Question 1: One way of classifying data is into the categories: structured, semi-structured and unstructured. Give a brief description of each type and an example of each type.

__Answer:__ **(1) Structured data is data that adheres to a pre-defined data model with a well-defined schema consisting of entity types and is therefore straightforward to analyze. For example, SQL. (2) Semi-structured data is a form of structured data that does not conform with the formal structure of data models associated with relational databases or other forms of data tables, but nonetheless contain tags or other markers to separate semantic elements and enforce hierarchies of records and fields within the data. For example, JSON files. (3) Unstructured data is information that either does not have a predefined data model or is not organized in a pre-defined manner. For example, audio files.**

__Online references:__ **https://www.bigdataframework.org/data-types-structured-vs-unstructured-data/**

Question 2: What is the difference between a type, e.g. VARCHAR, INT, and a *domain?* Given an example.

__Answer:__ **A type is an attribute that specifies the type of data that the object can hold. A domain is a data type with optional constraints (restrictions on the allowed set of values). For example, 'VARCHAR' is a data type, and 'VARCHAR(64) Not NULL' is a domain.**

Question 3: Assume that you know that access to a relational table will be primarily sequential. How would you lay the blocks out on a disk's cylinders, heads and sectors?

__Answer:__ **I would lay the blocks out sequentially on a disk's cylinders and sectors to reduce seek and rotate times. When a cylinder is full, I will move to the adjacent cylinder. Therefore, consecutive data is stored in consecutive sectors to improve the performance.**

Question 4: Most databases do not support hash indexes. What are three benefits of B+ Tree indexes over hash indexes? Express your answer by giving examples of SQL queries better supported by B+ Trees.

<u>Answer:</u> **(1) B+ Tree indexes can support column comparisons that use the =, >, >=, <, <=, or BETWEEN operators while hash indexes can only support equality comparisons that use =, <=> operators. (2) The B+ Tree indexes supports where clauses that not all key fields are present while hash indexes cannot. (3) B+ Tree indexes are self-balanced while hash indexes may have collisions. For example, consider the SQL query SELECT * FROM people WHERE salary <= 1000 AND salary >= 100′ . With B+ Tree indexes, it takes O(logn) time while for hash indexes, it takes O(n) time to search over the database.**

Question 5: What is the primary benefit of RAID-5 compared to RAID-0 and RAID-1?

<u>Answer:</u> **RAID-5 combines striping and parity to provide a fast and reliable setup, while RAID-0 only focuses on fast and RAID-1 only focuses on reliable.**

Question 6: Briefly explain the difference between fixed size records and variable size records.

<u>Answer:</u> **A fixed length record is one where the length of the fields in each record has been set to be a certain maximum number of characters long. A variable length record is one where the length of a field can change to allow data of any size to fit. Therefore, perfoming finding operations in fixed size records is much easier.**

<u>Online references:</u> **https://www.atg.world/view-article/Fixed%20And%20Variable%20Length%20Records%20-2127/fixed-and-variable-length-records**

Question 7: The database query optimizer may create a hash index to optimize a query. Give an example of a SELECT statement for which the optimizer may create a hash index.

<u>Answer:</u> **Consider the example 'SELECT name FROM people WHERE fatherName = 'David' '. Here, the query uses equality predicates and the WHERE clause maps to all index key columns, therefore the optimizer may create a hash index.**

Question 8: Least-recently-used is a common buffer pool replacement policy. But, sometimes LRU is the worst possible algorithm. Briefly give an example of when LRU is the worst algorithm and why.

<u>Answer:</u> Consider the example when computing the join of 2 relations r and s by a nested loop as follows.

```
for each tuple tr of r do
   for each tuple ts of s do
      if the tuples tr and ts match …
```

According to the rule of LRU, in the inner loop, every time we replace the first visited tuples in s to store new tuples. But after an outer loop, we will start from the beginning of s again, which has already been replaced, so we are always replacing during the loop. Therefore, LRU is the worst algorithm in this case.

Question 9: Briefly explain the concept of conflict serializable and how it differs from serializable.

<u>Answer:</u> A schedule is called conflict serializable if it can be transformed into a serial schedule by swapping non-conflicting operations. Conflict serializable is a subset of serializable, so a schedule is conflict serializable implies that the schedule is serializable.

Question 10: What is a deadlock? How do DBMS transaction managers break deadlock?

<u>Answer:</u> In a database, a deadlock is an unwanted situation in which two or more transactions are waiting indefinitely for one another to give up locks. DBMS transaction managers will choose a victim transaction to abort and roll back later. Also, they will analyze the operations of the transactions to prevent the deadlock from happening.

<u>Online references:</u> https://www.geeksforgeeks.org/deadlock-in-dbms/