type

SELECT @returnValue = BusinessUnitID FROM EmployeeCore

WHERE LastName = @EmployeeLastName

--Send the vc\_TagID back to the caller

RETURN @returnValue

END

GO

SELECT dbo.BULookup('Musk')

SELECT \* From BusinessUnit

--Thank you Rad Chad for a great semester!! And remember… don’t forget to be awesome.