Task 7

Q1 What is xml language?

What is XML?

- XML stands for eXtensible Markup Language
- XML is a markup language much like HTML
- XML was designed to store and transport data
- XML was designed to be self-descriptive
- XML is a W3C Recommendation

XML and HTML were designed with different goals:

- XML was designed to carry data - with focus on what data is
- HTML was designed to display data - with focus on how data looks
- XML tags are not predefined like HTML tags are

XML simplifies data sharing

XML simplifies data transport

XML simplifies platform changes

XML simplifies data availability

Many computer systems contain data in incompatible formats. Exchanging data between incompatible systems (or upgraded systems) is a time-consuming task for web developers. Large amounts of data must be converted, and incompatible data is often lost.

XML stores data in plain text format. This provides a software- and hardware-independent way of storing, transporting, and sharing data.

XML also makes it easier to expand or upgrade to new operating systems, new applications, or new browsers, without losing data.

With XML, data can be available to all kinds of "reading machines" like people, computers, voice machines, news feeds, etc.

Q2 why spark in analysis?

It processes streams of data, enabling real-time analytics. Spark Structured Streaming can accept data from many applications and in many formats. Streams can be processed like tables, and tables can be acted upon like streams. Structured Streaming removes the underlying complexity and is built on the Spark SQL engine.

Spark was created to address the limitations to MapReduce, by doing processing in-memory, reducing the number of steps in a job, and by reusing data across multiple parallel operations.

Q3 design patterns types?

What are Design Patterns?

Design patterns are reusable solutions to common problems in software design. They represent best practices used by experienced object-oriented software developers. Design patterns provide a standard terminology and are specific to particular scenarios.

Types of Software Design Patterns

There are three types of Design Patterns:

- Creational Design Pattern
  -Creational Design Pattern abstract the instantiation process. They help in making a system independent of how its objects are created, composed and represented.

1. Factory Method Design Pattern

The Factory Method pattern is used to create objects without specifying the exact class of object that will be created. This pattern is useful when you need to decouple the creation of an object from its implementation.

2. Abstract Factory Method Design Pattern

Abstract Factory pattern is almost similar to Factory Pattern and is considered as another layer of abstraction over factory pattern. Abstract Factory patterns work around a super-factory which creates other factories.

3. Singleton Method Design Pattern

The Singleton method or Singleton Design pattern is one of the simplest design patterns. It ensures a class only has one instance, and provides a global point of access to it.

4. Prototype Method Design Pattern

Prototype allows us to hide the complexity of making new instances from the client. The concept is to copy an existing object rather than creating a new instance from scratch, something that may include costly operations. The existing object acts as a prototype and contains the state of the object.

5. Builder Method Design Pattern

Builder pattern aims to "Separate the construction of a complex object from its representation so that the same construction process can create different representations." It is used to construct a complex object step by step and the final step will return the object.

- Structural Design Pattern
  - *Structural Design Patterns are concerned with how classes and objects are composed to form larger structures. Structural class patterns use inheritance to compose interfaces or implementations.*

1. Adapter Method Design Pattern

The adapter pattern convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces.

2. Bridge Method Design Pattern

The bridge pattern allows the Abstraction and the Implementation to be developed independently and the client code can access only the Abstraction part without being concerned about the Implementation part.

3. Composite Method Design Pattern

Composite pattern is a partitioning design pattern and describes a group of objects that is treated the same way as a single instance of the same type of object. The intent of a composite is to "compose" objects into tree structures to represent part-whole hierarchies.

4. Decorator Method Design Pattern

It allows us to dynamically add functionality and behavior to an object without affecting the behavior of other existing objects within the same class. We use inheritance to extend the behavior of the class. This takes place at compile-time, and all the instances of that class get the extended behavior.

5. Facade Method Design Pattern

Facade Method Design Pattern provides a unified interface to a set of interfaces in a subsystem. Facade defines a high-level interface that makes the subsystem easier to use.

6. Flyweight Method Design Pattern

This pattern provides ways to decrease object count thus improving application required objects structure. Flyweight pattern is used when we need to create a large number of similar objects.

7. Proxy Method Design Pattern

Proxy means 'in place of', representing' or 'in place of' or 'on behalf of' are literal meanings of proxy and that directly explains Proxy Design Pattern. Proxies are also called surrogates, handles, and wrappers. They are closely related in structure, but not purpose, to Adapters and Decorators.

- Behavioral Design Pattern

  - *Behavioral Patterns are concerned with algorithms and the assignment of responsibilities between objects. Behavioral patterns describe not just patterns of objects or classes but also the patterns of communication between them. These patterns characterize complex control flow that's difficult to follow at run-time.*

1. Chain Of Responsibility Method Design Pattern

Chain of responsibility pattern is used to achieve loose coupling in software design where a request from the client is passed to a chain of objects to process them. Later, the object in the chain will decide themselves who will be processing the request and whether the request is required to be sent to the next object in the chain or not.

2. Command Method Design Pattern

The Command Pattern is a behavioral design pattern that turns a request into a stand-alone object, containing all the information about the request. This object can be passed around, stored, and executed at a later time

3. Interpreter Method Design Pattern

Interpreter pattern is used to defines a grammatical representation for a language and provides an interpreter to deal with this grammar.

4. Mediator Method Design Pattern

It enables decoupling of objects by introducing a layer in between so that the interaction between objects happen via the layer.

5. Memento Method Design Patterns

It is used to restore state of an object to a previous state. As your application is progressing, you may want to save checkpoints in your application and restore back to those checkpoints later. Intent of Memento Design pattern is without

violating encapsulation, capture and externalize an object's internal state so that the object can be restored to this state later.

6. Observer Method Design Pattern

It defines a one-to-many dependency between objects, so that when one object (the subject) changes its state, all its dependents (observers) are notified and updated automatically.

7. State Method Design Pattern

A state design pattern is used when an Object changes its behavior based on its internal state. If we have to change the behavior of an object based on its state, we can have a state variable in the Object and use the if-else condition block to perform different actions based on the state.

8. Strategy Method Design Pattern

The Strategy Design Pattern allows the behavior of an object to be selected at runtime. It is one of the Gang of Four (GoF) design patterns, which are widely used in object-oriented programming. The Strategy pattern is based on the idea of encapsulating a family of algorithms into separate classes that implement a common interface.

9. Template Method Design Pattern

Template method design pattern is to define an algorithm as a skeleton of operations and leave the details to be implemented by the child classes. The overall structure and sequence of the algorithm are preserved by the parent class.

10. Visitor Method Design Pattern

It is used when we have to perform an operation on a group of similar kind of Objects. With the help of visitor pattern, we can move the operational logic from the objects to another class.

Q4 Database Normalization's types ?

Normal Form	Description
1NF	A relation is in 1NF if it contains an atomic value.
2NF	A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
3NF	A relation will be in 3NF if it is in 2NF and no transition dependency exists.
BCNF	A stronger definition of 3NF is known as Boyce Codd's normal form.
4NF	A relation will be in 4NF if it is in Boyce Codd's normal form and has no multi-valued dependency.
5NF	A relation is in 5NF. If it is in 4NF and does not contain any join dependency, joining should be lossless.

Q5 What is Database Indexing?

Indexing improves database performance by minimizing the number of disc visits required to fulfill a query. It is a data structure technique used to locate and quickly access data in databases. Several database fields are used to generate indexes. The main key or candidate key of the table is duplicated in the first column, which is the Search key. To speed up data retrieval, the values are also

kept in sorted order. It should be highlighted that sorting the data is not required. The second column is the Data Reference or Pointer which contains a set of pointers holding the address of the disk block where that particular key value can be found.

Q6 What is Database Schema?

A database schema defines how data is organized within a relational database; this is inclusive of logical constraints such as, table names, fields, data types and the relationships between these entities.

- Conceptual schemas offer a big-picture view of what the system will contain, how it will be organized, and which business rules are involved. Conceptual models are usually created as part of the process of gathering initial project requirements.
- Logical database schemas are less abstract, compared to conceptual schemas. They clearly define schema objects with information, such as table names, field names, entity relationships, and integrity constraints—i.e. any rules that govern the database. However, they do not typically include any technical requirements.
- Physical database schemas provide the technical information that the logical database schema type lacks in addition to the contextual information, such as table names, field names, entity relationships, et cetera. That is, it also includes the syntax that will be used to create these data structures within disk storage.

Q7 What is EER Diagram?

basically help in creating and maintaining excellent databases with the help of smart and efficient techniques. In addition to this, it is a visual representation of the plan or the overall outlook of the database you intend to create.

When to Use EER Diagrams?

- If an organization wants to manage the data of all of its employees.
- In addition to this, it provides an excellent framework for the management of information and its flow.
- This tool can be used in police departments as well. This is because it helps in maintaining detailed databases.
- Other than this, this tool can be used by universities to keep a record of every student.
- Last but not least, this tool can be used by system engineers, network engineers, and software developers as well.