Task (4)
"Hospital Patient Management System"

Description:

Develop a hospital patient management system to automate tasks like patient admission, treatment scheduling, discharge processing, and generating operational reports. Utilize PL/SQL procedures, functions, triggers, cursors, and transaction management to ensure data consistency and enforce hospital policies.

Tables & Relationships

Patients: (id, name, date_of_birth, status, total_bill) - Tracks patient details and their current status (admitted, discharged, etc.).
Doctors: (id, name, specialty, available_hours) - Records doctor information and their availability.
Appointments: (id, patient_id, doctor_id, appointment_date, status) - Logs scheduled appointments and their statuses (scheduled, completed, canceled).
Treatments: (id, patient_id, doctor_id, treatment_description, cost) - Records treatments provided to patients and their costs.
Rooms: (id, type, capacity, availability) - Tracks room inventory and availability.
AuditTrail: (id, table_name, operation, old_data, new_data, timestamp) - Logs changes to critical tables for audit purposes.
Warnings: (id, patient_id, warning_reason, warning_date) - Stores warnings for patients, such as failure to follow medical advice or payment delays.

Features to Be Covered:

1. **Patient Admission Validation**
   - Write a trigger to validate patient admission by checking room availability. The trigger should raise an error if no room of the requested type is available.
   - Enhance the trigger to update the Rooms table to reflect the room assignment upon successful admission and log this in the AuditTrail table.
2. **Appointment Scheduling**
   Write a PL/SQL procedure to schedule appointments. The procedure should:
   - Check if the doctor is available during the requested time slot.
   - Insert the appointment into the Appointments table with a "Scheduled" status.
   - Update the doctor's available hours.
3. **Treatment Cost Calculation**
   Write a PL/SQL function to calculate the total cost of treatments for a patient. The function should:
   - Aggregate the costs from the Treatments table for the given patient ID.
   - Automatically update the total_bill column in the Patients table when the function is executed.
4. **Room Assignment Tracking**
   Write a BEFORE INSERT trigger on the Patients table to:

   - Automatically assign an available room based on room type and capacity.

- Update the Rooms table to reflect the change in room availability.
- Insert an entry in the AuditTrail table with details of the operation.

5. **Discharge Processing**
   Write a PL/SQL procedure to process patient discharge. The procedure should:
   - Mark the patient's status as "Discharged" in the Patients table.
   - Update the assigned room's availability in the Rooms table.
   - Log the discharge details in the AuditTrail table.
6. **Hospital Performance Report**
   Write a PL/SQL cursor to generate a hospital performance report that includes:
   - Total admissions and discharges.
   - Average patient stay duration (in days).
   - The top three doctors based on the number of treatments handled.
7. **Multi-Appointment Cancellation**
   Write a PL/SQL block to cancel multiple appointments in a single transaction. The block should:
   - Check the status of each appointment before cancellation.
   - Roll back all changes if any cancellation fails, ensuring data consistency.
8. **Patient Warnings and Status Update**
   Write a PL/SQL procedure to issue warnings to patients. The procedure should:
   - Automatically create a warning if a patient misses an appointment or delays bill payment.
   - Update the patient's status to "Flagged" in the Patients table if they receive three warnings.
   - Insert the warning details and status change in the AuditTrail table.
9. **User Management and Privileges**
   - Create a Manager User and grant them a role of privileges to create two users. Let User 1 create the **Patients** and **Rooms** tables. Let User 2 insert 5 rows into each table for patients and rooms.
10. **Blocker-Waiting Situation**
    - Demonstrate generating a blocker-waiting situation using two transactions by User 1 and User 2. The transaction involves updating the `availability` field of the **Rooms** table for a specific room type.
11. **Identifying Blocker and Waiting Sessions**
    - Identify the sessions in the blocker-waiting situation using SID and SERIAL# for both the blocker and waiting sessions.
12. **Deadlock Demonstration**
    - Demonstrate a deadlock scenario using simultaneous transactions by User 1 and User 2 and display the expected results with appropriate rollback mechanisms and show did you handle the deadlock.

Instructions

- All team members should be aware of every task, as each member may be asked randomly about any task. Additionally, you must have a clear understanding of all the concepts studied in Oracle.
- All scripts should be placed in one single Oracle file with clear comments and documentation explaining the logic behind each point.
- **Prohibition of AI Tools: The use of AI tools such as ChatGPT or similar platforms is strictly prohibited. Submissions will be evaluated with AI detection tools. Teams found using AI tools will receive a zero for this task. Ensure the work reflects your own understanding and effort.**