Question #1: Define the following terms: [10 Points]

a) Database management system (DBMS)

1. A DBMS is a software system or package that is used to maintain and facilitate the creation of a computerized database. It can be used to store, retrieve, run queries on, and even delete data. An example of DBMS is Oracle or even mySQL.

b) File-processing systems

1. A file processing system is a method to address data written in files. In this approach, the data is stored in individual files. This is often used in businesses with simple data storage and retrieving requirements.

c) Data dictionary

1. A data dictionary is also called Meta data or Data definition. It's an integral part of a DBMS and helps determine its structure. Essentially, everytime you build a database, the DBMS creates metadata which can be described as data about the data. It holds information about the data such as its meaning, origin, usage, and format.

d) Data abstraction

1. Data abstraction refers to suppressing specific information regarding data organization or storage, highlighting only information that could provide valuable insights. Essentially, it is used to hide storage details and present the users with a conceptual view of the database

e) Physical data independence

1. Physical data independence means having the capacity to change the internal schema, such as the data stored in a database, without changing the conceptual schema. For example, you could change the hashing techniques or the location of the database without affecting the conceptual schema.

Question #2: Give examples of systems in which it may make sense to use traditional file processing instead of a database approach. [8 Points]

1. It would make more sense to use a traditional file processing approach if one was creating a system to keep track of their individual budget or perhaps their direct family's budget. There isn't too much data to be stored and it's a fairly simple task so it wouldn't make sense to pay for a DBMS and handle the extra overhead of handling a DBMS.

Question #3: Discuss the differences between DDL and DML. What operations would you typically expect to be available in each language? [8 Points]

1. DDL is used by administrators and designers to design or specify the various schemas. It essentially defines the database and works on creating it. Typical operations with a DDL would be CREATE, ALTER, RENAME, etc. These operations actually affect the structure of the database.
2. DML commands can be embedded in an application written in various languages, or can also be applied directly without the use of an application. Casual users work using DML commands. Typical operations with DML commands would be SELECT, UPDATE, INSERT, DELETE, etc. These operations allow us to manipulate the data within the database but not actually affect the true structure.

Question #4: What problems are caused by data redundancy? can data redundancies be completely eliminated when the database approach is used? why or why not? [8 Points]

1. A big issue caused by data redundancy is inconsistency. For example, if the same data is stored in multiple places but is only being updated in one place, it could lead to conflicting or incorrect information.
2. Data redundancies cannot be completely eliminated when a database approach is used because when a database is used, the data needs to relate to one another in some way. For example, to connect two tables together regarding students at a college, they would need to have some data (such as student ID) in both of the tables to be able to establish the connection.

Question #5: What is the difference between a database schema and a database state? Please, use examples to support your answer. [8 Points]

1. The database schema is the description of the database, such as database structure, data types, etc. The database state is the actual data that is being stored in the database at that specific moment in time. For example, if you have a database that stores students and their classes, if one specific student decides to drop a specific class that they're taking, it wouldn't change the schema but it would change the state. Since the actual data is being changed, the database state would change, but since the description of the database isn't changing (it's still only holding students and their classes), the database schema would change.
2. Overall, the schema changes very infrequently since it's less likely that you'll need to add another table or relationship attribute whereas the database state changes very often. Most end users have the ability to change the database state but only DBA (database administrator) have the ability to change the schema.

Question #6: [8 Points]

a) Describe at least 3 tables that might be used to store information for a video rental store.

1. Customers table
    a. This table would have a customer ID (primary key) , first and last name, membership (yes or no), phone number, email address, and physical address.
2. Videos table
    a. This table would have video ID (primary key), name of video, genre, date released, ratings, and availability.
3. Rental's table
    a. This table would have Rental ID (primary key), customer ID (foreign key), video ID (foreign key), due date, return date, and fee.

b) Identify some informal queries and update operations that you would expect to apply to the tables you specified in part(a).

1. Insert a new video in the Videos table. This would add another row in the table, and fill in the required information such as the name of the video, genre, date released, rating, and availability.
2. Check which customers have overdue rentals. For this one you would have to select from Rental's table where the return date has passed.
3. Check which videos are PG. From the videos table we could select the videos that have the rating 'PG'.

c) Specify all the relationships among the tables that you specified in part (a).

1. The primary key in the Videos and Customers table is a foreign key in the Rentals table. This creates the relationship between the tables.

d) Cite some examples of integrity constraints that you think can apply to the tables you specified in part (a).

1. Some integrity constraints that could occur with this table include duplicate ID's. For example if two customers had the same ID that could lead to collisions which is why the ID's in each table must be unique.
2. Another integrity constraint could be if the store decided to stop renting out a certain video. The videoID was deleted from the Videos table but not from the Rentals ID. This would allow the user to rent out a video that doesn't exist anymore and lead to issues.
3. Lastly, if the values in return date column were allowed to be null, that could also lead to issues.