CS 644: Homework 4
Instructions. Answer the following multiple choice questions by selecting all correct choices.

1. **Partitions and Partitioning**

   (a) Given a pair RDD (of key-value pairs), when we group values with the same key Spark collects key-value pairs with the same key on the same machine of our cluster.

   ☐ True   ☐ False

   (b) By default, Spark uses range partitioning to determine which key-value pair should be sent to which machine.

   ☐ True   ☐ False

   (c) Suppose we partition an RDD into a number of blocks. From the following statements, select the two that are true.

   ☐ A single block of the partition may be distributed across multiple machines in the cluster.
   ☐ A block of the partition is assigned to at most one machine of the cluster.
   ☐ At least one block of the partition is assigned to every machine in the cluster.
   ☐ At most one block of the partition is assigned to every machine in the cluster.
   ☐ More than one block of the partition may be assigned to the same machine in the cluster.

2. Consider a Pair RDD, with keys [8, 23, 39, 40, 97], and suppose we want to partition these data into 4 blocks.

   (a) Using hash partitioning with the identity as `hashCode()` function (`n.hashCode() == n`), check the boxes next to the numbers assigned to the given partition block.

   i. block 0: ☐ 8 ☐ 23 ☐ 39 ☐ 40 ☐ 97 ☐ none

   ii. block 1: ☐ 8 ☐ 23 ☐ 39 ☐ 40 ☐ 97 ☐ none

   iii. block 2: ☐ 8 ☐ 23 ☐ 39 ☐ 40 ☐ 97 ☐ none

   iv. block 3: ☐ 8 ☐ 23 ☐ 39 ☐ 40 ☐ 97 ☐ none

   (b) Using range partitioning with ranges [0, 20], [21, 40], [41, 60], [61, 100], check the boxes next to the numbers assigned to the given partition block.

   i. block 0: ☐ 8 ☐ 23 ☐ 39 ☐ 40 ☐ 97 ☐ none

   ii. block 1: ☐ 8 ☐ 23 ☐ 39 ☐ 40 ☐ 97 ☐ none

   iii. block 2: ☐ 8 ☐ 23 ☐ 39 ☐ 40 ☐ 97 ☐ none

   iv. block 3: ☐ 8 ☐ 23 ☐ 39 ☐ 40 ☐ 97 ☐ none

   (c) Which strategy would result in a more balanced distribution of the data across the partition?

   ☐ hash partitioning   ☐ range partitioning

3. (a) Which method can we use to determine whether Spark recognizes that a transformation or action will result in shuffling?

   ☐ `debugDAG` ☐ `isShuffled` ☐ `showSchema` ☐ `showExecutionPlan` ☐ `toDebugString`

   (b) How data is initially partitioned and arranged on the cluster doesn't matter, since Spark will always re-arrange your data to avoid shuffling.

   ☐ True ☐ False

   (c) `reduceByKey` running on a pre-partitioned ROD will computed values locally, requiring only the final reduced values to be sent from workers to the driver.

   ☐ True ☐ False

   (d) `join` called on two RDDs that are pre-partitioned with the same partitioner and cached on the same node will cause the join to be computed locally, with no shuffling across the network.

   ☐ True ☐ False

   (e) Suppose algorithm **A** joins two RDDs and then performs a filter on the result while algorithm **B** performs a filter on the two RDDs and then joins the results. Assume the two algorithms obtain the same result. In general, which algorithm do you expect will cause less data shuffling?

   ☐ **A** ☐ **B**

4. Answer the following parts by typing in the spaces provided. Select from among the following words or phrases: "at most one," "multiple," "fast," "slow," "some," or "none."

   (a) In a *narrow dependency*, each block of the parent RDD may be used by ____________ block(s) of the child RDD.

   Narrow dependencies are ____________ since they require ____________ of the data to be shuffled.

   (b) In a *wide dependency*, each block of the parent RDD may be used by ____________ block(s) of the child RDD.

   Wide dependencies are ____________ since they require ____________ of the data to be shuffled.

5. (a) The *query optimizer* of Spark SQL is called

   ☐ Catalyst ☐ Cobalt ☐ Map Reduce ☐ Platinum ☐ Tungsten

   (b) The *off-heap serializer* of Spark SQL is called

   ☐ Catalyst ☐ Cobalt ☐ Map Reduce ☐ Platinum ☐ Tungsten

6. (a) Conceptually, DataFrames are RDDs that contain

   - ☐ AWS S3 buckets
   - ☐ Microsoft Azure blobs
   - ☐ Excel spreadsheets
   - ☐ Row objects with a known schema
   - ☐ Row objects with type information that is checked at compile time

   (b) Which of the following can be used to construct a schema identical to the schema that spark would infer if given a collection of objects of type

   `case class Person(name: String, age: Int)`?

   ```
   ☐ Struct(List(Field("name", String), Field("age", Integer))
   ```
   ```
   ☐ StructType(List(Field("name", StringType, false),
                     Field("age", IntegerType, false))
   ```
   ```
   ☐ StructType(List(StrucField("name", TypedString)),
                List(StructField("age", TypedInteger))
   ```
   ```
   ☐ StructType(List(StrucField("name", StringType, true),
                     StructField("age", IntegerType, true))
   ```
   ```
   ☐ Structured(StructuredField("name", String, Boolean) ::
                     StructuredField("age", Integer, Boolean) )
   ```

7. (a) Navigate to the Spark API documentation and search for `RelationalGroupedDataset` (the type returned when one calls `groupBy` on a DataFrame). Which of the following is **not** a method of the `RelationalGroupedDataset` class?

   ☐ `agg` ☐ `as` ☐ `avg` ☐ `count` ☐ `min` ☐ `round` ☐ `sum`

   (b) Navigate to the Spark API documentation search for `DataFrame`, and notice that none of the results is about the `DataFrame` type itself. This is because

   - ☐ `DataFrame` is just an alias for `Dataset[Row]`.
   - ☐ `DataFrame` is not a type we really use in Spark or Spark SQL.
   - ☐ `DataFrame` is from Spark version 1.0; it is deprecated (no longer supported) in Spark 2.0 or Spark 3.0.
   - ☐ `DataFrame` should be spelled `Dataframe`; if you search for `Dataframe` instead, many results appear.

8. (a) `reduceByKey` is a useful method available for RDD's, but is not a method of the `Datasets` class.

   ☐ True ☐ False

   (b) If `reduceByKey` is not available for Datasets, which of the following approaches could be used to carry out a Map-reduce operation equivalent to `reduceByKey`?

   - ☐ `groupByKey` followed by `mapGroups`
   - ☐ `groupByKey` followed by `mapValues` followed by `reduceGroups`
   - ☐ `groupByKey` followed by `agg` with a specially constructed `Aggregator` object as argument
   - ☐ all of the above

9. (a) If you have unstructured data, you need to fine-tune and manage low-level details of RDD computations, and you have complex data types that cannot be serialized with Encoders, then you should

   ☐ RDDs ☐ DataFrames ☐ Datasets

   (b) If you have structured/semi-structured data and you want the best possible performance, automatically optimized for you, then you should use

   ☐ RDDs ☐ DataFrames ☐ Datasets

   (c) If you have structured/ semi-structured data, you want typesafety, you need to work with functional APls and you need good performance but it doesn't have to be the best, then you should use

   ☐ RDDs ☐ DataFrames ☐ Datasets