SQL Queries for QA & Sprint Insights Dashboard

MISSED ISSUES QA RATIO (FEBRUARY 2024)

QUERY-6

Calculate the ratio of missed bugs to total bugs per project/phase for specific February 2024 sprints. Final Output Fields: phase, project_name, total_bugs, missed_bugs, missed_bugs_ratio

```
WITH BaseBugData AS (
    -- Fetches relevant bug details for specific Feb 2024 sprints and active
projects
    SELECT
        i.`key`,
        i.project_id,
        p.name AS project_name,
        s.name AS sprint_name,
        i.user_id,
        CASE
            WHEN s.name LIKE '%design%' THEN 'design'
            WHEN s.name LIKE '%beta%' OR s.name LIKE '%api%' THEN 'beta'
            WHEN s.name LIKE '%alpha%' THEN 'alpha'
            WHEN s.name LIKE '%UAT%' THEN 'UAT'
            ELSE 'NA'
        END AS phase
    FROM issues AS i
    LEFT JOIN projects AS p ON p.id = i.project_id
    LEFT JOIN sprints AS s ON s.id = i.sprint_id
    WHERE
        s.start_date BETWEEN '2024-02-01' AND '2024-02-29' -- February 2024
filter
        AND p.is_active = 1
        AND i.deleted_at IS NULL
        AND i.issue_type = 'bug'
        AND i.status NOT LIKE '%cancelled%'
        AND (s.name LIKE '%CF%' OR s.name LIKE '%CR%' OR s.name LIKE
'%fixes%') -- Specific sprint types
        AND p.name NOT IN ('Code review', 'Pre-Sales Design Projects',
'Internal CRM Tekrevol', '...') -- Excluded projects),

QATeamAssignment AS (
    -- Assigns QA team based on user ID and determines if bug was 'missed' by
QA
    SELECT
        bbd.phase,
        bbd.project_id,
```

```
        bbd.project_name,
        bbd.`key`,
        -- Assigns a 'QA Team' label based on the user associated with the bug
        CASE
            WHEN u.name IN ('Aamir Saleem', 'Owais Uddin Ahmed', '...') THEN
'Aamir Team'
            WHEN u.name IN ('Adeel Sabzali', 'Bebark Sajjad', '...') THEN
'Adeel Team'
            -- Add all other team assignments here...
            ELSE 'NA' -- Bugs not assigned to a user in these lists are
implicitly 'missed' if they fall here
        END AS qa_team_assigned -- This column helps identify bugs potentially
handled by QA
    FROM BaseBugData bbd
    LEFT JOIN users AS u ON u.id = bbd.user_id -- Join to get user name for
team assignment),

TotalBugsPerPhase AS (
    -- Calculates the total number of bugs per project and phase
    SELECT
        i.project_id,
        CASE
            WHEN s.name LIKE '%design%' THEN 'design'
            WHEN s.name LIKE '%beta%' OR s.name LIKE '%api%' THEN 'beta'
            WHEN s.name LIKE '%alpha%' THEN 'alpha'
            WHEN s.name LIKE '%UAT%' THEN 'UAT'
            ELSE 'NA'
        END AS phase,
        COUNT(DISTINCT i.`key`) AS total_bugs
    FROM issues AS i
    LEFT JOIN projects AS p ON p.id = i.project_id
    LEFT JOIN sprints AS s ON s.id = i.sprint_id
    WHERE
        p.is_active = 1
        AND i.deleted_at IS NULL
        AND i.issue_type = 'bug'
        AND i.status NOT LIKE '%cancelled%'
        -- Re-apply relevant date/sprint filters if necessary to match the
scope of BaseBugData
        AND s.start_date BETWEEN '2024-02-01' AND '2024-02-29'
        AND (s.name LIKE '%CF%' OR s.name LIKE '%CR%' OR s.name LIKE
'%fixes%')
        AND p.name NOT IN ('Code review', 'Pre-Sales Design Projects',
'Internal CRM Tekrevol', '...')
    GROUP BY i.project_id, phase)

-- Final Calculation: Join assigned/missed bugs with total counts and
calculate ratio
```

```
SELECT
    qta.phase,
    qta.project_name,
    COALESCE(tbp.total_bugs, 0) AS total_bugs,
    -- Count bugs where QA team assignment is 'NA' as 'missed'
    COUNT(DISTINCT CASE WHEN qta.qa_team_assigned = 'NA' THEN qta.`key` END)
AS missed_bugs,
    -- Calculate ratio, avoiding division by zero
    (COUNT(DISTINCT CASE WHEN qta.qa_team_assigned = 'NA' THEN qta.`key` END)
* 100.0 / NULLIF(tbp.total_bugs, 0)) AS missed_bugs_ratio
FROM QATeamAssignment qta
LEFT JOIN TotalBugsPerPhase tbp ON qta.project_id = tbp.project_id AND
qta.phase = tbp.phase
GROUP BY qta.phase, qta.project_name, tbp.total_bugs
ORDER BY qta.project_name, qta.phase;
```

PROJECT START AND END DATES (BASED ON WORKLOGS)

QUERY-7:

Determine the earliest start and latest end date for each project based on worklog entries across relevant sprints.

Final Output Fields: project_id, project_name, project_start_date, project_end_date.

```
WITH ProjectSprintIssues AS (
    -- Selects relevant issues from active, non-excluded projects and sprints
    SELECT
        i.`key`,
        p.id AS project_id,
        p.name AS project_name,
        i.sprint_id,
        s.name AS sprint_name
    FROM issues AS i
    JOIN projects AS p ON p.id = i.project_id
    JOIN sprints AS s ON s.id = i.sprint_id
    WHERE
        p.name NOT IN (
            'Code review', 'Code Review Manual', 'Complete Focus - CA',
            'Grahamity - (discarded)', 'Internal CRM Tekrevol',
            'Learning (discarded)', 'Marketing Team (MR)',
            'Pre-Sales Design Projects', 'ProjectReview',
            'Revol Planner - Tekrevol', 'ServeEasy - CA',
            'ServeEasy Marketing Wing', 'Syllable Learning (discarded)',
            'Tekrevol 2.0', 'test', 'Business Analysis (BA)', 'ML/AI RCD')
        AND p.name NOT LIKE '%Marketing-TekRevol%'
        AND p.is_active = 1
        AND i.deleted_at IS NULL
        AND p.name IS NOT NULL
```

```
        AND s.name IS NOT NULL
        AND i.status NOT LIKE '%cancelled%'
        AND s.name NOT LIKE '%general%'
        AND i.issue_type IN ('Task','Suggestion','Bug','Epic','Story','Sub-
task')),

SprintWorklogBounds AS (
    -- Finds the min and max worklog creation timestamp for each sprint
    SELECT
        s.id AS sprint_id,
        MIN(w.created) AS sprint_worklog_start,
        MAX(w.created) AS sprint_worklog_end
    FROM sprints AS s
    LEFT JOIN worklogs AS w ON w.sprint_id = s.id
    WHERE
        s.deleted_at IS NULL
        AND w.deleted_at IS NULL
        AND w.created IS NOT NULL
    GROUP BY s.id)

-- Final Aggregation: Group by project and find overall min/max worklog dates
from associated sprints
SELECT
    psi.project_id,
    psi.project_name,
    MIN(swb.sprint_worklog_start) AS project_start_date,
    MAX(swb.sprint_worklog_end) AS project_end_date
FROM ProjectSprintIssues psi
JOIN SprintWorklogBounds swb ON psi.sprint_id = swb.sprint_id
GROUP BY psi.project_id, psi.project_name
ORDER BY psi.project_name;

-- SPRINT COMPLETION STATUS QUERY-8:
-- Summarize each sprint, showing total tasks, resolved tasks, and overall
completion status
-- Final Output Fields: sprint_id, sprint_info, total_tasks, Resolved,
Sprint_status

WITH SprintTaskStatus AS (
    -- Extracts tasks per sprint and classifies their status as 'Done' or
'Pending'
    SELECT
        i.`key` AS task_key,
        i.sprint_id,
        p.name AS project_name,
        s.name AS sprint_name,
        CASE
```

```
            WHEN i.status IN ('Done', 'Verified', 'Resolved', 'Development
Complete') THEN 'Done'
            ELSE 'Pending'
        END AS task_status
    FROM issues AS i
    JOIN projects AS p ON p.id = i.project_id
    JOIN sprints AS s ON s.id = i.sprint_id
    WHERE
        p.name IS NOT NULL
        AND s.name IS NOT NULL
        AND i.status NOT LIKE '%cancelled%'
        AND i.issue_type IN ('Task','Suggestion','Bug','Epic','Story','Sub-
task')
        AND p.is_active = 1
        AND i.deleted_at IS NULL),

SprintSummary AS (
    -- Aggregates task counts per sprint
    SELECT
        sprint_id,
        project_name,
        sprint_name,
        COUNT(DISTINCT task_key) AS total_tasks,
        COUNT(DISTINCT CASE WHEN task_status = 'Done' THEN task_key END) AS
resolved_tasks
    FROM SprintTaskStatus
    GROUP BY sprint_id, project_name, sprint_name)

-- Final Output: Format sprint info and determine completion status
SELECT
    ss.sprint_id,
    CONCAT(ss.project_name, ' (', ss.sprint_name, ')') AS sprint_info,
    ss.total_tasks,
    ss.resolved_tasks AS Resolved,
    CASE
        WHEN ss.total_tasks = ss.resolved_tasks THEN 'Completed'
        ELSE 'Incomplete'
    END AS Sprint_status
FROM SprintSummary ss
ORDER BY ss.total_tasks DESC, sprint_info;
```