Group 12 - Stage 3 Report

Database Systems 1 - DBMS I8Z01

Submission Date: 28/04/2025 **Student Name:** Cole Cumiskey

Student ID: D00281856

Section 1: Final ER Diagram (Mermaid.js)

```
erDiagram
    GAME ||--o{ BUG : has
    DEVELOPER } | --o{ BUGASSIGNMENT : works_on
    BUG ||--o{ BUGASSIGNMENT : assigned_to
    GAME {
        int game_id PK
        string title
        string genre
        string release_version
    }
    BUG {
        int bug_id PK
        string title
        string description
        string severity
        string status
        date reported_date
        int game_id FK
    }
    DEVELOPER {
        int developer_id PK
        string name
        string role
        string email
    }
    BUGASSIGNMENT {
        int bug_id PK, FK
        int developer_id PK, FK
        date assigned_date
    }
```

Section 2: SQL Queries

Query 1: List bugs and assigned developers

```
SELECT b.bug_id, b.title, d.name
FROM bug b
JOIN bug_assignment ba ON b.bug_id = ba.bug_id
JOIN developer d ON ba.developer_id = d.developer_id;
```

Display each bug and the name of developers assigned to fix it.

Output:

Enter the number of bug_id	the query to run: 1 title	name
301	Crash on startup	Alice Johnson
301	Crash on startup	Bob Smith
302	Missing textures	Devid Zhang
303	Audio delay	Carol Lee
304	Stuck in wall	Emily Fox
305	Login failure	Frank White
306	AI pathing error	Grace Kim
307	Leaderboard not updatingHenry Brown	
308	Invisible character	modelIvy Davis
309	Level does not load Jack Lee	
310	Incorrect score calculationKatie Morgan	
311	Multiplayer sync issueLeo Walker	
312	Settings reset	Mia Torres
313	Achievements not unlockingNathan Grey	
314	Incorrect dialogue	Olivia Reed
315	Game freeze after s	utsceneBob Smith
315	Game freeze after s	utsceneDevid Zhang

Query 2: Count of bugs per game

```
SELECT g.title, COUNT(b.bug_id)
FROM game g
JOIN bug b ON g.game_id = b.game_id
GROUP BY g.title;
```

Description:

Shows the number of bugs associated with each game.

Output:

Enter the number of title	the query to run: 2 COUNT(b.bug_id)
Alien Invasion	1
Astro Miner	1
Castle Siege	1
City Bulder Pro	1
Dark Realms	1
Escape Room	1
Fantasy Quest	1
Mech Battles	1
Monster Breach	1
Ocean Dive	1
Robo Rush	1
Sky Kingdoms	1
Space Explorer	1
Speed Fury	1
Zombie Horde	1

Query 3: Count of bugs grouped by severity

```
SELECT severity, COUNT(*) AS total
FROM bug
GROUP BY severity;
```

Aggregates bugs by severity level (Low, Medium, High).

Output:

Query 4: Developers assigned to the most recent bug

```
SELECT d.name, b.title
FROM bug_assignment ba
JOIN developer d ON ba.developer_id = d.developer_id
JOIN bug b ON ba.bug_id = b.bug_id
WHERE b.reported_date =
(
    SELECT MAX(reported_date) FROM bug
);
```

Description:

Lists developers assigned to the newest reported bug.

Output:

```
Enter the number of the query to run: 4
name title
Bob Smith Game freeze after sutscene
Devid Zhang Game freeze after sutscene
```

Query 5: Show bugs with readable severity levels (CASE)

```
SELECT title,

CASE severity

WHEN 'Low' THEN 'Minor'

WHEN 'Medium' THEN 'Moderate'

WHEN 'High' THEN 'Critical'

END AS severity_level

FROM bug;
```

Description:

Translates severity into more descriptive terms using CASE.

Output:

```
Enter the number of the query to run: 5
           severity_level
title
Crash on startup Critical
Missing textures Moderate
Audio delay Minor
Stuck in wall Critical
Login failure Critical
                   Minor
Audio delay
AI pathing error Moderate
Leaderboard not updatingMinor
Invisible character modelCritical
Level does not load Critical
Incorrect score calculationModerate
Multiplayer sync issueCritical
Settings reset
                    Minor
Achievements not unlockingModerate
Incorrect dialogue Minor
Game freeze after sutsceneCritical
```

Query 6: Bugs containing the word 'load' in the title

```
SELECT * FROM bug
WHERE title LIKE '%load%';
```

Searches bug titles for the keyword "load".

Output:



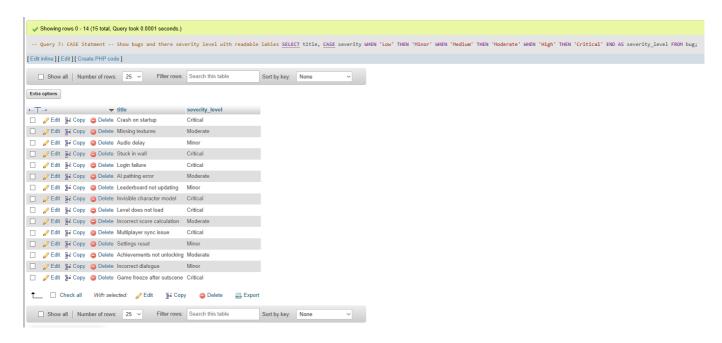
Query 7: Count games with more than one bug (GROUP BY + HAVING)

```
SELECT g.title AS game_title, COUNT(b.bug_id) AS bug_count
FROM game g
JOIN bug b ON g.game_id = b.game_id
GROUP BY g.title
HAVING COUNT(b.bug_id) > 1;
```

Description:

Filters to only show games that have more than one bug report.

Output:



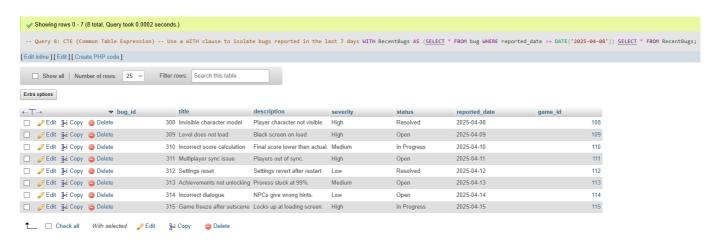
Query 8: View bugs reported in the last 7 days (CTE)

```
WITH RecentBugs AS
(
    SELECT * FROM bug
    WHERE reported_date >= DATE('2025-04-08')
```

```
)
SELECT * FROM RecentBugs;
```

A CTE (Common Table Expression) used to isolate recent bug reports.

Output:



Query 9: Create and use a view for bug details

```
-- View creation (run once)

CREATE VIEW BugDetails AS

SELECT b.bug_id, b.title, g.title AS game_title, d.name AS developer_name

FROM bug b

JOIN game g ON b.game_id = g.game_id

JOIN bug_assignment ba ON b.bug_id = ba.bug_id

JOIN developer d ON ba.developer_id = d.developer_id;

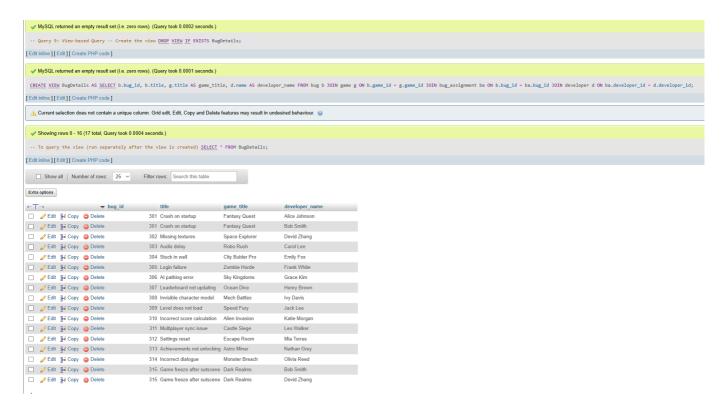
-- Use the view

SELECT * FROM BugDetails;
```

Description:

View simplifies access to full bug details with game and developer info.

Output:



Query 10: Subquery to show high-severity bugs only

```
SELECT * FROM bug
WHERE severity = 'High'
AND bug_id IN
(
    SELECT bug_id FROM bug WHERE severity = 'High'
);
```

Uses a subquery to filter for critical (high severity) bugs.

Output:



Section 3: Individual Contribution

| Task / Component | Cole Cumiskey | SQL Schema Design (schema.sql) | Done | Sample Data Creation (populate.sql) | Done

| SQL Query Development (queries.sql) | Done

| SQLite Database Creation | Done

| Java JDBC Application (Main, DB* classes) | Done

| Java Query Execution | Done

| Mermaid ER Diagram | Done

| Markdown Report / PDF Compilation | Done

| Screenshot Capture for Report | Done

| Testing: PHPMyAdmin, DB Browser, Java | Done

| ZIP Folder Preparation | Done

Section 4: Tools & References

Tool / Technology | Purpose

Visual Studio Code | Used as the main editor for writing SQL, Java, and Markdown

SQLite JDBC Driver | Java library to connect to SQLite databases

DB Browser for SQLite | Used to create, edit, and verify the final database.sqlite file

PHPMyAdmin (via XAMPP) | Used for initial schema and query testing in MySQL

Mermaid.js | Used to generate the ER diagram using Mermaid markdown syntax

Java SDK (OpenJDK 11) | Compiling and running the Java application

Markdown PDF Exporter | Used to convert the report from Markdown to PDF

Third-Party Content Declaration

As per Dkit policy, the following tools were used with transparency:

Stack Overflow (referenced, Not copy-pasted) Used fpr understading JDBC error messages and configuration examples.

Dkit Moodle (Not copy-pasted) Used for understanding how to use PHPmyAdmin and XAMPP

References

- Elmasri & Navath (2015). Fundamentals of Database Systems (7th Ed.)
- GeeksforGeeks SQL Normalization Articles https://www.geeksforgeeks.org/normalization-in-dbms/
- SQLite JDBC Driver https://github.com/xerial/sqlite-jdbc
- Mermaid.js ER Diagram Documentation https://mermaid.js.org/syntax/entityRelationshipDiagram.html