1.
   ```
   SELECT customer_id, SUM(order_total) AS total_amountspent
   FROM orders
   WHERE order_date >= '2023-08-14'
   GROUP BY customer_id
   ORDER BY total_spent DESC
   LIMIT 5;
   ```

2. a. Critical Priority: A customer reports a system outage affecting users.
   - This will be a critical issue affecting multiple issue. This must be resolve with outmost priority and urgency. As it impacts with the service of the users it must have an immediate action.

   b. Medium Priority: A customer requests a password reset.
   - This is also important to users' productivity which they cannot access the system without it. As this is a straightforward approach where can be done quickly.

   c. Low Priority: A customer requests help with a new feature.
   - This ticket can be planned with the user, since this is a knowledge support. And it does not affect any system interruptions.

   d. Low Priority: A customer reports a minor UI bug.
   - As the issue is minor and does not affect the functionality of the system. This can be schedule for future system update.

3. First, we can check whether there are recent changes in the database like schema and indexes. Second, we can plan for the execution plan using EXPLAIN to check how the database will execute the query. Third, we can monitor and test the performance changes with different conditions. And if it is all fails maybe there are external factors, we can look to it like network latency and server maintenance.

4. On handling a recurring error in production environment, the approach will be diagnosing and resolving a recurrent or non-reproducible problem in a production to get a comprehensive date, improve logging and pinpoint environmental variations. For a permanent solution, use controlled testing conditions to replicate the problem.

5. First, from ticket creation set the status to open or new tickets, so that we can assign the ticket to the team responsible for this issue and set an appropriate priority based on the severity of the issue. Second, is to Triage and the status is in progress, which the team members start the investigation or initial diagnosis. Third, is for resolution and the status is in fix in progress if the issue is confirmed and being fix. In this it involves working on a solution. Fourth, is testing and set the status to ready for testing. Where the team ensures that there is a fix to the ticket and issue is resolved. Lastly, closure and it will set the status to closed. If there are confirmation that the issues are resolved.

6. To optimize a slow-running SQL query here are the three different approaches I will try:
    a. To speed up the searches, I will provide the relevant indexes.
    b. Improving the current indexes to determine if they are being used effectively.
    c. Trying to limit the amount of data processed by the query.

7. Using Postman to Test an API with Headers and JSON Body.
    a. Open Postman, to start a new request.
    b. Enter the API endpoint URRL and select the HTTP method.
    c. In the Headers tab, add the required headers.
    d. Add the JSON body to the body tab.
    e. Send the request and check the response.

8. To fix and resolve the intermittent connectivity issue hosted on AWS, here are the steps:
    a. Asking the user for information of the problem.
    b. Check AWS services for performance issues.
    c. Check whether there is network and connection issue.
    d. Check the application code for any mistakes.
    e. Testing and Monitoring tools to see the problem.
    f. If it is necessary, change the AWS settings.
    g. Update the user for the resolution of the problem.

9. a. SQL queries:

```
SELECT o.order_id, o.customer_id, o.order_date, o.order_total, o.status
FROM orders o
LEFT JOIN customers c ON o.customer_id = c.customer_id
WHERE o.order_date >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
AND o.status = ‘completed’;
```

There are several possible reasons why records might be missing. 1. The orders might not mark as complete. 2. There is incorrect filtering in the application and 3. There is a data sync issues between backend services and the database.

b. Support Case Study:
Investigation Plan:
1. Gathering more information from the customer to analyze the recent update.
2. Check if the server and application logs for errors or timeouts.
3. Monitoring the system and network performance during peak usage hours.
4. Check if there are recent updates for performance failure.
5. Try to reproduce the issue in a staging environment.

Immediate Solution:
1. Redo or disable the affected feature if critical.

2. For the meantime, increasing the resources to handle the load.

Long Term Solution:

1. Optimize database queries.
2. Loading and testing the application under peak conditions before future updates.
3. Implementation of better logging and monitoring for early detection of issues.