Task1:

a.
    i. GPT3.5:
        1. Fine-tuning might not work fine, as 500 new orders are made daily
        2. Prompt will respond with most humanly readable way
    ii. Langchain prompt:
        1. Sometime it will query out wrong match when similarity matching
        2. Querying specific data will be very easy for it's better encapsulation
    iii. Bert Classification:
        1. Fine tuning might not work for both new orders and long context
        2. Will generate good ranking value

b. I'll choose langchain prompt with because it's better for database querying

c. prompt : "return me the product id, which has the highest returns between $date1 - $date2" this is basically the 2nd example with more ai readability and I might use a custom function with it.

d. I had to build a customer care using langchain, for my current company. Where we query out the best match for the asked question from database with guidelines considering which portal the customer is.

Task 2:

a. To be frank, i would not train the database to any model for this specific task,
    i. I'll give the header with two table schema, dropping the category as we can just include it inside the product. Then ask the query in the query section with additional action to generate the appropriate SQL query to find out data. And finally pass it with safe query so sql injection attack can not occur.

b. If i really had to preprocess the database, then i'll obviously sum of all product id for each week, in this manner I will get a good time series dataset to feed into my model.
        1. Group by sale_date
        2. Aggregate product id for total amount

c. If we cannot preprocess it in a time-series we will not get our desired query output, it will be scattered, (data but no information and it will be biased)

d. In text processing, I faced the challenge to keep to vital token and drop redundant token for each entry specially if it's written in bangla cause bangla has very poor tokenization.

Task 3:

a. User_query -> backend -> (preprocess and encapsulate) -> run model -> get query -> run query on database -> return back the values

b. Hard to distinguish
    i. will come through route (django)
    ii. Django model handler
    iii. Will be handled by frontend

c. (will check for attribute name consistency for each table)

Task 4.

a. Future dates crash might happend to preprocessing issue if we ask within range this should not be problem but if we index the date and search directly this will be problem, so i will check any direct access to dates and for 1000+ rows it’s probably due to poor pagination
b. For query we need to follow the rule “don’t trust the user, they will ask anything stupid the can ask. We have to validate each field before query processing.” for 1000+ entry we need to confirm that all data is not delivered at the same time.