Database Normalization Exercises (1NF, 2NF, 3NF)

Database Normalization Exercises (1NF, 2NF, 3NF)

First Normal Form (1NF) Exercises

Exercise 1: Project Management System

- **Initial Table:** Project(ProjectID, ProjectName, EmployeesAssigned, TaskList, StartDate, EndDate)
- **Violations of 1NF:**
  - EmployeesAssigned contains a list (e.g., "John, Mary, Peter").
  - TaskList contains a list (e.g., "Design, Code, Test").
- **Transformation to 1NF:**
  - **New Tables:**
    1. **Project**
       - ProjectID (PK)
       - ProjectName
       - StartDate
       - EndDate
    2. **ProjectEmployee**
       - ProjectID (FK)
       - EmployeeName
       - *Composite PK: ProjectID + EmployeeName*
    3. **ProjectTask**
       - ProjectID (FK)
       - TaskName

            - *Composite PK: ProjectID + TaskName*

Exercise 2: Event Booking System

- **Initial Table:** EventBooking(BookingID, CustomerName, EventDates, SeatNumbers, TicketPrices)
- **Violations of 1NF:**
    - EventDates, SeatNumbers, and TicketPrices contain multiple values.
- **Transformation to 1NF:**
    - **New Tables:**
        1. **EventBooking**
            - BookingID (PK)
            - CustomerName
        2. **EventBookingDetail**
            - BookingDetailID (PK)
            - BookingID (FK)
            - EventDate
            - SeatNumber
            - TicketPrice

Exercise 3: Library Management System

- **Initial Table:** Library(BookID, BookTitle, Authors, Borrowers, BorrowDates)
- **Violations of 1NF:**
    - Authors, Borrowers, and BorrowDates are lists.
- **Transformation to 1NF:**
    - **New Tables:**
        1. **Book**
            - BookID (PK)
            - BookTitle

2. **BookAuthor**
   - BookID (FK)
   - AuthorName
   - *Composite PK: BookID + AuthorName*
3. **BorrowRecord**
   - BorrowID (PK)
   - BookID (FK)
   - BorrowerName
   - BorrowDate

Exercise 4: Course Management System

- **Initial Table:** Course(CourseID, CourseName, Instructors, StudentsEnrolled, ExamDates)
- **Violations of 1NF:**
  - Instructors, StudentsEnrolled, and ExamDates contain lists.
- **Transformation to 1NF:**
  - **New Tables:**
    1. **Course**
       - CourseID (PK)
       - CourseName
    2. **CourseInstructor**
       - CourseID (FK)
       - InstructorName
       - *Composite PK: CourseID + InstructorName*
    3. **CourseStudent**
       - CourseID (FK)
       - StudentName

        - *Composite PK: CourseID + StudentName*

    4. **CourseExam**
        - CourseID (FK)
        - ExamDate
        - *Composite PK: CourseID + ExamDate*

Exercise 5: Online Sales System

- **Initial Table:** Order(OrderID, CustomerName, ProductList, Quantities, Prices, OrderDate)
- **Violations of 1NF:**
    - ProductList, Quantities, and Prices contain multiple values.
- **Transformation to 1NF:**
    - **New Tables:**
        1. **Order**
            - OrderID (PK)
            - CustomerName
            - OrderDate
        2. **OrderItem**
            - OrderItemID (PK)
            - OrderID (FK)
            - ProductName
            - Quantity
            - Price

Second Normal Form (2NF) Exercises

Exercise 6: Warehouse Management System

- **Initial Table (1NF):** Warehouse(WarehouseID, ProductID, ProductName, WarehouseLocation, StockQuantity)

- **Primary Key:** (WarehouseID, ProductID)
- **Analysis:**
    - ProductName depends only on ProductID → Partial Dependency.
    - WarehouseLocation depends only on WarehouseID → Partial Dependency.
- **Normalization to 2NF:**
    - Remove partial dependencies.
    - **New Tables:**
        1. **Warehouse**(WarehouseID, WarehouseLocation)
        2. **Product**(ProductID, ProductName)
        3. **Stock**(WarehouseID, ProductID, StockQuantity)
- **Relationships:**
    - Stock.WarehouseID → Warehouse.WarehouseID
    - Stock.ProductID → Product.ProductID

Exercise 7: Order Management System

- **Initial Table (1NF):** OrderDetails(OrderID, ProductID, CustomerID, ProductName, CustomerAddress, Quantity)
- **Primary Key:** (OrderID, ProductID)
- **Analysis:**
    - ProductName → ProductID → Partial Dependency
    - CustomerAddress → CustomerID → Partial Dependency
- **Normalization to 2NF:**
    - **New Tables:**
        1. **OrderDetails**(OrderID, ProductID, CustomerID, Quantity)
        2. **Product**(ProductID, ProductName)
        3. **Customer**(CustomerID, CustomerAddress)
- **Relationships:**

    - OrderDetails.ProductID → Product.ProductID
    - OrderDetails.CustomerID → Customer.CustomerID

Exercise 8: Class Management System

- **Initial Table (1NF):** ClassSchedule(ClassID, TeacherID, RoomID, TeacherName, RoomLocation, ClassTime)
- **Primary Key:** (ClassID, TeacherID)
- **Analysis:**
    - TeacherName → TeacherID → Partial Dependency
    - RoomLocation → RoomID → Partial Dependency
- **Normalization to 2NF:**
    - **New Tables:**
        1. **ClassSchedule**(ClassID, TeacherID, RoomID, ClassTime)
        2. **Teacher**(TeacherID, TeacherName)
        3. **Room**(RoomID, RoomLocation)
- **Relationships:**
    - ClassSchedule.TeacherID → Teacher.TeacherID
    - ClassSchedule.RoomID → Room.RoomID

Exercise 9: Delivery Management System

- **Initial Table (1NF):** Delivery(DeliveryID, DriverID, PackageID, DriverName, PackageWeight, DeliveryDate)
- **Primary Key:** (DeliveryID, PackageID)
- **Analysis:**
    - DriverName → DriverID → Partial Dependency
    - PackageWeight → PackageID → Partial Dependency
- **Normalization to 2NF:**
    - **New Tables:**
        1. **Delivery**(DeliveryID, DriverID, PackageID, DeliveryDate)

2. **Driver**(DriverID, DriverName)
3. **Package**(PackageID, PackageWeight)

- **Relationships:**
    - Delivery.DriverID → Driver.DriverID
    - Delivery.PackageID → Package.PackageID

Exercise 10: Event Management System

- **Initial Table (1NF):** EventRegistration(EventID, ParticipantID, EventName, ParticipantEmail, RegistrationDate)
- **Primary Key:** (EventID, ParticipantID)
- **Analysis:**
    - EventName → EventID → Partial Dependency
    - ParticipantEmail → ParticipantID → Partial Dependency
- **Normalization to 2NF:**
    - **New Tables:**
        1. **EventRegistration**(EventID, ParticipantID, RegistrationDate)
        2. **Event**(EventID, EventName)
        3. **Participant**(ParticipantID, ParticipantEmail)
- **Relationships:**
    - EventRegistration.EventID → Event.EventID
    - EventRegistration.ParticipantID → Participant.ParticipantID

Third Normal Form (3NF) Exercises

Exercise 11: Human Resource Management System

- **Initial Table (2NF):** Employee(EmployeeID, DepartmentID, DepartmentName, DepartmentLocation, Salary)
- **Primary Key:** EmployeeID
- **Analysis:**

    - DepartmentName, DepartmentLocation → DepartmentID → Transitive Dependency

- **Normalization to 3NF:**
    - **New Tables:**
        1. **Employee**(EmployeeID, DepartmentID, Salary)
        2. **Department**(DepartmentID, DepartmentName, DepartmentLocation)
- **Relationships:**
    - Employee.DepartmentID → Department.DepartmentID

Exercise 12: Customer Management System

- **Initial Table (2NF):** Customer(CustomerID, SalespersonID, SalespersonName, SalespersonRegion, PurchaseAmount)
- **Primary Key:** CustomerID
- **Analysis:**
    - SalespersonName, SalespersonRegion → SalespersonID → Transitive Dependency
- **Normalization to 3NF:**
    - **New Tables:**
        1. **Customer**(CustomerID, SalespersonID, PurchaseAmount)
        2. **Salesperson**(SalespersonID, SalespersonName, SalespersonRegion)
- **Relationships:**
    - Customer.SalespersonID → Salesperson.SalespersonID

Exercise 13: School Management System

- **Initial Table (2NF):** Student(StudentID, CourseID, CourseName, DepartmentID, DepartmentHead)
- **Primary Key:** StudentID
- **Analysis:**
    - CourseName → CourseID

    - DepartmentHead → DepartmentID → Transitive Dependency
- **Normalization to 3NF:**
    - **New Tables:**
        1. **Student**(StudentID, CourseID)
        2. **Course**(CourseID, CourseName, DepartmentID)
        3. **Department**(DepartmentID, DepartmentHead)
- **Relationships:**
    - Student.CourseID → Course.CourseID
    - Course.DepartmentID → Department.DepartmentID

Exercise 14: Hospital Management System

- **Initial Table (2NF):** Patient(PatientID, DoctorID, DoctorName, DepartmentID, DepartmentName)
- **Primary Key:** PatientID
- **Analysis:**
    - DoctorName → DoctorID → Transitive Dependency
    - DepartmentName → DepartmentID → Transitive Dependency
- **Normalization to 3NF:**
    - **New Tables:**
        1. **Patient**(PatientID, DoctorID, DepartmentID)
        2. **Doctor**(DoctorID, DoctorName)
        3. **Department**(DepartmentID, DepartmentName)
- **Relationships:**
    - Patient.DoctorID → Doctor.DoctorID
    - Patient.DepartmentID → Department.DepartmentID

Exercise 15: Restaurant Management System

- **Initial Table (2NF):** Order(OrderID, WaiterID, WaiterName, KitchenID, KitchenLocation, OrderTotal)

- **Primary Key:** OrderID
- **Analysis:**
  - WaiterName → WaiterID
  - KitchenLocation → KitchenID
- **Normalization to 3NF:**
  - **New Tables:**
    1. **Order**(OrderID, WaiterID, KitchenID, OrderTotal)
    2. **Waiter**(WaiterID, WaiterName)
    3. **Kitchen**(KitchenID, KitchenLocation)
- **Relationships:**
  - Order.WaiterID → Waiter.WaiterID
  - Order.KitchenID → Kitchen.KitchenID

Expert-Level Normalization Exercises (1NF to 3NF)

Exercise 1: Hotel Booking System

- **Initial Table:** HotelBookings(booking_id, guest_id, guest_phone, room_id, room_type, room_price, check_in, check_out, hotel_id, hotel_city)
- **Primary Key**: (booking_id, room_id, check_in)
- **1NF:**
  - The table is already in 1NF (no repeating groups or multi-valued attributes).
- **2NF:**
  - **Violations:**
    - guest_phone depends on guest_id
    - room_type, room_price, hotel_id depend on room_id
    - hotel_city depends on hotel_id
  - These are partial dependencies → violates 2NF.

  - **Decompose into:**
    1. HotelBookings(booking_id, room_id, check_in, check_out, guest_id)
    2. Guest(guest_id, guest_phone)
    3. Room(room_id, room_type, room_price, hotel_id)
    4. Hotel(hotel_id, hotel_city)
- **3NF:**
  - All transitive dependencies are already eliminated in the decomposition above → satisfies 3NF.

Exercise 2: Order Management with Multivalued Attribute

- **Initial Table:** Orders(order_id, customer_id, items, order_date)
- items is a JSON list of products with product_id, name, price, quantity.
- **1NF:**
  - Flatten the items array → separate table for order items.
- **Decompose into:**
  1. Order(order_id, customer_id, order_date)
  2. OrderItem(order_item_id, order_id, product_id, quantity)
  3. Product(product_id, name, price)
- **2NF & 3NF:**
  - No partial dependencies.
  - All non-key attributes depend only on the key.
  - No transitive dependencies → satisfies 3NF.

Exercise 3: Education System (Complex Relationships)

- **Initial Table:** StudentCourses(student_id, student_name, department_id, department_head, course_id, course_name, instructor_id, instructor_email, grade)
- **Primary Key:** (student_id, course_id)
- **Violations of 2NF and 3NF:**

    - student_name and department_id depend only on student_id → partial dependency (2NF violation)
    - department_head depends on department_id → transitive dependency (3NF violation)
    - course_name and instructor_id depend only on course_id → partial dependency
    - instructor_email depends on instructor_id → transitive dependency
- **Decompose into 3NF:**
    1. Student(student_id, student_name, department_id)
    2. Department(department_id, department_head)
    3. Course(course_id, course_name, instructor_id)
    4. Instructor(instructor_id, instructor_email)
    5. Enrollment(student_id, course_id, grade)

Exercise 4: IoT System (Unstructured Data)

- **Initial Table:** SensorReadings(sensor_id, timestamps, values, location_id, location_zone)
- **Violation of 1NF:**
    - timestamps and values are comma-separated strings → multi-valued fields
- **Transformation to 1NF:**
    - Flatten each pair of timestamp and value into individual rows.
    - SensorReading(reading_id, sensor_id, timestamp, value, location_id)
- **Assumption:** location_id → location_zone
- **Normalization to 3NF:**
    1. SensorReading(reading_id, sensor_id, timestamp, value, location_id)
    2. Location(location_id, location_zone)
- **Relationships:**
    - SensorReading.location_id → Location.location_id

Exercise 5: Project Management with Transitive Dependencies

- **Initial Table:** ProjectTasks(project_id, project_name, client_id, client_industry, task_id, task_description, employee_id, employee_department)
- **Primary Key:** (project_id, task_id)
- **Transitive Dependencies Violating 3NF:**
    - project_id → project_name, client_id
    - client_id → client_industry
    - task_id → task_description
    - employee_id → employee_department

Attributes like client_industry are transitively dependent on project_id through client_id, and employee_department is transitively dependent on project_id through employee_id. This violates 3NF.

- **Normalization to 3NF:**
    - **New Tables:**
        1. Project(project_id, project_name, client_id)
        2. Client(client_id, client_industry)
        3. Task(task_id, task_description)
        4. Employee(employee_id, employee_department)
        5. ProjectTaskAssignment(project_id, task_id, employee_id)
- **Relationships:**
    - Project.client_id → Client.client_id
    - ProjectTaskAssignment.project_id → Project.project_id
    - ProjectTaskAssignment.task_id → Task.task_id
    - ProjectTaskAssignment.employee_id → Employee.employee_id

Exercise 6: Retail Store (Composite Key)

- **Initial Table:** RetailSales(store_id, product_id, product_category, supplier_id, supplier_region, sale_date, quantity, price)
- **Primary Key**: (store_id, product_id, sale_date)

- **Why it doesn't satisfy 2NF:**
    - Attributes product_category and supplier_id depend only on product_id, not the full composite key.
    - supplier_region depends on supplier_id, which is not part of the full primary key either.
    - This means there are partial dependencies, violating 2NF.
- **Normalization to 3NF:**
    - **New Tables:**
        1. RetailSales(store_id, product_id, sale_date, quantity, price)
        2. Product(product_id, product_category, supplier_id)
        3. Supplier(supplier_id, supplier_region)
- **Relationships:**
    - RetailSales.product_id → Product.product_id
    - Product.supplier_id → Supplier.supplier_id

Exercise 7: Medical System (Nested Data)

- **Initial Table:** PatientRecords(patient_id, patient_name, visits)
- **Violation of 1NF:**
    - The visits field is an XML structure, which means the table contains nested data, violating 1NF.
- **Transformation to 1NF:**
    - Flatten the XML by extracting each <visit> element into a separate row:
        - PatientVisit(visit_id, patient_id, visit_date, doctor_id, diagnosis)
        - visit_id is a surrogate key to uniquely identify each visit.
- **Assumption:** doctor_id → doctor_name
- **Transformation to 3NF:**
    - **New Tables:**
        1. Patient(patient_id, patient_name)

        2. Doctor(doctor_id, doctor_name)
        3. PatientVisit(visit_id, patient_id, visit_date, doctor_id, diagnosis)

- **Relationships:**
  - PatientVisit.patient_id → Patient.patient_id
  - PatientVisit.doctor_id → Doctor.doctor_id

Exercise 8: Auction System (Multivalued Dependencies)

- **Initial Table:** AuctionBids(auction_id, item_name, bidder_ids, bid_amounts, current_winner_id)
- **Violation of 1NF:**
  - bidder_ids and bid_amounts are arrays (e.g., "B001,B002" and "100,150") → multi-valued attributes.
- **Transformation to 1NF:**
  - Create separate rows for each bidder and their corresponding bid.
  - **New Table:**
    - **Bid**(auction_id, bidder_id, bid_amount)
- **Assumptions (Functional Dependencies):**
  - auction_id → item_name, current_winner_id
  - bidder_id is atomic
- **Normalization to 3NF:**
  - **New Tables:**
    1. **Auction**(auction_id, item_name, current_winner_id)
    2. **Bid**(auction_id, bidder_id, bid_amount)
    3. **Bidder**(bidder_id) *(Optional, depending on system)*

Exercise 9: Inventory Management (Derived Attributes)

- **Initial Table:** Inventory(product_id, product_name, warehouse_id, warehouse_location, current_stock, reorder_level, last_restock_date)

- **Primary Key:** (product_id, warehouse_id)
- **Violations:**
    - product_name depends only on product_id → partial dependency → violates 2NF
    - warehouse_location depends only on warehouse_id → partial dependency → violates 2NF
    - reorder_level depends on both product_id and warehouse_id, but is a **derived/calculated attribute** based on business rules (e.g., safety stock levels), and might be recomputed → include it carefully
- **Normalization to 3NF:**
    - **New Tables:**
        1. **Product**(product_id, product_name)
        2. **Warehouse**(warehouse_id, warehouse_location)
        3. **Stock**(product_id, warehouse_id, current_stock, reorder_level, last_restock_date)
- **Relationships:**
    - Stock.product_id → Product.product_id
    - Stock.warehouse_id → Warehouse.warehouse_id

Exercise 10: Social Media (Recursive Relationships)

- **Initial Table:** UserPosts(post_id, user_id, user_name, post_content, parent_post_id, parent_post_user)
- **Primary Key:** post_id
- **Violations:**
    - user_name depends on user_id → transitive dependency → violates 3NF
    - parent_post_user depends on parent_post_id → transitive dependency → violates 3NF
- **Normalization to 3NF:**
    - **New Tables:**

1. **User**(user_id, user_name)
2. **Post**(post_id, user_id, post_content, parent_post_id)

- **Handling Recursive Relationship:**
    - parent_post_id is a foreign key in the **Post** table that references the **post_id** of another post in the same table.
    - This allows the system to model threaded or nested replies by associating a post with its parent.
    - In database schema:
        - Post.parent_post_id → Post.post_id (self-referencing foreign key)
        - If parent_post_id is NULL, the post is a top-level (root) post.
        - If parent_post_id is NOT NULL, the post is a reply to another post.
- **Relationships:**
    - Post.user_id → User.user_id
    - Post.parent_post_id → Post.post_id (recursive)**