ISIT312 Sample Exam Paper 1

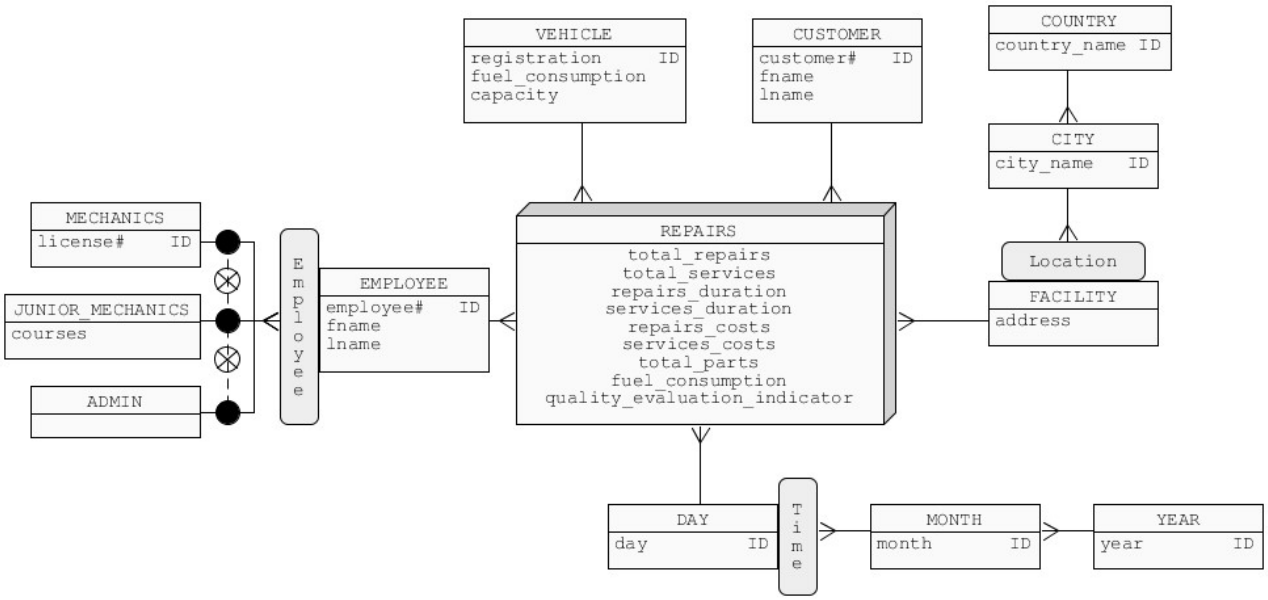
Question 1

To implement the IN application, the following code `conf.set("limit", otherArgs[0]);` is changed into `conf.set("inValue", otherArgs[0]);`. Note that `otherArgs[0]` is a string that consists of numbers separated by comma (`,`), for example, `10,15,20`.

After that, the map function is modified as follows.

```
public void map(Object key, Text value, Context context)
     throws IOException, InterruptedException {

     StringTokenizer itr = new StringTokenizer(value.toString());
     String[] inValueStr = context.getConfiguration()
                              .get("inValue").split(",");

     int[] inValue = new int[inValueStr.length()];
     for (int i = 0; i < inValueStr.length(); i++) {
          inValue[i] = Integer.parseInt(inValueStr[i]);
     }

     while (itr.hasMoreTokens()) {
          word.set(itr.nextToken());
          total = Integer.parseInt(itr.nextToken());

          for (int val: inValue) {
               if (total == inValue) {
                    counter.set(total);
                    context.write(word, counter);
                    break;
               }
          }
     }
}
```

Question 2



Question 3

Part (1)

```
SELECT PARTKEY, QUANTITY,
       MAX(QUANTITY) OVER (PARTITION BY PARTKEY)
FROM ORDERS;
```

Part (2)

```
SELECT PARTKEY,
       AVG(DISCOUNT) OVER (PARTITION BY PARTKEY)
FROM ORDERS
WHERE PARTKEY IN (1001, 1002);
```

Part (3)

```
SELECT PARTKEY, DISCOUNT,
       AVG(DISCOUNT) OVER (PARTITION BY PARTKEY
                           ORDER BY DISCOUNT
                           ROWS 1 PRECEDING)
FROM ORDERS;
```

Part (4)

```
SELECT PARTKEY, DISCOUNT,
       AVG(DISCOUNT) OVER (PARTITION BY PARTKEY
                           ORDER BY DISCOUNT
                           ROWS UNBOUNDED PRECEDING)
FROM ORDERS;
```

Question 4

Part (1)

```
create 'question4', 'ORDER', 'CUSTOMER', 'PRODUCT'
```

Part (2)

```
put 'question4', 'customer|12345678', 'CUSTOMER:phone', '12345678'
put 'question4', 'customer|12345678', 'CUSTOMER:fname', 'Alex'
put 'question4', 'customer|12345678', 'CUSTOMER:lname', 'Smith'

put 'question4', 'product|101', 'PRODUCT:code', '101'
put 'question4', 'product|101', 'PRODUCT:name', 'pencil'

put 'question4', 'product|102', 'PRODUCT:code', '102'
put 'question4', 'product|102', 'PRODUCT:name', 'pen'

put 'question4', 'order|001', 'CUSTOMER:phone', '12345678'
put 'question4', 'order|001', 'PRODUCT:code', '101'
put 'question4', 'order|001', 'ORDER:orderNumber', '001'
put 'question4', 'order|001', 'ORDER:quantity', '10'
put 'question4', 'order|001', 'ORDER:amountPaid', '150'
put 'question4', 'order|001', 'ORDER:discount', '10'

put 'question4', 'order|002', 'CUSTOMER:phone', '12345678'
put 'question4', 'order|002', 'PRODUCT:code', '102'
put 'question4', 'order|002', 'ORDER:orderNumber', '002'
put 'question4', 'order|002', 'ORDER:quantity', '10'
put 'question4', 'order|002', 'ORDER:amountPaid', '200',
put 'question4', 'order|002', 'ORDER:discount', '10'
```

Part (3)

```
get 'question4', 'order|123'

scan 'question4', {COLUMNS=>'PRODUCT:name', FILTER=>"QualifierFilter(=,
'binary:orderNumber') AND ValueFilter(=, 'binary:123')", VERSIONS=>2}

scan 'question4', {VERSIONS=>2}

scan 'question4', {COLUMNS=>'ORDER', FILTER=>'TimestampsFilter(1, 5)'}

scan 'question4', {COLUMNS=>'PRODUCT:code', FILTER=>"QualifierFilter(=,
'binary:phone') AND ValueFilter(=, 'binary:345')"}
```

Question 5

```
-- Load product.txt
product = load 'product.txt' using PigStorage('|') as (code:chararray,
name:chararray);

-- Load order.txt
order = load 'order.txt' using PigStorage('|') as (orderNumber:chararray,
quantity:int,    amountPaid:float,    discount:float,    code:chararray,
phone:chararray);

-- Find total quantities of products grouped by product name
joinOP = join product by code, order by code;
groupProductCode = group joinOP by product::name;
totalQty = foreach groupProductCode generate group,
                SUM(joinOP.order::quantity);
dump totalQty;
```

Question 6

Part (1)

```
case class OrderDetails(orderID: String, productID: String,
                        customerID: String, employeeID: String,
                        unitPrice: Float, quantity: Long, discount: Float)
val orderDetailsDF = spark.sparkContext.textFile("order-details.tbl").
                         map(_.split(",")).map(attributes=>OrderDetails(
                             attributes(0), attributes(1),
                             attributes(2), attributes(3),
                             attributes(4).trim.toFloat,
                             attributes(5).trim.toInt,
                             attributes(6).trim.toFloat)).
                         toDF()
```

Part (2)

```
orderDetailsDF.filter($"discount" < 0.5).show();
```

Part (3)

```
case class Customer(customerID: String, customerCode: String,
                    companyName: String, contactName: String,
                    contactTitle: String, city: String, region: String,
                    postal-code: String, country: String, phone:String,
                    fax:String)
val customerDF = spark.sparkContext.textFile("customer.tbl").
                         map(_.split(",")).map(attributes=>Customer(
                             attributes(0), attributes(1), attributes(2),
                             attributes(3), attributes(4), attributes(5),
                             attributes(6), attributes(7), attributes(8),
                             attributes(9), attributes(10)).
                         toDF()
customerDF.filter($"country" == "Germany").show()
```

Part (4)

```
orderDetailsDF.groupBy("customerID").count().show()
```

Part (5)

```
case class Product(productID: String, productName: String,
                   unitPrice: Float, unitsInStock: Long,
                   unitsOnOrder: Long, discontinuted: Boolean)
val productDF = spark.sparkContext.textFile("product.tbl").
                         map(_.split(",")).map(attributes=>OrderDetails(
                             attributes(0),
                             attributes(1),
                             attributes(2).trim.toFloat,
                             attributes(3).trim.toInt,
                             attributes(4).trim.toInt,
                             attributes(5).trim.toBoolean)).
                         toDF()
productDF.orderBy(col("unitPrice").desc).show(5)
```