

DATA ENGINEERING CAPSTONE PROJECT

Business objective

For big corporation employee's data from the 1980s, to design data model with all the tables to hold data, import the CSVs into a SQL database, transfer SQL database to HDFS/Hive, and perform analysis using Hive/Impala/Spark/SparkML using the data and create data and ML pipelines. Required to create end to end data pipeline and analyzing the data.

Contents

1. Data used and Description
2. Technology stack used
3. ER diagram (data model)
4. Create database & tables in MySQL server as per the above ER Diagram
5. Create Sqoop job to transfer the data from MySQL to HDFS (Data required to store in Parquet/Avro/Json format)
6. Create database in Hive as per the above ER Diagram and load the data into Hive tables
7. Work on Exploratory data analysis as per the analysis requirement using Hive/Impala and Spark SQL (expecting to get the data from hive tables).
 - a. EDA outputs in hive/impala
 - b. EDA outputs in SPARK
8. ML Model: - Classification Model
 - a. logistic Regression
 - b. Random Forest Classifier
9. Create entire data pipeline and ML pipe line
10. Challenges
11. Way ahead or conclusion

Data used and Description

Given tables with their attributes are -

1. Employees(employees.csv)

- emp_no – Employee Id – Integer – Not Null
- emp_titles_id – designation id – Not Null
- birth_date – Date of Birth – Date Time – Not Null
- first_name – First Name – Character – Not Null
- last_name – Last Name – Character – Not Null
- sex – Gender – Character – Not Null
- hire_date – Employee Hire date –Date Time -Not Null
- no_of_projects – Number of projects worked on – Integer – Not Null
- Last_performance_rating – Last year performance rating – Character – Not Null
- left – Employee left the organization – Boolean – Not Null
- Last_date - Last date of employment (Exit Date) – Date Time

2. Titles(titles.csv)

- title_id – Unique id of type of employee (designation id) – Character – Not Null
- title – Designation – Character – Not Null

3. Salary(salaries.csv)

- emp_no – Employee id – Integer – Not Null
- Salary – Employee's Salary – Integer – Not Null

4. Departments(departments.csv)

- dept_no - Unique id for each department – character – Not Null
- dept_name – Department Name – Character – Not Null

5. Department Managers (dept_manager.csv)

- dept_no - Unique id for each department – character – Not Null
- emp_no – Employee number (head of the department) – Integer – Not Null

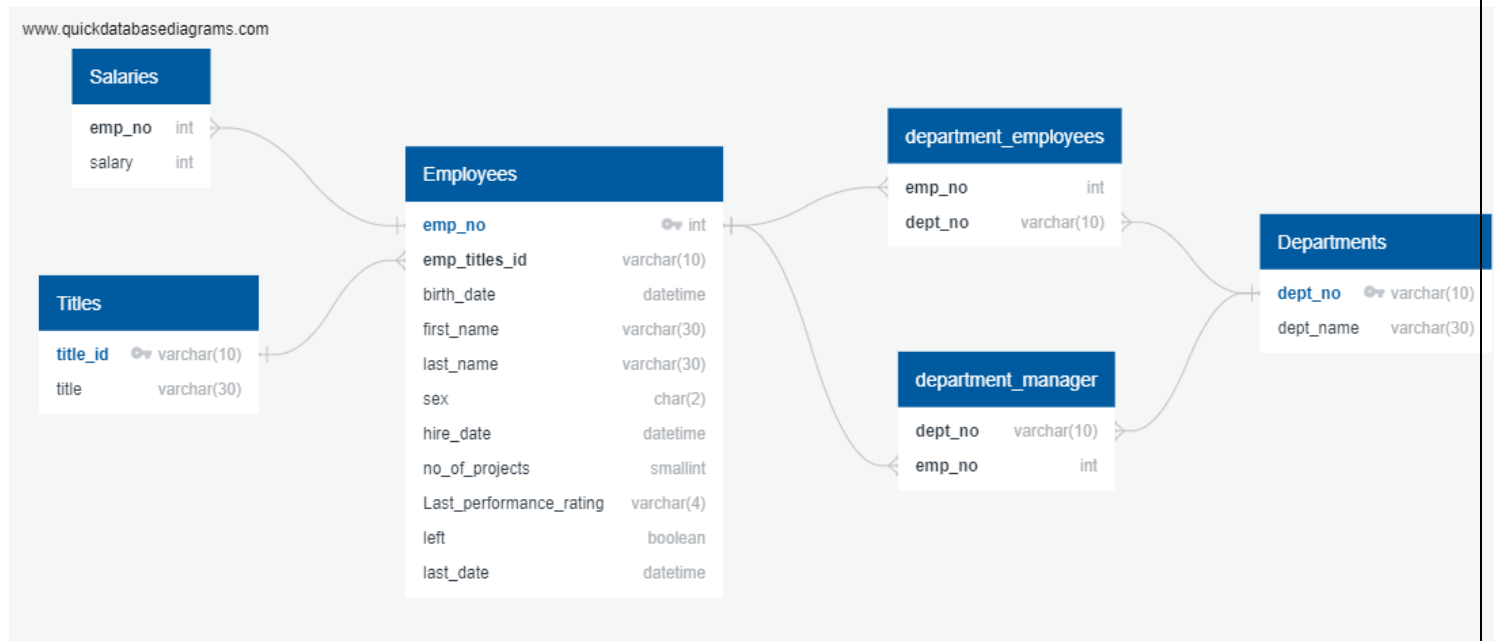
6. Department Employees(dept_emp.csv)

- emp_no – Employee id – Integer – Not Null
- dept_no - Unique id for each department – character – Not Null

Technology stack used

- MySQL (to create database)
- Linux Commands
- Sqoop (Transfer data from MySQL Server to HDFS/Hive)
- HDFS (to store the data)
- Hive (to create database)
- Hive & Impala (to perform the EDA)
- SparkSQL (to perform the EDA)
- SparkML (to perform model building)

ER Diagram (data model)



Create database & tables in MySQL server as per the above ER Diagram

MySQL codes:

1. Login to mysql from shell

```
mysql -u anabig114225 -pBigdata123
```

```
show databases;
```

```
use databasename;
```

2. a) Create tables in mysql manually

```
CREATE TABLE employees(  
  emp_no int not null,  
  emp_titles_id varchar(10) not null,  
  birth_date varchar(20) not null,  
  first_name varchar(30) not null,  
  last_name varchar(30) not null,  
  sex char(2) not null,  
  hire_date varchar(20) not null,  
  no_of_projects smallint not null,  
  Last_performance_rating varchar(4) not null,  
  left_company boolean not null,  
  last_date varchar(20),  
  PRIMARY KEY(emp_no),  
  CONSTRAINT FK_title_id FOREIGN KEY (emp_titles_id) REFERENCES titles(title_id));
```

```
CREATE TABLE titles(  
  title_id varchar(10) not null,  
  title varchar(30) not null,  
  PRIMARY KEY(title_id),  
  );
```

```
CREATE TABLE salaries(  
  emp_no int not null,  
  salary int not null,  
  CONSTRAINT FK_emp_no FOREIGN KEY (emp_no) REFERENCES employees(emp_no) );
```

```
CREATE TABLE departments(  
  dept_no varchar(10) not null,
```

```
dept_name varchar(30) not null,  
PRIMARY KEY(dept_no) );
```

```
CREATE TABLE department_manager(  
dept_no varchar(10) not null,  
emp_no int not null,  
CONSTRAINT FK_dept_no FOREIGN KEY (dept_no) REFERENCES departments(dept_no),  
CONSTRAINT FK_emp_no1 FOREIGN KEY (emp_no) REFERENCES employees(emp_no) );
```

```
CREATE TABLE department_employees(  
emp_no int not null,  
dept_no varchar(10) not null,  
CONSTRAINT FK_dept_no2 FOREIGN KEY (emp_no) REFERENCES employees(emp_no),  
CONSTRAINT FK_emp_no2 FOREIGN KEY (dept_no) REFERENCES departments(dept_no) );
```

2.b) . OR create table using the .sql file where all the above create commands written

upload create_tabeles.sql to ftp (<https://npbdh.cloudloka.com/ftp>)

run the below command to create tables under

```
-----CREATING TABLES-----  
mysql> source create_data.sql  
mysql> show tables;  
+-----+  
| Tables_in_anabig114225 |  
+-----+  
| department_employees |  
| department_manager |  
| departments |  
| employees |  
| salaries |  
| titles |  
+-----+  
6 rows in set (0.00 sec)  
|
```

3. Loading data

-----LOADING DATA-----

```
load data local infile '/home/anabig114225/Data/departments.csv' into table departments FIELDS TERMINATED BY ',' IGNORE 1 LINES;
load data local infile '/home/anabig114225/Data/dept_emp.csv' into table department_employees FIELDS TERMINATED BY ',' IGNORE 1 LINES;
load data local infile '/home/anabig114225/Data/dept_manager.csv' into table department_manager FIELDS TERMINATED BY ',' IGNORE 1 LINES;
load data local infile '/home/anabig114225/Data/salaries.csv' into table salaries FIELDS TERMINATED BY ',' IGNORE 1 LINES;
load data local infile '/home/anabig114225/Data/titles.csv' into table titles FIELDS TERMINATED BY ',' IGNORE 1 LINES;
load data local infile '/home/anabig114225/Data/employees.csv' into table employees FIELDS TERMINATED BY ',' IGNORE 1 LINES;
```

```
mysql> load data local infile '/home/anabig114225/Data/departments.csv' into table departments FIELDS TERMINATED BY ',' IGNORE 1 LINES;
Query OK, 9 rows affected (0.00 sec)
Records: 9 Deleted: 0 Skipped: 0 Warnings: 0
```

```
mysql> load data local infile '/home/anabig114225/Data/dept_emp.csv' into table department_employees FIELDS TERMINATED BY ',' IGNORE 1 LINES;
Query OK, 331603 rows affected (1.36 sec)
Records: 331603 Deleted: 0 Skipped: 0 Warnings: 0
```

```
mysql> load data local infile '/home/anabig114225/Data/dept_manager.csv' into table department_manager FIELDS TERMINATED BY ',' IGNORE 1 LINES;
Query OK, 24 rows affected (0.00 sec)
Records: 24 Deleted: 0 Skipped: 0 Warnings: 0
```

```
mysql> load data local infile '/home/anabig114225/Data/salaries.csv' into table salaries FIELDS TERMINATED BY ',' IGNORE 1 LINES;
Query OK, 300024 rows affected (1.20 sec)
Records: 300024 Deleted: 0 Skipped: 0 Warnings: 0
```

```
mysql> load data local infile '/home/anabig114225/Data/titles.csv' into table titles FIELDS TERMINATED BY ',' IGNORE 1 LINES;
Query OK, 7 rows affected (0.00 sec)
Records: 7 Deleted: 0 Skipped: 0 Warnings: 0
```

```
mysql> load data local infile '/home/anabig114225/Data/employees.csv' into table employees FIELDS TERMINATED BY ',' IGNORE 1 LINES;
Query OK, 300024 rows affected (3.14 sec)
Records: 300024 Deleted: 0 Skipped: 0 Warnings: 0
```

4. Verifying whether our data is properly inserted

-----CHECKING OUR DATA-----

```
select * from titles;
```

title_id	title
e0001	Assistant Engineer
e0002	Engineer
e0003	Senior Engineer
e0004	Technique Leader
m0001	Manager
s0001	Staff
s0002	Senior Staff

7 rows in set (0.00 sec)

```
mysql> select * from departments;
```

dept_no	dept_name
d001	"Marketing"
d002	"Finance"
d003	"Human Resources"
d004	"Production"
d005	"development"
d006	"Quality Management"
d007	"Sales"
d008	"Research"
d009	"Customer Service"

9 rows in set (0.00 sec)

```
mysql> select * from department_manager limit 10;
```

dept_no	emp_no
d001	110022
d001	110039
d002	110085
d002	110114
d003	110183
d003	110228
d004	110303
d004	110344
d004	110386
d004	110420

10 rows in set (0.00 sec)

```
mysql> select * from department_employees limit 10;
```

emp_no	dept_no
10001	d005
10002	d007
10003	d004
10004	d004
10005	d003
10006	d005
10007	d008
10008	d005
10009	d006
10010	d004

10 rows in set (0.00 sec)

```
mysql> select * from salaries limit 4;
```

emp_no	salary
10001	60117
10002	65828
10003	40006
10004	40054

```
4 rows in set (0.00 sec)
```

```
mysql> select * from employees limit 10;
```

emp_no	emp_titles_id	birth_date	first_name	last_name	sex	hire_date	no_of_projects	Last_performance_rating	left_company	last_date
10001	e0003	9/2/1953	Georgi	Facello	M	6/26/1986	9	C	1	7/30/1994
	s0001	6/2/1964	Bezalel	Simmel	F	11/21/1985	8	B	0	
	e0003	12/3/1959	Parto	Bamford	M	8/28/1986	1	C	0	
	e0003	5/1/1954	Chirstian	Koblick	M	12/1/1986	5	A	0	
	s0001	1/21/1955	Kyoichi	Maliniak	M	9/12/1989	6	A	0	
10007	e0003	4/20/1953	Anneke	Preusig	F	6/2/1989	10	B	0	
	s0001	5/23/1957	Tzvetan	Zielinski	F	2/10/1989	6	B	1	9/18/2002
	e0001	2/19/1958	Saniya	Kalloufi	M	9/15/1994	9	C	0	
	e0003	4/19/1952	Sumant	Peac	F	2/18/1985	8	B	0	
	e0002	6/1/1963	Duangkaew	Piveteau	F	8/24/1989	4	A	0	

```
10 rows in set (0.00 sec)
```

Create Sqoop job to transfer the data from MySQL to HDFS (Data required to store in Parquet/Avro/Json format)

In shell

Importing the data using sqoop :- saving in avro format at new directory as projectdata

```
sqoop import-all-tables --connect jdbc:mysql://ip-10-1-1-204.ap-south-1.compute.internal:3306/anabig114225  
--username anabig114225 --password Bigdata123 --compression-codec=snappy --as-avrodatafile --warehouse-  
dir=/user/anabig114225/projectdata --m 1 --driver com.mysql.jdbc.Driver
```

```
[anabig114225@ip-10-1-1-204 ~]$ hdfs dfs -ls projectdata
```

Found 6 items

```
drwxr-xr-x - anabig114225 anabig114225 0 2022-05-17 17:19 projectdata/department_employees  
drwxr-xr-x - anabig114225 anabig114225 0 2022-05-17 17:20 projectdata/department_manager  
drwxr-xr-x - anabig114225 anabig114225 0 2022-05-17 17:20 projectdata/departments  
drwxr-xr-x - anabig114225 anabig114225 0 2022-05-17 17:20 projectdata/employees  
drwxr-xr-x - anabig114225 anabig114225 0 2022-05-17 17:21 projectdata/salaries  
drwxr-xr-x - anabig114225 anabig114225 0 2022-05-17 17:21 projectdata/titles
```

```
[anabig114225@ip-10-1-1-204 ~]$ hdfs dfs -ls projectschema
```

Locating the schema :- schema is saved as .avsc format

```
[anabig114225@ip-10-1-1-204 ~]$ ls *.avsc
```

```
department_employees.avsc department_manager.avsc departments.avsc employees.avsc salaries.avsc titles.avsc
```

```
[anabig114225@ip-10-1-1-204 ~]$
```


creating a new directory as projectschema in hdfs where the schema will be saved

```
hdfs dfs -mkdir projectschema
```

```
hdfs dfs -copyFromLocal ~/*.avsc projectschema
```

```
[anabig114225@ip-10-1-1-204 ~]$ hdfs dfs -ls projectschema
Found 6 items
-rw-r--r--  3 anabig114225 anabig114225      437 2022-05-17 17:26 projectschema/department_employees.avsc
-rw-r--r--  3 anabig114225 anabig114225      431 2022-05-17 17:26 projectschema/department_manager.avsc
-rw-r--r--  3 anabig114225 anabig114225      420 2022-05-17 17:26 projectschema/departments.avsc
-rw-r--r--  3 anabig114225 anabig114225     1735 2022-05-17 17:26 projectschema/employees.avsc
-rw-r--r--  3 anabig114225 anabig114225      395 2022-05-17 17:26 projectschema/salaries.avsc
-rw-r--r--  3 anabig114225 anabig114225      399 2022-05-17 17:26 projectschema/titles.avsc
```

Now data has been successfully transferred to HDFS.

Create database in Hive as per the above ER Diagram and load the data into Hive tables

```
create database project_de;
```

```
use project_de;
```

```
CREATE EXTERNAL TABLE employees STORED AS AVRO
```

```
LOCATION '/user/anabig114225/projectdata/employees'
```

```
TBLPROPERTIES ('avro.schema.url'='/user/anabig114225/projectschema/employees.avsc');
```

```
CREATE EXTERNAL TABLE titles STORED AS AVRO
```

```
LOCATION '/user/anabig114225/projectdata/titles'
```

```
TBLPROPERTIES ('avro.schema.url'='/user/anabig114225/projectschema/titles.avsc');
```

```
CREATE EXTERNAL TABLE salaries STORED AS AVRO
```

```
LOCATION '/user/anabig114225/projectdata/salaries'
```

```
TBLPROPERTIES ('avro.schema.url'='/user/anabig114225/projectschema/salaries.avsc');
```

```
CREATE EXTERNAL TABLE departments STORED AS AVRO
```

```
LOCATION '/user/anabig114225/projectdata/departments'
```

```
TBLPROPERTIES ('avro.schema.url'='/user/anabig114225/projectschema/departments.avsc');
```

```
CREATE EXTERNAL TABLE department_manager STORED AS AVRO
```

```
LOCATION '/user/anabig114225/projectdata/department_manager'
```

```
TBLPROPERTIES ('avro.schema.url'='/user/anabig114225/projectschema/department_manager.avsc');
```

```
CREATE EXTERNAL TABLE department_employees STORED AS AVRO
```

```
LOCATION '/user/anabig114225/projectdata/department_employees'
```

```
TBLPROPERTIES ('avro.schema.url'='/user/anabig114225/projectschema/department_employees.avsc');
```

< project_de

Tables

(9) +

Filter...

- department
- department_employees
- department_manager
- departments
- employee_final
 - emp_no (int)
 - emp_titles_id (string)
 - birth_date (date)
 - first_name (string)
 - last_name (string)
 - sex (string)
 - hire_date (date)
 - no_of_projects (int)
 - last_performance_rating (string)
 - left_company (int)
 - last_date (string)
- employees
- salaries
- salary_dist
- titles

Database project_de Type text

```

1
2 -----project_de table creation -----
3
4 create database project_de;
5
6 use project_de;
7
8 CREATE EXTERNAL TABLE employees STORED AS AVRO
9 LOCATION '/user/anabig114225/projectdata/employees'
10 TBLPROPERTIES ('avro.schema.url'='/user/anabig114225/projectschema/employees.avsc');
11
12 CREATE EXTERNAL TABLE titles STORED AS AVRO
13 LOCATION '/user/anabig114225/projectdata/titles'
14 TBLPROPERTIES ('avro.schema.url'='/user/anabig114225/projectschema/titles.avsc');
15
16 CREATE EXTERNAL TABLE salaries STORED AS AVRO
17 LOCATION '/user/anabig114225/projectdata/salaries'
18 TBLPROPERTIES ('avro.schema.url'='/user/anabig114225/projectschema/salaries.avsc');
19
20 CREATE EXTERNAL TABLE departments STORED AS AVRO
21 LOCATION '/user/anabig114225/projectdata/departments'
22 TBLPROPERTIES ('avro.schema.url'='/user/anabig114225/projectschema/departments.avsc');
23
24 CREATE EXTERNAL TABLE department_manager STORED AS AVRO
25 LOCATION '/user/anabig114225/projectdata/department_manager'
26 TBLPROPERTIES ('avro.schema.url'='/user/anabig114225/projectschema/department_manager.avsc');
27
28 CREATE EXTERNAL TABLE department_employees STORED AS AVRO
29 LOCATION '/user/anabig114225/projectdata/department_employees'
30 TBLPROPERTIES ('avro.schema.url'='/user/anabig114225/projectschema/department_employees.avsc');
31
32 -----
33

```

Work on Exploratory data analysis as per the analysis requirement using Hive/Impala and Spark SQL (expecting to get the data from hive tables).

EDA outputs in hive

1. A list showing employee number, last name, first name, sex, and salary for each employee.

```
select e.emp_no, last_name, first_name, sex, salary from employees e
inner join salaries s on e.emp_no=s.emp_no;
```

```
23 select e.emp_no, last_name, first_name, sex, salary from employees e
24 inner join salaries s on e.emp_no=s.emp_no;
```

```
INFO : Completed executing command(queryId=hive_20220517194838_801faf9c-9fa7-44dd-b2d6-91a445d8b10c): Time taken: 37.814 seconds
INFO : OK
```

Query History

Saved Queries

Results (300,024)

	e.emp_no	last_name	first_name	sex	salary
1	10001	Facello	Georgi	M	60117
2	10002	Simmel	Bezalel	F	65828
3	10003	Bamford	Parto	M	40006
4	10004	Koblick	Chirstian	M	40054
5	10005	Maliniak	Kyoichi	M	78228
6	10006	Preusig	Anneke	F	40000
7	10007	Zielinski	Tzvetan	F	56724
8	10008	Kalloufi	Saniya	M	46671
9	10009	Peac	Sumant	F	60929
10	10010	Piveteau	Duangkaew	F	72488
11	10011	Sluis	Mary	F	42365
12	10012	Bridgland	Patricio	M	40000

2. A list showing first name, last name, and hire date for employees who were hired in 1986.

```
select first_name, last_name, hire_date from employees
where cast(substring_index(hire_date,"",-1) as int ) = 1986;
```

The screenshot displays a SQL query execution environment. At the top, a code editor shows the query: `select first_name, last_name, hire_date from employees where cast(substring_index(hire_date,"",-1) as int) = 1986;`. Below the editor, a status bar indicates the query was completed successfully, with a message: "INFO : Completed executing command(queryId=hive_20220517194047_d7af859d-fff0-4350-808a-82843e51655b): Time taken: 23.316 seconds". A link to the job details is provided: [job_1652166004796_4293](#). Below the status bar, a table titled "Results (36,150)" displays the query results. The table has four columns: a row number, first_name, last_name, and hire_date. The results show 11 employees hired in 1986.

	first_name	last_name	hire_date
1	Georgi	Facello	6/26/1986
2	Parto	Bamford	8/28/1986
3	Chirstian	Koblick	12/1/1986
4	Sanjiv	Zschoche	2/4/1986
5	Kwee	Schusler	2/26/1986
6	Kshitij	Gils	3/27/1986
7	Zhongwei	Rosen	10/30/1986
8	Xinglin	Eugenio	9/8/1986
9	Sudharsan	Flasterstein	8/12/1986
10	Kendra	Hofting	3/14/1986
11	Hilari	Morton	7/15/1986

3. A list showing the manager of each department with the following information: department number, department name, the manager's employee number, last name, first name.

```
select d.dept_no, d.dept_name, dm.emp_no, last_name, first_name from departments d
inner join department_manager dm on d.dept_no = dm.dept_no
inner join employees e on e.emp_no = dm.emp_no
```

```

26 select d.dept_no, d.dept_name, dm.emp_no, last_name, first_name from departments d
27 inner join department_manager dm on d.dept_no = dm.dept_no
28 inner join employees e on e.emp_no = dm.emp_no

```

```

INFO : Completed executing command(queryId=hive_20220517201354_f85be1e4-9e9b-481c-b032-ce12c08b5147): Time taken: 31.334 seconds
INFO : OK

```

Query History Saved Queries Results (24)

	d.dept_no	d.dept_name	dm.emp_no	last_name	first_name
1	d001	"Marketing"	110022	Markovitch	Margareta
2	d001	"Marketing"	110039	Minakawa	Vishwani
3	d002	"Finance"	110085	Alpin	Ebru
4	d002	"Finance"	110114	Legleitner	Isamu
5	d003	"Human Resources"	110183	Ossenbruggen	Shirish
6	d003	"Human Resources"	110228	Sigstam	Karsten

4. A list showing the department of each employee with the following information: employee number, last name, first name, and department name.

```

select e.emp_no, last_name, first_name, d.dept_name from employees e
inner join department_employees de on e.emp_no=de.emp_no
inner join departments d on de.dept_no=d.dept_no;

```

```

33 select e.emp_no, last_name, first_name, d.dept_name from employees e
34 inner join department_employees de on e.emp_no=de.emp_no
35 inner join departments d on de.dept_no=d.dept_no;

```

```

INFO : Completed executing command(queryId=hive_20220517202518_0eb6730a-f144-45ce-8003-c1f792a08che): Time taken: 45.857 seconds
INFO : OK

```

Query History Saved Queries Results (331,603)

	e.emp_no	last_name	first_name	d.dept_name
1	10001	Facello	Georgi	"development"
2	10002	Simmel	Bezalel	"Sales"
3	10003	Bamford	Parto	"Production"
4	10004	Koblick	Chirstian	"Production"
5	10005	Maliniak	Kyoichi	"Human Resources"
6	10006	Preusig	Anneke	"development"
7	10007	Zielinski	Tzvetan	"Research"
8	10008	Kalloufi	Saniya	"development"

5. A list showing first name, last name, and sex for employees whose first name is "Hercules" and last names begin with "B."

```
select first_name, last_name, sex from employees  
where first_name='Hercules' and last_name like 'B%';
```

```
30 select first_name, last_name, sex from employees  
31 where first_name='Hercules' and last_name like 'B%';
```

```
INFO : Completed executing command(queryId=hive_20220517201729_bdc9d6d5-981f-4f4f-bc07-61257963944f): Time taken: 22.412 seconds  
INFO : OK
```

Query History

Saved Queries

Results (20)

	first_name	last_name	sex
1	Hercules	Benzmuller	M
2	Hercules	Brendel	F
3	Hercules	Baranowski	M
4	Hercules	Barreiro	M
5	Hercules	Baer	M
6	Hercules	Bernardinello	F
7	Hercules	Basagni	M
8	Hercules	Biran	F

6. A list showing all employees in the Sales department, including their employee number, last name, first name, and department name.

```
create table department as  
select dept_no, substr(dept_name, 2, length(dept_name)-2) as dept_name from departments;
```

```
select e.emp_no, last_name, first_name, d.dept_name from employees e  
inner join department_employees de on e.emp_no=de.emp_no  
inner join department d on de.dept_no=d.dept_no  
where d.dept_name = 'Sales';
```

▼

job_1652166004796_4390



- 7.**

```
select e.emp_no, last_name, first_name, d.dept_name from employees e
inner join department_employees de on e.emp_no=de.emp_no
inner join department d on de.dept_no=d.dept_no
where d.dept_name IN ('Sales', 'development');
```


job_1652166004796_4392



8. A list showing the frequency count of employee last names, in descending order. (i.e., how many employees share each last name

```
select last_name, count(*) as count_of_employee_last_name from employees
group by last_name
order by count_of_employee_last_name desc;
```

```
41 select last_name, count(*) as count_of_employee_last_name from employees
42 group by last_name
43 order by count_of_employee_last_name desc;
```

INFO : Completed executing command(queryId=hive_20220517210749_8fa439f4-98ed-405f-bff5-70e5a90940ed): Time taken: 57.399 seconds
INFO : OK

Query History		Saved Queries		Results (1,638)	
	last_name			count_of_employee_last_name	
1	Baba			226	
2	Coorg			223	
3	Gelosh			223	
4	Sudbeck			222	
5	Farris			222	
6	Adachi			221	
7	Osgood			220	
8	Masada			218	
9	Neiman			218	
10	Mandell			218	
11	Wendorf			217	

9. Histogram to show the salary distribution among the employees

```
select
cast(hist.x as int) as bin_center,
cast(hist.y as bigint) as bin_height
from
(select
histogram_numeric(salary, 20) as A_hist
from
salaries) t
lateral view explode(A_hist) exploded_table as hist;
```

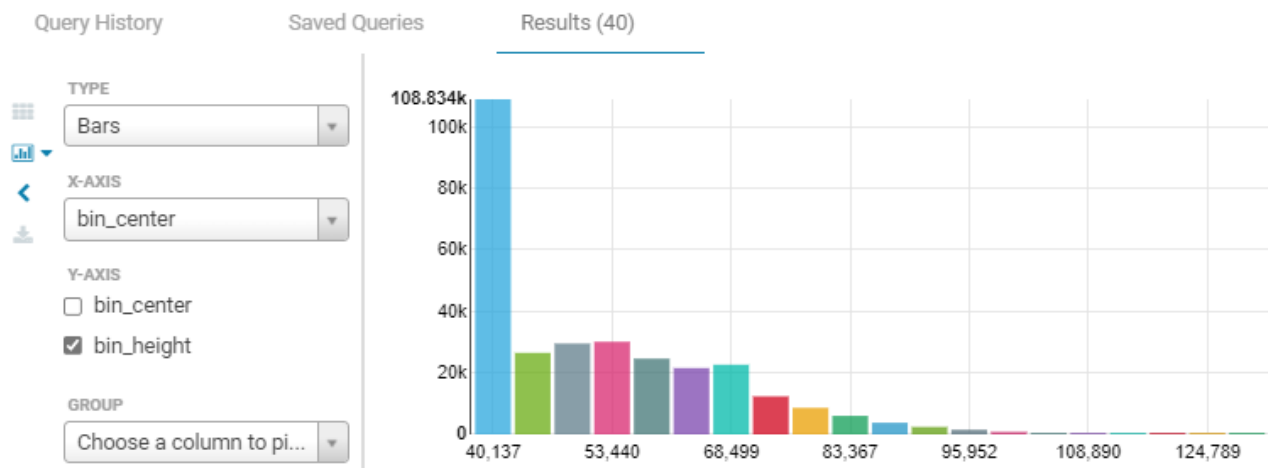


```

54 select
55 cast(hist.x as int) as bin_center,
56 cast(hist.y as bigint) as bin_height
57 from
58 (select
59 histogram_numeric(salary, 20) as A_hist
60 from
61 salaries) t
62 lateral view explode(A_hist) exploded_table as hist;

```

INFO : Completed executing command(queryId=hive_20220517214038_db6470bd-6d25-4056-b870-c83d5e1b8e9d): Time taken: 28.055 seconds
 INFO : OK

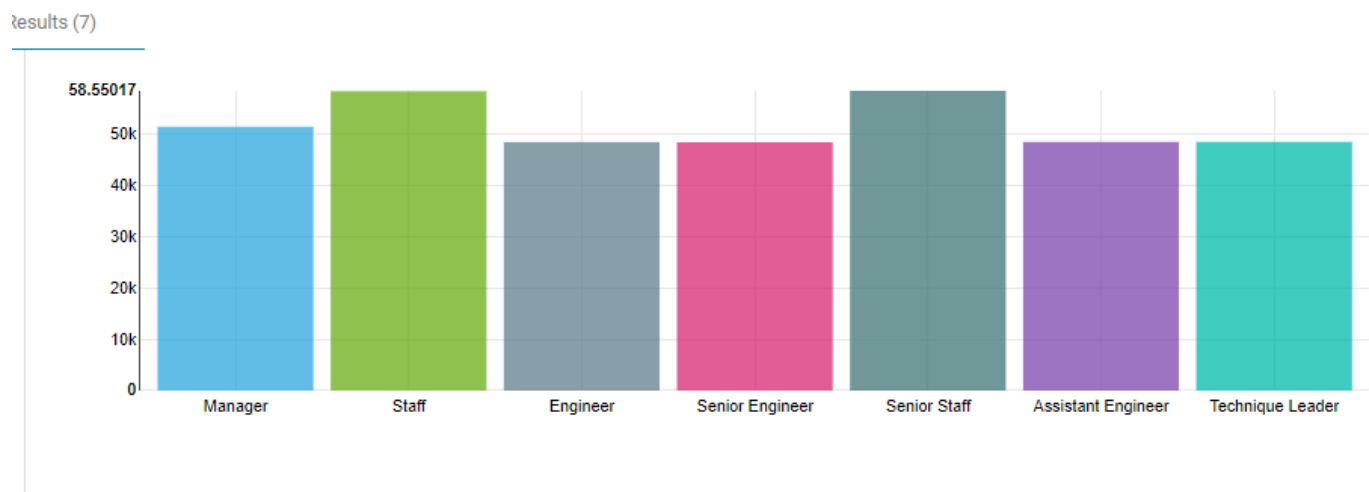


10. Bar graph to show the Average salary per title (designation)

```

select t.title, avg(s.salary) as avg_salary from titles t
inner join employees e on t.title_id = e.emp_titles_id
inner join salaries s on e.emp_no = s.emp_no
group by t.title;

```




11. Calculate employee tenure & show the tenure distribution among the employees

12. a) Count the number of employee's left and not left the company.

```
select left_company, count(*) from employees
group by left_company;
```

```
64 select left_company, count(*) from employees
65 group by left_company;
```

```
INFO : Completed executing command(queryId=hive_20220518054455_41cbf1c9-588e-4e3c-9bf9-bc113bc6e099): Time taken: 29.204
nds
INFO : OK
```





Query History		Saved Queries		Results (2)	
	<input checked="" type="checkbox"/>	COLUMNS (3) Q		left_company	_c1
	<input checked="" type="checkbox"/>	left_company	book	1 false	270157
	<input checked="" type="checkbox"/>	_c1	begin	2 true	29867

12. b) how many total employees per title in the company

```
select t.title, count(e.emp_no) as total_employee_per_title from titles t
inner join employees e on t.title_id = e.emp_titles_id
group by t.title;
```

```
131 --#12.b) how many total employees per title in the company
132
133 select t.title, count(e.emp_no) as total_employee_per_title from titles t
134 inner join employees e on t.title_id = e.emp_titles_id
135 group by t.title;#12.c) Total no employees per department in the company
136
```

```
INFO : Completed executing command(queryId=hive_20220520092943_f48ba02b-6c2f-4d78-82fb-d173245a8418): Time taken: 46.491 sec
nds
INFO : OK
```

Query History		Saved Queries		Results (7)	
		t.title		total_employee_per_title	
		1 Assistant Engineer		5835	
		2 Engineer		47303	
		3 Manager		24	
		4 Senior Engineer		97747	
		5 Senior Staff		26583	
		6 Staff		107384	
		7 Technique Leader		15148	

12.c) Total no employees per department in the company

```
select d.dept_name, count(e.emp_no) as count_of_employee_per_department from employees e
inner join project_de.department_employees de on e.emp_no=de.emp_no
inner join project_de.department d on de.dept_no=d.dept_no
group by dept_name order by count_of_employee_per_department desc;
```

```
137 --#12.c) Total no employees per department in the company
138
139 select d.dept_name, count(e.emp_no) as count_of_employee_per_department from employees e
140 inner join project_de.department_employees de on e.emp_no=de.emp_no
141 inner join project_de.department d on de.dept_no=d.dept_no
142 group by dept_name order by count_of_employee_per_department desc;
```

```
INFO : Completed executing command(queryId=hive_20220520093241_3a8fc16e-b033-4212-a5cb-4b4180099724): Time taken
ds
INFO : OK
```

[job_16530058](#)

[job_16530058](#)

Query History

Saved Queries

Results (9)

	d.dept_name	count_of_employee_per_department
1	development	85707
2	Production	73485
3	Sales	52245
4	Customer Service	23580
5	Research	21126
6	Marketing	20211
7	Quality Management	20117
8	Human Resources	17786
9	Finance	17346

12.d) top 3 department where employees are leaving the company

```
select d.dept_name, count(e.emp_no) as total_no_of_employees_left from project_de.employees e
inner join project_de.department_employees de on e.emp_no=de.emp_no
inner join project_de.department d on de.dept_no=d.dept_no
where left_company = "true" group by dept_name order by total_no_of_employees_left desc;
```

12. e) Create bins of Salary to show the frequency of number of employees in each salary group.

Create table salary_dist

select

case

when s.salary >= 40000 and s.salary < 50000 then '40-50k'

when s.salary >= 50000 and s.salary < 60000 then '50 -60k'

when s.salary >= 60000 and s.salary < 70000 then '60 -70k'

when s.salary >= 70000 and s.salary < 80000 then '70 -80k'

when s.salary >= 80000 and s.salary < 90000 then '80 -90k'

when s.salary >= 90000 and s.salary < 100000 then '90 -100k'

when s.salary >= 100000 then '100k+'

end as Salary_bins, e.emp_no

from employees e

inner join salaries s on e.emp_no = s.emp_no;

select Salary_bins , count(emp_no) freq from salary_dist

group by Salary_bins;

```
53 select Salary_bins , count(emp_no) freq from salary_dist
54 group by Salary_bins;
55
```

INFO : Completed executing command(queryId=hive_20220519121057_4417f47b-d83e-4311-ae22-59f24660811a): Time taken: 191.888 sec
onds
INFO : OK

Query History

Saved Queries

Results (7)

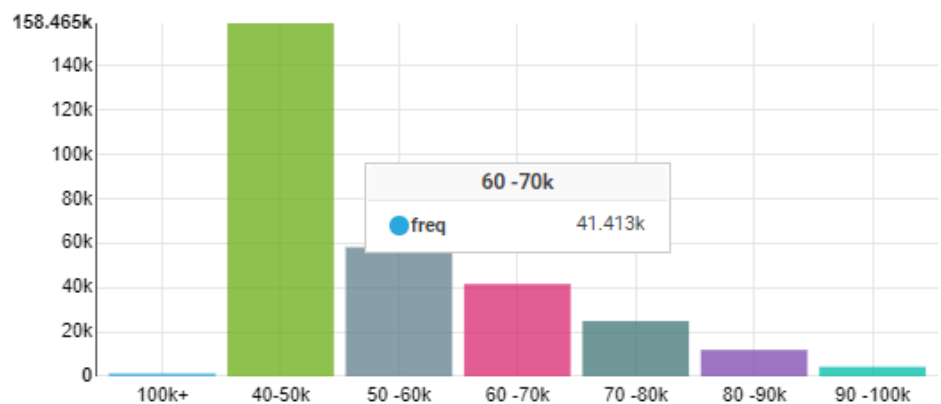
TYPE
Bars

X-AXIS
salary_bins

Y-AXIS
☒ freq

GROUP
Choose a column to pi...

LIMIT



12.f) list of emp_name, title, dept_name, salary for each employee

```
select concat(first_name," ",last_name) as name, title, dept_name, salary from project_de.employees e
inner join project_de.salaries s on e.emp_no=s.emp_no
inner join project_de.titles t on e.emp_titles_id=t.title_id
inner join project_de.department_employees de on e.emp_no=de.emp_no
inner join project_de.department d on de.dept_no=d.dept_no;
```

```
151 #12 f) list of emp_name, title, dept_name, salary for each employee
152
153 select concat(first_name," ",last_name) as name, title, dept_name, salary from project_de.employees e
154 inner join project_de.salaries s on e.emp_no=s.emp_no
155 inner join project_de.titles t on e.emp_titles_id=t.title_id
156 inner join project_de.department_employees de on e.emp_no=de.emp_no
157 inner join project_de.department d on de.dept_no=d.dept_no;
```

```
INFO : Total MapReduce CPU Time Spent: 16 seconds 500 msec
INFO : Completed executing command(queryId=hive_20220520093725_48477faf-035b-4fc5-8a3a-c468efe31e3:job_1653005874090_0447)
nds
INFO : OK
```

Query History Saved Queries Results (331,603)

	name	title	dept_name	salary
1	Georgi Facello	Senior Engineer	development	60117
2	Bezalel Simmel	Staff	Sales	65828
3	Parto Bamford	Senior Engineer	Production	40006
4	Chirstian Koblick	Senior Engineer	Production	40054
5	Kyoichi Maliniak	Staff	Human Resources	78228
6	Anneke Preusig	Senior Engineer	development	40000
7	Tzvetan Zielinski	Staff	Research	56724

EDA outputs in SPARK

```
In [1]: from pyspark.sql import SparkSession
```

```
In [2]: spark=(SparkSession.builder.master("local").appName("Capstone Project")\
        .config("hive.metastore.uris","thrift://ip-10-1-2-24.ap-south-1.compute.internal:9083")\
        .enableHiveSupport().getOrCreate())
        spark
```

```
Out[2]: SparkSession - hive
        SparkContext
```

[Spark UI](#)

Version

v2.4.0

Master

local

AppName

Capstone Project

1. EDA

```
In [5]: # importing all tables
        employees=spark.sql("select * from project_de.employees")
        titles=spark.sql("select * from project_de.titles")
        salaries=spark.sql("select * from project_de.salaries")
        departments=spark.sql("select * from project_de.department")
        department_manager=spark.sql("select * from project_de.department_manager")
        department_employees=spark.sql("select * from project_de.department_employees")
```

```
In [4]: #1 List showing employee number, Last name, first name, sex, and salary for each employee
        spark.sql('select e.emp_no, last_name, first_name, sex, salary from employees e inner join salaries s on e.emp_no=s.emp_no').show
```

emp_no	last_name	first_name	sex	salary
10001	Facello	Georgi	M	60117
10002	Simmel	Bezalel	F	65828
10003	Bamford	Parto	M	40006
10004	Koblick	Chirstian	M	40054
10005	Maliniak	Kyoichi	M	78228
10006	Preusig	Anneke	F	40000
10007	Zielinski	Tzvetan	F	56724
10008	Kalloufi	Saniya	M	46671
10009	Peac	Sumant	F	60929
10010	Piveteau	Duangkaew	F	72488
10011	Sluis	Mary	F	42365
10012	Bridgland	Patricio	M	40000
10013	Terkki	Eberhardt	M	40000
10014	Genin	Berni	M	46168
10015	Nooteboom	Guoxiang	M	40000
10016	Cappelletti	Kazuhito	M	70889
10017	Bouloucos	Cristinel	F	71380
10018	Peha	Kazuhide	F	55881
10019	Haddadi	Lillian	M	44276
10020	Warwick	Mayuko	M	40000

only showing top 20 rows

In [7]: #2. A List showing first name, last name, and hire date for employees who were hired in 1986.

```
spark.sql('select first_name, last_name, hire_date from employees \
where cast(substring_index(hire_date,"",-1) as int ) = 1986').show()
```

```
+-----+-----+-----+
|first_name|  last_name| hire_date|
+-----+-----+-----+
|   Georgi|   Facello| 6/26/1986|
|   Parto|   Bamford| 8/28/1986|
| Chirstian|   Koblick| 12/1/1986|
|   Sanjiv|   Zschoche| 2/4/1986|
|   Kwee|   Schusler| 2/26/1986|
| Kshitiij|     Gils| 3/27/1986|
| Zhongwei|   Rosen| 10/30/1986|
| Xinglin|   Eugenio| 9/8/1986|
| Sudharsan|Flasterstein| 8/12/1986|
|   Kendra|   Hofting| 3/14/1986|
|   Hilari|   Morton| 7/15/1986|
|   Akemi|   Birch| 12/2/1986|
|   Lunjin|   Giveon| 10/2/1986|
|   Xuejia|   Ullian| 8/22/1986|
|   Chikara|   Rissland| 1/23/1986|
| Domenick|   Peltason| 3/14/1986|
|   Zissis|   Pintelas| 2/11/1986|
|   Perry|   Shimshoni| 9/18/1986|
| Kazuhito| Encarnacion| 8/21/1986|
|   Xiadong|   Perry| 11/5/1986|
+-----+-----+-----+
only showing top 20 rows
```

In [9]: #3. A List showing the manager of each department with the following information: department number, department name, # the manager's employee number, Last name, first name.

```
spark.sql('select d.dept_no, d.dept_name, dm.emp_no, last_name, first_name from departments d \
inner join project_de.department_manager dm on d.dept_no = dm.dept_no \
inner join project_de.employees e on e.emp_no = dm.emp_no').show()
```

dept_no	dept_name	emp_no	last_name	first_name
d001	"Marketing"	110022	Markovitch	Margareta
d001	"Marketing"	110039	Minakawa	Vishwani
d002	"Finance"	110085	Alpin	Ebru
d002	"Finance"	110114	Legleitner	Isamu
d003	"Human Resources"	110183	Ossenbruggen	Shirish
d003	"Human Resources"	110228	Sigstam	Karsten
d004	"Production"	110303	Wegerle	Krassimir
d004	"Production"	110344	Cools	Rosine
d004	"Production"	110386	Kieras	Shem
d004	"Production"	110420	Ghazalie	Oscar
d005	"development"	110511	Hagimont	DeForest
d005	"development"	110567	DasSarma	Leon
d006	"Quality Management"	110725	Onuegbu	Peternela
d006	"Quality Management"	110765	Hofmeyr	Rutger
d006	"Quality Management"	110800	Quadeer	Sanjoy
d006	"Quality Management"	110854	Pesch	Dung
d007	"Sales"	111035	Kaelbling	Przemyslaw
d007	"Sales"	111133	Zhang	Hauke
d008	"Research"	111400	Staelin	Arie
d008	"Research"	111534	Kambil	Hilary

only showing top 20 rows

In [11]: #4. A List showing the department of each employee with the following information: employee number, Last name, first name, and department name.

```
spark.sql('select e.emp_no, last_name, first_name, d.dept_name from employees e \
inner join project_de.department_employees de on e.emp_no=de.emp_no \
inner join project_de.departments d on de.dept_no=d.dept_no').show()
```

emp_no	last_name	first_name	dept_name
10001	Facello	Georgi	"development"
10002	Simmel	Bezalel	"Sales"
10003	Bamford	Parto	"Production"
10004	Koblick	Chirstian	"Production"
10005	Maliniak	Kyoichi	"Human Resources"
10006	Preusig	Anneke	"development"
10007	Zielinski	Tzvetan	"Research"
10008	Kalloufi	Saniya	"development"
10009	Peac	Sumant	"Quality Management"
10010	Piveteau	Duangkaew	"Quality Management"
10010	Piveteau	Duangkaew	"Production"
10011	Sluis	Mary	"Customer Service"
10012	Bridgland	Patricio	"development"
10013	Terkki	Eberhardt	"Human Resources"
10014	Genin	Berni	"development"
10015	Nooteboom	Guoxiang	"Research"
10016	Cappelletti	Kazuhide	"Sales"
10017	Bouloucos	Cristinel	"Marketing"
10018	Peha	Kazuhide	"development"
10018	Peha	Kazuhide	"Production"

only showing top 20 rows

In [16]: #5. A List showing first name, last name, and sex for employees whose first name is "Hercules" and last names begin with "B."

```
spark.sql('select first_name, last_name, sex from employees where first_name="Hercules" and last_name like "B%").show()
```

first_name	last_name	sex
Hercules	Benzmuller	M
Hercules	Brendel	F
Hercules	Baranowski	M
Hercules	Barreiro	M
Hercules	Baer	M
Hercules	Bernardinello	F
Hercules	Basagni	M
Hercules	Biran	F
Hercules	Bernatsky	M
Hercules	Bail	F
Hercules	Birge	F
Hercules	Bisiani	F
Hercules	Bodoff	M
Hercules	Biron	F
Hercules	Buchter	M
Hercules	Bain	F
Hercules	Bahr	M
Hercules	Baak	M
Hercules	Benantar	F
Hercules	Berstel	F

In [18]: #6. A List showing all employees in the Sales department, including their employee number, last name, first name, and # department name.

```
spark.sql('select e.emp_no, last_name, first_name, d.dept_name from employees e \
inner join project_de.department_employees de on e.emp_no=de.emp_no \
inner join project_de.department d on de.dept_no=d.dept_no \
where d.dept_name ="Sales").show()
```

emp_no	last_name	first_name	dept_name
10002	Simmel	Bezalel	Sales
10016	Cappelletti	Kazuhiro	Sales
10034	Swan	Bader	Sales
10041	Lenart	Uri	Sales
10050	Dredge	Yinghua	Sales
10053	Zschoche	Sanjiv	Sales
10060	Billingsley	Breanna	Sales
10061	Herber	Tse	Sales
10068	Brattka	Charlene	Sales
10087	Eugenio	Xinglin	Sales
10088	Syrzycki	Jungsoon	Sales
10089	Flasterstein	Sudharsan	Sales
10093	Desikan	Sailaja	Sales
10095	Morton	Hilari	Sales
10099	Sullins	Valter	Sales
10101	Heyers	Perla	Sales
10107	Baca	Dung	Sales
10125	Hiltgen	Syozo	Sales
10136	Pintelas	Zissis	Sales
10148	Azumi	Douadi	Sales

only showing top 20 rows

In [19]: #7.A List showing all employees in the Sales and Development departments, including their employee number, Last name, # first name, and department name.

```
spark.sql('select e.emp_no, last_name, first_name, d.dept_name from employees e \
inner join project_de.department_employees de on e.emp_no=de.emp_no \
inner join project_de.department d on de.dept_no=d.dept_no \
where d.dept_name IN ("Sales", "development")').show()
```

emp_no	last_name	first_name	dept_name
10001	Facello	Georgi	development
10002	Simmel	Bezalel	Sales
10006	Preusig	Anneke	development
10008	Kalloufi	Saniya	development
10012	Bridgland	Patricio	development
10014	Genin	Berni	development
10016	Cappelletti	Kazuhito	Sales
10018	Peha	Kazuhide	development
10021	Erde	Ramzi	development
10022	Famili	Shahaf	development
10023	Montemayor	Bojan	development
10025	Heyers	Prasadram	development
10027	Reistad	Divier	development
10028	Tempesti	Domenick	development
10031	Joslin	Karsten	development
10034	Swan	Bader	Sales
10037	Makrucki	Pradeep	development
10040	Meriste	Weiyi	development
10041	Lenart	Uri	Sales
10043	Tzvieli	Yishay	development

only showing top 20 rows

In [20]: #8. A List showing the frequency count of employee Last names, in descending order. (i.e., how many employees share each Last name)

```
spark.sql('select last_name, count(*) as count_of_employee_last_name from employees \
group by last_name \
order by count_of_employee_last_name desc').show()
```

last_name	count_of_employee_last_name
Baba	226
Gelosh	223
Coorg	223
Sudbeck	222
Farris	222
Adachi	221
Osgood	220
Neiman	218
Mandell	218
Masada	218
Boudaillier	217
Wendorf	217
Mahnke	216
Solares	216
Pettis	216
Cummings	216
Emmart	215
Kulisch	215
Birjandi	215
Maksimenko	215

only showing top 20 rows

In [25]: #9. Histogram to show the salary distribution among the employees

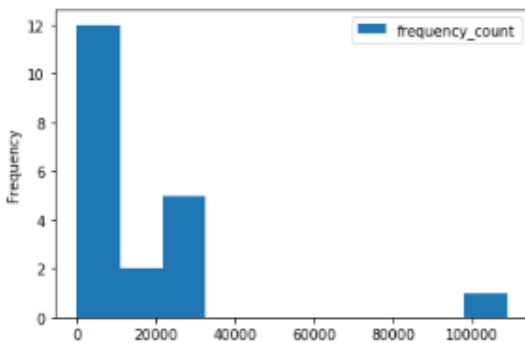
```
salary_histogram=spark.sql('select cast(hist.x as int) as salary, cast(hist.y as bigint) as frequency_count from \
(select histogram_numeric(salary, 20) as A_hist \
from salaries) t \
lateral view explode(A_hist) exploded_table as hist')
salary_histogram.show()
```

```
+-----+frequency_count+
|salary|frequency_count|
+-----+-----+
| 40137|          108834|
| 44331|          26476|
| 48695|          29510|
| 53440|          30087|
| 58232|          24609|
| 62954|          21554|
| 68499|          22627|
| 74140|          12348|
| 78817|           8624|
| 83367|           6011|
| 87753|           3774|
| 91865|           2417|
| 95952|           1506|
|100321|            817|
|104612|            432|
|108890|            234|
|113916|            116|
|119213|             38|
|124789|              9|
|129492|              1|
+-----+-----+
```

```
In [43]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

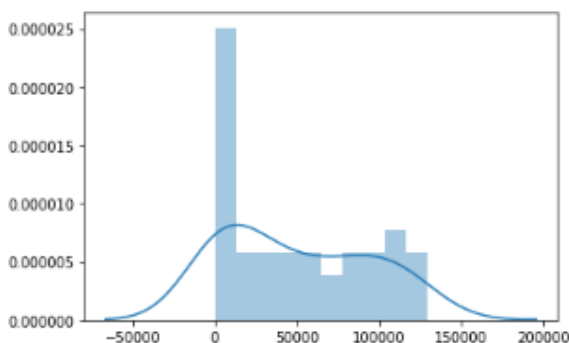
```
In [42]: salary_histogram.toPandas().plot(x='salary', y='frequency_count', kind = 'hist')
```

Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9f3a717050>



```
In [85]: sns.distplot(salary_histogram.toPandas(), bins = 10, kde=True)
```

Out[85]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9f3a2659d0>



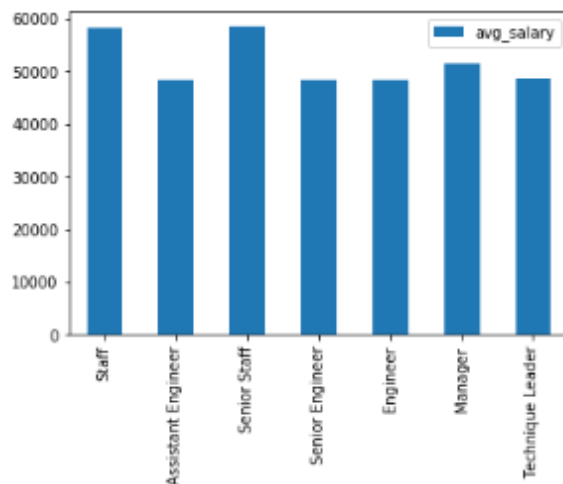
In [39]: #10. Bar graph to show the Average salary per title (designation)

```
avg_bar=spark.sql('select t.title, avg(s.salary) as avg_salary from titles t \
inner join employees e on t.title_id = e.emp_titles_id \
inner join salaries s on e.emp_no = s.emp_no \
group by t.title')
avg_bar.show()
```

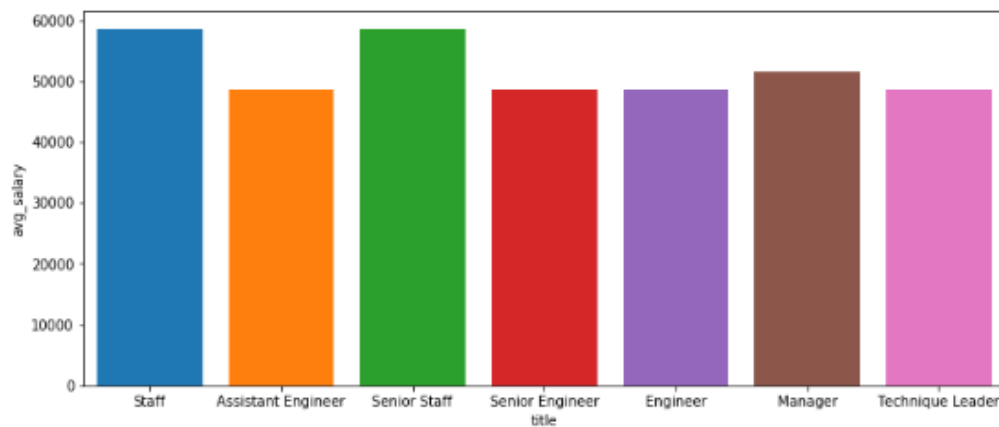
title	avg_salary
Staff	58465.38285033152
Assistant Engineer	48564.43444730077
Senior Staff	58550.17270435993
Senior Engineer	48506.79987109579
Engineer	48535.336511426336
Manager	51531.041666666664
Technique Leader	48582.89609189332

In [52]: avg_bar.toPandas().plot(x='title', y='avg_salary', kind='bar')

Out[52]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9f3aecfc50>



In [97]: plt.figure(figsize=(12,5))
sns.barplot(data=avg_bar.toPandas(), x='title', y='avg_salary', saturation=1)
plt.show()



In [55]: #12.a) how many employee have left the company

```
spark.sql('select left_company, count(*) as employees_left_company from project_de.employees group by left_company').show()
```

left_company	employees_left_company
true	29867
false	270157

In [56]: #12.b) how many total employees per title in the company

```
spark.sql('select t.title, count(e.emp_no) as total_employee_per_title from titles t \
inner join employees e on t.title_id = e.emp_titles_id \
group by t.title').show()
```

title	total_employee_per_title
Staff	107384
Assistant Engineer	5835
Senior Staff	26583
Senior Engineer	97747
Engineer	47303
Manager	24
Technique Leader	15148

In [104]: #12.c) Total no employees per department in the company

```
spark.sql('select d.dept_name, count(e.emp_no) as count_of_employee_per_department from employees e \
inner join project_de.department_employees de on e.emp_no=de.emp_no \
inner join project_de.department d on de.dept_no=d.dept_no \
group by dept_name order by count_of_employee_per_department desc').show()
```

dept_name	count_of_employee_per_department
development	85707
Production	73485
Sales	52245
Customer Service	23580
Research	21126
Marketing	20211
Quality Management	20117
Human Resources	17786
Finance	17346

In [114]: # 12 c i) with Left_company

```
spark.sql('select d.dept_name,left_company, count(e.emp_no) as count_of_employee_per_department from project_de.employees e \
inner join project_de.department_employees de on e.emp_no=de.emp_no \
inner join project_de.department d on de.dept_no=d.dept_no \
group by dept_name, left_company order by dept_name, left_company').show()
```

dept_name	left_company	count_of_employee_per_department
Customer Service	false	21166
Customer Service	true	2414
Finance	false	15699
Finance	true	1647
Human Resources	false	15989
Human Resources	true	1797
Marketing	false	18270
Marketing	true	1941
Production	false	66096
Production	true	7389
Quality Management	false	18099
Quality Management	true	2018
Research	false	19028
Research	true	2098
Sales	false	47036
Sales	true	5209

In [103]: #12.d) top 3 department where employees are Leaving the company

```
spark.sql('select d.dept_name, count(e.emp_no) as total_no_of_employees_left from project_de.employees e \
inner join project_de.department_employees de on e.emp_no=de.emp_no \
inner join project_de.department d on de.dept_no=d.dept_no \
where left_company = "true" group by dept_name order by total_no_of_employees_left desc').show()
```

dept_name	total_no_of_employees_left
development	8508
Production	7389
Sales	5209
Customer Service	2414
Research	2098
Quality Management	2018
Marketing	1941
Human Resources	1797
Finance	1647

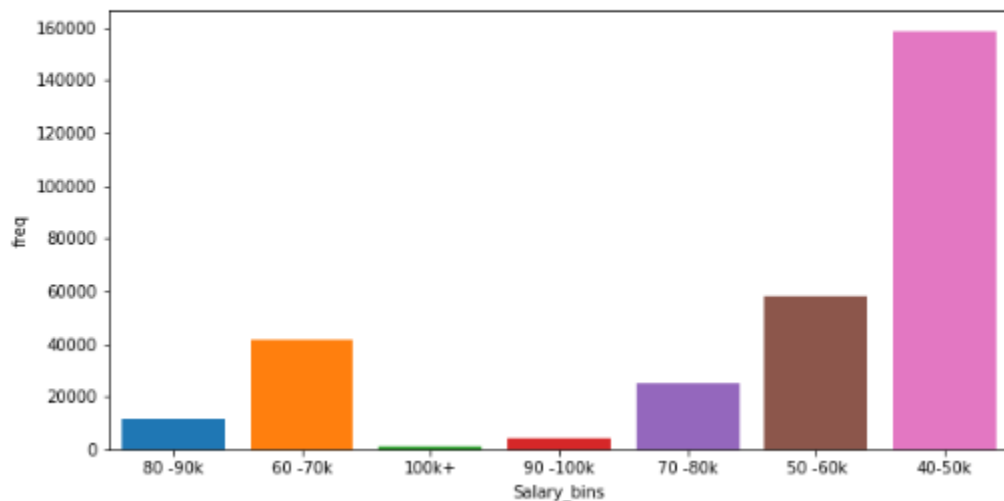
In [71]: #12. e)

```
salary_dist = spark.sql('select Salary_bins , count(emp_no) freq from project_de.salary_dist group by Salary_bins')
salary_dist.show()
```

Salary_bins	freq
80 -90k	11845
60 -70k	41413
100k+	1288
90 -100k	4284
70 -80k	24814
50 -60k	57915
40-50k	158465

In [96]: plt.figure(figsize=(10,5))

```
sns.barplot(data=salary_dist.toPandas(), x='Salary_bins', y='freq', saturation=1)
plt.show()
```



In [123]: #12 f) List of emp_name, title, dept_name, salary for each employee

```
spark.sql('select concat(first_name," ",last_name) as name, title, dept_name, salary from project_de.employees e \
inner join project_de.salaries s on e.emp_no=s.emp_no \
inner join project_de.titles t on e.emp_titles_id=t.title_id \
inner join project_de.department_employees de on e.emp_no=de.emp_no \
inner join project_de.department d on de.dept_no=d.dept_no').show()
```

name	title	dept_name	salary
Georgi Facello	Senior Engineer	development	60117
Bezalel Simmel	Staff	Sales	65828
Parto Bamford	Senior Engineer	Production	40006
Chirstian Koblick	Senior Engineer	Production	40054
Kyoichi Maliniak	Staff	Human Resources	78228
Anneke Preusig	Senior Engineer	development	40000
Tzvetan Zielinski	Staff	Research	56724
Saniya Kalloufi	Assistant Engineer	development	46671
Sumant Peac	Senior Engineer	Quality Management	60929
Duangkaew Piveteau	Engineer	Quality Management	72488
Duangkaew Piveteau	Engineer	Production	72488
Mary Sluis	Staff	Customer Service	42365
Patricio Bridgland	Senior Engineer	development	40000
Eberhardt Terkki	Senior Staff	Human Resources	40000
Berni Genin	Engineer	development	46168
Guoxiang Nooteboom	Senior Staff	Research	40000
Kazuhiro Cappelletti	Staff	Sales	70889
Cristinel Bouloucos	Staff	Marketing	71380
Kazuhide Peha	Senior Engineer	development	55881
Kazuhide Peha	Senior Engineer	Production	55881

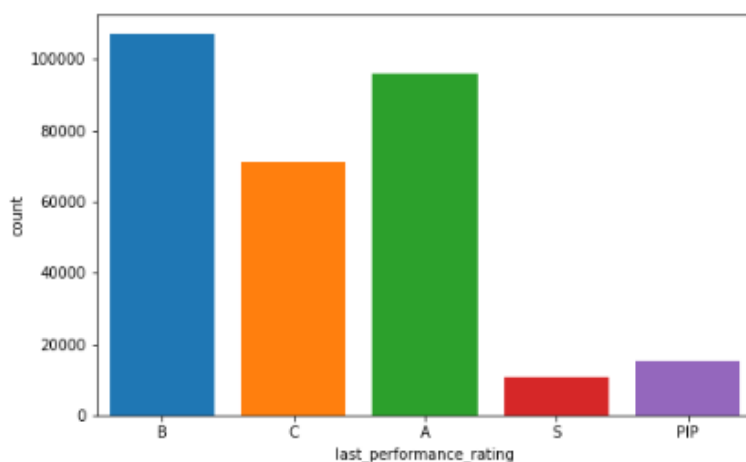
only showing top 20 rows

In [132]: #12 g) performance wise count of employee at Last year s being the top and PIP at Last

```
performance=employees.groupBy('last_performance_rating').count()
performance.show()
```

last_performance_rating	count
B	107154
C	71304
A	95919
S	10542
PIP	15105

```
plt.figure(figsize=(8,5))
sns.barplot(data=performance.toPandas(), x='last_performance_rating', y='count', saturation=1)
plt.show()
```



Build ML Model : - Classification Model

```
In [4]: # joining all tables and storing it as data
data=spark.sql('select * from project_de.employees e \
inner join project_de.salaries s on e.emp_no=s.emp_no \
inner join project_de.titles t on e.emp_titles_id=t.title_id \
inner join project_de.department_employees de on e.emp_no=de.emp_no \
inner join project_de.department d on de.dept_no=d.dept_no')
```

```
In [7]: type(data)
```

```
Out[7]: pyspark.sql.dataframe.DataFrame
```

```
In [11]: data.count()
```

```
Out[11]: 331603
```

```
In [14]: data.columns
```

```
Out[14]: ['emp_no',
'emp_titles_id',
'birth_date',
'first_name',
'last_name',
'sex',
'hire_date',
'no_of_projects',
'last_performance_rating',
'left_company',
'last_date',
'emp_no',
'salary',
'title_id',
'title',
'emp_no',
'dept_no',
'dept_no',
'dept_name']
```

```
In [52]: #information abt the dataset
data1.printSchema()
```

```
root
|-- emp_no: integer (nullable = true)
|-- emp_titles_id: string (nullable = true)
|-- birth_date: string (nullable = true)
|-- first_name: string (nullable = true)
|-- last_name: string (nullable = true)
|-- sex: string (nullable = true)
|-- hire_date: string (nullable = true)
|-- no_of_projects: integer (nullable = true)
|-- last_performance_rating: string (nullable = true)
|-- left_company: boolean (nullable = true)
|-- last_date: string (nullable = true)
|-- salary: integer (nullable = true)
|-- title: string (nullable = true)
|-- dept_no: string (nullable = true)
|-- dept_name: string (nullable = true)
```

converting the string dates to datetime

```
: # selecting dates from data1 in b and converting to pandas dataframe  
b=data1.select('emp_no', 'birth_date', 'hire_date', 'last_date').toPandas()
```

```
: b.head()
```

```
:  
   emp_no  birth_date  hire_date  last_date  
0    10001    9/2/1953    6/26/1986    7/30/1994  
1    10002    6/2/1964    11/21/1985         \r  
2    10003   12/3/1959    8/28/1986         \r  
3    10004    5/1/1954   12/1/1986         \r  
4    10005   1/21/1955    9/12/1989         \r
```

```
: b.birth_date.head(2)
```

```
: 0    9/2/1953  
1    6/2/1964  
Name: birth_date, dtype: object
```

```
: # converting the string date to datetime  
b.birth_date=pd.to_datetime(b.birth_date)  
b.hire_date=pd.to_datetime(b.hire_date)
```

```
: b.dtypes
```

```
: emp_no          int32  
birth_date      datetime64[ns]  
hire_date       datetime64[ns]  
last_date       object  
dtype: object
```

```
In [106]: b['last_date']=pd.to_datetime(b.last_date[b.last_date!="\r"])
```

```
In [108]: b.last_date.head(4)
```

```
Out[108]: 0    1994-07-30
          1         NaT
          2         NaT
          3         NaT
          Name: last_date, dtype: datetime64[ns]
```

```
In [109]: b.dtypes
```

```
Out[109]: emp_no          int32
          birth_date    datetime64[ns]
          hire_date     datetime64[ns]
          last_date     datetime64[ns]
          dtype: object
```

```
In [110]: b.head(4)
```

```
Out[110]:
```

	emp_no	birth_date	hire_date	last_date
0	10001	1953-09-02	1986-06-26	1994-07-30
1	10002	1964-06-02	1985-11-21	NaT
2	10003	1959-12-03	1986-08-28	NaT
3	10004	1954-05-01	1986-12-01	NaT

```
In [112]: #creating pandas dataframe to spark dataframe
          bdf=spark.createDataFrame(b)
```

```
In [113]: bdf.show(4)
```

```
+-----+-----+-----+-----+
|emp_no|    birth_date|    hire_date|    last_date|
+-----+-----+-----+-----+
| 10001|1953-09-02 00:00:00|1986-06-26 00:00:00|1994-07-30 00:00:00|
| 10002|1964-06-02 00:00:00|1985-11-21 00:00:00|              null|
| 10003|1959-12-03 00:00:00|1986-08-28 00:00:00|              null|
| 10004|1954-05-01 00:00:00|1986-12-01 00:00:00|              null|
+-----+-----+-----+-----+
only showing top 4 rows
```

Logistic Regression

```
In [115]: bdf.dtypes
```

```
Out[115]: [('emp_no', 'bigint'),  
          ('birth_date', 'timestamp'),  
          ('hire_date', 'timestamp'),  
          ('last_date', 'timestamp')]
```

```
In [126]: #now joining data1 and bdf on emp_no column and assigning into data12
```

```
data11=data1.drop('birth_date', 'hire_date', 'last_date')  
data12=data11.join(bdf, on='emp_no', how='inner' )
```

```
In [127]: #this is our correct table with all variable being correctly represented by their datatypes  
data12.dtypes
```

```
Out[127]: [('emp_no', 'int'),  
          ('emp_titles_id', 'string'),  
          ('first_name', 'string'),  
          ('last_name', 'string'),  
          ('sex', 'string'),  
          ('no_of_projects', 'int'),  
          ('last_performance_rating', 'string'),  
          ('left_company', 'boolean'),  
          ('salary', 'int'),  
          ('title', 'string'),  
          ('dept_no', 'string'),  
          ('dept_name', 'string'),  
          ('birth_date', 'timestamp'),  
          ('hire_date', 'timestamp'),  
          ('last_date', 'timestamp')]
```

```
In [129]: data12.count()
```

```
Out[129]: 394761
```

```
In [190]: #Keep a copy of the original dataframe for later use  
datafinal=data12
```

```
In [191]: #dropping irrelevant columns  
datafinal=datafinal.drop('emp_no','first_name','last_name')
```

```
In [192]: datafinal.dtypes
```

```
Out[192]: [('emp_titles_id', 'string'),  
          ('sex', 'string'),  
          ('no_of_projects', 'int'),  
          ('last_performance_rating', 'string'),  
          ('left_company', 'boolean'),  
          ('salary', 'int'),  
          ('title', 'string'),  
          ('dept_no', 'string'),  
          ('dept_name', 'string'),  
          ('birth_date', 'timestamp'),  
          ('hire_date', 'timestamp'),  
          ('last_date', 'timestamp')]
```

```
In [148]: from pyspark.sql import functions as F
```

```
In [193]: #converting Left_company boolean to int  
datafinal = datafinal.withColumn('left_company', F.when(datafinal['left_company']=='true',1).otherwise(0))
```

```
In [194]: datafinal.select('left_company').show(7)
```

```
+-----+  
|left_company|  
+-----+  
|          0|  
|          0|  
|          0|  
|          0|  
|          0|  
|          1|  
|          0|  
+-----+
```

only showing top 7 rows

```
In [195]: #Columns that will be used as features and their types
continuous_features = ['no_of_projects', 'salary']

categorical_features = ['emp_titles_id','sex',
                        'last_performance_rating', 'left_company',
                        'title','dept_no','dept_name']
```

preprocessing data

```
In [142]: #Encoding all categorical features
from pyspark.ml.feature import OneHotEncoder, StringIndexer, VectorAssembler, PolynomialExpansion, VectorIndexer
```

```
In [196]: # create object of StringIndexer class and specify input and output column
si_emp_titles_id = StringIndexer(inputCol='emp_titles_id',outputCol='emp_titles_id_index')
si_sex = StringIndexer(inputCol='sex',outputCol='sex_index')
si_last_performance_rating = StringIndexer(inputCol='last_performance_rating',outputCol='last_performance_rating_index')
si_left_company = StringIndexer(inputCol='left_company',outputCol='left_company_index')
si_title = StringIndexer(inputCol='title',outputCol='title_index')
si_dept_no = StringIndexer(inputCol='dept_no',outputCol='dept_no_index')
si_dept_name = StringIndexer(inputCol='dept_name',outputCol='dept_name_index')

# transform the data
datafinal = si_emp_titles_id.fit(datafinal).transform(datafinal)
datafinal = si_sex.fit(datafinal).transform(datafinal)
datafinal = si_last_performance_rating.fit(datafinal).transform(datafinal)
datafinal = si_left_company.fit(datafinal).transform(datafinal)
datafinal = si_title.fit(datafinal).transform(datafinal)
datafinal = si_dept_no.fit(datafinal).transform(datafinal)
datafinal = si_dept_name.fit(datafinal).transform(datafinal)

# view the transformed data
datafinal.select('emp_titles_id', 'emp_titles_id_index', 'sex', 'sex_index','last_performance_rating',
                  'last_performance_rating_index','left_company','left_company_index','title','title_index',
                  'dept_no','dept_no_index','dept_name','dept_name_index').show(10)
```

```
In [207]: # making a udf for StringIndexer and OneHotEncoder
def create_category_vars( dataset, field_name ):
    idx_col = field_name + "Index"
    col_vec = field_name + "Vec"

    month_stringIndexer = StringIndexer( inputCol=field_name, outputCol=idx_col )

    month_model = month_stringIndexer.fit( dataset )
    month_indexed = month_model.transform( dataset )

    month_encoder = OneHotEncoder( dropLast=True,inputCol=idx_col,outputCol= col_vec )

    return month_encoder.transform( month_indexed )
```

```
Exception ignored in: <function JavaWrapper.__del__ at 0x7f63302138c0>
Traceback (most recent call last):
  File "/opt/anaconda3/lib/python3.7/site-packages/pyspark/ml/wrapper.py", line 40, in __del__
    if SparkContext._active_spark_context and self._java_obj is not None:
AttributeError: 'OneHotEncoder' object has no attribute '_java_obj'
```

```
In [208]: for col in categorical_features:
            datafinal = create_category_vars( datafinal, col )
            datafinal.cache()
```

```
In [233]: datafinal.columns
```

```
Out[233]: ['emp_titles_id',
           'sex',
           'no_of_projects',
           'last_performance_rating',
           'left_company',
           'salary',
           'title',
           'dept_no',
           'dept_name',
           'birth_date',
           'hire_date',
           'last_date',
           'emp_titles_id_index',
           'sex_index',
           'last_performance_rating_index',
           'left_company_index',
           'title_index',
           'dept_no_index',
           'dept_name_index',
           'emp_titles_idIndex',
           'emp_titles_idVec',
           'sexIndex',
           'sexVec',
           'last_performance_ratingIndex',
           'last_performance_ratingVec',
           'left_companyIndex',
           'left_companyVec',
           'titleIndex',
           'titleVec',
           'dept_noIndex',
           'dept_noVec',
           'dept_nameIndex',
           'dept_nameVec']
```

```
In [223]: #Create vectors from all features column
featureCols = featureCols = ['no_of_projects',
                              'salary',
                              'emp_titles_idVec',
                              'sexVec',
                              'last_performance_ratingVec',
                              'titleVec',
                              'dept_noVec',
                              'dept_nameVec']
```

```
In [235]: # Creating the vector of all predictors
assembler = VectorAssembler( inputCols = featureCols, outputCol = "features")
```

```
In [236]: datafinal = assembler.transform( datafinal )
```

```
In [240]: # Setting the target variables
datafinal = datafinal.withColumn( "label", datafinal.left_companyIndex)
```

```
In [241]: datafinal.select( "features", "label" ).show( 5 )
```

```
+-----+-----+
|          features|label|
+-----+-----+
|(35,[0,1,3,8,9,14,...]| 0.0|
|(35,[0,1,4,15,25,...]| 0.0|
|(35,[0,1,4,15,25,...]| 0.0|
|(35,[0,1,4,15,20,...]| 0.0|
|(35,[0,1,4,15,20,...]| 0.0|
+-----+-----+
only showing top 5 rows
```

```
In [242]: #Split the dataset
train_df, test_df = datafinal.randomSplit( [0.7, 0.3], seed = 42 )
```

```
In [243]: #Train Linear Regression Model
from pyspark.ml.classification import LogisticRegression
```

```
In [255]: logistic = LogisticRegression(featuresCol='features', labelCol='label')
```

```
In [256]: # training model
model=logistic.fit(train_df)
```

```
In [259]: pred_train=model.transform(train_df)
```

```
In [265]: model.coefficientMatrix
```

```
Out[265]: DenseMatrix(1, 35, [0.0035, -0.0, 0.0116, 0.0252, 0.0232, 0.0062, 0.0393, -0.0022, ..., 0.004, 0.0135, 0.0252, 0.0262, 0.0051,
-0.0071, -0.009, 0.0042], 1)
```

```
In [266]: model.coefficients
```

```
Out[266]: DenseVector([0.0035, -0.0, 0.0116, 0.0252, 0.0232, 0.0062, 0.0393, -0.0022, 0.0022, -0.0404, -0.0249, -0.0205, -0.017, 0.0116,
0.0252, 0.0232, 0.0062, 0.0393, -0.0022, 0.004, 0.0135, 0.0252, 0.0262, 0.0051, -0.0071, -0.009, 0.0042, 0.004, 0.0135, 0.0252,
0.0262, 0.0051, -0.0071, -0.009, 0.0042])
```

```
In [267]: model.intercept
```

```
Out[267]: -2.1889212364292177
```

```
In [271]: training_summary = model.summary
```

```
In [275]: training_summary.roc.show()
```

```
+-----+-----+
|          FPR|          TPR|
+-----+-----+
|          0.0|          0.0|
|0.0090140392456165|0.0104|
|0.01501536483962321|0.01698181818181818|
|0.023989234569884915|0.02661818181818182|
|0.033846833637952156|0.03709090909090909|
|0.04421458555102533|0.04832727272727272|
|0.05237703107112013|0.05690909090909091|
|0.059221916487577575|0.0636|
|0.06711120930325976|0.07229090909090909|
|0.07364277249994979|0.07883636363636363|
|0.08116250577436783|0.08578181818181818|
|0.08816806925224448|0.09294545454545454|
|0.09618188756552652|0.10127272727272728|
|0.10249653537930065|0.10730909090909091|
|0.11100042177991122|0.11596363636363637|
|0.11811042599771034|0.12327272727272727|
|0.12397517523951074|0.12945454545454546|
|0.13149490851392878|0.13643636363636363|
|0.13912711643134026|0.14396363636363638|
|0.1453614252144048|0.14978181818181818|
+-----+-----+
only showing top 20 rows
```

Random Forest Classifier

```
In [276]: from pyspark.ml import Pipeline
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.feature import *
from pyspark.ml.evaluation import MulticlassClassificationEvaluator, BinaryClassificationEvaluator

from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
```

```
In [277]: def train(df, classifier):
    train, test = df.randomSplit([.7,.3])

    model = classifier.fit(train)

    pred = model.transform(test)

    eval_accuracy = (MulticlassClassificationEvaluator
                     (labelCol="label", predictionCol="prediction", metricName="accuracy"))

    eval_precision = (MulticlassClassificationEvaluator
                     (labelCol="label", predictionCol="prediction", metricName="weightedPrecision"))

    eval_recall = (MulticlassClassificationEvaluator
                  (labelCol="label", predictionCol="prediction", metricName="weightedRecall"))

    eval_f1 = (MulticlassClassificationEvaluator
              (labelCol="label", predictionCol="prediction", metricName="f1"))

    accuracy = eval_accuracy.evaluate(pred)

    precision = eval_precision.evaluate(pred)

    recall = eval_recall.evaluate(pred)

    f1 = eval_f1.evaluate(pred)

    print(f"""
Accuracy = {accuracy}
Error    = {1-accuracy}
Precision = {precision}
Recall   = {recall}
F1       = {f1}""")

    return model, pred
```

```
In [280]: rf = RandomForestClassifier(labelCol="label", featuresCol="features")

_, pred = train(datafinal, rf)

pred.select("prediction", "label", "features").show()
```

```
Accuracy = 0.9006221128232795
Error    = 0.09937788717672047
Precision = 0.811120190106268
Recall    = 0.9006221128232795
F1        = 0.8535312565646093
```

prediction	label	features
0.0	1.0	(35,[0,1,7,10,18,...
0.0	0.0	(35,[0,1,7,10,18,...
0.0	0.0	(35,[0,1,7,11,18,...
0.0	0.0	(35,[0,1,7,11,18,...
0.0	0.0	(35,[0,1,7,10,18,...
0.0	0.0	(35,[0,1,7,10,18,...
0.0	0.0	(35,[0,1,7,8,10,1...
0.0	0.0	(35,[0,1,7,8,10,1...
0.0	0.0	(35,[0,1,7,8,11,1...
0.0	0.0	(35,[0,1,7,8,10,1...
0.0	0.0	(35,[0,1,7,8,11,1...
0.0	0.0	(35,[0,1,7,8,11,1...
0.0	0.0	(35,[0,1,7,8,11,1...
0.0	0.0	(35,[0,1,4,9,15,2...
0.0	0.0	(35,[0,1,4,9,15,2...
0.0	0.0	(35,[0,1,4,12,15,...
0.0	0.0	(35,[0,1,4,10,15,...
0.0	0.0	(35,[0,1,4,10,15,...
0.0	1.0	(35,[0,1,4,10,15,...

only showing top 20 rows

Create entire data pipeline and ML pipe line

Create .sql file with commands of create database and table with queries.

File - > create_database_table_pipeline_sql.sql

```
drop database if exists project_de;
create database project_de;
use project_de;

drop table if exists employees
CREATE EXTERNAL TABLE employees STORED AS AVRO
LOCATION '/user/anabig114225/projectdata/employees'
TBLPROPERTIES ('avro.schema.url'='/user/anabig114225/projectschema/employees.avsc');

drop table if exists titles
CREATE EXTERNAL TABLE titles STORED AS AVRO
LOCATION '/user/anabig114225/projectdata/titles'
TBLPROPERTIES ('avro.schema.url'='/user/anabig114225/projectschema/titles.avsc');

drop table if exists salaries
CREATE EXTERNAL TABLE salaries STORED AS AVRO
LOCATION '/user/anabig114225/projectdata/salaries'
TBLPROPERTIES ('avro.schema.url'='/user/anabig114225/projectschema/salaries.avsc');

drop table if exists departments
CREATE EXTERNAL TABLE departments STORED AS AVRO
LOCATION '/user/anabig114225/projectdata/departments'
TBLPROPERTIES ('avro.schema.url'='/user/anabig114225/projectschema/departments.avsc');

drop table if exists department_manager
CREATE EXTERNAL TABLE department_manager STORED AS AVRO
LOCATION '/user/anabig114225/projectdata/department_manager'
TBLPROPERTIES ('avro.schema.url'='/user/anabig114225/projectschema/department_manager.avsc');

drop table if exists department_employees
CREATE EXTERNAL TABLE department_employees STORED AS AVRO
LOCATION '/user/anabig114225/projectdata/department_employees'
TBLPROPERTIES ('avro.schema.url'='/user/anabig114225/projectschema/department_employees.avsc');

select e.emp_no, last_name, first_name, sex, salary from employees e
inner join salaries s on e.emp_no=s.emp_no;
```

Create .sh file as mysql_sqoop_pipeline_sh.sh and execute it.

Commands are to be saved in mysql_sqoop_pipeline_sh.sh file are

```
hdfs dfs -rm -r projectdata
hdfs dfs -rm -r projectschema
```

```
sqoop import-all-tables --connect jdbc:mysql://ip-10-1-1-204.ap-south-1.compute.internal:3306/anabig114225
--username anabig114225 --password Bigdata123 --compression-codec=snappy --as-avrodatafile --warehouse-
dir=/user/anabig114225/projectdata --m 1 --driver com.mysql.jdbc.Driver
```

```
hdfs dfs -mkdir projectschema
hdfs dfs -copyFromLocal ~/*.avsc projectschema
hive -f create_database_table_pipeline_sql.sql > output.txt
```

```

hdfs dfs -rm -r projectdata
hdfs dfs -rm -r projectschema

sqoop import-all-tables --connect jdbc:mysql://ip-10-1-1-204.ap-south-1.compute.internal:3306/anabigl14225 --username anabigl14225 --password Bigdata123 --compression-codec=snappy --as-avrodatafile --warehouse-dir=/user/anabigl14225/projectdata --m 1 --driver com.mysql.jdbc.Driver

hdfs dfs -mkdir projectschema
hdfs dfs -copyFromLocal ~/.avsc projectschema

hive -f create_database_table_pipeline_sql.sql > output.txt

```

Create entire ML pipeline

Pipeline creation

```

In [ ]: # dataset

# define stage 1 : transform the category columns to numeric
stage1 = StringIndexer(inputCol= 'category_1', outputCol= 'category_1_index')

# define stage 2 : one hot encode the numeric category_2 column
stage2 = OneHotEncoder(inputCols=['category_1_index'], outputCols=['category_2_vec'])

# Creating the vector of all predictors
stage3 = VectorAssembler( inputCols = featureCols, outputCol = "features").transform(dataset)

# Setting the target variables -
stage4 = datafinal.withColumn( "label", datafinal.targetvariable)

# define stage 5: Logistic regression model
stage5 = LogisticRegression(featuresCol='features',labelCol='label')

# setup the pipeline
pipeline = Pipeline(stages=[stage1, stage2, stage3, stage4, stage5 ])

# fit the pipeline model and transform the data as defined
pipeline_model = pipeline.fit(train)
sample_train_pred = pipeline_model.transform(train)

# view the transformed data
sample_train_pred.show()

```

Challenges

- Creating the data tables as per their correct data type and then importing it to MySQL
- Deciding in which format to import the tables from MySQL into HDFS
- Finding avsc schema files of table and then saving them on hdfs into new directory
- Creating table in hive using the format as specified earlier while importing the tables, With mentioning data location and schema location.
- Taking hive tables to spark
- Converting datatypes of variables
- Building sparkML with different techniques
- Finding wayouts to create data pipeline and ML pipeline

Way ahead or Conclusion

On whole it's way very good learning project assimilating all interconnecting all tools mysql, sqoop, hdfs, hive, spark, sparkML for transferring the data tables, schemas and doing analysis on them. All of the tools being integrated into this one single project of data engineering.