I. Library Management Database Problem Statement

A library management system requires a comprehensive database to handle its various functions. This includes managing books, authors, members, loans, and staff. The database should facilitate efficient book tracking, member management, loan processing, and staff operations.

Detailed Requirements

1. **Books:**
    - Purpose: Store detailed information about each book available in the library.
    - Fields:
        - `book_id` (Primary Key): Unique identifier for each book.
        - `title`: Title of the book.
        - `author_id` (Foreign Key): References the author of the book.
        - `publication_year`: Year the book was published.
        - `genre`: Genre of the book (e.g., Fiction, Non-Fiction).
        - `isbn`: International Standard Book Number.
        - `available_copies`: Number of copies available in the library.
2. **Authors:**
    - Purpose: Record information about authors who have written books in the library.
    - Fields:
        - `author_id` (Primary Key): Unique identifier for each author.
        - `first_name`: Author's first name.
        - `last_name`: Author's last name.
        - `date_of_birth`: Author's date of birth.
        - `nationality`: Author's nationality.
3. **Members:**
    - Purpose: Manage details of library members.
    - Fields:
        - `member_id` (Primary Key): Unique identifier for each member.
        - `first_name`: Member's first name.
        - `last_name`: Member's last name.
        - `date_of_birth`: Member's date of birth.
        - `contact_number`: Member's phone number.
        - `email`: Member's email address.
        - `membership_date`: Date when the member joined the library.
4. **Loans:**
    - Purpose: Track the borrowing of books by members.
    - Fields:

        - `loan_id` (Primary Key): Unique identifier for each loan record.
        - `book_id` (Foreign Key): References the book being borrowed.
        - `member_id` (Foreign Key): References the member borrowing the book.
        - `loan_date`: Date when the book was borrowed.
        - `return_date`: Date when the book is due to be returned.
        - `actual_return_date`: Date when the book was actually returned.

5. **Staff:**
    - Purpose: Maintain information about library staff.
    - Fields:
        - `staff_id` (Primary Key): Unique identifier for each staff member.
        - `first_name`: Staff member's first name.
        - `last_name`: Staff member's last name.
        - `position`: Position of the staff member (e.g., Librarian, Assistant).
        - `contact_number`: Staff member's phone number.
        - `email`: Staff member's email address.
        - `hire_date`: Date when the staff member was hired.

Apply the Below Queries on DB

DDL Queries

1. Add a new column to the `books` table:
2. Rename the `position` column in the `staff` table to `job_title`:
3. drop the `email` column from the `members` table:

DML Queries

1. Insert new data into the books table:
2. Update a member's contact number:
3. Delete a specific loan record:
4. Insert a new loan record:

Join Queries

1. Retrieve all books along with their authors:
2. Find all books currently on loan along with member details:
3. List all books borrowed by a specific member:
4. Get the total number of books and the total available copies for each genre:
5. Find all staff members who are librarians and their hire dates:

II. Insurance Domain Database Problem Statement

An insurance company requires a comprehensive database to manage its operations effectively. The database needs to capture detailed information about customers, policies, claims, agents, and their interactions. The goal is to ensure that the database can support various operations, reporting, and analysis needs.

Detailed Requirements

1. **Customers:**
   - **Purpose:** Store detailed information about each customer.
   - **Fields:**
     - `customer_id` (Primary Key): Unique identifier for each customer.
     - `first_name`: Customer's first name.
     - `last_name`: Customer's last name.
     - `date_of_birth`: Customer's date of birth.
     - `gender`: Gender of the customer (e.g., Male, Female).
     - `contact_number`: Customer's phone number.
     - `email`: Customer's email address.
     - `address`: Customer's home address.
2. **Policies:**
   - **Purpose:** Record details about the insurance policies offered by the company.
   - **Fields:**
     - `policy_id` (Primary Key): Unique identifier for each policy.
     - `policy_name`: Name of the insurance policy.
     - `policy_type`: Type of policy (e.g., Health, Auto, Life).
     - `coverage_details`: Details of what is covered under the policy.
     - `premium`: The amount paid for the policy.
     - `start_date`: The start date of the policy.
     - `end_date`: The end date of the policy.
3. **Claims:**
   - **Purpose:** Track claims made by customers.
   - **Fields:**
     - `claim_id` (Primary Key): Unique identifier for each claim.
     - `claim_date`: The date when the claim was made.
     - `claim_amount`: The total amount claimed by the customer.
     - `approved_amount`: The amount approved by the insurance company.
     - `claim_status`: Status of the claim (e.g., Approved, Denied).

        - `policy_id` (Foreign Key): References the policy associated with the claim.
        - `customer_id` (Foreign Key): References the customer who made the claim.

4. **Agents:**
    - **Purpose:** Maintain information about insurance agents.
    - **Fields:**
        - `agent_id` (Primary Key): Unique identifier for each agent.
        - `first_name`: Agent's first name.
        - `last_name`: Agent's last name.
        - `contact_number`: Agent's phone number.
        - `email`: Agent's email address.
        - `hire_date`: The date when the agent was hired.
5. **Policy Assignments:**
    - **Purpose:** Manage the assignment of policies to customers.
    - **Fields:**
        - `assignment_id` (Primary Key): Unique identifier for each policy assignment.
        - `customer_id` (Foreign Key): References the customer to whom the policy is assigned.
        - `policy_id` (Foreign Key): References the assigned policy.
        - `start_date`: The start date of the policy assignment.
        - `end_date`: The end date of the policy assignment.
6. **Claim Processing:**
    - **Purpose:** Track the processing of claims, including claim amounts, approved amounts, and payment details.
    - **Fields:**
        - `processing_id` (Primary Key): Unique identifier for each claim processing record.
        - `claim_id` (Foreign Key): References the claim being processed.
        - `processing_date`: The date when the claim was processed.
        - `payment_amount`: The amount paid out for the claim.
        - `payment_date`: The date when the payment was made.

Relationships and Use Cases

1. **Customers and Policies:**
    - Customers can hold multiple policies. The relationship is managed through the `Policy Assignments` table, which links customers with the policies they hold.
2. **Policies and Claims:**

    - Each policy can have multiple claims associated with it. Claims are linked to policies through the `policy_id` field.
3. **Customers and Claims:**
    - Customers can file multiple claims. Each claim is associated with one customer through the `customer_id` field.
4. **Claims and Claim Processing:**
    - Each claim can undergo multiple processing steps. The `Claim Processing` table records each processing step related to a specific claim.
5. **Agents and Policies:**
    - Agents manage policies, which can be tracked through policy assignments. This relationship helps in identifying which agent is responsible for managing which policies.

Apply the Below Queries on DB

DDL Queries

1. Add a new column to the `agents` table:
2. Rename the `policy_name` column in the `policies` table to `policy_title`:
3. Drop the `address` column from the `customers` table:

DML Queries

1. Update a policy's premium amount:
2. Delete a specific claim:
3. Insert a new policy assignment:

Join Queries

1. Retrieve all customers with their assigned policies and agents:
2. Find all claims and the associated policy details:
3. List all claims along with the customer details:
4. Get the total claim amount and number of claims per policy type:
5. Find the most recent claim for each customer: