PROJECT DOCUMENTATION

- **Chinmay Suresh**

OBJECTIVE

Main objective of this project is to create an end-to-end data pipeline and analyze data based on database of employees from 1980s-1995s belonging to a big corporation.

Data Description

a) Employees (employees.csv):

Variable	Description	Type
emp_no	Employee Id	Integer
emp_titles_id	designation id	Character
birth_date	Date of Birth	Date Time
first_name	First Name	Character
last_name	Last Name	Character
sex	Gender	Character
hire_date	Employee Hire date	Date Time
no_of_projects	No. of projects worked on	Integer
left	Employee left the organization	Boolean
Last_date	Last date of employment (Exit Date)	Date Time

b) Titles (titles.csv):

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

title – Designation – Character – Not Null

c) Salaries:

emp_no – Employee id – Integer – Not Null

salary – Employee's Salary – Integer – Not Null

d) Departments (departments.csv):

dept_no – Unique id for each dept. – Character – Not Null

dept_name – Department Name – Character – Not null

e) Department Managers (dept_manager.csv):

dept_no – Unique id for each dept. – Character – Not Null

emp_no – Employee no. (head of the dept.) – Integer – Not Null

f) Department Employees (dept_emp.csv):

emp_no – Employee id – Integer – Not Null

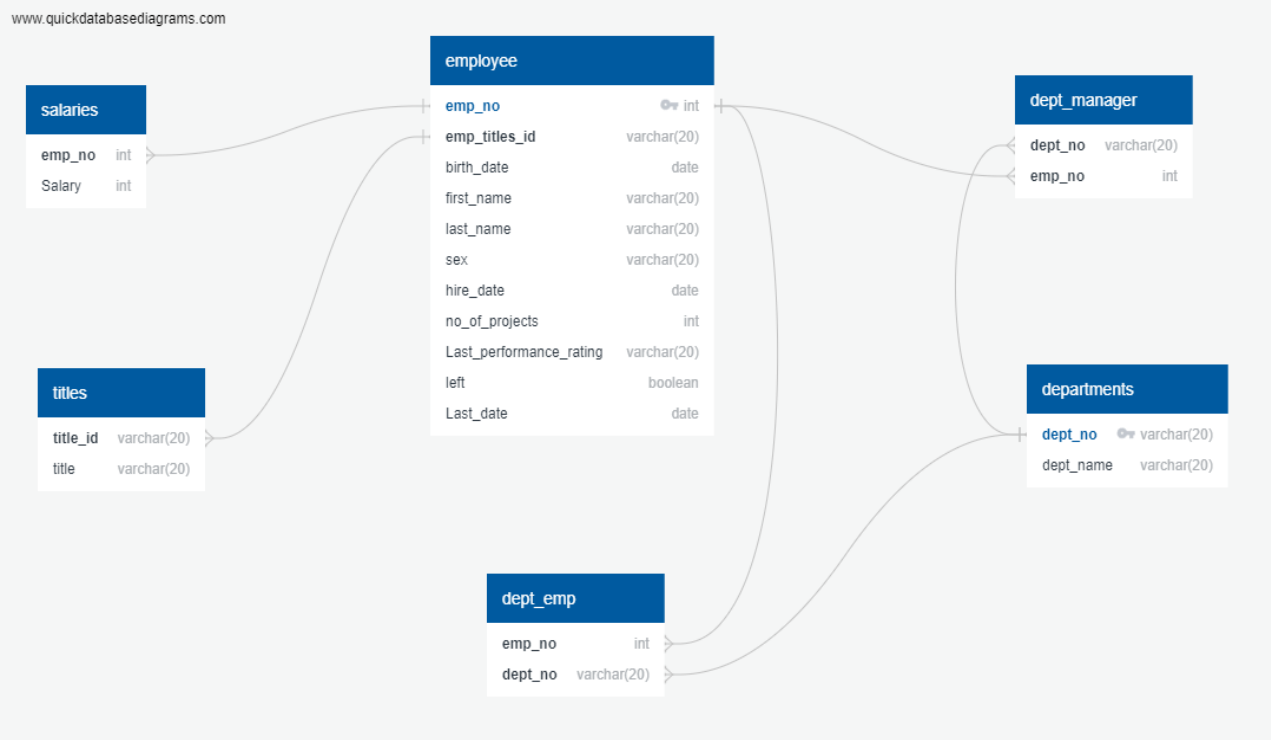
dept_no – Unique id for each dept. – Character – Not Null

Technology Stack

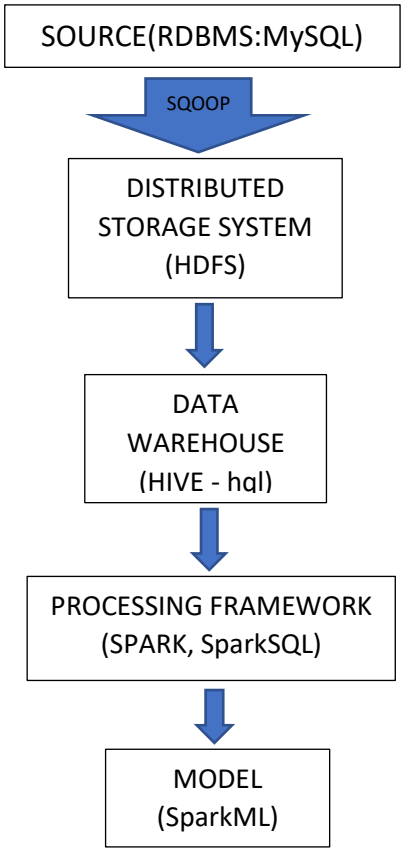
Technologies:

1) Linux
2) RDBMS (MySQL)
3) Sqoop (Transfer data from RDBMS to HDFS)
4) HDFS (Stores the data)
5) Hive (Create database and tables, load data into tables)
6) Impala (Performing EDA)
7) SparkSQL (Performing EDA)
8) SparkML (Model Building)

Entity Relationship Diagram (ERD Data Model)



Architecture of pipeline (stages)



Exploratory Data Analysis

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

Ans:

```
spark.sql("""SELECT E.emp_no, E.last_name, E.first_name, E.sex, S.salary FROM emp AS E
          INNER JOIN salary AS S ON E.emp_no=S.emp_no""").show()
```

```
+------+----------+----------+---+------+
|emp_no| last_name|first_name|sex|salary|
+------+----------+----------+---+------+
|473302|  Zallocco|  Hideyuki|  M| 40000|
|475053| Delgrande|     Byong|  F| 53422|
| 57444|      Babb|     Berry|  F| 48973|
|421786|  Verhoeff|     Xiong|  M| 40000|
|282238|   Baumann|Abdelkader|  F| 40000|
|263976|  Cusworth|      Eran|  M| 40000|
|273487|   Parfitt| Christoph|  M| 56087|
|461591|  Samarati|    Xudong|  M| 40000|
|477657| Magliocco|    Lihong|  M| 54816|
|219881|     Speek| Kwangyoen|  F| 40000|
| 29920|     Tyugu|   Shuichi|  F| 40000|
|208153|      Lunn|   Abdulah|  M| 50183|
| 13616|     Lorho|     Perry|  F| 40000|
|246449|Bultermann|     Subbu|  F| 87084|
| 21529|  Zallocco|     Bojan|  M| 40000|
| 17934|  Wuwongse|  Bilhanan|  M| 48795|
| 48085|      Gilg|Venkatesan|  M| 63016|
|239838|     Dulli|   Naftali|  M| 57715|
|240129|    Karnin|     Roddy|  M| 43548|
|205246|    Demizu|     Nevio|  F| 80252|
+------+----------+----------+---+------+
only showing top 20 rows
```

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

Ans:

```
spark.sql("SELECT first_name, last_name, hire_date FROM emp WHERE year(hire_date)=1986").show()
```

```
+----------+-----------+-------------------+
|first_name|  last_name|          hire_date|
+----------+-----------+-------------------+
|      Eran|   Cusworth|1986-11-14 00:00:00|
|     Bojan|   Zallocco|1986-10-14 00:00:00|
|     Nevio|     Demizu|1986-05-18 00:00:00|
|      Ziva|     Vecchi|1986-07-03 00:00:00|
|     Mohit|      Speek|1986-01-14 00:00:00|
|  Qunsheng|      Speer|1986-02-13 00:00:00|
|     Dines|Encarnacion|1986-08-02 00:00:00|
|  Harngdar|      Swick|1986-05-28 00:00:00|
|    Freyja|      Uhrig|1986-12-20 00:00:00|
|   Zhenhua|    Milicic|1986-08-04 00:00:00|
|     Bowen|   Schmezko|1986-05-30 00:00:00|
|    Reuven|      Munke|1986-04-13 00:00:00|
| Stabislas|    Domenig|1986-08-04 00:00:00|
| Juichirou|   Jumpertz|1986-09-18 00:00:00|
| Juichirou|     Ghelli|1986-12-13 00:00:00|
|    Shmuel|      Georg|1986-11-30 00:00:00|
|   Xiadong|       Bach|1986-04-29 00:00:00|
| Guangming|Butterworth|1986-10-01 00:00:00|
|     Irene|      Setia|1986-08-10 00:00:00|
|   Shuichi|     Swiler|1986-06-08 00:00:00|
+----------+-----------+-------------------+
only showing top 20 rows
```

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.**

Ans:

```
spark.sql("""SELECT D.dept_no, D.dept_name, E.emp_no, E.last_name, E.first_name
FROM emp E INNER JOIN depts_emp DE ON E.emp_no=DE.emp_no INNER JOIN dept D ON DE.dept_no=D.dept_no""").show()
```

```
+-------+--------------------+------+----------+----------+
|dept_no|           dept_name|emp_no| last_name|first_name|
+-------+--------------------+------+----------+----------+
|   d002|           "Finance"|473302|  Zallocco|  Hideyuki|
|   d004|        "Production"|475053| Delgrande|     Byong|
|   d004|        "Production"| 57444|      Babb|     Berry|
|   d003|   "Human Resources"|421786|  Verhoeff|     Xiong|
|   d006|"Quality Management"|282238|   Baumann|Abdelkader|
|   d006|"Quality Management"|263976|  Cusworth|      Eran|
|   d003|   "Human Resources"|273487|   Parfitt| Christoph|
|   d002|           "Finance"|461591|  Samarati|    Xudong|
|   d006|"Quality Management"|477657| Magliocco|    Lihong|
|   d009|  "Customer Service"|219881|     Speek| Kwangyoen|
|   d004|        "Production"| 29920|     Tyugu|   Shuichi|
|   d005|       "development"|208153|      Lunn|   Abdulah|
|   d008|          "Research"| 13616|     Lorho|     Perry|
|   d005|       "development"| 13616|     Lorho|     Perry|
|   d007|             "Sales"|246449|Bultermann|     Subbu|
|   d005|       "development"| 21529|  Zallocco|     Bojan|
|   d005|       "development"| 17934|  Wuwongse|  Bilhanan|
|   d002|           "Finance"| 48085|      Gilg|Venkatesan|
|   d004|        "Production"|239838|     Dulli|   Naftali|
|   d006|"Quality Management"|240129|    Karnin|     Roddy|
+-------+--------------------+------+----------+----------+
only showing top 20 rows
```

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

Ans:

```
spark.sql('''SELECT E.emp_no, E.last_name, E.first_name, D.dept_name FROM emp E
          INNER JOIN depts_emp DE ON E.emp_no=DE.emp_no INNER JOIN dept D ON DE.dept_no=D.dept_no''').show()
```

```
+------+----------+----------+--------------------+
|emp_no| last_name|first_name|           dept_name|
+------+----------+----------+--------------------+
|473302|  Zallocco|  Hideyuki|           "Finance"|
|475053| Delgrande|     Byong|        "Production"|
| 57444|      Babb|     Berry|        "Production"|
|421786|  Verhoeff|     Xiong|   "Human Resources"|
|282238|   Baumann|Abdelkader|"Quality Management"|
|263976|  Cusworth|      Eran|"Quality Management"|
|273487|   Parfitt| Christoph|   "Human Resources"|
|461591|  Samarati|    Xudong|           "Finance"|
|477657| Magliocco|    Lihong|"Quality Management"|
|219881|     Speek| Kwangyoen|  "Customer Service"|
| 29920|     Tyugu|   Shuichi|        "Production"|
|208153|      Lunn|   Abdulah|       "development"|
| 13616|     Lorho|     Perry|          "Research"|
| 13616|     Lorho|     Perry|       "development"|
|246449|Bultermann|     Subbu|             "Sales"|
| 21529|  Zallocco|     Bojan|       "development"|
| 17934|  Wuwongse|  Bilhanan|       "development"|
| 48085|      Gilg|Venkatesan|           "Finance"|
|239838|     Dulli|   Naftali|        "Production"|
|240129|    Karnin|     Roddy|"Quality Management"|
+------+----------+----------+--------------------+
only showing top 20 rows
```

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

Ans:

```
[ip-10-1-2-103.ap-south-1.compute.internal:21000] capst1chin> SELECT first_name, last_name, sex from emp where first_name="Hercules" and last_name LIKE "B%" LIMIT 10;SE
LECT first_name, last_name, sex from emp where first_name="Hercules" and last_name LIKE "B%" LIMIT 10;
Query: SELECT first_name, last_name, sex from emp where first_name="Hercules" and last_name LIKE "B%" LIMIT 10
Query submitted at: 2022-05-20 02:51:03 (Coordinator: http://ip-10-1-2-103.ap-south-1.compute.internal:25000)
Query progress can be monitored at: http://ip-10-1-2-103.ap-south-1.compute.internal:25000/query_plan?query_id=ff45713dc4ddb9d6:13cee2c000000000
+------------+--------------+-----+
| first_name | last_name    | sex |
+------------+--------------+-----+
| Hercules   | Baer         | M   |
| Hercules   | Biron        | F   |
| Hercules   | Birge        | F   |
| Hercules   | Berstel      | F   |
| Hercules   | Bernatsky    | M   |
| Hercules   | Bail         | F   |
| Hercules   | Bodoff       | M   |
| Hercules   | Benantar     | F   |
| Hercules   | Basagni      | M   |
| Hercules   | Bernardinello| F   |
+------------+--------------+-----+
Fetched 10 row(s) in 0.32s
```

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

Ans:

```
[ip-10-1-2-103.ap-south-1.compute.internal:21000] capst1chin> SELECT E.emp_no, E.last_name, E.first_name, D.dept_no, D.dept_name
                                                           >
                                                           > FROM emp E INNER JOIN depts_emp DE ON E.emp_no=DE.emp_no INNER JOIN dept D ON DE.dept_no=D.dept_no
                                                           >
                                                           > WHERE D.dept_name='"Sales"' LIMIT 10;
Query: SELECT E.emp_no, E.last_name, E.first_name, D.dept_no, D.dept_name
FROM emp E INNER JOIN depts_emp DE ON E.emp_no=DE.emp_no INNER JOIN dept D ON DE.dept_no=D.dept_no
WHERE D.dept_name='"Sales"' LIMIT 10
Query submitted at: 2022-05-20 02:46:36 (Coordinator: http://ip-10-1-2-103.ap-south-1.compute.internal:25000)
Query progress can be monitored at: http://ip-10-1-2-103.ap-south-1.compute.internal:25000/query_plan?query_id=5a4c95e344e76580:930f8cf700000000
+--------+--------------+------------+---------+-----------+
| emp_no | last_name    | first_name | dept_no | dept_name |
+--------+--------------+------------+---------+-----------+
| 246449 | Bultermann   | Subbu      | d007    | "Sales"   |
| 205246 | Demizu       | Nevio      | d007    | "Sales"   |
| 476443 | Asmuth       | Ziya       | d007    | "Sales"   |
| 424270 | Yoshizawa    | Kellyn     | d007    | "Sales"   |
| 280408 | Perl         | Elliott    | d007    | "Sales"   |
| 289261 | Nollmann     | Gad        | d007    | "Sales"   |
| 444985 | Verspoor     | Giap       | d007    | "Sales"   |
| 477628 | Beutelspacher| Duro       | d007    | "Sales"   |
| 42625  | Swick        | Harngdar   | d007    | "Sales"   |
| 85093  | Covnot       | Nectarios  | d007    | "Sales"   |
+--------+--------------+------------+---------+-----------+
Fetched 10 row(s) in 0.83s
```

7. **A list showing all employees in the Sales and Development departments, including their employee number, last name, first name, and department name.**

Ans:

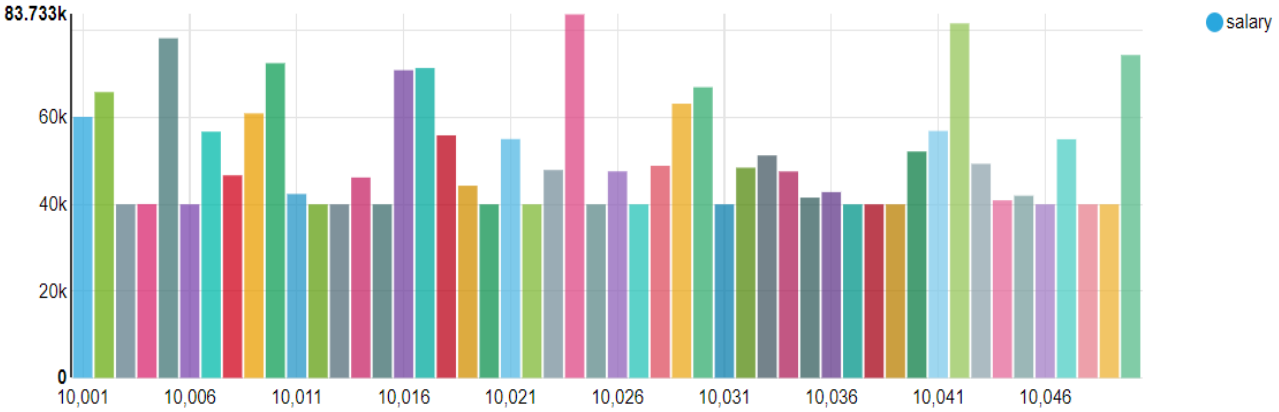
```
[ip-10-1-2-103.ap-south-1.compute.internal:21000] capst1chin> SELECT E.emp_no, E.last_name, E.first_name, D.dept_name
                                                           >
                                                           >        FROM emp E INNER JOIN depts_emp DE ON E.emp_no=DE.emp_no INNER JOIN dept D ON DE.dept_no=D.dept_no
                                                           >
                                                           > WHERE D.dept_name IN ('"Sales"','"development"') LIMIT 10;
Query: SELECT E.emp_no, E.last_name, E.first_name, D.dept_name
       FROM emp E INNER JOIN depts_emp DE ON E.emp_no=DE.emp_no INNER JOIN dept D ON DE.dept_no=D.dept_no
WHERE D.dept_name IN ('"Sales"','"development"') LIMIT 10
Query submitted at: 2022-05-20 02:48:09 (Coordinator: http://ip-10-1-2-103.ap-south-1.compute.internal:25000)
Query progress can be monitored at: http://ip-10-1-2-103.ap-south-1.compute.internal:25000/query_plan?query_id=73483eee3e1e4361:8b3bfbdb00000000
+--------+-----------+-----------+--------------+
| emp_no | last_name | first_name | dept_name    |
+--------+-----------+-----------+--------------+
| 208153 | Lunn      | Abdulah   | "development" |
| 13616  | Lorho     | Perry     | "development" |
| 246449 | Bultermann | Subbu    | "Sales"      |
| 21529  | Zallocco  | Bojan     | "development" |
| 17934  | Wuwongse  | Bilhanan  | "development" |
| 205246 | Demizu    | Nevio     | "Sales"      |
| 476443 | Asmuth    | Ziya      | "Sales"      |
| 424270 | Yoshizawa | Kellyn    | "Sales"      |
| 71530  | McAlpine  | Venkatesan | "development" |
| 280408 | Perl      | Elliott   | "Sales"      |
+--------+-----------+-----------+--------------+
Fetched 10 row(s) in 0.83s
```

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

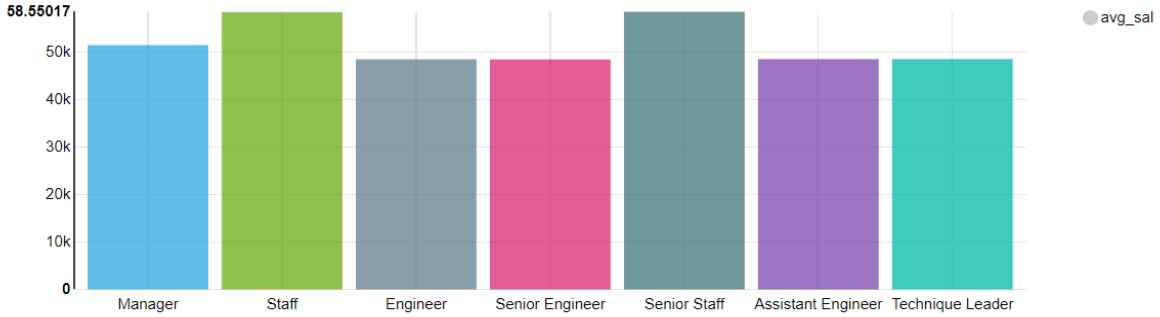
Ans:

```
[ip-10-1-2-103.ap-south-1.compute.internal:21000] capst1chin> SELECT last_name, count(last_name) cnt FROM emp GROUP BY last_name ORDER BY cnt DESC LIMIT 10;
Query: SELECT last_name, count(last_name) cnt FROM emp GROUP BY last_name ORDER BY cnt DESC LIMIT 10
Query submitted at: 2022-05-20 02:49:21 (Coordinator: http://ip-10-1-2-103.ap-south-1.compute.internal:25000)
Query progress can be monitored at: http://ip-10-1-2-103.ap-south-1.compute.internal:25000/query_plan?query_id=5b42548beb79c34b:5732d84400000000
+-----------+-----+
| last_name | cnt |
+-----------+-----+
| Baba      | 226 |
| Gelosh    | 223 |
| Coorg     | 223 |
| Sudbeck   | 222 |
| Farris    | 222 |
| Adachi    | 221 |
| Osgood    | 220 |
| Neiman    | 218 |
| Masada    | 218 |
| Mandell   | 218 |
+-----------+-----+
Fetched 10 row(s) in 0.32s
```

9. Histogram to show the salary distribution among the employees.



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



We can observe from the above bar graph that Senior Staff and Staff have almost the same average salary followed by Managers.

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

Ans:

```
max_year = spark.sql("""SELECT MAX(year(last_date)) FROM emp""") #2013
spark.sql("""SELECT e.employee_tenure, count(e.employee_tenure) cnt FROM
            (SELECT CASE WHEN lefts=0 THEN CONCAT(CEILING(DATEDIFF('2013',hire_date)/365.25)," yrs")
            WHEN lefts=1 THEN CONCAT(CEILING(DATEDIFF(last_date,hire_date)/365.25)," yrs")
            END AS employee_tenure from emp) AS e GROUP BY e.employee_tenure ORDER BY cnt DESC""").show()
```

```
+---------------+-----+
|employee_tenure|  cnt|
+---------------+-----+
|          27 yrs|32532|
|          28 yrs|31904|
|          26 yrs|30191|
|          25 yrs|28120|
|          24 yrs|25666|
|          23 yrs|23134|
|          22 yrs|20292|
|          21 yrs|18326|
|          20 yrs|15990|
|          19 yrs|13336|
|          18 yrs|10900|
|          17 yrs| 8600|
|          16 yrs| 6049|
|          15 yrs| 3736|
|          14 yrs| 3027|
|           4 yrs| 2497|
|           5 yrs| 2485|
|          11 yrs| 2468|
|           8 yrs| 2456|
|           9 yrs| 2437|
+---------------+-----+
only showing top 20 rows
```

We can observe that most of the employees has 25+ years of tenure, whereas most of the employees who left had a shorter tenure of below 10 years.

12. Gender-wise count by Designation.

```
spark.sql("""SELECT T.title, E.sex, COUNT(E.sex) gender_cnt
          FROM emp AS E INNER JOIN salary AS S ON E.emp_no=S.emp_no
          INNER JOIN title AS T on E.emp_title_id=T.title_id GROUP BY T.title, E.sex
          ORDER BY gender_cnt DESC,T.title,E.sex""").show()
```

```
+------------------+---+----------+
|             title|sex|gender_cnt|
+------------------+---+----------+
|             Staff|  M|     64534|
|   Senior Engineer|  M|     58608|
|             Staff|  F|     42850|
|   Senior Engineer|  F|     39139|
|          Engineer|  M|     28340|
|          Engineer|  F|     18963|
|      Senior Staff|  M|     15937|
|      Senior Staff|  F|     10646|
|  Technique Leader|  M|      9041|
|  Technique Leader|  F|      6107|
|Assistant Engineer|  M|      3502|
|Assistant Engineer|  F|      2333|
|           Manager|  F|        13|
|           Manager|  M|        11|
+------------------+---+----------+
```

Staff designation has the highest number of both Male and Female workers whereas Manager has the least number of Male and Female workers.

13. count and average tenure (in yrs) of employees who left by department

Ans:

```
spark.sql("""SELECT D.dept_name, AVG(CEILING(DATEDIFF(E.last_date,E.hire_date)/365.25)) as emp_tenure,
COUNT(E.lefts) left_cnt FROM employee E INNER JOIN dept_emp DE ON E.emp_no=DE.emp_no INNER JOIN depts D ON DE.dept_no=D.dept_no
INNER JOIN salaries S ON E.emp_no=S.emp_no WHERE E.lefts=1 GROUP BY D.dept_name ORDER BY left_cnt DESC""").show()
```

```
+------------------+----------+--------+
|         dept_name|emp_tenure|left_cnt|
+------------------+----------+--------+
|      "development"|    8.0112|    8508|
|       "Production"|    8.0742|    7389|
|          "Sales"|    7.9844|    5209|
| "Customer Service"|    8.0427|    2414|
|        "Research"|    8.1325|    2098|
|"Quality Management"|    8.0025|    2018|
|       "Marketing"|    7.9181|    1941|
|  "Human Resources"|    8.0161|    1797|
|         "Finance"|    7.8986|    1647|
+------------------+----------+--------+
```

We can observe that the count of employees who left is the maximum from Development department, whereas the least is from finance. Meanwhile average tenures of employees who left is around 8 years by each department.

14. Count of employees who left vs stayed.

```
spark.sql("SELECT lefts, COUNT(lefts) FROM employee GROUP BY lefts").show()
```

```
+-----+------------+
|lefts|count(lefts)|
+-----+------------+
|    1|       29867|
|    0|      270157|
+-----+------------+
```

As we can notice, the employees who stayed are in much larger numbers than the employees who left.

15. Last year performance ratings of employees.

```
spark.sql("""SELECT Last_performance_rating, count(Last_performance_rating) as cnt
            FROM emp GROUP BY Last_performance_rating ORDER BY cnt DESC""").show()
```

```
+----------------------+------+
|Last_performance_rating|   cnt|
+----------------------+------+
|                     B|107154|
|                     A| 95919|
|                     C| 71304|
|                   PIP| 15105|
|                     S| 10542|
+----------------------+------+
```

We can observe that Employees with average performance of B are present at maximum. Employees having excellent performance of S are present in less numbers.

ML Model:

Selecting all the required columns from all the tables for further processes:

```
data = spark.sql(""" SELECT E.emp_no, E.emp_title_id, DAY(E.birth_date) birth_day, MONTH(E.birth_date) birth_month,
                YEAR(E.birth_date) birth_year, E.first_name, E.last_name, E.sex, E.no_of_projects, E.last_performance_rating,
                E.lefts, D.dept_name, S.salary, T.title, DE.dept_no,
                CASE WHEN lefts=0 THEN CEILING(DATEDIFF('2013',hire_date)/365.25)
                WHEN lefts=1 THEN CEILING(DATEDIFF(last_date,hire_date)/365.25)
                END AS employee_tenure
                FROM emp E LEFT JOIN depts_emp DE ON E.emp_no=DE.emp_no
                LEFT JOIN dept D ON DE.dept_no=D.dept_no LEFT JOIN dept_man DM ON E.emp_no=DM.emp_no
                LEFT JOIN salary S ON E.emp_no=S.emp_no
                LEFT JOIN title AS T on E.emp_title_id=T.title_id """)
```

1) Feature Engineering:

Removing duplicate records:

```
# removing duplicated records
final = data.distinct()
final.show(vertical = True)
```

Checking for null values:

```
#checking no. of null values in each column
from pyspark.sql.functions import col,isnan, when, count
import pyspark.sql.functions as F
final.agg(*[F.count(F.when(F.isnull(c), c)).alias(c) for c in final.columns]).show(vertical=True)
```

```
-RECORD 0----------------------
 emp_no                  | 0
 emp_title_id            | 0
 birth_day               | 0
 birth_month             | 0
 birth_year              | 0
 sex                     | 0
 no_of_projects          | 0
 last_performance_rating | 0
 left                    | 0
 dept_name               | 0
 salary                  | 0
 title                   | 0
 dept_no                 | 0
 employee_tenure         | 0
 Name                    | 0
```

Schema:

```
final.printSchema()
```

```
root
 |-- emp_no: integer (nullable = true)
 |-- emp_title_id: string (nullable = true)
 |-- birth_day: integer (nullable = true)
 |-- birth_month: integer (nullable = true)
 |-- birth_year: integer (nullable = true)
 |-- sex: string (nullable = true)
 |-- no_of_projects: integer (nullable = true)
 |-- last_performance_rating: string (nullable = true)
 |-- left: integer (nullable = true)
 |-- dept_name: string (nullable = true)
 |-- salary: integer (nullable = true)
 |-- title: string (nullable = true)
 |-- dept_no: string (nullable = true)
 |-- employee_tenure: integer (nullable = true)
 |-- Name: string (nullable = true)
```

String Indexing and Vector Assembling:

```
numeric_cols = ['emp_no','birth_day','birth_month','birth_year','no_of_projects','salary','employee_tenure']
categorical_cols = ['emp_title_id','sex','last_performance_rating','dept_name','title','dept_no']
```

```
from pyspark.ml.feature import StringIndexer
from pyspark.ml.feature import VectorAssembler
label_indexer = StringIndexer(inputCol = 'left', outputCol = 'label')
empTitleid_indexer = StringIndexer(inputCol = 'emp_title_id', outputCol = 'empTitleid_index')
gender_indexer = StringIndexer(inputCol = 'sex', outputCol = 'sex_index')
lastperfRating_indexer = StringIndexer(inputCol = 'last_performance_rating', outputCol = 'lastperfRating_index')
deptName_indexer = StringIndexer(inputCol = 'dept_name', outputCol = 'deptName_index')
title_indexer = StringIndexer(inputCol = 'title', outputCol = 'title_index')
deptNo_indexer = StringIndexer(inputCol = 'dept_no', outputCol = 'deptNo_index')
```

```
assembler = VectorAssembler(
inputCols = ['empTitleid_index', 'sex_index','lastperfRating_index','deptName_index','title_index','deptNo_index']+numeric_cols,
outputCol = 'features')
```

ML Pipeline [Stages: String Indexer(s) + Label Indexer + Assembler+ ClassifierModel (RandomForest)]:

```
(train, test) = final.randomSplit([0.7, 0.3])
```

```
from pyspark.ml import Pipeline
from pyspark.ml.classification import RandomForestClassifier

classifier = RandomForestClassifier(labelCol = 'label', featuresCol = 'features')
pipeline = Pipeline(stages=[empTitleid_indexer, gender_indexer, lastperfRating_indexer , deptName_indexer, title_indexer,
                            deptNo_indexer, label_indexer, assembler, classifier])
model = pipeline.fit(train)
```

```
predictions = model.transform(test)
```

Model Validation:

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
eval_accuracy = (MulticlassClassificationEvaluator
                 (labelCol="label", predictionCol="prediction", metricName="accuracy"))
accuracy = eval_accuracy.evaluate(predictions)

eval_precision = (MulticlassClassificationEvaluator
        (labelCol="label", predictionCol="prediction", metricName="weightedPrecision"))
precision =  eval_precision.evaluate(predictions)

eval_recall = (MulticlassClassificationEvaluator
        (labelCol="label", predictionCol="prediction", metricName="weightedRecall"))
recall =  eval_recall.evaluate(predictions)

eval_f1 = (MulticlassClassificationEvaluator
        (labelCol="label", predictionCol="prediction", metricName="f1"))
f1 =  eval_f1.evaluate(predictions)
```

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

```
      Accuracy  = 0.9956020900159374
      Error     = 0.004397909984062642
      Precision = 0.9956958121088637
      Recall    = 0.9956020900159375
      F1        = 0.9956286601095841
```

Save the Model: `model.save("capstone1_rfclassifier.model")`

Challenges:

1) Choosing the appropriate data format for data transfer from RDBMS to HDFS: I ended up choosing the AVRO format as it posed less problems and gave good performance.

2) Trying to find the connection of pyspark with hive was time-consuming. I finally managed to import the right modules to connect to the hive metastore.

3) Transforming data after joining all the tables: There were lot of duplicate records which had to be removed for further stage of analysis.

4) Choosing the right model for the prediction and building a pipeline.

Next Steps:

Monitoring of pipeline is an important aspect after creation of a pipeline, as it keeps the pipeline operational, capable of extracting and loading the data and also helps maintain data integrity in the process.
This way the data flows from source to destination can be easily accessible, and meaningful to the end-users.