CIS 640: ADVANCED DATABASE SYSTEMS

EMPLOYEE REVIEW DATA REPORT

TEXT MINING PROJECT

AHMED ASHRAF SADEK, AMANDA LONG, ZAWAAD SHAH

Abstract

With large volumes of data being collected by businesses and corporations every day, it is important now, more than ever, to utilize different tools and techniques to analyze data being collected. Much of this data can often include large blocks of written text, which can make it difficult to discover even noticeable trends or patterns within the text. Text mining is a method of analyzing and extracting interesting insights that the data might include. This project discusses the use of text mining to analyze employee reviews in the FAANG (Facebook, Amazon, Apple, Netflix, Google and Microsoft) Database.

Table of Contents

Introduction

When considering the amount of data that is collected by businesses and corporations within the technology industry, there are many valuable reasons why gathering data is so important. From an employer’s standpoint, collecting employees’ personal information is generally a requirement for executing standard business tasks such as payroll and other administrative duties. From a business standpoint however, companies often look to collect other data including product reviews, customer demographics, marketing campaign data, etc. to assist in making optimal business decisions. Our project takes a look more specifically at employee reviews from the FAANG Database.

Our goal was to utilize different text mining tools to extract insights and discover possible trends within an employee review dataset to determine a set of key business questions. The areas in which our project focuses on are, determining the main drivers for pros and cons, company advice and overall summary, what the average ratings look like within each FAANG company, geographically where the largest locations of employees who submitted reviews, determining if we are able to build a text classification model to predict whether a review is positive or negative, and lastly to determine what other trends we may be able to find within the data that may not be clear or obvious under initial review of the data.

Initial Research and Assessment

During our initial research and assessment for this project, it was important to settle on some crucial decisions prior to implementation. Our discussion points covered a number of topics including initial text mining research, best choice of text analysis tools, choice of database management system, selecting a particular data set, data model discussion and different use cases.

We chose specifically to utilize SpaCy, which is an open-source software library for advanced natural language processing, for our text analysis, as opposed to Natural Language Toolkit(NLTK), which is a suite of libraries and programs for symbolic and statistical natural language processing. Many of the datasets we researched were already structured, so we opted for using Microsoft SQL Server as our main database management system. When it came to selecting a specific dataset, we decided we would be using employee reviews from the FAANG Database. Initially our plan was to extend the OES database, but after further discussion we opted against that. After our initial use case discussion, we decided our ultimate goal would be to do a sentiment analysis report on the employee reviews with visualized outputs. In order to introduce more flexibility to the project, we chose to use both Python and Power BI to produce visualizations from the output of our text analysis.

System Requirement Gathering

1. *Database*: We needed to decide on a reliable source for FAANG employee reviews. There were multiple options available online and each one of them had different sources. The repository we finally decided on was built by scraping the GlassDoor website since its inception in 2008. It was built in 2018, and it is the main source of reviews we are using.

2. *DBMS*: We decided on SQL Server since it is a reliable RDBMS and has a lot of support for integration with different platforms (such as Power BI) and services (such as Python's SQLAlchemy).

3. *Data Analysis tool*: We decided to use a combination of both Python (**SpaCy** for text analysis) and Power BI. This meant that we needed to learn how to use additional Python libraries such as **Pandas** for handling dataframes, **Numpy** for numerical analysis, **SciKit-Learn** for machine learning and text classification, **WordCloud** for word cloud visualizations and **SQLAlchemy** to connect to our DBMS.

4. *Research Objective*: We identified several business questions we want to answer using our data. The purpose of these questions was to guide our exploratory data analysis, our text analysis, and our project outputs. These questions are listed in our final findings section.

5. Project Outputs: Based on our research objectives, we decided to do three main things with our text mining activities.

    A) Build word clouds highlighting the most prominent keywords from employee reviews in different categories.

    B) Build a text classification model which can predict whether a review is positive (1) or negative (0).

C) Identify the top geographical locations where reviews originate from.

Database Design and Implementation

Microsoft SQL Server was selected as the Database Management System of choice to store the review data and to create tables to be used for analysis through Power BI. This decision was made as overall the data being used is structured with multiple columns such ratings, pros, cons, and dates. Queries can be run against these columns to obtain specific records when needed and the data can be organized for further analysis. For this project, the database was created using the Enterprise edition of SQL Server 2019 with SQL Server Management Studio as the integrated environment. The database can, however, be replicated with the included script on any version of SQL Server.

The employee reviews data was obtained online as a CSV file containing over 60,000 entries with the reviews obtained from Glassdoor.com. The data included values for

- Location: The location of the employee writing the review.
- Review date: The date the review was initially posted.
- Employee status: Is the reviewer a current or former employee.
- Position: The position or job of the reviewer.
- Summary: A summary of the review posted
- Pros: Pros of working at the company according to the reviewer
- Cons: Cons of working at the company according to the reviewer
- Advice to management: Advice to the management of the company according to the reviewer
- Overall rating: Overall rating by the reviewer
- Work balance stars: Overall rating by the reviewer
- Culture values stars: Rating of culture values by the reviewer
- Career opportunity stars: Rating of career opportunity by the reviewer

- Company benefits stars: Rating of company benefits by the reviewer
- Senior management stars: Rating of senior management by the reviewer
- Helpful count: A count of the number of people who found the review helpful
- Link: Link to the review
- Is anonymous: Yes or no whether the review is anonymous

The CSV document was imported into Microsoft SQL Server to create the employee_review table with some columns added including:

- Review_id: The primary key column of the employee_review table.
- Emp_id: Foreign key column connecting the table to the employee data table.
- Comp_id: Foreign key column connecting the table to the company data table.
- Avg_rating: An additional column which calculates the average rating for each review by taking each of the rating categories excluding the overall rating column. This is done to create better insights into the rating data for each rating category as the overall rating provided by reviewers is their personal rating as opposed to an average of individual categories.

A location index was created in the employee_review table for faster retrieval of location data. Reviews based on geographic data can provide various key insights into the data for further analysis.

Other tables created for the purpose of this database include an employee_data table which contains employee information. The columns created for this table include:

- Emp_id: The primary key for identifying employees
- Comp_id: Foreign key column connecting the table to the company data table for identifying the corresponding company the employee works at.
- Emp_fname: The employee's first name
- Emp_lname: The employee's last name
- Emp_email: The employee's email address

- Emp_phone: The employee’s phone number
- Emp_address: The employee’s current address

The values for these columns were generated randomly

The third table created is the company_data table. This table contains company information, that is the companies being reviewed by the employees. The columns for this table include:

- Comp_id: The primary key for identifying the company
- Comp_name: The name of the company. For this dataset, reviews for 6 companies were included: Facebook, Amazon, Apple, Netflix, Google, and Microsoft. These companies are often collectively called FAANG companies and are the major software and information technology companies in the industry.

The last table is the results table. This table was created to store the results obtained from text-mining and includes the visualizations and outputs. The files are added and stored using SQL Server’s FILESTREAM function. Files of any type can be stored in this way. The columns for this table include

- Results_id: A required unique identifying column representing each row containing a file with the ROWGUIDCOL and UNIQUE attributes.
- Root_directory: The path in the local disk where the file is stored
- File_link: A sharepoint link where the files are stored online and users can view them
- File_name: The name of the file
- File_create_date: The date the file was added to the database to maintain a historical record
- File_stream_col: Stores the content of the file in binary attribute

The database with just the objects and schema can be replicated by using the following script or Appendix A. The script for replicating the database with the values are included in Appendix B from this report.

```
USE [master]
GO
/****** Object:  Database [EmployeeReview]   Script Date: 7/22/2021 1:02:13 PM ******/
CREATE DATABASE [EmployeeReview]
 CONTAINMENT = NONE
 ON  PRIMARY
( NAME = N'EmployeeReview', FILENAME = N'D:\Microsoft SQL
Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\EmployeeReview.mdf' , SIZE = 270336KB , MAXSIZE =
UNLIMITED, FILEGROWTH = 65536KB ),
 FILEGROUP [Results] CONTAINS FILESTREAM  DEFAULT
( NAME = N'Results', FILENAME = N'D:\CIS 640\Results\Results' , MAXSIZE = UNLIMITED)
 LOG ON
( NAME = N'EmployeeReview_log', FILENAME = N'D:\Microsoft SQL
Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\EmployeeReview_log.ldf' , SIZE = 335872KB , MAXSIZE
= 2048GB , FILEGROWTH = 65536KB )
 WITH CATALOG_COLLATION = DATABASE_DEFAULT
GO
ALTER DATABASE [EmployeeReview] SET COMPATIBILITY_LEVEL = 150
GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [EmployeeReview].[dbo].[sp_fulltext_database] @action = 'enable'
end
GO
ALTER DATABASE [EmployeeReview] SET ANSI_NULL_DEFAULT OFF
GO
ALTER DATABASE [EmployeeReview] SET ANSI_NULLS OFF
GO
ALTER DATABASE [EmployeeReview] SET ANSI_PADDING OFF
GO
ALTER DATABASE [EmployeeReview] SET ANSI_WARNINGS OFF
GO
ALTER DATABASE [EmployeeReview] SET ARITHABORT OFF
GO
ALTER DATABASE [EmployeeReview] SET AUTO_CLOSE OFF
GO
ALTER DATABASE [EmployeeReview] SET AUTO_SHRINK OFF
GO
ALTER DATABASE [EmployeeReview] SET AUTO_UPDATE_STATISTICS ON
GO
ALTER DATABASE [EmployeeReview] SET CURSOR_CLOSE_ON_COMMIT OFF
GO
ALTER DATABASE [EmployeeReview] SET CURSOR_DEFAULT  GLOBAL
GO
ALTER DATABASE [EmployeeReview] SET CONCAT_NULL_YIELDS_NULL OFF
GO
ALTER DATABASE [EmployeeReview] SET NUMERIC_ROUNDABORT OFF
```

```
GO
ALTER DATABASE [EmployeeReview] SET QUOTED_IDENTIFIER OFF
GO
ALTER DATABASE [EmployeeReview] SET RECURSIVE_TRIGGERS OFF
GO
ALTER DATABASE [EmployeeReview] SET  DISABLE_BROKER
GO
ALTER DATABASE [EmployeeReview] SET AUTO_UPDATE_STATISTICS_ASYNC OFF
GO
ALTER DATABASE [EmployeeReview] SET DATE_CORRELATION_OPTIMIZATION OFF
GO
ALTER DATABASE [EmployeeReview] SET TRUSTWORTHY OFF
GO
ALTER DATABASE [EmployeeReview] SET ALLOW_SNAPSHOT_ISOLATION OFF
GO
ALTER DATABASE [EmployeeReview] SET PARAMETERIZATION SIMPLE
GO
ALTER DATABASE [EmployeeReview] SET READ_COMMITTED_SNAPSHOT OFF
GO
ALTER DATABASE [EmployeeReview] SET HONOR_BROKER_PRIORITY OFF
GO
ALTER DATABASE [EmployeeReview] SET RECOVERY FULL
GO
ALTER DATABASE [EmployeeReview] SET  MULTI_USER
GO
ALTER DATABASE [EmployeeReview] SET PAGE_VERIFY CHECKSUM
GO
ALTER DATABASE [EmployeeReview] SET DB_CHAINING OFF
GO
ALTER DATABASE [EmployeeReview] SET FILESTREAM( NON_TRANSACTED_ACCESS = OFF )
GO
ALTER DATABASE [EmployeeReview] SET TARGET_RECOVERY_TIME = 60 SECONDS
GO
ALTER DATABASE [EmployeeReview] SET DELAYED_DURABILITY = DISABLED
GO
ALTER DATABASE [EmployeeReview] SET ACCELERATED_DATABASE_RECOVERY = OFF
GO
EXEC sys.sp_db_vardecimal_storage_format N'EmployeeReview', N'ON'
GO
ALTER DATABASE [EmployeeReview] SET QUERY_STORE = OFF
GO
USE [EmployeeReview]
GO
/****** Object:  Table [dbo].[company_data]    Script Date: 7/22/2021 1:02:13 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
```

```
GO
CREATE TABLE [dbo].[company_data](
    [comp_id] [int] NOT NULL,
    [comp_name] [nvarchar](50) NOT NULL,
 CONSTRAINT [PK_company_data_comp_id] PRIMARY KEY CLUSTERED
(
    [comp_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[employee_data]   Script Date: 7/22/2021 1:02:13 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[employee_data](
    [emp_id] [int] NOT NULL,
    [comp_id] [int] NULL,
    [emp_fname] [nvarchar](50) NULL,
    [emp_lname] [nvarchar](50) NULL,
    [emp_email] [nvarchar](50) NULL,
    [emp_phone] [nvarchar](50) NULL,
    [emp_address] [nvarchar](100) NULL,
 CONSTRAINT [PK_employee_data_emp_id] PRIMARY KEY CLUSTERED
(
    [emp_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[employee_reviews]        Script Date: 7/22/2021 1:02:13 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[employee_reviews](
    [review_id] [int] NOT NULL,
    [emp_id] [int] NOT NULL,
    [comp_id] [int] NOT NULL,
    [location] [nvarchar](255) NULL,
    [review_date] [datetime] NULL,
    [employee_status] [nvarchar](255) NULL,
    [position] [nvarchar](255) NULL,
    [summary] [nvarchar](255) NULL,
    [pros] [nvarchar](max) NULL,
    [cons] [nvarchar](max) NULL,
    [advice_to_mgmt] [nvarchar](max) NULL,
    [overall_ratings] [float] NULL,
    [work_balance_stars] [float] NULL,
    [culture_values_stars] [float] NULL,
    [career_opportunities_stars] [float] NULL,
    [comp_benefit_stars] [float] NULL,
    [senior_management_stars] [float] NULL,
    [helpful_count] [float] NULL,
    [link] [nvarchar](255) NULL,
    [is_anonymous] [bit] NULL,
    [avg_rating] [decimal](2, 1) NULL,
 CONSTRAINT [PK_employee_reviews_review_id] PRIMARY KEY NONCLUSTERED
(
    [review_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Index [IDX_PK_employee_reviews_review_id]  Script Date: 7/22/2021 1:02:13 PM
******/
CREATE CLUSTERED INDEX [IDX_PK_employee_reviews_review_id] ON [dbo].[employee_reviews]
(
    [review_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[results]    Script Date: 7/22/2021 1:02:13 PM ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[results](
    [results_id] [uniqueidentifier] ROWGUIDCOL  NOT NULL,
    [root_directory] [varchar](max) NULL,
    [file_link] [varchar](max) NULL,
    [file_name] [varchar](max) NULL,
    [file_create_date] [datetime] NULL,
    [file_stream_col] [varbinary](max) FILESTREAM  NULL,
UNIQUE NONCLUSTERED
(
    [results_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY] FILESTREAM_ON [Results]
GO
SET ANSI_PADDING ON
GO
/****** Object:  Index [locationIndex]  Script Date: 7/22/2021 1:02:13 PM ******/
CREATE NONCLUSTERED INDEX [locationIndex] ON [dbo].[employee_reviews]
(
    [location] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
GO
ALTER TABLE [dbo].[employee_data]  WITH CHECK ADD  CONSTRAINT [FK_employee_data_comp_id]
FOREIGN KEY([comp_id])
REFERENCES [dbo].[company_data] ([comp_id])
GO
ALTER TABLE [dbo].[employee_data] CHECK CONSTRAINT [FK_employee_data_comp_id]
GO
ALTER TABLE [dbo].[employee_reviews]  WITH CHECK ADD  CONSTRAINT
[FK_employee_reviews_comp_id] FOREIGN KEY([comp_id])
REFERENCES [dbo].[company_data] ([comp_id])
GO
ALTER TABLE [dbo].[employee_reviews] CHECK CONSTRAINT [FK_employee_reviews_comp_id]
GO
ALTER TABLE [dbo].[employee_reviews]  WITH CHECK ADD  CONSTRAINT
[FK_employee_reviews_emp_id] FOREIGN KEY([emp_id])
REFERENCES [dbo].[employee_data] ([emp_id])
GO
ALTER TABLE [dbo].[employee_reviews] CHECK CONSTRAINT [FK_employee_reviews_emp_id]
GO
USE [master]
GO
ALTER DATABASE [EmployeeReview] SET  READ_WRITE
GO
```

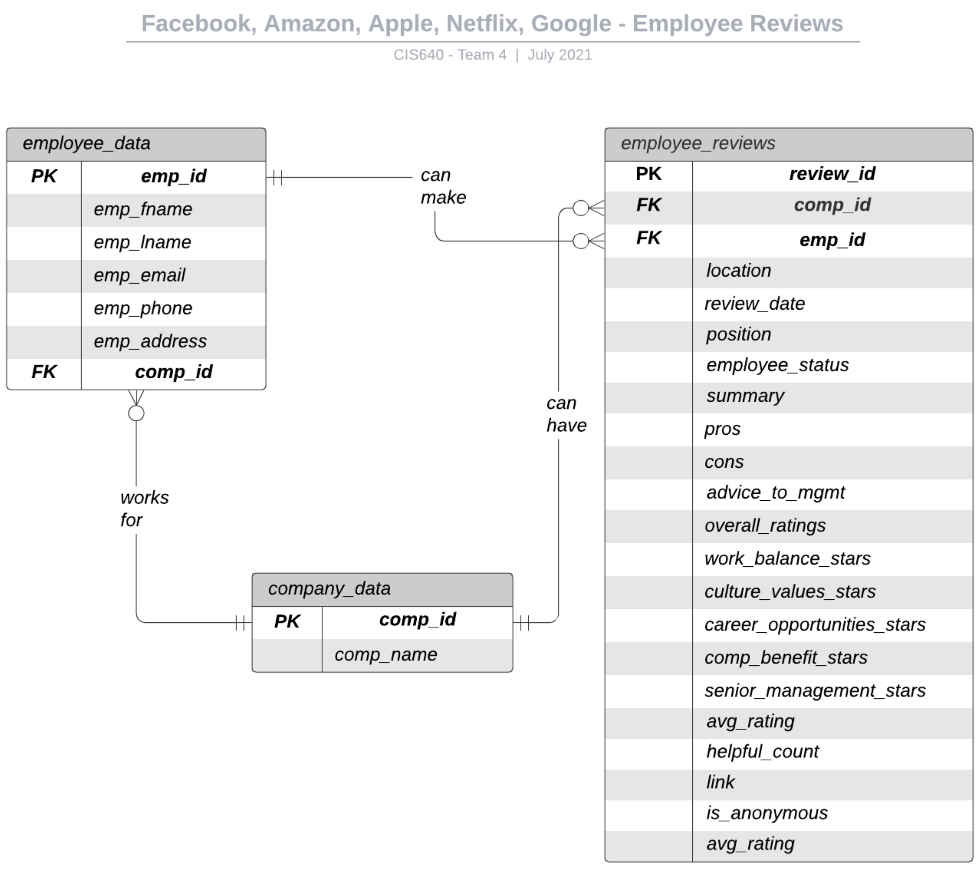


Fig. Entity relationship diagram

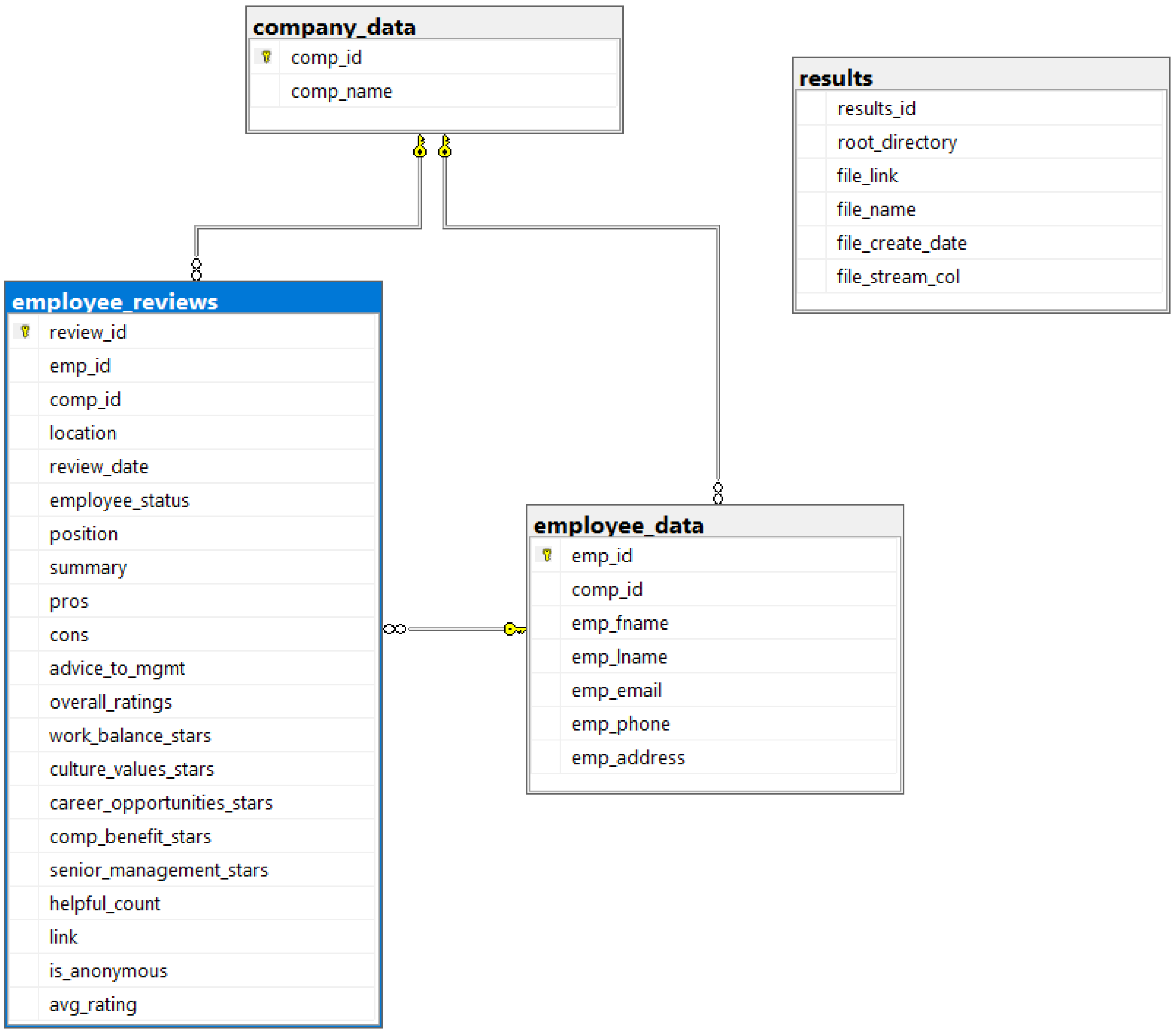
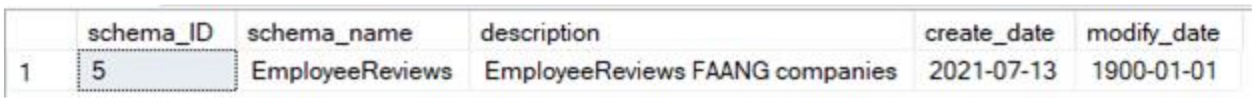


Fig. Database diagram

Data governance use case:

Schema name: EmployeeReview



	schema_ID	schema_name	description	create_date	modify_date
1	5	EmployeeReviews	EmployeeReviews FAANG companies	2021-07-13	1900-01-01

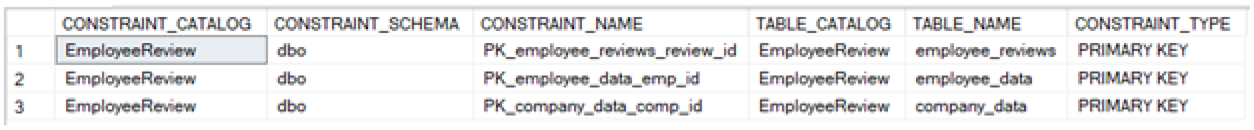
Area: Machine Learning

Module: Text Mining

Tables added to DGT000121- DGT000124

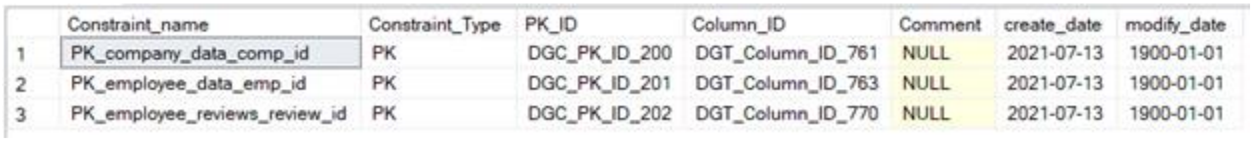
Columns added to DGT_Column_ID_761- DGT_Column_ID_800

Primary Keys:



	CONSTRAINT_CATALOG	CONSTRAINT_SCHEMA	CONSTRAINT_NAME	TABLE_CATALOG	TABLE_NAME	CONSTRAINT_TYPE
1	EmployeeReview	dbo	PK_employee_reviews_review_id	EmployeeReview	employee_reviews	PRIMARY KEY
2	EmployeeReview	dbo	PK_employee_data_emp_id	EmployeeReview	employee_data	PRIMARY KEY
3	EmployeeReview	dbo	PK_company_data_comp_id	EmployeeReview	company_data	PRIMARY KEY

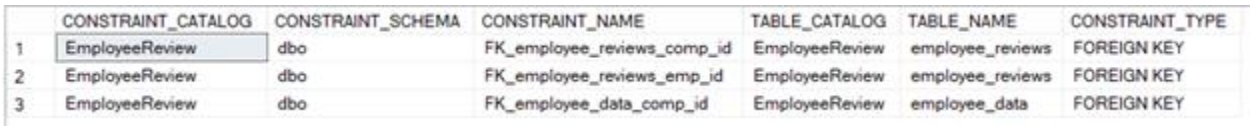
Results from EmployeeReview database



	Constraint_name	Constraint_Type	PK_ID	Column_ID	Comment	create_date	modify_date
1	PK_company_data_comp_id	PK	DGC_PK_ID_200	DGT_Column_ID_761	NULL	2021-07-13	1900-01-01
2	PK_employee_data_emp_id	PK	DGC_PK_ID_201	DGT_Column_ID_763	NULL	2021-07-13	1900-01-01
3	PK_employee_reviews_review_id	PK	DGC_PK_ID_202	DGT_Column_ID_770	NULL	2021-07-13	1900-01-01

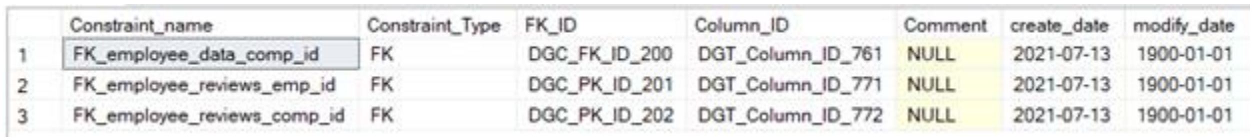
Results from Zoeta database after insert

Foreign Keys:



	CONSTRAINT_CATALOG	CONSTRAINT_SCHEMA	CONSTRAINT_NAME	TABLE_CATALOG	TABLE_NAME	CONSTRAINT_TYPE
1	EmployeeReview	dbo	FK_employee_reviews_comp_id	EmployeeReview	employee_reviews	FOREIGN KEY
2	EmployeeReview	dbo	FK_employee_reviews_emp_id	EmployeeReview	employee_reviews	FOREIGN KEY
3	EmployeeReview	dbo	FK_employee_data_comp_id	EmployeeReview	employee_data	FOREIGN KEY

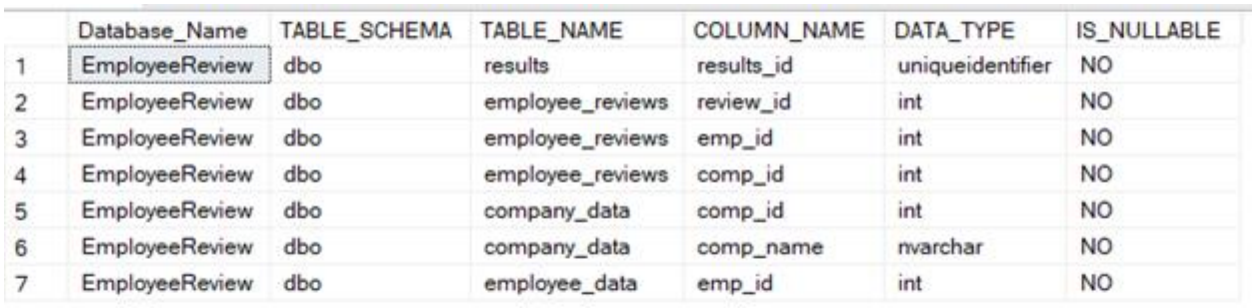
Results from EmployeeReview database



	Constraint_name	Constraint_Type	FK_ID	Column_ID	Comment	create_date	modify_date
1	FK_employee_data_comp_id	FK	DGC_FK_ID_200	DGT_Column_ID_761	NULL	2021-07-13	1900-01-01
2	FK_employee_reviews_emp_id	FK	DGC_PK_ID_201	DGT_Column_ID_771	NULL	2021-07-13	1900-01-01
3	FK_employee_reviews_comp_id	FK	DGC_PK_ID_202	DGT_Column_ID_772	NULL	2021-07-13	1900-01-01

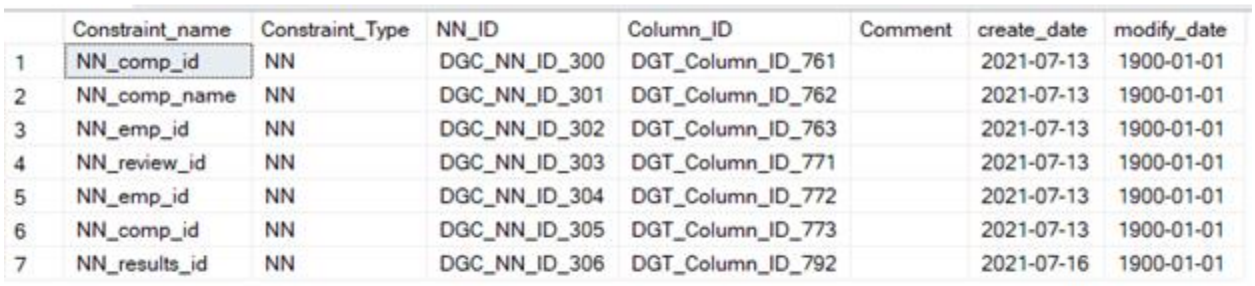
Results from Zoeta database after insert

Not Nulls:



	Database_Name	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	DATA_TYPE	IS_NULLABLE
1	EmployeeReview	dbo	results	results_id	uniqueidentifier	NO
2	EmployeeReview	dbo	employee_reviews	review_id	int	NO
3	EmployeeReview	dbo	employee_reviews	emp_id	int	NO
4	EmployeeReview	dbo	employee_reviews	comp_id	int	NO
5	EmployeeReview	dbo	company_data	comp_id	int	NO
6	EmployeeReview	dbo	company_data	comp_name	nvarchar	NO
7	EmployeeReview	dbo	employee_data	emp_id	int	NO

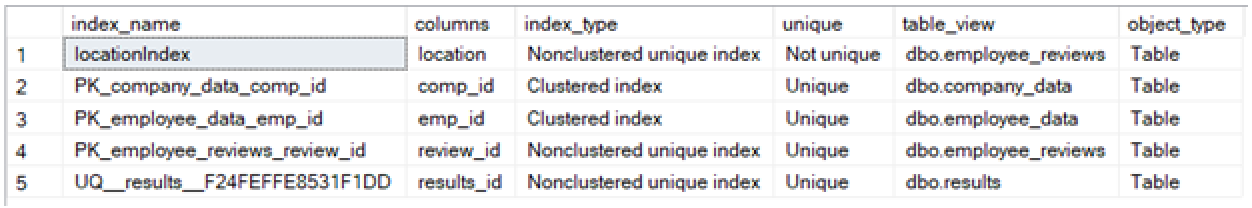
Results from EmployeeReview database



	Constraint_name	Constraint_Type	NN_ID	Column_ID	Comment	create_date	modify_date
1	NN_comp_id	NN	DGC_NN_ID_300	DGT_Column_ID_761		2021-07-13	1900-01-01
2	NN_comp_name	NN	DGC_NN_ID_301	DGT_Column_ID_762		2021-07-13	1900-01-01
3	NN_emp_id	NN	DGC_NN_ID_302	DGT_Column_ID_763		2021-07-13	1900-01-01
4	NN_review_id	NN	DGC_NN_ID_303	DGT_Column_ID_771		2021-07-13	1900-01-01
5	NN_emp_id	NN	DGC_NN_ID_304	DGT_Column_ID_772		2021-07-13	1900-01-01
6	NN_comp_id	NN	DGC_NN_ID_305	DGT_Column_ID_773		2021-07-13	1900-01-01
7	NN_results_id	NN	DGC_NN_ID_306	DGT_Column_ID_792		2021-07-16	1900-01-01

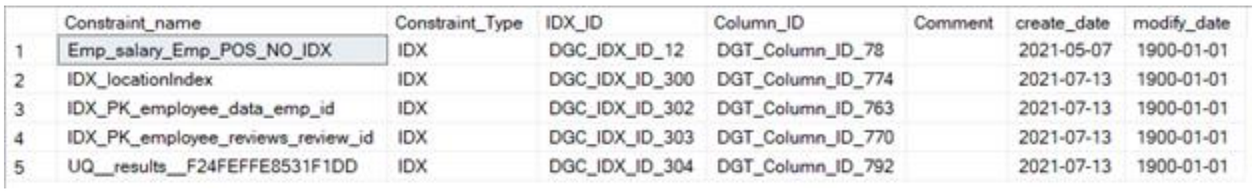
Results from Zoeta database after insert

Indexes:



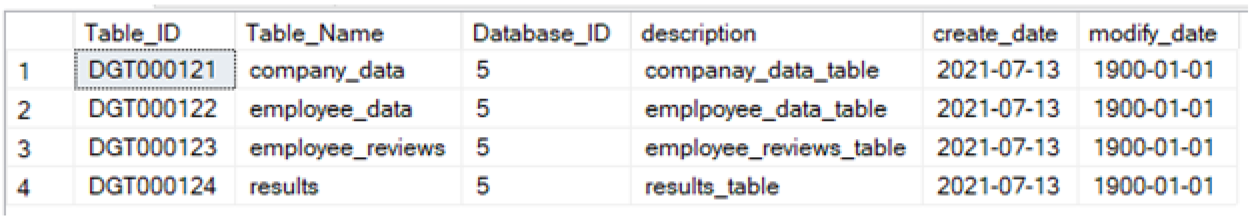
	index_name	columns	index_type	unique	table_view	object_type
1	locationIndex	location	Nonclustered unique index	Not unique	dbo.employee_reviews	Table
2	PK_company_data_comp_id	comp_id	Clustered index	Unique	dbo.company_data	Table
3	PK_employee_data_emp_id	emp_id	Clustered index	Unique	dbo.employee_data	Table
4	PK_employee_reviews_review_id	review_id	Nonclustered unique index	Unique	dbo.employee_reviews	Table
5	UQ__results__F24FEFFE8531F1DD	results_id	Nonclustered unique index	Unique	dbo.results	Table

Results from EmployeeReview database



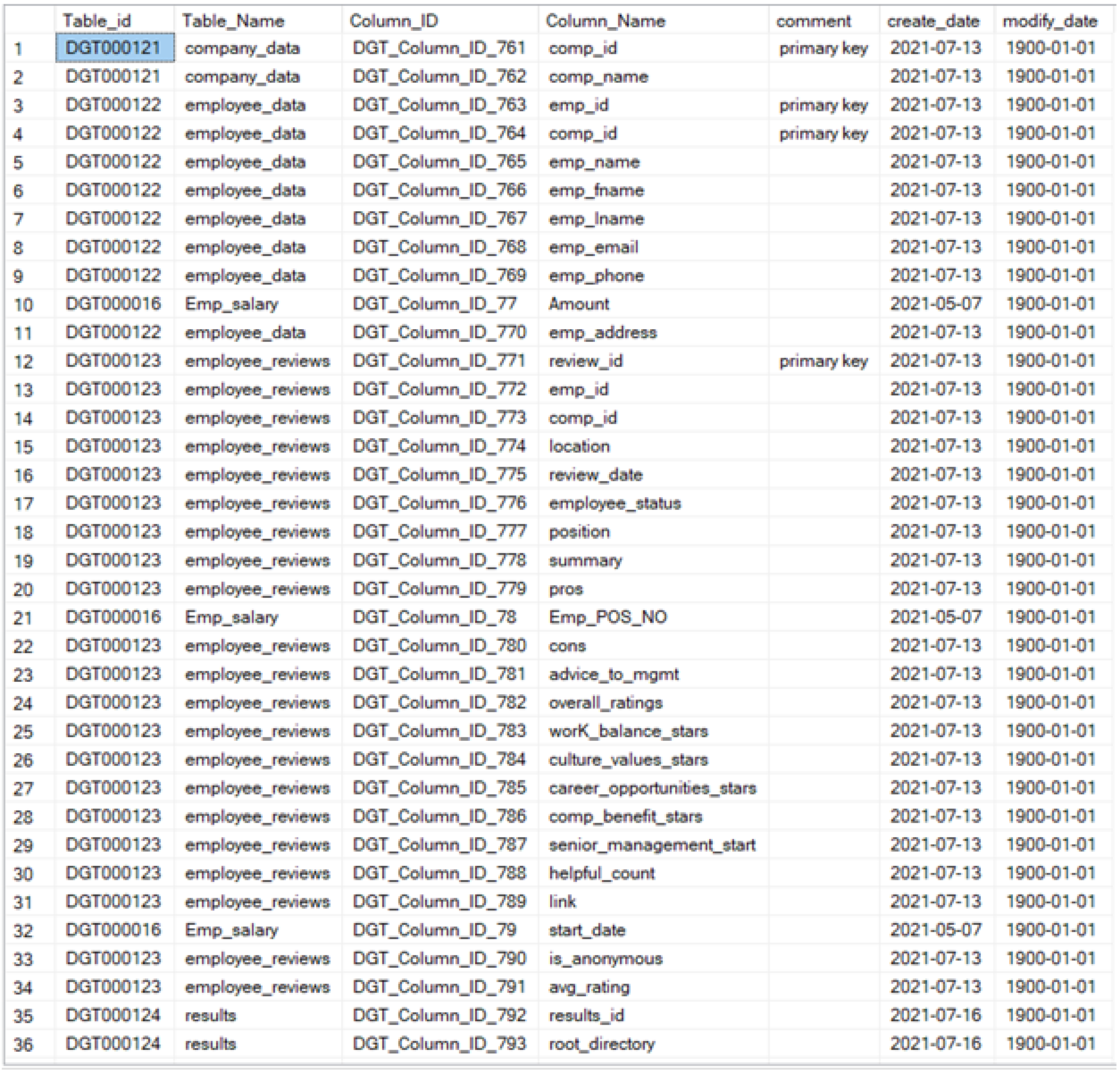
	Constraint_name	Constraint_Type	IDX_ID	Column_ID	Comment	create_date	modify_date
1	Emp_salary_Emp_POS_NO_IDX	IDX	DGC_IDX_ID_12	DGT_Column_ID_78		2021-05-07	1900-01-01
2	IDX_locationIndex	IDX	DGC_IDX_ID_300	DGT_Column_ID_774		2021-07-13	1900-01-01
3	IDX_PK_employee_data_emp_id	IDX	DGC_IDX_ID_302	DGT_Column_ID_763		2021-07-13	1900-01-01
4	IDX_PK_employee_reviews_review_id	IDX	DGC_IDX_ID_303	DGT_Column_ID_770		2021-07-13	1900-01-01
5	UQ__results__F24FEFFE8531F1DD	IDX	DGC_IDX_ID_304	DGT_Column_ID_792		2021-07-13	1900-01-01

Results from Zoeta database after insert



	Table_ID	Table_Name	Database_ID	description	create_date	modify_date
1	DGT000121	company_data	5	companay_data_table	2021-07-13	1900-01-01
2	DGT000122	employee_data	5	emplpoyee_data_table	2021-07-13	1900-01-01
3	DGT000123	employee_reviews	5	employee_reviews_table	2021-07-13	1900-01-01
4	DGT000124	results	5	results_table	2021-07-13	1900-01-01

Tables in EmployeeReview database inserted into Zoeta database



	Table_id	Table_Name	Column_ID	Column_Name	comment	create_date	modify_date
1	DGT000121	company_data	DGT_Column_ID_761	comp_id	primary key	2021-07-13	1900-01-01
2	DGT000121	company_data	DGT_Column_ID_762	comp_name		2021-07-13	1900-01-01
3	DGT000122	employee_data	DGT_Column_ID_763	emp_id	primary key	2021-07-13	1900-01-01
4	DGT000122	employee_data	DGT_Column_ID_764	comp_id	primary key	2021-07-13	1900-01-01
5	DGT000122	employee_data	DGT_Column_ID_765	emp_name		2021-07-13	1900-01-01
6	DGT000122	employee_data	DGT_Column_ID_766	emp_fname		2021-07-13	1900-01-01
7	DGT000122	employee_data	DGT_Column_ID_767	emp_lname		2021-07-13	1900-01-01
8	DGT000122	employee_data	DGT_Column_ID_768	emp_email		2021-07-13	1900-01-01
9	DGT000122	employee_data	DGT_Column_ID_769	emp_phone		2021-07-13	1900-01-01
10	DGT000016	Emp_salary	DGT_Column_ID_77	Amount		2021-05-07	1900-01-01
11	DGT000122	employee_data	DGT_Column_ID_770	emp_address		2021-07-13	1900-01-01
12	DGT000123	employee_reviews	DGT_Column_ID_771	review_id	primary key	2021-07-13	1900-01-01
13	DGT000123	employee_reviews	DGT_Column_ID_772	emp_id		2021-07-13	1900-01-01
14	DGT000123	employee_reviews	DGT_Column_ID_773	comp_id		2021-07-13	1900-01-01
15	DGT000123	employee_reviews	DGT_Column_ID_774	location		2021-07-13	1900-01-01
16	DGT000123	employee_reviews	DGT_Column_ID_775	review_date		2021-07-13	1900-01-01
17	DGT000123	employee_reviews	DGT_Column_ID_776	employee_status		2021-07-13	1900-01-01
18	DGT000123	employee_reviews	DGT_Column_ID_777	position		2021-07-13	1900-01-01
19	DGT000123	employee_reviews	DGT_Column_ID_778	summary		2021-07-13	1900-01-01
20	DGT000123	employee_reviews	DGT_Column_ID_779	pros		2021-07-13	1900-01-01
21	DGT000016	Emp_salary	DGT_Column_ID_78	Emp_POS_NO		2021-05-07	1900-01-01
22	DGT000123	employee_reviews	DGT_Column_ID_780	cons		2021-07-13	1900-01-01
23	DGT000123	employee_reviews	DGT_Column_ID_781	advice_to_mgmt		2021-07-13	1900-01-01
24	DGT000123	employee_reviews	DGT_Column_ID_782	overall_ratings		2021-07-13	1900-01-01
25	DGT000123	employee_reviews	DGT_Column_ID_783	worK_balance_stars		2021-07-13	1900-01-01
26	DGT000123	employee_reviews	DGT_Column_ID_784	culture_values_stars		2021-07-13	1900-01-01
27	DGT000123	employee_reviews	DGT_Column_ID_785	career_opportunities_stars		2021-07-13	1900-01-01
28	DGT000123	employee_reviews	DGT_Column_ID_786	comp_benefit_stars		2021-07-13	1900-01-01
29	DGT000123	employee_reviews	DGT_Column_ID_787	senior_management_start		2021-07-13	1900-01-01
30	DGT000123	employee_reviews	DGT_Column_ID_788	helpful_count		2021-07-13	1900-01-01
31	DGT000123	employee_reviews	DGT_Column_ID_789	link		2021-07-13	1900-01-01
32	DGT000016	Emp_salary	DGT_Column_ID_79	start_date		2021-05-07	1900-01-01
33	DGT000123	employee_reviews	DGT_Column_ID_790	is_anonymous		2021-07-13	1900-01-01
34	DGT000123	employee_reviews	DGT_Column_ID_791	avg_rating		2021-07-13	1900-01-01
35	DGT000124	results	DGT_Column_ID_792	results_id		2021-07-16	1900-01-01
36	DGT000124	results	DGT_Column_ID_793	root_directory		2021-07-16	1900-01-01

Columns in EmployeeReview database inserted into Zoeta database

Scripts for querying primary key, foreign key, not nulls and indexes from EmployeeReview database

```
-- Primary keys
USE EmployeeReview
SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME, TABLE_CATALOG,
TABLE_NAME, CONSTRAINT_TYPE
FROM information_schema.table_constraints
WHERE constraint_type = 'Primary Key';

-- Foreign keys
USE EmployeeReview
SELECT CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME, TABLE_CATALOG,
TABLE_NAME, CONSTRAINT_TYPE
FROM information_schema.table_constraints
WHERE constraint_type = 'Foreign Key';

-- List of not null columns in EmployeeReview
SELECT TABLE_CATALOG AS Database_Name, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME,
DATA_TYPE, IS_NULLABLE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE IS_NULLABLE = 'NO'

-- List all indexes
SELECT i.[name] as index_name,
  substring(column_names, 1, len(column_names)-1) as [columns],
  case when i.[type] = 1 then 'Clustered index'
    when i.[type] = 2 then 'Nonclustered unique index'
    when i.[type] = 3 then 'XML index'
    when i.[type] = 4 then 'Spatial index'
    when i.[type] = 5 then 'Clustered columnstore index'
    when i.[type] = 6 then 'Nonclustered columnstore index'
    when i.[type] = 7 then 'Nonclustered hash index'
    end as index_type,
  case when i.is_unique = 1 then 'Unique'
    else 'Not unique' end as [unique],
  schema_name(t.schema_id) + '.' + t.[name] as table_view,
  case when t.[type] = 'U' then 'Table'
    when t.[type] = 'V' then 'View'
    end as [object_type]
FROM sys.objects t
  inner join sys.indexes i
    on t.object_id = i.object_id
  cross apply (select col.[name] + ', '
          from sys.index_columns ic
            inner join sys.columns col
              on ic.object_id = col.object_id
              and ic.column_id = col.column_id
```

```
                    where ic.object_id = t.object_id
                        AND ic.index_id = i.index_id
                            order by key_ordinal
                            for xml path ('') ) D (column_names)
WHERE t.is_ms_shipped <> 1 AND index_id > 0 ORDER BY i.[name];
```

Scripts for querying primary key, foreign key, not nulls, indexes, columns, tables, schema added to Zoeta database

```
--List primary keys
SELECT *
FROM DG_Column_PK
WHERE Column_ID IN ('DGT_Column_ID_761', 'DGT_Column_ID_763', 'DGT_Column_ID_770');

--List foreing keys
SELECT *
FROM DG_Column_FK
WHERE Column_ID IN ('DGT_Column_ID_761', 'DGT_Column_ID_771', 'DGT_Column_ID_772');

--List not nulls
SELECT *
FROM DG_Column_NotNull
WHERE Column_ID >= 'DGT_Column_ID_761' AND Column_ID <= 'DGT_Column_ID_792';

--List indexes
SELECT *
FROM DG_Column_Index
WHERE Column_ID >= 'DGT_Column_ID_763' AND Column_ID <= 'DGT_Column_ID_792';

--List schema
SELECT *
FROM DG_Schema
WHERE schema_ID = 5;

--List columns
SELECT *
FROM DG_Columns
WHERE Column_ID >= 'DGT_Column_ID_761' AND Column_ID <= 'DGT_Column_ID_800';

--List tables
SELECT *
FROM DG_Tables
WHERE Table_ID >= 'DGT000121' AND Table_ID <= 'DGT000125';
```

Text Analysis

SpaCy Text Analysis Process

We used SpaCy to do our initial text analysis from the employee reviews. SpaCy is an open-source software library for advanced natural language processing, which provides a great source of functions that can be applied to accomplish different forms of analysis. The main functions in which we utilized for our analysis Tokenization, Lemmatisation, part-of-speech tagging, entity recognition, dependency parsing and word-to-vector transformations. These functions cover the overall process we took to produce our final text-analysis results.

The first step in the process of text mining requires Tokenization, which is the process of splitting a piece of text into words, symbols, punctuation, spaces and other elements, thereby creating "tokens". This allows us to break the text up into individual elements that can be used to execute other functions.

One of the issues we come across after tokenization is dealing with Stop Words. Stop Words happen when the tokenization does not remove punctuation or tokenize the text correctly. This can include adverbs or verbs which have punctuation within the middle of the word. The next step in the process would be to get rid of the punctuation and Stop Words prior to any further analysis. To do this, we can use a simple 'if' statement to run against the text.

The next step in the process that we took is Lemmatization. Lemmatization is another function which further cleans up the text. It is essentially a form of normalizing the text by reducing words to their original base version. This process helps us provide more accurate results

Once all of the text was cleaned up with the Lemmatization process, we then ran the Part-of-Speech tagging, which defines a particular word's function within a sentence. This process identifies forms of speech like nouns, verbs, adjectives, etc. and is extremely useful for additional steps within the process.

From Part-of-Speech Tagging, we move to Entity Detection which helps us identify significant or important elements within the text including people, places, organizations, dates and other similar entities. With the Entity Detection complete, the next step was to use Dependency Parsing, which is used to show how a sentence is structured, and to determine the meaning of a sentence.

The last step we did was to use Word-to-Vector Representation. This is a necessary step to build a Classification Model for our sentiment analysis. In this process, each word is translated into a vector of numbers representation. The numbers communicate the relationship of a word to other words.

Text Classification Model

The goal of our text classification model is to determine whether or not a review is positive or negative. We needed to build a separate labeled dataset of reviews from the database we already had. We combined the pros column with the cons column in one single “review” column with an additional labelling column containing “1” for positive reviews and “0” for negative reviews.

After preparing the new dataframe, we started building our tokenizer function. This basically uses SpaCy to get a list of lemmatized tokens for each review excluding stop words and punctuation. This function will be used as the base for building the more advanced functions which we will use in the pipeline for building our machine learning model. These functions consist of a cleaner function, a vectorizer function, and the model function we will use.

Since this is a classification problem, we will be using the logistic regression model.It is a predictive algorithm that uses independent variables to predict a dependent categorical variable. Our dependent variable here is a binary category denoting the type of review (Positive: 1 or Negative: 0). Since we have more than 100,000 records in our combined dataframe, we should expect the accuracy to be high, but we will still do a split of the frame by randomly allocating 30% of the reviews to a test dataframe to verify the accuracy of our predictive model.

Connecting to Microsoft SQL Server from Python using SQLAlchemy

SQLAlchemy is a python library used to easily connect to a SQL database in python. The first step is to import the necessary libraries. The next step is to establish a connection to the database by specifying the server and database names. With the connection established and tested it is now possible to run queries against the database and obtain data to perform text analysis on by pulling the data into a Pandas dataframe. The following screenshot shows the script to perform these operations. In this instance Jupyter notebook (which is an open-source web application for creating live code and visualizations) is being used.

Importing Libraries.

```
In [1]: import pyodbc
        import sqlalchemy as sal
        from sqlalchemy import create_engine
        import pandas as pd
        import warnings
        warnings.filterwarnings('ignore')
```

Creating the Database Driver Engine.

```
In [2]: engine = sal.create_engine('mssql+pyodbc://DESKTOP-24EBAA8/EmployeeReview?driver=SQL Server?Trusted_Connection=yes')
```

Establishing the connection.

```
In [3]: conn=engine.connect()
```

Verifying the connection is functional by pulling table names from our database.

```
In [4]: print(engine.table_names())

['company_data', 'employee_data', 'employee_reviews', 'sysdiagrams']
```

Creating a list of column names for the table we want to do text analysis on: `employee_reviews`

```
In [5]: q=engine.execute("SELECT COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = 'employee_reviews' ORDER BY ORDINAL_POSI
        ls = []
        for row in q:
            ls.append(row[0])
        print(ls)

['review_id', 'comp_id', 'emp_id', 'pros', 'cons', 'overall_ratings', 'work_balance_stars', 'culture_values_stars', 'career_opp
ortunities_stars', 'comp_benefit_stars', 'senior_management_stars']
```

Pulling the data from SQL Server into a Pandas Dataframe.

```
In [6]: sql_query = pd.read_sql_query('SELECT * FROM EmployeeReview.dbo.employee_reviews', engine)
        df = pd.DataFrame(sql_query, columns = ls)
        df.head()
```

Out[6]:

	review_id	comp_id	emp_id	pros	cons	overall_ratings	work_balance_stars	culture_values_stars	career_opportunities_stars	comp_benefit_stars
0	1	1	1	People are smart and friendly	Bureaucracy is slowing things down	5.0	4.0	5.0	5.0	4.0
1	2	1	2	1) Food, food, food. 15+ cafes on main campus ...	1) Work/life balance. What balance? All those ...	4.0	2.0	3.0	3.0	5.0
2	3	1	3	* If you're a software engineer, you're among ...	* It *is* becoming larger, and with it comes g...	5.0	5.0	4.0	5.0	5.0
3	4	1	4	You can't find a more well-regarded company th...	I live in SF so the commute can take between 1...	5.0	2.0	5.0	5.0	4.0
4	5	1	5	Google is a world of its own. At every other c...	If you don't work in MTV (HQ), you will be giv...	5.0	5.0	5.0	5.0	5.0

Python Implementation code

```
import spacy as sp

import pandas as pd

import numpy as np

#loading the english module

nlp= sp.load('en_core_web_sm')


df = pd.read_csv('review_data.csv')


df.info()


#converting amazon pro reviews into a document

pros_amazon=df[df['company']=='amazon']['pros']

pros_amazon

pros_amazon.to_csv('pandas.txt', header=None, index=None, quoting =
None,sep=' ', mode='a')


#loading the document as text

with open('pandas.txt', 'r',encoding='utf8') as file:

    txt = file.read().replace('\n', '')




txt = txt[:999999]


#applying spacy pipeline on the document
```

```
#does all the work in advance

#creating a doc object which we can use to access the methods

doc= nlp(txt)

doc


# spacy provides a one-stop-shop for tasks commonly used in any NLP
project

# including:

# Tokenization

# Lemmatisation

# Part-of-speech tagging

# Entity recognition

# Dependency parsing

# Sentence recognition

# Word-to-vector transformations

# Many convenience methods for cleaning and normalising text




#Tokenization

#Tokenization is a foundational step in many NLP tasks.

# Tokenizing text is the process of splitting a piece of text into words,

# symbols, punctuation, spaces and other elements, thereby creating
"tokens".


token_list = []

for token in doc:
    token_list.append(token.text)
print(token_list[:100])


# We can see from the output that it doesn't remove punctuation
(',",!,.,etc)
# It also doesn't split verb and adverb ("was", "n't")
# we need to clean this by removing stop words and punctuation
#Stop words with SpaCy


stop = sp.lang.en.stop_words.STOP_WORDS
print('Number of Stop words:',len(stop))
print('Sample Stop Words:',list(enumerate(stop,10))[:5])



#Python provides a list of punctuation marks
import string
print(string.punctuation)


#But we can also use Spacy's is_punct and is_stop to clean them


token_clean_text = []
for token in doc:
    if token.is_punct == False and token.is_stop == False:
        token_clean_text.append(token.text)
```

```
print(token_clean_text[:100])

#we can see that we successfully removed stop words and punctuation.

#Now that we removed this, we can start doing more text cleaning with
spacy.


#we will do Lemmatization next, which are a form of normalizing text by
reducing

#words to their original base version ,for example "Runner, Running, Runs"
can

#all be boiled down to "Run"


tokens = []

for token in doc:

    if token.is_punct == False and token.is_stop == False:

        tokens.append(token)


for x in tokens[:100]:

    print(x,x.lemma_)

#We can see that "communicated" is reduced to "communicate", "friends" to
"friend", etc.


#Now for Part of Speech Tagging "POS"

#A word’s part of speech defines its

# function within a sentence.

for x in tokens[:100]:

    print(x,x.pos_)
```

```
#we can see that it defines stuff like

#  Adjectives, Nouns, Verbs, Numbers,etc..




#Entity Detection helps identify

#important elements like places, people,

#organizations, dates within a text.

#We can visualize it using the Displacy package from spacy.

sp.displacy.render(doc, style = "ent",jupyter = True)




#Dependency Parsing

#It is used to show how a sentence is structured,

#which helps us better

#determine the meaning of a sentence.

#It is also included in spacy and

#can be visulaised using displacy package


sample = txt.split('"')[1][:50]

test = nlp(sample)

sp.displacy.render(test,style='dep',jupyter=True)


#Word Vector Representation

#This is a necessary step in order to build a classification model for

#sentiment analysis
```

```
#Each word is represented as a vector of numbers, and those numbers

#communicate the relationship of the word to other words, similar to how

#GPS coordinates work.


#Example vector

amazon_vector = nlp('amazon')

print(amazon_vector.vector.shape)

print(amazon_vector.vector)


#We can use these vectors for text classification with other libraries.
```

```
from wordcloud import WordCloud as wc


len(tokens)

l = " ".join([token.orth_ for token in tokens])


cloud = wc(width=1600,height=800,collocations=False).generate(l)




import matplotlib.pyplot as plt


plt.figure(figsize=[20,10])
plt.imshow(cloud, interpolation='bilinear')

plt.axis("off")




df['company'].value_counts()

cons_ms =df[df['company']=='microsoft']['cons']

cons_ms

cons_ms.to_csv('ms_cons.txt', header=None, index=None, quoting =
None,sep=' ', mode='a')


with open('ms_cons.txt', 'r',encoding='utf8') as file:

    txt = file.read().replace('\n', '')




txt = txt[:999999]


#applying spacy pipeline on the document

#does all the work in advance

#creating a doc object which we can use to access the methods

doc= nlp(txt)


ms_tokens = []

for token in doc:

    if token.is_punct == False and token.is_stop == False:

        ms_tokens.append(token)
```

```
l_ms = " ".join([token.orth_ for token in ms_tokens])



ms_cloud =
wc(width=1600,height=800,collocations=False,background_color='white').gene
rate(l_ms)



plt.figure(figsize=[20,10])

plt.imshow(ms_cloud,interpolation='bilinear')

plt.axis("off")


###########################################

df['company'].value_counts()

df.info()

summary_apple =df[df['company']=='apple']['summary']

summary_apple

summary_apple .to_csv('apple_summary.txt', header=None, index=None,
quoting = None,sep=' ', mode='a')


with open('apple_summary.txt', 'r',encoding='utf8') as file:

    txt = file.read().replace('\n', '')
txt = txt[:999999]


#applying spacy pipeline on the document

#does all the work in advance

#creating a doc object which we can use to access the methods

doc= nlp(txt)


apple_tokens = []

for token in doc:

    if token.is_punct == False and token.is_stop == False:

        apple_tokens.append(token)




l_apple = " ".join([token.orth_ for token in apple_tokens])




apple_cloud =
wc(width=1600,height=800,collocations=False,colormap='autumn',background_c
olor='white').generate(l_apple)




plt.figure(figsize=[20,10])

plt.imshow(apple_cloud,interpolation='bilinear')

plt.axis("off")
```

```python
###################################################

df['company'].value_counts()

df.info()

google_adv =df[df['company']=='google']['advice.to.mgmt']

google_adv

google_adv .to_csv('google_adv.txt', header=None, index=None, quoting = None,sep=' ', mode='a')


with open('google_adv.txt', 'r',encoding='utf8') as file:

    txt = file.read().replace('\n', '')




txt = txt[:999999]


#applying spacy pipeline on the document

#does all the work in advance

#creating a doc object which we can use to access the methods

doc= nlp(txt)


g_tokens = []

for token in doc:

    if token.is_punct == False and token.is_stop == False:

        g_tokens.append(token)
```

```
l_g = " ".join([token.orth_ for token in g_tokens])




g_cloud =
wc(width=1600,height=800,collocations=False,colormap='Spectral',background
_color='black').generate(l_g)


t = [x for x in l_g.split() if x != 'none']

g_cloud =
wc(width=1600,height=800,collocations=False,colormap='Spectral',background
_color='black').generate(" ".join(t))




plt.figure(figsize=[20,10])

plt.imshow(g_cloud,interpolation='bilinear')

plt.axis("off")




########################################################################




df.info()

#### dates visualization

df['year'].plot(kind='bar')

import seaborn as sns

df['year'] = df['year'].astype(np.int64)
```

```
df['year'].value_counts().plot(kind='bar')

df['year'].fillna(2016,inplace=True)

sns.countplot(df['year'])




#### location Visualization

df['location'].value_counts().head(10)


df['location'].value_counts()[1:10].plot(kind='bar')

#employee.status visualization

df['employee.status'].value_counts()

import matplotlib.pyplot as plt

plt.bar(df['employee.status'].value_counts().index,df['employee.status'].v
alue_counts().values)


#calculated_rating average


df.info()


sns.barplot(x=df.groupby('company').mean()['overall.ratings'].index,y=df.g
roupby('company').mean()['overall.ratings'])


sns.barplot(df.groupby('company').mean()['overall.ratings'].sort_values().
index,
df.groupby('company').mean()['overall.ratings'].sort_values().values

,palette='CMRmap_r')




######################################## Logistic Regression prediction
model #############

df.info()

df_pros = df['pros']

df_pros['type'] = 1

df_pros

df_pros.drop('type',inplace=True)

dfp = df_pros.copy()

import pandas as pd

import numpy as np

df_pos = pd.DataFrame(dfp)

df_pos.info()

df_pos['type'] = 1

dfn = df['cons'].copy()

df_neg = pd.DataFrame(dfn)

df_neg['type'] = 0

df_pos.rename(columns={'pros':'review'},inplace=True)

df_neg.rename(columns={'cons':'review'},inplace=True)

cdf = df_pos.append(df_neg)

model_df = cdf.copy()
```

```
model_df.reset_index(drop=True,inplace=True)

model_df

mdf = model_df.sample(frac=1).reset_index(drop=True)

ndf = mdf.copy()

ndf


import string

from spacy.lang.en.stop_words import STOP_WORDS

import spacy as sp

import pandas as pd

import numpy as np

from spacy.lang.en import English


#loading the English module

nlp= sp.load('en_core_web_sm')

punctuations = string.punctuation

stop_words = STOP_WORDS

#parser = English()


#Creating tokenizer function

def spacy_tokenizer(sentence):

    #Creating the token object.

    mytokens = nlp(sentence)

    #Lemmatizing each token and converting each token into lowercase
    mytokens = [ word.lemma_.lower().strip() if word.lemma_ != "-PRON-"
else word.lower_ for word in mytokens ]

    #Removing stop words

    mytokens = [ word for word in mytokens if word not in stop_words and
word not in punctuations ]

    #Returning preprocessed list of tokens

    return mytokens


#Importing thenecessary libraries

from sklearn.feature_extraction.text import
CountVectorizer,TfidfVectorizer

from sklearn.base import TransformerMixin

from sklearn.pipeline import Pipeline


#Creating a Custom transformer using SpaCy

class predictors(TransformerMixin):

    def transform(self, X, **transform_params):

        # Cleaning Text

        return [clean_text(text) for text in X]


    def fit(self, X, y=None, **fit_params):

        return self


    def get_params(self, deep=True):

        return {}
```

```
#Creating a Basic function to clean the text
def clean_text(text):
    # Removing whitespaces and converting text into lowercase
    return text.strip().lower()


#CountVectorizer is a great tool provided by the scikit-learn library in
Python.
#It is used to transform a given text into a vector on the basis of the
frequency
#(count) of each word that occurs in the entire text.
#BOW = Bag of Words Matrix, which is what the Countvectorizer generates.
#ngrams is to parse word by word (Unigrams)
bow_vector = CountVectorizer(tokenizer = spacy_tokenizer,
ngram_range=(1,1))


#TF-IDF (Term Frequency-Inverse Document Frequency)
#it's a way of representing how important a particular term is in the
context of a
#given document, based on how many times the term appears and how many
other documents
#that same term appears in. The higher the TF-IDF, the more important that
term is.
tfidf_vector = TfidfVectorizer(tokenizer = spacy_tokenizer)


#Splitting the data into Train/Test datasets
from sklearn.model_selection import train_test_split
```

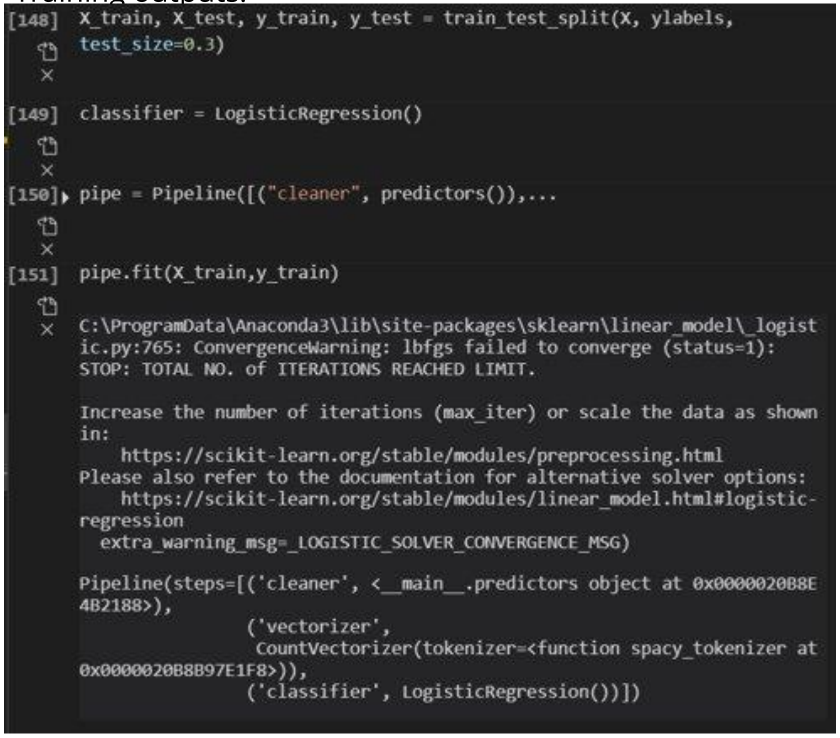
```
X = ndf['review'] #The textual reviews

ylabels = ndf['type'] # the labels we want to test against (positive 1 /
negative 0)

#splitting 30% of the data into a test dataset to test the accuracy of our
model

X_train, X_test, y_train, y_test = train_test_split(X, ylabels,
test_size=0.3)

#Using logistic regression

from sklearn.linear_model import LogisticRegression

classifier = LogisticRegression()

# Create pipeline using Bag of Words

pipe = Pipeline([("cleaner", predictors()),

                 ('vectorizer', bow_vector),

                 ('classifier', classifier)])

#Model generation and training

pipe.fit(X_train,y_train)

#Now to testing the model we generated.

from sklearn import metrics

# Predicting with a test dataset

predicted = pipe.predict(X_test)



# Model Accuracy

print("Logistic Regression Accuracy:",metrics.accuracy_score(y_test,
predicted))

print("Logistic Regression Precision:",metrics.precision_score(y_test,
predicted))
```

```
print("Logistic Regression Recall:",metrics.recall_score(y_test,
predicted))

###################

ndf['length'] = ndf['review'].apply(len)

ndf['length'].value_counts()

ndf = ndf[ndf['length'] > 1]

ndf

#########################

plt.style.use('ggplot')

plt.bar(df['employee.status'].value_counts().index,df['employee.status'].v
alue_counts().values)


sns.barplot(df.groupby('company').mean()['overall.ratings'].sort_values().
index,

df.groupby('company').mean()['overall.ratings'].sort_values().values

,palette='CMRmap_r')

plt.yticks(range(6))

plt.title('Average Rating by Company')


df.info()
```

This code is also provided as a .py file as an appendix so you could retrace it in VSCode or any other python interpreter. It does not work as a script, but it is enough to recreate the desired outputs.

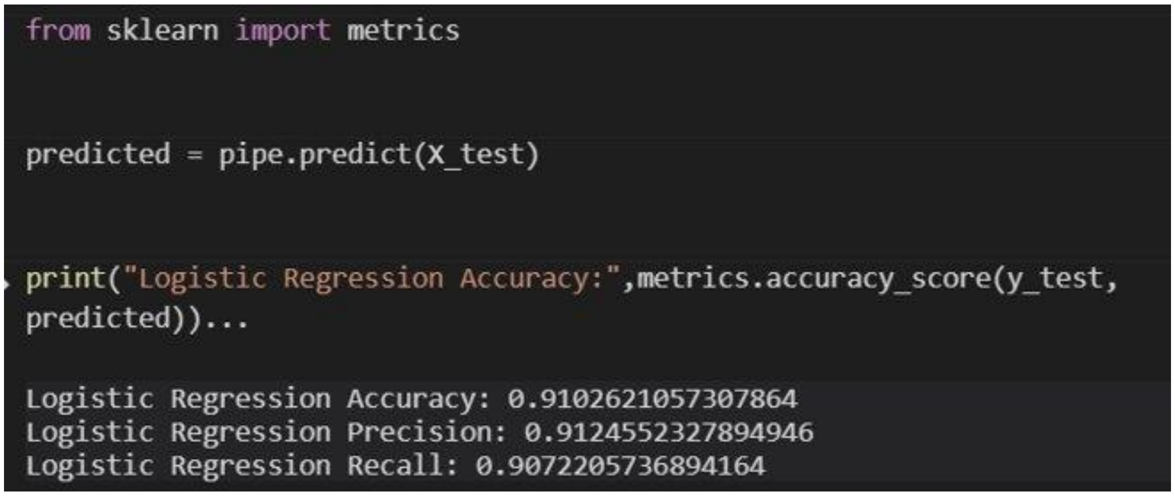
Here are the outputs for our text classification model:

- Training outputs:



```
[148]  X_train, X_test, y_train, y_test = train_test_split(X, ylabels,
       test_size=0.3)

[149]  classifier = LogisticRegression()

[150]  pipe = Pipeline([("cleaner", predictors()),...

[151]  pipe.fit(X_train,y_train)

       C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logist
       ic.py:765: ConvergenceWarning: lbfgs failed to converge (status=1):
       STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

       Increase the number of iterations (max_iter) or scale the data as shown
       in:
           https://scikit-learn.org/stable/modules/preprocessing.html
       Please also refer to the documentation for alternative solver options:
           https://scikit-learn.org/stable/modules/linear_model.html#logistic-
       regression
         extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)

       Pipeline(steps=[('cleaner', <__main__.predictors object at 0x0000020B8E
       4B2188>),
                       ('vectorizer',
                        CountVectorizer(tokenizer=<function spacy_tokenizer at
       0x0000020B8B97E1F8>)),
                       ('classifier', LogisticRegression())])
```

- Testing Prediction accuracy:



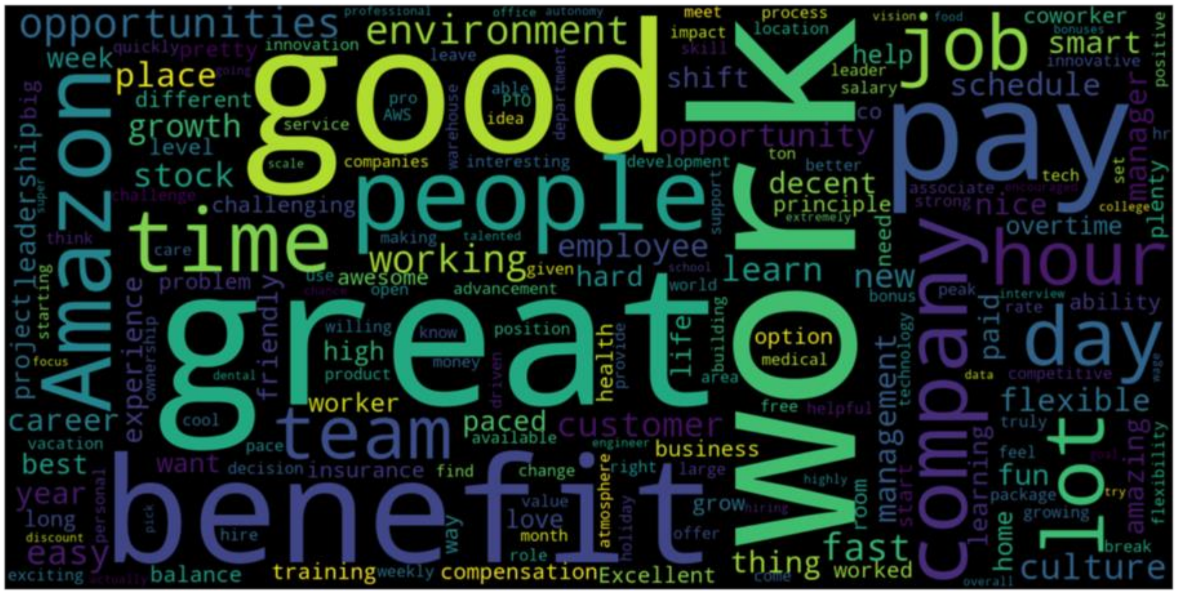
```
from sklearn import metrics

predicted = pipe.predict(X_test)

print("Logistic Regression Accuracy:",metrics.accuracy_score(y_test,
predicted))...

Logistic Regression Accuracy: 0.9102621057307864
Logistic Regression Precision: 0.9124552327894946
Logistic Regression Recall: 0.9072205736894164
```

We finally got about %91 accuracy which is a good number for predicting whether or not a review is positive.

Final Analysis and Findings with Power BI & Python

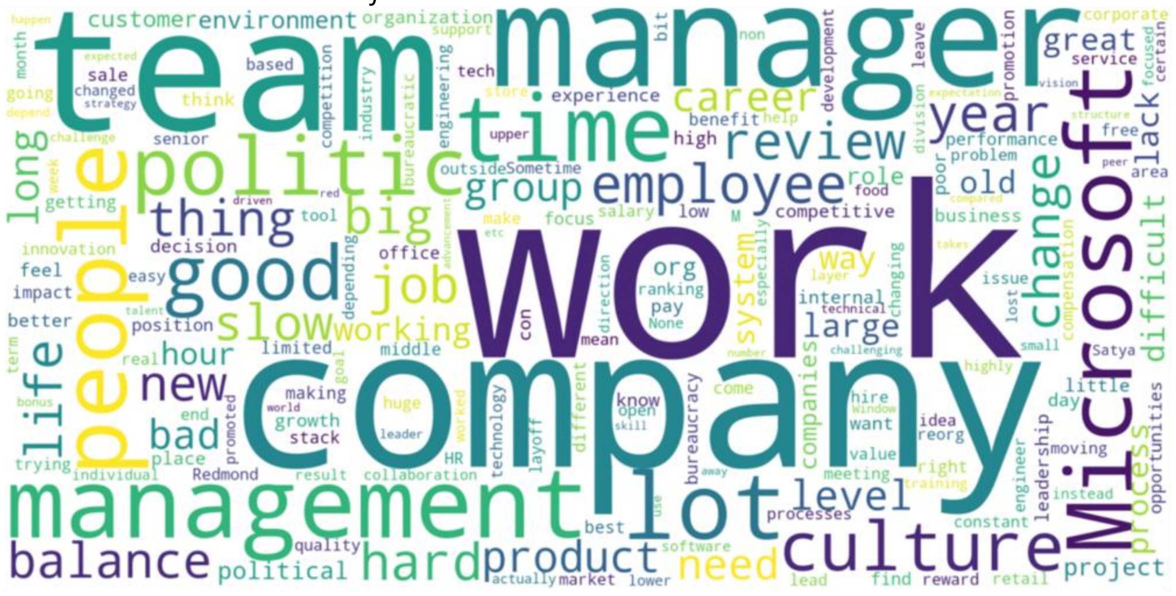
With the use of Python and Power BI we were able to generate some very useful data visualizations of the results from our text analysis through the text mining process. One set of visualizations were Word Clouds. Word Clouds depict a group of significant words that are extracted from the data. The groups of words are typically laid out in random fashion and in multiple colors, but the most significant or most used words appear larger, which from there we can associate specific words with review pros and cons from each company in the database. Below are a few examples of word clouds we generated from Python output.

Word Cloud Results: Amazon - Pros

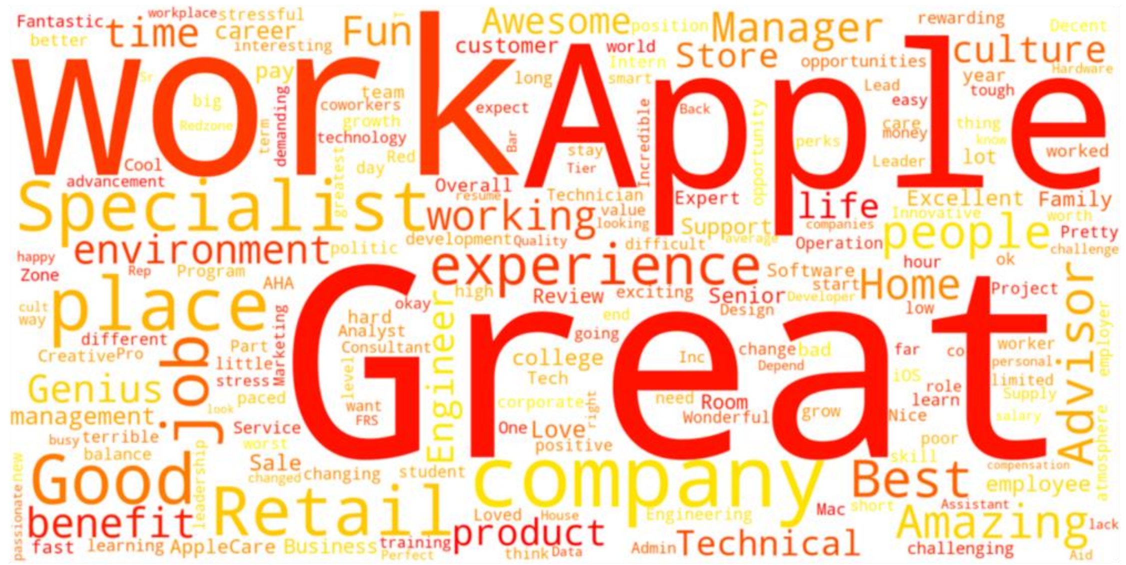


We can see, based off of the Amazon word cloud, that many of the employees used words such as ‘great’, ‘good’, ‘work’, ‘benefit’, and ‘pay’ as being significant in describing the pros of the company. These visualizations offer some interesting insights in comparison to the following word cloud which represents the results from Microsoft cons reviews and includes significant words such as ‘work’, ‘company’, ‘team’, and ‘manager’.

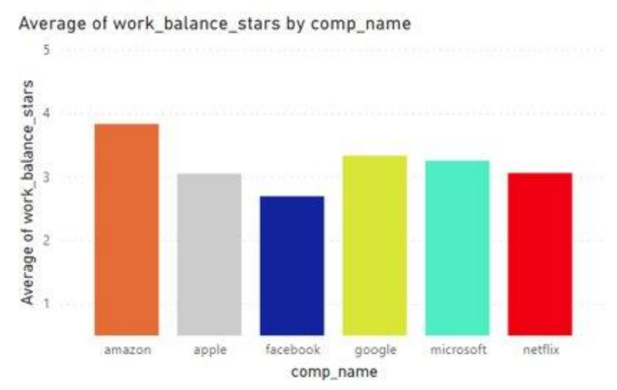
Word Cloud Results: Microsoft - Cons

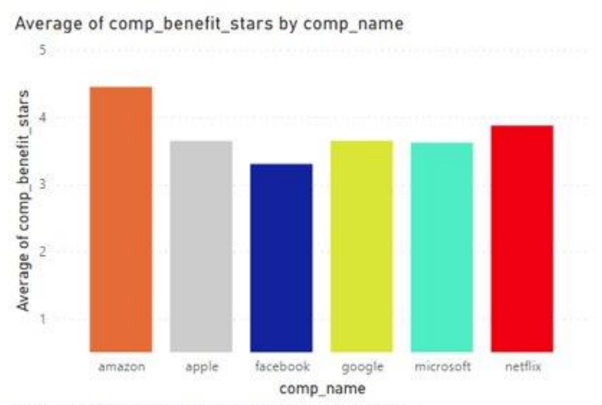


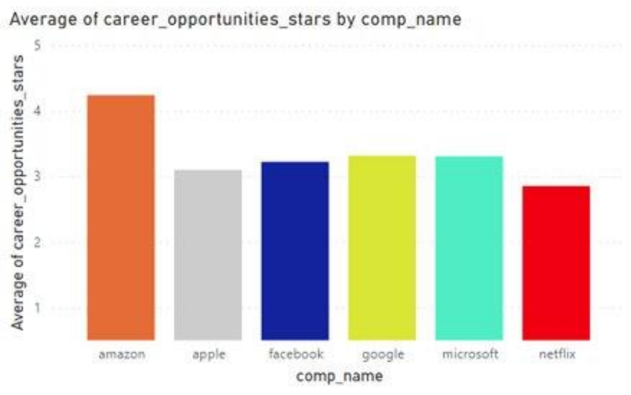
We were also able to generate word clouds based on the summary of reviews overall for each company. The following shows the results from Apple summary reviews. This word cloud could provide insight to how employees feel about working at a company such as Apple.

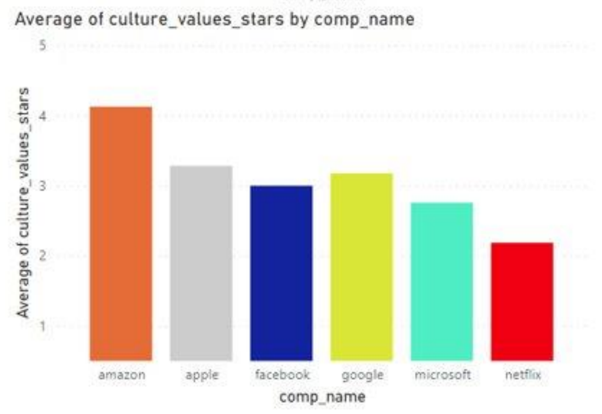


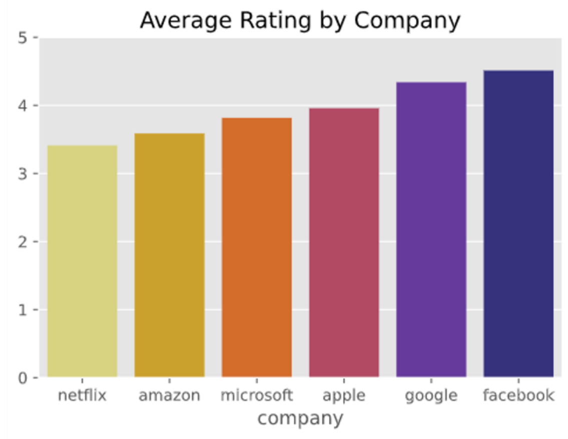
Using Power BI, we were also able to generate a handful of other useful graphs, from review ratings in specific categories such as work balance, career opportunities, company benefits, and culture and values. The following visualizations show the data averages of ratings in each category respectively for each company.

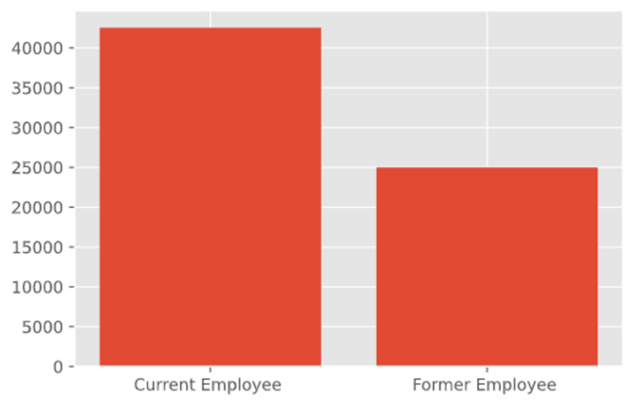


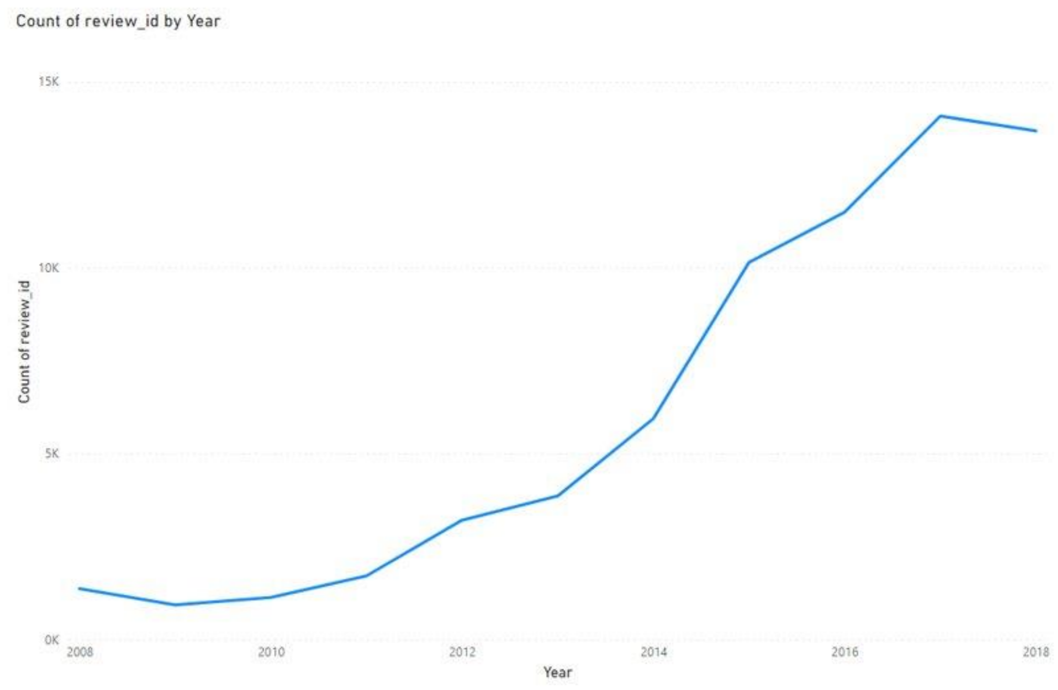


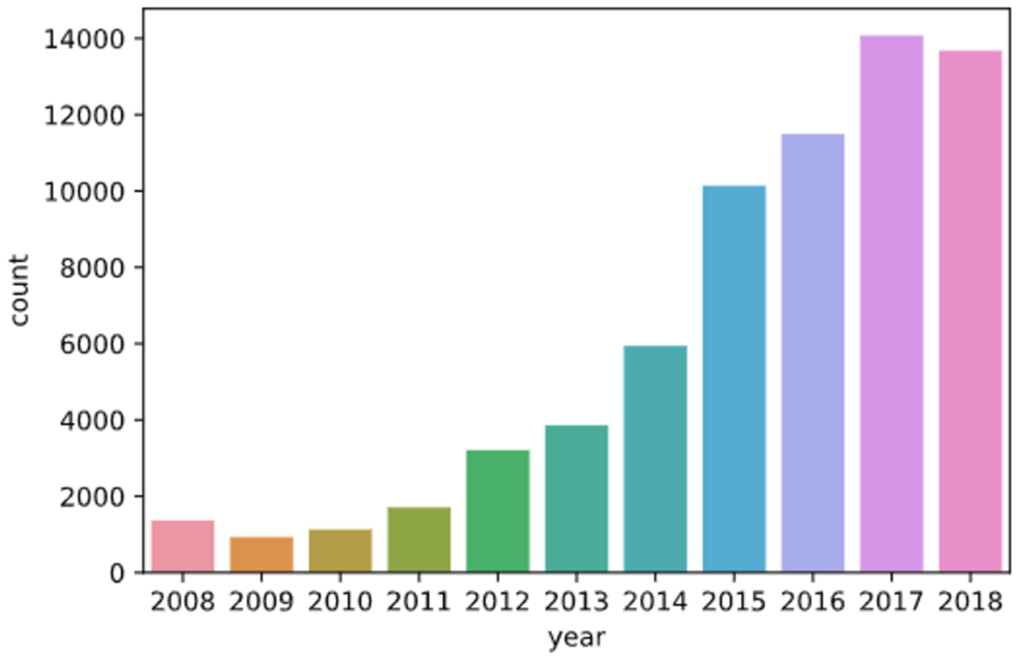


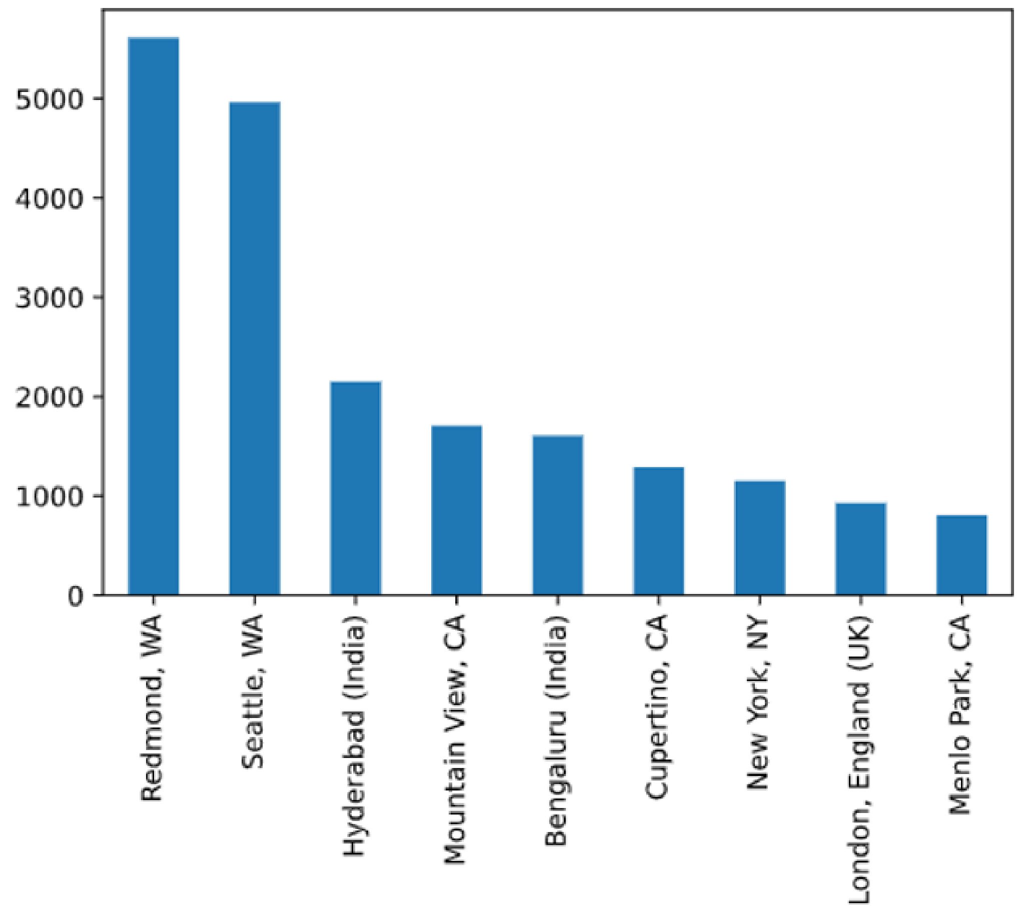







Count of review_id by Year
15K
10K
5K
0K
Count of review_id
2008
2010
2012
2014
2016
2018
Year


14000
12000
10000
8000
6000
4000
2000
0
count
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
year



Conclusion

In conclusion we had a number of successful outcomes from the result of our project. We were successfully able to build a database capable of inserting additional employee reviews for future use, while maintaining historical data of employees, companies and reviews. We were able to analyze the textual data within our current dataset and discover some key insights that were unique to each company in the FAANG Database. We successfully created a pipeline of analysis between SQL Server, Python and Power BI to generate a wide range of visualization outputs to reflect our results. We uncovered the major geographical sources of employee reviews through the innovative tools of Power BI. And lastly we were able to build and train a text classification model to predict whether a review is positive or negative. This prediction proved to be roughly %90 accurate.

Appendices