The Script from hell solution:

**Question 1**

**What ways can you propose to achieve this? - Please, start with the simpler ideas and give an advantage and a disadvantage of each**

A number of solutions exist for password management :

1-Use a configuration file stored in a separate folder than the project folder and is only accessible internally

* Advantage: Straight forward to implement
* Disadvantage: If a hacker gets access to the file, our security is compromised. The file is a unique point of failure

2- Use a master password encrypted hash file

* Advantage: A hacker cannot use the file even if he gets access to it.
* Disadvantage: The master password needs to be protected at all times

3- Use a password vault/manager

* Advantage: Better security and safer way to implement
* Disadvantage: requires some work to be setup

**Question 2**

**How would you change the script so that it is easy to check whether it ran successfully?**

The script can have more :

* Meaningful log message when the important steps are completed i.e. data is correctly read from API, transformed, and written to the Database.

**How would you change the script so that, if it fails, it is easy to diagnose what the problem was?**

* Two main ways :

1- Exception handling in the script: Critical conditions should be checked, and an exception raised in case of failures

2- Logging: Adding proper log messages will allow us to quickly see the execution steps

**Which ways (or tools) do you know that could be used to run scheduled jobs like this one?**

* Airflow is one of the standard orchestration and scheduling tools in the market. It offers scheduling possibilities as well as monitoring and logging to make sure the pipeline is running as expected
* A basic cron job
* Alternatively, we can use a tool like
  + Perfect: Similar to Airflow, allows scheduling and monitoring jobs

**Question 3**  
**We know that we will need to write another script soon that will access the backend API and generate some PDF reports. Since the requirements include accessing the backend API, a possible starting point would be to copy this script and modify it. Is this a good approach? What alternatives exist?**   
Copying the code is never a good idea. If the original script had bugs, that would spread around the codebase. It would be hard to fix all the occurrences of the bug and would take a much larger time.   
   
A better approach would be to build one class for accessing the API. Later on, this script and the new one can import the API class and use its functions. Every script then will have specific functions on how to handle and write the data to the desired output.

**Question 4**

**4 - How would you improve this script in order to make it suitable for a development environment with a continuous integration infrastructure? The same code should run on the developers’ machines, staging environment, production environment, and the continuous integration system. What building blocks would such an environment have? What tools would you use?**

The pipeline parameters such as the API address and keys, the Database schema, and server can be parameterized depending on the environment where the pipeline is being deployed and run in

A number of tools can be used to create multiple development environments, allow cooperation between developer teams and insure automatic deployment.

Two open-source examples come to mind:

* Jenkins
* GitLab

More solutions exist on the market for specific use cases for example cloud vendors offer their own CI/CD tools