* Explain relational algebra operatn's using mapreduce.

1. Selectn
2. Projectn
3. Intersectn & Union
4. Natural Join
5. Grouping & Aggregatn

Selectn -

- Apply a cond. c to each tuple in the relatn & produce ~~an O/P~~ as O/P only those tuples that satisfy c.
- Result denoted by ~~$\sigma_c(R)$ $\sigma_c(R)$~~ $\sigma_c(R)$ ~~[illegible]~~
- Can be done most conveniently in map portn alone, although they could be done in reduce portn also.

Write pseudocode & give eg.

pseudocode :

```
Map(key, value):
    for tuple in value :
        if tuple satisfies c :
            emit(tuple, tuple)
Reduce (key, values):
    emit (key, key)
```

Projectn -

For some subset s of attribute of relatn, produce from each tuple only the components for the attributes in S.

Result denoted as $\Pi_s(R)$

Performed similarly to selectn

May cause same tuple to apper several times reduce fn eliminates duplicates.

pseudocode:

```
Map(key, value):
    for tuple in value:
        ts = tuple with only the comps for att.in S.
        emit(ts, ts)
Reduce (key, values):
    emit (key, key)
```

Union -

- Both selectⁿ & projectⁿ are operatⁿs that are applied on single table, whereas union, intersectⁿ & diff. are among operatⁿs that are applied on 2 or more tables.
- Map fⁿ - For each row r generate key-value pair (r, r).
- Reduce fⁿ - With each key there can be 1 or 2 values (As we dont have duplicate rows), in either case just O/P 1st value.
- Has Map fⁿ of selectⁿ & Reduce fⁿ of projectⁿ

Intersectⁿ-

- Map fⁿ - For each row r gen key-val pair (r, r).
- Reduce fⁿ - With each key there can be 1/2 values, in case we have length of list as 2 we O/P 1st val. else we O/P nothing.

Diff -

Map fⁿ - For each row r create key-value pair (r, T1) if row is from table 1 else produce key-value pair (r, T2).

Reduce - O/P row iff only if val. in list is T1 otherwise O/P nothing.

5	6
6	3

Output of difference of the tables

For the difference operation we notice that we cannot get rid of the reduce part and hence have to send data across the workers as the context of from which table the value came is needed. Hence it will be **more expensive** operation as compared to selection, projection, union and intersection.

4. **Natural join :** Please refer Section 2.8..

5. **Grouping and Aggregation Using Map Reduce**

Usually understanding grouping and aggregation takes a bit of time when we learn SQL, but not in case when we understand these operations using map reduce. The logic is already there in the working of the map. Map workers implicitly group keys and the reduce function acts upon the aggregated values to generate output.

- **Map Function :** For each row in the table, take the attributes using which grouping is to be done as the key, and value will be the ones on which aggregation is to be performed. For example, If a relation has 4 columns A, B, C, D and we want to group by A, B and do an aggregation on C we will make (A, B) as the key and C as the value.
- **Reduce Function :** Apply the aggregation operation (sum, max, min, avg, ...) on the list of values and output the result.

Initial data

The data after application of map f
and C as value and D is discarded as

Reduce w
Key
(1, 2
(2.

Applying partitio
Files for the

get rid of the reduce stage. The context of aggregation function makes it necessary for the values to be in one place for a single key. This operation is also inefficient as compared to selection, projection, union, and intersection. The column that is not in aggregation or grouping clause is ignored and isn't required. So, if the data be stored in a columnar format, we can save cost of loading a lot of data. Usually there are only a few columns involved in grouping and aggregation it does save up a lot of cost both in terms of data that is sent over the network and the data that needs to be loaded to main memory for execution.

2.8 NATURAL JOIN USING MAP REDUCE

The natural join will keep the rows that matches the values in the common column for both tables. To perform natural join, we will have to keep track of from which table the value came from. If the values for the same key are from different tables we need to form pairs of those values along with key to get a single row of the output. Join can explode the number of rows as we have to form each and every possible combination of the values for both tables.

- **Map Function :** For two relations Table 1(A, B) and Table 2(B, C) the map function will create key-value pairs of form b: [(T1, a)] for table 1 where T1 represents the fact that the value a came from table 1, for table 2 key-value pairs will be of the form b: [(T2, c)].
- **Reduce Function :** For a given key b construct all possible combinations for the values where one value is from table T1 and the other value is from table T2. The output will consist of key-value pairs of form b: [(a, c)] which represent one row a, b, c for the output table.