1.)

For this query, I struggled a bit with this on and am unsure if this is correct or if it is in the right direction:

Mapper 1a - Key: lo_orderday Value: lo_revenue
Mapper 1b - Key: d_datekey Value: d_year
Reducer receives: lo_revenue and d_year

Mapper 2a - Key: p_partkey Value: p_brand1
Mapper 2b - Key: lo_suppkey Value: ----------
Reducer receives: p_brand1

Mapper 3 - Key: lo_revenue, d_year, p_brand1
Reducer receives the sum of lo_revenue, d_year, p_brand1 all grouped by d_year and p_brand1 and ordered by d_year and p_brand1.

2.)

a.) For 5-node Hadoop cluster –
93 blocks / 5 nodes = 18.6 or 19 iterations
19 iterations (per minute) * 3 (replication) = 57 minutes

b.) For 20 Hadoop worker nodes –
93 blocks / 20 = 4.65 or 5 iterations
5 iterations (per minute) * 1 (replication) = 5 minutes

c.) For 20 Hadoop worker nodes –
93 blocks / 20 = 4.65 or 5 iterations
5 iterations (per minute) * 3 (replication) = 15 minutes

d.) For 100 Hadoop worker nodes –
93 blocks / 100 = .93 or 1 iteration
1 iteration (per minute) * 1 (replication) = 1 minute

e.) For 100 Haddop worker nodes –
93 blocks / 100 = .93 or 1 iteration
1 iteration (per minute) * 3 (replication) = 3 minutes

3.)

a.) Using the hash function (key) and MOD 3 (for 3 reducers):
R1: partition_0 would include keys: 6, 12, 15
R2: partition_1would include keys: 1,4,16,25,52,88
R3: partition2 would include keys: 8,11,17,26,29,50,59,89,95,98

b.) Using the default partitioner creates an obvious imbalance which would impact the MapReduce performance. To address this, a custom sorting partitioner can be created. One possible method could be to take the total number of keys and divide them by 3, then put that number of keys into one of the 3 reducers. In this case, there are 19 keys, which means that they can be divided into two groups of 6 and one group of 7. Another possibility would be to combine the keys with their values and then execute a sort on that composite key.

c.) For a custom partitioner, it must first be balanced. The downside of a custom partitioner is that the reducer run-time can be affected which can kill overall performance.

4.)

For this problem, I ran into issues with the Reducr.py code. The tax values displayed are "off" by a row, different from the tax values that were input from the Mapper.py code:

For example, the following command, executing the Mapper code produces:

```
[ec2-user@ip-172-31-30-225 ~]$ head -n 100 lineorder.tbl | python test_mapper.py
TRUCK   12
FOB     14
SHIP    17
RAIL    12
FOB     18
FOB     16
FOB     10
SHIP    15
FOB     14
REG AIR 12
TRUCK   13
FOB     15
RAIL    17
REG AIR 17
REG AIR 11
SHIP    16
AIR     10
SHIP    10
SHIP    13
FOB     17
TRUCK   16
RAIL    18
```

Testing the Reducer code yields:

```
[ec2-user@ip-172-31-30-225 ~]$ head -n 100 lineorder.tbl | python test_mapper.py | python test_reducer.py
TRUCK   1       14
FOB     1       17
SHIP    1       12
RAIL    1       18
FOB     3       15
SHIP    1       14
FOB     1       12
REG AIR 1       13
TRUCK   1       15
FOB     1       17
RAIL    1       17
REG AIR 2       16
SHIP    1       10
AIR     1       10
SHIP    2       17
FOB     1       16
TRUCK   1       18
RAIL    1       18
```

The second row contains the counts, and they are "off", because the code is only counting them if they are sequential, for example, FOB appears 3 times in row, hence it has at least one value of 3. Also, the tax values (3rd column) are all off by a row. I thought using a dictionary would be the solution, but the professor indicated that a dictionary is NOT the solution, hence, I did not use it. I look forward to receiving some feedback on this problem.

Mapper.py

```
#!/usr/bin/python
```

```
import sys

# lo_quantity = field 8
# lo_tax      = field 14
# ship mode   = field 16

number = []

for line in sys.stdin:
    line = line.strip()
    vals = line.split("|")

    range_1 = range(16,28,1)

    if int(vals[8]) in range_1:

        number.append(vals[14])

        print(vals[16] + '\t1'  + vals[14])
```

Reducer.py

```
#!/usr/bin/python

import sys

curr_id = None
curr_cnt = 0
id = None

idtax = []

for line in sys.stdin:

   line = line.strip()
   id, tax = line.split('\t')
   tax = int(tax)

   if curr_id == id:
       curr_cnt += 1
       idtax = []
   else:
       if curr_id:
           print( '%s\t%d\t%s' % (curr_id, curr_cnt, tax) )

       curr_id = id
       curr_cnt = 1

if curr_id == id:
    print( '%s\t%d\t%s' % (curr_id, curr_cnt, tax) )
```

5.)

a.) QQQQZZZZZAAAANNN = 16 bytes

b.) Q4Z5A4N3 = 8 bytes

c.) Q = 00001 Z = 00010 A = 00100 N = 01000

4 * 5	5 * 5	4 *5	3 * 5
20 its	25 bits	20 bits	15 bits

80 bits or 10 bytes

d.) QQBQQZZUZZAANN = 14 bytes
Q2B1Q2Z2U1Z2A2N2 = 16 bytes

Q = 00001 B = 00010 Z = 00100 A = 01000 N = 10000 U = 00011

(2*5 + 2*5)	1* 5	(2 *5 + 2*5)	(2*5)	(2*5)	(1*5)
20 bits	5 bits	20 bits	10 bits	10 bits	5 bits

70 bits or 8.75 bytes

6.)

```
hbase(main):026:0> scan 'employees'
ROW                             COLUMN+CELL
 ID1                            column=private:address, timestamp=1652156441056, value=243 N. Wabash Av.
 ID1                            column=private:cell, timestamp=1652156540159, value=312-234-5678
 ID1                            column=private:firstname, timestamp=1652156574934, value=John
 ID1                            column=private:lastname, timestamp=1652156599546, value=Smith
 ID1                            column=private:ssn, timestamp=1652156351932, value=111-222-234
 ID2                            column=private:ssn, timestamp=1652156379098, value=222-338-446
 ID3                            column=favoriteColor:car, timestamp=1652157027574, value=Blue
 ID3                            column=favoriteTeam:city, timestamp=1652156948767, value=Milwaukee
 ID3                            column=private:address, timestamp=1652156412481, value=123 State St.
 ID4                            column=favoriteColor:car, timestamp=1652157037420, value=Black
 ID4                            column=favoriteTeam:city, timestamp=1652156915633, value=Chicago
 ID4                            column=public:nickname, timestamp=1652156705046, value=Pooter
4 row(s) in 0.0570 seconds
```