Problem 1

For batch size 1, converge after 130 epochs. Accuracy: 99.6
For batch size 5, converge after 31 epochs. Accuracy: 86.4
For batch size 10, converge after 60 epochs. Accuracy: 86.0
For batch size 75, converge after 311 epochs. Accuracy: 83.9

a. We can see that with better accuracy, the training process converges slower. In SGD, if the convergence threshold is lower, it will take more time for the loss to get within the threshold, and thus more time to break from the while loop. However, with a relatively lower convergence threshold, we get higher accuracy because the loss would be lower as we break from the while loop. We should be aware that the tradeoff can be affected by multiple factors, including the dataset size and learning rate. For instance, with a larger dataset, we tend to converge slower but have a higher chance of reaching higher accuracy; with a smaller learning rate we have more iterations which makes us converge slower and have higher accuracy, but with also a chance of overfitting that lowers accuracy.
b. Batch sizes determine how many samples to look at before making a weight update. In my case, the smaller batch size achieves higher accuracy. Intuitively with a larger batch size we converge faster, but in fact, with batch size 5, I have the fastest converge speed, while as it increases, the converging speed becomes much slower. I think an explanation would be that a smaller batch size means more frequent weight updates and therefore leads to faster convergence. Also, a larger batch size may lead to overfitting which would also reduce accuracy. Overall, for this specific dataset, the batch size works best with the sample size and the learning rate 0.03, and thus having the best performance.

Problem 2

For 'workclass', 'marital-status', 'occupation', 'relationship', 'race', and 'native country', we one-hot encode them. We can notice that these categories correspond to information that are not numbers, so we use one-hot encoding to transform categorical data into numerical. This will help us to use them in Machine Learning algorithms more conveniently. Compare to enumerations of values for a feature, one-hot encoding can prevent the model from assuming there is natural ordering exists between categories. For enumeration we typically assign integer values, and machine learning algorithm can believe that higher number means higher significance, which we do not want it to happen. One-hot encoding assigns binary values to new categorical columns for each categorical value. Each integer value is represented as a binary vector, so as to save the model from this situation.

Problem 3

We can observe that the accuracy decreased significantly, while the converge speed became faster, with the model stopping training after only 2 epochs. Normalizing data is important because it changes the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. Since we don't want features with larger numbers as their value to have higher significance during training, we want them all on the same scale to make it more "fair". This will increase the stability and performance of our model.

Problem 4

The accuracy of running this dataset is 99.7, with 116 epochs and batch size 1, and with the increase of batch size, we can observe an average higher accuracy than the result of the normal dataset. Therefore, we can conclude there is not a correlation between sex/race and education shown by the result of this prediction model. If there is, without this information in the data file, we would have observed a decrease in the accuracy since they would be significant for the model to predict the result. Now without this information, we still get the same or even higher level of accuracy, so this proves that it makes no impact on the model's prediction. Nevertheless, we should be careful about making any definite conclusion on whether there is such a relationship without further analysis and investigation, given that this is just a result of a single prediction model.