1. **innodb_lock_wait_timeout = 120** : This setting is high and may cause long transaction wait times. The suggested value is only 50 seconds.

   Sundew: innodb_lock_wait_timeout = 120
   - **Default Value:** The default value is usually 50 seconds.
   - **Potential Stalled Transactions:** Longer timeouts may lead to transactions being held up for extended periods, which can affect overall system performance and responsiveness if lock contention is high.
   - **Resource Utilization:** Extended lock waits may lead to resource contention, particularly if many transactions are waiting on locks. This can impact the performance of the entire database system.

   Sundew Reply: Now We have configured innodb_lock_wait_timeout = 50

2. **default-authentication-plugin=mysql_native_password** : This is the older authentication method. We are more concerned about Security.

   Sundew: Now We have configured default_authentication_plugin = caching_sha2_password

3. **performance-schema=0** : Disabling the performance schema improves performance but removes access to detailed performance metrics.

   Sundew: Now We have configured performance_schema = 1

4. **max_allowed_packet=268435456**: This value is very high and a high max_allowed_packet value can cause network congestion and failures. The suggested value is 67 MB or less.

   Sundew: max_allowed_packet=268435456 (256 MB).

   - **Implication:** This setting is high and can handle very large queries and data, but it may be more than necessary for many Drupal sites. Using such a high value can potentially lead to inefficient use of server resources if not needed.
   - **Default Value:** The default setting is typically 4 MB (4,194,304 bytes).
   - **Recommended Values:** For most Drupal 10 sites, a value between 64 MB (67,108,864 bytes) is often sufficient. This range balances the need for handling large data while avoiding excessively high memory usage.

   Sundew: Now We have configured max_allowed_packet = 64 MB

5. **Maximum Execution Time:** The error shows that the PHP script execution is timing out after 500 seconds (Maximum execution time of 500 seconds exceeded), which is too long. Typical maximum execution times for web applications should range between 30 to 120 seconds, depending on the complexity of the task being executed.

- Drupal Performance: The long execution time indicates that some processes or queries are taking too long to complete. This could be due to inefficiencies in database queries, large data processing, or resource-heavy tasks.
- Recommendations for Optimizing Performance:
- Reduce the max_execution_time. For most Drupal websites, a typical setting is 30 to 120 seconds, and it should not exceed 120 seconds. Currently, the website is running with a 30-second setting.

Sundew: Based on your suggestion here's the updated configuration what we followed:

max_execution_time = 120 seconds

innodb_lock_wait_timeout = 50

max_allowed_packet = 64 MB

performance_schema = 1

innodb_lock_wait_timeout = 50

After configuring the suggested settings by the IITK IT Team, we proceed to upload the database both on the AWS server and localhost. Based on our tests, the database upload is taking approximately 8 minutes on localhost and 9 minutes on the server.

Please find our response time

Local Server: https://drive.google.com/file/d/1r0iP8fT-6BSjYDd9XNVposamEDnhXssk/view

AWS server: https://drive.google.com/file/d/1xc0cmaI1mq5QrAh4fr0dKiJwwLEc24yV/view

We understand that performance can vary slightly depending on the environment, and we are providing a detailed video demonstration to give further insights into the process.

6. A detailed review of the recommendations and additional insights for managing BLOB data in Drupal 10 and the key areas where optimization can be improved:

   Sundew: The data storage of workflow is controlled by Drupal Core's Entity API. Sundew Team did not implement any custom SQL insert or update operations, instead, all data persistence, including BLOB handling, is managed natively through Drupal's core entity and field storage mechanisms. Altar of the core settings is not recommended as it may impact the Database performance.

   Existing Recommendations (BY SUNDEW):

1. **Move Search Indexing to Solr**:

- **Action**: Offloading search indexing from the database to Solr can significantly reduce the size of indexing tables and improve search performance.
- **Optimization**: This can be a major performance boost if your search queries involve complex filtering or if the database is handling high concurrency.
- **Ideal Blob Size Consideration**: The BLOB data for search indexing would not be significant post-Solr integration, but any stored search-related data will now reside on Solr, which reduces the BLOB size in the database.
- **Note**: In case, SOLR gives any issues or malfunctions, then the previous issues that were shared should not repeat. No junk data should be displayed.

Sundew: The website will work as intended as long as the necessary modules, settings, and configurations are left unchanged.

2. **Manage Watchdog Table (Logging Data)**:
   - **Action**: Limit the number of logs stored in the watchdog table and implement log rotation. Housekeeping scripts can regularly clean old logs to prevent table growth.
   - **Optimization**: This is crucial as logging tables like watchdog can grow quickly and lead to performance degradation. Logging overhead should be offloaded or stored in a separate system (e.g., using external logging tools like ELK Stack or syslog).
   - **BLOB Data Size**: The current size of watchdog (472.5 MiB) indicates a significant volume of data. Periodically truncate or archive old logs, aiming to keep this table size under control (~100-200 MiB).

Sundew: The 'watchdog' table stores user activity logs. To optimize performance, the admin can implement a housekeeping strategy by retaining only critical log data and discarding less important entries. By configuring a log retention policy, the admin can limit the number of stored logs, reducing the database load while ensuring that only relevant data is retained for reference. This approach will enhance overall system performance without overwhelming the database with excessive or unnecessary log data.

Drupal already handles its own logging (e.g., through the `watchdog` table), by limiting no of log records or by housekeeping activity Admin can control the DB size. ELK stack (Elasticsearch, Logstash, Kibana) can be a powerful tool for logging and analytics, with several limitations when integrating it with Drupal:

- Setting up and maintaining ELK for a Drupal site adds complexity, requiring additional infrastructure and expertise in Elasticsearch, Logstash, and Kibana.
- Managing log storage, scaling Elasticsearch clusters, and handling node failures can be resource-intensive.
- ELK consumes significant system resources (CPU, RAM, and storage). For Drupal sites hosted on modest infrastructure, the overhead from ELK can negatively impact site performance.

- ELK does not have native Drupal support. Custom modules or additional configurations may be required to send log data from Drupal to Logstash, adding development overhead.
- Without proper management, this can result in redundant data, increased database size, and unnecessary resource usage.
- Managing user roles and permissions for Kibana dashboards may also introduce additional layers of complexity.

The IITK IT team already highlighted concerns regarding adding too many dependencies for the website, and integrating ELK would introduce multiple new dependencies (Elasticsearch, Logstash, and Kibana), which may conflict with existing Drupal dependencies or increase the system's complexity. The introduction of ELK is not recommended by the Sundew team

In addition, we can introduce syslogs to generate logs outside the database. These logs would be stored in the "/var/log/syslog" file, which is a more robust solution for handling larger volumes of logs. **However, we will need to ensure that the system has proper write permissions to this directory so that the logs can be written successfully**.

3. **File Storage Optimization:**
   - Action: Move BLOB data for large files (e.g., images, videos) to the file system instead of storing them in the database. Drupal supports file storage outside of the database, reducing the burden on the database engine and improving performance. -> SUNDEW will explain more regarding this
   - Optimization: Storing large media files as BLOBs in the database can cause bloating and slower access times. Ensure that large files are stored using Drupal's file system and only the metadata (e.g., file references) is stored in the database.
   - Ideal BLOB Size: For BLOB data stored in the database (if required), consider capping the ideal size at 50-100 MiB per table to prevent performance bottlenecks.

Sundew: The data storage of workflow is controlled by Drupal Core's Entity API. Sundew Team did not implement any custom SQL insert or update operations, instead, all data persistence, including BLOB handling, is managed natively through Drupal's core entity and field storage mechanisms. Altar of the core settings is not recommended as it may impact the Database performance.

3. **Batch Table**:
   - **Action**: The batch table (currently 12.5 MiB) can accumulate data if batch processes are run frequently. Clearing old batch records will keep this table's size manageable.
   - **Optimization**: Regular maintenance scripts can keep the table clean.
   - **BLOB Data Size**: Ideally, keep the batch table under 5 MiB with regular housekeeping.

Sundew: The batch table in Drupal stores serialized data related to the execution and progress of batch processes.It is critical to recognize that Drupal Core governs database management, including the batch table, through its structured batch processing subsystem. This subsystem handles queueing, state management, and the execution of long-running processes. Direct intervention to truncate or purge batch records risks disrupting the underlying batch processing lifecycle, potentially corrupting incomplete operations or triggering unforeseen state inconsistencies. Such actions could compromise system integrity, leading to site instability. Proper maintenance should leverage Drupal's built-in housekeeping mechanisms to mitigate risks while preserving system coherence.

Ideal BLOB Size:

- **Watchdog Logs**: Regular rotation to keep this table under 100-200 MiB.
- **Other Tables (e.g., key_value, router)**: Ideally, these tables should be small, usually below 10 MiB.

Sundew: The data storage of workflow is controlled by Drupal Core’s Entity API. Sundew Team did not implement any custom SQL insert or update operations, instead, all data persistence, including BLOB handling, is managed natively through Drupal's core entity and field storage mechanisms. Altar of the core settings is not recommended as it may impact the Database performance.

In addition, we can introduce syslogs to generate logs outside the database. These logs would be stored in the “/var/log/syslog” file, which is a more robust solution for handling larger volumes of logs. **However, we will need to ensure that the system has proper write permissions to this directory so that the logs can be written successfully**.