Hive Assignment 1

Car Insurance Cold Calls Data Analysis

Note: Replace all the path, directory names, table and column names the way you have created it in your system

Problem 1: Data Loading and Inspection

1.1. Load the car insurance data into your HDFS (Hadoop Distributed File System).

```
hadoop fs -put local_file_path /hdfs/path/to/directory
```

1.2. Create a Hive table Insurance_Cold_Calls that matches the schema of the loaded data. Make sure to choose the correct data types for each column.

```
CREATE EXTERNAL TABLE IF NOT EXISTS
Insurance_Cold_Calls (
  Id INT,
  Age INT,
  Job STRING,
  Marital STRING,
  Education STRING,
  Default STRING,
  Balance INT,
  Housing STRING,
  Loan STRING,
  Contact STRING,
  Day INT,
  Month STRING,
  Duration INT,
  Campaign INT,
  Pdays INT,
  Previous INT,
  Poutcome STRING,
  CarInsurance STRING
)
ROW FORMAT DELIMITED
```

```
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION '/hdfs/path/to/directory';
```

1.3. Load data into the Insurance_Cold_Calls table.

```
LOAD DATA INPATH '/hdfs/path/to/directory' INTO TABLE
Insurance_Cold_Calls;
```

1.4. Write a HiveQL query to display the first 10 records.

```
SELECT * FROM Insurance_Cold_Calls LIMIT 10;
```

Problem 2: Data Exploration

2.1. Write a HiveQL query to count the total number of records in the dataset.

```
SELECT COUNT(*) FROM Insurance_Cold_Calls;
```

2.2. Write a HiveQL query to find out the count of each outcome (Success, Failure, Other, No previous contact).

```
SELECT Poutcome, COUNT(*) as Count FROM
Insurance_Cold_Calls GROUP BY Poutcome;
```

2.3. Write a HiveQL query to find the count of customers grouped by Education Level.

```
SELECT Education, COUNT(*) as Count FROM
Insurance_Cold_Calls GROUP BY Education;
```

Problem 3: Complex Queries

3.1. Write a HiveQL query to find out the average call duration for each outcome.

```
SELECT Poutcome, AVG(Duration) as average_duration
FROM Insurance_Cold_Calls
```

GROUP BY Poutcome;

3.2. Find out the maximum number of contacts performed for a single customer before this campaign and for a single customer during this campaign.

SELECT MAX(Previous) as max_previous, MAX(Campaign) as max_campaign FROM Insurance_Cold_Calls;

3.3. Write a HiveQL query to find the number of days that passed by after the client was last contacted from a previous campaign, grouped by the outcome of the campaign.

SELECT Poutcome, AVG(Pdays) as avg_days_passed FROM Insurance_Cold_Calls WHERE Pdays != -1 GROUP BY Poutcome;

Problem 4: Advanced Analysis

4.1. Write a HiveQL query to find the age group that has the highest number of insurance purchases. Assume age groups as follows: 18-30, 31-40, 41-50, 51-60, 61-70, 70+.

```
SELECT age_group, COUNT(*) as purchases
FROM Insurance_Cold_Calls_with_AgeGroup
WHERE CarInsurance = 'yes'
GROUP BY age_group
ORDER BY purchases DESC
LIMIT 1;
```

4.2. Write a HiveQL query to find the correlation between the call duration and the success of the campaign.

SELECT corr(Duration, if(CarInsurance = 'yes', 1, 0)) as correlation FROM Insurance_Cold_Calls;

4.3. Write a HiveQL query to find out if the month of contact affects the success of the campaign. Provide the success rate for each month.

```
SELECT Month, AVG(if(CarInsurance = 'yes', 1, 0)) as
success_rate FROM Insurance_Cold_Calls GROUP BY Month;
```

Problem 5: Optimization

5.1. Partition the table by the outcome column and measure the query performance improvement.

```
CREATE TABLE Insurance_Cold_Calls_Partitioned (Id INT,
Age INT, ..., CarInsurance STRING)
PARTITIONED BY (Poutcome STRING);
INSERT OVERWRITE TABLE
Insurance_Cold_Calls_Partitioned PARTITION (Poutcome)
SELECT * FROM Insurance_Cold_Calls;
```

5.2. Write a HiveQL query to create a bucketed table by 'Education' with 4 buckets and load data into it.

```
CREATE TABLE Insurance_Cold_Calls_Bucketed (Id INT, Age
INT, ..., CarInsurance STRING)
CLUSTERED BY (Education) INTO 4 BUCKETS;
INSERT OVERWRITE TABLE
Insurance_Cold_Calls_Bucketed SELECT * FROM
Insurance_Cold_Calls;
```

Problem 6: Derived Insights

6.1. Write a HiveQL query to identify the top 5 professions that are most likely to buy insurance.

```
SELECT Job, COUNT(*) as Count
FROM Insurance_Cold_Calls
WHERE CarInsurance = 'yes'
GROUP BY Job
ORDER BY Count DESC
```

```
LIMIT 5;
```

6.2. Write a HiveQL query to find out the total calls made, grouped by the day of the week.

```
SELECT Day, COUNT(*) as Count
FROM Insurance_Cold_Calls
GROUP BY Day;
```

6.3. Write a HiveQL query to find out the hour of the day with the highest call success rate.

```
SELECT Hour, AVG(if(CarInsurance = 'yes', 1, 0)) as
success_rate
FROM Insurance_Cold_Calls
GROUP BY Hour
ORDER BY success_rate DESC
LIMIT 1;
```

Problem 7: Advanced Data Manipulation

7.1. Create a new column age_group in the Insurance_Cold_Calls table based on the age of the customer as follows: 18-30, 31-40, 41-50, 51-60, 61-70, 70+. Write a HiveQL query to achieve this.

```
CREATE TABLE Insurance_Cold_Calls_with_AgeGroup AS
SELECT *,
CASE
   WHEN Age BETWEEN 18 AND 30 THEN '18-30'
   WHEN Age BETWEEN 31 AND 40 THEN '31-40'
   WHEN Age BETWEEN 41 AND 50 THEN '41-50'
   WHEN Age BETWEEN 51 AND 60 THEN '51-60'
   WHEN Age BETWEEN 61 AND 70 THEN '61-70'
   ELSE '70+'
END AS age_group
FROM Insurance_Cold_Calls;
```

7.2. Write a HiveQL query to identify and replace any null or 'unknown' values in the job and education columns with the most frequent value in each column.

```
CREATE TABLE Insurance_Cold_Calls_Updated AS
SELECT
  CASE WHEN Job IS NULL OR Job = 'unknown' THEN (
    SELECT Job FROM (
      SELECT Job, COUNT(*) as cnt
      FROM Insurance_Cold_Calls
      WHERE Job IS NOT NULL AND Job != 'unknown'
      GROUP BY Job
      ORDER BY cnt DESC
      LIMIT 1
    ) tmp
  ) ELSE Job END as Job,
  CASE WHEN Education IS NULL OR Education = 'unknown'
THEN (
    SELECT Education FROM (
      SELECT Education, COUNT(*) as cnt
      FROM Insurance_Cold_Calls
      WHERE Education IS NOT NULL AND Education !=
'unknown'
      GROUP BY Education
      ORDER BY cnt DESC
      LIMIT 1
    ) tmp
  ) ELSE Education END as Education,
  -- Include the other columns as well
FROM Insurance_Cold_Calls;
```

7.3. Write a HiveQL query to delete records where the duration of the call is zero.

```
CREATE TABLE Insurance_Cold_Calls_Cleaned AS
SELECT *
FROM Insurance_Cold_Calls
WHERE Duration != 0;
```

GROW DATA SKILLS

Problem 8: Advanced Data Exploration

8.1. Write a HiveQL query to find the count of insurance purchased by each age group and sort them in descending order.

```
SELECT age_group, COUNT(*) as Count
FROM Insurance_Cold_Calls_with_AgeGroup
WHERE CarInsurance = 'yes'
GROUP BY age_group
ORDER BY Count DESC;
```

8.2. Write a HiveQL query to find out the number of calls made, grouped by the marital status and job of the customers.

```
SELECT Marital, Job, COUNT(*) as Count
FROM Insurance_Cold_Calls
GROUP BY Marital, Job;
```

8.3. Write a HiveQL query to find the customers (identify by customer numbers) who have been contacted more than twice and still did not purchase the insurance.

```
SELECT Id
FROM Insurance_Cold_Calls
WHERE Campaign > 2 AND CarInsurance = 'no';
```

Problem 9: Optimization & Performance

9.1. Compare the performance of a join operation before and after enabling map joins (hint: set hive.auto.convert.join to true).

9.2. Write a HiveQL query to create a view that includes customers' job, marital status, education, and outcome of the campaign. Use this view in subsequent queries and measure any performance improvements.

9.3. Test the performance of your Hive queries with different file formats (e.g., text file, SequenceFile, Avro, Parquet). Document your observations on read and write times.

The queries in this problem require comparing the performance of different operations in Hive, which requires running the queries and timing their execution in a live Hive environment.