Q1) A database connection is a facility in computer science that allows client software to talk to database server software, whether on the same machine or not. A connection is required to send commands and receive answers, usually in the form of a result set. Connections are a key concept in data-centric programming. Since some DBMS engines require considerable time to connect, connection pooling was invented to improve performance. No command can be performed against a database without an "open and available" connection to it. Connections are built by supplying an underlying driver or provider with a connection string, which is a way of addressing a specific database or server and instance as well as user authentication credentials. Once a connection has been built it can be opened and closed at will, and properties (such as the command time-out length, or transaction, if one exists) can be set. The Connection String is composed of a set of key/value pairs as dictated by the data access interface and data provider being used. Many databases (such as PostgreSQL) only allow one operation to be performed at a time on each connection. If a request for data (a SQL Select statement) is sent to the database and a result set is returned, the connection is open but not available for other operations until the client finishes consuming the result set. Other databases, like SQL Server 2005 (and later), do not impose this limitation. However, databases that provide multiple operations per connection usually incur far more overhead than those that permit only a single operation task at a time.

Q2)Connection pooling is a technique designed to alleviate this problem. A pool of database connections can be created and then shared among the applications that need to access the database. The connection object obtained from the connection pool is often a wrapper around the actual database connection. The wrapper understands its relationship with the pool and hides the details of the pool from the application. For example, the wrapper object can implement a "close" method that can be called just like the "close" method on the database connection. Unlike the method on the database connection, the method on the wrapper may not actually close the database connection, but instead return it to the pool. The application need not be aware of the connection pooling when it calls the methods on the wrapper object. This approach encourages the practice of opening a connection in an application only when needed, and closing it as soon as the work is done, rather than holding a connection open for the entire life of the application. In this manner, a relatively small number of connections can service a large number of requests. This is also called multiplexing. In a client/server architecture, on the other hand, a persistent connection is typically used so that the server state can be managed. This "state" includes server-side cursors, temporary products, connection-specific functional settings, and so on. An application failure occurs when the connection pool overflows. This can occur if all of the connections in the pool are in use when an application requests a connection. For example, the application may use a connection for too long when too many clients attempt to access the website or one or more operations are blocked or simply inefficient.