**Question 1**

Question 1a). catalogue/book/author

Question 1b). sum(/catalogue/book/@stock)

Question 1c). catalogue/book[author="Proust, Marcel"]/title

Question 1d). catalogue/book[tags[contains(.,"xpath")]]/title

Question 1e). sum(catalogue/book/(@stock \* price))

**Question 2**

Question 2a). for $x in doc("books.xml")/catalogue/book

order by $x/title

return $x/title

Question 2b). <ul> { for $x in doc("books.xml")/catalogue/book

order by $x/title

return <li value ="{$x/@id}">{$x/title/text()} </li>

} </ul>

Questions 2c). for $x in doc("books.xml")/catalogue/book

where $x/price<10

return $x/tags

Question 2d). for $x in doc("books.xml")/catalogue/book

where $x/tags[contains(., "xquery")]

return $x/title

**Question 4**

Question 4a).

XML is a text-based markup language. It was designed to store and transport data in a human and machine-readable format. XML allows to for the creation of your own tags, meaning you can design tailor-maid xml files to suit the format in which you want to store data and fit it for the application you are using.

XSLT is a styling language for XML data and is used to transform the way in which XML documents look. It defines a set of rules or a structure in which to transform the XML document. This transformed document can be in the form of another XML document, HTML or a text-based document. It uses XPath in order to locate specific attributes or parts or nodes in an XML document to transform.

XPath is a querying language which can be used to identify specific nodes and attributes in an XML document. It is used in XSLT and XQuery to locate parts of an XML document to perform their respective tasks.

XQuery is a language used to query and process data that is stored in as XML. It can be used to query XML databases, XML documents as well as relational databases that hold data in XML formats. It uses XPath in order to query for specific attributes in XML data.

XLink is a member of the XML family and is used to create hyperlinks in XML documents. It behaves similarly to how the <a> tag is used in HTML.

XPointer is a member of the XML family that is used in combination with XLink in order to direct towards specific parts of an external XML document. It uses XPath to navigate the desired document.

Question 4b).

Benefits of XML technologies for an organization:

One of the advantages of using XML is that it is extensible. It allows an organisation to store and organise its data using custom, self-made tags. This provides a benefit of being able to create a data format tailor made to fit an organisations and their users’ requirements.

XML is the current industry standard, and thus is widely understood by many. There are also many platforms and programs that operate using XML technologies, adopting the family of technologies allows for an organisation to utilise such platforms.

Another advantage is that XML uses the Unicode character set and supports multilingual documents. This extremely large character set can store text from every language. For organisations that operate with users on a global scale this can be a great benefit.

XML technologies help simplify the process of data sharing. It is often the case that systems and programs contain data in formats that are not compatible with one another. As XML data is stored in a plain text format, XML comes as a go between for the applications, converting from their individual format to XML in order to share data.

XML is human and machine readable, this comes as a big advantage as even to new users of XML or those who do not have technical knowledge, it should be easy to interpret.

XML documents keeps the content separate from the presentation. As the visual output of an XML file can be controlled by the subset of XSL technologies, a single XML document can have multiple views and outputs without changing any part of the original document.

XML can embed any possible data type, this includes each type of media data such as images, videos and sound. This means that an organisation can utilise XML for anything they wish to do with the data.

XML does not cost anything to use, this is a huge benefit to organisations, particularly those operating on a small scale. It can be written using just a simple text editor. There are many free platforms that operate using XML also.

**Question 6**

Question 6a).

Polyglot persistence refers to the notion of using multiple technologies to store data. Each data storage method is used to fit the way in which an organisation needs to use their data. Relational databases are fit for the use of structured data of a pre-defined format; they are rigid and are not suited to handle big data. Big data refers to data that comes in large volumes with often comes in an unstructured format with lots of variety. Social media platforms such as twitter is a place in which big data is found commonly, and a relational database would not be suited to handle billions of tweets that can come in different formats and incorporate different types of media such as videos, photos or text based. Unlike traditional relational databases, NoSQL databases are more flexible in the ways in which they can store and process data.

A common need of organisations is the ability to handle data that will scale with an organisations growth, a relational database with the structure of millions of rows or many tables would cause the databases to perform slowly when trying to perform queries and if changes in schema or structure are required then it can become a problem that is difficult to combat. NoSQL databases do not require a fix and rigid schema, meaning they are way more suited to handle unstructured data that is present in a variety of forms. Many types of NoSQL databases are designed to run on clusters, offering the ability to store large sets of data and analytical data which organisations can used to aid the decision-making process. Many organisations require using their data in an analytical sense to help improve company decision making, utilise something like a graph NoSQL database can help to discover relationships between large sets of data to improve this feat.

Question 6b).

Scalability is an important factor for a database system, the ability to retain high performance upon growing and holding a larger amount of data is needed for organisations. Vertical scaling in databases refers to the process of adding more resources and improving the hardware of an existing database server in order to increase its performance. This can be done by increasing the number of CPU’s or improving the quality of the CPUs to improve the ability of the system to process data or to add more memory to increase the amount of data that can be held. Horizontal scaling refers to the process of increasing the number of servers on the system and distributing the data over that number of servers, resulting in a distributed system.

In terms of scalability, the main difference between the two methods is that vertical scaling has a limit, you are restricted by how powerful the current hardware is and it can be expensive to purchase the top end hardware. It is more suited to small scale organisations. Horizontal on the other hand there is no limit, you can always add more servers to the system, and it isn’t necessary to purchase the top-level hardware for every server, this is more fitting for organisations that use big data whereas vertical scaling is more suited to smaller-sized systems. A key difference is that vertical scaling does not change the structure of the system and just improves its performance, meaning it is often easier to implement and results in a less complex system as it is continuously running on a single instance whereas in horizontal scaling, the overall system might expand too so many servers that it is difficult to manage, and it can also be a time-consuming process to implement. In a system using vertical scaling if it is plausible that frequent querying will be a thing it can be difficult for a single server to handle this number of requests which can result in downtime. Horizontal scaling lowers the risk of such downtime due to the shared workload between all the servers involved. Vertical scaling is more so used in relational databases and virtualisation. Horizontal scaling is more used when working with NoSQL databases.

Question 6c).

Key/Value Databases

Key value pairs are more effective when working with simple data sets which require more simple queries. Based on the scenario above in a key value database, the invoice ID and the attribute of the invoice would be the key and there would be a key value pair created for each part of the invoice. In order to get information about the customers and products individually like the scenario is suggesting is of high interest, there would have to be key value pair for customers and products, where the CustomerID might be the key and then information about the customer would be the value. This would quickly increase the number of pairs required to hold the information and would not be the most efficient method of storage out of the 3 options offered.

e.g. InvoiceIDCustomerID: 1

InvoiceIDProducts: “pen, paper”

Whilst a key value database would be able to handle the large and growing number of invoices, should the invoices become rather complex, this format would not be too efficient to store this kind of data. If there happened to be 10+ attributes assigned to each invoice, it would require 10 key value pairs to enter this data or one large data set per key. Due to the potentially complexity of the data stored and lack of ability to easily query and discover relationships, key-value databases would not be suited for the persistence layer.

Document Databases

A document database is an expanded form of a key-value database where a document can be made up of many key-value pairs in XML or JSON formats. Many documents are stored in a collection. In the scenario provided, it is likely that there will be a collection for the 3 main parts of the scenario, the invoices, the products and the customers.

The invoice collection might hold documents about the invoice, it is likely to store nested arrays for both the customer and product inside each document. The customer array would hold the information about the customer and the product array would hold the list of products that they purchased for that specific invoice.

The customers collection might hold customer information such as a name, address and payment information.

The product collection might hold all the information about the products, with examples being name and price.

Holding data in this format can allow for easy querying about specific invoices. It can easily deal with the large and growing collection of data that will be present. Due to having no rigid structure, it will be optimal for adding new kinds of products and various kinds of invoice to their respective collections. Document databases would be suited for the role of the persistence layer.

Graph databases

Graph databases are made up of objects and connections between those objects, known as nodes and edges. The main advantage offered by graph databases is the ability to identify relationships between parts of the database. Applying this knowledge to the example provided, the data might be stored in a format where each invoice, product and customer are stored as their own respective nodes with the properties needed for each one. They would then be connected to display the relationships between them.

An example could be having five invoices linked to a customer, and 3 products linked to a particular invoice. As the scenario says that customers and the products they purchase are of high interest, being able to display the relationships between the data can help the organisation discover trends between which customers buy which products, which can assist in marketing decisions for an organisation. For example, what age range mainly buys which products, which sex?.

Graph databases makes it easy to format complex queries. Even on particularly large sets of data, graph database retains a high-performance level in regards of query execution time. Thus, graph databases would be very suitable to act as the persistence layer in this case.