📄 Database Normalization and Denormalization Report

Objective

The purpose of this report is to:

- Understand and apply normalization techniques (1NF, 2NF, 3NF) to real-world data scenarios.
- Eliminate data redundancy and improve data integrity.
- Provide clear step-by-step examples for each form.
- Describe all types of normalization up to 5NF.
- Explore the concept of de-normalization—its purpose, advantages, and appropriate use cases.

What is Normalization?

Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves dividing large, complex tables into smaller, manageable ones using rules designed to ensure that each table contains only data related to a single subject.

Types of Normalization

🔹 First Normal Form (1NF)

Definition:

- A relation is in 1NF if it only contains atomic (indivisible) values.
- No repeating groups or arrays.

Use Case:
Used when raw data contains multiple values in a single field or repeating columns.

Example:

Unnormalized Table:

CustomerID	Name	Phone Numbers
1	Ali	9988776655 , 8899001122
2	Ahmed	7766554433

After 1NF:

CustomerID	Name	Phone Numbers
1	Ali	9988776655
1	Ali	8899001122
2	Ahmed	7766554433

🔹 Second Normal Form (2NF)

Definition:

- Must be in 1NF.
- Removes **partial dependencies** (i.e., non-key attributes must depend on the whole primary key).

Use Case:
Applied when the table has a **composite primary key** and non-key attributes depend only on part of it.

Example:

1NF Table:

StudentID	**CourseID**	**StudentName**	**CourseName**
1	CS101	Ali	DB
1	CS101	Ahmed	DB

After 2NF:
Students Table:

StudentID	**StudentName**
1	Ali
1	Ahmed

Courses Table:

CourseID	**CourseName**
CS101	DB
CS101	DB

Enrollment Table:

StudentID	CourseID
1	CS101
1	CS101

🔹 Third Normal Form (3NF)

Definition:

- Must be in 2NF.
- Removes **transitive dependencies** (non-key fields should not depend on other non-key fields).

Use Case:
When non-key attributes are indirectly dependent on the primary key through another non-key attribute.

Example:

2NF Table:

EmployeeID	DeptID	Name	DeptName
1	101	Salim	HR
1	102	Ali	IT

After 3NF:

Employee Table:

EmployeeID	DeptID	Name
1	101	Salim
1	102	Ali

Department Table:

DeptID	DeptName
101	HR
102	IT

✅ Higher Normal Forms

🔹 Boyce-Codd Normal Form (BCNF)

Definition:

- A stricter version of 3NF.
- Every determinant must be a candidate key.

Use Case:
Use when anomalies exist in 3NF due to overlapping candidate keys.

🔹 Fourth Normal Form (4NF)

Definition:

- Must be in BCNF.
- Eliminates multi-valued dependencies.

Example:
If an employee can have multiple skills and multiple projects independently, they must be separated.

🔹 Fifth Normal Form (5NF)

Definition:

- Deals with **join dependencies**.
- Ensures that data cannot be reconstructed incorrectly from its smaller pieces.

🔄 De-Normalization

🔹 What is De-Normalization?

De-normalization is the process of combining normalized tables into fewer tables to improve read performance.

🔹 Why Apply It?

- Faster query performance for read-heavy applications.
- Reduces complex joins.

🔹 When to Apply It?

- In reporting systems or OLAP databases.
- When performance is prioritized over update anomalies.

- When joins are becoming a bottleneck.

🔹 Example:

Normalized Schema:

- Customers Table:

CustomerID	CustomerName
C001	Ali
C002	Ahmed

- Product table:

ProductID	ProductName
P101	Laptop
P102	PC

- Orders Table:

OrderID	CustomerID	OrderDate
O101	C001	2024-04-15
O102	C002	2024-04-17

- OrderDetails Table:

ProductID	OrderID
P101	O101
P102	O101
P102	O102

De-normalized Table:

OrderID	CustomerName	OrderDate	ProductName
O101	Ali	2024-04-15	Laptop
O102	Ahmed	2024-04-17	PC

✅ Conclusion

Normalization is essential for data integrity, avoiding redundancy, and ensuring consistency in databases. However, in real-world systems—especially in analytical and reporting use cases—**de-normalization** may be applied for performance optimization. Understanding when and how to apply each form is critical for efficient database design.
