sqoop import --connect jdbc:mysql://zeyod2.czgbk94pezaa.ap-south-1.rds.amazonaws.com/itv004068 --username root --password Aditya908 --table cust --m 1 --where "location='chennai' and mode='cash'" --columns name,location --delete-target-dir --target-dir /user/itv004068/chennaidir

hadoop fs -ls /user/itv004068/chennaidir

hadoop fs -cat /user/itv004068/chennaidir/part-m-00000

2 task

sqoop import --connect jdbc:mysql://zeyod2.czgbk94pezaa.ap-south-1.rds.amazonaws.com/zdb --username root --password Aditya908 --query "select a.\*,b.location,b.mode from c1 a join c2 b on a.id=b.id where \$CONDITIONS" --m 1 --delete-target-dir --target-dir /user/itv004068/joindir

hadoop fs -ls /user/itv004068/joindir

hadoop fs -cat /user/itv004068/joindir/part-m-00000

Lab Incremental Folks

=========

mysql --host=zeyod2.czgbk94pezaa.ap-south-1.rds.amazonaws.com --user=root --password=Aditya908

create database itv004068;

use itv004068;

create table custi(id int,name varchar(100),location varchar(100),mode varchar(100));

insert into custi values(1,'zeyo','chennai','cash');

insert into custi values(2,'hema','hyderabad','credit');

insert into custi values(3,'haas','chennai','credit');

insert into custi values(4,'vasu','bangalore','cash');

insert into custi values(5,'ravi','hyderabad','cash');

select \* from custi;

quit

sqoop import --connect jdbc:mysql://zeyod2.czgbk94pezaa.ap-south-1.rds.amazonaws.com/itv004068 --username root --password Aditya908 --table custi --m 1 --delete-target-dir --target-dir /user/itv004068/indir

hadoop fs -ls /user/itv004068/indir

hadoop fs -cat /user/itv004068/indir/part-m-00000

=====

add data

=====

mysql --host=zeyod2.czgbk94pezaa.ap-south-1.rds.amazonaws.com --user=root --password=Aditya908

use itv004068;

insert into custi values(6,'rani','bangalore','cash');

insert into custi values(7,'vinu','hyderabad','cash');

select \* from custi;

quit

sqoop import --connect jdbc:mysql://zeyod2.czgbk94pezaa.ap-south-1.rds.amazonaws.com/itv004068 --username root --password Aditya908 --table custi --m 1 --target-dir /user/itv004068/indir --incremental append --check-column id --last-value 5

hadoop fs -ls /user/itv004068/indir

hadoop fs -cat /user/itv004068/indir/part-m-00001

sqoop import --connect jdbc:mysql://zeyod2.czgbk94pezaa.ap-south-1.rds.amazonaws.com/itv004068 --username root --password Aditya908 --table cust --m 1 --where "location='chennai' and mode='cash'" --columns name,location --delete-target-dir --target-dir /user/itv004068/chennaidir

hadoop fs -ls /user/itv004068/chennaidir

hadoop fs -cat /user/itv004068/chennaidir/part-m-00000

======================

Lab Job

======================

mysql --host=zeyod2.czgbk94pezaa.ap-south-1.rds.amazonaws.com --user=root --password=Aditya908

create database itv004068;

use itv004068;

create table custij(id int,name varchar(100),location varchar(100),mode varchar(100));

insert into custij values(1,'zeyo','chennai','cash');

insert into custij values(2,'hema','hyderabad','credit');

insert into custij values(3,'haas','chennai','credit');

insert into custij values(4,'vasu','bangalore','cash');

insert into custij values(5,'ravi','hyderabad','cash');

select \* from custij;

quit

sqoop job --create injob -- import --connect jdbc:mysql://zeyod2.czgbk94pezaa.ap-south-1.rds.amazonaws.com/itv004068 --username root --password Aditya908 --table custij --m 1 --target-dir /user/itv004068/jobdir --incremental append --check-column id --last-value 0

sqoop job --list

sqoop job --exec injob =====give password as Aditya908

hadoop fs -ls /user/itv004068/jobdir

hadoop fs -cat /user/itv004068/jobdir/part-m-00000

mysql --host=zeyod2.czgbk94pezaa.ap-south-1.rds.amazonaws.com --user=root --password=Aditya908

use itv004068;

insert into custij values(6,'rani','bangalore','cash');

insert into custij values(7,'vinu','hyderabad','cash');

select \* from custij;

quit

sqoop job --exec injob =====give password as Aditya908

hadoop fs -ls /user/itv004068/jobdir

hadoop fs -cat /user/itv004068/jobdir/part-m-00001

cat /home/itv004068/.sqoop/metastore.db.script | grep 'last.value'

----------

01-10-2022

----------

Task 1

==============

\*Cloudera Password File\*

==============

cd

echo -n cloudera>pfile

mysql -uroot -pcloudera

create database datap;

use datap;

create table intab (id int,name varchar(100),amt int);

insert into intab values(1,'zeyo',4);

insert into intab values(2,'sai',5);

insert into intab values(3,'vasu',7);

select \* from intab;

quit

sqoop job --create pjob -- import --connect jdbc:mysql://localhost/datap --username root --password-file file:///home/cloudera/pfile --m 1 --table intab --target-dir /user/cloudera/datap --incremental append --check-column id --last-value 0

sqoop job --list

sqoop job --exec pjob //-- it will not ask the password

hadoop fs -ls /user/cloudera/datap

hadoop fs -cat /user/cloudera/datap/part-m-00000

mysql -uroot -pcloudera

use datap;

insert into intab values(4,'rani',5);

insert into intab values(5,'hema',7);

select \* from intab;

quit

sqoop job --exec pjob //-- it will not ask the password

hadoop fs -ls /user/cloudera/datap

hadoop fs -cat /user/cloudera/datap/part-m-00001

==============

\*lab Password File\*

==============

cd

echo -n Aditya908>pfile

mysql --host=zeyod2.czgbk94pezaa.ap-south-1.rds.amazonaws.com --user=root --password=Aditya908

create database itv004068;

use itv004068;

create table intab (id int,name varchar(100),amt int);

insert into intab values(1,'zeyo',4);

insert into intab values(2,'sai',5);

insert into intab values(3,'vasu',7);

select \* from intab;

quit

sqoop job --create pjob -- import --connect jdbc:mysql://zeyod2.czgbk94pezaa.ap-south-1.rds.amazonaws.com/itv004068 --username root --password-file file:///home/itv004068/pfile --m 1 --table intab --target-dir /user/itv004068/datap --incremental append --check-column id --last-value 0

sqoop job --list

sqoop job --exec pjob //-- it will not ask the password

hadoop fs -ls /user/itv004068/datap

hadoop fs -cat /user/itv004068/datap/part-m-00000

mysql --host=zeyod2.czgbk94pezaa.ap-south-1.rds.amazonaws.com --user=root --password=Aditya908

use itv004068;

insert into intab values(4,'rani',5);

insert into intab values(5,'hema',7);

select \* from intab;

quit

sqoop job --exec pjob

hadoop fs -ls /user/itv004068/datap

hadoop fs -cat /user/itv004068/datap/part-m-00001

-------

Task 2

--------

Cloudera Folks

==============

mysql -uroot -pcloudera

create database datap1;

use datap1;

create table intab (id int,name varchar(100),amt int);

insert into intab values(1,'zeyo',4);

insert into intab values(2,'sai',5);

insert into intab values(3,'vasu',7);

select \* from intab;

quit

sqoop import -Dfs.s3a.access.key=AKIAV6SZ74UIVD7ZWL4U -Dfs.s3a.secret.key=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 -Dfs.s3a.endpoint=s3.ap-south-1.amazonaws.com --connect jdbc:mysql://localhost/datap1 --username root --password cloudera --table intab --m 1 --target-dir s3a://skbuck/itv004068

---------

Lab Folks

---------

cd

rm -rf awscli-bundle.zip

curl https://s3.amazonaws.com/aws-cli/awscli-bundle-1.16.188.zip -o awscli-bundle.zip

unzip awscli-bundle.zip

./awscli-bundle/install -i /home/itv004068/aws -b /home/itv004068/bin/aws

aws=/home/itv004068/bin/aws

cd

rm -rf .aws

mkdir .aws

cd .aws

wget https://skbuck.s3.amazonaws.com/credentials

cd

======

To validate data

======

aws s3 ls s3://skbuck/

Cloudera To validate data

cd

rm -rf .aws

mkdir .aws

cd .aws

wget https://skbuck.s3.amazonaws.com/credentials

cd

======

To validate data

======

aws s3 ls s3://skbuck/

===================

02-10-2022 12 class

===================

=====================

Cloudera File formats

=====================

cd

mysql -uroot -pcloudera

create database add1;

use add1;

create table ttab(id int,name varchar(100),amount int);

insert into ttab values(1,'zeyo',40);

insert into ttab values(2,'vasu',50);

insert into ttab values(3,'rani',70);

select \* from ttab;

quit

sqoop import --connect jdbc:mysql://localhost/add1 --username root --password cloudera --table ttab --m 1 --delete-target-dir --target-dir /user/cloudera/pdata --as-parquetfile

hadoop fs -ls /user/cloudera/pdata/

hadoop fs -cat /user/cloudera/pdata/\*

===============

Lab File format

===============

mysql --host=zeyod2.czgbk94pezaa.ap-south-1.rds.amazonaws.com --user=root --password=Aditya908

create database itv004068;

use itv004068;

create table ttab(id int,name varchar(100),amount int);

insert into ttab values(1,'zeyo',40);

insert into ttab values(2,'vasu',50);

insert into ttab values(3,'rani',70);

select \* from ttab;

quit

sqoop import --connect jdbc:mysql://zeyod2.czgbk94pezaa.ap-south-1.rds.amazonaws.com/itv004068 --username root --password Aditya908 --table ttab --m 1 --delete-target-dir --target-dir /user/itv004068/pdata --as-parquetfile

hadoop fs -ls /user/itv004068/pdata

hadoop fs -cat /user/itv004068/pdata/

'Task 1 --- AVRO imports

=========================

Cloudera Folks

cd

mysql -uroot -pcloudera

create database add1;

use add1;

create table ttab2(id int,name varchar(100),amount int);

insert into ttab2 values(1,'zeyo',40);

insert into ttab2 values(2,'vasu',50);

insert into ttab2 values(3,'rani',70);

select \* from ttab2;

quit

sqoop import --connect jdbc:mysql://localhost/add1 --username root --password cloudera --table ttab2 --m 1 --delete-target-dir --target-dir /user/cloudera/adata --as-avrodatafile

hadoop fs -ls /user/cloudera/adata

hadoop fs -cat /user/cloudera/adata/\*

=========

Lab Folks

=========

mysql --host=zeyod2.czgbk94pezaa.ap-south-1.rds.amazonaws.com --user=root --password=Aditya908

create database itv004068;

use itv004068;

create table ttab2(id int,name varchar(100),amount int);

insert into ttab2 values(1,'zeyo',40);

insert into ttab2 values(2,'vasu',50);

insert into ttab2 values(3,'rani',70);

select \* from ttab2;

quit

sqoop import -Dmapreduce.job.user.classpath.first=true --connect jdbc:mysql://zeyod2.czgbk94pezaa.ap-south-1.rds.amazonaws.com/itv004068 --username root --password Aditya908 --table ttab --m 1 --delete-target-dir --target-dir /user/itv004068/adata --as-avrodatafile

hadoop fs -ls /user/itv004068/adata

hadoop fs -cat /user/itv004068/adata/\*

Task 2 --- Take Notes or In paint

=================================

Put in paint or Nodes (Compression and Notes)

==============

15-10-2022

==============

Cloudera Folks

==============

Type hive and enter

create database zeyodb;

!hadoop fs -ls /user/hive/warehouse;

use zeyodb;

create table atab(id int);

!hadoop fs -ls /user/hive/warehouse/zeyodb.db;

=========

Lab Folks

=========

hive -e "SET hive.metastore.warehouse.dir = /user/itv004068/warehouse;drop database itv004068 cascade;create database itv004068;use itv004068;create table atab(id int);"

hadoop fs -ls /user/itv004068/warehouse

hadoop fs -ls /user/itv004068/warehouse/itv004068.db

========

Cloudera

========

cd

echo 1,zeyo>data.csv

echo 2,vish>>data.csv

hive

create database if not exists zeyodb;

use zeyodb;

create table datatab (id int,name string) row format delimited fields terminated by ',';

load data local inpath '/home/cloudera/data.csv' into table datatab;

select \* from datatab;

=========

Lab Folks

=========

cd

echo 1,zeyo>data.csv

echo 2,vish>>data.csv

hive -e "SET hive.metastore.warehouse.dir = /user/<itv004068>/warehouse;create database if not exists itv004068;use itv004068;

create table datatab (id int,name string) row format delimited fields terminated by ',';load data local inpath '/home/itv004068/data.csv' into table datatab;select \* from datatab;"

======

Task 1-

======

==============

Cloudera Folks

==============

cd

echo 1,zeyo>data.csv

echo 2,vish>>data.csv

hadoop fs -put data.csv /user/cloudera/

hive

create database if not exists zeyodb1;

use zeyodb1;

create table datatab (id int,name string) row format delimited fields terminated by ',';

load data inpath '/user/cloudera/data.csv' into table datatab;

select \* from datatab;

=========

Lab Folks

=========

cd

echo 1,zeyo>data.csv

echo 2,vish>>data.csv

hive -e "SET hive.metastore.warehouse.dir = /user/itv004068/warehouse;create database if not exists itv004068;use itv004068;

create table datatab1 (id int,name string) row format delimited fields terminated by ',';load data inpath '/user/itv004068/data.csv' into table datatab1;select \* from datatab1;"

ERROR

=================

Updated Lab Folks

=================

cd

echo 1,zeyo>data.csv

echo 2,vish>>data.csv

hadoop fs -put data.csv /user/itv004068/

hive -e "SET hive.metastore.warehouse.dir = /user/itv004068/warehouse;create database if not exists itv004068;use itv004068;

create table datatab1 (id int,name string) row format delimited fields terminated by ',';load data inpath '/user/itv004068/data.csv' into table datatab1;select \* from datatab1;"

ERROR

=================

Updated Lab Folks

=================

cd

echo 1,zeyo>data.csv

echo 2,vish>>data.csv

hadoop fs -put data.csv /user/itv004068/

hive -e "SET hive.metastore.warehouse.dir = /user/itv004068/warehouse;create database if not exists itv004068;use itv004068;drop table datatab1;

create table datatab1 (id int,name string) row format delimited fields terminated by ',';load data inpath '/user/itv004068/data.csv' into table datatab1;select \* from datatab1;"

insert overwrite local directory '/home/carter/staging' row format delimited fields terminated by ',' select \* from hugetable;

==========

16-10-2022

==========

==============

Cloudera Folks

==============

cd

echo 1,sai>data.csv

echo 2,zeyo>>data.csv

hadoop fs -mkdir /user/cloudera/ddir

hadoop fs -put -f data.csv /user/cloudera/ddir/

hadoop fs -mkdir /user/cloudera/rdir

hive

create table ltab(id int,name string) row format delimited fields terminated by ',' location '/user/cloudera/ddir';

select \* from ltab;

create table rtab(id int,name string) row format delimited fields terminated by ',' location '/user/cloudera/rdir';

insert into rtab select \* from ltab where id>1;

select \* from rtab;

!hadoop fs -ls /user/cloudera/rdir;

!hadoop fs -cat /user/cloudera/rdir/\*;

=========

Lab Folks

=========

cd

echo 1,sai>data.csv

echo 2,zeyo>>data.csv

hadoop fs -mkdir /user/itv004068/ddir

hadoop fs -put -f data.csv /user/itv004068/ddir/

hadoop fs -mkdir /user/itv004068/rdir

hive -e "SET hive.metastore.warehouse.dir = /user/itv004068/warehouse;create database if not exists itv004068;use itv004068;

create table ltab(id int,name string) row format delimited fields terminated by ',' location '/user/itv004068/ddir';select \* from ltab;create table rtab(id int,name string) row format delimited fields terminated by ',' location '/user/itv004068/rdir';insert into rtab select \* from ltab where id>1;select \* from rtab;\*"

hadoop fs -ls /user/itv004068/rdir

hadoop fs -cat /user/itv004068/rdir/

==============

Cloudera Folks

==============

cd

echo 1,sai>data.csv

echo 2,zeyo>>data.csv

hadoop fs -mkdir /user/cloudera/mdir

hadoop fs -mkdir /user/cloudera/edir

hadoop fs -put data.csv /user/cloudera/mdir

hadoop fs -put data.csv /user/cloudera/edir

hive

create table mtab(id int,name string) row format delimited fields terminated by ',' location '/user/cloudera/mdir';

select \* from mtab;

create external table etab(id int,name string) row format delimited fields terminated by ',' location '/user/cloudera/edir';

select \* from etab;

!hadoop fs -ls /user/cloudera/;

drop table mtab;

drop table etab;

!hadoop fs -ls /user/cloudera/;

=========

Lab Folks

=========

cd

echo 1,sai>data.csv

echo 2,zeyo>>data.csv

hadoop fs -mkdir /user/itv004068/mdir

hadoop fs -mkdir /user/itv004068/edir

hadoop fs -put data.csv /user/itv004068/mdir

hadoop fs -put data.csv /user/itv004068/edir

hive -e "SET hive.metastore.warehouse.dir = /user/itv004068/warehouse;create database if not exists itv004068;use itv004068;create table mtab(id int,name string) row format delimited fields terminated by ',' location '/user/itv004068/mdir';select \* from mtab;create external table etab(id int,name string) row format delimited fields terminated by ',' location '/user/itv004068/edir';select \* from etab;drop table mtab;drop table etab;"

hadoop fs -ls /user/itv004068/

hadoop fs -ls /user/itv004068/

Thank you for your business! We look forward to working with you again.

========

Task 1 ----

========

Cloudera Folks

cd

echo 1,sai>data.csv

echo 2,zeyo>>data.csv

hadoop fs -mkdir /user/cloudera/emdir

hadoop fs -put data.csv /user/cloudera/emdir

hive

create database tcheck;

use tcheck;

create table mtab1(id int,name string) row format delimited fields terminated by ',' location '/user/cloudera/emdir';

create external table etab1(id int,name string) row format delimited fields terminated by ',' location '/user/cloudera/emdir';

drop table etab1;

select \* from mtab1 ; ----> u will see the data

create external table etab1(id int,name string) row format delimited fields terminated by ',' location '/user/cloudera/emdir';

drop table mtab1;

select \* from etab1 ; ----> u will not see the data

!hadoop fs -ls /user/cloudera/; -----> U will not see the directory emdir

=========

Lab Folks

=========

cd

echo 1,sai>data.csv

echo 2,zeyo>>data.csv

hadoop fs -mkdir /user/itv004068/emdir

hadoop fs -put data.csv /user/itv004068/emdir

hive -e "SET hive.metastore.warehouse.dir = /user/<itv004068>/warehouse;create database if not exists itv004068;use itv004068;

create table mtab1(id int,name string) row format delimited fields terminated by ',' location '/user/itv004068/emdir';create external table etab1(id int,name string) row format delimited fields terminated by ',' location '/user/itv004068/emdir';drop table etab1;select \* from mtab1 ;create external table etab1(id int,name string) row format delimited fields terminated by ',' location '/user/itv004068/emdir'; drop table mtab1;select \* from etab1 ;"

!hadoop fs -ls /user/itv004068/; -----> U will not see the directory emdir

============

SQL Scenario

============

=======

Task 2 ------

=======

mysql -uroot -pcloudera

create database prac1;

use prac1;

create table mtab (id int,name varchar(100),amount int);

insert into mtab values(1,'zeyo',40);

insert into mtab values(2,'hema',40);

insert into mtab values(3,'ravi',5);

insert into mtab values(4,'vasu',4);

insert into mtab values(5,'vani',7);

insert into mtab values(6,'rani',1);

insert into mtab values(7,'rita',4);

insert into mtab values(8,'riya',4);

insert into mtab values(9,'raj',4);

insert into mtab values(10,'siji',3);

insert into mtab values(11,'roni',5);

insert into mtab values(12,'visu',5);

select count(1) from mtab;

select count(1) from mtab where id>3;

select sum(amount) as total from mtab;

=========

Lab Folks

=========

mysql --host=zeyod2.czgbk94pezaa.ap-south-1.rds.amazonaws.com --user=root --password=Aditya908

create database itv004068;

use itv004068;

create table mtab4 (id int,name varchar(100),amount int);

insert into mtab4 values(1,'zeyo',40);

insert into mtab4 values(2,'hema',40);

insert into mtab4 values(3,'ravi',5);

insert into mtab4 values(4,'vasu',4);

insert into mtab4 values(5,'vani',7);

insert into mtab4 values(6,'rani',1);

insert into mtab4 values(7,'rita',4);

insert into mtab4 values(8,'riya',4);

insert into mtab4 values(9,'raj',4);

insert into mtab4 values(10,'siji',3);

insert into mtab4 values(11,'roni',5);

insert into mtab4 values(12,'visu',5);

select \* from mtab4;

select count(1) from mtab4;

select count(1) from mtab4 where id>3;

select sum(amount) as total from mtab4;

==========

22-10-2022

==========

=================

Cloudera Folks -- static load

=================

cd

echo 1,Sai,I>INDTxns.csv

echo 2,zeyo,I>>INDTxns.csv

echo 3,Hema,K>UKTxnx.csv

echo 4,ravi,K>>UKTxnx.csv

echo 5,Jai,S>USTxns.csv

echo 6,Swathi,S>>USTxns.csv

hive

create database if not exists pdb;

use pdb;

create table parttab(id int,name string,chk string) partitioned by (country string) row format delimited fields terminated by ',' location '/user/cloudera/pdir';

load data local inpath '/home/cloudera/INDTxns.csv' into table parttab partition(country='INDIA');

load data local inpath '/home/cloudera/USTxns.csv' into table parttab partition(country='USA');

load data local inpath '/home/cloudera/UKTxnx.csv' into table parttab partition(country='UK');

select \* from parttab;

!hadoop fs -ls /user/cloudera/pdir/;

!hadoop fs -rmr /user/cloudera/pdir/\*;

=================

lab Folks -- Static Load

=================

echo 1,Sai,I>INDTxns.csv

echo 2,zeyo,I>>INDTxns.csv

echo 3,Hema,K>UKTxnx.csv

echo 4,ravi,K>>UKTxnx.csv

echo 5,Jai,S>USTxns.csv

echo 6,Swathi,S>>USTxns.csv

hive -e "set hive.metastore.warehouse.dir=/user/itv004068/warehouse;create database if not exists itv004068;use itv004068;drop table parttab;create table parttab(id int,name string,chk string) partitioned by (country string) row format delimited fields terminated by ',' location '/user/itv004068/pdir';load data local inpath '/home/itv004068/INDTxns.csv' into table parttab partition(country='INDIA');load data local inpath '/home/itv004068/USTxns.csv' into table parttab partition(country='USA');load data local inpath '/home/itv004068/UKTxnx.csv' into table parttab partition(country='UK')"

hadoop fs -ls /user/itv004068/pdir

hadoop fs -ls /user/itv004068/pdir/country=INDIA

=================

Cloudera Folks -- static insert

=================

cd

echo 1,Sai,I,IND>allcountry.csv

echo 2,zeyo,I,IND>>allcountry.csv

echo 3,Hema,K,UK>>allcountry.csv

echo 4,Gomathi,K,UK>>allcountry.csv

echo 5,Jai,S,US>>allcountry.csv

echo 6,Swathi,S,US>>allcountry.csv

hive

create database if not exists pdb;

use pdb;

create table sitab(id int,name string,chk string) partitioned by (country string) row format delimited fields terminated by ',' location '/user/cloudera/sidir';

create table srctab(id int,name string,chk string,country string) row format delimited fields terminated by ',' location '/user/cloudera/sdir';

load data local inpath '/home/cloudera/allcountry.csv' into table srctab;

insert into sitab partition(country='USA') select id,name,chk from srctab where country='US';

!hadoop fs -ls /user/cloudera/sdir;

=================

Lab Folks -- static insert

=================

cd

echo 1,Sai,I,IND>allcountry.csv

echo 2,zeyo,I,IND>>allcountry.csv

echo 3,Hema,K,UK>>allcountry.csv

echo 4,Gomathi,K,UK>>allcountry.csv

echo 5,Jai,S,US>>allcountry.csv

echo 6,Swathi,S,US>>allcountry.csv

hive -e "set hive.metastore.warehouse.dir=/user/itv004068/warehouse;create database if not exists itv004068;use itv004068;drop table sitab;create table sitab(id int,name string,chk string) partitioned by (country string) row format delimited fields terminated by ',' location '/user/itv004068/sidir';create table srctab(id int,name string,chk string,country string) row format delimited fields terminated by ',' location '/user/itv004068/sdir';load data local inpath '/home/itv004068/allcountry.csv' into table srctab;insert into sitab partition(country='USA') select id,name,chk from srctab where country='US';"

hadoop fs -ls /user/itv004068/sdir

==========

23-10-2022

==========

=================

Cloudera Folks

=================

cd

echo 1,Sai,I,IND>allcountry.csv

echo 2,zeyo,I,IND>>allcountry.csv

echo 3,Hema,K,UK>>allcountry.csv

echo 4,Gomathi,K,UK>>allcountry.csv

echo 5,Jai,S,US>>allcountry.csv

echo 6,Swathi,S,US>>allcountry.csv

hive

create database if not exists pdb;

use pdb;

create table dyntab(id int,name string,chk string) partitioned by (country string) row format delimited fields terminated by ',' location '/user/cloudera/dyndir';

create table sttab(id int,name string,chk string,country string) row format delimited fields terminated by ',' location '/user/cloudera/stdir';

load data local inpath '/home/cloudera/allcountry.csv' into table sttab;

set hive.exec.dynamic.partition.mode=nonstrict;

insert into dyntab partition(country) select id,name,chk,country from sttab;

=================

Lab Folks

=================

cd

echo 1,Sai,I,IND>allcountry.csv

echo 2,zeyo,I,IND>>allcountry.csv

echo 3,Hema,K,UK>>allcountry.csv

echo 4,Gomathi,K,UK>>allcountry.csv

echo 5,Jai,S,US>>allcountry.csv

echo 6,Swathi,S,US>>allcountry.csv

hive -e "set hive.metastore.warehouse.dir=/user/itv004068/warehouse;create database if not exists itv004068;use itv004068;create table dyntab(id int,name string,chk string) partitioned by (country string) row format delimited fields terminated by ',' location '/user/itv004068/dyndir';create table sttab(id int,name string,chk string,country string) row format delimited fields terminated by ',' location '/user/itv004068/stdir';load data local inpath '/home/itv004068/allcountry.csv' into table sttab;set hive.exec.dynamic.partition.mode=nonstrict;insert into dyntab partition(country) select id,name,chk,country from sttab;"

hadoop fs -ls /user/itv004068/dyndir

====================================

Verify and read data Cloudera Folks

====================================

hadoop fs -ls /user/cloudera/dyndir

hadoop fs -ls /user/cloudera/dyndir/country=IND

hadoop fs -ls /user/cloudera/dyndir/country=US

hadoop fs -ls /user/cloudera/dyndir/country=UK

hadoop fs -cat /user/cloudera/dyndir/country=IND/\*

hadoop fs -cat /user/cloudera/dyndir/country=US/\*

hadoop fs -cat /user/cloudera/dyndir/country=UK/\*

====================================

Verify and read data <itv004068> Folks

====================================

hadoop fs -ls /user/itv004068/dyndir

hadoop fs -ls /user/itv004068/dyndir/country=IND

hadoop fs -ls /user/itv004068/dyndir/country=US

hadoop fs -ls /user/itv004068/dyndir/country=UK

hadoop fs -cat /user/itv004068/dyndir/country=IND/\*

hadoop fs -cat /user/itv004068/dyndir/country=US/\*

hadoop fs -cat /user/itv004068/dyndir/country=UK/\*

==========================

Task 1 --- Sub partitions

==========================

==================

Sub partitions-- Cloudera Folks

==================

============

Task 1 -----

============

cd

echo 1,Sai,I,IND,cash>allc.csv

echo 2,zeyo,I,IND,credit>>allc.csv

echo 3,Hema,K,UK,cash>>allc.csv

echo 4,Gomathi,K,UK,credit>>allc.csv

echo 5,Jai,S,US,cash>>allc.csv

echo 6,Swathi,S,US,credit>>allc.csv

echo 7,Sai,I,IND,credit>>allc.csv

echo 8,zeyo,I,IND,cash>>allc.csv

echo 9,Hema,K,UK,credit>>allc.csv

echo 10,Gomathi,K,UK,cash>>allc.csv

echo 11,Jai,S,US,credit>>allc.csv

echo 12,Swathi,S,US,cash>>allc.csv

create table srcs(id int,name string,chk string,country string,spendby string) row format delimited fields terminated by ',' location '/user/cloudera/srcd';

load data local inpath '/home/cloudera/allc.csv' into table srcs;

create table tars(id int,name string,chk string) partitioned by (country string,spendby string) row format delimited fields terminated by ',' location '/user/cloudera/tard';

set hive.exec.dynamic.partition.mode=nonstrict;

insert into tars partition (country,spendby) select id,name,chk,country,spendby from srcs;

!hadoop fs -ls /user/cloudera/tard;

==================

Sub partitions-- Lab Folks

==================

cd

echo 1,Sai,I,IND,cash>allc.csv

echo 2,zeyo,I,IND,credit>>allc.csv

echo 3,Hema,K,UK,cash>>allc.csv

echo 4,Gomathi,K,UK,credit>>allc.csv

echo 5,Jai,S,US,cash>>allc.csv

echo 6,Swathi,S,US,credit>>allc.csv

echo 7,Sai,I,IND,credit>>allc.csv

echo 8,zeyo,I,IND,cash>>allc.csv

echo 9,Hema,K,UK,credit>>allc.csv

echo 10,Gomathi,K,UK,cash>>allc.csv

echo 11,Jai,S,US,credit>>allc.csv

echo 12,Swathi,S,US,cash>>allc.csv

hive -e "set hive.metastore.warehouse.dir=/user/itv004068/warehouse;set hive.exec.dynamic.partition.mode=nonstrict;create database if not exists itv004068;use itv004068;create table srcs(id int,name string,chk string,country string,spendby string) row format delimited fields terminated by ',' location '/user/itv004068/srcd';load data local inpath '/home/itv004068/allc.csv' into table srcs;create table tars(id int,name string,chk string) partitioned by (country string,spendby string) row format delimited fields terminated by ',' location '/user/itv004068/tard';insert into tars partition (country,spendby) select id,name,chk,country,spendby from srcs;"

hadoop fs -ls /user/itv004068/tard

============

SQL Scenario

============

=============

Task 2 ------

=============

mysql -uroot -pcloudera

create database prac2;

use prac2;

create table mtab (id int,name varchar(100),amount int);

insert into mtab values(1,'zeyo',40);

insert into mtab values(2,'hema',40);

insert into mtab values(3,'ravi',5);

insert into mtab values(4,'vasu',4);

insert into mtab values(5,'vani',7);

insert into mtab values(6,'rani',1);

insert into mtab values(7,'rita',4);

insert into mtab values(8,'riya',4);

insert into mtab values(9,'raj',4);

insert into mtab values(10,'siji',3);

insert into mtab values(11,'roni',5);

insert into mtab values(12,'visu',5);

select \*,current\_date from mtab;

select \*,current\_timestamp from mtab;

create table mtabnew as select \* from mtab where id>5;

select \* from mtabnew;

delete from mtabnew;

select \* from mtabnew;

insert into mtabnew select \* from mtab where id<5;

select \* from mtabnew;

=========

Lab Folks

=========

mysql --host=zeyod2.czgbk94pezaa.ap-south-1.rds.amazonaws.com --user=root --password=Aditya908

create database itv004068;

use itv004068;

drop table mtab4;

create table mtab4 (id int,name varchar(100),amount int);

insert into mtab4 values(1,'zeyo',40);

insert into mtab4 values(2,'hema',40);

insert into mtab4 values(3,'ravi',5);

insert into mtab4 values(4,'vasu',4);

insert into mtab4 values(5,'vani',7);

insert into mtab4 values(6,'rani',1);

insert into mtab4 values(7,'rita',4);

insert into mtab4 values(8,'riya',4);

insert into mtab4 values(9,'raj',4);

insert into mtab4 values(10,'siji',3);

insert into mtab4 values(11,'roni',5);

insert into mtab4 values(12,'visu',5);

select \* from mtab4;

select \*,current\_date from mtab;

select \*,current\_timestamp from mtab;

create table mtabnew as select \* from mtab where id>5;

select \* from mtabnew;

delete from mtabnew;

select \* from mtabnew;

insert into mtabnew select \* from mtab where id<5;

select \* from mtabnew;

==========

29-10-2022

==========

hadoop@35.154.148.13

zeyovmnew.ppk-key

=============

Cloudera avro

=============

mysql -uroot -pcloudera

create database dataa;

use dataa;

drop table atab;

create table atab(id int,name varchar(100),amount int);

insert into atab values(1,'rajesh',40);

insert into atab values(2,'vishnu',10);

insert into atab values(3,'rani',60);

select \* from atab;

quit;

mkdir avd

cd avd

rm -rf \*

sqoop import --connect jdbc:mysql://localhost/dataa --username root --password cloudera --table atab --m 1 --delete-target-dir --target-dir /user/cloudera/adir --as-avrodatafile

hadoop fs -put -f /home/cloudera/avd/atab.avsc /user/cloudera/

hive -e """

create database if not exists adb;

use adb;

drop table atab;

create external table atab ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe' STORED AS AVRO LOCATION '/user/cloudera/adir' TBLPROPERTIES ('avro.schema.url'='/user/cloudera/atab.avsc');

select \* from atab """

mysql -uroot -pcloudera

use dataa;

alter table atab drop column name;

insert into atab values(4,90);

insert into atab values(5,20);

select \* from atab;

quit

sqoop import --connect jdbc:mysql://localhost/dataa --username root --password cloudera --table atab --m 1 --target-dir /user/cloudera/adir --as-avrodatafile --incremental append --check-column id --last-value 3

hive -e "select \* from adb.atab"

========

Lab avro

========

mysql --host=zeyod2.czgbk94pezaa.ap-south-1.rds.amazonaws.com --user=root --password=Aditya908

create database if not exists itv004068;

use itv004068;

drop table atab;

create table atab(id int,name varchar(100),amount int);

insert into atab values(1,'rajesh',40);

insert into atab values(2,'vishnu',10);

insert into atab values(3,'rani',60);

select \* from atab;

quit;

mkdir avd

cd avd

rm -rf \*

sqoop import -Dmapreduce.job.user.classpath.first=true --connect jdbc:mysql://zeyod2.czgbk94pezaa.ap-south-1.rds.amazonaws.com/itv004068 --username root --password Aditya908 --table atab --m 1 --delete-target-dir --target-dir /user/itv004068/adir --as-avrodatafile

hadoop fs -put -f /home/itv004068/avd/atab.avsc /user/itv004068/

hive -e """

create database if not exists itv004068;

use itv004068;

drop table atab;

create external table atab ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe' STORED AS AVRO LOCATION '/user/itv004068/adir' TBLPROPERTIES ('avro.schema.url'='/user/itv004068/atab.avsc');

select \* from atab """

mysql --host=zeyod2.czgbk94pezaa.ap-south-1.rds.amazonaws.com --user=root --password=Aditya908

create database if not exists itv004068;

use itv004068;

alter table atab drop column name;

insert into atab values(4,90);

insert into atab values(5,20);

select \* from atab;

quit

sqoop import -Dmapreduce.job.user.classpath.first=true --connect jdbc:mysql://zeyod2.czgbk94pezaa.ap-south-1.rds.amazonaws.com/itv004068 --username root --password Aditya908 --table atab --m 1 --target-dir /user/itv004068/adir --as-avrodatafile --incremental append --check-column id --last-value 3

hive -e "select \* from itv004068.atab"

hive -e "set hive.metastore.warehouse.dir=/user/itv004068/warehouse;

create database if not exists itv004068;

use itv004068;

create table dyntab1(id int,name string,chk string) partitioned by (country string) row format delimited fields terminated by ',' location '/user/itv004068/dyndir';

create table sttab1(id int,name string,chk string,country string) row format delimited fields terminated by ',' location '/user/itv004068/stdir';

load data local inpath '/home/itv004068/allcountry.csv' into table sttab1;

set hive.exec.dynamic.partition.mode=nonstrict;

insert into dyntab1 partition(country) select id,name,chk,country from sttab1;"

Cloudera Folks Project

===============

Go to Mysql

===============

mysql -uroot -pcloudera

Create database if not exists prodb;

use prodb;

select \* from customer\_total;

create table customer\_src(id int(10),username varchar(100),sub\_port varchar(100),host varchar(100),date\_time varchar(100),hit\_count\_val\_1 varchar(100),hit\_count\_val\_2 varchar(100),hit\_count\_val\_3 varchar(100),timezone varchar(100),method varchar(100),`procedure` varchar(100),value varchar(100),sub\_product varchar(100),web\_info varchar(100),status\_code varchar(100));

insert into customer\_src select \* From customer\_total where id>=301 and id<=330;

quit

=============================

Edge Node

=============================

echo -n cloudera>/home/cloudera/passfile

rm -rf /home/cloudera/avsrcdir

mkdir /home/cloudera/avsrcdir

cd /home/cloudera/avsrcdir

sqoop job --delete inpjob

sqoop job --create inpjob -- import --connect jdbc:mysql://localhost/prodb --username root --password-file file:///home/cloudera/passfile -m 1 --table customer\_src --target-dir /user/cloudera/customer\_stage\_loc --incremental append --check-column id --last-value 0 --as-avrodatafile

sqoop job --list

sqoop job --exec inpjob

hadoop fs -mkdir /user/cloudera/avscdirpro

hadoop fs -put /home/cloudera/avsrcdir/customer\_src.avsc /user/cloudera/avscdirpro

====================================

Hive shell

====================================

hive

create database prodb;

use prodb;

create table customer\_src ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe' STORED AS AVRO LOCATION '/user/cloudera/customer\_stage\_loc' TBLPROPERTIES ('avro.schema.url'='/user/cloudera/avscdirpro/customer\_src.avsc');

select \* from customer\_src;

Lab Folks

===============

Go to Mysql

===============

mysql -u nyse\_user -h ms.itversity.com -pitversity

use nyse\_export;

create table customer\_src\_itv004068(id int(10),username varchar(100),sub\_port varchar(100),host varchar(100),date\_time varchar(100),hit\_count\_val\_1 varchar(100),hit\_count\_val\_2 varchar(100),hit\_count\_val\_3 varchar(100),timezone varchar(100),method varchar(100),`procedure` varchar(100),value varchar(100),sub\_product varchar(100),web\_info varchar(100),status\_code varchar(100));

insert into customer\_src\_itv004068 select \* From customer\_total where id>=301 and id<=330;

quit

=============================

Edge Node

=============================

echo -n itversity>/home/itv004068/passfile

rm -rf /home/itv004068/avsrcdir

mkdir /home/itv004068/avsrcdir

cd /home/itv004068/avsrcdir

sqoop import -Dmapreduce.job.user.classpath.first=true --connect jdbc:mysql://ms.itversity.com/nyse\_export --username nyse\_user --password-file file:///home/itv004068/passfile -m 1 --table customer\_src\_itv004068 --target-dir /user/itv004068/customer\_stage\_loc --incremental append --check-column id --last-value 0 --as-avrodatafile

hadoop fs -mkdir /user/itv004068/avscdirpro

hadoop fs -put /home/itv004068/avsrcdir/customer\_src\_itv004068.avsc /user/itv004068/avscdirpro

====================================

Hive shell

====================================

hive

SET hive.metastore.warehouse.dir = /user/itv004068/warehouse;

create database prodb\_itv004068;

use prodb\_itv004068;

create table customer\_src ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe' STORED AS AVRO LOCATION '/user/itv004068/customer\_stage\_loc' TBLPROPERTIES ('avro.schema.url'='/user/itv004068/avscdirpro/customer\_src\_itv004068.avsc');

select \* from customer\_src; === U will see the data

1) Verify you have customer\_total table

2) create customer\_src table like customer\_total

3) Insert into customser\_src from customer\_total where id>300 and id<350

4) create a password file your edge Node == (echo -n cloudera>passfile)

5)Create a directory in Edge Node

6) Go inside that directory

7) create sqoop job for customer\_src table with id last value 0 along with password file

8)Execute the Job

9) Verify whether data got imported

10) AVSC file should have generated locally-- Copy that to HDFS

11) Go inside Hive shell and create database prodb;

12) Now create Hive table on top of Raw data with AVSC FILE as reference

===========

05-11-2022

===========

Downloads

Windows ----- java installation

https://drive.google.com/file/d/1vGKX-r6SzT6ND-NQ0co6vKUD61Xd2\_zl/view?usp=sharing

Windows ----- Scala IDE

http://downloads.typesafe.com/scalaide-pack/4.7.0-vfinal-oxygen-212-20170929/scala-SDK-4.7.0-vfinal-2.12-win32.win32.x86\_64.zip

Windows ----- Intellij Download

https://www.jetbrains.com/idea/download/download-thanks.html?platform=windows&code=IIC

Windows ---- WinUtils Download

https://github.com/steveloughran/winutils/raw/master/hadoop-2.7.1/bin/winutils.exe

Spark Download ----MAC/Windows/Ubuntu

https://archive.apache.org/dist/spark/spark-2.4.7/spark-2.4.7-bin-hadoop2.6.tgz

MAC Users --- Intellij Download

https://www.jetbrains.com/idea/download/download-thanks.html?platform=mac&code=IIC

Ubuntu Users -- Intellij Download

https://www.jetbrains.com/idea/download/download-thanks.html?platform=linux&code=IIC

==========

06-11-2022

==========

Task 1 --- Print Hello world in your Eclipse / Intellij

Try Eclipse video First.. If not working

Straight away go ahead with IntellIj

Eclipse

https://youtu.be/aLe9Yjm\_1Jw

Intellij (Ubuntu/MAC)

https://youtu.be/\_2jC-I1EKz4

Task 2 ---- You have to create 5 Projects Like that --- Eclipse/ Intellij

(Optional)

Task 3 ---- In the Project. Do the insert statement

insert into projdb.customer\_target\_tab partition (current\_day,year,month,day) select id, username,sub\_port,host,date\_time,hit\_count\_val\_1,hit\_count\_val\_2,hit\_count\_val\_3,timezone,method,procedure,value,sub\_product,web\_info,status\_code,current\_date,year(from\_unixtime(unix\_timestamp(date\_time,'dd/MMM/yyyy:HH:mm:ss Z'),'yyyy-MM-dd')) , MONTH(from\_unixtime(unix\_timestamp(date\_time,'dd/MMM/yyyy:HH:mm:ss Z'),'yyyy-MM-dd')) , DAY(from\_unixtime(unix\_timestamp(date\_time,'dd/MMM/yyyy:HH:mm:ss Z'),'yyyy-MM-dd'))from customer\_src where not(upper(web\_info) like'%JAKARTA%');

!hadoop fs -ls /user/cloudera/customer\_target\_tab;

!hadoop fs -ls /user/cloudera/customer\_target\_tab/current\_day=2022-07-29;

!hadoop fs -ls /user/cloudera/customer\_target\_tab/current\_day=2022-07-29/year=2011;

set hive.exec.max.dynamic.partitions=1000;

set hive.exec.dynamic.partition.mode=nonstrict;

insert into prodb\_itv004068.customer\_target\_tab partition (current\_day,year,month,day) select id, username,sub\_port,host,date\_time,hit\_count\_val\_1,hit\_count\_val\_2,hit\_count\_val\_3,timezone,method,'dummy' as proc,value,sub\_product,web\_info,status\_code,current\_date,year(from\_unixtime(unix\_timestamp(date\_time,'dd/MMM/yyyy:HH:mm:ss Z'),'yyyy-MM-dd')) , MONTH(from\_unixtime(unix\_timestamp(date\_time,'dd/MMM/yyyy:HH:mm:ss Z'),'yyyy-MM-dd')) , DAY(from\_unixtime(unix\_timestamp(date\_time,'dd/MMM/yyyy:HH:mm:ss Z'),'yyyy-MM-dd'))from customer\_src where not(upper(web\_info) like'%JAKARTA%');

!hadoop fs -ls /user/itv004068/customer\_target\_tab;

Task 4 ---- (optional) ---- What is diff between. RowNumer,Rank and DenseRank

==========

12-11-2022

==========

Gonna be crucial one .. Having your eclipse/Intellij on or below URL. ( anyone )

https://www.tutorialspoint.com/compile\_scala\_online.php

https://www.jdoodle.com/compile-scala-online/

https://onecompiler.com/scala

[10:23 AM, 11/12/2022] Sai Aditya Big Data Mentor: Mail --

Boss --- Sai Hope you received the Laptop

I already kept a small data all.csv in your windows Laptop

C:/data/all.csv

As a beginner . First Can you read this data using spark in eclipse and print it

Sai --- Zaharia , Can you please help me. How to read this data file and show it through Spark

Zaharia --- Sai, In spark you can read data using two Methods

textFile Method

csv Method

Sai --- Zaharia, Where,how Can I utilize this method

Zaharia -- You can see or call this methods from two classes

textFile ------> SparkContext Class

csv ------> SparkSession Class

Sai -- Zaharia, Where how ,we can see SparkContext and SparkSession Class

Zaharia --

textFile ------> SparkContext ----> spark-core.jar

csv ------> SparkSession ----> spark-sql.jar

Sai --- Got good Understanding

How can you use this jars,class,method to read a file

Add the jars

Initialize class

Call Method to read a file

Zaharia ---- Its 5 Step process

Open Eclipse ,create project with Scala Nature --- Done

Add Spark Jars to the project (spark-core,spark-sql) --- Done

Call the classes SparkContext and SparkSession

Inside Code Give powers of SparkContext and SparkSession to a variable

Call the Methods from that variable

package pack

object obj {

def main(args:Array[String]):Unit={

println("========Started=======")

val a = 2

println(a)

val b = "zeyo"

println(b)

}

package pack

object obj {

def main(args:Array[String]):Unit={

println("========Started=======")

val a = 2

println(a)

val b = "zeyo"

println(b)

}

}

https://www.tutorialspoint.com/compile\_scala\_online.php

https://www.jdoodle.com/compile-scala-online/

https://onecompiler.com/scala

package pack

object project7 {

def main(args:Array[String]):Unit={

println("=======Started=========")

val a=2

println(a)

val b="zeyo"

println(b)

val lis = List(1,2,3,4)

println(lis)

println("======indv print=====")

lis.foreach(println)

}

}

Task 1 -----

def main(args:Array[String]):Unit={

println("========Started=======")

val a = 2

println(a)

val b = "zeyo"

println(b)

val lis = List(1,2,3,4,5)

println(lis)

println("=======raw list====")

lis.foreach(println)

val plis = lis.filter( x => x>2)

println("=====processed List===")

plis.foreach(println)

}

==========

13-11-2022

==========

-------

Task1

-------

package pack

object project9 {

def main(args:Array[String]):Unit={

println("======Started=======")

val lis = List(4,7,21,50)

println("======raw list======")

println(lis)

val proclis = lis.filter ( x => x < 20)

println("=====proc List=====")

println(proclis)

}

}

-------

Task2

-------

package pack

object project10 {

def main(args:Array[String]):Unit={

println("======Started=======")

println

val lis = List(1,2,3,4)

println("======raw list======")

lis.foreach(println)

val mullis = lis.map(x => x\*2)

println("=====proc List=====")

mullis.foreach(println)

}

}

-------

Task3

-------

package pack

object obj {

def main(args:Array[String]):Unit={

println("========Started=======")

val listr = List("zeyobron","analytics","zeyo")

println("=====raw List====")

listr.foreach(println)

val mapstr = listr.map( x => x.concat( ",sai" ))

println("=====concat List====")

mapstr.foreach(println)

val repstr = listr.map( x => x.replace("zeyo", "tera"))

println("=====replace List====")

repstr.foreach(println)

}

}

-----

Task4

-----

package pack

object project11 {

def main(args:Array[String]):Unit={

println("========Started=======")

val listr = List("zeyobron","analytics","zeyo")

println("=====raw List====")

listr.foreach(println)

val mapstr = listr.map( x => x.concat( ",siva" ))

println("=====concat List====")

mapstr.foreach(println)

val repstr = listr.map( x => x.replace("zeyo", "tera"))

println("=====replace List====")

repstr.foreach(println)

}

}

-------

Task 1 ---

-------

Task 1 -- Solution

package pack

object obj {

def main(args:Array[String]):Unit={

println("========Started=======")

val lisstr= List( "A~B" , "C~D" , "E~F" )

println("=====raw List====")

lisstr.foreach(println)

val flatstr = lisstr.flatMap( x => x.split("~") )

println("======flat List ====")

flatstr.foreach(println)

}

}

Complete Flat Map

Task 2 -----

Cloudera Folks

mysql -uroot -pcloudera

create database prac5;

use prac5;

create table mtab (id int,name varchar(100),amount int);

insert into mtab values(10,'zeyo',40);

insert into mtab values(20,'hema',40);

insert into mtab values(30,'ravi',5);

insert into mtab values(40,'vasu',4);

select \* , row\_number() from mtab;

Lab Folks

mysql --host=zeyod2.czgbk94pezaa.ap-south-1.rds.amazonaws.com --user=root --password=Aditya908

create database itv004068;

use itv004068;

create table mtab (id int,name varchar(100),amount int);

insert into mtab values(10,'zeyo',40);

insert into mtab values(20,'hema',40);

insert into mtab values(30,'ravi',5);

insert into mtab values(40,'vasu',4);

select \* , row\_number() from mtab;

Task 2 --- Solution ( \*Lab and Cloudera )\*

--------------

Cloudera Folks

--------------

mysql -uroot -pcloudera

create database prac5;

use prac5;

create table mtab (id int,name varchar(100),amount int);

insert into mtab values(10,'zeyo',40);

insert into mtab values(20,'hema',40);

insert into mtab values(30,'ravi',5);

insert into mtab values(40,'vasu',4);

set @row\_number := 0;

select \*,(@row\_number:=@row\_number + 1) AS num from mtab;

---------

Lab Folks

---------

mysql --host=zeyod2.czgbk94pezaa.ap-south-1.rds.amazonaws.com --user=root --password=Aditya908

create database itv004068;

use itv004068;

create table mtab (id int,name varchar(100),amount int);

insert into mtab values(10,'zeyo',40);

insert into mtab values(20,'hema',40);

insert into mtab values(30,'ravi',5);

insert into mtab values(40,'vasu',4);

set @row\_number := 0;

select \*,(@row\_number:=@row\_number + 1) AS num from mtab;

Optional Task ----

val liststr = List("Amazon-Jeff-America",

"Microsoft-BillGates-America",

"TCS-TATA-india",

"Reliance-Ambani-india"

)

Filter elements Contains "india"

From that result Flatten with -

From that Result replace "india" with "local"

From that Result contact ",zeyo"

package pack

object obj {

def main(args:Array[String]):Unit={

println("========Started=======")

val liststr = List("Amazon-Jeff-America",

"Microsoft-BillGates-America",

"TCS-TATA-india",

"Reliance-Ambani-india"

)

println("=====raw List====")

liststr.foreach(println)

val filstr = liststr.filter( x => x.contains("india"))

println("=====filter List====")

filstr.foreach(println)

println("=====flatmap List====")

val flatdata = filstr.flatMap( x => x.split("-"))

flatdata.foreach(println)

println("=====replace List====")

val repdata = flatdata.map( x => x.replace("india","local"))

repdata.foreach(println)

println("===concat list =====")

val mapdata = repdata.map( x => x.concat(",sai"))

mapdata.foreach(println)

}

}

============

19-11-2022

============

package pack

object obj {

def main(args:Array[String]):Unit={

println("Hello World")

println("========Started=======")

val liststr = List("Amazon-Jeff-America",

"Microsoft-BillGates-America",

"TCS-TATA-india",

"Reliance-Ambani-india"

)

println

println("========raw List=======")

liststr.foreach(println)

println

//Filter elements Contains "india"

val indlis = liststr.filter( x =>x.contains("india"))

println("========India List=======")

indlis.foreach(println)

println

//From that result Flatten with -

val flatdata = indlis.flatMap( x => x.split("-"))

println("========flat List=======")

flatdata.foreach(println)

println

//From that Result replace "india" with "local"

val repdata = flatdata.map( x => x.replace("india","local"))

println("========repdata List=======")

repdata.foreach(println)

println

//From that Result contact ",zeyo"

val mapdata = repdata.map( x => x + ",zeyo")

println("========mapdata List=======")

mapdata.foreach(println)

println

}

}

val liststr=

List(

"State->TamilNadu~City->Chennai",

"State->Karnataka~City->Bangalore",

"State->Telangana~City->Hyderabad"

)

https://www.tutorialspoint.com/compile\_scala\_online.php

https://www.jdoodle.com/compile-scala-online/

https://onecompiler.com/scala

========raw list=======

State->TamilNadu~City->Chennai

State->Karnataka~City->Bangalore

State->Telangana~City->Hyderabad

========flatdata list=======

State->TamilNadu

City->Chennai

State->Karnataka

City->Bangalore

State->Telangana

City->Hyderabad

========statedata list=======

State->TamilNadu

State->Karnataka

State->Telangana

========citydata list=======

City->Chennai

City->Bangalore

City->Hyderabad

========finalstate list=======

TamilNadu

Karnataka

Telangana

========finalcity list=======

Chennai

Bangalore

Hyderabad

val liststr=

List(

"State->TamilNadu~City->Chennai",

"State->Karnataka~City->Bangalore",

"State->Telangana~City->Hyderabad"

)

println("========raw list=======")

println

liststr.foreach(println)

val flatdata = liststr.flatMap( x=>x.split("~"))

println("========flatdata list=======")

flatdata.foreach(println)

println

val statedata = flatdata.filter( x => x.contains("State"))

println("========statedata list=======")

println

statedata.foreach(println)

val citydata = flatdata.filter( x => x.contains("City"))

println("========citydata list=======")

println

citydata.foreach(println)

val finalstate = statedata.map( x => x.replace("State->",""))

println("========finalstate list=======")

println

finalstate.foreach(println)

val finalcity = citydata.map( x => x.replace("City->",""))

println("========finalcity list=======")

println

finalcity.foreach(println)

Add Jars to the Project

Give import Statements

Initialize Conf and Context

Read the File and Process it

package pack

import org.apache.spark.SparkContext

import org.apache.spark.SparkConf

object obj {

def main(args:Array[String]):Unit={

val conf = new SparkConf().setAppName("First").setMaster("local[\*]")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val data= sc.textFile("file:///C:/data/scdata.txt")

println("===== Raw Rdd=======")

println

data.foreach(println)

println

val flatdata= data.flatMap( x => x.split("~"))

println("===== flatdata Rdd=======")

println

flatdata.foreach(println)

println

val statedata = flatdata.filter( x => x.toLowerCase().contains("state"))

println("===== statedata Rdd=======")

println

statedata.foreach(println)

println

val citydata = flatdata.filter( x => x.contains("City"))

println("===== citydata Rdd=======")

println

citydata.foreach(println)

println

val finalstate = statedata.map( x => x.replace("State->", ""))

println("===== finalstate Rdd=======")

println

finalstate.foreach(println)

println

val finalcity = citydata.map( x => x.replace("City->", ""))

println("===== finalcity Rdd=======")

println

finalcity.foreach(println)

println

finalstate.coalesce(1).saveAsTextFile("file:///C:/data/statedata")

}

}

val liststr= List(

"BigData-Spark-Hive",

"Spark-Hadoop-Hive",

"Sqoop-Hive-Spark",

"Sqoop-BD-Hive"

)

Expected Result

Tech->BigData Trainer->Sai

Tech->Spark Trainer->Sai

Tech->Hive Trainer->Sai

Tech->Hadoop Trainer->Sai

Tech->Sqoop Trainer->Sai

Tech->BigData Trainer->Sai

FlatMap with -

Remove duplicates using distinct

Prefix "Tech->" and suffix " Trainer->Sai"

Replace BD with BigData

===============

package pack

import org.apache.spark.SparkContext

import org.apache.spark.SparkConf

object project13 {

def main(args:Array[String]):Unit={

val liststr= List(

"BigData-Spark-Hive",

"Spark-Hadoop-Hive",

"Sqoop-Hive-Spark",

"Sqoop-BD-Hive"

)

println("====== Raw List=====")

liststr.foreach(println)

println

val flatlis = liststr.flatMap(x => x.split("-")).distinct

println("====== Flattened List=====")

flatlis.foreach(println)

println

val concatlis = flatlis.map(x => "Tech->" + x + "Trainer->Sai")

println("====== Prefix & Suffix List=====")

concatlis.foreach(println)

println

val finallis = concatlis.map(x => x.replace("BD","BigData"))

println("====== Final List=====")

finallis.foreach(println)

println

}

}

==============

host --ms.itversity.com

username -- hr\_user

password -- itversity

To connect mysql

mysql -u hr\_user -h ms.itversity.com -pitversity

DB to use

use hr\_export;

===========

20-11-2022

===========

First read this Data Done

Filter the Rows which has length>200 Done

Flatten the data with ,Done

Remove hyphon (-) from all the flatten Rows Done

Concat ,zeyo for each string ,Done

Write the results to a file

package pack

import org.apache.spark.\_

object obj {

def main(args:Array[String]):Unit={

val conf = new SparkConf().setAppName("first").setMaster("local[\*]")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val data = sc.textFile("file:///C:/data/usdata.csv")

println("=====Raw data==== ")

data.take(10).foreach(println)

println

val fildata = data.filter( x => x.length() > 200)

println("=====fildata data==== ")

fildata.foreach(println)

println

val flatdata = fildata.flatMap( x => x.split(","))

println("=====flatdata data==== ")

flatdata.foreach(println)

println

val repdata = flatdata.map( x => x.replace("-",""))

println("=====repdata data==== ")

repdata.foreach(println)

println

val condata = repdata.map( x => x+ ",zeyo")

println("=====condata data==== ")

condata.foreach(println)

println

}

}

For rdd every row is a element

Column based processing is possible by default

Mapsplit

define case class

impose case class

Filter with Column itself

Dataframe conversion ( schemardd.toDF() )

df.write.parquet("")

package pack

import org.apache.spark.\_

import org.apache.spark.sql.SparkSession

object obj {

case class zeyoschema(id:String, category:String,product:String,mode:String)

def main(args:Array[String]):Unit={

val conf = new SparkConf().setAppName("first").setMaster("local[\*]")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession.builder().getOrCreate()

import spark.implicits.\_

val data = sc.textFile("file:///C:/data/datatxns.txt")

println("===raw data===")

println

data.foreach(println)

println

val gymdata = data.filter( x => x.contains("Gymnastics"))

println("===Row gymdata ===")

println

gymdata.foreach(println)

println

val mapsplit = data.map( x => x.split(","))

val schemardd = mapsplit.map( x => zeyoschema(x(0),x(1),x(2),x(3)))

val finalfilter = schemardd.filter( x => x.product.contains("Gymnastics"))

println("===Column gymdata ===")

finalfilter.foreach(println)

println

val df = finalfilter.toDF()

println("==Dataframe ===")

df.show()

}

}

===========

Task 1 ---

===========

Read datatxns.txt

do the map split

create a case class (id,category,product,mode)

Impose case class

Filter product contains Gymnastics and id>20

Task 2 (Optional) ----

Cloudera

mysql -uroot -pcloudera

create database scen,

use scen;

CREATE TABLE sales\_table (

Employee\_Name VARCHAR(45) NOT NULL,

Year INT NOT NULL,

Country VARCHAR(45) NOT NULL,

Product VARCHAR(45) NOT NULL,

Sale DECIMAL(12,2) NOT NULL,

PRIMARY KEY(Employee\_Name, Year)

);

INSERT INTO sales\_table VALUES

('Stephen', 2017, 'India', 'Laptop', 10000),

('Stephen', 2018, 'India', 'Laptop', 15000),

('Stephen', 2019, 'India', 'TV', 20000),

('Bob', 2017, 'US', 'Computer', 15000),

('Bob', 2018, 'US', 'Computer', 10000),

('Bob', 2019, 'US', 'TV', 20000),

('Mandy', 2017, 'Canada', 'Mobile', 20000),

('Mandy', 2018, 'Canada', 'Calculator', 1500),

('Mandy', 2019, 'Canada', 'Mobile', 25000);

select \* from sales\_table;

==================

Find the Total sale of each employee

==================

select employee\_name,sum(Sale) as total from sales\_table group by Employee\_Name;

==================

Find the Total sale of each employee for each product

==================

select employee\_name,sum(Sale),product as total from sales\_table group by Employee\_Name,product;

==================

Find the Maximum sale by an each employee

==================

select employee\_name,max(Sale) as max\_sale from sales\_table group by employee\_name;

==================

Find the average of sale by an each employee

==================

select employee\_name,avg(Sale) as max\_sale from sales\_table group by employee\_name;

===============

lab

===============

mysql -u nyse\_user -h ms.itversity.com -pitversity

use nyse\_export;

CREATE TABLE sales\_table\_itv004068 (

Employee\_Name VARCHAR(45) NOT NULL,

Year INT NOT NULL,

Country VARCHAR(45) NOT NULL,

Product VARCHAR(45) NOT NULL,

Sale DECIMAL(12,2) NOT NULL,

PRIMARY KEY(Employee\_Name, Year)

);

INSERT INTO sales\_table\_itv004068 VALUES

('Stephen', 2017, 'India', 'Laptop', 10000),

('Stephen', 2018, 'India', 'Laptop', 15000),

('Stephen', 2019, 'India', 'TV', 20000),

('Bob', 2017, 'US', 'Computer', 15000),

('Bob', 2018, 'US', 'Computer', 10000),

('Bob', 2019, 'US', 'TV', 20000),

('Mandy', 2017, 'Canada', 'Mobile', 20000),

('Mandy', 2018, 'Canada', 'Calculator', 1500),

('Mandy', 2019, 'Canada', 'Mobile', 25000);

==================

Find the Total sale of each employee

==================

select employee\_name,sum(Sale) as total from sales\_table\_itv004068 group by Employee\_Name;

==================

Find the Total sale of each employee for each product

==================

select employee\_name,sum(Sale),product as total from sales\_table\_itv004068 group by Employee\_Name,product;

==================

Find the Maximum sale by an each employee

==================

select employee\_name,max(Sale) as max\_sale from sales\_table\_itv004068 group by employee\_name;

==================

Find the average of sale by an each employee

==================

select employee\_name,avg(Sale) as max\_sale from sales\_table\_itv004068 group by employee\_name;

========

task1

========

object project13 {

def main(args:Array[String]):Unit={

val conf = new SparkConf().setAppName("first").setMaster("local[\*]")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val data = sc.textFile("file:///D:/data/usdata.csv")

println("=====Raw data======")

data.take(10).foreach(println)

println

val fildata = data.filter( x => x.length()>200)

println("=====fildata data======")

fildata.foreach(println)

println

val flatdata = fildata.flatMap( x => x.split(","))

println("=====flatdata data======")

flatdata.foreach(println)

println

val repdata = flatdata.map( x => x.replace("-",""))

println("======repdata data======")

repdata.foreach(println)

println

val condata = repdata.map( x => x + ",zeyo")

println("======condata data======")

condata.foreach(println)

println

}

}

==========================

package pack

import org.apache.spark.\_

import org.apache.spark.sql.SparkSession

object project13 {

case class zeyoschema(id:String, category:String,product:String,mode:String)

def main(args:Array[String]):Unit={

val conf = new SparkConf().setAppName("first").setMaster("local[\*]")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession.builder().getOrCreate()

import spark.implicits.\_

val data = sc.textFile("file:///D:/data/datatxns.txt")

println("===raw data===")

println

data.foreach(println)

println

val gymdata = data.filter( x => x.contains("Gymnastics"))

println("===Row gymdata ===")

println

gymdata.foreach(println)

println

val mapsplit = data.map( x => x.split(","))

val schemardd = mapsplit.map( x => zeyoschema(x(0),x(1),x(2),x(3)))

val finalfilter = schemardd.filter( x => x.product.contains("Gymnastics"))

println("===Column gymdata ===")

finalfilter.foreach(println)

println

val df = finalfilter.toDF()

println("==Dataframe ===")

df.show()

}

}

======================

Assignment Task 1

======================

package pack

import org.apache.spark.\_

import org.apache.spark.sql.SparkSession

object project13 {

case class zeyoschema(id:String, category:String,product:String,mode:String)

def main(args:Array[String]):Unit={

val conf = new SparkConf().setAppName("first").setMaster("local[\*]")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession.builder().getOrCreate()

import spark.implicits.\_

val data = sc.textFile("file:///D:/data/datatxns.txt")

println("===raw data===")

println

data.foreach(println)

println

val gymdata = data.filter( x => x.contains("Gymnastics"))

println("===Row gymdata ===")

println

gymdata.foreach(println)

println

val mapsplit = data.map( x => x.split(","))

val schemardd = mapsplit.map( x => zeyoschema(x(0),x(1),x(2),x(3)))

val finalfilter = schemardd.filter( x => x.product.contains("Gymnastics") && x.id.toInt>20)

println

}

}

=============

26-11-2022

=============

package pack

import org.apache.spark.\_

import org.apache.spark.sql.SparkSession

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.functions.\_

import org.apache.spark.sql.\_

object obj {

case class zeyoschema(

id:Int,

category:String,

product:String,

mode:String

)

def main(args:Array[String]):Unit={

val conf = new SparkConf().setAppName("first").setMaster("local[\*]")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession.builder.getOrCreate()

import spark.implicits.\_

// Using Schema rdd

println("====Schema rdd output=====")

println

val data = sc.textFile("file:///C:/data/datatxns.txt")

data.foreach(println)

val mapsplit = data.map( x => x.split(","))

val schemardd = mapsplit.map( x => zeyoschema(x(0).toInt,x(1),x(2),x(3)))

val filterdata = schemardd.filter( x=>

x.product.contains("Gymnastics") &&

x.id.toInt>20

)

println

filterdata.foreach(println)

val df = filterdata.toDF()

println

df.show()

//df.write.parquet("file:///C:/data/schemawrite")

println("=====row rdd output====")

val rdata = sc.textFile("file:///C:/data/datatxns.txt")

println

rdata.foreach(println)

val rmapsplit = rdata.map( x => x.split(","))

val rowrdd = rmapsplit.map( x => Row(x(0),x(1),x(2),x(3)))

val rfilterdata = rowrdd.filter( x =>

x(2).toString().contains("Gymnastics") &&

x(0).toString().toInt>20

)

println

rfilterdata.foreach(println)

val rschema = StructType(Array(

StructField("id",StringType),

StructField("category",StringType),

StructField("product",StringType),

StructField("mode", StringType)

))

val rdf = spark.createDataFrame(rfilterdata,rschema)

println

rdf.show()

rdf.write.parquet("file:///C:/data/rowwrite")

}

}

==========

Task 1 --

==========

Read datatxns.txt as dataframe with csv format

val df = spark

.read

.format("csv")

.load("file:///C:/data/datatxns.txt")

df.show()

df.createOrReplaceTempView("tab")

val finaldf = spark.sql("select \* from tab where \_c1='Gymnastics'")

finaldf.show()

==========

Task 2 ---

==========

val df2 = spark

.read

.format("json")

.load("file:///C:/data/devices.json")

df2.show()

df2.createOrReplaceTempView("jsontab")

val finaldf2 = spark.sql("select \* from jsontab where humidity>60")

finaldf2.show()

=============

27-11-2022

=============

package pack

import org.apache.spark.\_

import org.apache.spark.sql.SparkSession

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.functions.\_

import org.apache.spark.sql.\_

object project13 {

case class zeyoschema(

id:Int,

category:String,

product:String,

mode:String

)

def main(args:Array[String]):Unit={

val conf = new SparkConf().setAppName("first").setMaster("local[\*]")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession.builder.getOrCreate()

import spark.implicits.\_

val df = spark

.read

.format("csv")

.option("header","true")

.load("file:///D:/data/usdata.csv")

println("============================csvdf===========================")

println

df.show(5)

println

val jsondf = spark

.read

.format("json")

.load("file:///D:/data/devices.json")

println("======================jsondf=========================")

println

jsondf.show(5)

println

val orcdf = spark

.read

.format("orc")

.load("file:///D:/data/data.orc")

println("============================orcdf=====================")

println

orcdf.show(5)

println

val parquetdf = spark

.read

.format("parquet")

.load("file:///D:/data/part.parquet")

println("============================parquetdf=============")

println

parquetdf.show(5)

println

val avrodf = spark

.read

.format("avro")

.load("file:///D:/data/part.avro")

println("============================avrodf=============")

println

avrodf.show(5)

println

}

}

Task 2

package pack

import org.apache.spark.\_

import org.apache.spark.sql.SparkSession

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.functions.\_

import org.apache.spark.sql.\_

object obj {

case class zeyoschema(

id:Int,

category:String,

product:String,

mode:String

)

def main(args:Array[String]):Unit={

val conf = new SparkConf().setAppName("first").setMaster("local[\*]")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession.builder.getOrCreate()

import spark.implicits.\_

val df = spark

.read

.format("csv")

.option("header","true")

.load("file:///C:/semidata/usdata.csv")

println("============================csvdf===========================")

println

df.sho…

package pack

import org.apache.spark.SparkContext // rdd

import org.apache.spark.sql.SparkSession // dataframe

import org.apache.spark.SparkConf

import org.apache.spark.sql.\_

object obj {

def main(args:Array[String]):Unit={

val conf = new SparkConf().setMaster("local[\*]").setAppName("first")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession.builder.getOrCreate()

import spark.implicits.\_

val sqldf = spark

.read

.format("jdbc")

.option("url","jdbc:mysql://database-1.cwv8krqq8b83.ap-south-1.rds.amazonaws.com/zeyodb")

.option("driver","com.mysql.jdbc.Driver")

.option("dbtable","cashdata")

.option("user","root")

.option("password","Aditya908")

.load

sqldf.show()

}

}

Task 1 ---

Read devices.json as json file

df.createOrReplaceTempView("jsontab")

val finaldf=spark.sql("select \*,row\_number() from jsontab")

package pack

import org.apache.spark.SparkContext // rdd

import org.apache.spark.sql.SparkSession // dataframe

import org.apache.spark.SparkConf

import org.apache.spark.sql.\_

object project13 {

def main(args:Array[String]):Unit={

val conf = new SparkConf().setMaster("local[\*]").setAppName("first")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession.builder.getOrCreate()

import spark.implicits.\_

val df = spark.read.format(source = "json").load(path="file:///D:/data/devices.json")

df.createOrReplaceTempView(viewName = "jsontab")

val finaldf = spark.sql("select \* from jsontab")

finaldf.show()

}

}

Task 2 --- (If you have winutils solve it in doubts Session,

EXIT CODE =1156424535-SOMETHING)

Read usdata.csv as csv

Write as parquet

Write as json

write as orc

write as avro (add the Jar)

package pack

import org.apache.spark.SparkContext // rdd

import org.apache.spark.sql.SparkSession // dataframe

import org.apache.spark.SparkConf

import org.apache.spark.sql.\_

object project13 {

def main(args:Array[String]):Unit={

val conf = new SparkConf().setMaster("local[\*]").setAppName("first")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession.builder.getOrCreate()

import spark.implicits.\_

val usdatadf = spark

.read

.format("csv")

.option("header","true")

.load("file:///D:/data/usdata.csv")

println("=========csvdf=========")

println

usdatadf.show()

println

val parquetdf = usdatadf

.write

.format("parquet")

.option("header","true")

.save("file:///D:/write data/usdata\_parquet")

println("=========usdata\_parquet file Write as parquet=========")

println

usdatadf.show()

println

val jsondf = usdatadf

.write

.format("json")

.option("header","true")

.save("file:///D:/write data/usdata\_json")

println("=========usdata\_json file Write as json=========")

println

usdatadf.show()

println

val orcdf = usdatadf

.write

.format("orc")

.option("header","true")

.save("file:///D:/write data/usdata\_orc")

println("=========usdata\_orc file Write as orc=========")

println

usdatadf.show()

println

val avrodf = usdatadf

.write

.format("avro")

.option("header","true")

.save("file:///D:/write data/usdata\_avro")

println("=========usdata\_avro file Write as avro=========")

println

usdatadf.show()

println

}

}

Optional Task ----

SQOOP --

HIVE --

=============

03-12-2022

=============

Test each Mode separately

val df = spark.read.format("json").load("file:///C:/data/devices.json")

val finaldf = df.filter("humidity>40")

==========

Run it ---- It would be successful

==========

finaldf.write.format("csv").mode("error").save("file:///C:/data/dhumid")

==========

Run it --- It will through Error

==========

finaldf.write.format("csv").mode("error").save("file:///C:/data/dhumid")

==========

Run it --- it will append the data

==========

finaldf.write.format("csv").mode("append").save("file:///C:/data/dhumid")

==========

Run it --- it will overwrite the data

==========

finaldf.write.format("csv").mode("overwrite").save("file:///C:/data/dhumid")

==========

Run it --- it will ignore the data

==========

finaldf.write.format("csv").mode("ignore").save("file:///C:/data/dhumid")

package pack

import org.apache.spark.SparkContext // rdd

import org.apache.spark.sql.SparkSession // dataframe

import org.apache.spark.SparkConf

import org.apache.spark.sql.\_

object obj {

def main(args:Array[String]):Unit={

val conf = new SparkConf().setMaster("local[\*]").setAppName("first")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession.builder.getOrCreate()

import spark.implicits.\_

val df = spark

.read

.format("xml")

.option("rowtag","book")

.load("file:///C:/data/book.xml")

df.show()

}

}

===========

04-12-2022

===========

package pack

import org.apache.spark.SparkContext // rdd

import org.apache.spark.sql.SparkSession // dataframe

import org.apache.spark.SparkConf

import org.apache.spark.sql.\_

object project13 {

def main(args:Array[String]):Unit={

val conf = new SparkConf().setMaster("local[\*]").setAppName("first")

.set("fs.s3a.access.key","AKIASSGV4N5GIVX6MPVE")

.set("fs.s3a.secret.key","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")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession.builder.config(conf).getOrCreate()

import spark.implicits.\_

val df = spark

.read

.format("json")

.load("s3a://zdevb/cashdata")

df.show()

val finaldf = df.filter("category='Team Sports'")

finaldf.show()

}

}

======

Task 2

======

package pack

import org.apache.spark.SparkContext // rdd

import org.apache.spark.sql.SparkSession // dataframe

import org.apache.spark.SparkConf

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.\_

object obj {

def main(args:Array[String]):Unit={

val conf = new SparkConf().setMaster("local[\*]").setAppName("first")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession.builder.config(conf).getOrCreate()

import spark.implicits.\_

val dschema = StructType(Array(

StructField("id",StringType,true),

StructField("name",StringType,true),

StructField("chk",StringType,true),

StructField("country", StringType, true)

))

val df = spark

.read

.format("csv")

.schema(dschema)

.load("file:///C:/data/allcountry1.csv")

df.show()

df.write

.format("csv")

.partitionBy("country")

.mode("overwrite")

.save("file:///C:/data/countrypart")

}

}

Task 1 ----

Read dt.txt as a csv with header true

Create a temp View tdf

Find a way to design a query if spendby =cash then 1 else 0 as status

Task 2 ----

For the same tempview

Find the total sum of each category using SQL and order by it with Category

Task 3 ----

Create an sql in such a way that instead of tdate,I should have year in the column with column name as year -- Split operation

Task 4 ----

Using dataframe

df.dropDuplicates("category").show()

package pack

import org.apache.spark.SparkContext // rdd

import org.apache.spark.sql.SparkSession // dataframe

import org.apache.spark.SparkConf

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.functions.\_

object project13 {

def main(args:Array[String]):Unit={

val conf = new SparkConf().setMaster("local[\*]").setAppName("first")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession.builder.config(conf).getOrCreate()

import spark.implicits.\_

val df = spark

.read

.format("csv")

.option("header","true")

.load("file:///D:/data/dt.txt")

df.show()

//Task 1

df.createOrReplaceTempView("tdf")

val df1= spark.sql("""select \*,case

when spendby='cash'

then 1

else 0 end as status

from tdf""")

println("========Task 1========")

df1.show()

// Task 2

val df2 = spark.sql("""select category,

sum(amount) as total

from tdf

group by category

order by category""")

println("========Task 2========")

df2.show()

// Task 3

val df3 = spark.sql("""select id,

split(tdate,'-')[2] as year,

amount,

category,

product,

spendby from tdf""")

println("========Task 3========")

df3.show()

// Task 4

val df4= df.dropDuplicates("category")

println("========Task 4========")

df4.show()

}

}

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

| id| tdate|amount| category| product|spendby|

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

|00000000|06-26-2011| 200| Exercise|GymnasticsPro| cash|

|00000001|05-26-2011| 300| Exercise|Weightlifting| credit|

|00000002|06-01-2011| 100| Exercise|GymnasticsPro| cash|

|00000003|06-05-2011| 100| Gymnastics| Rings| credit|

|00000004|12-17-2011| 300|Team Sports| Field| cash|

|00000005|02-14-2011| 200| Gymnastics| Ring| cash|

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

========Task 1========

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

| id| tdate|amount| category| product|spendby|status|

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

|00000000|06-26-2011| 200| Exercise|GymnasticsPro| cash| 1|

|00000001|05-26-2011| 300| Exercise|Weightlifting| credit| 0|

|00000002|06-01-2011| 100| Exercise|GymnasticsPro| cash| 1|

|00000003|06-05-2011| 100| Gymnastics| Rings| credit| 0|

|00000004|12-17-2011| 300|Team Sports| Field| cash| 1|

|00000005|02-14-2011| 200| Gymnastics| Ring| cash| 1|

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

========Task 2========

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

| category|total|

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

| Exercise|600.0|

| Gymnastics|300.0|

|Team Sports|300.0|

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

========Task 3========

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

| id|year|amount| category| product|spendby|

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

|00000000|2011| 200| Exercise|GymnasticsPro| cash|

|00000001|2011| 300| Exercise|Weightlifting| credit|

|00000002|2011| 100| Exercise|GymnasticsPro| cash|

|00000003|2011| 100| Gymnastics| Rings| credit|

|00000004|2011| 300|Team Sports| Field| cash|

|00000005|2011| 200| Gymnastics| Ring| cash|

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

========Task 4========

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

| id| tdate|amount| category| product|spendby|

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

|00000003|06-05-2011| 100| Gymnastics| Rings| credit|

|00000004|12-17-2011| 300|Team Sports| Field| cash|

|00000000|06-26-2011| 200| Exercise|GymnasticsPro| cash|

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

==========

10-12-2022

==========

import org.apache.spark.SparkContext // rdd

import org.apache.spark.sql.SparkSession // dataframe

import org.apache.spark.SparkConf

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.functions.\_

Read dt.txt using header true as csv

df1 == filter spendby=cash

df2 == filter category=Exercise and spendby=cash

df3 == filter category=Exercise or spendby=casht

df4 == filter category =Exercise Gymnastics

===============

package pack

import org.apache.spark.SparkContext // rdd

import org.apache.spark.sql.SparkSession // dataframe

import org.apache.spark.SparkConf

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.functions.\_

object obj {

def main(args:Array[String]):Unit={

val conf = new SparkConf().setMaster("local[\*]").setAppName("first")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession

.builder

.config(conf)

.getOrCreate()

import spark.implicits.\_

val df = spark

.read

.format("csv")

.option("header","true")

.load("file:///C:/data/dt.txt")

df.show()

println

println("===Equals Filter===")

println

val df1 = df.filter(

col("spendby") === "cash"

)

df1.show()

println

println("===Multi Column and Filter===")

println

val df2 = df.filter(

col("category")==="Exercise"

&&

col("spendby")=== "cash"

)

df2.show()

println

println("===Multi Column or Filter===")

println

val df3 = df.filter(

col("category")==="Exercise"

||

col("spendby")=== "cash"

)

df3.show()

println

println("===Multi value Filter===")

println

val df4 = df.filter(

col("category") isin ("Exercise","Gymnastics")

)

df4.show()

val df7 = df.filter(

! ( col("category") === "Gymnastics" )

)

df7.show()

val df8 = df.filter(

! ( col("category") isin ("Gymnastics" ,"Exercise") )

)

df8.show()

}

}

==============================

package pack

import org.apache.spark.SparkContext // rdd

import org.apache.spark.sql.SparkSession // dataframe

import org.apache.spark.SparkConf

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.functions.\_

object project13 {

def main(args:Array[String]):Unit={

val conf = new SparkConf().setMaster("local[\*]").setAppName("first")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession

.builder

.config(conf)

.getOrCreate()

import spark.implicits.\_

val df = spark

.read

.format("csv")

.option("header","true")

.load("file:///D:/data/dt.txt")

df.show()

println

println("===Equals Filter===")

println

val df1 = df.filter(

col("spendby") === "cash"

)

df1.show()

println

println("===Multi Column and Filter===")

println

val df2 = df.filter(

col("category")==="Exercise"

&&

col("spendby")=== "cash"

)

df2.show()

println

println("===Multi Column or Filter===")

println

val df3 = df.filter(

col("category")==="Exercise"

||

col("spendby")=== "cash"

)

df3.show()

println

println("===Multi value Filter===")

println

val df4 = df.filter(

col("category") isin ("Exercise","Gymnastics")

)

df4.show()

println

println("===Multi value Filter===")

println

val df5 = df.filter(

! ( col("category") === "Gymnastics" )

)

df5.show()

println

println("===Multi value Filter===")

println

val df6 = df.filter(

! ( col("category") isin ("Gymnastics" ,"Exercise") )

)

df6.show()

}

}

Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties

22/12/10 10:06:27 INFO SparkContext: Running Spark version 2.4.7

22/12/10 10:06:28 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

22/12/10 10:06:28 INFO SparkContext: Submitted application: first

22/12/10 10:06:28 INFO SecurityManager: Changing view acls to: dell

22/12/10 10:06:28 INFO SecurityManager: Changing modify acls to: dell

22/12/10 10:06:28 INFO SecurityManager: Changing view acls groups to:

22/12/10 10:06:28 INFO SecurityManager: Changing modify acls groups to:

22/12/10 10:06:28 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(dell); groups with view permissions: Set(); users with modify permissions: Set(dell); groups with modify permissions: Set()

22/12/10 10:06:30 INFO Utils: Successfully started service 'sparkDriver' on port 64835.

22/12/10 10:06:30 INFO SparkEnv: Registering MapOutputTracker

22/12/10 10:06:30 INFO SparkEnv: Registering BlockManagerMaster

22/12/10 10:06:30 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information

22/12/10 10:06:30 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up

22/12/10 10:06:30 INFO DiskBlockManager: Created local directory at C:\Users\dell\AppData\Local\Temp\blockmgr-6ac07dfa-2709-42f7-8bbc-0a0c4926f257

22/12/10 10:06:30 INFO MemoryStore: MemoryStore started with capacity 866.4 MB

22/12/10 10:06:30 INFO SparkEnv: Registering OutputCommitCoordinator

22/12/10 10:06:30 INFO Utils: Successfully started service 'SparkUI' on port 4040.

22/12/10 10:06:30 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://Dell:4040

22/12/10 10:06:30 INFO Executor: Starting executor ID driver on host localhost

22/12/10 10:06:30 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 64858.

22/12/10 10:06:30 INFO NettyBlockTransferService: Server created on Dell:64858

22/12/10 10:06:30 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication policy

22/12/10 10:06:31 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, Dell, 64858, None)

22/12/10 10:06:31 INFO BlockManagerMasterEndpoint: Registering block manager Dell:64858 with 866.4 MB RAM, BlockManagerId(driver, Dell, 64858, None)

22/12/10 10:06:31 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, Dell, 64858, None)

22/12/10 10:06:31 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, Dell, 64858, None)

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

| id| tdate|amount| category| product|spendby|

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

|00000000|06-26-2011| 200| Exercise|GymnasticsPro| cash|

|00000001|05-26-2011| 300| Exercise|Weightlifting| credit|

|00000002|06-01-2011| 100| Exercise|GymnasticsPro| cash|

|00000003|06-05-2011| 100| Gymnastics| Rings| credit|

|00000004|12-17-2011| 300|Team Sports| Field| cash|

|00000005|02-14-2011| 200| Gymnastics| Ring| cash|

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

===Equals Filter===

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

| id| tdate|amount| category| product|spendby|

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

|00000000|06-26-2011| 200| Exercise|GymnasticsPro| cash|

|00000002|06-01-2011| 100| Exercise|GymnasticsPro| cash|

|00000004|12-17-2011| 300|Team Sports| Field| cash|

|00000005|02-14-2011| 200| Gymnastics| Ring| cash|

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

===Multi Column and Filter===

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

| id| tdate|amount|category| product|spendby|

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

|00000000|06-26-2011| 200|Exercise|GymnasticsPro| cash|

|00000002|06-01-2011| 100|Exercise|GymnasticsPro| cash|

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

===Multi Column or Filter===

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

| id| tdate|amount| category| product|spendby|

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

|00000000|06-26-2011| 200| Exercise|GymnasticsPro| cash|

|00000001|05-26-2011| 300| Exercise|Weightlifting| credit|

|00000002|06-01-2011| 100| Exercise|GymnasticsPro| cash|

|00000004|12-17-2011| 300|Team Sports| Field| cash|

|00000005|02-14-2011| 200| Gymnastics| Ring| cash|

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

===Multi value Filter===

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

| id| tdate|amount| category| product|spendby|

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

|00000000|06-26-2011| 200| Exercise|GymnasticsPro| cash|

|00000001|05-26-2011| 300| Exercise|Weightlifting| credit|

|00000002|06-01-2011| 100| Exercise|GymnasticsPro| cash|

|00000003|06-05-2011| 100|Gymnastics| Rings| credit|

|00000005|02-14-2011| 200|Gymnastics| Ring| cash|

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

===Multi value Filter===

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

| id| tdate|amount| category| product|spendby|

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

|00000000|06-26-2011| 200| Exercise|GymnasticsPro| cash|

|00000001|05-26-2011| 300| Exercise|Weightlifting| credit|

|00000002|06-01-2011| 100| Exercise|GymnasticsPro| cash|

|00000004|12-17-2011| 300|Team Sports| Field| cash|

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

===Multi value Filter===

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

| id| tdate|amount| category|product|spendby|

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

|00000004|12-17-2011| 300|Team Sports| Field| cash|

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

===========================

val df = spark

.read

.format("csv")

.option("header","true")

.load("file:///C:/data/dt.txt")

df.show()

val df1 = df.selectExpr(

"id",

"split(tdate,'-')[2] as year",

"amount",

"UPPER(category) as category",

"product",

"spendby",

"case when spendby='cash' then 0 else 1 end as status"

)

df1.show()

df1.printSchema()

======================================

===========================

10-12-2022 Assignment task1

===========================

package pack

import org.apache.spark.SparkContext // rdd

import org.apache.spark.sql.SparkSession // dataframe

import org.apache.spark.SparkConf

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.functions.\_

object project13 {

def main(args:Array[String]):Unit={

val conf = new SparkConf().setMaster("local[\*]").setAppName("first")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession

.builder

.config(conf)

.getOrCreate()

import spark.implicits.\_

val df = spark

.read

.format("csv")

.option("header","true")

.load("file:///D:/data/dt.txt")

df.show()

println("====== Task1 ======")

val df1 = df.selectExpr(

"id",

"split (tdate,'-')[2] as year",

"amount",

"UPPER(category) as category",

"product",

"spendby",

"case when spendby='cash' then 0 else 1 end as status"

)

df1.show()

df1.printSchema()

println

println("==== Change\_date\_format ====")

println

val df2 = df.selectExpr(exprs = "\*",

"""date\_format

(to\_date(tdate,'MM-dd-yyyy'),

'yyyy-MM-dd') as Change\_date\_format"""

)

df2.show()

}

}

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

| id| tdate|amount| category| product|spendby|

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

|00000000|06-26-2011| 200| Exercise|GymnasticsPro| cash|

|00000001|05-26-2011| 300| Exercise|Weightlifting| credit|

|00000002|06-01-2011| 100| Exercise|GymnasticsPro| cash|

|00000003|06-05-2011| 100| Gymnastics| Rings| credit|

|00000004|12-17-2011| 300|Team Sports| Field| cash|

|00000005|02-14-2011| 200| Gymnastics| Ring| cash|

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

====== Task1 ======

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

| id|year|amount| category| product|spendby|status|

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

|00000000|2011| 200| EXERCISE|GymnasticsPro| cash| 0|

|00000001|2011| 300| EXERCISE|Weightlifting| credit| 1|

|00000002|2011| 100| EXERCISE|GymnasticsPro| cash| 0|

|00000003|2011| 100| GYMNASTICS| Rings| credit| 1|

|00000004|2011| 300|TEAM SPORTS| Field| cash| 0|

|00000005|2011| 200| GYMNASTICS| Ring| cash| 0|

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

root

|-- id: string (nullable = true)

|-- year: string (nullable = true)

|-- amount: string (nullable = true)

|-- category: string (nullable = true)

|-- product: string (nullable = true)

|-- spendby: string (nullable = true)

|-- status: integer (nullable = false)

==== Change\_date\_format ====

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

| id| tdate|amount| category| product|spendby|Change\_date\_format|

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

|00000000|06-26-2011| 200| Exercise|GymnasticsPro| cash| 2011-06-26|

|00000001|05-26-2011| 300| Exercise|Weightlifting| credit| 2011-05-26|

|00000002|06-01-2011| 100| Exercise|GymnasticsPro| cash| 2011-06-01|

|00000003|06-05-2011| 100| Gymnastics| Rings| credit| 2011-06-05|

|00000004|12-17-2011| 300|Team Sports| Field| cash| 2011-12-17|

|00000005|02-14-2011| 200| Gymnastics| Ring| cash| 2011-02-14|

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

=============

11-12-2022

=============

package pack

import org.apache.spark.SparkContext // rdd

import org.apache.spark.sql.SparkSession // dataframe

import org.apache.spark.SparkConf

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.functions.\_

object project13 {

def main(args:Array[String]):Unit={

val conf = new SparkConf().setMaster("local[\*]").setAppName("first")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession

.builder

.config(conf)

.getOrCreate()

import spark.implicits.\_

val df = spark

.read

.format("csv")

.option("header","true")

.load("file:///D:/data/dt.txt")

df.show()

val df1 = df.withColumn("year",expr("split(tdate,'-')[2]"))

.withColumnRenamed("tdate","year")

.withColumn("status",expr("case when spendby = 'cash' then 1 else 0 end"))

df1.show()

}

}

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

| id| tdate|amount| category| product|spendby|

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

|00000000|06-26-2011| 200| Exercise|GymnasticsPro| cash|

|00000001|05-26-2011| 300| Exercise|Weightlifting| credit|

|00000002|06-01-2011| 100| Exercise|GymnasticsPro| cash|

|00000003|06-05-2011| 100| Gymnastics| Rings| credit|

|00000004|12-17-2011| 300|Team Sports| Field| cash|

|00000005|02-14-2011| 200| Gymnastics| Ring| cash|

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

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

| id| year|amount| category| product|spendby|year|status|

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

|00000000|06-26-2011| 200| Exercise|GymnasticsPro| cash|2011| 1|

|00000001|05-26-2011| 300| Exercise|Weightlifting| credit|2011| 0|

|00000002|06-01-2011| 100| Exercise|GymnasticsPro| cash|2011| 1|

|00000003|06-05-2011| 100| Gymnastics| Rings| credit|2011| 0|

|00000004|12-17-2011| 300|Team Sports| Field| cash|2011| 1|

|00000005|02-14-2011| 200| Gymnastics| Ring| cash|2011| 1|

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

=========

Join Code

=========

val df1 = spark.read.format("csv")

.option("header","true")

.load("file:///C:/data/j11.csv")

df1.show()

val df2 = spark.read.format("csv")

.option("header","true")

.load("file:///C:/data/j22.csv")

df2.show()

println

println("===inner join===")

println

val innerdf = df1.join( df2, Seq("id") , "inner")

innerdf.show()

println

println("===left join===")

println

val leftdf = df1.join( df2, Seq("id") , "left")

leftdf.show()

println

println("===right join===")

println

val rightdf = df1.join( df2, Seq("id") , "right")

rightdf.show()

println

println("===full join===")

println

val fulldf = df1.join( df2, Seq("id") , "full")

.orderBy("id")

fulldf.show()

val df2 = spark.read.format("csv")

.option("header","true")

.load("file:///C:/data/j22.csv")

df2.show()

val finaldf = df1.join(df2,Seq("id"),"left\_anti")

finaldf.show()

/\* val listids = df2

.select("id")

.rdd

.map( x => x.mkString(","))

.collect()

.toList

println(listids)

val finaldf = df1.filter( ! (col("id").isin(listids: \_\*)))

finaldf.show()

\*/

===========

Task 1 ---

Practise all the Joins

Task 2 ---

val crossdf=df1.crossJoin(df2)

crossdf.show()

Task 3 --- Optional

Analyse uber case (Dataset and Requirement)

Task 4 ----Optional

If possible Perform all the joins using SQL tempview

===========

======================

Assignment Tasks 1 & 2

======================

package pack

import org.apache.spark.SparkContext // rdd

import org.apache.spark.sql.SparkSession // dataframe

import org.apache.spark.SparkConf

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.functions.\_

object project13 {

def main(args:Array[String]):Unit={

val conf = new SparkConf().setMaster("local[\*]").setAppName("first")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession

.builder

.config(conf)

.getOrCreate()

import spark.implicits.\_

val df1 = spark

.read

.format("csv")

.option("header","true")

.load("file:///D:/data/j11.csv")

df1.show()

val df2 = spark

.read

.format("csv")

.option("header","true")

.load("file:///D:/data/j22.csv")

df2.show()

println

println("======Left join======")

val leftdf = df1.join(df2,Seq("id"),"left")

leftdf.show()

println

println("======Inner join======")

val innerdf = df1.join(df2,Seq("id"),"inner")

innerdf.show()

println

println("======Right join======")

val rightdf = df1.join(df2,Seq("id"),"right")

rightdf.show()

println

println("======Full join======")

val fulldf = df1.join(df2,Seq("id"),"full")

.orderBy("id")

fulldf.show()

println

println("======Left\_anti join======")

val finaldf = df1.join(df2,Seq("id"),"left\_anti")

leftdf.show()

println

println("======Cross join======")

val crossdf = df1.crossJoin(df2)

crossdf.show()

}

}

Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties

22/12/15 11:34:40 INFO SparkContext: Running Spark version 2.4.7

22/12/15 11:34:40 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

22/12/15 11:34:40 INFO SparkContext: Submitted application: first

22/12/15 11:34:40 INFO SecurityManager: Changing view acls to: dell

22/12/15 11:34:40 INFO SecurityManager: Changing modify acls to: dell

22/12/15 11:34:40 INFO SecurityManager: Changing view acls groups to:

22/12/15 11:34:40 INFO SecurityManager: Changing modify acls groups to:

22/12/15 11:34:40 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(dell); groups with view permissions: Set(); users with modify permissions: Set(dell); groups with modify permissions: Set()

22/12/15 11:34:42 INFO Utils: Successfully started service 'sparkDriver' on port 64445.

22/12/15 11:34:42 INFO SparkEnv: Registering MapOutputTracker

22/12/15 11:34:42 INFO SparkEnv: Registering BlockManagerMaster

22/12/15 11:34:42 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information

22/12/15 11:34:42 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up

22/12/15 11:34:42 INFO DiskBlockManager: Created local directory at C:\Users\dell\AppData\Local\Temp\blockmgr-53f191c0-c919-4e35-ac36-5951567c3bbc

22/12/15 11:34:42 INFO MemoryStore: MemoryStore started with capacity 866.4 MB

22/12/15 11:34:42 INFO SparkEnv: Registering OutputCommitCoordinator

22/12/15 11:34:42 INFO Utils: Successfully started service 'SparkUI' on port 4040.

22/12/15 11:34:42 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://Dell:4040

22/12/15 11:34:42 INFO Executor: Starting executor ID driver on host localhost

22/12/15 11:34:43 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 64471.

22/12/15 11:34:43 INFO NettyBlockTransferService: Server created on Dell:64471

22/12/15 11:34:43 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication policy

22/12/15 11:34:43 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, Dell, 64471, None)

22/12/15 11:34:43 INFO BlockManagerMasterEndpoint: Registering block manager Dell:64471 with 866.4 MB RAM, BlockManagerId(driver, Dell, 64471, None)

22/12/15 11:34:43 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, Dell, 64471, None)

22/12/15 11:34:43 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, Dell, 64471, None)

+---+----+

| id|name|

+---+----+

| 1| raj|

| 2|ravi|

| 3| sai|

| 5|rani|

+---+----+

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

| id|product|

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

| 1|cookies|

| 3| mobile|

| 7| laptop|

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

======Left join======

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

| id|name|product|

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

| 1| raj|cookies|

| 2|ravi| null|

| 3| sai| mobile|

| 5|rani| null|

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

======Inner join======

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

| id|name|product|

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

| 1| raj|cookies|

| 3| sai| mobile|

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

======Right join======

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

| id|name|product|

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

| 1| raj|cookies|

| 3| sai| mobile|

| 7|null| laptop|

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

======Full join======

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

| id|name|product|

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

| 1| raj|cookies|

| 2|ravi| null|

| 3| sai| mobile|

| 5|rani| null|

| 7|null| laptop|

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

======Left\_anti join======

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

| id|name|product|

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

| 1| raj|cookies|

| 2|ravi| null|

| 3| sai| mobile|

| 5|rani| null|

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

======Cross join======

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

| id|name| id|product|

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

| 1| raj| 1|cookies|

| 2|ravi| 1|cookies|

| 3| sai| 1|cookies|

| 5|rani| 1|cookies|

| 1| raj| 3| mobile|

| 2|ravi| 3| mobile|

| 3| sai| 3| mobile|

| 5|rani| 3| mobile|

| 1| raj| 7| laptop|

| 2|ravi| 7| laptop|

| 3| sai| 7| laptop|

| 5|rani| 7| laptop|

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

============

Task3

package pack

import org.apache.spark.SparkContext // rdd

import org.apache.spark.sql.SparkSession // dataframe

import org.apache.spark.SparkConf

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.functions.\_

object project13 {

def main(args:Array[String]):Unit={

val conf = new SparkConf().setMaster("local[\*]").setAppName("first")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession

.builder

.config(conf)

.getOrCreate()

import spark.implicits.\_

val df = spark.read.format("csv")

.option("header","true")

.load("file:///D:/data/uber.csv")

df.show()

df.createOrReplaceTempView("df1temp")

val df2 = spark.sql("""select dispatching\_base\_number,date,

sum(active\_vehicles) as AV From df1temp

group by dispatching\_base\_number,date

order by AV desc""")

println("====== Aggregate Query ======")

df2.show()

}

}

Result:

Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties

22/12/16 02:01:38 INFO SparkContext: Running Spark version 2.4.7

22/12/16 02:01:38 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

22/12/16 02:01:38 INFO SparkContext: Submitted application: first

22/12/16 02:01:39 INFO SecurityManager: Changing view acls to: dell

22/12/16 02:01:39 INFO SecurityManager: Changing modify acls to: dell

22/12/16 02:01:39 INFO SecurityManager: Changing view acls groups to:

22/12/16 02:01:39 INFO SecurityManager: Changing modify acls groups to:

22/12/16 02:01:39 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(dell); groups with view permissions: Set(); users with modify permissions: Set(dell); groups with modify permissions: Set()

22/12/16 02:01:40 INFO Utils: Successfully started service 'sparkDriver' on port 61756.

22/12/16 02:01:40 INFO SparkEnv: Registering MapOutputTracker

22/12/16 02:01:40 INFO SparkEnv: Registering BlockManagerMaster

22/12/16 02:01:40 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information

22/12/16 02:01:40 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up

22/12/16 02:01:40 INFO DiskBlockManager: Created local directory at C:\Users\dell\AppData\Local\Temp\blockmgr-f61355f2-faca-4463-9ef9-c392899be237

22/12/16 02:01:40 INFO MemoryStore: MemoryStore started with capacity 866.4 MB

22/12/16 02:01:40 INFO SparkEnv: Registering OutputCommitCoordinator

22/12/16 02:01:40 INFO Utils: Successfully started service 'SparkUI' on port 4040.

22/12/16 02:01:40 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://Dell:4040

22/12/16 02:01:40 INFO Executor: Starting executor ID driver on host localhost

22/12/16 02:01:41 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 61779.

22/12/16 02:01:41 INFO NettyBlockTransferService: Server created on Dell:61779

22/12/16 02:01:41 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication policy

22/12/16 02:01:41 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, Dell, 61779, None)

22/12/16 02:01:41 INFO BlockManagerMasterEndpoint: Registering block manager Dell:61779 with 866.4 MB RAM, BlockManagerId(driver, Dell, 61779, None)

22/12/16 02:01:41 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, Dell, 61779, None)

22/12/16 02:01:41 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, Dell, 61779, None)

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

|dispatching\_base\_number| date|active\_vehicles|trips|

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

| B02512|1/1/2017| 190| 1132|

| B02765|1/1/2017| 225| 1765|

| B02764|1/1/2017| 3427|29421|

| B02682|1/1/2017| 945| 7679|

| B02617|1/1/2017| 1228| 9537|

| B02598|1/1/2017| 870| 6903|

| B02598|1/2/2017| 785| 4768|

| B02617|1/2/2017| 1137| 7065|

| B02512|1/2/2017| 175| 875|

| B02682|1/2/2017| 890| 5506|

| B02765|1/2/2017| 196| 1001|

| B02764|1/2/2017| 3147|19974|

| B02765|1/3/2017| 201| 1526|

| B02617|1/3/2017| 1188|10664|

| B02598|1/3/2017| 818| 7432|

| B02682|1/3/2017| 915| 8010|

| B02512|1/3/2017| 173| 1088|

| B02764|1/3/2017| 3215|29729|

| B02512|1/4/2017| 147| 791|

| B02682|1/4/2017| 812| 5621|

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

only showing top 20 rows

====== Aggregate Query ======

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

|dispatching\_base\_number| date| AV|

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

| B02764|2/13/2017|4395.0|

| B02764|2/20/2017|4384.0|

| B02764|2/27/2017|4253.0|

| B02764| 2/6/2017|4170.0|

| B02764|2/12/2017|4137.0|

| B02764|2/14/2017|4129.0|

| B02764|1/30/2017|4124.0|

| B02764|2/19/2017|4110.0|

| B02764|2/26/2017|4101.0|

| B02764| 2/5/2017|4093.0|

| B02764|1/23/2017|4040.0|

| B02764|2/21/2017|3981.0|

| B02764|1/16/2017|3975.0|

| B02764|2/24/2017|3965.0|

| B02764|1/29/2017|3959.0|

| B02764|2/28/2017|3952.0|

| B02764|1/31/2017|3947.0|

| B02764|2/25/2017|3934.0|

| B02764|1/22/2017|3889.0|

| B02764| 2/4/2017|3856.0|

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

only showing top 20 rows

================

17-12-2022

================

import org.apache.spark.SparkContext // rdd

import org.apache.spark.sql.SparkSession // dataframe

import org.apache.spark.SparkConf

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.types.IntegerType

import org.apache.spark.sql.functions.upper

import org.apache.spark.sql.catalyst.expressions.Upper

-------------------------------------------------------

package pack

import org.apache.spark.SparkContext // rdd

import org.apache.spark.sql.SparkSession // dataframe

import org.apache.spark.SparkConf

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.types.IntegerType

import org.apache.spark.sql.functions.upper

import org.apache.spark.sql.catalyst.expressions.Upper

import org.apache.spark.sql.functions.\_

object obj {

def main(args:Array[String]):Unit={

val conf = new SparkConf().setMaster("local[\*]").setAppName("first")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession

.builder

.config(conf)

.getOrCreate()

import spark.implicits.\_

val df1 = spark.read.format("csv")

.option("header","true")

.load("file:///C:/data/dt.txt")

df1.show()

val finaldf = df1

.groupBy("category")

.agg(

sum("amount")

.cast(IntegerType)

.as("total")

,

count("amount")

.as("cnt")

)

finaldf.show()

val finaldf1 = df1

.groupBy("category","spendby")

.agg(

sum("amount")

.cast(IntegerType)

.as("total")

,

count("amount")

.as("cnt")

)

finaldf1.show()

}

}

---------------------------------------------------------------------

package pack

import org.apache.spark.SparkContext // rdd

import org.apache.spark.sql.SparkSession // dataframe

import org.apache.spark.SparkConf

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.types.IntegerType

import org.apache.spark.sql.functions.upper

import org.apache.spark.sql.catalyst.expressions.Upper

import org.apache.spark.sql.functions.\_

object project13 {

def main(args:Array[String]):Unit={

val conf = new SparkConf().setMaster("local[\*]").setAppName("first")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession

.builder

.config(conf)

.getOrCreate()

import spark.implicits.\_

val df1 = spark.read.format("csv")

.option("header","true")

.load("file:///D:/data/dt.txt")

df1.show()

val finaldf = df1

.groupBy("category")

.agg(

sum("amount")

.cast(IntegerType)

.as("total")

,

count("amount")

.as("cnt")

)

finaldf.show()

val finaldf1 = df1

.groupBy("category","spendby")

.agg(

sum("amount")

.cast(IntegerType)

.as("total")

,

count("amount")

.as("cnt")

)

finaldf1.show()

}

}

Out put

============

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

| id| tdate|amount| category| product|spendby|

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

|00000000|06-26-2011| 200| Exercise|GymnasticsPro| cash|

|00000001|05-26-2011| 300| Exercise|Weightlifting| credit|

|00000002|06-01-2011| 100| Exercise|GymnasticsPro| cash|

|00000003|06-05-2011| 100| Gymnastics| Rings| credit|

|00000004|12-17-2011| 300|Team Sports| Field| cash|

|00000005|02-14-2011| 200| Gymnastics| Ring| cash|

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

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

| category|total|cnt|

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

| Gymnastics| 300| 2|

|Team Sports| 300| 1|

| Exercise| 600| 3|

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

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

| category|spendby|total|cnt|

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

| Gymnastics| cash| 200| 1|

| Exercise| cash| 300| 2|

| Exercise| credit| 300| 1|

|Team Sports| cash| 300| 1|

| Gymnastics| credit| 100| 1|

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

----------------------------------------------------------------

========================================

----------------------------------------

Revision data

package pack

import org.apache.spark.SparkContext // rdd

import org.apache.spark.sql.SparkSession // dataframe

import org.apache.spark.SparkConf

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.types.IntegerType

import org.apache.spark.sql.functions.upper

import org.apache.spark.sql.catalyst.expressions.Upper

import org.apache.spark.sql.functions.\_

object obj {

case class schema(txnno:String,

txndate:String,

custno:String,

amount:String,

category:String,

product:String,

city:String,

state:String,

spendby:String)

def main(args:Array[String]):Unit={

val conf = new SparkConf().setAppName("Revision").setMaster("local[\*]")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val liscol = List("txnno","txndate","custno","amount","category","product","city","state","spendby")

val spark = SparkSession.builder().getOrCreate()

import spark.implicits.\_

println

println("======schema df=======")

println

val file1 = sc.textFile("file:///C:/data/revdata/file1.txt")

val gymdata = file1.filter( x => x.contains("Gymnastics") )

val mapsplit = gymdata.map( x => x.split(","))

val schemardd = mapsplit.map( x => schema(x(0),x(1),x(2),x(3),x(4),x(5),x(6),x(7),x(8)))

val filterschemardd = schemardd.filter( x=> x.product.contains("Gymnastics"))

//filterschemardd.take(10).foreach(println)

val schemadf = filterschemardd.toDF()

schemadf.show(5)

println

println("====== rowdf======")

println

val file2 = sc.textFile("file:///C:/data/revdata/file2.txt")

val mapsplit1 = file2.map( x => x.split(","))

val rowrdd = mapsplit1.map( x => Row(x(0),x(1),x(2),x(3),x(4),x(5),x(6),x(7),x(8)))

val simpleSchema = StructType(Array(

StructField("txnno",StringType,true),

StructField("txndate",StringType,true),

StructField("custno",StringType,true),

StructField("amount", StringType, true),

StructField("category", StringType, true),

StructField("product", StringType, true),

StructField("city", StringType, true),

StructField("state", StringType, true),

StructField("spendby", StringType, true)

))

val rowdf = spark.createDataFrame(rowrdd, simpleSchema)

rowdf.show(5)

println

println("====== file 3 csvdf======")

println

val csvdf = spark

.read

.format("csv")

.option("header","true")

.load("file:///C:/data/revdata/file3.txt")

csvdf.show(5)

println

println("====== file 4 jsondf======")

println

val jsondf = spark

.read

.format("json")

.load("file:///C:/data/revdata/file4.json")

.select(liscol.map(col): \_\*)

jsondf.show(5)

println

println("====== file 5 parquetdf======")

println

val parquetdf = spark

.read

.load("file:///C:/data/revdata/file5.parquet")

parquetdf.show(5)

println

println("====== file 6 xmldf======")

println

val xmldf = spark

.read

.format("xml")

.option("rowtag","txndata")

.load("file:///C:/data/revdata/file6")

.select(liscol.map(col): \_\*)

xmldf.show(6)

println

println("====== union df======")

println

val uniondf = schemadf

.union(rowdf)

.union(csvdf)

.union(jsondf)

.union(parquetdf)

.union(xmldf)

uniondf.show(5)

println(uniondf.count)

println

println("====== proc df======")

println

val procdf = uniondf.withColumn("txndate", expr("split(txndate,'-')[2]"))

.withColumnRenamed("txndate","year")

.withColumn("status",expr("case when spendby='cash' then 1 else 0 end"))

.filter( col("txnno") > 50000 )

procdf.show(5)

println

println("====== agg df======")

println

val aggdf = procdf.groupBy("category")

.agg(sum("amount").as("total"))

aggdf.show()

println

println("====== write df======")

println

/\*uniondf

.write

.format("avro").mode("append")

.partitionBy("category")

.save("file:///C:/data/uniondfrev")\*/

println("====data written ===")

}

}

-----------------------------------------

=========================================

===============

Spark Revision

===============

package pack

import org.apache.spark.SparkContext // rdd

import org.apache.spark.sql.SparkSession // dataframe

import org.apache.spark.SparkConf

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.types.IntegerType

import org.apache.spark.sql.functions.upper

import org.apache.spark.sql.catalyst.expressions.Upper

import org.apache.spark.sql.functions.\_

object project13 {

case class schema(txnno:String,

txndate:String,

custno:String,

amount:String,

category:String,

product:String,

city:String,

state:String,

spendby:String)

def main(args:Array[String]):Unit={

val conf = new SparkConf().setAppName("Revision").setMaster("local[\*]")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val liscol = List("txnno","txndate","custno","amount","category","product","city","state","spendby")

val spark = SparkSession.builder().getOrCreate()

import spark.implicits.\_

println

println("======schema df=======")

println

val file1 = sc.textFile("file:///D:/data/revdata/file1.txt")

val gymdata = file1.filter( x => x.contains("Gymnastics") )

val mapsplit = gymdata.map( x => x.split(","))

val schemardd = mapsplit.map( x => schema(x(0),x(1),x(2),x(3),x(4),x(5),x(6),x(7),x(8)))

val filterschemardd = schemardd.filter( x=> x.product.contains("Gymnastics"))

//filterschemardd.take(10).foreach(println)

val schemadf = filterschemardd.toDF()

schemadf.show(5)

println

println("====== rowdf======")

println

val file2 = sc.textFile("file:///D:/data/revdata/file2.txt")

val mapsplit1 = file2.map( x => x.split(","))

val rowrdd = mapsplit1.map( x => Row(x(0),x(1),x(2),x(3),x(4),x(5),x(6),x(7),x(8)))

val simpleSchema = StructType(Array(

StructField("txnno",StringType,true),

StructField("txndate",StringType,true),

StructField("custno",StringType,true),

StructField("amount", StringType, true),

StructField("category", StringType, true),

StructField("product", StringType, true),

StructField("city", StringType, true),

StructField("state", StringType, true),

StructField("spendby", StringType, true)

))

val rowdf = spark.createDataFrame(rowrdd, simpleSchema)

rowdf.show(5)

println

println("====== file 3 csvdf======")

println

val csvdf = spark

.read

.format("csv")

.option("header","true")

.load("file:///D:/data/revdata/file3.txt")

csvdf.show(5)

println

println("====== file 4 jsondf======")

println

val jsondf = spark

.read

.format("json")

.load("file:///D:/data/revdata/file4.json")

.select(liscol.map(col): \_\*)

jsondf.show(5)

println

println("====== file 5 parquetdf======")

println

val parquetdf = spark

.read

.load("file:///D:/data/revdata/file5.parquet")

parquetdf.show(5)

println

println("====== file 6 xmldf======")

println

val xmldf = spark

.read

.format("xml")

.option("rowtag","txndata")

.load("file:///D:/data/revdata/file6")

.select(liscol.map(col): \_\*)

xmldf.show(6)

println

println("====== union df======")

println

val uniondf = schemadf

.union(rowdf)

.union(csvdf)

.union(jsondf)

.union(parquetdf)

.union(xmldf)

uniondf.show(5)

println(uniondf.count)

println

println("====== proc df======")

println

val procdf = uniondf.withColumn("txndate", expr("split(txndate,'-')[2]"))

.withColumnRenamed("txndate","year")

.withColumn("status",expr("case when spendby='cash' then 1 else 0 end"))

.filter( col("txnno") > 50000 )

procdf.show(5)

println

println("====== agg df======")

println

val aggdf = procdf.groupBy("category")

.agg(sum("amount").as("total"))

aggdf.show()

println

println("====== write df======")

println

uniondf

.write

.format("avro").mode("append")

.partitionBy("category")

.save("file:///D:/data/uniondfrev")

println("====data written ====")

val df1 = spark

.read

.format("csv")

.option("header","true")

.load("file:///D:/data/j11.csv")

df1.show()

val df2 = spark

.read

.format("csv")

.option("header","true")

.load("file:///D:/data/j22.csv")

df2.show()

println

println("======Left join======")

val leftdf = df1.join(df2,Seq("id"),"left")

leftdf.show()

println

println("======Inner join======")

val innerdf = df1.join(df2,Seq("id"),"inner")

innerdf.show()

println

println("======Right join======")

val rightdf = df1.join(df2,Seq("id"),"right")

rightdf.show()

println

println("======Full join======")

val fulldf = df1.join(df2,Seq("id"),"full")

.orderBy("id")

fulldf.show()

println

println("======Left\_anti join======")

val finaldf = df1.join(df2,Seq("id"),"left\_anti")

leftdf.show()

println

println("======Cross join======")

val crossdf = df1.crossJoin(df2)

crossdf.show()

println

println("======Left Semi join======")

val left\_semidf = df1.join(df2,Seq("id"),"left\_semi")

left\_semidf.show()

}

}

======================================

-----------------Out put--------------

======schema df=======

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

| txnno| txndate| custno|amount| category| product| city| state|spendby|

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

|00000003|06-05-2011|4002199|198.19|Gymnastics| Gymnastics Rings| Milwaukee|Wisconsin| credit|

|00000023|05-02-2011|4007596|099.50|Gymnastics| Gymnastics Rings| Springfield| Illinois| credit|

|00000082|01-16-2011|4001098|021.23|Gymnastics| Gymnastics Rings| Tampa| Florida| cash|

|00000087|06-05-2011|4001050|089.56|Gymnastics| Gymnastics Mats|West Valley City| Utah| credit|

|00000111|03-05-2011|4000401|173.56|Gymnastics|Gymnastics Protec...| Portland| Oregon| credit|

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

only showing top 5 rows

====== rowdf======

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

| txnno| txndate| custno|amount| category| product| city| state|spendby|

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

|00021035|08-06-2011|4008418|161.62| Exercise & Fitness|Cardio Machine Ac...| Gresham| Oregon| credit|

|00021036|05-20-2011|4001083|162.04|Outdoor Play Equi...| Outdoor Playsets|Saint Paul| Minnesota| credit|

|00021037|05-18-2011|4004039|162.40| Gymnastics| Pommel Horses|Pittsburgh| Pennsylvania| credit|

|00021038|06-29-2011|4004018|067.71| Outdoor Recreation| Riding Scooters| Gresham| Oregon| credit|

|00021039|05-16-2011|4008593|147.31| Outdoor Recreation| Deck Shuffleboard| Cambridge|Massachusetts| credit|

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

only showing top 5 rows

====== file 3 csvdf======

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

| txnno| txndate| custno|amount| category| product| city| state|spendby|

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

|00047931|03-22-2011|4000263|041.89| Team Sports| Curling| Dayton| Ohio| credit|

|00047932|05-03-2011|4001699|169.15| Jumping|Trampoline Access...| New York| New York| credit|

|00047933|06-16-2011|4000001|032.52|Racquet Sports| Tennis|Los Angeles|California| credit|

|00047934|01-09-2011|4002860|124.31| Water Sports| Surfing| Columbia| Missouri| credit|

|00047935|12-27-2011|4001211|150.54| Puzzles| Jigsaw Puzzles|Los Angeles|California| credit|

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

only showing top 5 rows

====== file 4 jsondf======

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

| txnno| txndate| custno|amount| category| product| city| state|spendby|

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

|00060747|09-06-2011|4002143|074.68| Winter Sports| Sledding|Long Beach|California| credit|

|00060748|09-17-2011|4000219|138.63| Jumping| Pogo Sticks| Vancouver|Washington| credit|

|00060749|10-02-2011|4009381|173.42| Team Sports| Cricket| Everett|Washington| credit|

|00060750|02-08-2011|4007813|155.85|Exercise & Fitness|Weightlifting Mac...| Buffalo| New York| credit|

|00060751|09-09-2011|4009254|101.17| Indoor Games| Foosball|Long Beach|California| credit|

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

only showing top 5 rows

====== file 5 parquetdf======

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

| txnno| txndate| custno|amount| category| product| city| state|spendby|

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

|00075990|10-27-2011|4004658|063.14| Water Sports| Whitewater Rafting|Springfield| Illinois| credit|

|00075991|11-23-2011|4009784|005.46|Outdoor Play Equi...| Outdoor Playsets| Cincinnati| Ohio| cash|

|00075992|04-20-2011|4000281|175.44| Exercise & Fitness|Abdominal Equipment| Long Beach|California| credit|

|00075993|05-12-2011|4005096|061.08| Team Sports| Cheerleading| Montgomery| Alabama| credit|

|00075994|12-16-2011|4002516|005.20| Outdoor Recreation| Rock Climbing|Chattanooga| Tennessee| cash|

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

only showing top 5 rows

====== file 6 xmldf======

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

|txnno| txndate| custno|amount| category| product| city| state|spendby|

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

|88973|10-23-2011|4000564|180.42| Winter Sports| Snowshoeing| Portland| Oregon| credit|

|88974|11-24-2011|4004615| 76.45|Exercise & Fitness|Cardio Machines| Kansas City| Kansas| credit|

|88975|04-14-2011|4009978|187.67| Winter Sports| Snowshoeing| New York| New York| credit|

|88976|04-11-2011|4003490|196.06| Indoor Games| Air Hockey| Boise| Idaho| credit|

|88977|04-02-2011|4007867| 36.07| Team Sports| Basketball|Coral Springs| Florida| cash|

|88978|02-12-2011|4006836| 40.78| Gymnastics| Springboards| Santa Ana|California| cash|

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

only showing top 6 rows

====== union df======

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

| txnno| txndate| custno|amount| category| product| city| state|spendby|

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

|00000003|06-05-2011|4002199|198.19|Gymnastics| Gymnastics Rings| Milwaukee|Wisconsin| credit|

|00000023|05-02-2011|4007596|099.50|Gymnastics| Gymnastics Rings| Springfield| Illinois| credit|

|00000082|01-16-2011|4001098|021.23|Gymnastics| Gymnastics Rings| Tampa| Florida| cash|

|00000087|06-05-2011|4001050|089.56|Gymnastics| Gymnastics Mats|West Valley City| Utah| credit|

|00000111|03-05-2011|4000401|173.56|Gymnastics|Gymnastics Protec...| Portland| Oregon| credit|

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

only showing top 5 rows

75534

====== proc df======

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

| txnno|year| custno|amount| category| product| city| state|spendby|status|

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

|00050001|2011|4004079|189.90| Games|Poker Chips & Sets| Cincinnati| Ohio| credit| 0|

|00050002|2011|4009271|092.19| Team Sports| Cheerleading| Richmond | Virginia| credit| 0|

|00050003|2011|4004362|051.91|Exercise & Fitness| Foam Rollers| Westminster| Colorado| credit| 0|

|00050004|2011|4001893|039.72| Water Sports| Swimming|San Francisco|California| credit| 0|

|00050005|2011|4004986|065.25| Water Sports| Wetsuits| St. Louis | Missouri| credit| 0|

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

only showing top 5 rows

====== agg df======

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

| category| total|

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

| Gymnastics| 311946.1299999998|

| Winter Sports| 298795.6600000001|

| Jumping|176760.88999999998|

| Team Sports| 572796.0100000001|

| Air Sports| 98705.80000000003|

| Indoor Games|259715.26999999993|

| Games| 339716.3|

|Outdoor Play Equi...| 263790.9900000002|

| Water Sports|496075.46000000014|

| Puzzles|57356.020000000004|

| Outdoor Recreation| 796503.0499999998|

| Racquet Sports|151250.82000000007|

| Combat Sports| 155792.32|

| Dancing|39585.350000000006|

| Exercise & Fitness| 691531.6700000004|

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

====== write df======

====data written ====

+---+----+

| id|name|

+---+----+

| 1| raj|

| 2|ravi|

| 3| sai|

| 5|rani|

+---+----+

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

| id|product|

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

| 1|cookies|

| 3| mobile|

| 7| laptop|

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

======Left join======

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

| id|name|product|

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

| 1| raj|cookies|

| 2|ravi| null|

| 3| sai| mobile|

| 5|rani| null|

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

======Inner join======

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

| id|name|product|

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

| 1| raj|cookies|

| 3| sai| mobile|

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

======Right join======

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

| id|name|product|

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

| 1| raj|cookies|

| 3| sai| mobile|

| 7|null| laptop|

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

======Full join======

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

| id|name|product|

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

| 1| raj|cookies|

| 2|ravi| null|

| 3| sai| mobile|

| 5|rani| null|

| 7|null| laptop|

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

======Left\_anti join======

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

| id|name|product|

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

| 1| raj|cookies|

| 2|ravi| null|

| 3| sai| mobile|

| 5|rani| null|

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

======Cross join======

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

| id|name| id|product|

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

| 1| raj| 1|cookies|

| 2|ravi| 1|cookies|

| 3| sai| 1|cookies|

| 5|rani| 1|cookies|

| 1| raj| 3| mobile|

| 2|ravi| 3| mobile|

| 3| sai| 3| mobile|

| 5|rani| 3| mobile|

| 1| raj| 7| laptop|

| 2|ravi| 7| laptop|

| 3| sai| 7| laptop|

| 5|rani| 7| laptop|

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

======Left Semi join======

+---+----+

| id|name|

+---+----+

| 1| raj|

| 3| sai|

+---+----+

--------------------------------------

======================================

==========================

Task 1 -----

Flatten pic.json

--------------------

package pack

import org.apache.spark.SparkContext // rdd

import org.apache.spark.sql.SparkSession // dataframe

import org.apache.spark.SparkConf

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.types.IntegerType

import org.apache.spark.sql.functions.upper

import org.apache.spark.sql.catalyst.expressions.Upper

import org.apache.spark.sql.functions.\_

object project13 {

def main(args:Array[String]):Unit={

val conf = new SparkConf().setAppName("Revision").setMaster("local[\*]")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession.builder().getOrCreate()

import spark.implicits.\_

println("====== Task1 Pic.json ======")

val df = spark

.read

.format("json")

.option("multiline","true")

.load("file:///D:/data/pic.json")

df.show()

df.printSchema()

val flatdf = df.select(

"id",

"type",

"name",

"image.height",

"image.url",

"image.width"

)

flatdf.show()

flatdf.printSchema()

/\*val flatdf = df.select(

"Years",

"address.permanentAddress",

"address.temporaryAddress",

"org",

"trainer"

)

flatdf.show()

flatdf.printSchema()\*/

}

}

-------------

Out put:

--------

====== Task1 Pic.json ======

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

| id| image| name| type|

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

|000|[200, images/0001...|Non cream|donut|

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

root

|-- id: string (nullable = true)

|-- image: struct (nullable = true)

| |-- height: long (nullable = true)

| |-- url: string (nullable = true)

| |-- width: long (nullable = true)

|-- name: string (nullable = true)

|-- type: string (nullable = true)

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

| id| type| name|height| url|width|

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

|000|donut|Non cream| 200|images/0001.jpg| 200|

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

root

|-- id: string (nullable = true)

|-- type: string (nullable = true)

|-- name: string (nullable = true)

|-- height: long (nullable = true)

|-- url: string (nullable = true)

|-- width: long (nullable = true)

========

Method-2

========

package pack

import org.apache.spark.SparkContext // rdd

import org.apache.spark.sql.SparkSession // dataframe

import org.apache.spark.SparkConf

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.types.IntegerType

import org.apache.spark.sql.functions.upper

import org.apache.spark.sql.catalyst.expressions.Upper

import org.apache.spark.sql.functions.\_

import org.apache.spark.sql.expressions.Window

object project13 {

def main(args:Array[String]):Unit={

val conf = new SparkConf().setAppName("Revision").setMaster("local[\*]")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession.builder().getOrCreate()

import spark.implicits.\_

val df = spark.read.format("json").option("multiline","true").load("file:///D:/data/pic.json")

df.show()

df.printSchema()

val flatdf = df.withColumn("height",expr("image.height"))

.withColumn("url",expr("image.url"))

.withColumn("width",expr("image.width"))

.drop("image")

flatdf.show()

flatdf.printSchema()

}

}

------------

Out put:

-------

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

| id| image| name| type|

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

|000|[200, images/0001...|Non cream|donut|

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

root

|-- id: string (nullable = true)

|-- image: struct (nullable = true)

| |-- height: long (nullable = true)

| |-- url: string (nullable = true)

| |-- width: long (nullable = true)

|-- name: string (nullable = true)

|-- type: string (nullable = true)

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

| id| name| type|height| url|width|

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

|000|Non cream|donut| 200|images/0001.jpg| 200|

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

root

|-- id: string (nullable = true)

|-- name: string (nullable = true)

|-- type: string (nullable = true)

|-- height: long (nullable = true)

|-- url: string (nullable = true)

|-- width: long (nullable = true)

=======================================================

Task 2 -- Uber UseCase

Find the Maximum number of Trips along with Day for each dispatching base number

=================================================================================

package pack

import org.apache.spark.SparkContext // rdd

import org.apache.spark.sql.SparkSession // dataframe

import org.apache.spark.SparkConf

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.types.IntegerType

import org.apache.spark.sql.functions.upper

import org.apache.spark.sql.catalyst.expressions.Upper

import org.apache.spark.sql.functions.\_

import org.apache.spark.sql.expressions.Window

object project13 {

def main(args:Array[String]):Unit={

val conf = new SparkConf().setAppName("Revision").setMaster("local[\*]")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession.builder().getOrCreate()

import spark.implicits.\_

val df = spark

.read

.format("csv")

.option("header","true")

.load("file:///D:/data/uber.csv")

val finaldf = df.withColumn("Day",expr("DATE\_FORMAT(TO\_DATE(DATE,'MM/dd/yyyy'),'EEE')"))

.groupBy("dispatching\_base\_number","Day")

.agg(sum("trips").alias("Sum"))

.withColumn("rnk",rank.over(Window.partitionBy("dispatching\_base\_number").orderBy("Sum")))

.filter("rnk=1")

.drop("rnk")

finaldf.show()

}

}

----------------

Out put:

--------

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

|dispatching\_base\_number|Day| Sum|

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

| B02512|Wed| 10487.0|

| B02598|Thu| 60882.0|

| B02682|Thu| 74939.0|

| B02765|Thu| 21974.0|

| B02617|Thu| 80591.0|

| B02764|Thu|214116.0|

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

======================================================================

Task 3 --- Gold Video

==========================

=======================

24-12-2022

=======================

Handson Task1

-------------

package pack

import org.apache.spark.SparkContext // rdd

import org.apache.spark.sql.SparkSession // dataframe

import org.apache.spark.SparkConf

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.types.IntegerType

import org.apache.spark.sql.functions.upper

import org.apache.spark.sql.catalyst.expressions.Upper

import org.apache.spark.sql.functions.\_

import org.apache.spark.sql.expressions.Window

object project13 {

def main(args:Array[String]):Unit={

val conf = new SparkConf().setAppName("Revision").setMaster("local[\*]")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession.builder().getOrCreate()

import spark.implicits.\_

val df = spark.read.format("json").option("multiline","true").load("file:///D:/data/place.json")

df.show()

df.printSchema()

val flattendf = df.select(

"place",

"user.address.\*",

"user.name"

)

flattendf.show()

flattendf.printSchema()

}

}

-------------------------------

Out put:

--------

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

| place| user|

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

|Hyderabad|[[40, 400209, ash...|

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

root

|-- place: string (nullable = true)

|-- user: struct (nullable = true)

| |-- address: struct (nullable = true)

| | |-- number: string (nullable = true)

| | |-- pin: string (nullable = true)

| | |-- street: string (nullable = true)

| |-- name: string (nullable = true)

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

| place|number| pin| street|name|

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

|Hyderabad| 40|400209|ashok nagar|zeyo|

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

root

|-- place: string (nullable = true)

|-- number: string (nullable = true)

|-- pin: string (nullable = true)

|-- street: string (nullable = true)

|-- name: string (nullable = true)

=============================

2nd Handson Task Complex data

package pack

import org.apache.spark.SparkContext // rdd

import org.apache.spark.sql.SparkSession // dataframe

import org.apache.spark.SparkConf

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.types.IntegerType

import org.apache.spark.sql.functions.upper

import org.apache.spark.sql.catalyst.expressions.Upper

import org.apache.spark.sql.functions.\_

import org.apache.spark.sql.expressions.Window

object project13 {

def main(args:Array[String]):Unit={

val conf = new SparkConf().setAppName("Revision").setMaster("local[\*]")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession.builder().getOrCreate()

import spark.implicits.\_

val df = spark.read.format("json").option("multiline","true").load("file:///D:/data/address1.json")

df.show()

df.printSchema()

val flatten2df = df.select(

"age",

"date\_of\_birth",

"email\_address",

"first\_name",

"height\_cm",

"is\_alive",

"last\_name",

"address.billing\_address.\*"

)

flatten2df.show()

flatten2df.printSchema()

}

}

--------------

Out put:

--------

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

| address|age|date\_of\_birth| email\_address|first\_name|height\_cm|is\_alive|last\_name|

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

|[[502, Main Marke...| 30| null|rajeev@ezeelive.com| Rajeev| 185.2| true| Sharma|

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

root

|-- address: struct (nullable = true)

| |-- billing\_address: struct (nullable = true)

| | |-- address: string (nullable = true)

| | |-- city: string (nullable = true)

| | |-- postal\_code: string (nullable = true)

| | |-- state: string (nullable = true)

|-- age: long (nullable = true)

|-- date\_of\_birth: string (nullable = true)

|-- email\_address: string (nullable = true)

|-- first\_name: string (nullable = true)

|-- height\_cm: double (nullable = true)

|-- is\_alive: boolean (nullable = true)

|-- last\_name: string (nullable = true)

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

|age|date\_of\_birth| email\_address|first\_name|height\_cm|is\_alive|last\_name| address| city|postal\_code| state|

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

| 30| null|rajeev@ezeelive.com| Rajeev| 185.2| true| Sharma|502, Main Market,...|Vasai Raod, Palghar| 401208|Maharashtra|

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

root

|-- age: long (nullable = true)

|-- date\_of\_birth: string (nullable = true)

|-- email\_address: string (nullable = true)

|-- first\_name: string (nullable = true)

|-- height\_cm: double (nullable = true)

|-- is\_alive: boolean (nullable = true)

|-- last\_name: string (nullable = true)

|-- address: string (nullable = true)

|-- city: string (nullable = true)

|-- postal\_code: string (nullable = true)

|-- state: string (nullable = true)

=====================================================

Address1 json process and Generation:

------------------------------------

package pack

import org.apache.spark.SparkContext // rdd

import org.apache.spark.sql.SparkSession // dataframe

import org.apache.spark.SparkConf

import org.apache.spark.sql.\_

import org.apache.spark.sql.types.\_

import org.apache.spark.sql.types.IntegerType

import org.apache.spark.sql.functions.upper

import org.apache.spark.sql.catalyst.expressions.Upper

import org.apache.spark.sql.functions.\_

import org.apache.spark.sql.expressions.Window

object project13 {

def main(args:Array[String]):Unit={

val conf = new SparkConf().setAppName("Revision").setMaster("local[\*]")

val sc = new SparkContext(conf)

sc.setLogLevel("ERROR")

val spark = SparkSession.builder().getOrCreate()

import spark.implicits.\_

val df = spark

.read

.format("json")

.option("multiline","true")

.load("file:///D:/data/address1.json")

df.show()

df.printSchema()

val flattendf = df.select(

"address.billing\_address.\*",

"age",

"date\_of\_birth",

"email\_address",

"first\_name",

"height\_cm",

"is\_alive",

"last\_name"

)

flattendf.show()

flattendf.printSchema()

val complexdf = flattendf.select(

struct(

struct(

col("address"),

col("city"),

col("postal\_code"),

col("state")

).as("billing\_address")

).as("address"),

col("age"),

col("date\_of\_birth"),

col("email\_address"),

col("first\_name"),

col("height\_cm"),

col("is\_alive"),

col("last\_name")

)

complexdf.show()

complexdf.printSchema()

}

}

--------------------

Out put:

--------

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

| address|age|date\_of\_birth| email\_address|first\_name|height\_cm|is\_alive|last\_name|

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

|[[502, Main Marke...| 30| null|rajeev@ezeelive.com| Rajeev| 185.2| true| Sharma|

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

root

|-- address: struct (nullable = true)

| |-- billing\_address: struct (nullable = true)

| | |-- address: string (nullable = true)

| | |-- city: string (nullable = true)

| | |-- postal\_code: string (nullable = true)

| | |-- state: string (nullable = true)

|-- age: long (nullable = true)

|-- date\_of\_birth: string (nullable = true)

|-- email\_address: string (nullable = true)

|-- first\_name: string (nullable = true)

|-- height\_cm: double (nullable = true)

|-- is\_alive: boolean (nullable = true)

|-- last\_name: string (nullable = true)

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

| address| city|postal\_code| state|age|date\_of\_birth| email\_address|first\_name|height\_cm|is\_alive|last\_name|

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

|502, Main Market,...|Vasai Raod, Palghar| 401208|Maharashtra| 30| null|rajeev@ezeelive.com| Rajeev| 185.2| true| Sharma|

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

root

|-- address: string (nullable = true)

|-- city: string (nullable = true)

|-- postal\_code: string (nullable = true)

|-- state: string (nullable = true)

|-- age: long (nullable = true)

|-- date\_of\_birth: string (nullable = true)

|-- email\_address: string (nullable = true)

|-- first\_name: string (nullable = true)

|-- height\_cm: double (nullable = true)

|-- is\_alive: boolean (nullable = true)

|-- last\_name: string (nullable = true)

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

| address|age|date\_of\_birth| email\_address|first\_name|height\_cm|is\_alive|last\_name|

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

|[[502, Main Marke...| 30| null|rajeev@ezeelive.com| Rajeev| 185.2| true| Sharma|

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

root

|-- address: struct (nullable = false)

| |-- billing\_address: struct (nullable = false)

| | |-- address: string (nullable = true)

| | |-- city: string (nullable = true)

| | |-- postal\_code: string (nullable = true)

| | |-- state: string (nullable = true)

|-- age: long (nullable = true)

|-- date\_of\_birth: string (nullable = true)

|-- email\_address: string (nullable = true)

|-- first\_name: string (nullable = true)

|-- height\_cm: double (nullable = true)

|-- is\_alive: boolean (nullable = true)

|-- last\_name: string (nullable = true)