1. Describe a business situation in which Hadoop would be a better choice for storing the data than a relational database and explain why (give at least three reasons)

An analytics environment built to support medical research in a hospital would be better served by Hadoop, than by a relational database, for the following reasons:

a. Diverse Data Sets - The data you would capture in a hospital environment, that would be useful to medical research, would most likely come in unstructured formats. This data could include patient charts, medical imaging files, prescriptions, etc. Hadoop is better suited to processing unstructured or semi-structured data. A relational database requires the data to conform to a highly structured schema.

b. Data Volume - The amount of medical data collected in a hospital would be sufficiently large to justify the use of a "big data" environment such as Hadoop. Relational databases will "max out" due to hardware limitations, if the data set being processed is too large.

c. Scalability - Hadoop is designed to scale horizontally using commodity hardware, which is more cost-effective and fault tolerant. As data volumes grow and processing requirements are added, the capacity of the environment can be easily expanded. Scaling a relational database environment usually means scaling a single server vertically to handle growing volumes of data, which is much more complex and expensive.

2. Describe a business situation in which a relational database would be the better option and explain why (give at least three reasons)

Online shopping sites are a common example of where a relational database would be a more sensible approach, rather than Hadoop, for the following reasons:

a. Low-latency - Popular shopping websites process high volumes of data in near real-time. Users may need to search a large product catalog or retrieve their order history, which means selecting a small number of records out of a set of millions. This all has to be done within seconds or less, to maintain a positive user experience that will make the user want to continue to use the application. Relational databases offer low-latency access, whereas data retrieval in Hadoop is comparatively slow. It can take minutes, hours, days or weeks, depending on the size of the data set; and performs poorly on even small data sets.

b. Transactions - Relational databases excel at providing support for logical units of work called transactions. Transactions allow for recovery in the event of failure; and allow concurrent access to data to provide the same results as if the operations were sequential. In the context of our online shopping application, concurrent access is an important consideration, given that a popular site can have thousands of concurrent users accessing the database. With respect to failure, relational databases allow the application programmer to handle failures gracefully and in a manner consistent with user expectations. For example, if a failure occurs during checkout, the user wouldn't want their credit card to be billed, without the product being shipped. By grouping these 2 operations into a single transaction, you can guarantee that both operations occur or neither occurs. Transaction support in Hadoop is limited, thus it cannot fully provide these benefits.

c. Application Support - Relational databases have been around for decades. There is an enormous base of IT talent that is familiar with relational database technology and SQL. Database administrators, database programmers and database analysts are all required resources for supporting applications like the online shopping site we are discussing. It would be easy for a business to staff these roles today and replace those resources as turnover occurs. However, Hadoop is a much newer technology with far fewer experts available in the resource pool.

3. Describe a business situation where MongoDB would a better choice than either and again give at least three reasons

A contact management application might be a good example of where Mongo DB might be a better choice than a relational database or Hadoop. Some of the benefits that a document database would provide over relational databases or Hadoop would include:

a. Locality - There could be several one-to-many relationships in a contact management application, between a contact and its addresses, phone numbers, email addresses, URLs, notes, etc. In a relational database, each of these entities would be represented by a separate table and the interfacing application would need to either execute multiple queries to gather all of the contact's information, or perform complex join logic to collect it all. In Hadoop, MapReduce logic has to be run on every node which contains a block of data that is relevant to the contact and the results all have to be aggregated. In MongoDB, a single query could return a document which has all of the required information embedded within it.

b. Schema flexibility - Not every contact may look the same. You may have email addresses for some contacts, but not for others. You may have notes for some contacts, but not for others. And for some contacts, you may want to add details which are not relevant to other contacts. This implies, at the very least, some level of schema flexibility in the data. Document databases are typically schema-less, however application logic may impose some expectations of what the data in the documents will look like, meaning the schema is more flexible than concrete. This is a stark contrast to structured data found in relational databases or Hadoop, where the data structures are pre-defined and the application must adhere to those constraints. Schema flexibility may be found in unstructured data in Hadoop, but comes at a significant performance cost.

c. Simpler Application Code - In our contact management system example, our data (contacts) has a document-like structure, thus intuitively we would select a document database like MongoDB over a relational database or Hadoop. The data itself is modelled to look similarly to how it would be presented to the user, meaning there is little need for complex application logic to make changes to the data before presenting it to the user. By contrast, storing the data in a relational database or Hadoop, would not only make it more difficult to extract all of the details of a contact, but more work would be expended to format/render the data in a manner that is simple and meaningful to the user.

4. What is the point in having and using Pig if we have Hive so can use SQL (again, three advantages)?

Both Pig and Hive provide a means for simplifying the development of MapReduce programs to be executed in Hadoop. Hive provides a convenient means for querying data, whereas Pig can be used to develop ETL processes, which means Pig has the following advantages over Hive:

a. Apache Pig can be used to create data flows involving all types of data (structured or unstructured). Hive works only with structured data.

b. Pig is a procedural language, which allows the developer to specify how the data will be processed to achieve the desired end-result. Hive is declarative, which means the developer can only specify what results should be returned, not how the end-result should be achieved. This means a lot of important decisions, particularly performance related decisions, are being left up to the query optimizer, rather than being left up to the discretion of the developer.

c. Debugging is easier in a procedural language like Pig Latin, due to its step-by-step nature. Conversely, a declarative language like SQL doesn't decompose its work into steps. You specify what work you would like to be completed, but that declaration is decomposed into steps by the query optimizer, not the programmer. This makes it impossible to inspect intermediate results when trying to determine the source of a problem.