3 NF Justification Report

Functional Dependencies of the Database

Developer

Schema: developer(developer_id, developer_name)
FD: developer_id → developer_name
Primary Key: developer_id
Justification: Attribute depends fully on the key; no transitive dependencies → 3NF.

Publisher

Schema: publisher(publisher_id, publisher_name)
FD: publisher_id → publisher_name
Primary Key: publisher_id
Justification: No partial or transitive dependencies → 3NF.

Genre

Schema: genre(genre_id, genre_name)
FD: genre_id → genre_name
Primary Key: genre_id
Justification: Fully dependent on key → 3NF.

Platform

Schema: platform(platform_id, platform_name)
FD: platform_id → platform_name
Primary Key: platform_id
Justification: Only one attribute depends on the key → 3NF.

Game

Schema: game(game_id, name, year_of_release, developer_id, genre_id, platform_id, publisher_id)
FD:

- game_id → name, year_of_release, developer_id, genre_id, platform_id, publisher_id

Primary Key: game_id

Normalization Rationale:

- All attributes depend entirely on game_id.
- Non-key fields (developer_name, genre_name, etc.) are not stored here, avoiding transitive dependencies.
- Therefore, the Game table satisfies 3NF.

Sales

Schema: sales(sales_id, game_id, global_sales, na_sales, eu_sales, jp_sales, other_sales)
FDs:

- sales_id → game_id, global_sales, na_sales, eu_sales, jp_sales, other_sales
- game_id → global_sales, na_sales, eu_sales, jp_sales, other_sales (due to 1-to-1 relationship)

Primary Key: sales_id
Justification: Each sales metric is functionally dependent on sales_id (or game_id). No transitive or partial dependencies → 3NF.

Reviews

Schema: reviews(review_id, game_id, critic_score, critic_count, user_score, user_count)
FDs:

- review_id → game_id, critic_score, critic_count, user_score, user_count
- game_id → critic_score, critic_count, user_score, user_count (1-to-1 relationship)

Primary Key: review_id
Justification: All attributes depend fully on the primary key → 3NF.

Design Choices & Why the Schema Meets 3NF

The database schema is intentionally decomposed into separate entities (Developer, Publisher, Genre, Platform, Game, Sales, Reviews) to ensure:

No Partial Dependencies

Every table's primary key is a single column, so partial dependencies cannot exist.

No Transitive Dependencies

Attributes such as developer_name, genre_name, platform_name, publisher_name are stored only in their respective dimension tables, not inside the game table. This prevents chains like:

- game → developer_id → developer_name

Each Table Represents One Concept

This avoids mixing information and ensures clarity in relationships.

Foreign Keys Maintain Integrity

All child tables (game, sales, reviews) reference parent dimension tables correctly.

How the Schema Avoids Data Anomalies

1. Update Anomalies Prevented

- Changing a publisher name updates exactly one row in the publisher table.
- No duplicate descriptive data exists across multiple tables.

2. Insert Anomalies Prevented

- A game cannot be inserted with an invalid developer, genre, or platform.
- Sales/Reviews cannot be inserted without a valid game.

3. Delete Anomalies Prevented

- Deleting a sales record does not delete the game.
- Deleting a game will delete sales/reviews only if cascading is explicitly enabled.

Conclusion

The designed schema:

- Is fully normalized to Third Normal Form (3NF).
- Eliminates redundancy and ensures high data quality.
- Avoids all major data anomalies (update, insert, delete).
- Supports efficient analytics and future scalability.

Github Repo Link : https://github.com/prajesh-1003/Video_Game_Sales_and_Ratings_Analytics