**DATABASE IMPLEMENTATION SYSTEMS**

Assignment1

Practical Implementation of PL/pgSQL with Real-World Project

**Total Marks: 25**

**Deadline: 20th June 2025**

Dear Students, you are required to design and implement a complete PostgreSQL database system for a real-world scenario of your choice

The project must demonstrate your ability to:

* Design and create a normalized relational schema
* Populate tables with meaningful sample data
* Write and use **PL/pgSQL functions**
* Implement **stored procedures**
* Use **loops** and **cursors** effectively

### ****Tasks and Requirements****

#### **1. Concept & Design (3 Marks)**

* Choose your own real-world mini-project.
* Create an **ER diagram** or **relational schema** (minimum 3 tables).
* Write a brief description of your system and what it does (1–2 paragraphs).

#### **2. Schema Creation (2 Marks)**

* Implement the database schema in PostgreSQL.
* Include relevant **constraints** (primary keys, foreign keys, NOT NULL, etc).

#### **3. Procedures (5 Marks)**

* Create **at least one PROCEDURE** that:
  + Modifies data (e.g., insert/update/delete)
  + Uses **transaction control**: BEGIN, COMMIT, ROLLBACK
  + Uses **IF conditions or CASE**

Example: a procedure to register a new order, but rollback if stock is unavailable.

#### **4. Functions (5 Marks)**

* Create **at least two FUNCTIONS**:
  + One that returns a single value (e.g., average score, total items sold)
  + One that returns a table (e.g., list of top customers or failed students)
  + Use **loop**, **IF**, or **RETURN NEXT**

#### **5. Cursors (4 Marks)**

* Use a **cursor** to iterate over rows and perform logic such as:
  + Processing each customer order
  + Generating messages or logs
  + Updating marks, points, or balance based on criteria

#### **6. Transactions: COMMIT and ROLLBACK (4 Marks)**

* Demonstrate the use of **transactions** in procedures:
  + Rollback if an error occurs (use EXCEPTION)
  + Commit if everything is successful
  + Clearly show an example of **error handling**

#### **7. Report and Script (2 Marks)**

* Submit a **short report** including (Max 7 pages):
  + Description of your system
  + Schema diagram
  + Description of each function and procedure
  + Screenshots and SQL script of each function and its results

**Choose the title and its requirements from the following list**

**1. School or University System**

**Example Project**: Student Result Management

Tables: student, course, enrolment, marks, grade

Features:

* + Function to calculate GPA
  + Procedure to register a student and auto-assign courses
  + Cursor to loop through students and update academic status
  + Transaction rollback on duplicate enrolment

**2. Hospital or Clinic Management**

**Example Project**: Patient Appointment & Billing

Tables: patient, doctor, appointment, treatment, invoice

Features:

* + Procedure to schedule an appointment
  + Function to calculate total bill per patient
  + Cursor to list patients treated by each doctor
  + Rollback if appointment slot is already booked

**3. Retail Store or Supermarket**

**Example Project**: Sales & Inventory System

Tables: product, category, sale, customer, inventory

Features:

* Function to get top 5 best-selling products
* Procedure to process a sale (update stock)
* Use cursor to re-calculate restock levels
* Rollback if stock falls below threshold

**4. Library System**

**Example Project**: Library Lending System

Tables: member, book, loan, fine, reservation

Features:

* Function to calculate overdue days and fines
* Procedure to issue a book and update the stock
* Cursor to notify members with overdue books
* Rollback if the book is already on loan

**5. Transportation or Delivery Services**

**Example Project**: Parcel Tracking and Delivery

Tables: customer, parcel, delivery, route, vehicle

Features:

* Function to track delivery status by parcel ID
* Procedure to assign a driver and a route
* Cursor to update parcel statuses daily
* Rollback if the delivery person is unavailable

**6. E-learning or Training Portal**

**Example Project**: Online Course Enrollment System

Tables: student, course, instructor, enrolment, assessment

Features:

* Function to calculate course completion rate
* Procedure to enrol students into available courses
* Cursor to evaluate course attendance
* Rollback on double-enrollment

**7. Hotel or Tourism Booking**

**Example Project**: Hotel Room Reservation System

Tables: guest, room, reservation, payment, service

Features:

* Function to check available rooms for a date range
* Procedure to book a room
* Cursor to auto-assign housekeeping tasks
* Rollback if payment fails

**8. Farm or Fishing Business**

**Example Project**: Produce Sales and Inventory System

Tables: product, harvest, sale, customer, inventory

Features:

* Function to calculate daily harvest totals
* Procedure to log a new sale and adjust inventory
* Cursor to report stock levels
* Rollback on invalid transaction

**9. Event or Conference Management**

**Example Project**: Event Registration & Schedule Planner

Tables: event, participant, session, speaker, registration

Features:

* Function to list available sessions per event
* Procedure to register participants
* Cursor to generate event reports
* Rollback on session overbooking

**10. Restaurant or Café Ordering System**

Tables: menu, order, item, customer, payment

Features:

* Function to calculate total bill with tax
* Procedure to place and track an order
* Cursor to process kitchen queue
* Rollback on unavailable ingredients

**11. Gym or Fitness Center Management**

Tables: member, trainer, session, payment, schedule

Features:

* Function to count active members per trainer
* Procedure to book training sessions
* Cursor to update the schedule weekly
* Rollback on time conflicts

**12. NGO or Charity Donation Tracker**

Tables: donor, donation, project, beneficiary, report

Features:

* Function to summarize donations per donor
* Procedure to allocate funds to projects
* Cursor to update project progress
* Rollback on duplicate donor entries

**13. Car Rental System**

Tables: vehicle, customer, rental, return, invoice

Features:

* Function to calculate the total rental cost
* Procedure to rent and return a car
* Cursor to track overdue rentals
* Rollback on double booking

**14. Cinema or Movie Ticket Booking**

Tables: movie, screen, showtime, ticket, customer

Features:

* Function to show seat availability
* Procedure to book tickets
* Cursor to update the show status
* Rollback on ticket conflict

**15. Online Marketplace System**

Tables: user, product, order, payment, review

Features:

* Function to get user ratings
* Procedure to place an order and update stock
* Cursor to list seller performance
* Rollback on payment error

CREATE OR REPLACE PROCEDURE place\_order(

p\_user\_id INT,

p\_product\_id INT,

p\_quantity INT

)

LANGUAGE plpgsql

AS $$

DECLARE

v\_price NUMERIC;

v\_total NUMERIC;

v\_order\_id INT;

BEGIN

-- Check stock

IF (SELECT stock FROM products WHERE product\_id = p\_product\_id) < p\_quantity THEN

RAISE EXCEPTION 'Not enough stock';

END IF;

-- Create order

INSERT INTO orders (user\_id) VALUES (p\_user\_id) RETURNING order\_id INTO v\_order\_id;

-- Get price

SELECT price INTO v\_price FROM products WHERE product\_id = p\_product\_id;

-- Insert order item

INSERT INTO order\_items (order\_id, product\_id, quantity, price)

VALUES (v\_order\_id, p\_product\_id, p\_quantity, v\_price);

-- Update stock

UPDATE products

SET stock = stock - p\_quantity

WHERE product\_id = p\_product\_id;

-- Create payment (dummy logic)

v\_total := p\_quantity \* v\_price;

INSERT INTO payments (order\_id, amount, status) VALUES (v\_order\_id, v\_total, 'Pending');

END;

$$;

DO $$

DECLARE

v\_orderid INT;

BEGIN

BEGIN

-- Create order

INSERT INTO orders (userid, productid, price, quantity)

VALUES (1, 1, 100, 2)

RETURNING orderid INTO v\_orderid;

-- Simulate error

RAISE EXCEPTION 'Payment failed';

-- Insert payment (won't happen)

INSERT INTO payments (orderid, amount, date, status)

VALUES (v\_orderid, 200, CURRENT\_TIMESTAMP, 'Success');

EXCEPTION

WHEN OTHERS THEN

RAISE NOTICE 'Rolling back: %', SQLERRM;

ROLLBACK;

END;

END;

$$;