**TEAM PUZZLE (NSS)**

Q. Probable use cases of joins in real time applications.

1. E-commerce Applications:

- Product Listings: Joining tables for products, categories, and brands to display comprehensive product information with relevant category and brand details.

- Order Management: Joining tables for orders, customers, and products to retrieve order details along with customer and product information.

2. Social Media Platforms:

- Friend Connections: Joining tables for users and connections to determine mutual friends or recommended connections.

- Posts and Comments: Joining tables for posts, comments, and users to display post content along with user information and associated comments.

3. Financial Systems:

- Account Transactions: Joining tables for accounts, transactions, and customers to retrieve transaction details with corresponding account and customer information.

- Reporting and Analytics: Joining multiple tables to generate complex financial reports and perform analysis across different entities such as accounts, customers, and transactions.

4. Content Management Systems:

- Content Relationships: Joining tables for articles, categories, and tags to establish relationships between articles and associated categories or tags.

- Content Filtering: Joining tables for articles, authors, and user preferences to filter and recommend personalized content based on user preferences and article metadata.

5. Travel and Booking Systems:

- Flight Reservations: Joining tables for flights, passengers, and bookings to retrieve detailed flight reservation information along with passenger and booking details.

- Hotel Bookings: Joining tables for hotels, bookings, and guests to retrieve hotel booking information along with guest details.

Q. Give a read on normalization and types of normalization.

Normalization is a technique used in database management systems (DBMS) to design and structure relational databases in a way that reduces data redundancy and dependency. The goal of normalization is to eliminate anomalies and ensure data integrity, consistency, and efficiency in a database.

There are several normal forms in database normalization, each building upon the previous one. The most commonly used normal forms are:

1. First Normal Form (1NF): In 1NF, the data is organized into tables with rows and columns, and each column contains atomic (indivisible) values. There should be no repeating groups or arrays within a table.

2. Second Normal Form (2NF): 2NF builds on 1NF by ensuring that each non-key column in a table is fully functionally dependent on the entire primary key. It means that no partial dependencies exist, where a non-key column depends on only a portion of the primary key.

3. Third Normal Form (3NF): 3NF builds on 2NF by eliminating transitive dependencies. Transitive dependency occurs when a non-key column depends on another non-key column rather than directly on the primary key. To achieve 3NF, such dependencies must be removed by creating separate tables.

4. Boyce-Codd Normal Form (BCNF): BCNF is an extension of 3NF and addresses situations where there are overlapping candidate keys and non-key dependencies. It ensures that every determinant (column or set of columns that uniquely determines another column) is a candidate key.

5. Fourth Normal Form (4NF): 4NF deals with multi-valued dependencies. It eliminates redundant data that arises when a table contains independent multi-valued facts for a single primary key.

EXAMPLES

1. First Normal Form (1NF):

Consider a table called "Books" with the following columns: Book\_ID, Title, Authors, and Genres. To convert it to 1NF, we need to ensure that each column contains atomic values. We can split the "Authors" column into multiple rows, each representing a single author, like this:

| Book\_ID | Title | Author | Genre |

| 1 | To Kill a Mockingbird | Harper Lee | Fiction |

| 2 | 1984 | George Orwell | Fiction |

| 3 | The Great Gatsby | F. Scott Fitzgeral | Fiction |

2. Second Normal Form (2NF):

Let's consider a table called "Books" with the following columns: Book\_ID, Title, Author\_ID, Author\_Name, and Genre. In this case, we need to eliminate partial dependencies, where a non-key column depends on only a portion of the primary key. We can split the table into two separate tables: "Books" and "Authors," like this:

Table: Books

| Book\_ID | Title | Author\_ID | Genre |

| 1 | To Kill a Mockingbird | 1 | Fiction |

| 2 | 1984 | 2 | Fiction |

| 3 | The Great Gatsby | 3 | Fiction |

Table: Authors

| Author\_ID | Author\_Name |

| 1 | Harper Lee |

| 2 | George Orwell |

| 3 | F. Scott Fitzgerald |

3. Third Normal Form (3NF):

Let's consider a table called "Books" with the following columns: Book\_ID, Title, Author\_ID, Author\_Name, Genre, and Publisher. In this case, we need to eliminate transitive dependencies. We can split the table into three separate tables: "Books," "Authors," and "Publishers," like this:

Table: Books

| Book\_ID | Title | Author\_ID | Genre | Publisher\_ID |

| 1 | To Kill a Mockingbird | 1 | Fiction | 1 |

| 2 | 1984 | 2 | Fiction | 2 |

| 3 | The Great Gatsby | 3 | Fiction | 3 |

Table: Authors

| Author\_ID | Author\_Name |

| 1 | Harper Lee |

| 2 | George Orwell |

| 3 | F. Scott Fitzgerald |

Table: Publishers

| Publisher\_ID | Publisher\_Name |

| 1 | Penguin Classics |

| 2 | Vintage Books |

| 3 | Scribner |

4. Boyce-Codd Normal Form (BCNF):

Consider a table called "Employees" with the following columns: Employee\_ID, Employee\_Name, Department\_ID, and Department\_Name. Let's assume that each department has a unique ID and a unique name. In this case, the Department\_Name column depends on the Department\_ID, which is a part of the candidate key. To convert it to BCNF, we split the table into two separate tables: "Employees" and "Departments."

Table: Employees

| Employee\_ID | Employee\_Name |

| 1 | John Doe |

| 2 | Jane Smith |

Table: Departments

| Department\_ID | Department\_Name |

| 1 | HR |

| 2 | Sales |

In the "Employees" table, the Department\_ID becomes a foreign key referencing the "Departments" table.

5. Fourth Normal Form (4NF):

Consider a table called "Students" with the following columns: Student\_ID, Course\_ID, Course\_Name, and Course\_Teacher. Assume that each student can enroll in multiple courses, and each course can have multiple teachers. In this case, we have a multi-valued dependency between the Course\_ID and Course\_Teacher. To achieve 4NF, we split the table into three separate tables: "Students," "Courses," and "Teachers."

Table: Students

| Student\_ID | Student\_Name |

| 1 | John Doe |

| 2 | Jane Smith |

Table: Courses

| Course\_ID | Course\_Name |

| 1 | Math |

| 2 | English |

| 3 | Science |

Table: Teachers

| Course\_ID | Course\_Teacher |

| 1 | Mr. Johnson |

| 1 | Mrs. Smith |

| 2 | Mrs. Anderson |

| 3 | Mr. Davis |

In the "Teachers" table, the Course\_ID becomes a foreign key referencing the "Courses" table. Each row represents a single teacher for a specific course.