Univent Database Schema – 2NF Justification

SQL Code for Database and Tables

```
CREATE DATABASE Univent;
USE Univent;

-- USER Table
CREATE TABLE User (
    user_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    age INT,
    role VARCHAR(50),
    email VARCHAR(100) UNIQUE,
    college VARCHAR(50)
);

-- COLLEGE Table
CREATE TABLE College (
    college_id INT PRIMARY KEY,
    name VARCHAR(100),
    location VARCHAR(100)
);

-- SUPER_ADMIN Table
CREATE TABLE Super_Admin (
    admin_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100),
    password VARCHAR(100),
    designation VARCHAR(100),
    college_id INT,
    FOREIGN KEY (college_id) REFERENCES College(college_id)
);

-- CLUB_OR_SOCIETY Table
CREATE TABLE Club (
    club_id INT PRIMARY KEY,
    name VARCHAR(100),
    email VARCHAR(100),
    Category VARCHAR(100),
    secretary_name VARCHAR(100),
    secretary_id INT,
    college_id INT,
    FOREIGN KEY (college_id) REFERENCES College(college_id),
    FOREIGN KEY (secretary_id)REFERENCES USER(user_id)
);

-- EVENT Table
CREATE TABLE Event (
    event_id INT PRIMARY KEY,
    name VARCHAR(100),
    type_of_event VARCHAR(100),
    date DATE,
    location VARCHAR(100),
```

```
    status VARCHAR(50),
    organised_BY INT,
    max_num_of_participants INT,
    FOREIGN KEY (organised_BY) REFERENCES Club(club_id)
);

-- Event Scheduler for Past Events
SET GLOBAL event_scheduler = ON;
DELIMITER $$

CREATE EVENT move_old_events_to_past
ON SCHEDULE EVERY 1 DAY
DO
BEGIN
  INSERT INTO Past_Event (event_id, name, type_of_event, date, location,
status)
  SELECT event_id, name, type_of_event, date, location, status
  FROM Event
  WHERE date < CURDATE() - INTERVAL 2 DAY;

  DELETE FROM Event
  WHERE date < CURDATE() - INTERVAL 2 DAY;
END$$

DELIMITER ;

-- COMPETITION Table
CREATE TABLE Competition (
    comp_id INT auto_increment PRIMARY KEY,
    name VARCHAR(100),
    type_of_comp VARCHAR(100),
    date DATE,
    venue VARCHAR(100),
    event_id INT,
    FOREIGN KEY (event_id) REFERENCES Event(event_id)
);

-- TRANSACTION Table
CREATE TABLE Transaction (
    trans_id INT AUTO_INCREMENT PRIMARY KEY,
    amount DECIMAL(10, 2),
    description TEXT,
    trans_type VARCHAR(50),
    transferred_to INT,
    FOREIGN KEY (transferred_to) REFERENCES Club(club_id)
);

-- REGISTERS Table
CREATE TABLE Registers (
    reg_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    event_id INT,
    UNIQUE (user_id, event_id),
    FOREIGN KEY (user_id) REFERENCES User(user_id),
    FOREIGN KEY (event_id) REFERENCES Event(event_id)
);

-- REQUESTS_APPROVAL Table
CREATE TABLE Requests_Approval (
    request_id INT PRIMARY KEY,
    club_id INT,
```

```
    source VARCHAR(100),
    status VARCHAR(50),
    approved_by INT,
    rejected_by INT,
    FOREIGN KEY (club_id) REFERENCES Club(club_id),
    FOREIGN KEY (approved_by) REFERENCES Super_Admin(admin_id),
    FOREIGN KEY (rejected_by) REFERENCES Super_Admin(admin_id)
);

-- FEEDBACK Table
CREATE TABLE Feedback (
    feedback_id INT PRIMARY KEY,
    event_id INT,
    user_id INT,
    time TIMESTAMP,
    rating INT CHECK (rating BETWEEN 1 AND 5),
    comment TEXT,
    FOREIGN KEY (event_id) REFERENCES Event(event_id),
    FOREIGN KEY (user_id) REFERENCES User(user_id)
);
```

Why the Univent Schema is in 2NF

What is Second Normal Form (2NF)?

A table is in **2NF** if:

1. It is already in **First Normal Form (1NF)** (i.e., atomic values and unique rows).
2. It has **no partial dependency**—which means that all non-prime (non-key) attributes are fully functionally dependent on the **entire primary key**, not just part of it.

Univent Schema Analysis

Here's why each table is in **2NF**:

Table Name	Primary Key	2NF Justification
User	`user_id`	All columns like name, age, role, email depend fully on user_id.
College	`college_id`	`name` and `location` are atomic and depend on the full key.
Super_Admin	`admin_id`	Attributes depend on `admin_id`; `college_id` is a foreign key, not part of PK.
Club	`club_id`	Attributes like `name`, `email`, etc., depend entirely on `club_id`.
Event	`event_id`	All fields are fully dependent on `event_id`.

Competition	`comp_id`	All fields depend on `comp_id`; `event_id` is a FK.
Transaction	`trans_id`	Each attribute is fully dependent on `trans_id`.
Registers	`reg_id` (and unique `user_id, event_id`)	Fully functionally dependent on the primary key.
Requests_Approval	`request_id`	Each attribute depends fully on `request_id`.
Feedback	`feedback_id`	All attributes are fully functionally dependent.

Key Observations

- No table has **composite primary keys** where only part of the key is used to determine a non-key attribute.
- Each table contains **atomic, non-redundant data** with all fields depending **only on their respective primary keys**.
- The **foreign keys** (like `college_id`, `event_id`, etc.) are used only to **connect related tables**, and don't violate 2NF principles.

Conclusion

The **Univent Database Schema** is fully normalized to **2NF**:

- **No partial dependencies**
- **No data redundancy**
- **Improved data integrity**

This ensures better **efficiency**, **scalability**, and **clean relational structure** for managing event and college-related data.

Steps to Move from 2NF to 3NF:

1. **Remove `secretary_name`** from the `Club` table to eliminate the transitive dependency.
2. **Ensure `secretary_id`** is retained in the `Club` table as it correctly references `user_id` from the `User` table.

SQL Code:

```
-- Remove the secretary_name column from the Club table
ALTER TABLE Club
DROP COLUMN secretary_name;
```

Explanation of the Fix:

- **Before**: The `secretary_name` in the `Club` table depended on `secretary_id`, which in turn depended on `user_id` in the `User` table. This created a transitive dependency.
- **After**: By removing `secretary_name` from the `Club` table, we ensure that all non-key attributes in `Club` depend directly on the primary key (`club_id`), which satisfies the 3NF condition.

Transition from 3NF to BCNF:

Overview of BCNF:

To transition a schema from **3NF** to **BCNF** (Boyce-Codd Normal Form), we must ensure that for every functional dependency in a table, the left-hand side of the dependency (the determinant) is a **superkey**. A **superkey** is any set of attributes that can uniquely identify a record in a table.

---

1. Identification of BCNF Violations:

In our schema, after achieving **3NF**, the next step is to verify if all functional dependencies have **superkeys** on the left-hand side. A table is in **BCNF** if and only if for every non-trivial functional dependency, the determinant is a superkey.

Step-by-Step Analysis:

1. User Table:

- **Primary Key**: user_id
- Functional dependencies:
  - user_id → (first_name, last_name, age, role, email)
- Since user_id is the primary key, this table satisfies **BCNF**.

2. College Table:

- **Primary Key**: college_id
- Functional dependencies:
  - college_id → (name, location)
- Since college_id is the primary key, this table satisfies **BCNF**.

3. Super_Admin Table:

- **Primary Key**: admin_id
- Functional dependencies:
  - admin_id → (first_name, last_name, email, password, designation, college_id)
- Since admin_id is the primary key, this table satisfies **BCNF**.

4. Club Table:

- **Primary Key**: club_id
- Functional dependencies:
  - club_id → (name, email, category, secretary_id, college_id)
  - secretary_id → secretary_name (However, this dependency was resolved in 3NF by removing the secretary_name column from the Club table.)
- As the remaining dependencies in the table are on the primary key (club_id) and foreign keys referencing other tables, this table satisfies **BCNF**.

5. Event Table:

- **Primary Key**: event_id

- Functional dependencies:
  - event_id → (name, type_of_event, date, location, status, organised_BY, max_num_of_participants)
  - organised_BY → (club_id) (via foreign key organised_BY referencing Club(club_id))
- Since event_id is the primary key, this table satisfies **BCNF**.

6. Competition Table:

- **Primary Key**: comp_id
- Functional dependency:
  - comp_id → (name, type_of_comp, date, venue, event_id)
- Since comp_id is the primary key, this table satisfies **BCNF**.

7. Transaction Table:

- **Primary Key**: trans_id
- Functional dependency:
  - trans_id → (amount, description, trans_type, transferred_to)
  - transferred_to → (club_id) (via foreign key transferred_to referencing Club(club_id))
- Since trans_id is the primary key, this table satisfies **BCNF**.

8. Registers Table:

- **Primary Key**: reg_id
- Functional dependencies:
  - reg_id → (user_id, event_id)
  - user_id → user_details (since we know user details are already in the User table, this is not problematic.)
- Since reg_id is the primary key, this table satisfies **BCNF**.

9. Requests_Approval Table:

- **Primary Key**: request_id
- Functional dependencies:
  - request_id → (club_id, source, status, approved_by, rejected_by)
- Since request_id is the primary key, this table satisfies **BCNF**.

10. Feedback Table:

- **Primary Key**: feedback_id
- Functional dependency:
  - feedback_id → (event_id, user_id, time, rating, comment)
- Since feedback_id is the primary key, this table satisfies **BCNF**.

Conclusion:

After analyzing all the tables in the schema, we see that **all of them satisfy BCNF**. There are no violations where a non-superkey determines other attributes. Hence, the schema is already in **BCNF**.