**1. VERSIONS of Server.**  You have founded a new company with two friends. Your new application (app) uses a SQL Server database to store information. You are unsure whether your app will be successful but if it is, you will need both high performance and space for large volumes of data. However, you have not yet launched, so are unsure how many people will use your app. Which edition of SQL Server 2016 should you use for this system? Why?

Azure SQL Database, because it allows the user to start small and scale up as required. As a startup, using Azure SQL Database also means the user does not need to buy server hardware to run SQL Server. Because Azure SQL Database is Software as a Service (SaaS), the user pays for what they use without high upfront costs.

**2.  What is predicate logic?** How is it a different way of thinking about data and the processing that you want to do on data? What does it mean?  What is a "predicate?"  Describe each of the below T-SQL elements as it relates to predicates:

Predicate logic is a framework for expressing logical tests that return true or false. It is a different way of thinking about data processing because in real life we often encounter situations where the answer is: “it depends.” But with predicate logic, the answer is only true or false. A predicate is a property or expression that is true or false.

* WHERE clauses

WHERE clauses use predicates to determine which rows to return.

* JOIN conditions

JOIN conditions use predicates to determine which rows from different tables to join into a single result row.

* HAVING clauses

HAVING clauses use predicates to determine which groups to return.

* WHILE statements

WHILE statements use predicates to determine when to stop executing T-SQL statements in a batch.

**3. Going wild with wildcards.**  Using wildcards as arguments (such as Select \*) is not recommended. Why? What if you really need to do such a "wild" search, select, or filtering? What trade-offs should you consider? Alternatives?

 When working with extremely large datasets (Big Data), wildcards can consume a significant amount of memory and time while evaluating every value, and could potentially return an enormous amount of data. The trade-off is getting a grouping of data that is easier for the user to find what they are looking for. Alternatives might be to use CONTAINS, or to break down the components of what the user wants into separate tables and then use the wildcard argument on the smaller dataset.

**4.  JOINS.**  Inner and Outer joins are talked about a lot as we use tables and database systems.  What do these mean, and why you'd need to use them. What if you're operating in a non-SQL kind of environment, where you don't have JOIN operators? How might you accomplish the same results?

An INNER JOIN matches rows in separate tables by filtering out rows that fail to meet the conditions expressed in the ON clause predicate, resulting in only rows that match both tables. For example, filtering for EmployeeID values that are matched in HR.Employees and Sales.Orders tables.

With an OUTER JOIN you can choose to display all the rows from one table along with rows that matched from the second table. For example, a LEFT OUTER JOIN displays the results of an INNER JOIN while also displaying all the rows from the statement on the left of JOIN. In columns where there is no match, NULL will be displayed.

In a non-SQL environment, you can accomplish the same results by using a nested IF statement.

**5. Procedural, Object, or Data: Perspectives?**  As a programmer, you have to focus your thinking either on the step-by-step of what you want to do, or the object-by-object, data-item-by-data-item that you need to do those steps to.  How does your viewpoint or thought process change as you "think in SQL?"

SQL uses a declarative programming paradigm that requires the user to specify *what* they want (object-by-object).

Other programming languages like C# use an imperative programming paradigm where the user must specify *how* to get what they want (step-by-step).

As you “think in SQL” your thought process must change from procedural perspective to a data perspective that tells the computer what you want and lets the program determine how best to retrieve the data.