Web: Inceptez.com Mail: info@inceptez.com Call: 7871299810, 7871299817

Creditcard Defaulters and Insurance Premium Batch Project 2

Linux shell, script, python, Sqoop, HDFS, Hive, Hbase and Phoenix.

Important Note (Please read) :

This project is mainly created with Banking and Insurance domain data that majorly covers the INTERVIEW SCENARIOS (refer the interview documents attached for more reference).

If you spend your valuable time in executing this project reading each and every line of comments and rather than just copy paste the code and execute, for sure you can get a complete understanding of the realworld project with comprehensive knowledge in HDFS, Shell Scripting, Sqoop, Hive Analytical Queries, UDF Integration, Hbase and Phoenix.

Project Synopsis:

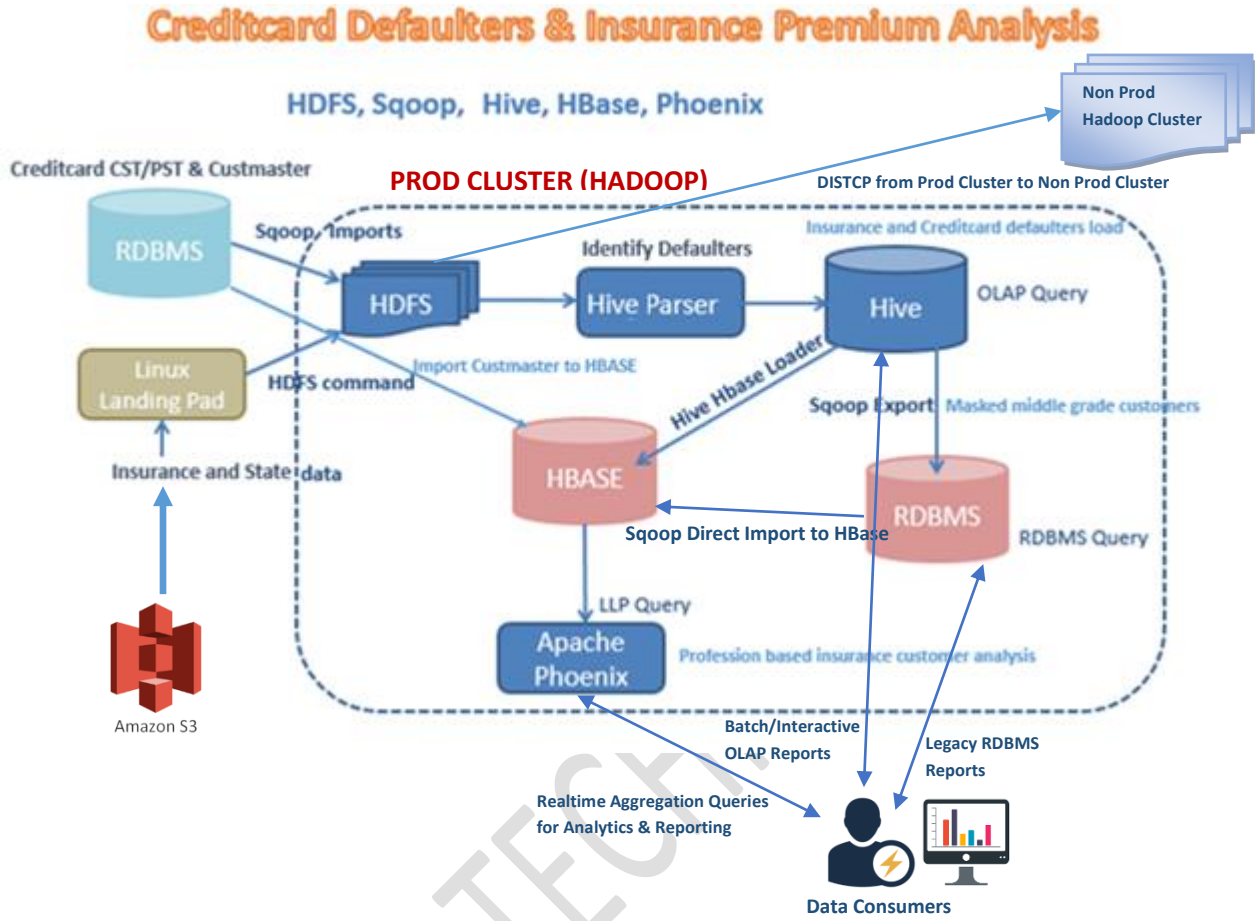
This project helps the Financial institutions such and Insurance and Banking systems understands the Creditcard defaulters to define the Insurance premium accordingly, it analyze a wide variety of online and offline customer data including the customer transactions, customer master data, insurance data etc. Bank can analyze this data to generate insights about individual consumer behaviors and preferences. This tool builds the strategies, Key Performance Indicators (KPI) definitions and implementation roadmaps that assists our esteemed clients in their Analytics & Information ecosystem journey right from Strategy definition to large scale Global Implementations & Support using the bigdata ecosystems.

As a part of Enterprise Data lake build, the data will be persisted into Hadoop file system, HBASE and Hive are injected using Sqoop for data exchange from DB sources, SFTP/SCP for file transfer from Cloud platform. Data transformation and processing such as aggregation, filtering, grouping using Hive for Batch data and Phoenix for online data access on Hbase for realtime data aggregation and reporting as per the reporting needs.

Synopsis:

This project module is to track and identify the creditcard defaulters and define the insurance premium according to the list of defaulters by joining data from insurance system, creditcard systems, state code fixed width data and the customer master datasets.

Architecture:



Project Flow:

1. Data ingestion and acquisition is done through Sqoop from RDBMS and into Linux file system from Cloud or any other sources.
2. Merge the 2 dataset using hive and split the defaulters and non-defaulters into 2 data sets and load into hdfs.
3. Create a hive table with header line count as 1.
4. Create one more fixed width hive table to load the fixed width states_fixed width data.
5. Create and load penality data into the hive table serialized with orc.
6. Export the Defaulters and Non Defaulters data into HDFS with comma delimiter, where non defaulters data added with trailer data for consumer systems validation.
7. Perform dist copy of non defaulters data from Prod cluster to non prod clusters that will be used for analytics purposes.
8. Run a shell script to pull the data from Cloud S3, validate, remove trailer data, move to HDFS and archive the data for backup
9. Create a partitioned table and run a shell script to which has to generate the load command and it has to load the data into the below insurance table automatically
10. Create a fixed width hive table to load the fixed width states_fixedwidth data using Regex Serde
11. Apply Data Governance - Redaction and Masking using Hive and Python
12. Data Provisioning to the Consumers into legacy Databases using Sqoop including Validation
13. Complex type ETL using Hive
14. Data movement & migration from DB to HBase using Sqoop
15. Data movement & migration from Hive to HBase using Storage Handler

16. Data Provisioning to the Consumers using Apache Phoenix to support Realtime aggregation and Low Latency Query processing
17. Write queries to build cubes at different levels to aggregate the data in realtime to populate in the report such as average age, sum of bill amount, average bill amount etc.,

Login to VMWare –
Prerequisites:
Start Hadoop, history server, hive in mr mode, mysql service, hive metastore, hbase and phoenix client.

Legend:

Blue Color – Represents descriptions
Red Color – Important points
Brown text – Represents Code, scripts, SQLs

Go to:
Player -> Manage -> Virtual Machine Settings -> change the memory as 3072 MB or upto 6208 MB and cpu core to 2

Extraction of Source code & data:

Download into /home/hduser and Untar the data provided in creditcard_insurance.tar.gz

```
cd ~
tar xvzf creditcard_insurance.tar.gz
```

DataSet:

All extracted scripts, datasets and other files will be in the given below location.

/home/hduser/creditcard_insurance

Start the Following services:

Hadoop:

```
start-all.sh
```

```
mr-jobhistory-daemon.sh start historyserver
```

Login, start mysql service and exit:

```
service mysqld start
```

```
sudo mysql -u root -p
```

password: root

Data preparation in the source systems

DB to Hadoop Data Ingestion:

Note: (This will be happening automatically in the source systems)

Insert into the mysql database - custmaster data in a table and the credit data into 2 tables of 2 timezones:

```
create database if not exists custdb;

use custdb;

drop table if exists credits_pst;

drop table if exists credits_cst;

drop table if exists custmaster;

create table if not exists credits_pst (id integer,lmt integer,sex integer,edu integer,marital integer,age integer,pay integer,billamt integer,defaulter integer,issuerid1 integer,issuerid2 integer,tz varchar(3));

create table if not exists credits_cst (id integer,lmt integer,sex integer,edu integer,marital integer,age integer,pay integer,billamt integer,defaulter integer,issuerid1 integer,issuerid2 integer,tz varchar(3));

create table if not exists custmaster (id integer,fname varchar(100),lname varchar(100),ageval integer,profession varchar(100));

source /home/hduser/creditcard_insurance/2_2_creditcard_defaulters_pst

source /home/hduser/creditcard_insurance/2_creditcard_defaulters_cst

source /home/hduser/creditcard_insurance/custmaster

sqoop import --connect jdbc:mysql://inceptez/custdb --username root --password root --table credits_cst --delete-target-dir --target-dir /user/hduser/credits_cst/ -m 1

sqoop import --connect jdbc:mysql://inceptez/custdb --username root --password root --table credits_pst --delete-target-dir --target-dir /user/hduser/credits_pst/ -m 1
```

Sqoop integration with Hive + ETL & ELT operations (Merge the 2 dataset using Hive and split the defaulters and non-defaulters into 2 data sets and load into hdfs.)

Sqoop and Hive integration with more options:

Import the creditcard datasets of cst and pst timezones into hive table with the below given steps.

1. ***Create a database in hive as insure.***
   ```
   create database if not exists insure;
   ```

2. ***Sqoop import the data into hive table insure.credits_pst and insure.credits_cst with options such as hive overwrite, number of mappers 2, where hive table created by sqoop with id as***

bigint, billamt as float (this is do able with –map-column-hive option check the cookboo/online/project solution document if you can't do it)

<<provide your solution here>>

ETL & ELT using Hive

3. **Filter the cst and pst tables with the billamt > 0 and union all both dataset and load into a single table called insure.cstpstreorder using create table as select (CTAS)**
   <<provide your solution here>>

4. ***Note: We can merge the step 2 and 3 and make the above 2 sqoop statements as 1 sqoop statement to directly import data into the insure.cstpstreorder by writing –query with billamt>0 filter and union of both credits_pst and credits_cst at the MYSQL level itself finally convert the insure.cstpstreorder into external table. If you have some time try it out..***

<<Try the solution here>>

5. **Create another external table insure.cstpstpenality in orc file format (show create table *insure.cstpstreorder*) to get the ddl of the above table and alter as per the below columns for faster table creation and populate with the below ETL transformations applied in the above table cstpstreorder as given below**

```
CREATE external TABLE `insure.cstpstpenality`(
  `id` bigint,
  `issuerid1` int,
  `issuerid2` int,
  `lmt` int,
  `newlmt` int,
  `sex` int,
  `edu` int,
  `marital` int,
  `pay` int,
  `billamt` float,
  `newbillamt` float,
  `defaulter` int)
  stored as orc
LOCATION
  '/user/hduser/cstpstpenality';
```

   a. **id,issuerid1,issuerid2,lmt,case defaulter when 1 then lmt-(lmt*0.04) else lmt end as newlmt ,sex,edu,marital,pay,billamt,case defaulter when 1 then billamt+(billamt*0.02) else billamt end as newbillamt, defaulter;**
      <<provide your solution here>>

Data Provisioning using Hive, Sqoop and DistCp

b. **Export and overwrite the above data into /user/hduser/defaultersout/ we will be using this defaultersout data in a later point of time and /user/hduser/nondefaultersout/ locations with defaulter=1 and defaulter=0 respectively using ',' delimiter. We will be sending this nondefaultersout data to external systems using Distcp/SFTP.**

<<provide your solution here>>

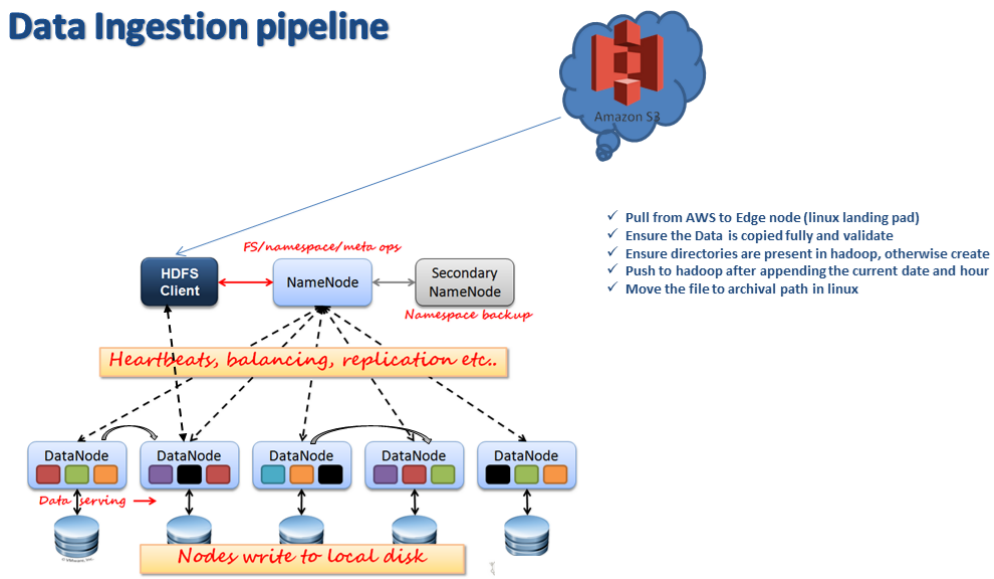
Data Provisioning to the Consumers using DistCP

Copy the data non defaulters data from one cluster (Prod) to another cluster (Non Prod) to /tmp/promocustomers location in hdfs for analytics purpose.

<<provide your solution here>>

Data preparation in the source systems

File system data source ingestion from Cloud using Linux Shell Script



Execute the below shell script to pull the data from Cloud S3, validate, remove trailer data, move to HDFS and archive the data for backup.

bash sfm_insuredata.sh https://s3.amazonaws.com/in.inceptez.bucket1/insurance_project/insuranceinfo.csv

<<provide your solution here by creating the sfm_insuredata.sh>>

Ensure data is imported from cloud to hdfs and get the date and take a note of the timestamp in the file

hadoop fs -ls /user/hduser/insurance_clouddata

ETL & ELT using Hive

Create a hive table with header line count as 1 and load the insurance dataset.

hive --service metastore

hive

create database if not exists insure;

use insure;

drop table if exists insurance;

CREATE TABLE insurance (IssuerId1 int,IssuerId2 int,BusinessYear int,StateCode string,SourceName string,NetworkName string,NetworkURL string,RowNumber int,MarketCoverage string,DentalOnlyPlan string)
row format delimited fields terminated by ','
TBLPROPERTIES **<<provide your solution here to add the property to ignore the header record>>**

load data inpath '/home/hduser/insurance_clouddata' into table insurance;

(Or)

Create a partitioned table and load manually or by modifying the below script which has to generate the load command and it has to load the data into the below insurance table automatically.

use insure;

CREATE TABLE insurance (IssuerId1 int,IssuerId2 int,BusinessYear int,StateCode string,SourceName string,NetworkName string,NetworkURL string,RowNumber int,MarketCoverage string,DentalOnlyPlan string)
Partitioned by (datadt date,hr int)
row format delimited fields terminated by ','
TBLPROPERTIES **<<provide your solution here to add the property to ignore the header record>>;**

bash /home/hduser/creditcard_insurance/hivepart.sh /user/hduser/insurance_clouddata insure.insurance creditcard_insurance

<<provide your solution here by creating the hivepart.sh script to create the below load commands based on the cloud imported data>>

~~load data inpath '/user/hduser/insurance_clouddata/creditcard_insurance_2020091612' overwrite into table insurance partition(datadt='2020-09-16',hr=12);~~

Delete the invalid data with null issuerid1 and issuerid2 using insert select query

<<provide your solution here to add the property to ignore the header record>>

Create a fixed width hive table to load the fixed width states_fixedwidth data using Regex Serde

```
drop table if exists insure.state_master;

CREATE EXTERNAL TABLE insure.state_master (statecd STRING, statedesc STRING)
ROW FORMAT SERDE 'org.apache.hadoop.hive.contrib.serde2.RegexSerDe'
WITH SERDEPROPERTIES ("<<provide your solution here to add the property to manage fixed width data>>" )

LOCATION '/user/hduser/states';

load data local inpath '/home/hduser/creditcard_insurance/states_fixedwidth' overwrite into table insure.state_master;
```

Create a managed table on top of the hive output defaulter's dataset created above.

```
use insure;

CREATE TABLE defaulters (id int,IssuerId1 int,IssuerId2 int,lmt int,newlmt double,sex int,edu int,marital int,pay int,billamt int,newbillamt float,defaulter int)
row format delimited fields terminated by ','
LOCATION '/user/hduser/defaultersout';
```

Create a final managed table (later convert to external table) in orc with snappy compression and load the above 2 tables joined by applying different functions as given below .

This table should not allow duplicates when it is empty or if not using overwrite option.

```
use insure;

CREATE TABLE insurancestg (IssuerId int,BusinessYear int,StateCode string,statedesc string,SourceName string,NetworkName string,NetworkURL string,RowNumber int,MarketCoverage string,DentalOnlyPlan string,id int,lmt int,newlmt int,reduced_lmt int,sex varchar(6),grade varchar(20),marital int,pay int,billamt int,newbillamt float,penality float,defaulter int)
row format delimited fields terminated by ','
LOCATION '/user/hduser/insurancestg';

insert overwrite table insurancestg select concat(i.IssuerId1,i.IssuerId2) as issuerid,i.businessyear,i.statecode,s.statedesc as statedesc,i.sourcename,i.networkname,i.networkurl,i.rownumber,i.marketcoverage,i.dentalonlyplan,d.id,d.lmt,d.newlmt,d.newlmt-d.lmt as reduced_lmt,case when d.sex=1 then 'male' else 'female' end as sex ,case when d.edu=1 then 'lower grade' when d.edu=2 then 'lower middle grade' when d.edu=3 then 'middle grade' when d.edu=4 then 'higher grade' when d.edu=5 then 'doctrate grade' end as grade ,d.marital,d.pay,d.billamt,d.newbillamt,d.newbillamt-d.billamt as penality,d.defaulter
from insurance i inner join defaulters d
```

```
on (i.IssuerId1=d.IssuerId1 and i.IssuerId2=d.IssuerId2)
inner join state_master s
on (i.statecode=s.statecd);


dfs -rm -r -f /user/hduser/insuranceorc;

drop table if exists insuranceorc;


CREATE TABLE insuranceorc (IssuerId int,BusinessYear int,StateCode string,statedesc string,SourceName
string,NetworkName string,NetworkURL string,RowNumber int,MarketCoverage string,DentalOnlyPlan
string,id int,lmt int,newlmt int,reduced_lmt int,sex varchar(6),grade varchar(20),marital int,pay
int,billamt int,newbillamt float,penality int,defaulter int)
row format delimited fields terminated by ','
<<provide your solution here to create orc table>>

LOCATION '/user/hduser/insuranceorc'

TBLPROPERTIES (<<provide your solution here to make this table immutable and snappy
compressed>>);


Insert into insuranceorc select *  from insurancestg where issuerid is not null;
```

Retry running the above same insert query once again and see what happens??

If you get the below error then drop the above table and recreate and insert.

FAILED: SemanticException [Error 10256]: Inserting into a non-empty immutable table is not allowed insuranceorc

Convert the above table from managed to external, usually we use the below statement if we can't create external table in the initial stage itself for example sqoop import hive table can't be created as external initially.

<<provide your solution here>>

write common table expression queries in hive

```
with T1 as ( select max(penality) as penalitymale from insuranceorc where sex='male'),
T2 as ( select max(penality) as penalityfemale from insuranceorc where sex='female')
select penalitymale,penalityfemale
from T1 inner join T2
ON 1=1;
```

Data Governance - Redaction and Masking using Hive and Python

create view in hive to restrict few columns, store queries, and apply some masking on sensitive columns using either query or by using the mask_insure.py function given in the project document.

```
drop view if exists middlegradeview;

create view middlegradeview as
select issuerid,businessyear,statedesc,sourcename, sex,grade,marital,newbillamt,defaulter
,translate(translate(translate(translate(translate(networkurl,'a','x'),'b','y'),'c','z'),'s','r'),'.com','.aaa') as
maskednetworkurl
from insuranceorc
where grade='middle grade'
and issuerid is not null;
```

(or)

```
create view middlegradeview as
select transform(issuerid,businessyear,statedesc,sourcename, defaulter ,networkurl) using 'python
/home/hduser/mask_insure.py'
as (issuerid,businessyear,statedesc,sourcename, defaulter ,maskednetworkurl)
from insuranceorc
where grade='middle grade'
and issuerid is not null;

select * from middlegradeview;
```

<<provide your solution here to create the mask_insure.py python script>>

Export the above view data into hdfs location /user/hduser/defaulterinfo with the pipe '|' delimiter the following columns
issuerid,businessyear,statedesc,sourcename,maskednetworkurl,sex,grade,marital,newbillamt,defaulter

<<provide your solution here>>

Data Provisioning to the Consumers into legacy Databases using Sqoop including Validation

Data export using Sqoop into DB

Export the above data into mysql using sqoop

```
mysql -u root -p
password: root

use custdb;
drop table if exists middlegrade;
```

create table middlegrade (issuerid integer,businessyear integer,maskedstatedesc varchar(200),maskedsourcename varchar(100), defaulter varchar(20),maskednetworkurl varchar(200));

quit;

1. **Export the masked data into mysql using sqoop export as per the above table structure.**
2. **Validate the export is properly happened using --validate option in sqoop**

<<provide your solution here>>

Complex type ETL using Hive

Create a Complex type table to understand how to group the like issuers in a single row using array, struct and map.

Drop table if exists insuranceorc_collection ;

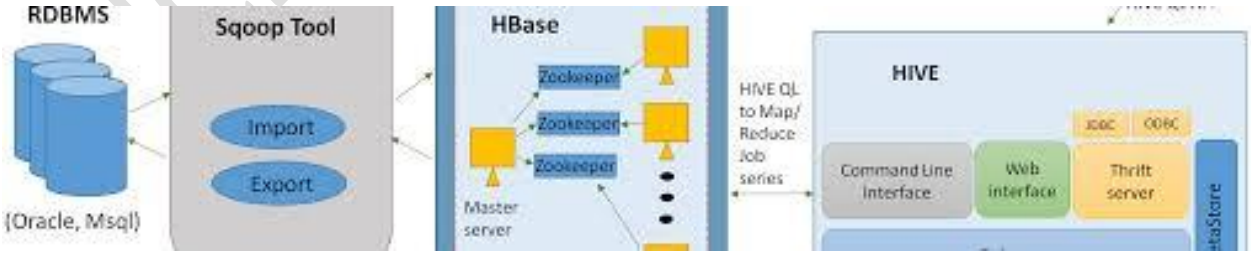
Create table insuranceorc_collection
row format delimited collection items terminated by ', '
stored as orcfile
location '/user/hduser/insuranceorc_collection/'
as select issuerid,named_struct('marital',marital,'sex',sex,'grade',grade) as personalinfo,collect_set(networkname) as networkname, collect_set(networkurl) networkurl
from insuranceorc group by issuerid,named_struct('marital',marital,'sex',sex,'grade',grade);

alter table insuranceorc_collection SET TBLPROPERTIES('EXTERNAL'='TRUE');

Select only the issuerid,grade,second networkname,second networkurl where we have more than 1 networkname and networkurl accessed by the customers.

<<provide your solution here>>

Data movement & migration from DB to HBase using Sqoop



Hive HBase handler data load (~~Striked out which ever is just for reference~~)

Join insurance and credit card data and load into hbase table created with 2 column families credit and insurance.

~~Copy the jars to the hive lib directory from hbase~~

~~**Add the below line in hive-env.sh to locate hbase lib path as auxiliary hive jar path to use hbase jars**~~

~~cd /usr/local/hive/conf/~~
~~mv hive-env.sh.template hive-env.sh~~

~~vi hive-env.sh~~

~~export HIVE_AUX_JARS_PATH=/usr/local/hbase/lib~~

~~**Copy the jars to the hive lib directory from hbase**~~

~~cp /usr/local/hbase/lib/hbase-common-0.98.4-hadoop2.jar /usr/local/hive/lib/~~
~~cp /usr/local/hbase/lib/zookeeper-3.4.6.jar /usr/local/hive/lib/~~
~~cp /usr/local/hbase/lib/guava-12.0.1.jar /usr/local/hive/lib/~~
~~cp /usr/local/hbase/lib/hbase-protocol-0.98.4-hadoop2.jar /usr/local/hive/lib/~~
~~cp /usr/local/hbase/lib/hbase-server-0.98.4-hadoop2.jar /usr/local/hive/lib/~~

~~**Start Zookeeper and Hbase:**~~

~~zkServer.sh start~~
~~start-hbase.sh~~

hbase shell

create 'custmaster', 'customer'

quit

1. **Import using sqoop from db into hbase custmaster data and save the output log into a log file.**

<<provide your solution here>>

~~**(OR) - run as a free form query just for syntax purpose,**~~

~~sqoop import --connect jdbc:mysql://inceptez/custdb --username root --password root --query "select concat(id,'-',ageval) as id ,lname,fname,ageval,profession from custmaster where \$CONDITIONS" --hbase-table custmaster --column-family customer --hbase-row-key id -m 1~~

2. **Validate the sqoop import is properly happened using --validate option in sqoop and create a _SUCCESS file if the validation is successful in /home/hduser/creditcard_insurance location using linux commands.**

<<provide your solution here>>

Data movement & migration from Hive to HBase using Storage Handler

Create a hbase handler table in hive using hbase storage handler referring to insurancehive table that will be automatically created in hbase with insurance and credit card column families when we create the below hive table.

```
Use insure;
drop table if exists insurancehive;

CREATE TABLE insurancehive (idkey int, issuerid int,id int,businessyear int,statedesc string,networkurl string,pay int,defaulter string,billamt int,newbillamt float,penality float)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH SERDEPROPERTIES ("hbase.columns.mapping" =
":key,insurance:issuerid,insurance:id,insurance:businessyear,insurance:statedesc,insurance:networkurl,creditcard:pay,creditcard:defaulter,creditcard:billamt,creditcard:newbillamt,creditcard:penality")
TBLPROPERTIES ("hbase.table.name" = "insurancehive",
"hbase.mapred.output.outputtable"="insurancehive");
```

Insert incremental rowkey as the row_key into HBASE.

```
insert into table insurancehive select row_number() over() as idkey,issuerid,id,businessyear,statedesc,networkurl,pay,defaulter,billamt,newbillamt,penality
from insuranceorc where issuerid is not null;
```

Data Provisioning to the Consumers using Apache Phoenix to support Realtime aggregation and Low Latency Query processing

Create a phoenix table view on the above hbase table and analyze profession based total payment and average payment.

```
sqlline.py localhost

!set maxwidth 1000

drop view if exists "insurancehive";
```

```
create view "insurancehive" (idkey varchar(100) primary key,"insurance"."issuerid"
varchar,"insurance"."id" varchar,"insurance"."businessyear" varchar,"insurance"."statedesc"
varchar,"insurance"."networkurl" varchar,"creditcard"."pay" varchar,"creditcard"."defaulter"
varchar,"creditcard"."billamt" varchar,"creditcard"."newbillamt" varchar,"creditcard"."penality"
varchar);

drop view if exists "custmaster";

create view "custmaster" (id varchar primary key,"customer"."fname"
varchar,"customer"."lname" varchar,"customer"."profession" varchar,"customer"."ageval" varchar);
```

Write queries to build cubes at different levels to aggregate the data in realtime to populate in the report such as average age, sum of bill amount, average bill amount etc.,

```
select next value for cubeseq as id,lvl, avg_age, sum_billamt, avg_billamt, avg_newbillamt from (
select 1 as lvl,"profession",avg(cast(to_number("ageval") as integer)) as avg_age ,0 as sum_billamt,0 as
avg_billamt,0 as avg_newbillamt
from "custmaster"
group by lvl,"profession"
union all
select 2 as lvl,"profession", cast(to_number("ageval") as integer) as avg_age ,0 as sum_billamt,0 as
avg_billamt,0 as avg_newbillamt
from "custmaster"
union all
select 3 as lvl,"custmaster"."profession",0 as avg_age,sum(cast(to_number("insurancehive"."billamt") as
bigint)) as sum_billamt,avg(cast(to_number("insurancehive"."newbillamt") as double)) as avg_billamt,
avg(cast(to_number("insurancehive"."newbillamt") as double)) as avg_newbillamt
from "insurancehive" as i inner join "custmaster" as c
on "insurancehive"."id"=c.id
group by "custmaster"."profession") as temp
order by lvl;
```

Practice with more Queries referring Phoenix site documentation….