ChatGPT

Email Automation Schedule Setup (phpMyAdmin/WordPress)

This guide outlines how a solopreneur can automate the delivery of the **Bloating Myth-Busting Email Course** using WordPress with phpMyAdmin for database management. It leverages WordPress's built-in cron system and an SMTP plugin for reliable email delivery.

1. Configure Email Deliverability

WordPress uses PHP's `mail()` function by default, which is basic and not authenticated. Emails sent via this function are more likely to be marked as spam and may not reach recipients [1]. To improve deliverability, install and configure the **WP Mail SMTP** plugin. This plugin routes WordPress emails through a third-party SMTP service (e.g., SendLayer, Gmail, or SMTP.com) to ensure your messages are authenticated and reach the inbox [2].

2. Create Database Tables

In phpMyAdmin, create two tables:

- `users` – stores subscriber data (user_id, name, email, signup_date).
- `course_schedule` – stores course send schedule (id, user_id, course_id, day_number, scheduled_send_datetime, sent_status). The `course_id` field allows you to reuse the same table for multiple nurture sequences.

Populate `course_schedule` when a user signs up: insert five rows, one for each email day, with scheduled send dates computed relative to the signup time (e.g., signup + 1 day, signup + 2 days, etc.).

3. Set Up WordPress Cron Events

WordPress has a built-in cron system for scheduled tasks. A tutorial on automatic email sending explains that you need to:

1. **Schedule a cron event** that runs at regular intervals (e.g., every hour) [3].
2. **Query the database** during each run to find schedule rows where `sent_status = 0` and `scheduled_send_datetime <= now()` [4].
3. **Check for duplicates:** Ensure each email is sent only once by updating `sent_status` immediately after sending [3].
4. **Send the email** using `wp_mail()` (or a mailing library) with the appropriate template and personalized data [5].
5. **Log the send** by setting `sent_status = 1` and recording the sent timestamp, so future cron runs don't resend the same email [6].

Here is a simplified PHP snippet to schedule and process the cron event:

```
// Schedule the cron event when plugin/theme is activated
function fnm_schedule_email_cron() {
  if (!wp_next_scheduled('fnm_send_course_emails')) {
    wp_schedule_event(time(), 'hourly', 'fnm_send_course_emails');
  }
}
register_activation_hook(__FILE__, 'fnm_schedule_email_cron');

// Cron callback: send pending course emails
function fnm_send_course_emails() {
  global $wpdb;
  $table_name = $wpdb->prefix . 'course_schedule';
  $rows = $wpdb->get_results( $wpdb->prepare(
    "SELECT * FROM $table_name WHERE sent_status = %d AND
scheduled_send_datetime <= NOW()",
    0
  ) );
  foreach ($rows as $row) {
    $user = get_userdata($row->user_id);
    $email_content = fnm_get_email_template($row->day_number, $user);
    wp_mail($user->user_email, $email_content['subject'],
$email_content['body']);
    // mark as sent
    $wpdb->update( $table_name, [ 'sent_status' => 1, 'sent_timestamp' =>
current_time('mysql') ], [ 'id' => $row->id ] );
  }
}
add_action('fnm_send_course_emails', 'fnm_send_course_emails');
```

4. Manage Templates

Define a function (e.g., `fnm_get_email_template($day, $user)`) that returns the subject and body for each day. Pull content from the JSON file generated for the course to keep templates separate from logic. Use placeholders like `[FirstName]` to personalize the salutation.

5. Scale for Multiple Courses

The presence of a `course_id` column in `course_schedule` allows you to schedule different courses simultaneously. When a new course is added, insert rows with the appropriate `course_id` and email count. Your cron callback should select rows based on both `course_id` and unsent status.

6. Testing & Monitoring

- Use a plugin such as **WP Crontrol** to view and manually trigger scheduled cron events during testing [7].

- Install an email logging plugin to record outgoing messages and verify that emails are sent as expected [7].
- Always test with a small group before rolling out to all subscribers. Add error logging in your cron callback so you can diagnose issues without sending unintended emails [8].

By combining phpMyAdmin for scheduling, WordPress cron for automation, and a reliable SMTP service, a solopreneur can deliver a professional, scalable nurture sequence without manual intervention.

---

[1] [2] How to Send Automated Email in WordPress
https://wpmailsmtp.com/how-to-send-automated-email-in-wordpress/

[3] [4] [5] [6] [7] [8] Automatically Sending an Email using the WordPress Cron – xnau webdesign
https://xnau.com/automatically-sending-an-email-using-the-wordpress-cron/