INTRODUCTION TO BIG DATA

Phase III

Group 5

Data Cleaning:

Run test.py

1. **Validity:** In the first step of data cleaning, we are checking in each table if the primary keys are being repeated or not. Primary keys across all the tables in our dataset are user_id, business_id,review_id, category_id. Our dataset data is represented in such a way that there can't be primary key duplicates in a table.
2. **Completeness:** We are checking if our dataset contains Null values in any table. This checks the completeness of the dataset.
3. **Accuracy:** Accuracy of the dataset is cross validated by checking the values of columns which have numerical values and checking if the value for each tuple is in the proper range as it should be. For example, the stars column values can range from 0 to max 5, it cannot go below 0 or above 5.
4. **Timeliness**: Timeliness is done by checking the date column of the dataset which shows that even the recent months data is contained in the dataset.

Data Integration:

The following views were created:

1. CREATE MATERIALIZED VIEW business_tips as select distinct b.name, t.tip from tip as t join review as r on t.business_id = r.business_id join business as b on b.business_id = t.business_id join users as u on t.user_id = u.user_id and t.user_id = r.user_id where t.complement_count > 2 and r.stars > 2 and u.fans > 50

2. CREATE MATERIALIZED VIEW number_of_restaurants as select ad.postal_code,ad.city, count(bc.business_id) from category as c join business_category as bc on c.id = bc.category_id join address as ad on bc.business_id = ad.business_id where c.category_type LIKE 'Restaurants' group by ad.postal_code,ad.city, c.id order by count(bc.business_id) DESC
3. CREATE MATERIALIZED VIEW Business_name_category AS SELECT bc.business_id,name as Business_name,Category_type as Category FROM business as b JOIN business_category as bc ON bc.business_id = b.business_id JOIN category as c ON c.id = bc.category_id;
4. CREATE MATERIALIZED VIEW TopUsersTips AS SELECT b.name as business name,u.name as username,tip FROM users as u JOIN tip as t ON u.user_id = t.user_id JOIN business as b ON b.business_id = t.business_id WHERE review_count>50 and fans=100;

View number 1 contains a business name and a tip for that business name.These printed tips are by users with compliment count greater than 2 and number of fans greater than 50.Also the ratings for a business is greater than 2.

View number 2 contains a postal code, the city name for that postal code and the number of business in that city with the category = 'Restaurants'

View number 3 contains all the business id's, their names and their category names.

View number 4 contains a business name, a user name and a tip by the user for that business.

ITEMSET MINING

itemsetMining.py

Using itemset mining on the yelp dataset we try to find out the number of users who have commonly given a tip to a business.For this the steps are as:

1. Build the best_tips table from the tip table by pulling out user_id, business_id where the compliment count was greater than 0.
2. From the best_tips we built the L1 lattice table by taking frequent itemsets of size 1 from best_tips with a minimum support that the users in the itemsets have at least greater than 1 business in common.
3. Using this minimum support we find the lattice tables upto the nth level until the lattice level at the nth level is empty.

4. In the last step we take the n-1 th lattice which will be non empty and join it with users table in order to display user names.

```
totale execution time 0.6700742244720459
users : Bella, Grace, MarVy, Sharr, Christalle, visited frequency :2
users : Mike, Grace, MarVy, Sharr, Christalle, visited frequency :2
users : Vi, Grace, MarVy, Sharr, Christalle, visited frequency :2
users : Mimi, Grace, MarVy, Sharr, Christalle, visited frequency :2
users : Terri, Christie, Joanna, Kevin, Amanda, visited frequency :2
users : Jordan, Grace, MarVy, Sharr, Christalle, visited frequency :2
users : Joe, Grace, MarVy, Sharr, Christalle, visited frequency :2
users : Jennifer, Grace, MarVy, Sharr, Christalle, visited frequency :2
```

RDBMS OR Document Oriented for our Model: Relational Databases works better for our dataset than the document-oriented databases as writing complex lattice queries in a Relational Database is much easier when compared to MongoDB. It makes sense as the lattice size keeps increasing, the queries get only bigger which is why RDBMS is better for our dataset. Also, writing code which generates queries dynamically can be done in RDBMS without any hassle. Document oriented would work better if the dataset is too large which is not the case with our dataset. Moreover, it is difficult to perform the cleaning process in MongoDB searching through each document for null values and since there exists no foreign key constraint in document oriented, it gets difficult to maintain consistent data which is required for itemset mining and data cleaning.