NAME:- Bryce Ferreira
ENROLLMENT NO:- MITU20BTCS0076
ROLL NO.:- 2203129
Class:- TY CSE CORE:- 3

Practical 8

Part A

Practical Objective:
Implement MapReduce example in MongoDB with suitable dataset

Prerequisite: Understanding the usage of mongodb shell

Software: Mongodb

CO Mapping:
CO3: To get hands on exposure on NOSQL(Mongo) DB.

Practical Outcomes: At the end of this practical student will be able to:
Do mapreduce

Theory:

a) db.orders.insertMany([ ])

b)
```
var mapFunction1 = function() {
emit(this.cust_id, this.price);
};
```

c)
```
var reduceFunction1 = function(keyCustId, valuesPrices) {
return Array.sum(valuesPrices);
};
```

d)
```
db.orders.mapReduce(
    mapFunction1,
    reduceFunction1,
    { out: "map_reduce_example" }
)
```

e) db.map_reduce_example.find().sort({ _id: 1 })

Procedure:

1. Formulate the function for given problem.
2. Write the NOSQL query with proper input.
3. Execute the query.

Practice Exercise:

Create a sample collection orders with these documents:

```
{ _id: 1, cust_id: "Ant O. Knee", ord_date: new Date("2020-03-01"), price: 25, items: [ { sku: "oranges",
{ _id: 2, cust_id: "Ant O. Knee", ord_date: new Date("2020-03-08"), price: 70, items: [ { sku: "oranges",
{ _id: 3, cust_id: "Busby Bee", ord_date: new Date("2020-03-08"), price: 50, items: [ { sku: "oranges", qt
```

```
{ _id: 4, cust_id: "Busby Bee", ord_date: new Date("2020-03-18"), price: 25, items: [ { sku: "oranges", qty: 10, price: 2.5 } ],
{ _id: 5, cust_id: "Busby Bee", ord_date: new Date("2020-03-19"), price: 50, items: [ { sku: "chocolates", qty: 5, price: 10 } ]
{ _id: 6, cust_id: "Cam Elot", ord_date: new Date("2020-03-19"), price: 35, items: [ { sku: "carrots", qty: 10, price: 1.0 }, { s
{ _id: 7, cust_id: "Cam Elot", ord_date: new Date("2020-03-20"), price: 25, items: [ { sku: "oranges", qty: 10, price: 2.5 } ],
{ _id: 8, cust_id: "Don Quis", ord_date: new Date("2020-03-20"), price: 75, items: [ { sku: "chocolates", qty: 5, price: 10 }, {
{ _id: 9, cust_id: "Don Quis", ord_date: new Date("2020-03-20"), price: 55, items: [ { sku: "carrots", qty: 5, price: 1.0 }, { sk
{ _id: 10, cust_id: "Don Quis", ord_date: new Date("2020-03-23"), price: 25, items: [ { sku: "oranges", qty: 10, price: 2.5 } ],
])
```

2. Define the map function to process each input document:

In the function, this refers to the document that the map-reduce operation is processing.

The function maps the price to the cust_id for each document and emits the cust_id and price.

```
user> var mapFunction1 = function() {
... emit(this.cust_id, this.price);
... };
```

3. Define the corresponding reduce function with two arguments keyCustId and values Prices:

   a. The values Prices is an array whose elements are the price values emitted by the map function and grouped by keyCustId.
   b. The function reduces the values Price array to the sum of its elements.

4. Perform map-reduce on all documents in the orders collection using the mapFunction1 map function and the reduceFunction1 reduce function:

```
user> var reduceFunction1 = function(keyCustId, valuesPrices) {
... return Array.sum(valuesPrices);
... };
```

5. This operation outputs the results to a collection named map_reduce_example. If the map_reduce_example collection already exists, the operation will replace the contents with the results of this map-reduce operation.

```
user> db.orders.mapReduce(
...     mapFunction1,
...     reduceFunction1,
...     { out: "map_reduce_example" }
... )
DeprecationWarning: Collection.mapReduce() is deprecated. Use an aggregation instead.
See https://docs.mongodb.com/manual/core/map-reduce for details.
{ result: 'map_reduce_example', ok: 1 }
```

6. Query the map_reduce_example collection to verify the results:

```
user> db.map_reduce_example.find().sort( { _id: 1 } )
[
  { _id: 'Ant O. Knee', value: 95 },
  { _id: 'Busby Bee', value: 125 },
  { _id: 'Cam Elot', value: 60 },
  { _id: 'Don Quis', value: 155 }
]
```

Instructions:

1. Write and execute the in MONGODB.
2. Paste the snapshot of the output in input & output section.

Part B

Code and Output:
Perform the operation and paste the running code here.

Observation & Learning:
Write your observation and learning after performing the task.

Conclusion:
Write statement of conclusion here.