JSON File Extraction using Pig - (Yelp_academic_dataset.json)

Review Dataset:

```
reviews = load '/user/cali/priyanka/yelp_review/yelp_academic_dataset_review.json' using JsonLoader
('votes:(funny:int,useful:int,cool:int),user_id:chararray,review_id:chararray,stars:int,date:chararray,text:chararray,
type:chararray,business_id:chararray');

review_final = FOREACH reviews generate FLATTEN(votes), user_id, review_id, stars,
REPLACE(REPLACE(text, '\n', ''), '\t', ''), date, type, business_id;

STORE review_final INTO '/user/cali/priyanka/yelp_review_new' USING
org.apache.pig.piggybank.storage.CSVExcelStorage('#', 'YES_MULTILINE', 'WINDOWS',
'WRITE_OUTPUT_HEADER');
```

Tips Dataset:

```
tips = LOAD '/user/cali/priyanka/yelp_tip/yelp_academic_dataset_tip.json' using JsonLoader
('user_id:chararray,text:chararray,business_id:chararray,likes:int,date:chararray,type:chararray');

tips_final = FOREACH tips GENERATE user_id, REPLACE(REPLACE(text, '\n', ''), '\t',
''),business_id,likes,date,type;

STORE tips_final INTO '/user/cali/priyanka/yelp_new_tips'
USING org.apache.pig.piggybank.storage.CSVExcelStorage('#', 'YES_MULTILINE', 'WINDOWS',
'WRITE_OUTPUT_HEADER');
```

JSON File Extraction and Cleaning using HQL

Business dataset:

```
CREATE EXTERNAL TABLE IF NOT EXISTS yelp_business(
 json_response STRING)
```

```
STORED AS TEXTFILE

LOCATION "/user/cali/priyanka/yelp";


CREATE TABLE IF NOT EXISTS business_table_new (

business_id STRING,full_address STRING,open STRING,categories STRING,city STRING,review_cnt
STRING,name STRING,longitude STRING,state STRING,

stars STRING,latitude STRING,live_music STRING,background_music STRING,karaoke STRING,video
STRING,dj STRING,jukebox STRING,park_lot STRING,

street STRING,garage STRING,valet STRING,validated STRING,Happy_Hour STRING,Price_Range
STRING,Wi_Fi STRING,Ages_Allowed STRING,Open_24_Hours STRING,vegan STRING,dairy_free
STRING,halal STRING,gluten_free STRING,kosher STRING,soy_free STRING,vegetarian STRING);


FROM yelp_business

INSERT OVERWRITE TABLE business_table_new

SELECT get_json_object(json_response,
'$.business_id'),REGEXP_REPLACE(REGEXP_REPLACE(get_json_object(json_response, '$.full_address'),'\n',' '),'\t',' '),get_json_object(json_response, '$.open'),

get_json_object(json_response, '$.categories'),get_json_object(json_response,
'$.city'),get_json_object(json_response, '$.review_count'),

REGEXP_REPLACE(get_json_object(json_response, '$.name'),'\t',' '),get_json_object(json_response,
'$.longitude'),get_json_object(json_response, '$.state'),get_json_object(json_response,
'$.stars'),get_json_object(json_response, '$.latitude'),get_json_object(json_response, '$.attributes.Music.live'),

get_json_object(json_response, '$.attributes.Music.background_music'),get_json_object(json_response,
'$.attributes.Music.karaoke'),

get_json_object(json_response, '$.attributes.Music.video'),get_json_object(json_response,
'$.attributes.Music.dj'),get_json_object(json_response, '$.attributes.Music.jukebox'),get_json_object(json_response,
'$.attributes.Parking.lot'),get_json_object(json_response, '$.attributes.Parking.street'),

get_json_object(json_response, '$.attributes.Parking.garage'),get_json_object(json_response,
'$.attributes.Parking.valet'),get_json_object(json_response,
'$.attributes.Parking.validated'),get_json_object(json_response, '$.attributes.Happy
Hour'),get_json_object(json_response, '$.attributes.Price Range'),

get_json_object(json_response, '$.attributes.Wi-Fi'),get_json_object(json_response, '$.attributes.Ages
Allowed'),get_json_object(json_response, '$.attributes.Open 24 Hours'),get_json_object(json_response,
'$.attributes.Dietary Restrictions.vegan'),get_json_object(json_response, '$.attributes.Dietary Restrictions.dairy-
free'),

get_json_object(json_response, '$.attributes.Dietary Restrictions.halal'),get_json_object(json_response,
'$.attributes.Dietary Restrictions.gluten-free'),

get_json_object(json_response, '$.attributes.Dietary Restrictions.kosher'),get_json_object(json_response,
'$.attributes.Dietary Restrictions.soy-free'),

get_json_object(json_response, '$.attributes.Dietary Restrictions.vegetarian');
```

```
--DATA CLEANING - DONE FOR ONLY (LAS VEGAS,PITTSBURGH,CHARLOTTE,URBANA,PHOENIX,MADISON) CITIES

CREATE TABLE IF NOT EXISTS final_yelp_business AS

SELECT * FROM business_table_new WHERE

(city LIKE concat('%','Las Vegas','%') OR

city LIKE concat('Pittsburg','%') OR

city LIKE concat('%','Charlot','%') OR

city LIKE concat('Phoenix','%') OR

city IN ('Madison','Urbana') ) AND

state IN ('NV','PA','NC','IL','AZ','WI');


--BUSINESS TABLE STORED WITH ONLY RESTAURANTS/Foodchain - REMAINING CATEGORIES REMOVED

CREATE TABLE IF NOT EXISTS business_final_new

ROW FORMAT DELIMITED

FIELDS TERMINATED BY "\t"

STORED AS TEXTFILE

LOCATION "/user/cali/priyanka/yelp_new_business"

AS

SELECT * FROM final_yelp_business WHERE categories LIKE concat('%','Food','%')

OR categories LIKE concat('%','Restaurant','%') ;

---------- --
```

User Dataset:

```
--DROP hive tables if existing

DROP TABLE yelp_user;

DROP TABLE users;

DROP table user_final;


CREATE EXTERNAL TABLE IF NOT EXISTS yelp_user(

 json_response STRING)
```

```
STORED AS TEXTFILE

LOCATION "/user/cali/priyanka/yelp_user";


CREATE TABLE IF NOT EXISTS users (

yelping_yr INT,yelping_mon INT,funny INT,cool INT,useful INT,review_count INT,name STRING,user_id STRING,friends STRING,

fans INT,average_stars DECIMAL,type STRING,profile INT,cute INT,cfunny INT,plain INT,writer INT,photos INT,hot INT,ccool INT,

zmore INT,elite STRING);


--Extraction of User.json file and data cleaning

FROM yelp_user

INSERT OVERWRITE TABLE users

SELECT SUBSTR(get_json_object(json_response, '$.yelping_since'),1,4),

SUBSTR(get_json_object(json_response, '$.yelping_since'),6,8),

get_json_object(json_response, '$.votes.funny'),

get_json_object(json_response, '$.votes.cool'),

get_json_object(json_response, '$.votes.useful'),

get_json_object(json_response, '$.review_count'),

get_json_object(json_response, '$.name'),

get_json_object(json_response, '$.user_id'),

get_json_object(json_response, '$.friends'),

get_json_object(json_response, '$.fans'),

get_json_object(json_response, '$.average_stars'),

get_json_object(json_response, '$.type'),

get_json_object(json_response, '$.compliments.profile'),

get_json_object(json_response, '$.compliments.cute'),

get_json_object(json_response, '$.compliments.funny'),

get_json_object(json_response, '$.compliments.plain'),

get_json_object(json_response, '$.compliments.writer'),

get_json_object(json_response, '$.compliments.photos'),

get_json_object(json_response, '$.compliments.hot'),
```

```
get_json_object(json_response, '$.compliments.cool'),
get_json_object(json_response, '$.compliments.more'),
get_json_object(json_response, '$.elite');

--User table uploaded
CREATE TABLE IF NOT EXISTS user_final
ROW FORMAT DELIMITED
FIELDS TERMINATED BY "#"
STORED AS TEXTFILE
LOCATION "/user/cali/priyanka/yelp_new_user"
tblproperties ("skip.header.line.count"="0")
AS
SELECT * FROM users;
```

Review Dataset Extraction:

```
CREATE TABLE IF NOT EXISTS review
( funny INT,
cool INT,
useful INT,
user_id STRING,
review_id STRING,
stars INT,
dates STRING,
text STRING,
type STRING,
business_id STRING);

FROM yelp_review
INSERT OVERWRITE TABLE review
SELECT get_json_object(json_response, '$.votes.funny'),
get_json_object(json_response, '$.votes.cool'),
```

```
get_json_object(json_response, '$.votes.useful'),
get_json_object(json_response, '$.user_id'),
get_json_object(json_response, '$.review_id'),
get_json_object(json_response, '$.stars'),
get_json_object(json_response, '$.date'),
REGEXP_REPLACE(REGEXP_REPLACE(get_json_object(json_response, '$.text'),'\n',' '),'\t',' '),
get_json_object(json_response, '$.type'),
get_json_object(json_response, '$.business_id');

CREATE TABLE IF NOT EXISTS review_final
ROW FORMAT DELIMITED
FIELDS TERMINATED BY "#"
STORED AS TEXTFILE
LOCATION "/user/cali/priyanka/yelp_review_hive"
tblproperties ("skip.header.line.count"=" ")
AS
SELECT * FROM review;
```

ANALYSIS QUERIES ON YELP DATASET

Query 1 - In which year Yelp has got maximum number of users?

```
select yelping_since, count(user_id) as c1 from user_final group by yelping_since order by yelping_since;
```

Query 2 - Forecast of users joining Yelp in coming years.

```
select r.user_id, u.name, count(r.review_id) as r1 from review_final r join user_final u on r.user_id=u.user_id
group by user_id,name order by r1 desc limit 100;
```

Query 3 - Who are the most active users on Yelp ?

```
SELECT u.name, r.review_id, r.total from review_final r join user_final u
on r.user_id=u.userid order by r.total desc limit 100;
```

Query 4 - Who are the most active users on Yelp ?

```
select city, count(open) as c1 from business_final_new where open="FALSE" group by city order by c1 desc limit
20;
```

Query 5 - Sentiment Analysis on User Review for the top 8 Food Chains(based on US Survey Food Survey)

```
--Analysis done only for top 8 food chains in US
CREATE TABLE IF NOT EXISTS business_top AS
SELECT * FROM business_final_new WHERE name LIKE CONCAT('%','starbucks','%')
OR name LIKE CONCAT('%','Starbucks','%')
OR name IN ('Burger King','Burger King ','Wendys','Wendy''s','Taco Bell',
'Pizza Hut','KFC','McDonald''s','McDonalds','McDonald''s - MGM Grand The District Food Court','Subway');

--Review table and business table joined to obtain review text for sentiment analysis
CREATE TABLE IF NOT EXISTS business_review
AS SELECT b.business_id,b.city,b.review_cnt,b.name,b.longitude,b.state,b.latitude,
r.user_id,r.review_id,r.dates,r.text
FROM business_top b INNER JOIN review_new r
ON b.business_id = r.business_id;

--Create views to separate the words and sentences in the review text
create view IF NOT EXISTS br1
AS SELECT business_id,text1
FROM business_review
lateral view explode(sentences(lower(text))) dummy as text1;

--Create view to seperate the group of words
```

```
create view IF NOT EXISTS br2
AS SELECT business_id,text
FROM br1
lateral view explode( text1 ) dummy as text;

--Compare the reviews words with dictionary words and store the polarity
create view IF NOT EXISTS br3
AS SELECT business_id,br2.text,
case d.polarity
 when 'negative' then -1
 when 'positive' then 1
 else 0 end as polarity
from br2 left outer join dictionary d on br2.text = d.word;

--The Optimized Row Columnar (ORC) file format provides a highly efficient way to store Hive data. It was designed to overcome limitations of the other Hive file formats. Using --ORC files improves performance when Hive is reading, writing, and processing data
--Summation of polarity to understand the whole review
create table IF NOT EXISTS review_sentiment
stored as orc
AS SELECT business_id,
 case
 when sum( polarity ) > 0 then 'positive'
 when sum( polarity ) < 0 then 'negative'
 else 'neutral' end as sentiment
from br3 group by business_id;

--Output the join table based on business id for the top 8 Food chains
CREATE TABLE IF NOT EXISTS yelp_sentiment
ROW FORMAT DELIMITED
FIELDS TERMINATED BY "#"
STORED AS TEXTFILE
```

```
LOCATION "/user/cali/priyanka/yelp_sentiment"
AS SELECT
 br.business_id,br.city,br.name,br.longitude,br.state,br.latitude,
 br.user_id,br.review_id,
 case rs.sentiment
 when 'positive' then 2
 when 'neutral' then 1
 when 'negative' then 0
 end as sentiment
FROM business_review br LEFT OUTER JOIN review_sentiment rs
 on br.business_id = rs.business_id;
```

Query 6 - Analysis to help tourist to choose restaurants in the city they are based on Price Range,Paking,Wi-Fi connection and Food

type and the maximum reviews on each restaurants

```
CREATE TABLE IF NOT EXISTS tourist
ROW FORMAT DELIMITED
FIELDS TERMINATED BY "\t"
STORED AS TEXTFILE
LOCATION "/user/cali/priyanka/yelp_tourist_analysis"
AS SELECT name,open,categories,city,review_cnt,state,park_lot,
street,garage,valet,validated,Happy_Hour,Price_Range,Wi_Fi
FROM business_final_new WHERE park_lot IS NOT NULL AND
street IS NOT NULL AND
garage IS NOT NULL AND
valet IS NOT NULL AND
validated IS NOT NULL AND
Happy_Hour IS NOT NULL AND
Price_Range IS NOT NULL AND
Wi_Fi IS NOT NULL;
```
