DATABASE MANAGEMENT AND DATABASE DESIGN

(INFO 6210)

PROJECT PORTFOLIO

(JOBS DATABASE)

SUBMITTED BY:

SAJIDA MOHAMMAD

(1491878)

PAVAN KUMAR REDDY

(1356451)

CONTENTS:

JOBS DATABASE

ABSTRACT:

Looking for jobs and internships seems a task of its own and the search is no longer on sole fulfilment of the required skills, but a lot of networking and recommendations is involved around too. The amount of work involved in finding the correct job builds a great amount of anxiety among the job seekers and the recruiters who want the right talent for their company. First, matching the job seekers with the right employers and second, provide guidance to aspiring job seekers on the skills that are in demand so that they can build them to stay relevant in the job market. The job providers and job seekers form a large amount of data which provides for many interesting trends for analysis and interpretation to make the most of data available. The presence of information on job skills, salaries and user tendencies in many existing websites such as Glassdoor, LinkedIn and many other websites can be utilized to match people to the suitable positions. The jobs database would be a one stop solution to reduce the job search and talent acquisition stress levels.

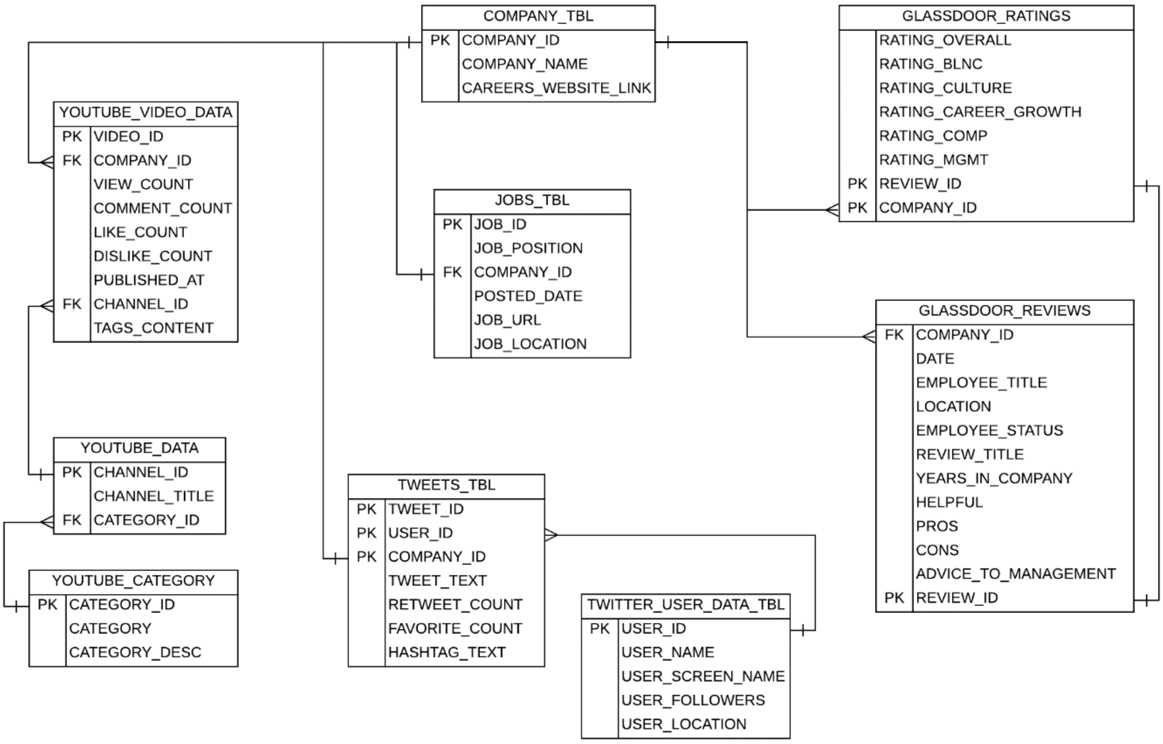
INTRODUCTION:

Job searching is the most important phase in anyone’s life. A right job can make someone’s life to the best than he could ever had imagined. In this present internet era, there are many websites that are posting the numerous jobs and the specific skill sets required for those jobs.

DATA:

The first step in any database project is the getting the necessary data. The data which we want can be collected from various sources. Now there are many online platforms which can provide you the necessary data for the creation of the project. First, we collected the list of 300 companies in finance domain and then we collected the data from those company websites using the web scraping. We also scraped the Glassdoor website and collected company reviews and ratings for about 28 companies from the list. We also scraped Twitter data and YouTube data using their API’s collected the relevant tweets and video data.

ER DIAGRAM:



DESCRIPTION ABOUT TABLES:

There are about nine tables in the JOBSDB_DMDD database.

- **COMPANY_TBL**
- **JOBS_TBL**
- **TWEETS_TBL**
- **TWITTER_USER_DATA_TBL**
- **YOUTUBE_DATA**
- **YOUTUBE_CATEGORY**
- **YOUTUBE_VIDEO_DATA**
- **GLASSDOOR_RATINGS**
- **GLASSDOOR_REVIEWS**

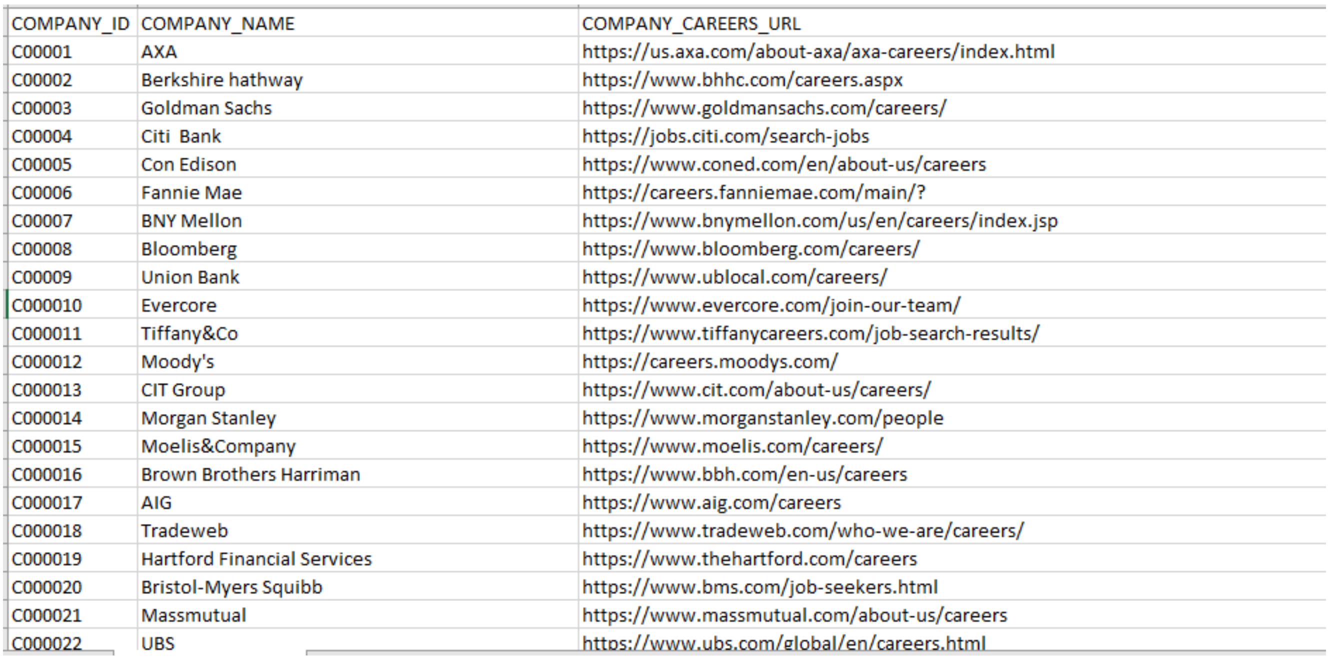
1) COMPANY_TBL (company table):

This table contains the following attributes:

COMPANY_ID(PK): As we collected the data of the 300 finance company's we created a unique id for each of the company. This column is the primary key of the table and each row can be uniquely identified using this primary key.

COMPANY_NAME: This attribute or the column contains all the company names which we collected for the finance department.

CAREERS_WEBSITE_LINK: Each company has its own website and it has its careers page through which one can apply for the job. We collected the career page links for each of the 300 companies.



COMPANY_ID	COMPANY_NAME	COMPANY_CAREERS_URL
C00001	AXA	https://us.axa.com/about-axa/axa-careers/index.html
C00002	Berkshire hathway	https://www.bhhc.com/careers.aspx
C00003	Goldman Sachs	https://www.goldmansachs.com/careers/
C00004	Citi Bank	https://jobs.citi.com/search-jobs
C00005	Con Edison	https://www.coned.com/en/about-us/careers
C00006	Fannie Mae	https://careers.fanniemae.com/main/?
C00007	BNY Mellon	https://www.bnymellon.com/us/en/careers/index.jsp
C00008	Bloomberg	https://www.bloomberg.com/careers/
C00009	Union Bank	https://www.ublocal.com/careers/
C000010	Evercore	https://www.evercore.com/join-our-team/
C000011	Tiffany&Co	https://www.tiffanycareers.com/job-search-results/
C000012	Moody's	https://careers.moodys.com/
C000013	CIT Group	https://www.cit.com/about-us/careers/
C000014	Morgan Stanley	https://www.morganstanley.com/people
C000015	Moelis&Company	https://www.moelis.com/careers/
C000016	Brown Brothers Harriman	https://www.bbh.com/en-us/careers
C000017	AIG	https://www.aig.com/careers
C000018	Tradeweb	https://www.tradeweb.com/who-we-are/careers/
C000019	Hartford Financial Services	https://www.thehartford.com/careers
C000020	Bristol-Myers Squibb	https://www.bms.com/job-seekers.html
C000021	Massmutual	https://www.massmutual.com/about-us/careers
C000022	UBS	https://www.ubs.com/global/en/careers.html

JOBS_TBL:

This table has the following attributes:

JOB_ID:

When we check for any company website for the jobs we can see that there will be a job id uniquely defined for each and every different type of the job they post. This JOB_ID refers to the same thing.

JOB_POSITION:

Job position refers to the what type of position he is applying for. For example, he may apply to software Engineering, Data scientist etc.

COMPANY_ID:

This is the foreign key in this table which refers to the primary key in the COMPANY_TBL.

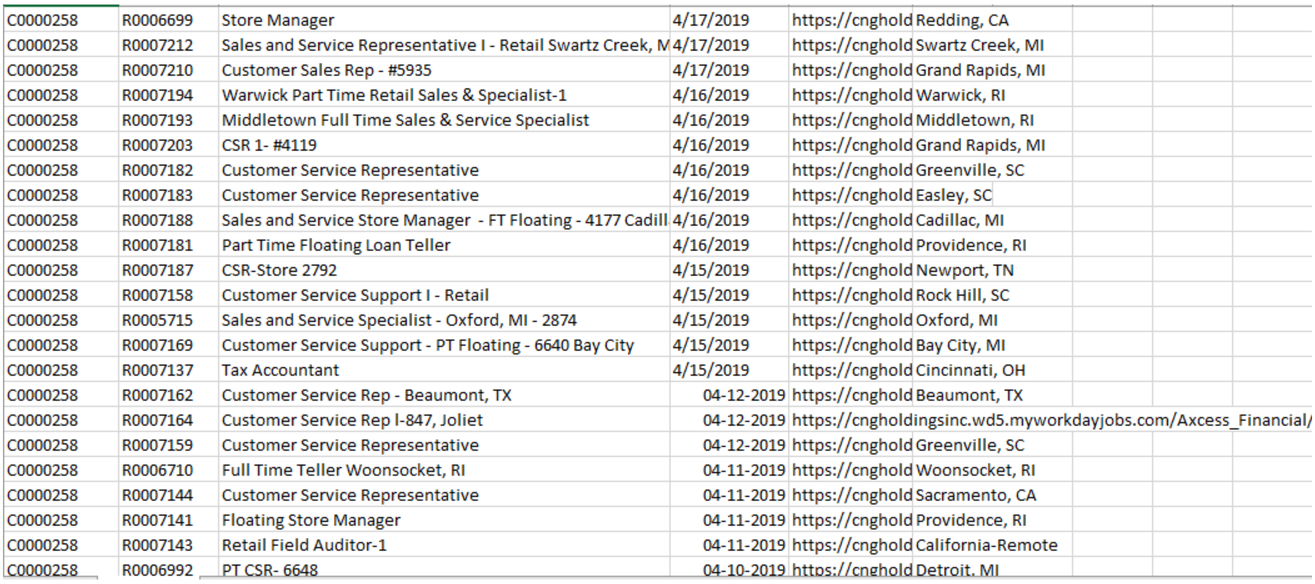
Foreign keys are used to provide the perfect link between the tables.

POSTED_DATE:

Posted date refers to on which date the job position is posted on the website.

JOB_URL: Job URL refers to the link to apply for that position. We go to any careers website we will see that click here to apply and after clicking that we will be redirected to the new page. This job_url stores the link of the page.

JOB_LOCATION: Job location refers to on which location this job is available.



C0000258	R0006699	Store Manager	4/17/2019	https://cnghold	Redding, CA
C0000258	R0007212	Sales and Service Representative I - Retail Swartz Creek, M	4/17/2019	https://cnghold	Swartz Creek, MI
C0000258	R0007210	Customer Sales Rep - #5935	4/17/2019	https://cnghold	Grand Rapids, MI
C0000258	R0007194	Warwick Part Time Retail Sales & Specialist-1	4/16/2019	https://cnghold	Warwick, RI
C0000258	R0007193	Middletown Full Time Sales & Service Specialist	4/16/2019	https://cnghold	Middletown, RI
C0000258	R0007203	CSR 1- #4119	4/16/2019	https://cnghold	Grand Rapids, MI
C0000258	R0007182	Customer Service Representative	4/16/2019	https://cnghold	Greenville, SC
C0000258	R0007183	Customer Service Representative	4/16/2019	https://cnghold	Easley, SC
C0000258	R0007188	Sales and Service Store Manager - FT Floating - 4177 Cadill	4/16/2019	https://cnghold	Cadillac, MI
C0000258	R0007181	Part Time Floating Loan Teller	4/16/2019	https://cnghold	Providence, RI
C0000258	R0007187	CSR-Store 2792	4/15/2019	https://cnghold	Newport, TN
C0000258	R0007158	Customer Service Support I - Retail	4/15/2019	https://cnghold	Rock Hill, SC
C0000258	R0005715	Sales and Service Specialist - Oxford, MI - 2874	4/15/2019	https://cnghold	Oxford, MI
C0000258	R0007169	Customer Service Support - PT Floating - 6640 Bay City	4/15/2019	https://cnghold	Bay City, MI
C0000258	R0007137	Tax Accountant	4/15/2019	https://cnghold	Cincinnati, OH
C0000258	R0007162	Customer Service Rep - Beaumont, TX	04-12-2019	https://cnghold	Beaumont, TX
C0000258	R0007164	Customer Service Rep I-847, Joliet	04-12-2019	https://cngholdingsinc.wd5.myworkdayjobs.com/Axcess_Financial/	
C0000258	R0007159	Customer Service Representative	04-12-2019	https://cnghold	Greenville, SC
C0000258	R0006710	Full Time Teller Woonsocket, RI	04-11-2019	https://cnghold	Woonsocket, RI
C0000258	R0007144	Customer Service Representative	04-11-2019	https://cnghold	Sacramento, CA
C0000258	R0007141	Floating Store Manager	04-11-2019	https://cnghold	Providence, RI
C0000258	R0007143	Retail Field Auditor-1	04-11-2019	https://cnghold	California-Remote
C0000258	R0006992	PT CSR- 6648	04-10-2019	https://cnghold	Detroit, MI

TWEETS_TBL:

This table has the following attributes:

The primary key in this table is the composite key i.e which is the combination of TWEET_ID, USER_ID, COMPANY_ID. The combination of this three can identify each row of uniquely without any ambiguity.

TWEET_ID: When the user posts any tweet in twitter it will generate a unique id for each and every tweet. This unique id is the TWEET_ID.

USER_ID: Each person who has a twitter account will be given a unique id. This id is the USER_ID.

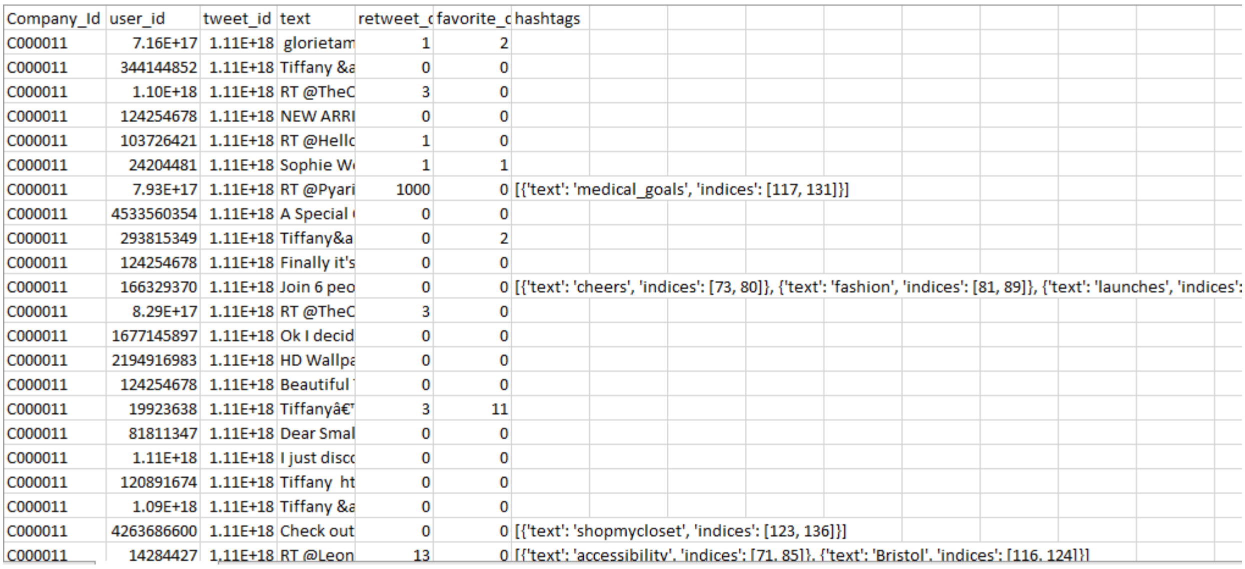
COMPANY_ID: This is the unique id which is given to company.

TWEET_TEXT: Tweet text is the text posted by the user in twitter.

RETWEET_COUNT: Retweet count is the number of people retweeted or replied to any person.

FAVORITE_COUNT: Favorite count is the number of people who liked the particular posts.

HASHTAG_TEXT: Hashtag text is the combination of the hashtags the user posted.



Company_Id	user_id	tweet_id	text	retweet_c	favorite_c	hashtags
C000011	7.16E+17	1.11E+18	glorietam	1	2	
C000011	344144852	1.11E+18	Tiffany &a	0	0	
C000011	1.10E+18	1.11E+18	RT @TheC	3	0	
C000011	124254678	1.11E+18	NEW ARRI	0	0	
C000011	103726421	1.11E+18	RT @Hellc	1	0	
C000011	24204481	1.11E+18	Sophie W	1	1	
C000011	7.93E+17	1.11E+18	RT @Pyari	1000	0	[{'text': 'medical_goals', 'indices': [117, 131]}]
C000011	4533560354	1.11E+18	A Special	0	0	
C000011	293815349	1.11E+18	Tiffany&a	0	2	
C000011	124254678	1.11E+18	Finally it's	0	0	
C000011	166329370	1.11E+18	Join 6 peo	0	0	[{'text': 'cheers', 'indices': [73, 80]}, {'text': 'fashion', 'indices': [81, 89]}, {'text': 'launches', 'indices':
C000011	8.29E+17	1.11E+18	RT @TheC	3	0	
C000011	1677145897	1.11E+18	Ok I decid	0	0	
C000011	2194916983	1.11E+18	HD Wallpa	0	0	
C000011	124254678	1.11E+18	Beautiful	0	0	
C000011	19923638	1.11E+18	Tiffanyâ€	3	11	
C000011	81811347	1.11E+18	Dear Smal	0	0	
C000011	1.11E+18	1.11E+18	I just disc	0	0	
C000011	120891674	1.11E+18	Tiffany ht	0	0	
C000011	1.09E+18	1.11E+18	Tiffany &a	0	0	
C000011	4263686600	1.11E+18	Check out	0	0	[{'text': 'shopmycloset', 'indices': [123, 136]}]
C000011	14284427	1.11E+18	RT @Leon	13	0	[{'text': 'accessibility', 'indices': [71, 85]}, {'text': 'Bristol', 'indices': [116, 124]}]

TWITTER_USER_DATA_TBL:

This table has the following attributes and the primary key in the table is USER_ID which is used to uniquely define each row.

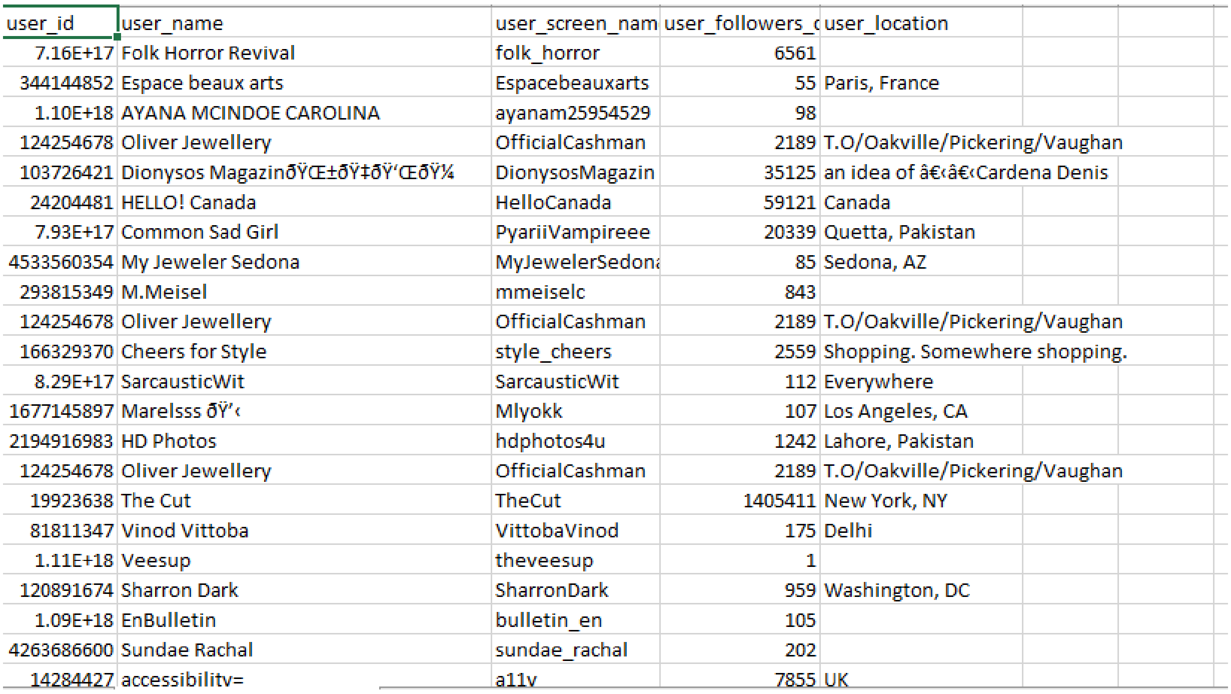
USER_ID: Each person who had a twitter account will be given a unique id. This id is the USER_ID.

USER_NAME: User name is the name of the user who posted the particular tweet.

USER_SCREEN_NAME: User screen name is the screen name of the user in the twitter.

USER_FOLLOWERS: Number of followers of a particular user.

USER_LOCATION: Location of the user.



user_id	user_name	user_screen_nam	user_followers_c	user_location
7.16E+17	Folk Horror Revival	folk_horror	6561	
344144852	Espace beaux arts	Espacebeauxarts	55	Paris, France
1.10E+18	AYANA MCINDOE CAROLINA	ayanam25954529	98	
124254678	Oliver Jewellery	OfficialCashman	2189	T.O/Oakville/Pickering/Vaughan
103726421	Dionysos MagazinðŸŒ±ðŸ‡ðŸ'ŒðŸ¼	DionysosMagazin	35125	an idea of â€‹â€‹Cardena Denis
24204481	HELLO! Canada	HelloCanada	59121	Canada
7.93E+17	Common Sad Girl	PyariiVampireee	20339	Quetta, Pakistan
4533560354	My Jeweler Sedona	MyJewelerSedona	85	Sedona, AZ
293815349	M.Meisel	mmeiselc	843	
124254678	Oliver Jewellery	OfficialCashman	2189	T.O/Oakville/Pickering/Vaughan
166329370	Cheers for Style	style_cheers	2559	Shopping. Somewhere shopping.
8.29E+17	SarcausticWit	SarcausticWit	112	Everywhere
1677145897	Marelsss ðŸ'‹	Mlyokk	107	Los Angeles, CA
2194916983	HD Photos	hdphotos4u	1242	Lahore, Pakistan
124254678	Oliver Jewellery	OfficialCashman	2189	T.O/Oakville/Pickering/Vaughan
19923638	The Cut	TheCut	1405411	New York, NY
81811347	Vinod Vittoba	VittobaVinod	175	Delhi
1.11E+18	Veesup	theveesup	1	
120891674	Sharron Dark	SharronDark	959	Washington, DC
1.09E+18	EnBulletin	bulletin_en	105	
4263686600	Sundae Rachal	sundae_rachal	202	
14284427	accessibility=	a11y	7855	UK

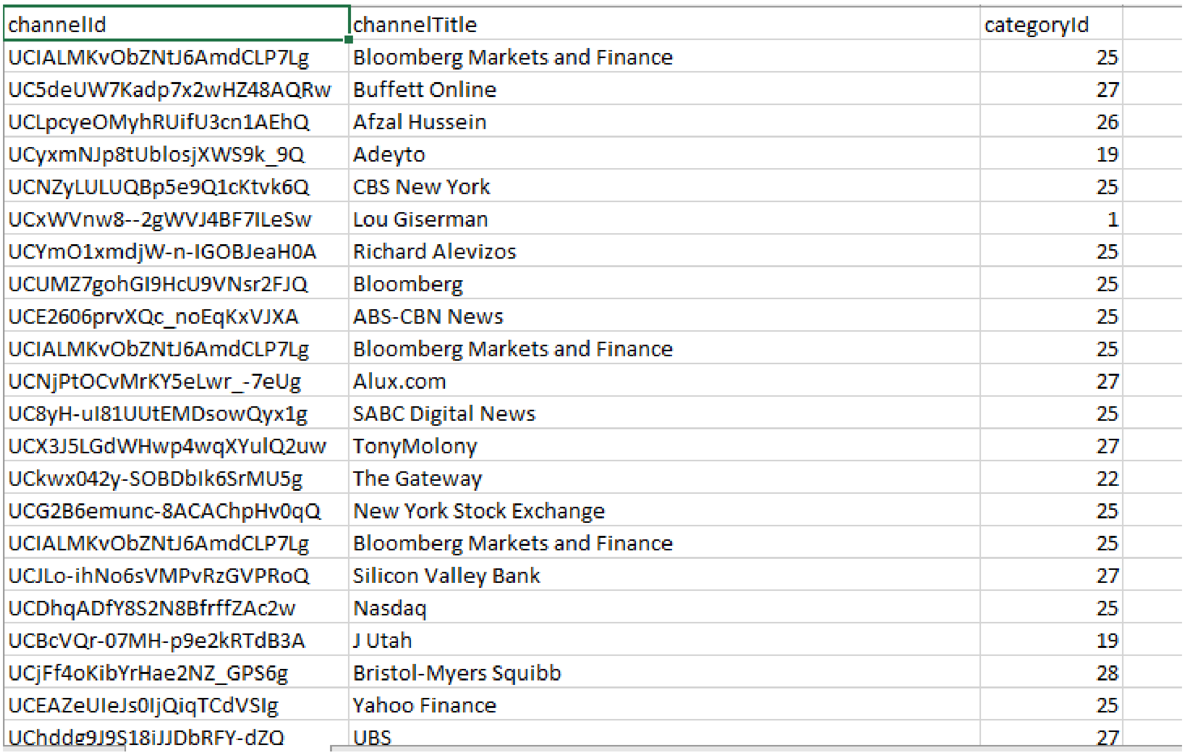
YOUTUBE_DATA:

This table has the following attributes and the primary key in this table is channel_id and the foreign key in this table is category_id which references youtube_category table.

CHANNEL_ID: Each channel in the YouTube is given the unique id. This unique id is called the CHANNEL_ID.

CHANNEL_TITLE: Channel_title is the title of the YouTube channel

CATEGORY_ID: This is the unique id given to each and every category in the category table mainly to eliminate the data redundancy.



channelId	channelTitle	categoryId
UCIALMKvObZNtJ6AmdCLP7Lg	Bloomberg Markets and Finance	25
UC5deUW7Kadp7x2wHZ48AQRw	Buffett Online	27
UCLpcyeOMyhRUifU3cn1AEhQ	Afzal Hussein	26
UCyxmNJp8tUblosjXWS9k_9Q	Adeyto	19
UCNZyLULUQBp5e9Q1cKtvk6Q	CBS New York	25
UCxWVnw8--2gWVJ4BF7ILeSw	Lou Giserman	1
UCYmO1xmdjW-n-IGOBJeaH0A	Richard Alevizos	25
UCUMZ7gohGI9HcU9VNsr2FJQ	Bloomberg	25
UCE2606prvXQc_noEqKxVJXA	ABS-CBN News	25
UCIALMKvObZNtJ6AmdCLP7Lg	Bloomberg Markets and Finance	25
UCNjPtOCvMrKY5eLwr_-7eUg	Alux.com	27
UC8yH-uI81UUtEMDsowQyx1g	SABC Digital News	25
UCX3J5LGdWHwp4wqXYulQ2uw	TonyMolony	27
UCkwx042y-SOBDbIk6SrMU5g	The Gateway	22
UCG2B6emunc-8ACAChpHv0qQ	New York Stock Exchange	25
UCIALMKvObZNtJ6AmdCLP7Lg	Bloomberg Markets and Finance	25
UCJLo-ihNo6sVMPvRzGVPRoQ	Silicon Valley Bank	27
UCDhqADfY8S2N8BfrffZAc2w	Nasdaq	25
UCBcVQr-07MH-p9e2kRTdB3A	J Utah	19
UCjFf4oKibYrHae2NZ_GPS6g	Bristol-Myers Squibb	28
UCEAZeUleJs0IjQiqTCdVSIg	Yahoo Finance	25
UChddg9J9S18iJJDbRFY-dZQ	UBS	27

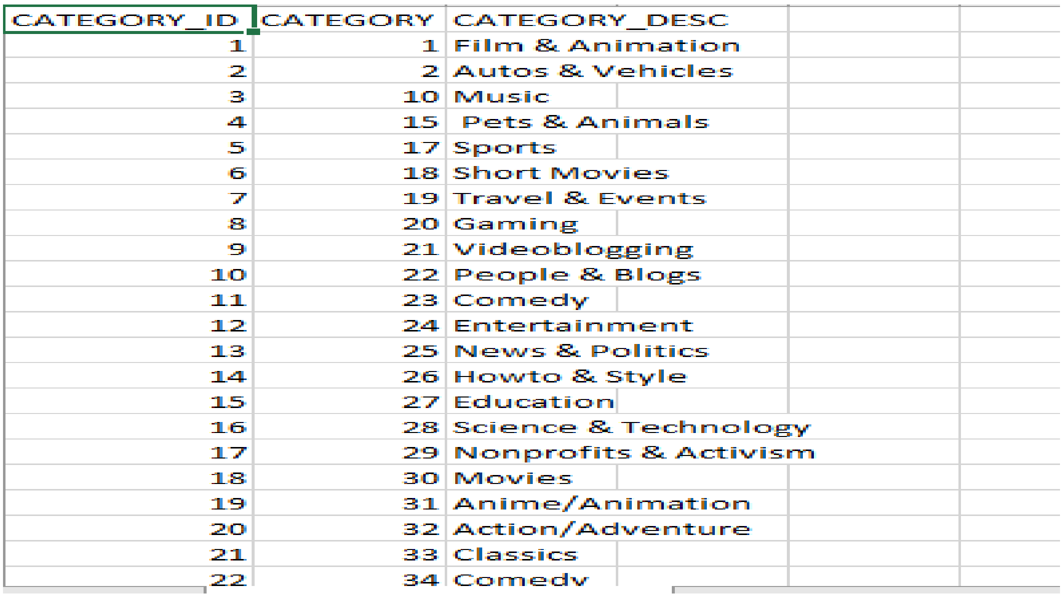
YOUTUBE_CATEGORY:

This table has the following attributes and the primary key in this table is the category_id.

CATEGORY_ID: This is the unique id given to each category in the category table.

CATEGORY: Category gives the information of the category of the video.

CATEGORY_DESC: Description of that specific category. For example, category 24 in YouTube describes the entertainment.



CATEGORY_ID	CATEGORY	CATEGORY_DESC
1	1	Film & Animation
2	2	Autos & Vehicles
3	10	Music
4	15	Pets & Animals
5	17	Sports
6	18	Short Movies
7	19	Travel & Events
8	20	Gaming
9	21	Videoblogging
10	22	People & Blogs
11	23	Comedy
12	24	Entertainment
13	25	News & Politics
14	26	Howto & Style
15	27	Education
16	28	Science & Technology
17	29	Nonprofits & Activism
18	30	Movies
19	31	Anime/Animation
20	32	Action/Adventure
21	33	Classics
22	34	Comedy

YOUTUBE_VIDEO_DATA:

This table has the following attributes and the primary key in this table is the VIDEO_ID and the foreign key in this table are COMPANY_ID which references COMPANY_TBL and the CHANNEL_ID references the YOUTUBE_DATA table.

VIDEO_ID: VIDEO_ID represents the unique id given to each video posted in the YouTube.

COMPANY_ID: This is the foreign key which references the COMPANY_ID in the COMPANY_TBL.

VIEW_COUNT: VIEW_COUNT represents the number of views for that table.

COMMENT_COUNT: COMMENT_COUNT represents the number of counts for that video.

LIKE_COUNT: LIKE_COUNT represents the number of likes for that video.

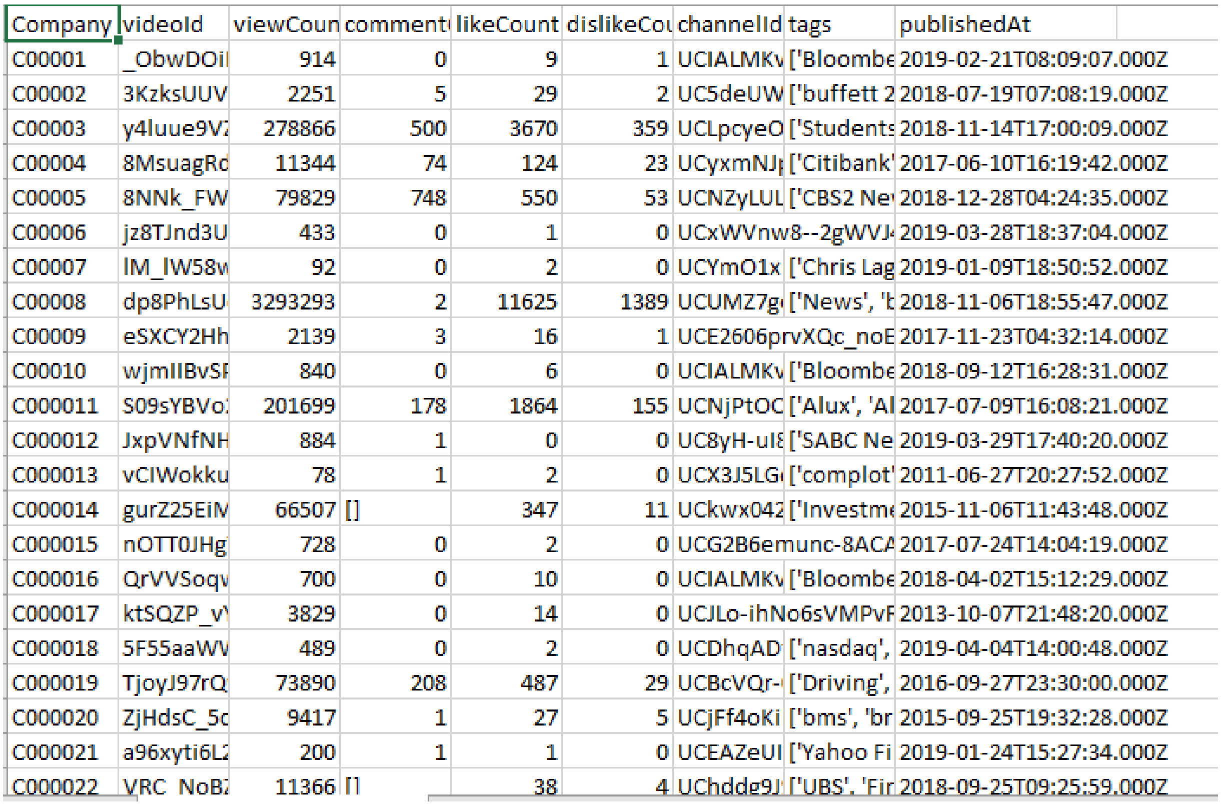
DISLIKE_COUNT: DISLIKE_COUNT represents the number of dislikes for that video.

FAVORITE_COUNT: FAVORITE_COUNT represents the number of favorites for that video.

CHANNEL_ID: This is the unique id given to each category in the category table. This is the foreign key in the table.

TAGS_CONTENT: Contents of the tag in the video.

PUBLISHED_AT: The time at which the video is posted or published.



Company	videoId	viewCoun	comment	likeCount	dislikeCo	channelId	tags	publishedAt
C00001	_ObwDOil	914	0	9	1	UCIALMKv	['Bloombe	2019-02-21T08:09:07.000Z
C00002	3KzksUUV	2251	5	29	2	UC5deUW	['buffett 2	2018-07-19T07:08:19.000Z
C00003	y4luue9V	278866	500	3670	359	UCLpcyeO	['Students	2018-11-14T17:00:09.000Z
C00004	8MsuagRd	11344	74	124	23	UCyxmNJ	['Citibank'	2017-06-10T16:19:42.000Z
C00005	8NNk_FW	79829	748	550	53	UCNZyLUL	['CBS2 Ne	2018-12-28T04:24:35.000Z
C00006	jz8TJnd3U	433	0	1	0	UCxWVnw8--2gWVJ		2019-03-28T18:37:04.000Z
C00007	lM_lW58w	92	0	2	0	UCYmO1x	['Chris Lag	2019-01-09T18:50:52.000Z
C00008	dp8PhLsU	3293293	2	11625	1389	UCUMZ7g	['News', 'b	2018-11-06T18:55:47.000Z
C00009	eSXCY2Hh	2139	3	16	1	UCE2606prvXQc_noE		2017-11-23T04:32:14.000Z
C00010	wjmIIBvSF	840	0	6	0	UCIALMKv	['Bloombe	2018-09-12T16:28:31.000Z
C000011	S09sYBVo	201699	178	1864	155	UCNjPtOC	['Alux', 'Al	2017-07-09T16:08:21.000Z
C000012	JxpVNfNH	884	1	0	0	UC8yH-ul8	['SABC Ne	2019-03-29T17:40:20.000Z
C000013	vCIWokku	78	1	2	0	UCX3J5LG	['complot'	2011-06-27T20:27:52.000Z
C000014	gurZ25EiM	66507	[]		347	11	UCkwx042 ['Investme	2015-11-06T11:43:48.000Z
C000015	nOTT0JHg	728	0	2	0	UCG2B6emunc-8ACA		2017-07-24T14:04:19.000Z
C000016	QrVVSoqv	700	0	10	0	UCIALMKv	['Bloombe	2018-04-02T15:12:29.000Z
C000017	ktSQZP_v	3829	0	14	0	UCJLo-ihNo6sVMPvF		2013-10-07T21:48:20.000Z
C000018	5F55aaWV	489	0	2	0	UCDhqAD	['nasdaq',	2019-04-04T14:00:48.000Z
C000019	TjoyJ97rQ	73890	208	487	29	UCBcVQr-	['Driving',	2016-09-27T23:30:00.000Z
C000020	ZjHdsC_5c	9417	1	27	5	UCjFf4oKi	['bms', 'br	2015-09-25T19:32:28.000Z
C000021	a96xyti6L	200	1	1	0	UCEAZeUI	['Yahoo Fi	2019-01-24T15:27:34.000Z
C000022	VRC_NoB	11366	[]	38	4	UChdde9J	['UBS', 'Fir	2018-09-25T09:25:59.000Z

GLASS_DOOR_RATINGS:

This table has the following attributes and the foreign keys in this table

COMPANY_ID: This is the foreign key which references the COMPANY_ID in the COMPANY_TBL.

RATING_OVERALL: It represents the overall Glassdoor rating of that particular company.

RATING_BLNC: It represents the rating of the work life balance for that particular company.

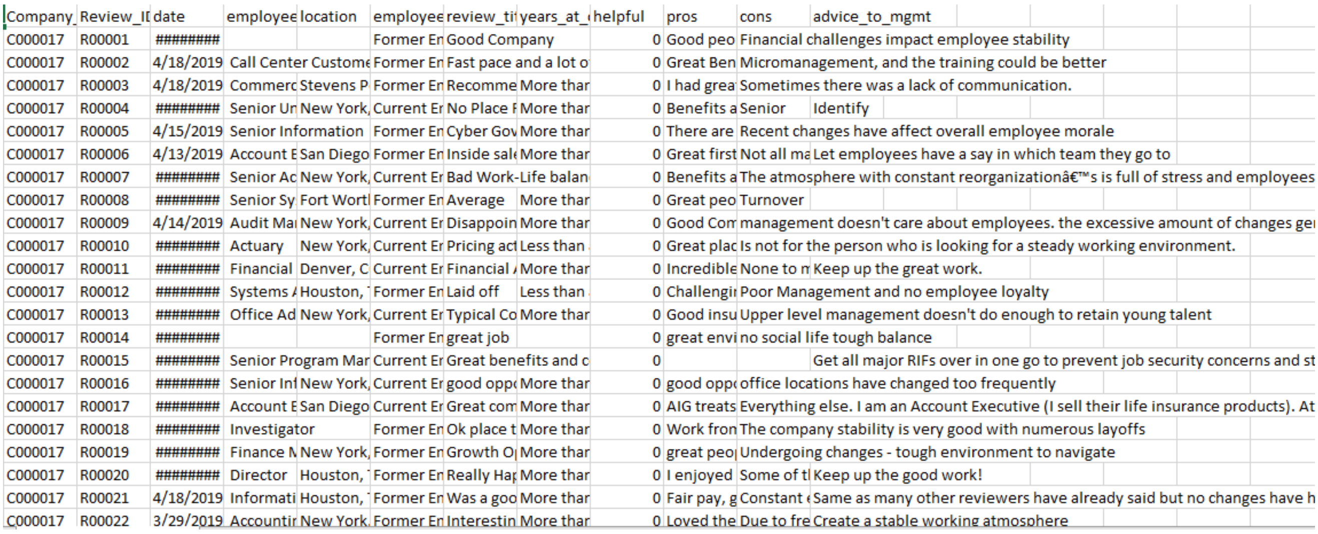
RATING_CULTURE: It represents the culture rating of that company.

RATING_CAREER_GROWTH: It represents the rating of the career growth i.e how frequently the employer is promoted to the higher positions.

RATING_COMP: It represents the rating for the compensation i.e how well the company pays for the service you have done.

RATING_MGMT: It represents the rating for the management of the company.

REVIEW_ID: It represents the unique id given to the each and every review.



Company_	Review_II	date	employee	location	employee	review_ti	years_at_	helpful	pros	cons	advice_to_mgmt
C000017	R00001	#########			Former En	Good Company		0	Good peo	Financial challenges impact employee stability	
C000017	R00002	4/18/2019	Call Center Custome		Former En	Fast pace and a lot o		0	Great Ben	Micromanagement, and the training could be better	
C000017	R00003	4/18/2019	Commerc	Stevens P	Former En	Recomme	More thar	0	I had grea	Sometimes there was a lack of communication.	
C000017	R00004	#########	Senior Ur	New York,	Current Er	No Place F	More thar	0	Benefits a	Senior	Identify
C000017	R00005	4/15/2019	Senior Information		Former En	Cyber Gov	More thar	0	There are	Recent changes have affect overall employee morale	
C000017	R00006	4/13/2019	Account E	San Diego	Former En	Inside sale	More thar	0	Great first	Not all ma	Let employees have a say in which team they go to
C000017	R00007	#########	Senior Ac	New York,	Current Er	Bad Work-Life balan		0	Benefits a	The atmosphere with constant reorganizationâ€™s is full of stress and employees	
C000017	R00008	#########	Senior Sy	Fort Wortl	Former En	Average	More thar	0	Great peo	Turnover	
C000017	R00009	4/14/2019	Audit Mai	New York,	Current Er	Disappoin	More thar	0	Good Com	management doesn't care about employees. the excessive amount of changes gei	
C000017	R00010	#########	Actuary	New York,	Current Er	Pricing act	Less than	0	Great plac	Is not for the person who is looking for a steady working environment.	
C000017	R00011	#########	Financial	Denver, C	Current Er	Financial /	More thar	0	Incredible	None to m	Keep up the great work.
C000017	R00012	#########	Systems /	Houston, T	Former En	Laid off	Less than	0	Challengi	Poor Management and no employee loyalty	
C000017	R00013	#########	Office Ad	New York,	Current Er	Typical Co	More thar	0	Good insu	Upper level management doesn't do enough to retain young talent	
C000017	R00014	#########			Former En	great job		0	great envi	no social life tough balance	
C000017	R00015	#########	Senior Program Mar		Current Er	Great benefits and c		0			Get all major RIFs over in one go to prevent job security concerns and st
C000017	R00016	#########	Senior Inf	New York,	Current Er	good oppo	More thar	0	good oppo	office locations have changed too frequently	
C000017	R00017	#########	Account E	San Diego	Current Er	Great com	More thar	0	AIG treats	Everything else. I am an Account Executive (I sell their life insurance products). At	
C000017	R00018	#########	Investigator		Former En	Ok place t	More thar	0	Work fron	The company stability is very good with numerous layoffs	
C000017	R00019	#########	Finance M	New York,	Former En	Growth O	More thar	0	great peo	Undergoing changes - tough environment to navigate	
C000017	R00020	#########	Director	Houston, T	Former En	Really Hap	More thar	0	I enjoyed	Some of tl	Keep up the good work!
C000017	R00021	4/18/2019	Informati	Houston, T	Former En	Was a goo	More thar	0	Fair pay, g	Constant (	Same as many other reviewers have already said but no changes have h
C000017	R00022	3/29/2019	Accountir	New York,	Former En	Interestin	More thar	0	Loved the	Due to fre	Create a stable working atmosphere

GLASSDOOR_REVIEWS:

This table has the following attributes:

COMPANY_ID: This is the foreign key which references the COMPANY_ID in the COMPANY_TBL.

DATE: It give the information about on which date the user posted the review.

EMPLOYEE_TITLE: Employee title is the title in which this person worked and it will be asked while providing the review.

LOCATION: The location of the company where he worked.

EMPLOYEE_STATUS: Employee status is the status of the employment i.e he may be a current employee or a Former employee.

REVIEW_TITLE: Title given to each review.

YEARS_IN_COMPANY: Number of years in worked in that company.

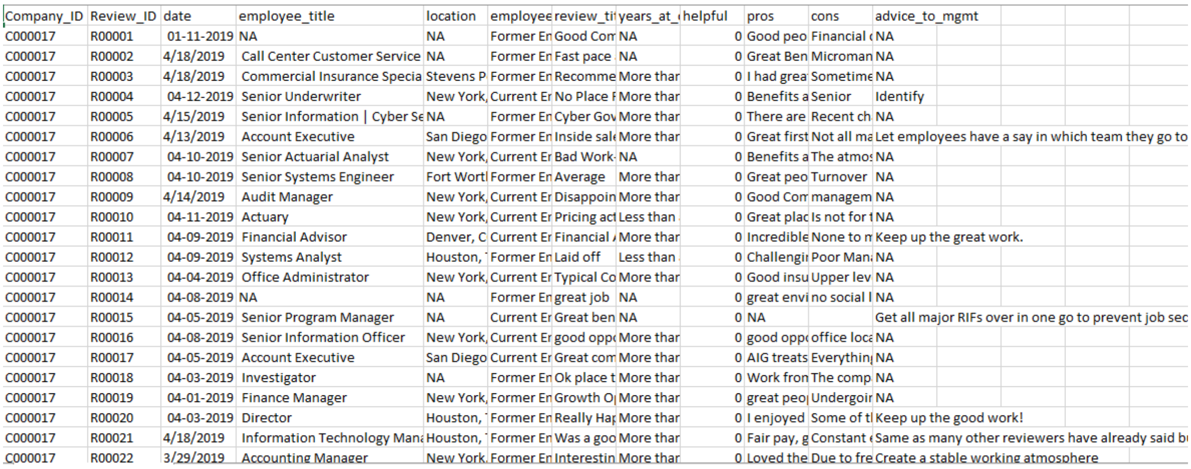
HELPFUL: Is this review helpful or not to others who are reading it.

PROS: Pros about the company.

CONS: Cons about the company

ADVICE_TO_MANAGEMENT: Any advice prescribed to the management.

REVIEW_ID: It represents the unique id given to the each and every review.



Company_ID	Review_ID	date	employee_title	location	employee	review_ti	years_at_	helpful	pros	cons	advice_to_mgmt
C000017	R00001	01-11-2019	NA	NA	Former En	Good Com	NA	0	Good peo	Financial	NA
C000017	R00002	4/18/2019	Call Center Customer Service	NA	Former En	Fast pace	NA	0	Great Ben	Microman	NA
C000017	R00003	4/18/2019	Commercial Insurance Specia	Stevens P	Former En	Recomme	More thar	0	I had grea	Sometime	NA
C000017	R00004	04-12-2019	Senior Underwriter	New York,	Current Er	No Place F	More thar	0	Benefits a	Senior	Identify
C000017	R00005	4/15/2019	Senior Information | Cyber Se	NA	Former En	Cyber Gov	More thar	0	There are	Recent ch	NA
C000017	R00006	4/13/2019	Account Executive	San Diego	Former En	Inside sale	More thar	0	Great first	Not all ma	Let employees have a say in which team they go to
C000017	R00007	04-10-2019	Senior Actuarial Analyst	New York,	Current Er	Bad Work-	NA	0	Benefits a	The atmos	NA
C000017	R00008	04-10-2019	Senior Systems Engineer	Fort Wortl	Former En	Average	More thar	0	Great peo	Turnover	NA
C000017	R00009	4/14/2019	Audit Manager	New York,	Current Er	Disappoin	More thar	0	Good Com	managem	NA
C000017	R00010	04-11-2019	Actuary	New York,	Current Er	Pricing act	Less than	0	Great plac	Is not for t	NA
C000017	R00011	04-09-2019	Financial Advisor	Denver, C	Current Er	Financial /	More thar	0	Incredible	None to m	Keep up the great work.
C000017	R00012	04-09-2019	Systems Analyst	Houston,	Former En	Laid off	Less than	0	Challengi	Poor Man	NA
C000017	R00013	04-04-2019	Office Administrator	New York,	Current Er	Typical Co	More thar	0	Good insu	Upper lev	NA
C000017	R00014	04-08-2019	NA	NA	Former En	great job	NA	0	great envi	no social l	NA
C000017	R00015	04-05-2019	Senior Program Manager	NA	Current Er	Great ben	NA	0	NA		Get all major RIFs over in one go to prevent job sec
C000017	R00016	04-08-2019	Senior Information Officer	New York,	Current Er	good oppo	More thar	0	good oppo	office loca	NA
C000017	R00017	04-05-2019	Account Executive	San Diego	Current Er	Great com	More thar	0	AIG treats	Everythin	NA
C000017	R00018	04-03-2019	Investigator	NA	Former En	Ok place t	More thar	0	Work fron	The comp	NA
C000017	R00019	04-01-2019	Finance Manager	New York,	Former En	Growth O	More thar	0	great peo	Undergoir	NA
C000017	R00020	04-03-2019	Director	Houston,	Former En	Really Ha	More thar	0	I enjoyed	Some of t	Keep up the good work!
C000017	R00021	4/18/2019	Information Technology Mana	Houston,	Former En	Was a goo	More thar	0	Fair pay, g	Constant	Same as many other reviewers have already said b
C000017	R00022	3/29/2019	Accounting Manager	New York,	Former En	Interestin	More thar	0	Loved the	Due to fre	Create a stable working atmosphere

NORMALIZATION:

After the tables are created then the next step is data normalization. Normalization is used to reduce the data redundancy. We can't eliminate the data redundancy completely, but we can reduce the redundancy by dividing the repeating columns in the particular table into a new table and generate a unique Id to that table. Now instead of repeating of all the columns we will give this unique id to the table and it acts a s link between them.

1ST NORMALIZATION FORM:

A table in 1NF should be atomic and have non repeating rows and columns.

Our tables are in 1NF as they satisfy each requirement of first Normalization form.

2ND NORMALIZATION FORM:

There should not be any partial dependency, which means that no value in the table should be dependent on a part of primary key.

Our tables are in 2NF as they satisfy every requirement of second Normalization form.

3RD NORMALIZATION FORM:

A table is said to be in 3NF if no non primary attribute in the table should be dependent on other nonprimary attribute in the table.

Our tables are in 3NF as they satisfy every requirement of third Normalization form.

USES CASES:

Use case-1: Select company with most job postings

Code:

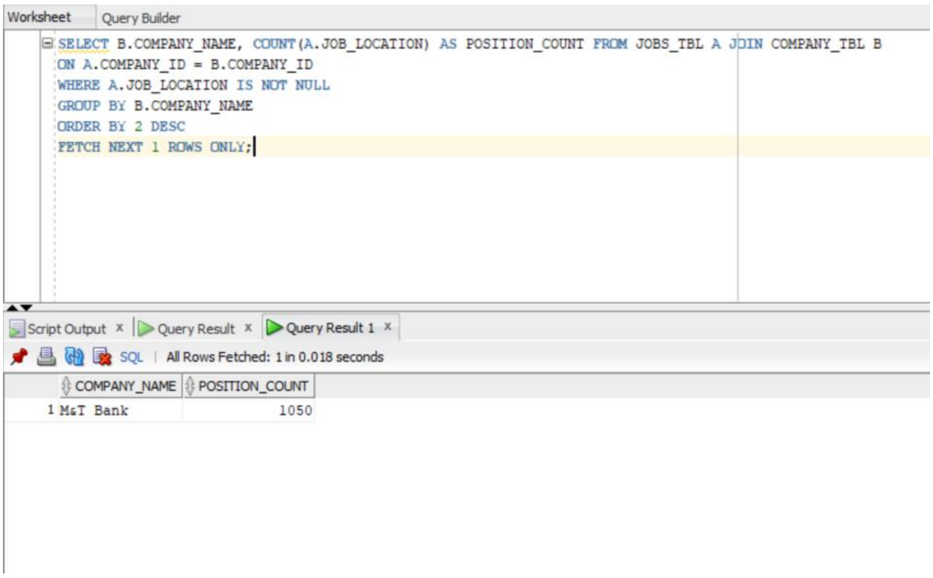
```
SELECT B.COMPANY_NAME, COUNT(A.JOB_LOCATION) AS POSITION_COUNT FROM JOBS_TBL A

JOIN COMPANY_TBL B

ON A.COMPANY_ID = B.COMPANY_ID

WHERE A.JOB_LOCATION IS NOT NULL
```

GROUP BY B. COMPANY_NAME

ORDER BY 2 DESC

FETCH NEXT 1 ROWS ONLY;

Output:



```
SELECT B.COMPANY_NAME, COUNT(A.JOB_LOCATION) AS POSITION_COUNT FROM JOBS_TBL A JOIN COMPANY_TBL B
ON A.COMPANY_ID = B.COMPANY_ID
WHERE A.JOB_LOCATION IS NOT NULL
GROUP BY B.COMPANY_NAME
ORDER BY 2 DESC
FETCH NEXT 1 ROWS ONLY;
```

All Rows Fetched: 1 in 0.018 seconds

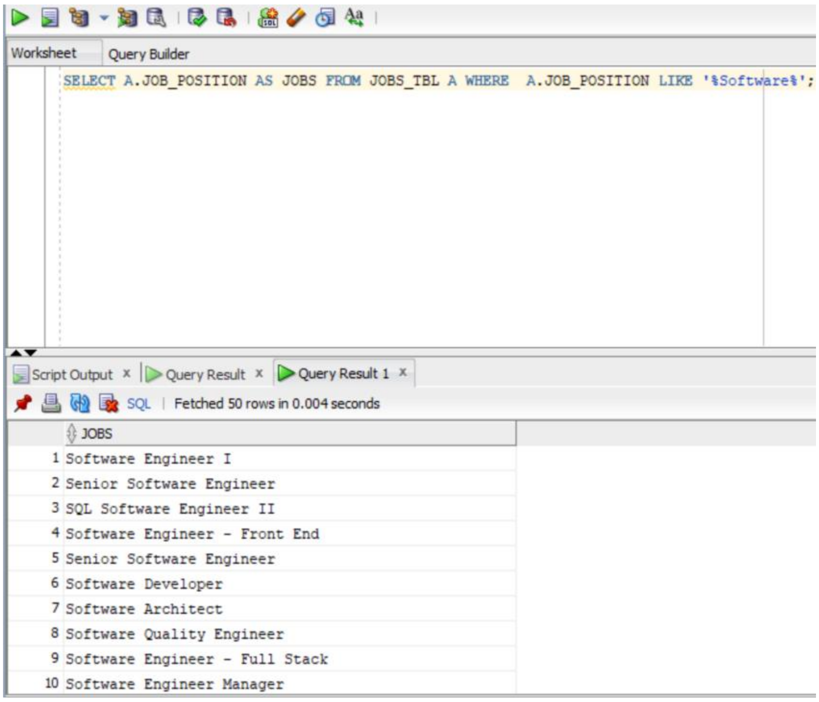
```
   COMPANY_NAME  POSITION_COUNT
 1 M&T Bank                1050
```

Use case-2: Get a List of 'Software Engineering' Jobs

Code:

SELECT A.JOB_POSITION AS JOBS FROM JOBS_TBL A WHERE A.JOB_POSITION LIKE '%Software%';

Output:



```
SELECT A.JOB_POSITION AS JOBS FROM JOBS_TBL A WHERE  A.JOB_POSITION LIKE '%Software%';
```

Fetched 50 rows in 0.004 seconds

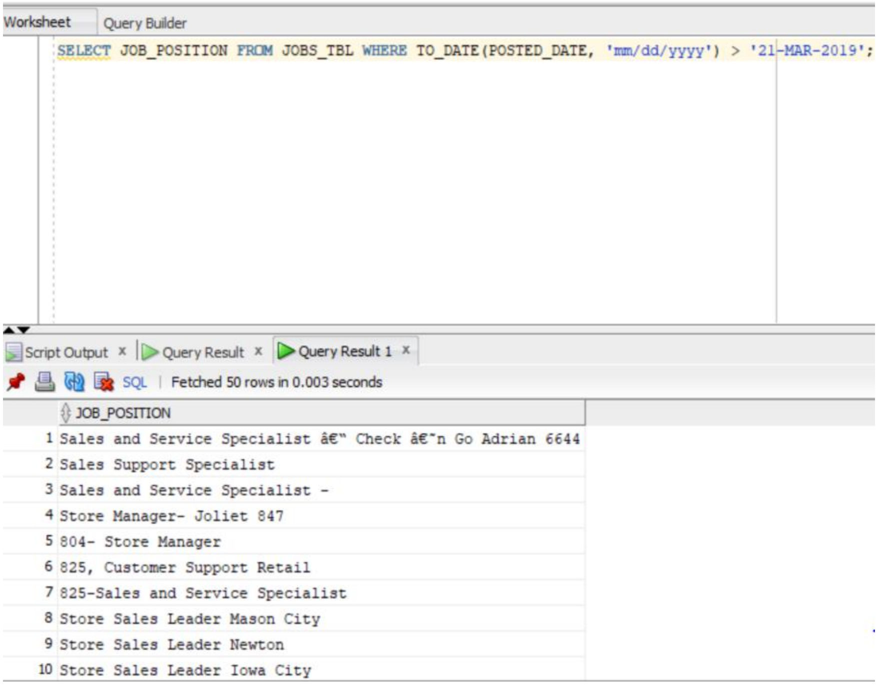
```
    JOBS
  1 Software Engineer I
  2 Senior Software Engineer
  3 SQL Software Engineer II
  4 Software Engineer - Front End
  5 Senior Software Engineer
  6 Software Developer
  7 Software Architect
  8 Software Quality Engineer
  9 Software Engineer - Full Stack
 10 Software Engineer Manager
```

Use case-3: Get a List of most recent postings Jobs

Code:

SELECT JOB_POSITION FROM JOBS_TBL WHERE TO_DATE(POSTED_DATE, 'mm/dd/yyyy') > '21-MAR-2019';

Output:



```sql
SELECT JOB_POSITION FROM JOBS_TBL WHERE TO_DATE(POSTED_DATE, 'mm/dd/yyyy') > '21-MAR-2019';
```

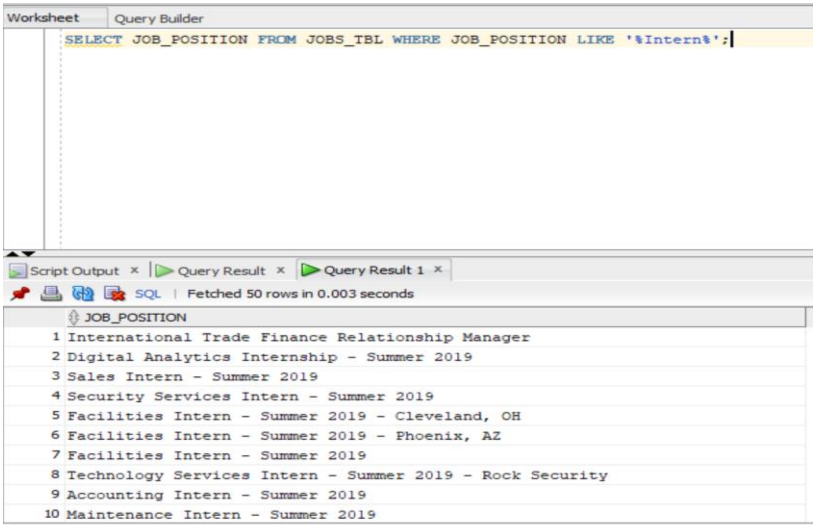
Fetched 50 rows in 0.003 seconds

	JOB_POSITION
1	Sales and Service Specialist â€" Check â€˜n Go Adrian 6644
2	Sales Support Specialist
3	Sales and Service Specialist -
4	Store Manager- Joliet 847
5	804- Store Manager
6	825, Customer Support Retail
7	825-Sales and Service Specialist
8	Store Sales Leader Mason City
9	Store Sales Leader Newton
10	Store Sales Leader Iowa City

Use case-4: Get a list of all the internships

Code:

Output:



```sql
SELECT JOB_POSITION FROM JOBS_TBL WHERE JOB_POSITION LIKE '%Intern%';
```

Fetched 50 rows in 0.003 seconds

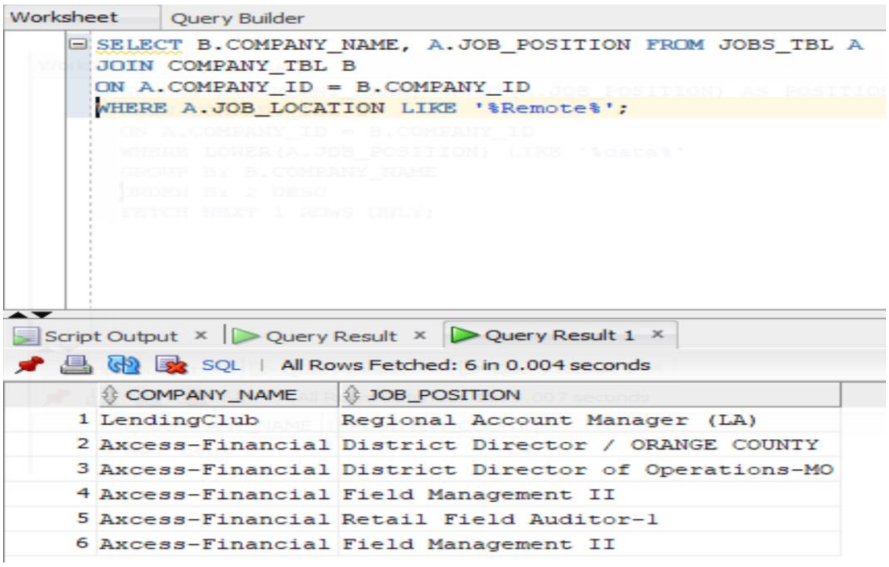
	JOB_POSITION
1	International Trade Finance Relationship Manager
2	Digital Analytics Internship - Summer 2019
3	Sales Intern - Summer 2019
4	Security Services Intern - Summer 2019
5	Facilities Intern - Summer 2019 - Cleveland, OH
6	Facilities Intern - Summer 2019 - Phoenix, AZ
7	Facilities Intern - Summer 2019
8	Technology Services Intern - Summer 2019 - Rock Security
9	Accounting Intern - Summer 2019
10	Maintenance Intern - Summer 2019

Use case-5: Get a list of all the companies with remote jobs

Code:

SELECT B.COMPANY_NAME, A.JOB_POSITION FROM JOBS_TBL A JOIN COMPANY_TBL B ON A.COMPANY_ID = B.COMPANY_ID WHERE A.JOB_LOCATION LIKE '%Remote%';

Output:



```
SELECT B.COMPANY_NAME, A.JOB_POSITION FROM JOBS_TBL A
JOIN COMPANY_TBL B
ON A.COMPANY_ID = B.COMPANY_ID
WHERE A.JOB_LOCATION LIKE '%Remote%';
```

All Rows Fetched: 6 in 0.004 seconds

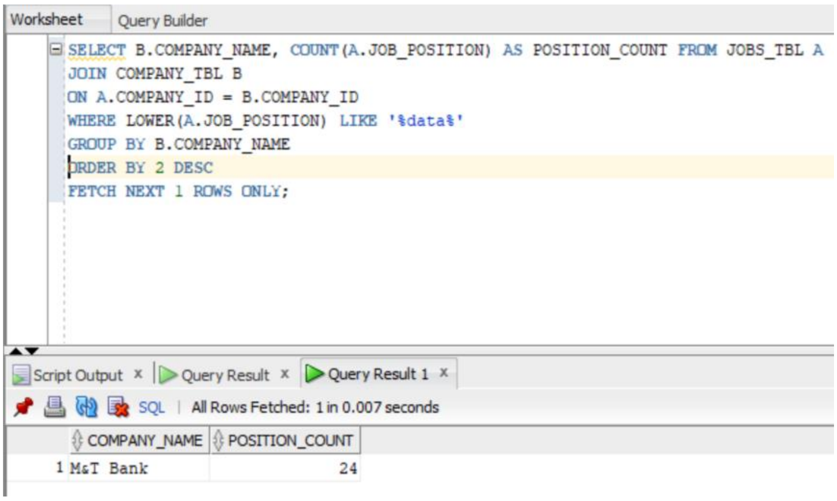
	COMPANY_NAME	JOB_POSITION
1	LendingClub	Regional Account Manager (LA)
2	Axcess-Financial	District Director / ORANGE COUNTY
3	Axcess-Financial	District Director of Operations-MO
4	Axcess-Financial	Field Management II
5	Axcess-Financial	Retail Field Auditor-1
6	Axcess-Financial	Field Management II

Use case-6: Get the name of the company with more data related jobs

Code:

SELECT B.COMPANY_NAME, COUNT(A.JOB_POSITION) AS POSITION_COUNT FROM JOBS_TBL A JOIN COMPANY_TBL B ON A.COMPANY_ID = B.COMPANY_ID WHERE LOWER(A.JOB_POSITION) LIKE '%data%' GROUP BY B.COMPANY_NAME ORDER BY 2 DESC FETCH NEXT 1 ROWS ONLY;

Output:



```
SELECT B.COMPANY_NAME, COUNT(A.JOB_POSITION) AS POSITION_COUNT FROM JOBS_TBL A
JOIN COMPANY_TBL B
ON A.COMPANY_ID = B.COMPANY_ID
WHERE LOWER(A.JOB_POSITION) LIKE '%data%'
GROUP BY B.COMPANY_NAME
ORDER BY 2 DESC
FETCH NEXT 1 ROWS ONLY;
```

All Rows Fetched: 1 in 0.007 seconds

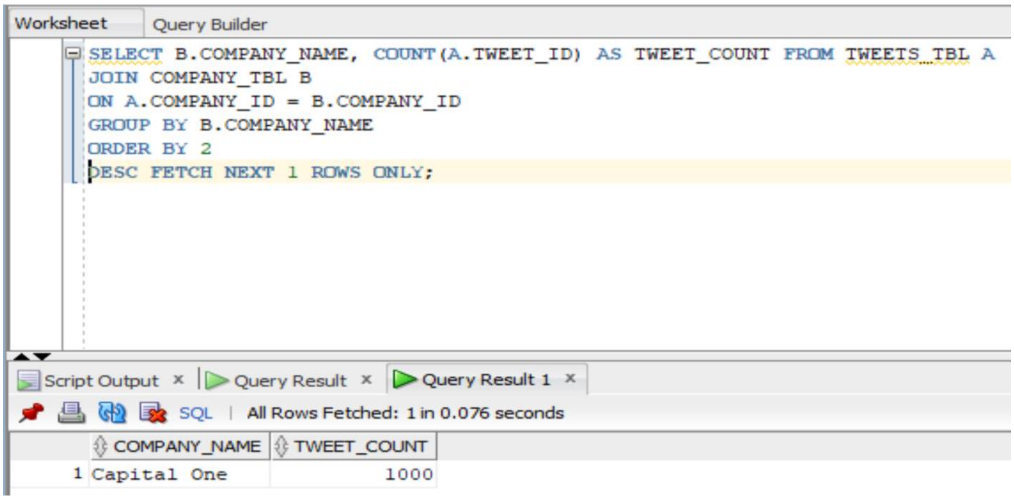
	COMPANY_NAME	POSITION_COUNT
1	M&T Bank	24

Use case-7: Get the name of the company which is active on twitter, the company with most tweets

Code:

SELECT B.COMPANY_NAME, COUNT(A.TWEET_ID) AS TWEET_COUNT FROM TWEETS_TBL A JOIN COMPANY_TBL B ON A.COMPANY_ID = B.COMPANY_ID GROUP BY B.COMPANY_NAME ORDER BY 2 DESC FETCH NEXT 1 ROWS ONLY;

Output:



```
SELECT B.COMPANY_NAME, COUNT(A.TWEET_ID) AS TWEET_COUNT FROM TWEETS_TBL A
JOIN COMPANY_TBL B
ON A.COMPANY_ID = B.COMPANY_ID
GROUP BY B.COMPANY_NAME
ORDER BY 2
DESC FETCH NEXT 1 ROWS ONLY;
```

All Rows Fetched: 1 in 0.076 seconds

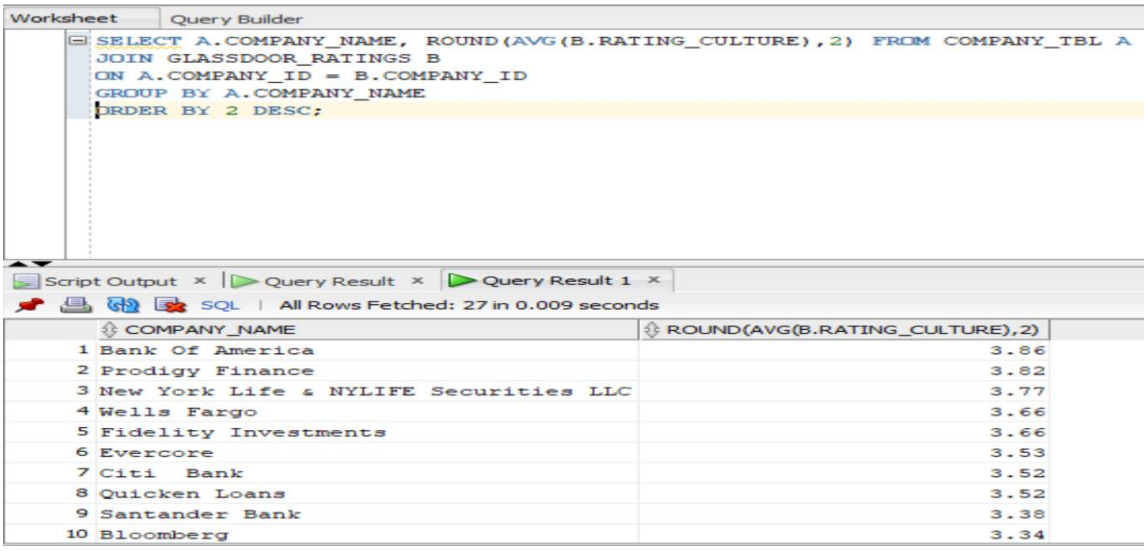
	COMPANY_NAME	TWEET_COUNT
1	Capital One	1000

Use case-8: Get the AVERAGE rating_culture of the companies

Code:

SELECT A.COMPANY_NAME, ROUND(AVG(B.RATING_CULTURE),2) FROM COMPANY_TBL A JOIN GLASSDOOR_RATINGS B ON A.COMPANY_ID = B.COMPANY_ID GROUP BY A.COMPANY_NAME ORDER BY 2 DESC;

Output:



```
SELECT A.COMPANY_NAME, ROUND(AVG(B.RATING_CULTURE),2) FROM COMPANY_TBL A
JOIN GLASSDOOR_RATINGS B
ON A.COMPANY_ID = B.COMPANY_ID
GROUP BY A.COMPANY_NAME
ORDER BY 2 DESC;
```

All Rows Fetched: 27 in 0.009 seconds

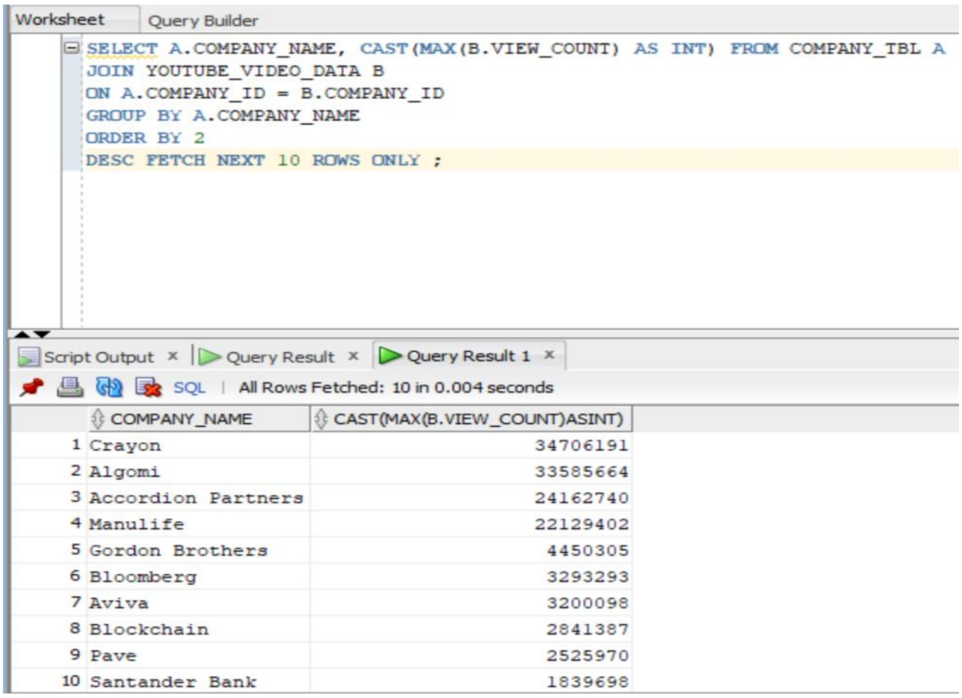
	COMPANY_NAME	ROUND(AVG(B.RATING_CULTURE),2)
1	Bank Of America	3.86
2	Prodigy Finance	3.82
3	New York Life & NYLIFE Securities LLC	3.77
4	Wells Fargo	3.66
5	Fidelity Investments	3.66
6	Evercore	3.53
7	Citi Bank	3.52
8	Quicken Loans	3.52
9	Santander Bank	3.38
10	Bloomberg	3.34

Use case-9: Get the names of the companies with the most YouTube video views¶

Code:

SELECT A.COMPANY_NAME, CAST(MAX(B.VIEW_COUNT) AS INT) FROM COMPANY_TBL A JOIN YOUTUBE_VIDEO_DATA B ON A.COMPANY_ID = B.COMPANY_ID GROUP BY A.COMPANY_NAME ORDER BY 2 DESC FETCH NEXT 10 ROWS ONLY ;

Output:



```
SELECT A.COMPANY_NAME, CAST(MAX(B.VIEW_COUNT) AS INT) FROM COMPANY_TBL A
JOIN YOUTUBE_VIDEO_DATA B
ON A.COMPANY_ID = B.COMPANY_ID
GROUP BY A.COMPANY_NAME
ORDER BY 2
DESC FETCH NEXT 10 ROWS ONLY ;
```

All Rows Fetched: 10 in 0.004 seconds

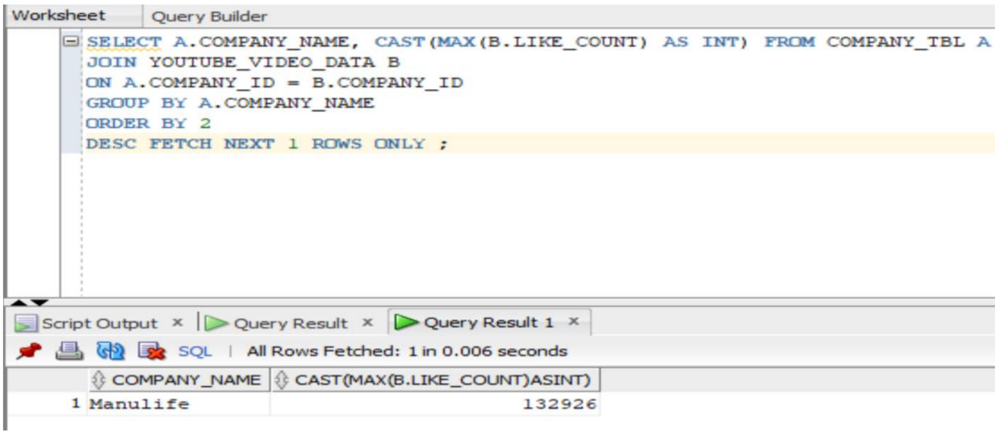
```
     COMPANY_NAME          CAST(MAX(B.VIEW_COUNT)ASINT)
   1 Crayon                                    34706191
   2 Algomi                                    33585664
   3 Accordion Partners                        24162740
   4 Manulife                                  22129402
   5 Gordon Brothers                            4450305
   6 Bloomberg                                  3293293
   7 Aviva                                      3200098
   8 Blockchain                                 2841387
   9 Pave                                       2525970
  10 Santander Bank                             1839698
```

Use case-10: Get the name of the company with more video likes

Code:

SELECT A.COMPANY_NAME, CAST(MAX(B.LIKE_COUNT) AS INT) FROM COMPANY_TBL A JOIN YOUTUBE_VIDEO_DATA B ON A.COMPANY_ID = B.COMPANY_ID GROUP BY A.COMPANY_NAME ORDER BY 2 DESC FETCH NEXT 1 ROWS ONLY ;

Output:



```
SELECT A.COMPANY_NAME, CAST(MAX(B.LIKE_COUNT) AS INT) FROM COMPANY_TBL A
JOIN YOUTUBE_VIDEO_DATA B
ON A.COMPANY_ID = B.COMPANY_ID
GROUP BY A.COMPANY_NAME
ORDER BY 2
DESC FETCH NEXT 1 ROWS ONLY ;
```

All Rows Fetched: 1 in 0.006 seconds

```
     COMPANY_NAME  CAST(MAX(B.LIKE_COUNT)ASINT)
   1 Manulife                            132926
```

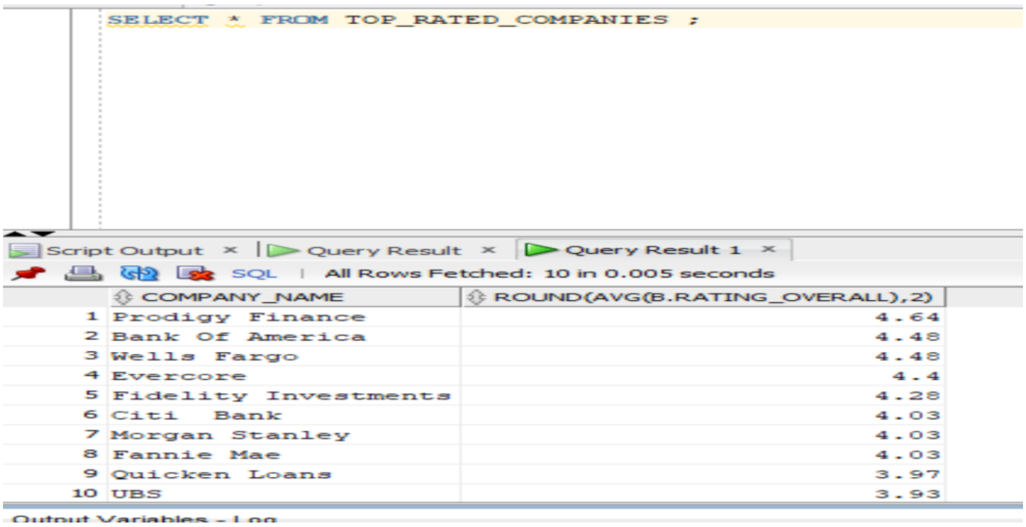
VIEWS:

View-1: Get the TOP 10 rated companies and their average glass door ratings

Code:

```
CREATE VIEW TOP_RATED_COMPANIES AS SELECT A.COMPANY_NAME,
ROUND(AVG(B.RATING_OVERALL),2) FROM COMPANY_TBL A JOIN
GLASSDOOR_RATINGS B ON A.COMPANY_ID = B.COMPANY_ID GROUP BY
A.COMPANY_NAME ORDER BY 2 DESC FETCH NEXT 10 ROWS ONLY;
```

Output:



```
SELECT * FROM TOP_RATED_COMPANIES ;
```

All Rows Fetched: 10 in 0.005 seconds

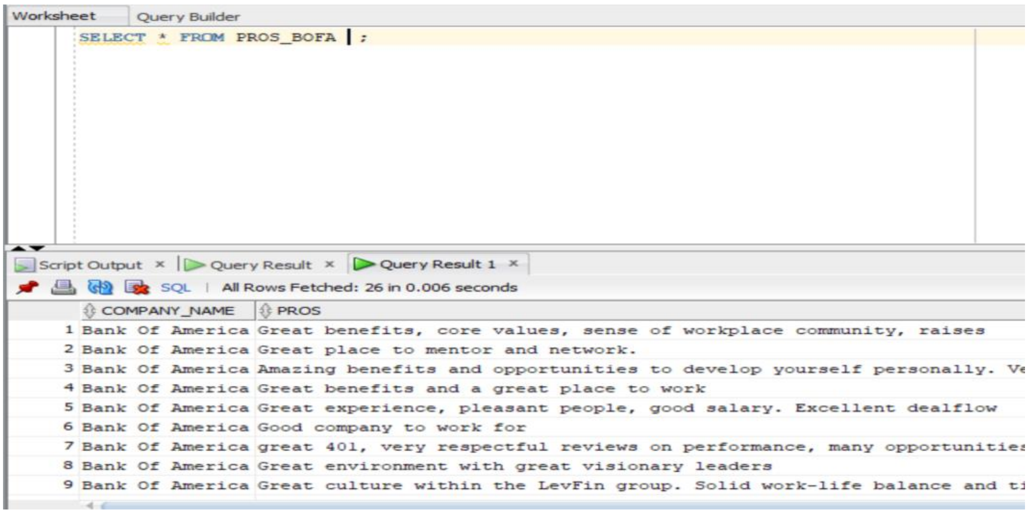
	COMPANY_NAME	ROUND(AVG(B.RATING_OVERALL),2)
1	Prodigy Finance	4.64
2	Bank Of America	4.48
3	Wells Fargo	4.48
4	Evercore	4.4
5	Fidelity Investments	4.28
6	Citi Bank	4.03
7	Morgan Stanley	4.03
8	Fannie Mae	4.03
9	Quicken Loans	3.97
10	UBS	3.93

View-2: Get the company and their glass door pros - BOFA

Code:

```
CREATE VIEW PROS_BOFA AS SELECT A.COMPANY_NAME, B.PROS FROM
COMPANY_TBL A JOIN GLASSDOOR_REVIEWS B ON A.COMPANY_ID = B.COMPANY_ID
AND A.COMPANY_NAME = 'Bank Of America';
```

Output:



```
SELECT * FROM PROS_BOFA ;
```

All Rows Fetched: 26 in 0.006 seconds

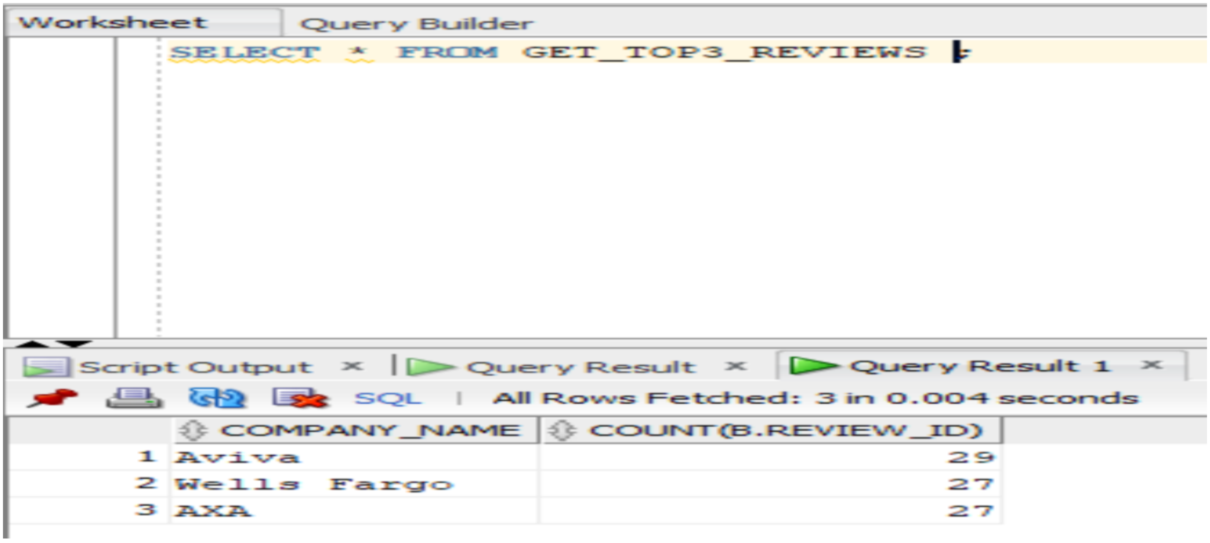
	COMPANY_NAME	PROS
1	Bank Of America	Great benefits, core values, sense of workplace community, raises
2	Bank Of America	Great place to mentor and network.
3	Bank Of America	Amazing benefits and opportunities to develop yourself personally. Ve
4	Bank Of America	Great benefits and a great place to work
5	Bank Of America	Great experience, pleasant people, good salary. Excellent dealflow
6	Bank Of America	Good company to work for
7	Bank Of America	great 401, very respectful reviews on performance, many opportunitie
8	Bank Of America	Great environment with great visionary leaders
9	Bank Of America	Great culture within the LevFin group. Solid work-life balance and ti

View-3: Get the TOP 3 company with more reviews

Code:

CREATE VIEW GET_TOP3_REVIEWS AS SELECT A.COMPANY_NAME, COUNT(B.REVIEW_ID) FROM COMPANY_TBL A JOIN GLASSDOOR_REVIEWS B ON A.COMPANY_ID = B.COMPANY_ID GROUP BY A.COMPANY_NAME ORDER BY 2 DESC FETCH NEXT 3 ROWS ONLY;

Output:



```
SELECT * FROM GET_TOP3_REVIEWS
```

All Rows Fetched: 3 in 0.004 seconds

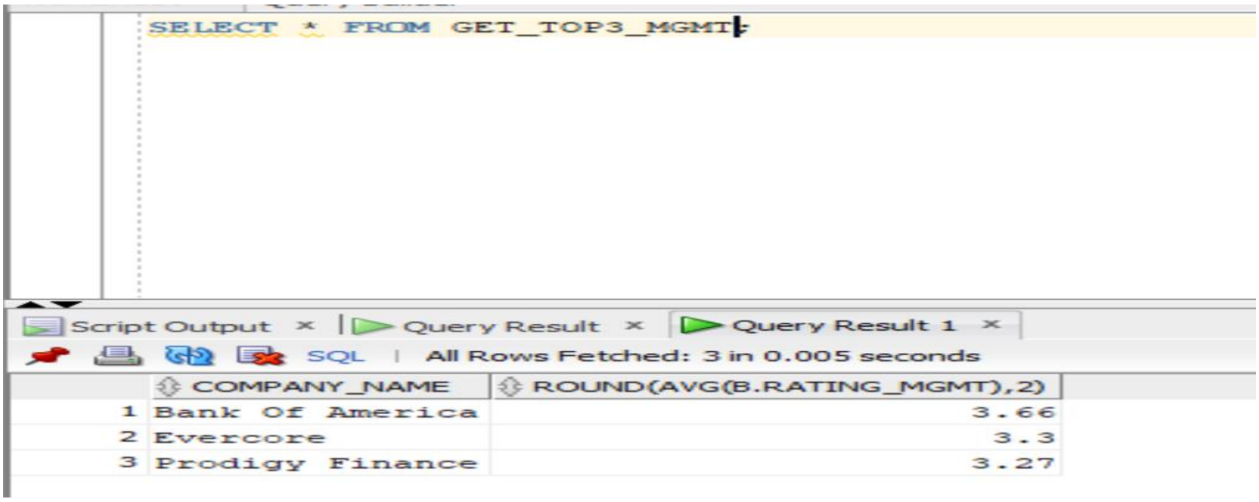
	COMPANY_NAME	COUNT(B.REVIEW_ID)
1	Aviva	29
2	Wells Fargo	27
3	AXA	27

View-4: Get the company and their glass door ratings

Code:

CREATE VIEW GET_TOP3_MGMT AS SELECT A.COMPANY_NAME, ROUND(AVG(B.RATING_MGMT),2) FROM COMPANY_TBL A JOIN GLASSDOOR_RATINGS B ON A.COMPANY_ID = B.COMPANY_ID GROUP BY A.COMPANY_NAME ORDER BY 2 DESC FETCH NEXT 3 ROWS ONLY ;

Output:



```
SELECT * FROM GET_TOP3_MGMT
```

All Rows Fetched: 3 in 0.005 seconds

	COMPANY_NAME	ROUND(AVG(B.RATING_MGMT),2)
1	Bank Of America	3.66
2	Evercore	3.3
3	Prodigy Finance	3.27

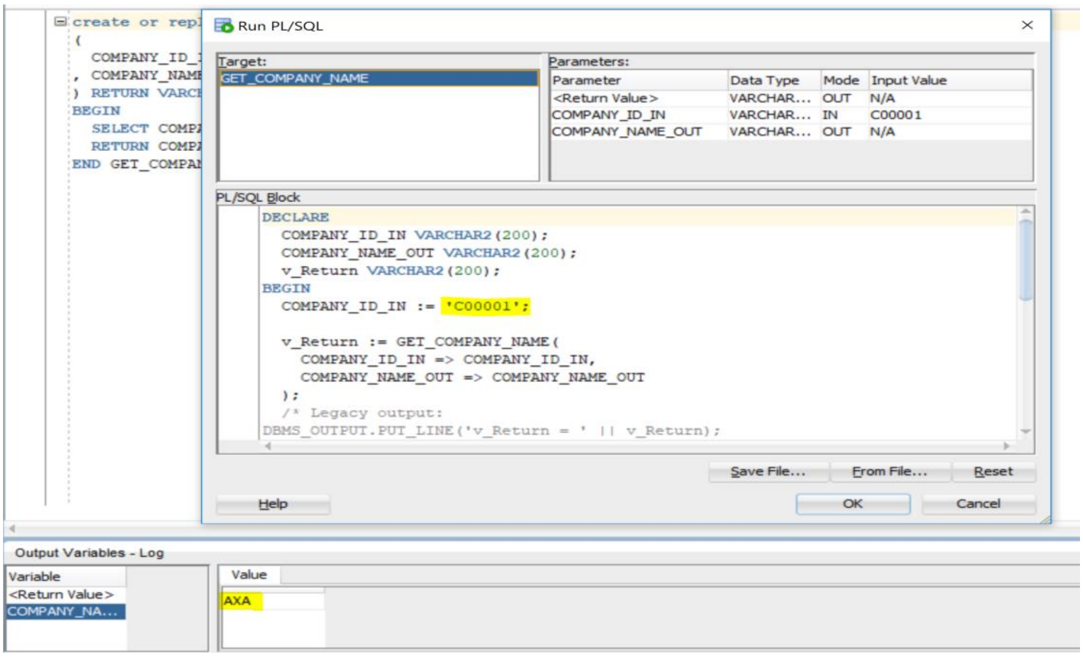
FUNCTIONS:

Function-1: Get the name of the company given COMPANY_ID

Code:

CREATE OR REPLACE FUNCTION GET_COMPANY_NAME (COMPANY_ID_IN IN VARCHAR2 , COMPANY_NAME_OUT OUT VARCHAR2) RETURN VARCHAR2 AS BEGIN SELECT COMPANY_NAME INTO COMPANY_NAME_OUT FROM COMPANY_TBL WHERE COMPANY_ID = COMPANY_ID_IN ; RETURN COMPANY_NAME_OUT; END GET_COMPANY_NAME;

Output:



```
create or repl
(
  COMPANY_ID_I
, COMPANY_NAME
) RETURN VARCH
BEGIN
  SELECT COMPA
  RETURN COMPA
END GET_COMPAN
```

Run PL/SQL

Target: GET_COMPANY_NAME

Parameters:

Parameter	Data Type	Mode	Input Value
<Return Value>	VARCHAR...	OUT	N/A
COMPANY_ID_IN	VARCHAR...	IN	C00001
COMPANY_NAME_OUT	VARCHAR...	OUT	N/A

PL/SQL Block

```
DECLARE
  COMPANY_ID_IN VARCHAR2(200);
  COMPANY_NAME_OUT VARCHAR2(200);
  v_Return VARCHAR2(200);
BEGIN
  COMPANY_ID_IN := 'C00001';

  v_Return := GET_COMPANY_NAME(
    COMPANY_ID_IN => COMPANY_ID_IN,
    COMPANY_NAME_OUT => COMPANY_NAME_OUT
  );
  /* Legacy output:
DBMS_OUTPUT.PUT_LINE('v_Return = ' || v_Return);
```

Output Variables - Log

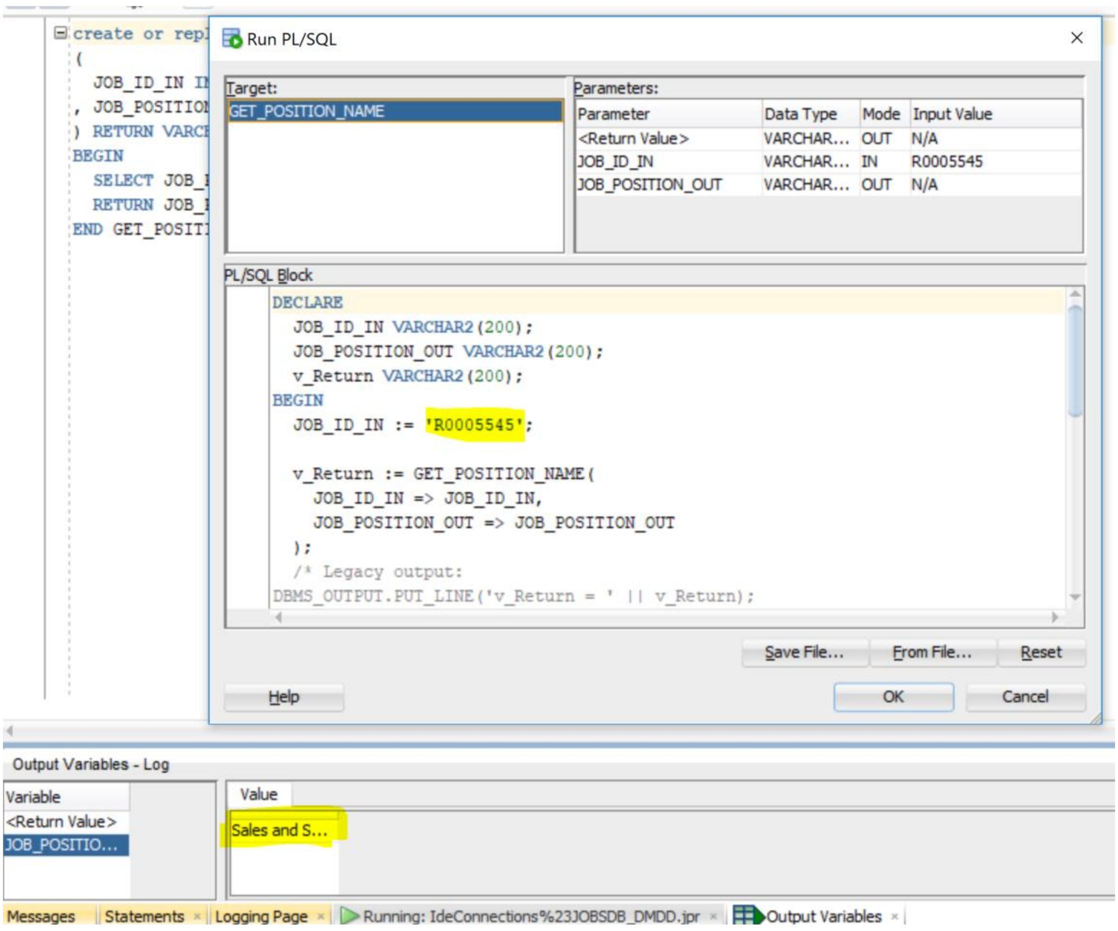
Variable	Value
<Return Value>	
COMPANY_NA...	AXA

Function-2: Get the count of user followers given user_id - ex - 798908432

Code:

CREATE OR REPLACE FUNCTION GET_USER_FOLLOWERS (USER_ID_INPUT IN VARCHAR2 , USER_FOLLOWERS_OUTPUT OUT VARCHAR2) RETURN VARCHAR2 AS BEGIN SELECT USER_FOLLOWERS INTO USER_FOLLOWERS_OUTPUT FROM TWITTER_USER_DATA_TBL WHERE USER_ID = USER_ID_INPUT; RETURN USER_FOLLOWERS_OUTPUT; END GET_USER_FOLLOWERS;

Output:



```
create or repl
(
  JOB_ID_IN I
, JOB_POSITIO
) RETURN VARC
BEGIN
  SELECT JOB_
  RETURN JOB_
END GET_POSITI
```

Target:
GET_POSITION_NAME

Parameters:

Parameter	Data Type	Mode	Input Value
<Return Value>	VARCHAR...	OUT	N/A
JOB_ID_IN	VARCHAR...	IN	R0005545
JOB_POSITION_OUT	VARCHAR...	OUT	N/A

PL/SQL Block

```
DECLARE
  JOB_ID_IN VARCHAR2(200);
  JOB_POSITION_OUT VARCHAR2(200);
  v_Return VARCHAR2(200);
BEGIN
  JOB_ID_IN := 'R0005545';

  v_Return := GET_POSITION_NAME(
    JOB_ID_IN => JOB_ID_IN,
    JOB_POSITION_OUT => JOB_POSITION_OUT
  );
  /* Legacy output:
DBMS_OUTPUT.PUT_LINE('v_Return = ' || v_Return);
```

Output Variables - Log

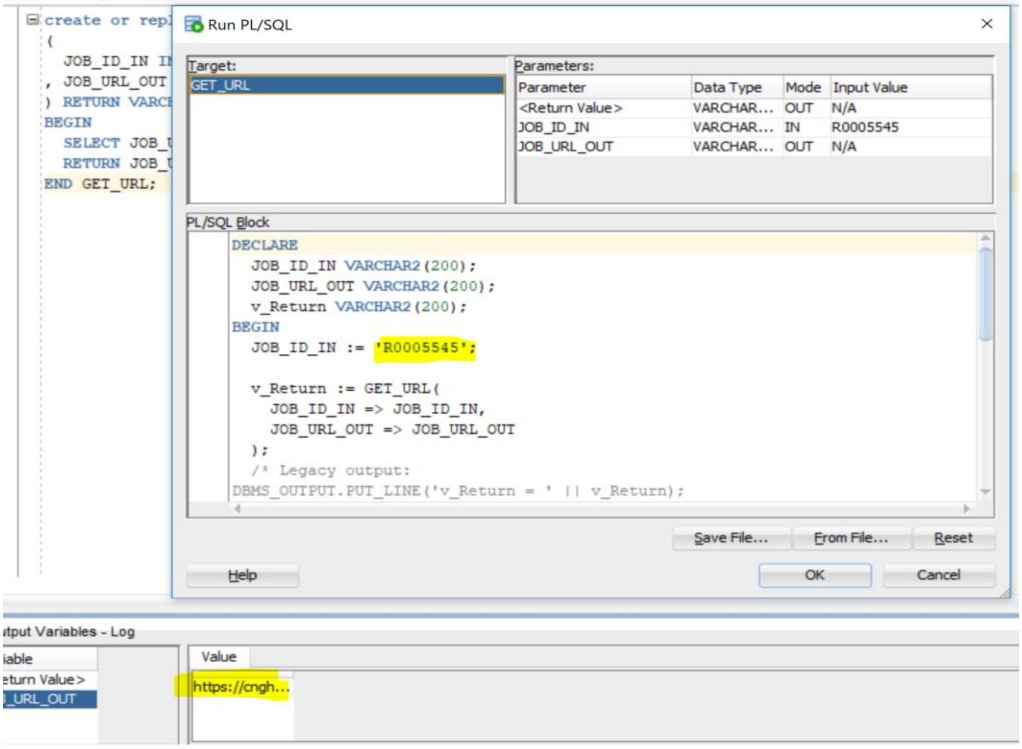
Variable	Value
<Return Value>	
JOB_POSITIO...	Sales and S...

Function-3: Get the position name given the JOB_ID

Code:

CREATE OR REPLACE FUNCTION GET_POSITION_NAME (JOB_ID_IN IN VARCHAR2 , JOB_POSITION_OUT OUT VARCHAR2) RETURN VARCHAR2 AS BEGIN SELECT JOB_POSITION INTO JOB_POSITION_OUT FROM JOBS_TBL WHERE JOB_ID = JOB_ID_IN; RETURN JOB_POSITION_OUT; END GET_POSITION_NAME;

Output:



```
create or repl
(
  JOB_ID_IN IN
, JOB_URL_OUT
) RETURN VARCH
BEGIN
  SELECT JOB_
  RETURN JOB_
END GET_URL;
```

Run PL/SQL

Target:
GET_URL

Parameters:

Parameter	Data Type	Mode	Input Value
<Return Value>	VARCHAR...	OUT	N/A
JOB_ID_IN	VARCHAR...	IN	R0005545
JOB_URL_OUT	VARCHAR...	OUT	N/A

PL/SQL Block

```
DECLARE
  JOB_ID_IN VARCHAR2(200);
  JOB_URL_OUT VARCHAR2(200);
  v_Return VARCHAR2(200);
BEGIN
  JOB_ID_IN := 'R0005545';

  v_Return := GET_URL(
    JOB_ID_IN => JOB_ID_IN,
    JOB_URL_OUT => JOB_URL_OUT
  );
  /* Legacy output:
DBMS_OUTPUT.PUT_LINE('v_Return = ' || v_Return);
```

tput Variables - Log

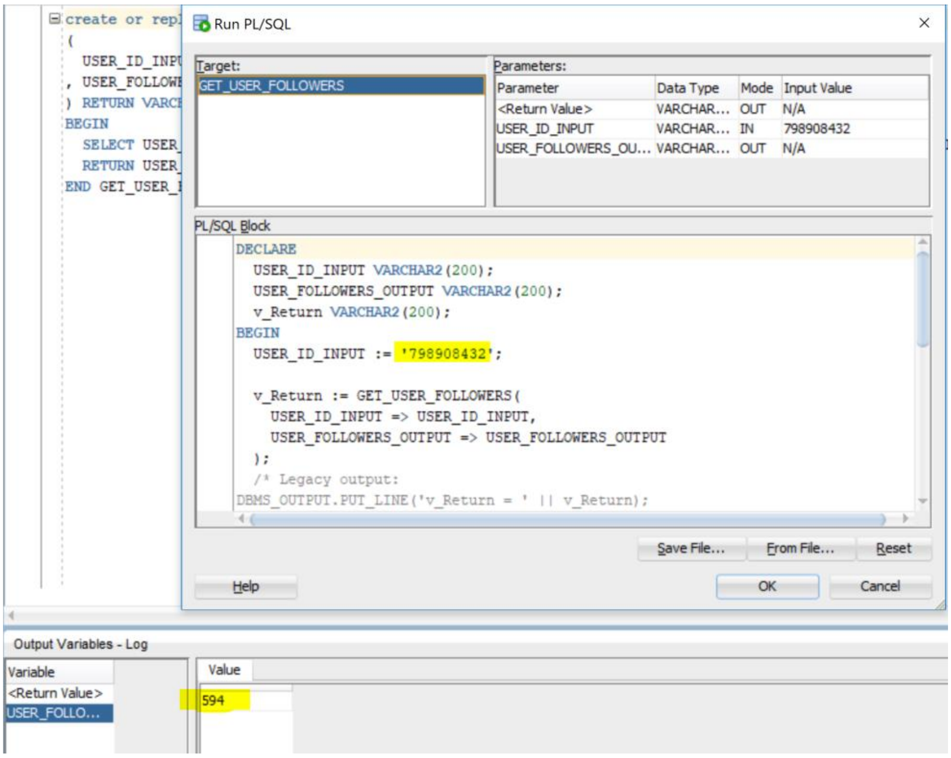
iable	Value
eturn Value>	
_URL_OUT	https://cngh...

Function-4: Get the URL of the job given job_id

Code:

CREATE OR REPLACE FUNCTION GET_URL (JOB_ID_IN IN VARCHAR2 , JOB_URL_OUT OUT VARCHAR2) RETURN VARCHAR2 AS BEGIN SELECT JOB_URL INTO JOB_URL_OUT FROM JOBS_TBL WHERE JOB_ID = JOB_ID_IN; RETURN JOB_URL_OUT; END GET_URL;

Output:



```
create or repl
(
  USER_ID_INPU
, USER_FOLLOWE
) RETURN VARCH
BEGIN
  SELECT USER_
  RETURN USER_
END GET_USER_F
```

Run PL/SQL

Target:
GET_USER_FOLLOWERS

Parameters:

Parameter	Data Type	Mode	Input Value
<Return Value>	VARCHAR...	OUT	N/A
USER_ID_INPUT	VARCHAR...	IN	798908432
USER_FOLLOWERS_OU...	VARCHAR...	OUT	N/A

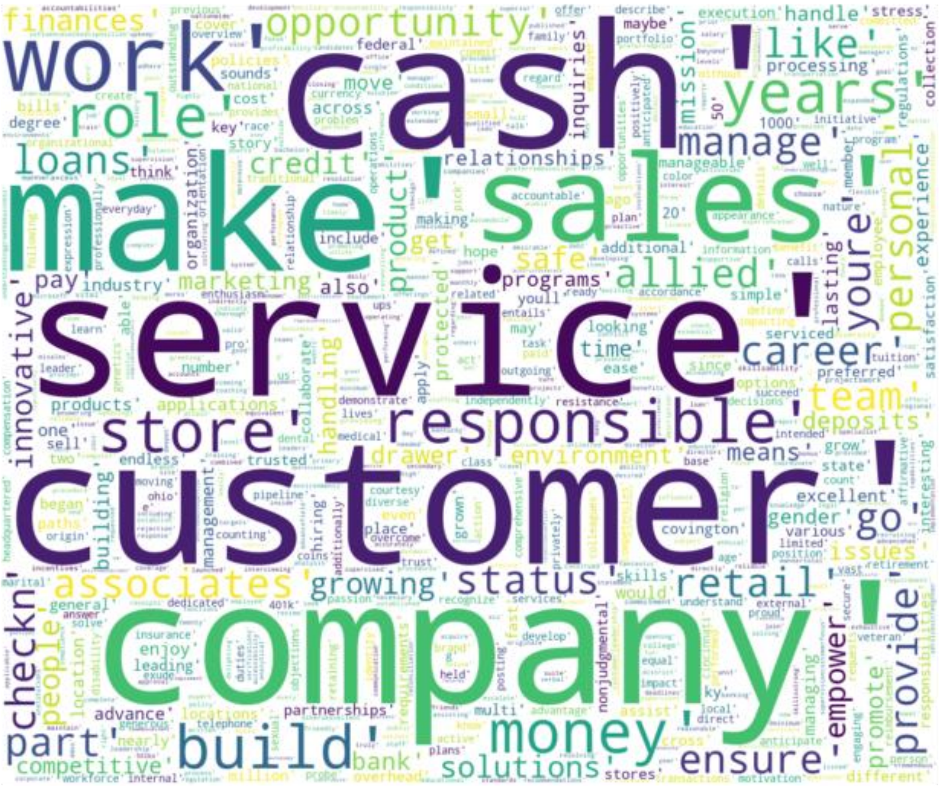
PL/SQL Block

```
DECLARE
  USER_ID_INPUT VARCHAR2(200);
  USER_FOLLOWERS_OUTPUT VARCHAR2(200);
  v_Return VARCHAR2(200);
BEGIN
  USER_ID_INPUT := '798908432';

  v_Return := GET_USER_FOLLOWERS(
    USER_ID_INPUT => USER_ID_INPUT,
    USER_FOLLOWERS_OUTPUT => USER_FOLLOWERS_OUTPUT
  );
  /* Legacy output:
DBMS_OUTPUT.PUT_LINE('v_Return = ' || v_Return);
```

Output Variables - Log

Variable	Value
<Return Value>	
USER_FOLLO...	594

WORD CLOUD:



We can infer from the word cloud that most of the fintech companies focus on customer service. Also you can see finance related terms like cash, money, checkings, deposit etc.