**NUMBER 1**

A Database Management System (DBMS) is software that uses a standard method to store and organize data. It allows users to create, read, update, and delete data in a database in a systematic way.

For Crane Management Services (CMS), a DBMS can improve operations in the following ways:

1. **Centralized Data:** All property listings, client info, transactions, and agent details are stored in one place.
2. **Data Integrity:** Ensures accurate and consistent data by using constraints like primary keys.
3. **Automation:** Automates tasks like report generation, tracking performance, and matching client preferences.
4. **Better Reporting:** Generates real-time reports on sales, rentals, and leases for quick decision-making.
5. **Collaboration:** Multiple users can access and work on data simultaneously without conflicts.
6. **Scalability:** Can handle more data as CMS grows without losing performance.

B); **A DBMS can help ensure data security and access control through several mechanisms**:

1. **Authentication:** Only authorized users can access the DBMS with valid credentials.
2. **Authorization:** Controls what data each user can access and what actions they can perform.
3. **Encryption:** Protects sensitive data both during transfer and when stored.
4. **Audit Trails:** Keeps logs of user activities to monitor and detect any suspicious behaviour.
5. **Backup and Recovery:** Regular data backups ensure recovery in case of issues.

**c) Identify the types of database users and their roles/privileges involved in the real estate business.**

In a real estate business like CMS, the following types of database users and their roles/privileges can be identified:

1. **Database Administrators (DBAs):** Manage the whole database system. They set up user accounts, define security policies, optimize performance, and ensure data accuracy.
2. **Data Analysts:** Read and analyse data to create reports and insights. They need permissions to view and query the data.
3. **Real Estate Agents:** Update property listings, manage client information, and log transactions. They need to be able to create, read, update, and delete relevant data.
4. **Clients:** Have limited access to view property listings and their own transaction history. This boosts transparency and trust.
5. **Support Staff:** Perform administrative tasks like data entry and updating records. They need access to specific parts of the database to do their jobs efficiently.

**d) List and explain the role of the following database languages in the real estate DBMS; i. DDL (Data Definition Language).**

**DDL (Data Definition Language):**

* DDL is part of SQL used to define and manage the structure of a database.

**Role in Real Estate DBMS:**

1. **Creating Tables:** Define tables for properties, agents, clients, and transactions using the CREATE TABLE statement.
2. **Altering Tables:** Modify existing tables to meet new requirements, like adding a new column.
3. **Dropping Tables:** Remove tables that are no longer needed to keep the database organized.
4. **Defining Constraints:** Ensure data integrity with primary keys, foreign keys, and other constraints.

**Example DDL Command:**

SQL

CREATE TABLE Properties (

PropertyID INT PRIMARY KEY,

Address VARCHAR (255) NOT NULL,

City VARCHAR (100) NOT NULL,

Type VARCHAR (50) CHECK (Type IN ('Residential', 'Commercial', 'Industrial')),

Size INT NOT NULL,

Price DECIMAL (10,2) NOT NULL,

Status VARCHAR (20) CHECK (Status IN ('Available', 'Sold', 'Rented'))

);

**Other SQL Subsets:**

* **DML (Data Manipulation Language):** Manages data with commands like SELECT, INSERT, UPDATE, and DELETE.
* **DCL (Data Control Language):** Controls access with commands like GRANT and REVOKE.

**Database Normalization:**

* **1NF (First Normal Form):** Eliminate repeating groups.
* **2NF (Second Normal Form):** Remove partial dependencies.
* **3NF (Third Normal Form):** Remove transitive dependencies.

**Create a Properties Table**

SQL

CREATE TABLE Properties (

PropertyID INT PRIMARY KEY,

Address VARCHAR (255) NOT NULL,

City VARCHAR (100) NOT NULL,

Type VARCHAR (50),

Size INT NOT NULL,

Price DECIMAL (10,2) NOT NULL,

Status VARCHAR (20)

);

**ii. Insert Sample Records**

SQL

INSERT INTO Properties (PropertyID, Address, City, Type, Size, Price, Status)

VALUES

(1, '123 Main St', 'Kampala', 'Residential', 2000, 100000.00, 'Available'),

(2, '456 Oak St', 'Kampala', 'Commercial', 3000, 200000.00, 'Sold'),

(3, '789 Pine St', 'Kampala', 'Industrial', 5000, 300000.00, 'Rented');

**iii. Retrieve Available Properties in a City**

SQL

SELECT \* FROM Properties

WHERE City = 'Kampala' AND Status = 'Available';

**iv. Update Property Status**

SQL

UPDATE Properties

SET Status = 'Sold'

WHERE PropertyID = 1;

**4. Handling Non-Existent PropertyID in Transactions**

The foreign key constraint in the Transactions table ensures that a PropertyID must exist in the Properties table, preventing issues. If a transaction is recorded with a non-existent PropertyID, the database will reject it.

**5. Simplified Custom Constraints**

**i. Property Status**

SQL

ALTER TABLE Properties

ADD CONSTRAINT chk\_status CHECK (Status IN ('Available', 'Sold', 'Rented'));

**ii. Agent Commission Rate**

SQL

ALTER TABLE Agents

ADD CONSTRAINT chk\_commissionrate CHECK (Commission Rate BETWEEN 1 AND 15);

**iii. Transaction Amount**

SQL

CREATE TRIGGER trg\_transaction\_amount

BEFORE INSERT OR UPDATE ON Transactions

FOR EACH ROW

BEGIN

DECLARE property\_price DECIMAL (10,2);

SELECT Price INTO property\_price FROM Properties WHERE PropertyID = NEW.PropertyID;

IF NEW.Amount > property\_price THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Transaction amount cannot exceed property price';

END IF;

END;

**NUMBER 2**

**1. 1. Primary Entities and Relationships:**

* Patient: Patients receiving treatment.
* Visit Record: Details of each visit a patient makes to the hospital.
* Treatment: Information about treatments administered.
* Lab Test: Records of laboratory tests conducted.
* Facility: Different facilities within the hospital.
* Resource: Medical supplies and resources.
* User Role: Defines roles within the system for secure access.

**Relationships:**

* A Patient can have multiple Visit Records.
* A Visit Record can have multiple Treatments.
* A Lab Test can be linked to a Visit Record.
* A Facility can have multiple Resources.
* User Role is associated with user access control**.**

**2. Potential Errors in ERD:**

* Missing Foreign Keys: Ensure all relationships are represented correctly.
* Redundant Entities: Merge any duplicate entities.
* Inconsistent Naming: Use consistent naming conventions.
* Incomplete Attributes: Include all necessary details.
* Cardinality Issues: Ensure correct one-to-one, one-to-many, many-to-many relationships.

**3. Patient Data and Visit Record Relationship:**

* One-to-Many Relationship: A patient can have multiple visit records.
* Importance: Tracks multiple visits, monitors disease progression, treatment outcomes, and ensures continuity of care.

**4. Role of User Role Table:**

* Defines Roles: Admin, Doctor, Nurse, Lab Technician, etc.
* Ensures Security: Users access only necessary data and functions**.**

**5. Managing Low Medical Supplies:**

* Resource Table: Tracks current inventory.
* Supply Chain Table: Information about suppliers and procurement.
* Resolution: Identify low supplies and procure from suppliers**.**

**6. Redraw ERD in Chen's Notation:**

* **Use rectangles for entities, diamonds for relationships, and ovals for attributes.**

**7. Expanding to Non-Malaria Diseases:**

* Patient: No change needed.
* Visit Record: Add attribute for disease treated.
* Treatment: Ensure it accommodates various treatments.
* Lab Test: Include tests for different diseases.
* Resource: Track supplies for other diseases**.**

**8. SQL Code for Database Tables:**

**SQL**

CREATE TABLE Patient (

PatientID INT PRIMARY KEY,

Name VARCHAR (100) NOT NULL,

Age INT,

Gender VARCHAR (10),

ContactInfo VARCHAR (255)

);

CREATE TABLE VisitRecord (

VisitID INT PRIMARY KEY,

PatientID INT,

VisitDate DATE NOT NULL,

Diagnosis VARCHAR (255),

Treatment Plan VARCHAR (255),

FOREIGN KEY (PatientID) REFERENCES Patient (PatientID)

);

CREATE TABLE Treatment (

TreatmentID INT PRIMARY KEY,

VisitID INT,

Medication VARCHAR (255),

Dosage VARCHAR (50),

FOREIGN KEY (VisitID) REFERENCES VisitRecord (VisitID)

);

CREATE TABLE Lab Test (

TestID INT PRIMARY KEY,

VisitID INT,

Test Type VARCHAR(100),

Test Result VARCHAR(255),

FOREIGN KEY (VisitID) REFERENCES VisitRecord (VisitID)

);

CREATE TABLE Facility (

FacilityID INT PRIMARY KEY,

Facility Name VARCHAR (100),

Location VARCHAR (255)

);

CREATE TABLE Resource (

ResourceID INT PRIMARY KEY,

FacilityID INT,

Resource Name VARCHAR (100),

Quantity INT,

FOREIGN KEY (FacilityID) REFERENCES Facility (FacilityID)

);

CREATE TABLE User Role (

RoleID INT PRIMARY KEY,

Role Name VARCHAR(50),

Permissions VARCHAR (255)

);

Be sure to upload the SQL script to your GitHub account and include the GitHub link to your coursework document. Additionally, record a video demonstrating the ERD (Chen's Notation) and the SQL script operation, upload it to your YouTube channel, and include the link in your submission.