Task 1 – MapReduce Programming 1: Basics

1.1 Count the number of occurrences for each individual drawn lottery number ("lotterynumbers"). Use the class LotteryCount in the package lottery. Copy the output of the file created by MapReduce into your result PDF.

01	82		**23**	90		**45**	102		**67**	10
02	117		**24**	102		**46**	107		**68**	19
03	101		**25**	112		**47**	80		**69**	13
04	100		**26**	84		**48**	110		**70**	13
05	101		**27**	101		**49**	88		**71**	13
06	90		**28**	94		**50**	105		**72**	13
07	97		**29**	104		**51**	111		**73**	13
08	91		**30**	90		**52**	97		**74**	17
09	104		**31**	108		**53**	102		**75**	9
10	105		**32**	87		**54**	81			
11	103		**33**	85		**55**	74			
12	106		**34**	81		**56**	89			
13	99		**35**	113		**57**	14			
14	111		**36**	105		**58**	19			
15	96		**37**	79		**59**	15			
16	94		**38**	102		**60**	12			
17	108		**39**	105		**61**	13			
18	100		**40**	98		**62**	16			
19	90		**41**	89		**63**	15			
20	113		**42**	95		**64**	14			
21	97		**43**	89		**65**	12			
22	96		**44**	95		**66**	15			

1.2 You want to play the lottery based on an evaluation of historical drawings. Therefore, you want to discover the 5 individual numbers (“lotterynumbers”) that have been drawn most often. Limit the output on the exact top 5 occurrences of drawn numbers using the MapReduce paradigm. Use the class LotteryCountTop5 in the package lottery. Copy the table below into your result PDF and fill in the results.

Drawn number	Occurrence
2	117
35	113
20	113
25	112
14	111

Task 2 – MapReduce Programming 2: Joins

2.1 Explain how a generic Reduce-side Join for an arbitrary relationship (1:1, 1:n as well as n:m) works. How is the data processed in each step? Make sure that the output of the Map step is general enough to support all kinds of relationship types in the Reduce step. Copy the table below into your result PDF and insert the explanations to the corresponding step. (5 points)

Step	**Explanation**
Map	emit <join attribute (key)>, <table-id, tuple> so as not to lose the information of where it comes from. The tuple has all other information of that data entry.
Combine	aggregate all values that have the same key
Partition	hash partition on key, meaning the tuples will be distributed to nodes according to their keys
Shuffle	transfers data from the mappers to reducers, splitting it according to key
Sort	sorts the data within each reducer by its key
Reduce	joins the data based on the key and the desired kind of join (inner, outer, left, etc)

2.2 Aggregate the distance and the earnings from the taxi trip data set per "dropoff_date". Use the class TaxiAggregate in the package taxi. Copy the table below into your result PDF and insert the sum of the distance as well as the earnings per "dropoff_date".

dropoff_date	distance	Earnings
06-01-2015	137223.42999999973	1409103.739999991
06-02-2015	131085.86999999988	1366296.2500000065
06-03-2015	139624.7500000004	1378500.4699999925
06-04-2015	152275.7299999987	1472577.3499999992
06-05-2015	180111.99999999965	1741694.440000008
06-06-2015	220171.5000000023	2005130.620000007
06-07-2015	208587.1599999983	1795657.6600000008
06-08-2015	3212.6099999999983	16144.989999999993

2.3 Based on the results of 2.2, implement a reduce-side join with the fuel prices. Join on the “dropoff_date” and “businessday” (fuel prices) columns. Assume that the averaged fuel efficiency is 21.3 miles per gallon (mpg). Calculate the theoretical gross margin per day if all drivers would have tanked up in Buffalo only. Use the class TaxiJoin in the package taxi. Copy the table below into your result PDF and insert the results. Note that not necessarily all cells will be needed. Explain briefly why!

Date	Gross Margin (Buffalo)
06-01-2015	1391258.251685437
06-02-2015	1349495.1032816968
06-03-2015	1359883.8366666592
06-04-2015	1452988.8288638492
06-05-2015	1718017.7451643273
06-06-2015	1976291.254507049
06-07-2015	1768727.3928638508
06-08-2015	0

Not all cells were needed because there was a difference in the period of days considered by the taxi and the fuel tables. The dropoff_date from the taxi table went on until the 08. 06. , while the businessday from the fuel one only considered 7 days, until the 07. 06..
When the tables were joined, it was impossible to calculate the gross margin for a day that didn't have a corresponding fuel price.

Task 3 – Hive

3.1 What are the most 5 frequently drawn individual lottery numbers (see 1.2)? Solve this task using HiveQL. Copy your HiveQL query as well as the result of this query in the result document. Hint: Have a look at the split() function provided by HiveQL.

```
SELECT number, count(1) as count from
(SELECT explode(split(lotterynumbers, ' ')) as number FROM lottery) temptable
GROUP BY number
ORDER BY count desc LIMIT 5
```

	number	count
1	02	117
2	35	113
3	20	113
4	25	112
5	14	111

3.2 Using the taxi trip data set, aggregate the distance and earnings per "dropoff_date". Use the result and join on the "dropoff_date" and "businessday" (fuel prices) columns. Assume that the averaged fuel efficiency is 21.3 miles per gallon (mpg). Calculate the theoretical gross margin per day if all drivers would have tanked up in Buffalo only. See 2.2 & 2.3, but use HiveQL for this task. Copy your HiveQL query as well as the result of this query in the results.

```
SELECT date, earnings - (price * (distance / 21.3)) as gross_margin
from
(SELECT f.businessday as date, f.buffalo_price as price, sum(t.distance) as distance,
sum(t.earnings) as earnings
FROM taxi as t join fuel as f on (t.dropoff_date = f.businessday)
GROUP BY f.businessday, f.buffalo_price) table
```

	date	gross_margin
1	2015-06-01	1391258.2516854343
2	2015-06-02	1349495.103281697
3	2015-06-03	1359883.8366666706
4	2015-06-04	1452988.828863838
5	2015-06-05	1718017.7451643394
6	2015-06-06	1976291.2545070811
7	2015-06-07	1768727.3928638466