**NOTES**

To apply Equivalence Class Partitioning (ECP), Boundary Value Analysis (BVA), and Risk Analysis to the testing of the dashboard module of the MetaBase Web application, we'll break down the testing requirements into different scenarios and identify potential risks. Here's a summary of what you should consider for each technique:

**Equivalence Class Partitioning (ECP):** ECP is used to identify different groups or classes of input values and ensure that you test representative values from each class.

**Login Page Functionality:**

**ECP:** Valid username and password, Invalid username, Invalid password, Empty username, Empty password.

Test for both successful and unsuccessful login attempts.

**Filtering Functionality:**

**ECP:** Valid date range, Invalid date range, Valid category, Invalid category.

Test different filter combinations, including valid and invalid inputs.

**User Preferences:**

**ECP:** Valid theme selection, Valid language selection, Invalid theme, Invalid language.

Test different preference combinations, ensuring that valid choices are applied and invalid choices are rejected.

**Boundary Value Analysis (BVA):** BVA helps identify the edge conditions or boundaries of the input values, where issues are likely to occur.

**Login Page Functionality:**

**BVA:** Test the minimum and maximum allowed characters for the username and password.

Ensure the system handles these boundary values correctly.

**Data Export Feature:**

**BVA:** Test exporting a minimal dataset and an exceptionally large dataset.

Ensure the export process can handle both extremes.

**Risk Analysis**: Risk analysis helps identify areas of higher risk and prioritize testing efforts.

Error Handling:

**Risk:** Input validation issues, such as SQL injection, cross-site scripting, and other security vulnerabilities.

Prioritize testing for these high-risk areas to ensure the application's security.

**Performance and Scalability:**

**Risk:** System performance degrading with a large dataset or simultaneous users.

Test under load and scalability scenarios to identify and mitigate performance risks.

**Rate Limiting:**

**Risk:** Overloading the API due to excessive requests.

Test the API to ensure rate limiting is effective in pareves.