PART1: PHP and CodIgniter

1. Advanced PHP Concepts:

Dependency Injection (DI) is a design pattern that allows the injection of the dependencies required by an object rather than creating them within the object itself. In the context of CodeIgniter, here's an example:
Let's suppose we have a controller in CodeIgniter that handles sending emails. Instead of directly instantiating the email-sending class within the controller, we could use DI to inject an instance of the email-sending class through its constructor. This makes the code more flexible because you can easily change the implementation of the email-sending class without modifying the controller itself.
Simplified example:

```php
class EmailController extends CI_Controller {
    private $emailSender;

    public function __construct(EmailSender $emailSender) {
        parent::__construct();
        $this->emailSender = $emailSender;
    }

    public function sendEmail() {
        // Use $this->emailSender to send the email
    }
}
```

2. CodeIgniter Specifics:

CodeIgniter manages sessions using its built-in Session class.
Database Sessions:
Pros: Scalability, Data Persistence, Security
Cons: Complexity, Potential Performance Overhead
File-Based Sessions:
Pros: Simplicity, Performance
Cons: Limited Scalability, Limited Data Sharing

3. Database Interactions in CodeIgniter:

- **Identify Slow Query:** Check server logs or use database profiling to find the slow query.
- **Analyze Query Execution:** Use EXPLAIN or a similar tool to analyze the query execution plan.
- **Optimize the Query:** Rewrite the query to use appropriate indexes and reduce complexity.
- **Optimize Database Configuration:** Adjust database server settings for better performance.
- **Cache Results:**Implement caching mechanisms to reduce the need for repetitive queries.
- **Monitor Performance:** Continuously monitor query performance and adjust as needed.

4. Architecture and Design:

One scenario where we would implement Domain-Driven Design (DDD) is in a CodeIgniter e-commerce application to manage product inventory, pricing, and order processing. This approach ensures a clear separation of business logic, enhancing maintainability and flexibility.