Questions:

- **What questions or details would you ask to the IT department to help you design such an anomaly detection system?**

First, I would ask for general network infrastructure information: a network diagram including subnets, VLANs and IP addresses of critical servers. With this information, I should be able to determine where to place the data extraction node and all the necessary infrastructure configurations.

Then I would ask for the types of attacks they want to avoid so that I could define the feature list and design a data collection plan. This plan would include attack execution in a controlled environment so that I could obtain representative data for each threat. At this point, I would prepare a document defining the data collection plan and send it to the IT department for approval.

Lastly, I would clarify with them the scope of the anomaly detection system. It is especially relevant to know if the system needs to detect the type of attack or only notify it, since this information will be necessary to define the ML problem we are facing (standard classification or binomial classification).

- **How would you collect and process the data necessary to train the ML model? What would be their structure, type, size, etc.?**

I will split the response to this question into two parts: data collection and data processing.

To collect the necessary data, I would perform network traffic captures. As stated in the previous response, the captures must include standard traffic (no attack) and anomalous traffic (during an attack).

If the network is a typical Ethernet hierarchical network, I would configure a monitor port in a switch with access to all the necessary VLANs, then connect the sniffing node to it.

Regarding data processing, we should take into account that analyzing single packets usually does not provide substantial information, except for UDP stateless traffic. For this reason I would extract all the TCP connections from the packet capture. For UDP traffic, I would study the possible attacks using it. If any of those uses an upper-layer stateful protocol (e.g. QUIC protocol), I would extract all connections related to it. If not, we can assume that a single packet can provide enough information to detect an anomaly.

For each dataset entry (each connection), I would need to extract some features. The basic ones would be:
- Source IP (32 bits unsigned int)
- Destination IP (32 bits unsigned int)
- Source port (16 bits unsigned int)
- Destination port (16 bits unsigned int)
- Protocol (categorical)
- Duration (32 bits unsigned int): number of packets sniffed for the connection

For each connection, I would also add time-related and content-related features. For instance:
- Connection count (32 bits unsigned int): number of equal connections performed in the last X minutes

- Critical (boolean): determines if the connection includes actions considered as critical (e.g. remote directory access, program execution or login)

Depending on the type of attacks to detect, we can set some more content-related features. For instance:
- Login attempts (16 bits unsigned int): number of remote login attempts in the last minute. Useful for detecting brute force attacks.
- DHCP server (boolean): determines if a DHCP Offer packet is sent by a trusted DHCP server. Useful for detecting rogue DHCP servers.

Further literature reading should be performed in order to define additional useful features.

Since the attacks are performed in a controlled scenario, each entry can be easily labeled through logical conditions, separating the anomalous traffic from the standard one. Depending on the kind of detection system needed, it is possible to specify the type of attack. Therefore, the label can be a boolean or a categorical variable.

- **Based on your previous answer, what ML algorithms would you consider for your system? How would you select the best model amongst the ones considered, and how would you evaluate the model's performance?**

I would consider the main classification ML algorithms: SVC, Neural Networks, KNN, Decision Tree (+ Random Forest), and Naive Bayes.

In order to choose the metrics to measure the model's performance, it's important to note that for this particular problem (IDS) we want to avoid false negative results since those represent a cyberattack not being detected. False Positive results can be bothersome for the organization, but they do not represent a security threat.

For this reason and since the dataset is not completely balanced, accuracy may not be the most representative metric of the model's performance. Instead, we will be using:

- Recall = TP/(TP+FN) => measures the percentage of true positive detections above all real positive values. A higher recall indicates fewer false-negative detections.

- F2 Score => for a given threshold value, measures the weighted harmonic mean of the precision (TP/(TP+FP)) and recall metrics, giving more weight to the recall (important since we want to avoid false negatives). Useful as a global performance metric for this particular problem.

- AUC => performance measurement for the classification problems at various threshold settings. Useful since F2 is limited to a particular threshold value.

I will train a model with each ML algorithm and evaluate its performance using the metrics stated.

- **The IT department warned the team that the company introduces or discontinues the use of applications (that produce network traffic) with a moderate frequency. This decision can affect the performance of the system proposed. Given this extra information, what changes would you make to the proposed system to work properly in production? What factors do you need to consider for ensuring a good performance in the proposed ML system?**

Since the application environment has become dynamic, the model should receive feedback on the “currently” running applications. The company can maintain a list of running applications, which can be offered to the model as a new feature. The maintenance of the aforementioned list may be automated through monitoring and network discovery software.

So, I would add a new feature to make the model context-aware, and I would design a new data collection plan, including application context changes.