Q1 - Monitoring

Considering that this involves important data with potential legal implications - it is important that we set the following requirements for the logging process:

1) We need to know whether jobs succeed or fail, and if they fail - which atomic operations in the database have been completed before failure.
2) We need to know some basic infrastructure data on when a job runs, so we can potentially correlate script failure with things such as cpu or storage.
3) We need to be able to see payment failures over time - both in total and per user to detect any suspicious activity as well as information about these transactions to further drill down into the cause and effect.
4) We need a historical audit trail that is trustworthy and easily queryable.
5) We need to send alerts whenever failures or suspicious activity happen.

As a general plan of attack - I would recommend the following:

- Make sure the script directly writes down its results into the DB into two new tables - one for total script results summaries - and one for each individual transaction done during execution. These tables can be linked to each other using foreign keys, as well as to other data such as user accounts.
    - This ensures a valid, trustworthy audit trail that follows data duplication, backup, restore rules etc. that are already in place
    - Since data is relationally linked, we can correlate data
    - If there are many failed transactions expected, a regular data archiving process can be set up.
- We then set up a logging framework such as the ELK stack or Grafana and connect the DB as a data source, as well as monitor the infrastructure where the script runs and the database infrastructure directly.
    - Since the DB is relational, we can easily query it and get our data in time series.
    - This allows us to aggregate data, spot trends and setup alerts as well as visualise the data for human use.

So - to answer the questions posed in the original assignment more directly:

1) **Which helpful data attributes would you extract from the cron process? (should something be added in the code?)**
   For the new database tables as output for the script, I would recommend at least the following columns:

   Table 1 - PaymentMonitorJobs
   - (PK) Id
   - Date started
   - Time spent running
   - Total results uncovered
   - Execution status (Success or failure)

   Table 2 - PaymentMonitorResults
   - (PK) Id
   - (FK) JobID
   - (FK) UserId
   - (FK) PaymentID / SubscriptionId
   - Timestamp
   - Description of the change
   - Previous State (If applicable, e.g. old ID)
   - New State (If applicable, e.g. new ID)
   - Action Type (delete, new or update)

2) **What should be monitored in the infrastructure?**
   On both the server that runs the script as the actual database server, we should gather some basic metrics such as:
   - CPU usage
   - Memory usage
   - Network usage
   - Disk I/O usage
   - Disk space usage

   This would allow us to potentially correlate this issues with script failures, or even with original payment/subscription failures.

3) **Which tools would you use in order to collect/monitor/present/notify?**
   I think you mentioned you already use Grafana, which would be good for this. Slack is also mentioned, which would be useful for sending alert notifications to.

4) **Describe the setup and the work needed in order to implement it**

- Setup the two new tables in the database
- Update the script to perform the new row inserts
- Install a Grafana setup, with agents on both the infrastructure machines
- Setup the infrastructure monitoring on Grafana
- Setup the SQL monitoring on Grafana
- Setup alerts for failures / suspicious activity.