Big Data Hadoop and Spark Developer

Certification Project



**Car Insurance Analysis**

**Objective:** To use the Hadoop features for data engineering or analysis of car insurance, share patterns, and actionable insights

**Problem Statement:**

A car insurance company wants to look at its historical data to understand and predict the probability of a customer making a claim based on multiple features other than MVR\_POINTS. The data set comprises of 10K plus submitted claim records 14 plus features. The scope of this project is limited to data engineering and analysis.

**Steps to perform:**

1. Create the following hive table as per the schema provided below:

TABLE: < Your own database>.CAR\_INSURANCE

|  |  |
| --- | --- |
| **Column Name** | **Datatype** |
| Policyid | LONGINT |
| Age | INT |
| Income | INT |
| Parent1 | String |
| Home\_value | INT |
| Married | String |
| Gender | String |
| Education | String |
| Occupation | String |
| Travel\_time | Int |
| Car\_usage | String |
| IDV | INT |
| Car\_type | String |
| Red\_car | String |
| Old\_claim | INT |
| Claim\_freq | INT |
| Mvr\_points | INT |
| Claim\_amount | INT |
| Car\_age | INT |
| Claim\_status | INT |
| Urbanicity | String |
| Car\_age | String |
| age\_category | String |

**Solution:**

create table rs\_bdhs.car\_insurance(policyid bigint, age int, cust\_age\_category string,income int, parent1 string, home\_value int, married string, gender string, education string, occupation string, travel\_time int, car\_usage string, IDV int, car\_type string, red\_car string, old\_claim int, claim\_freq int, mvr\_points int, claim\_amount int, car\_age int,car\_age\_category string, claim\_status int, urbanicity string);

1. Create a data pipeline using Sqoop, Flume to pull the table data below from the MYSQL server into the HBase table with one column family.

MYSQL DATABASE NAME: BDHS\_PROJECT

Table: Car\_insurance

**Solution:**

Step 1: Use sqoop to pull data from mysql table to local file system.

sqoop import -fs local -jt local --connect jdbc:mysql://ip-10-0-1-10.ec2.internal/bdhs\_project --username labuser --password simplilearn --table=car\_insurance --target-dir=file:///home/rakeshsrivastva75\_gmail/car\_insurance -m 1

Step 2: Create a table in HBase for storing car\_insurance data.

create\_namespace 'rs\_bdhs'

create 'rs\_bdhs:hbase\_car\_insurance','car\_ins\_details'

Step 3: Run the flume agent to read from sqoop imported directory and insert into HBase table created in step 2 above.

Agent configuration below:

car\_agent.sources = dirSource

car\_agent.sources.dirSource.type = spoolDir

car\_agent.sources.dirSource.spoolDir = /home/rakeshsrivastva75\_gmail/car\_insurance

car\_agent.channels = fileChannel

car\_agent.channels.fileChannel.type = file

car\_agent.sinks = hbaseSink

car\_agent.sinks.hbaseSink.type = hbase

car\_agent.sinks.hbaseSink.table=rs\_bdhs:hbase\_car\_insurance

car\_agent.sinks.hbaseSink.columnFamily=car\_ins\_details

car\_agent.sinks.hbaseSink.serializer=org.apache.flume.sink.hbase.SimpleHbaseEventSerializer

car\_agent.sources.dirSource.channels = fileChannel

car\_agent.sinks.hbaseSink.channel = fileChannel

Step 4: Running the agent.

flume-ng agent --conf-file car\_insurance\_flume\_agent.conf --name car\_agent

1. Create a staging Hive table pointing to the HBase table.

**Solution:**

Step 1: Create a table in Hive mapping to table in HBase.

CREATE EXTERNAL TABLE car\_insurance\_staging(key string, records string) STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH SERDEPROPERTIES ("hbase.columns.mapping" = "car\_ins\_details:pCol") TBLPROPERTIES ("hbase.table.name" = "rs\_bdhs:hbase\_car\_insurance", "hbase.mapred.output.outputtable" = "rs\_bdhs:hbase\_car\_insurance");

Step 2: Create a table new temp table in hive matching the structure of imported car insurance data.

create table rs\_bdhs.car\_insurance\_temp(policy\_id bigINT, DOB string, age INT, income string, parent1 string, home\_value string, married string, gender string, education string, occupation string, travel\_time INT, car\_usage string, IDV string, car\_type string, red\_car string, old\_claim string, claim\_freq INT, MVR\_points INT, claim\_amount string, car\_age INT, claim\_status INT, urbanicity string);

Step 3: Change the delimiter of this temp table to ','

ALTER TABLE rs\_bdhs.car\_insurance\_temp set serde 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe' WITH SERDEPROPERTIES ('field.delim' = ',');

Step 4: Read the data from car\_insurance\_staging table and create a data file under the hive folder of table created in the above step 3.

INSERT OVERWRITE DIRECTORY '/user/hive/warehouse/rs\_bdhs.db/car\_insurance\_temp' stored as textfile select records from car\_insurance\_staging;

1. Perform the following transformations before loading the data from staging table to the main table.
   * 1. Filter out the records where age is missing.
     2. Convert $values to integer values.
     3. Remove additional character (z\_) from all the fields wherever it is present.
     4. Perform feature engineering for age field and add a new field (cust\_age\_category) to the main Hive table as shown below:
        1. 16 to 18: Learner
        2. 19 to 40: Young
        3. 41 to 60: mid-age
        4. 60+: old-age
     5. Perform feature engineering for cust\_car\_age field and add a new field (car\_age\_category) to the main Hive table as shown below:
        1. 0 to 5: New
        2. 6 to 12: used
        3. 13 to 20: old
        4. 21+: vintage

**Solution (i,ii,iii):**

create table car\_insurance\_clean as select policy\_id, DOB, age,income,parent1,trim(regexp\_replace(home\_value,"\\$","")) home\_value,trim(regexp\_replace(married,"z\_","")) married,trim(regexp\_replace(gender,"z\_","")) gender,trim(regexp\_replace(education,"z\_","")) education,trim(regexp\_replace(occupation,"z\_","")) occupation,travel\_time, car\_usage,trim(regexp\_replace(IDV,"\\$","")) IDV,trim(regexp\_replace(car\_type,"z\_","")) car\_type,red\_car, trim(regexp\_replace(old\_claim,"\\$","")) old\_claim,claim\_freq,MVR\_points,trim(regexp\_replace(claim\_amount,"\\$","")) claim\_amount,car\_age,claim\_status,trim(regexp\_replace(urbanicity,"z\_","")) urbanicity from car\_insurance\_temp where age > 0;

**Solution (iv,v):**

use rs\_bdhs;

insert into car\_insurance select policy\_id, age, case when age between 16 and 18 then "Learner" when age between 19 and 40 then "young" when age between 41 and 60 then "mid-age" when age > 60 then "old-age" else "invalid age" end as cust\_age\_category,income ,parent1 ,home\_value ,married ,gender ,education ,occupation ,travel\_time ,car\_usage ,IDV ,car\_type ,red\_car ,old\_claim ,claim\_freq ,MVR\_points ,claim\_amount ,car\_age, case when car\_age between 0 and 5 then "New" when car\_age between 6 and 12 then "used" when car\_age between 13 and 20 then "old" else "vintage" end as car\_age\_category,claim\_status ,urbanicity from car\_insurance\_clean where car\_age >= 0;

1. Perform data analysis using Hive to display the repeated claim customers based on the age category, marital status, gender, and the urbanicity falling in top 75%.

**Solution:**

select cust\_age\_category,married,gender,urbanicity,cume\_dist() over (order by claim\_freq\_count) as claim\_dist from (select cust\_age\_category,married,gender,urbanicity,count(claim\_freq) claim\_freq\_count from car\_insurance group by cust\_age\_category,married,gender,urbanicity having count(claim\_freq) > 1) x order by claim\_dist desc;

**Output:**

age\_category married gender urbanicity claim\_dist

mid-age Yes F Highly Urban; Urban 1.0 mid-age Yes M Highly Urban; Urban 0.9615384615384616 mid-age No M Highly Urban; Urban 0.9230769230769231

mid-age No F Highly Urban; Urban 0.8846153846153846

young Yes F Highly Urban; Urban 0.8461538461538461

young No F Highly Urban; Urban 0.8076923076923077

1. Find the relation between car\_claim, car\_age, and car\_usage.

**Solution:**

select car\_age\_category,car\_usage,count(claim\_freq) as claim\_freq from car\_insurance where claim\_freq > 0 group by car\_age\_category,car\_usage order by claim\_freq desc;

**Output:**

+-------------------+-------------+-------------+--+ | car\_age\_category | car\_usage | claim\_freq | +-------------------+-------------+-------------+--+

| Used | Private | 890 |

| New | Private | 734 |

| Used | Commercial | 711 |

| New | Commercial | 539 |

| Old | Private | 519 |

| Old | Commercial | 307 |

| Vintage | Private | 35 |

| Vintage | Commercial | 19 |

+-------------------+-------------+-------------+--+

1. Find the top two car types with the maximum claim amounts.

**Solution:**

select car\_type, sum(claim\_amount) claim\_amt from car\_insurance group by car\_type order by claim\_amt desc limit 2;

**Output:**

+--------------+------------+--+ | car\_type | claim\_amt |

+--------------+------------+--+

| SUV | 4287058 |

| Pickup | 2830903 |

+--------------+------------+--+ 2 rows selected (32.723 seconds)

1. Display the average rate of claim approval for repeat claims vs. new claims.

**Solution:**

select new\_claim\_app.app\_count/(new\_claim\_app.app\_count+new\_claim\_rej.rej\_count) as new\_claim\_app\_rate, repeat\_claim\_app.app\_count/(repeat\_claim\_app.app\_count+repeat\_claim\_rej.rej\_count) as repeat\_claim\_app\_rate from (select count(1) as app\_count from car\_insurance where claim\_freq = 1 and claim\_status = 1) new\_claim\_app, (select count(1) as rej\_count from car\_insurance where claim\_freq = 1 and claim\_status = 0) new\_claim\_rej,(select count(1) as app\_count from car\_insurance where claim\_freq>1 and claim\_status =1) repeat\_claim\_app,(select count(1) as rej\_count from car\_insurance where claim\_freq > 1 and claim\_status = 0) repeat\_claim\_rej;

**Output:**

new\_claim\_app\_rate repeat\_claim\_app\_rate

0.3951137320977254 0.41137514608492404 Time taken: 121.518 seconds, Fetched: 1 row(s)