DBMS Project

Government Database Management System

A Project Report
On
The Government Database Management System

Course Name: Database Management System

Student's Name:

Name	Seat No	PRN
Suthar Yogesh Omprakash	486176	8022054344
Shah Bhavyakumar Smitalbhai	486166	8022053501
Prajapati Jatinbhai Rameshbhai	486155	8022054091

INDEX

Description

The Government Database Management System (GDBMS) is an extensive project designed to revolutionize the management of governmental data across multiple sectors. It serves as a centralized platform for storing, accessing, and analyzing data related to various government departments, initiatives, and citizen services. By integrating disparate databases and streamlining data management processes, the GDBMS aims to enhance administrative efficiency, facilitate evidence-based decision-making, and improve service delivery to citizens.

The GDBMS comprises several interconnected databases, each catering to specific domains and functionalities within the government. These databases are structured to capture and organize information effectively, ensuring seamless access and retrieval by authorized users. Let's explore the key components of the system in more detail:

Government_Database:

This database stores essential information about government departments, including unique department IDs and names.
It serves as a foundational component of the GDBMS, providing a framework for organizing departmental data.

Citizen_Database:

The Citizen Database manages data related to individual citizens, including unique identification numbers (UIDs), PAN card details, Aadhar card information, and demographic details such as names, addresses, gender, and contact information.
It facilitates the creation and maintenance of comprehensive citizen profiles, enabling efficient delivery of government services and benefits.

Infrastructure:

The Infrastructure database tracks infrastructure projects undertaken by government departments, including construction projects, tender details, company information, and project statuses.
It also records performance metrics such as ratings and completion times, facilitating project monitoring and evaluation.

Travel_And_Tourism:

This database focuses on managing tourism-related information, including tourist attractions, tourism development projects, budgets, and timelines.
It aims to promote tourism initiatives, attract visitors, and support the growth of the tourism sector.

Education:

The Education database handles data related to educational institutions, schemes, budgets, and queries.

It supports educational planning and policymaking by providing insights into enrollment trends, scheme effectiveness, and resource allocation.

Communications And IT:

This database manages information related to personnel, equipment, and projects in the communications and information technology sectors.
It facilitates efficient communication infrastructure management and technology deployment across government agencies.

Ministry:

The Ministry database provides details about government ministers, their departments, budgets, and governance areas.
It serves as a repository of ministerial information, supporting coordination and collaboration among government agencies.

Agriculture:

The Agriculture database focuses on agricultural projects, farmer information, schemes, and related queries.
It aims to promote agricultural development, support farmers, and ensure food security through targeted interventions.

Defence:

The Defence database manages data related to personnel, equipment, missions, and outcomes in the defense sector.
It plays a crucial role in national security planning, defense preparedness, and strategic decision-making.

Election Commission:

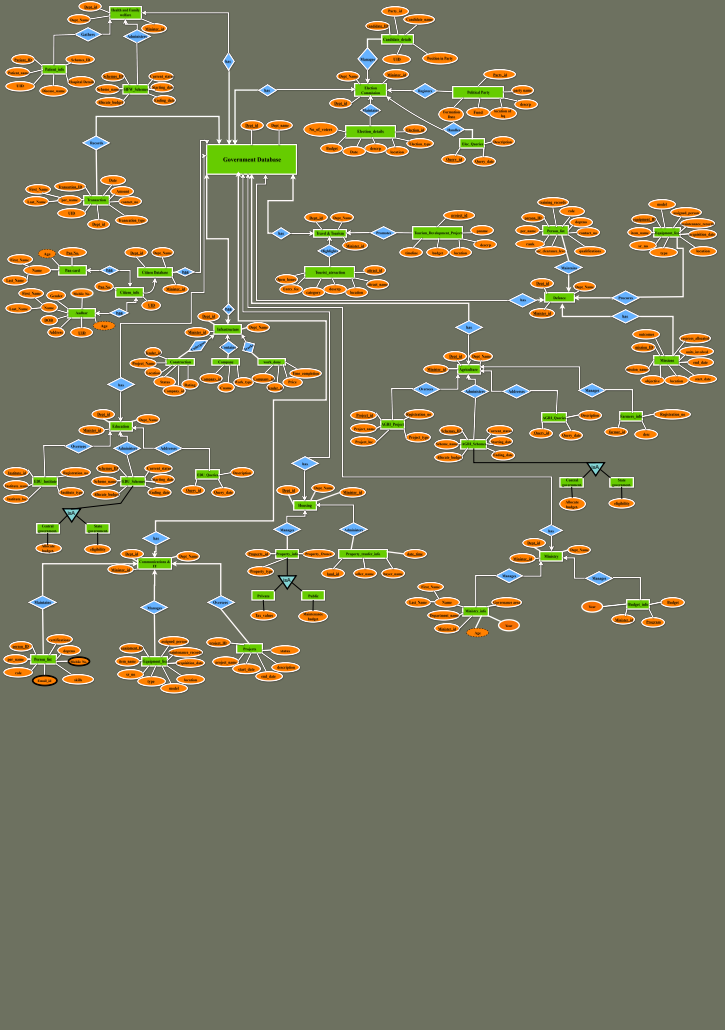
This database stores information about candidates, political parties, elections, and queries related to electoral processes.
It supports the conduct of free and fair elections, ensuring transparency and accountability in the electoral process.

Health and Family Welfare:

The Health and Family Welfare database tracks patient information, healthcare schemes, and transactions in the health sector.
It aims to improve public health outcomes, enhance healthcare access, and promote wellness initiatives.

Housing:

The Housing database manages property information, including transfers and budgets for maintenance and taxation.
It supports urban planning efforts, housing development initiatives, and property management functions.


Government Database

TABLES

1) Government_Database(dept_id,dept_name)

2) Citizen_Database(Dept_id, Minister_id)

- Citizen_info(UID, Pan_No, criminal_record, Dept_id, Minister_id)
- Aadhar(UID, Name(First_Name, Last_Name) , DOB, Address, Gender, Age, Mobile_No)
- Pan_card(Pan_No, Name(First_Name, Last_Name),Age)

3) Infrastructure(Dept_id, Minister_id)

- Construction(tender_id, Dept_id, Minister_id ,project_name, location, status, company_id, rating)
- Company(company_id,Dept_id,Minister_id,company_name, work_type)
- Work_done(company_id, tender_id, Dept_id, Minister_id price, time_completion)

4) Travel_And_Tourism(Dept_id, Minister_id)

- Tourist_attraction(attract_id, Dept_id, Minister_id attract_name, location, descrp, category, entry_fee, open_hours)
- Tourism_Development_Project(project_id, Dept_id, Minister_id pname, descrp, location, budget, timeline)

5) Education(Dept_id, Minister_id)

- Edu_Institute(Institute_id, Institute_name, Institute_loc, Registration_no, Institute_type, Dept_id, Minister_id)
- Edu_schemes(Schemes_id, Scheme_name, Allocate_budget, Current_status, Starting_date, Ending_date, Dept_id, Minister_id)
    - Central_government(Dept_id, Minister_id, Scheme_id, Allocate_budget)
    - State_government(Dept_id, Minister_id, Scheme_id, Eligibility)
- Edu_Queries(Query_id, Query_date, Description, Dept_id, Minister_id)

6) Communications_And_IT(Dept_id, Minister_id)

- Person_list(person_id, per_name, deptno, role, email_id, skills, contact_no, certification, Dept_id, Minister_id)
- Person_contact(person_id, contact_no, Dept_id, Minister_id)
- Person_contact(person_id, email_id, Dept_id, Minister_id)
- Equipment_list(equipment_id, item_name, sr_no, type, model, location, acquisition_date, maintenance_records, assigned_person, Dept_id, Minister_id)
- Projects(project_id, project_name, start_date, end_date, desc, status, Dept_id, Minister_id)

7) Ministry(Dept_id, Minister_id)

- Ministry_info(Minister_id, , Dept_id, Name(First_Name, Last_Name) , department_name, Age, Year, governance_area)
- Budget_info(Minister_id, , Dept_id, Program, Budget, Year)

8) Agriculture(Dept_id, Minister_id)

- Agri_project(project_id, project_name, project_loc, registration_no, project_type, Dept_id, Minister_id)
- Farmers_info(farmer_id, desc, registration_no, Dept_id, Minister_id)
- Agri_schemes(Schemes_id, Scheme_name, Allocate_budget, current_status, Starting_date, Ending_date, Dept_id, Minister_id)
    - Central_government(Dept_id, Minister_id, Scheme_id, Allocate_budget)
    - State_government(Dept_id, Minister_id, Scheme_id, Eligibility)
- Agri_queries(Query_id, Query_date, Description, Dept_id, Minister_id)

9) Defence(Dept_id, Minister_id)

- Person_list(person_id, per_name, deptno, role, rank, sec_clearance_level, qualifications, contact_no, training_records, Dept_id, Minister_id)
- Equipment_list(equipment_id, item_name, sr_no, type, model, location, acquisition_date, maintenance_records, assigned_person, Dept_id, Minister_id)
- Missions(mission_id, mission_name, objective, location, start_date, end_date, units_involved, resources_allocated, outcomes, Dept_id, Minister_id)

10) Election Commision(Dept_id, Dept_name, Minister_id)

- Candidate_details(candidate_id, candidate_name, UID, Party_id, Position_in_party, Dept_id, Minister_id)

- Political_Party(Party_id, Party_name, descrp, location_of_hq, fund, formation_date, Dept_id, Minister_id)
- Election_details(election_id, election_type, location, descrp, date, budget, No_of_voters, Dept_id, Minister_id)
- Elec_Queries(Query_id, Query_date, Description, Dept_id, Minister_id)

11) Health and Family welfare(Dept_id, Minister_id)

- Patient_info(patient_id, patient_name, UID, Disease_name, Hospital_details, schemes_id, Dept_id, Minister_id)
- HFW_schemes(schemes_id, scheme_name, allocate_budget, current_status, Starting_date, Ending_date, Dept_id, Minister_id)
- Transaction(transaction_id, per_name, UID, transaction_type, amount, contact_no, date, Dept_id, Minister_id)

12) Housing(Dept_id, Minister_id)

- Property_info(property_id, property_type, property_owner, Dept_id, Minister_id)
    - Private(property_id, Tax_values, Dept_id, Minister_id)
    - Public(property_id, Maintenance_budget, Dept_id, Minister_id)

- Property_transfer_info(land_id, seller_name, buyer_name, date_time, Dept_id, Minister_id)

DDLS

1) Government_Database

```
CREATE TABLE Government_Database (
   dept_id INT PRIMARY KEY,
);
```

2) Citizen_Database

```
CREATE TABLE Citizen_Database (
   Dept_id INT,
   Minister_id INT,
   PRIMARY KEY (Dept_id, Minister_id)
);

CREATE TABLE Citizen_info (
   UID INT PRIMARY KEY,
   Pan_No VARCHAR(100),
   Minister_id INT,
   Dept_id INT,
   FOREIGN KEY (Dept_id, Minister_id) REFERENCES Citizen_Database(Dept_id,
Minister_id)
);

CREATE TABLE Aadhar (
   UID INT PRIMARY KEY,
   First_Name VARCHAR(100),
   Last_Name VARCHAR(100),
   DOB DATE,
   Address VARCHAR(255),
   Gender VARCHAR(10),
   Mobile_No VARCHAR(20)
);

CREATE TABLE Pan_card (
   Pan_No VARCHAR(100) PRIMARY KEY,
   First_Name VARCHAR(100),
   Last_Name VARCHAR(100)
);
```

3) Infrastructure

```
CREATE TABLE Infrastructure (
   Dept_id INT,
   Minister_id INT,
   PRIMARY KEY (Dept_id, Minister_id)
);

CREATE TABLE Construction (
   Dept_id INT,
   Minister_id INT,
   tender_id INT PRIMARY KEY,
   project_name VARCHAR(255),
   location VARCHAR(255),
   status VARCHAR(50),
   company_id INT,
   rating DECIMAL(5,2),
   FOREIGN KEY (Dept_id, Minister_id) REFERENCES Infrastructure(Dept_id,
Minister_id)
);

CREATE TABLE Company (
   Dept_id INT,
   Minister_id INT,
   company_id INT PRIMARY KEY,
   company_name VARCHAR(100),
   work_type VARCHAR(100),
   FOREIGN KEY (Dept_id, Minister_id) REFERENCES Infrastructure(Dept_id,
Minister_id)
);

CREATE TABLE Work_done (
   Dept_id INT,
   Minister_id INT,
   company_id INT,
   tender_id INT,
   price DECIMAL(15,2),
   time_completion DATETIME,
   PRIMARY KEY (Dept_id, Minister_id, company_id, tender_id),
   FOREIGN KEY (Dept_id, Minister_id) REFERENCES Infrastructure(Dept_id,
Minister_id),
   FOREIGN KEY (Dept_id, Minister_id, company_id) REFERENCES Company(Dept_id,
Minister_id, company_id),
   FOREIGN KEY (Dept_id, Minister_id, tender_id) REFERENCES Construction(Dept_id,
Minister_id, tender_id)
);
```

4) Travel_And_Tourism

```
CREATE TABLE Travel_And_Tourism (
   Dept_id INT,
   Minister_id INT,
   PRIMARY KEY (Dept_id, Minister_id)
);

CREATE TABLE Tourist_attraction (
   Dept_id INT,
   Minister_id INT,
   attract_id INT PRIMARY KEY,
   attract_name VARCHAR(255),
   location VARCHAR(255),
   descrp TEXT,
   category VARCHAR(100),
   entry_fee DECIMAL(10,2),
   open_hours VARCHAR(50),
   FOREIGN KEY (Dept_id, Minister_id) REFERENCES Travel_And_Tourism(Dept_id,
Minister_id)
);

CREATE TABLE Tourism_Development_Project (
   Dept_id INT,
   Minister_id INT,
   project_id INT PRIMARY KEY,
   pname VARCHAR(255),
   descrp TEXT,
   location VARCHAR(255),
   budget DECIMAL(15,2),
   timeline VARCHAR(100),
   FOREIGN KEY (Dept_id, Minister_id) REFERENCES Travel_And_Tourism(Dept_id,
Minister_id)
);
```


5) Education

```
CREATE TABLE Education (
   Dept_id INT,
   Minister_id INT,
   PRIMARY KEY (Dept_id, Minister_id)
);

CREATE TABLE Edu_Institute (
   Dept_id INT,
   Minister_id INT,
   Institute_id INT PRIMARY KEY,
   Institute_name VARCHAR(255),
   Institstute_loc VARCHAR(255),
```

```
    Registration_no VARCHAR(100),
    Institute_type VARCHAR(100),
    FOREIGN KEY (Dept_id, Minister_id) REFERENCES Education(Dept_id, Minister_id)
);

CREATE TABLE Edu_schemes (
    Dept_id INT,
    Minister_id INT,
    Schemes_id INT PRIMARY KEY,
    Scheme_name VARCHAR(255),
    Allocate_budget DECIMAL(15,2),
    Current_status VARCHAR(100),
    Starting_date DATE,
    Ending_date DATE,
    FOREIGN KEY (Dept_id, Minister_id) REFERENCES Education(Dept_id, Minister_id)
);

CREATE TABLE Central_government (
    Dept_id INT,
    Minister_id INT,
    Schemes_id INT,
    property_id INT,
    Allocate_budget DECIMAL(15,2),
    PRIMARY KEY (Dept_id, Minister_id, Schemes_id),
    FOREIGN KEY (Dept_id, Minister_id) REFERENCES Education(Dept_id, Minister_id),
    FOREIGN KEY (Dept_id, Minister_id, Schemes_id) REFERENCES
Edu_schemes(Dept_id, Minister_id, Schemes_id)
);

CREATE TABLE State_government (
    Dept_id INT,
    Minister_id INT,
    Schemes_id INT,
    property_id INT,
    Eligibility VARCHAR(255),
    PRIMARY KEY (Dept_id, Minister_id, Schemes_id),
    FOREIGN KEY (Dept_id, Minister_id) REFERENCES Education(Dept_id, Minister_id),
    FOREIGN KEY (Dept_id, Minister_id, Schemes_id) REFERENCES
Edu_schemes(Dept_id, Minister_id, Schemes_id)
);

CREATE TABLE Edu_Queries (
    Dept_id INT,
    Minister_id INT,
    Query_id INT PRIMARY KEY,
    Query_date DATE,
    Description TEXT,
```

```
    FOREIGN KEY (Dept_id, Minister_id) REFERENCES Education(Dept_id, Minister_id)
);
```

6) Communications_And_IT

```
CREATE TABLE Communications_And_IT (
    Dept_id INT,
    Minister_id INT,
    PRIMARY KEY (Dept_id, Minister_id)
);

CREATE TABLE Person_list (
    Dept_id INT,
    Minister_id INT,
    person_id INT PRIMARY KEY,
    per_name VARCHAR(255),
    deptno INT,
    role VARCHAR(100),
    email_id VARCHAR(255),
    skills TEXT,
    contact_no VARCHAR(20),
    certification VARCHAR(255),
    FOREIGN KEY (Dept_id, Minister_id) REFERENCES
Communications_And_IT(Dept_id, Minister_id)
);

CREATE TABLE Person_contact (
    Dept_id INT,
    Minister_id INT,
    person_id INT,
    contact_no VARCHAR(20),
    FOREIGN KEY (Dept_id, Minister_id, person_id) REFERENCES Person_list(Dept_id,
Minister_id, person_id)
);

CREATE TABLE Person_email (
    Dept_id INT,
    Minister_id INT,
    person_id INT,
    email_id VARCHAR(255),
    FOREIGN KEY (Dept_id, Minister_id, person_id) REFERENCES Person_list(Dept_id,
Minister_id, person_id)
);


CREATE TABLE Equipment_list (
    Dept_id INT,
    Minister_id INT,
```

```
    equipment_id INT PRIMARY KEY,
    item_name VARCHAR(255),
    sr_no VARCHAR(100),
    type VARCHAR(100),
    model VARCHAR(100),
    location VARCHAR(255),
    acquisition_date DATE,
    maintenance_records TEXT,
    assigned_person INT,
    FOREIGN KEY (Dept_id, Minister_id) REFERENCES
Communications_And_IT(Dept_id, Minister_id)
);

CREATE TABLE Projects (
    Dept_id INT,
    Minister_id INT,
    project_id INT PRIMARY KEY,
    project_name VARCHAR(255),
    start_date DATE,
    end_date DATE,
    desc TEXT,
    status VARCHAR(100),
    FOREIGN KEY (Dept_id, Minister_id) REFERENCES
Communications_And_IT(Dept_id, Minister_id)
);
```

7) Ministry

```
CREATE TABLE Ministry (
    Dept_id INT,
    Minister_id INT,
    PRIMARY KEY (Dept_id, Minister_id)
);

CREATE TABLE Ministry_info (
    Dept_id INT,
    Minister_id INT,
    First_Name VARCHAR(100),
    Last_Name VARCHAR(100),
    department_name VARCHAR(255),
    Year INT,
    governance_area VARCHAR(255),
    PRIMARY KEY (Dept_id, Minister_id)
);

CREATE TABLE Budget_info (
    Dept_id INT,
    Minister_id INT,
```

```
  Program VARCHAR(255),
  Budget DECIMAL(15,2),
  Year INT,
  PRIMARY KEY (Dept_id, Minister_id)
);
```

8) Agriculture

```
CREATE TABLE Agriculture (
  Dept_id INT,
  Minister_id INT,
  PRIMARY KEY (Dept_id, Minister_id)
);

CREATE TABLE Agri_project (
  Dept_id INT,
  Minister_id INT,
  project_id INT PRIMARY KEY,
  project_name VARCHAR(255),
  project_loc VARCHAR(255),
  registration_no VARCHAR(100),
  project_type VARCHAR(100),
  FOREIGN KEY (Dept_id, Minister_id) REFERENCES Agriculture(Dept_id, Minister_id)
);

CREATE TABLE Farmers_info (
  Dept_id INT,
  Minister_id INT,
  farmer_id INT PRIMARY KEY,
  desc TEXT,
  registration_no VARCHAR(100),
  FOREIGN KEY (Dept_id, Minister_id) REFERENCES Agriculture(Dept_id, Minister_id)
);

CREATE TABLE Agri_schemes (
  Dept_id INT,
  Minister_id INT,
  Schemes_id INT PRIMARY KEY,
  Scheme_name VARCHAR(255),
  Allocate_budget DECIMAL(15,2),
  current_status VARCHAR(100),
  Starting_date DATE,
  Ending_date DATE,
  FOREIGN KEY (Dept_id, Minister_id) REFERENCES Agriculture(Dept_id, Minister_id)
);

CREATE TABLE Central_government_agri (
  Dept_id INT,
```

```
    Minister_id INT,
    Schemes_id INT,
    Allocate_budget DECIMAL(15,2),
    PRIMARY KEY (Dept_id, Minister_id, Schemes_id),
    FOREIGN KEY (Dept_id, Minister_id) REFERENCES Agriculture(Dept_id,
Minister_id),
    FOREIGN KEY (Dept_id, Minister_id, Schemes_id) REFERENCES
Agri_schemes(Dept_id, Minister_id, Schemes_id)
);

CREATE TABLE State_government_agri (
    Dept_id INT,
    Minister_id INT,
    Schemes_id INT,
    Eligibility VARCHAR(255),
    PRIMARY KEY (Dept_id, Minister_id, Schemes_id),
    FOREIGN KEY (Dept_id, Minister_id) REFERENCES Agriculture(Dept_id,
Minister_id),
    FOREIGN KEY (Dept_id, Minister_id, Schemes_id) REFERENCES
Agri_schemes(Dept_id, Minister_id, Schemes_id)
);

CREATE TABLE Agri_queries (
    Dept_id INT,
    Minister_id INT,
    Query_id INT PRIMARY KEY,
    Query_date DATE,
    Description TEXT,
    FOREIGN KEY (Dept_id, Minister_id) REFERENCES Agriculture(Dept_id, Minister_id)
);
```


9) Defence

```
CREATE TABLE Defence (
    Dept_id INT,
    Minister_id INT,
    PRIMARY KEY (Dept_id, Minister_id)
);

CREATE TABLE Defence_Person_list (
    Dept_id INT,
    Minister_id INT,
    person_id INT PRIMARY KEY,
    per_name VARCHAR(255),
    deptno INT,
    role VARCHAR(100),
    rank VARCHAR(100),
    sec_clearance_level VARCHAR(100),
```

```
    qualifications TEXT,
    contact_no VARCHAR(20),
    training_records TEXT,
    FOREIGN KEY (Dept_id, Minister_id) REFERENCES Defence(Dept_id, Minister_id)
);

CREATE TABLE Defence_Equipment_list (
    Dept_id INT,
    Minister_id INT,
    equipment_id INT PRIMARY KEY,
    item_name VARCHAR(255),
    sr_no VARCHAR(100),
    type VARCHAR(100),
    model VARCHAR(100),
    location VARCHAR(255),
    acquisition_date DATE,
    maintenance_records TEXT,
    assigned_person INT,
    FOREIGN KEY (Dept_id, Minister_id) REFERENCES Defence(Dept_id, Minister_id)
);

CREATE TABLE Missions (
    Dept_id INT,
    Minister_id INT,
    mission_id INT PRIMARY KEY,
    mission_name VARCHAR(255),
    objective TEXT,
    location VARCHAR(255),
    start_date DATE,
    end_date DATE,
    units_involved VARCHAR(255),
    resources_allocated VARCHAR(255),
    outcomes TEXT,
    FOREIGN KEY (Dept_id, Minister_id) REFERENCES Defence(Dept_id, Minister_id)
);
```


10) Election Commission

```
CREATE TABLE Election_Commission (
    Dept_id INT,
    Minister_id INT,
    PRIMARY KEY (Dept_id, Minister_id)
);

CREATE TABLE Candidate_details (
    Dept_id INT,
    Minister_id INT,
```

```
  candidate_id INT PRIMARY KEY,
  candidate_name VARCHAR(255),
  UID INT,
  Party_id INT,
  Position_in_party VARCHAR(100),
  FOREIGN KEY (Dept_id, Minister_id) REFERENCES Election_Commission(Dept_id,
Minister_id)
);

CREATE TABLE Political_Party (
  Dept_id INT,
  Minister_id INT,
  Party_id INT PRIMARY KEY,
  Party_name VARCHAR(255),
  descrp TEXT,
  location_of_hq VARCHAR(255),
  fund DECIMAL(15,2),
  formation_date DATE,
  FOREIGN KEY (Dept_id, Minister_id) REFERENCES Election_Commission(Dept_id,
Minister_id)
);

CREATE TABLE Election_details (
  Dept_id INT,
  Minister_id INT,
  election_id INT PRIMARY KEY,
  election_type VARCHAR(100),
  location VARCHAR(255),
  descrp TEXT,
  date DATE,
  budget DECIMAL(15,2),
  No_of_voters INT,
  FOREIGN KEY (Dept_id, Minister_id) REFERENCES Election_Commission(Dept_id,
Minister_id)
);

CREATE TABLE Elec_Queries (
  Dept_id INT,
  Minister_id INT,
  Query_id INT PRIMARY KEY,
  Query_date DATE,
  Description TEXT,
  FOREIGN KEY (Dept_id, Minister_id) REFERENCES Election_Commission(Dept_id,
Minister_id)
);
```

11) Health and Family Welfare

```
CREATE TABLE Health_Family_Welfare (
  Dept_id INT,
  Minister_id INT,
  PRIMARY KEY (Dept_id, Minister_id)
);

CREATE TABLE Patient_info (
  Dept_id INT,
  Minister_id INT,
  patient_id INT PRIMARY KEY,
  patient_name VARCHAR(255),
  UID INT,
  Disease_name VARCHAR(255),
  Hospital_details VARCHAR(255),
  schemes_id INT,
  FOREIGN KEY (Dept_id, Minister_id) REFERENCES Health_Family_Welfare(Dept_id,
Minister_id)
);

CREATE TABLE HFW_schemes (
  Dept_id INT,
  Minister_id INT,
  schemes_id INT PRIMARY KEY,
  scheme_name VARCHAR(255),
  allocate_budget DECIMAL(15,2),
  current_status VARCHAR(100),
  Starting_date DATE,
  Ending_date DATE,
  FOREIGN KEY (Dept_id, Minister_id) REFERENCES Health_Family_Welfare(Dept_id,
Minister_id)
);
```


12) Housing

```
CREATE TABLE Housing (
  Dept_id INT,
  Minister_id INT,
  PRIMARY KEY (Dept_id, Minister_id)
);

CREATE TABLE Property_info (
  Dept_id INT,
  Minister_id INT,
  property_id INT PRIMARY KEY,
  property_type VARCHAR(100),
  property_owner VARCHAR(100),
  FOREIGN KEY (Dept_id, Minister_id) REFERENCES Housing(Dept_id, Minister_id)
```

```
);

CREATE TABLE Private (
   Dept_id INT,
   Minister_id INT,
   property_id INT PRIMARY KEY,
   Tax_values DECIMAL(15, 2),
   FOREIGN KEY (Dept_id, Minister_id, property_id) REFERENCES
Property_info(Dept_id, Minister_id, property_id)
);

CREATE TABLE Public (
   Dept_id INT,
   Minister_id INT,
   property_id INT PRIMARY KEY,
   Maintenance_budget DECIMAL(15, 2),
   FOREIGN KEY (Dept_id, Minister_id, property_id) REFERENCES
Property_info(Dept_id, Minister_id, property_id)
);

CREATE TABLE Property_transfer_info (
   Dept_id INT,
   Minister_id INT,
   land_id INT PRIMARY KEY,
   seller_name VARCHAR(100),
   buyer_name VARCHAR(100),
   date_time TIMESTAMP,
   FOREIGN KEY (Dept_id, Minister_id) REFERENCES Housing(Dept_id, Minister_id)
);
```

NORMALIZATION

1) Government Database:

1NF: Unique ID and name for each department.
2NF: No partial dependencies; all attributes fully depend on the department ID.
3NF: No transitive dependencies; attributes depend only on the department ID.

2) Citizen_Database:

1NF: Each table has unique identifiers to ensure there are no duplicate rows.
2NF: No partial dependencies; all attributes in each table are dependent on the primary key.
3NF: No transitive dependencies; attributes depend only on the primary keys.

Citizen_info:

1NF: Each attribute has atomic values, ensuring there are no multi-valued attributes.
2NF: All attributes depend on the UID and Pan_No, which serve as the composite primary key.
3NF: No transitive dependencies; attributes like Minister_id and Dept_id are directly dependent on the primary key.

Aadhar:

1NF: Atomic values for each attribute, preventing multi-valued attributes.
2NF: All attributes depend on the UID, which serves as the primary key.
3NF: No transitive dependencies; attributes like Name, DOB, Address, Gender, and Mobile_No directly depend on the UID.

Pan_card:

1NF: Atomic values for each attribute, ensuring there are no multi-valued attributes.

2NF: All attributes depend on the Pan_No, which serves as the primary key.
3NF: No transitive dependencies; attributes like Name directly depend on the Pan_No.

3) Infrastructure:

1NF: Each table has unique identifiers to ensure there are no duplicate rows.
2NF: No partial dependencies; all attributes in each table are dependent on the primary key.
3NF: No transitive dependencies; attributes depend only on the primary keys.

Construction:

1NF: Each attribute has atomic values, ensuring there are no multi-valued attributes.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, tender_id).
3NF: No transitive dependencies; attributes like project_name, location, status, company_id, and rating directly depend on the primary key.

Company:

1NF: Atomic values for each attribute, preventing multi-valued attributes.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, company_id).
3NF: No transitive dependencies; attributes like company_name and work_type directly depend on the primary key.

Work_done:

1NF: Each attribute has atomic values, ensuring there are no multi-valued attributes.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, company_id, tender_id).
3NF: No transitive dependencies; attributes like price and time_completion directly depend on the primary key.

4) **Travel_And_Tourism:**

1NF: Each table has unique identifiers to ensure there are no duplicate rows.
2NF: No partial dependencies; all attributes in each table are dependent on the primary key.
3NF: No transitive dependencies; attributes depend only on the primary keys.

Tourist_attraction:

1NF: Each attribute has atomic values, preventing multi-valued attributes.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, attract_id).
3NF: No transitive dependencies; attributes like attract_name, location, descrp, category, entry_fee, and open_hours directly depend on the primary key.

Tourism_Development_Project:

1NF: Atomic values for each attribute, ensuring there are no multi-valued attributes.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, project_id).
3NF: No transitive dependencies; attributes like pname, descrp, location, budget, and timeline directly depend on the primary key.

5) Education:

1NF: Each table has unique identifiers for departments, institutes, and schemes.
2NF: All attributes depend on the primary keys, ensuring no partial dependencies.
3NF: No transitive dependencies; attributes directly depend on the primary keys.

Edu_Institute:

1NF: Atomic values for each attribute, preventing multi-valued attributes.

2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, Institute_id).
3NF: No transitive dependencies; attributes like Institute_name, Institute_loc, Registration_no, and Institute_type directly depend on the primary key.

Edu_schemes:

1NF: Attributes have atomic values, ensuring no multi-valued attributes.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, Schemes_id).
3NF: No transitive dependencies; attributes like Scheme_name, Allocate_budget, Current_status, Starting_date, and Ending_date directly depend on the primary key.

Central_government and State_government:

1NF: Each table has unique identifiers for departments, schemes, ministers, and properties.
2NF: All attributes depend on the composite primary key (Dept_id, Schemes_id, Minister_id, property_id).
3NF: No transitive dependencies; attributes directly depend on the primary key.

Edu_Queries:

1NF: Unique identifiers for departments, queries, and dates.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, Query_id).
3NF: No transitive dependencies; attributes like Query_date and Description directly depend on the primary key.

6) Communications_And_IT:

1NF: Each table has unique identifiers for departments and ministers.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id).
3NF: No transitive dependencies; attributes directly depend on the primary key.

Person_list:

1NF: Unique identifiers for persons, along with their roles, skills, and certifications.

2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, person_id).
3NF: No transitive dependencies; attributes like role and skills directly depend on the primary key.

Person_contact and Person_email:
1NF: Unique identifiers for persons along with their contact information (email and phone number).
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, person_id).
3NF: No transitive dependencies; contact information directly depends on the primary key.

Equipment_list:
1NF: Unique identifiers for equipment along with their specifications and maintenance records.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, equipment_id).
3NF: No transitive dependencies; equipment specifications directly depend on the primary key.

Projects:
1NF: Unique identifiers for projects along with their descriptions and statuses.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, project_id).
3NF: No transitive dependencies; project details directly depend on the primary key.

7) Ministry:
1NF: Each table has unique identifiers for departments and ministers.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id).
3NF: No transitive dependencies; attributes directly depend on the primary key.

Ministry_info:
1NF: Unique identifiers for ministers along with their first and last names, department names, and governance areas.

2NF: All attributes depend on the composite primary key (Dept_id, Minister_id).
3NF: No transitive dependencies; minister details directly depend on the primary key.

Budget_info:

1NF: Unique identifiers for programs along with their budgets and years.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, Program, Year).
3NF: No transitive dependencies; budget details directly depend on the primary key.

8) Agriculture:

1NF: Each table has unique identifiers for departments and ministers.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id).
3NF: No transitive dependencies; attributes directly depend on the primary key.

Agri_project:

1NF: Unique identifiers for agriculture projects along with project details.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, project_id).
3NF: No transitive dependencies; project details directly depend on the primary key.

Farmers_info:

1NF: Unique identifiers for farmers along with their details.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, farmer_id).
3NF: No transitive dependencies; farmer details directly depend on the primary key.

Agri_schemes:

1NF: Unique identifiers for agricultural schemes along with scheme details.

2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, Schemes_id).
3NF: No transitive dependencies; scheme details directly depend on the primary key.

Central_government and State_government:

These tables store information about the allocation of schemes by the central and state governments.
They are in 1NF, 2NF, and 3NF as all attributes depend on their respective composite primary keys.

Agri_queries:

This table holds queries related to agriculture, including query details.
It's in 1NF, 2NF, and 3NF with all attributes depending on the composite primary key (Dept_id, Minister_id, Query_id).

9) Defence:

1NF: Unique identifiers for departments and ministers, along with department details.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id).
3NF: No transitive dependencies; attributes directly depend on the primary key.

Person_list:

1NF: Unique identifiers for personnel along with their details.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, person_id).
3NF: No transitive dependencies; personnel details directly depend on the primary key.

Equipment_list:

1NF: Unique identifiers for equipment along with equipment details.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, equipment_id).
3NF: No transitive dependencies; equipment details directly depend on the primary key.

Missions:

1NF: Unique identifiers for missions along with mission details.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, mission_id).
3NF: No transitive dependencies; mission details directly depend on the primary key.

10) Election Commission:

1NF: Unique identifiers for departments and ministers, along with department details.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id).
3NF: No transitive dependencies; attributes directly depend on the primary key.

Candidate_details:

1NF: Unique identifiers for candidates along with their details.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, candidate_id).
3NF: No transitive dependencies; candidate details directly depend on the primary key.

Political_Party:

1NF: Unique identifiers for parties along with party details.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, Party_id).
3NF: No transitive dependencies; party details directly depend on the primary key.

Election_details:

1NF: Unique identifiers for elections along with election details.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, election_id).
3NF: No transitive dependencies; election details directly depend on the primary key.

Elec_Queries:

1NF: Unique identifiers for queries along with query details.

2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, Query_id).
3NF: No transitive dependencies; query details directly depend on the primary key.

11) Health and Family Welfare:

1NF: Unique identifiers for departments and ministers, along with department details.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id).
3NF: No transitive dependencies; attributes directly depend on the primary key.

Patient_info:

1NF: Unique identifiers for patients along with their details.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, patient_id).
3NF: No transitive dependencies; patient details directly depend on the primary key.

HFW_schemes:

1NF: Unique identifiers for schemes along with scheme details.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, schemes_id).
3NF: No transitive dependencies; scheme details directly depend on the primary key.

12) Housing:

1NF: Unique identifiers for departments and ministers, along with department details.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id).
3NF: No transitive dependencies; attributes directly depend on the primary key.

Property_info:

1NF: Unique identifiers for properties along with property details.

2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, property_id).
3NF: No transitive dependencies; property details directly depend on the primary key.

Private and Public:

1NF: Unique identifiers for private and public properties along with their respective details.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, property_id).
3NF: No transitive dependencies; property details directly depend on the primary key.

Property_transfer_info:

1NF: Unique identifiers for property transfers along with transfer details.
2NF: All attributes depend on the composite primary key (Dept_id, Minister_id, land_id).
3NF: No transitive dependencies; transfer details directly depend on the primary key.

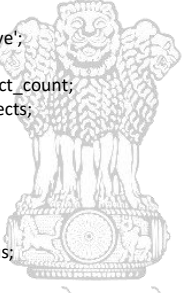
FUNCTIONS

- **Write a function to find remaining budget for the particular scheme in the agriculture sector.**

```
CREATE OR REPLACE FUNCTION calculate_remaining_budget(scheme_id IN
NUMBER) RETURN NUMBER IS
  total_budget Agri_schemes.Allocate_budget%TYPE;
  used_budget NUMBER;
  remaining_budget NUMBER;
BEGIN
  SELECT Allocate_budget INTO total_budget
  FROM Agri_schemes
  WHERE Schemes_id = scheme_id;

  SELECT SUM(Allocate_budget) INTO used_budget
  FROM Agri_schemes
  WHERE Schemes_id != scheme_id;

  remaining_budget := total_budget - used_budget;

  RETURN remaining_budget;
END calculate_remaining_budget;
/

DECLARE
  v_scheme_id NUMBER := 123;
  v_remaining_budget NUMBER;
BEGIN
  v_remaining_budget := calculate_remaining_budget(v_scheme_id);
  DBMS_OUTPUT.PUT_LINE('Remaining budget for scheme ' || v_scheme_id ||
': ' || v_remaining_budget);
END;
/
```

- **Write a function which return the number of active projects in Communication and IT department.**

```
CREATE OR REPLACE FUNCTION count_active_projects RETURN
NUMBER IS
   active_project_count NUMBER;
BEGIN
   SELECT COUNT(*) INTO active_project_count
   FROM Projects
   WHERE status = 'Active';

   RETURN active_project_count;
END count_active_projects;
/

DECLARE
BEGIN
   -- Call the procedure
   update_project_status;
EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/
```



PROCEDURES

- **Write procedure which increases the duration of health scheme end date.**

```
CREATE OR REPLACE PROCEDURE extend_health_scheme(
  scheme_id IN NUMBER,
  new_end_date IN DATE
) IS
BEGIN
  UPDATE HFW_schemes
  SET Ending_date = new_end_date
  WHERE schemes_id = scheme_id;

  COMMIT;
  DBMS_OUTPUT.PUT_LINE('Health scheme extended successfully!');
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Failed to extend health scheme.');
END extend_health_scheme;
/
DECLARE
  v_scheme_id NUMBER := 123;
  v_new_end_date DATE := TO_DATE('31-12-2024, 'DD-MM-YYYY');
BEGIN
  extend_health_scheme(v_scheme_id, v_new_end_date);
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
/
```

- **Write procedure which update the status of project in Communications and It department.**

```
CREATE OR REPLACE PROCEDURE update_project_status IS
BEGIN
  UPDATE Projects
  SET status = 'Completed'
  WHERE end_date < SYSDATE
  AND status <> 'Completed';

  COMMIT;
  DBMS_OUTPUT.PUT_LINE('Project statuses updated successfully!');
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Failed to update project statuses.');
END update_project_status;
/

DECLARE

BEGIN

  -- Call the procedure

  update_project_status;

EXCEPTION

  WHEN OTHERS THEN

    DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);

END;

/
```

PACKAGES

- In the construction project for choosing the best company's tender bid price and with best rating write pl/sql block using packages.

```
CREATE OR REPLACE PACKAGE tender_assign_company AS
        CURSOR c1 RETURN construction%ROWTYPE;
        CURSOR c2(s NUMBER) RETURN tender_company_bid%ROWTYPE;
        FUNCTION f1(a IN construction.tender_id%TYPE) RETURN BOOLEAN;
        FUNCTION f2(a IN work_done.company_id%TYPE) RETURN FLOAT;
        PROCEDURE p1(a IN construction.tender_id%TYPE, b IN
       company.company_id%TYPE);
END tender_assign_company;

CREATE OR REPLACE PACKAGE BODY tender_assign_company AS
  CURSOR c1 RETURN construction%ROWTYPE IS
  SELECT * FROM construction;

  CURSOR c2(s NUMBER) RETURN tender_company_bid%ROWTYPE IS
  SELECT * FROM tender_company_bid WHERE tender_id = s;

 FUNCTION f1(a IN construction.tender_id%TYPE) RETURN BOOLEAN IS
  r1 construction%ROWTYPE;

 BEGIN
  SELECT * INTO r1 FROM construction WHERE tender_id = a;
  RETURN r1.status = 'Pending' AND r1.last_date_for_apply - SYSDATE < 0;
 END f1;

 FUNCTION f2(a IN work_done.company_id%TYPE) RETURN FLOAT IS
  avg_rating FLOAT := 0;
 BEGIN
  SELECT AVG(rating) INTO avg_rating FROM work_done WHERE company_id
= a;
  RETURN avg_rating;
 END f2;
```

```
PROCEDURE p1(a IN construction.tender_id%TYPE, b IN
company.company_id%TYPE) IS
  BEGIN
    UPDATE construction
    SET company_id = b
    WHERE tender_id = a;
  END p1;
END tender_assign_company;


DECLARE
  r1 construction%ROWTYPE;
  r2 tender_company_bid%ROWTYPE;
  min_bid_company company.company_id%TYPE;
  min_bid_price NUMBER;
BEGIN
  FOR r1 IN tender_assign_company.c1 LOOP
    IF (tender_assign_company.f1(r1.tender_id)) THEN
      min_bid_price := 99999;
      FOR r2 IN tender_assign_company.c2(r1.tender_id) LOOP
        IF (r2.BID_PRICE < min_bid_price) THEN
          min_bid_price := r2.BID_PRICE;
          min_bid_company := r2.company_id;
        END IF;
        IF (r2.BID_PRICE = min_bid_price AND
tender_assign_company.f2(r2.company_id) >
tender_assign_company.f2(min_bid_company)) THEN
          min_bid_price := r2.BID_PRICE;
          min_bid_company := r2.company_id;
        END IF;
      END LOOP;

      tender_assign_company.p1(r1.tender_id, min_bid_company);

      DBMS_OUTPUT.PUT_LINE(r1.tender_id || ' ' || min_bid_company || ' ' ||
min_bid_price);
    END IF;
  END LOOP;
END;
```

➢ For candidate registration successful or not for the Election write pl/sql block using packages.

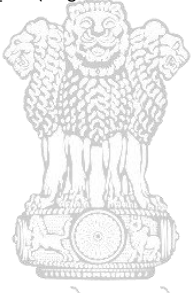
```
CREATE OR REPLACE PACKAGE Candidate_Registration IS
  FUNCTION valid_candidate(
    candidate_name VARCHAR2,
    uid VARCHAR2,
    party_id NUMBER
  ) RETURN BOOLEAN;

  PROCEDURE register_candidate(
    candidate_name VARCHAR2,
    uid VARCHAR2,
    party_id NUMBER,
    candidate_id OUT NUMBER
  );

  FUNCTION generate_candidate_id RETURN NUMBER;
END Candidate_Registration;


CREATE OR REPLACE PACKAGE BODY Candidate_Registration IS
  FUNCTION valid_candidate(
    candidate_name VARCHAR2,
    uid VARCHAR2,
    party_id NUMBER
  ) RETURN BOOLEAN IS
    c_record BOOLEAN;
    valid_party BOOLEAN;
  BEGIN
    SELECT criminal_record INTO c_record FROM citizen_info c WHERE c.uid =
uid;

    SELECT COUNT(*) INTO valid_party
    FROM Political_Party
    WHERE Party_id = party_id
      AND descrp LIKE '%active%';

    IF c_record = TRUE OR NOT valid_party THEN
      RETURN FALSE;
```

```
  ELSE
    RETURN TRUE;
  END IF;
 END valid_candidate;


 PROCEDURE register_candidate(
  candidate_name VARCHAR2,
  uid VARCHAR2,
  party_id NUMBER,
  candidate_id OUT NUMBER
 ) IS
 BEGIN
  IF valid_candidate(candidate_name, uid, party_id) THEN
    candidate_id := generate_candidate_id;

    INSERT INTO Candidate_details (candidate_id, candidate_name, UID,
Party_id)
    VALUES (candidate_id, candidate_name, uid, party_id);

    DBMS_OUTPUT.PUT_LINE('Candidate registration successful! Candidate ID:
' || candidate_id);
  ELSE
    DBMS_OUTPUT.PUT_LINE('Candidate registration failed: Invalid details or
party.');
  END IF;
 END register_candidate;
```



```
FUNCTION generate_candidate_id RETURN NUMBER IS
  candidate_id NUMBER;
 BEGIN
  SELECT NVL(MAX(candidate_id), 0) + 1 INTO candidate_id FROM
Candidate_details;

  RETURN candidate_id;
 END generate_candidate_id;
END Candidate_Registration;
```

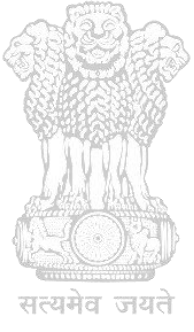
```
DECLARE
  candidate_name VARCHAR2(100) := '    Amit Shah';
  uid VARCHAR2(100) := '    544266871026';
  party_id NUMBER := 1;
  candidate_id NUMBER;
BEGIN
  Candidate_Registration.register_candidate(candidate_name,
uid, party_id, candidate_id);
  DBMS_OUTPUT.PUT_LINE('Registered Candidate ID: ' ||
candidate_id);
END;
/
```



- Write trigger using package for Total budget gain and the allocate budget not exceed max value in education scheme.

```
CREATE OR REPLACE PACKAGE EducationPackage AS
  FUNCTION GetTotalBudget(Dept_id IN NUMBER) RETURN DECIMAL;
  FUNCTION IsStateEligible(Schemes_id IN NUMBER) RETURN BOOLEAN;
END EducationPackage;
/

CREATE OR REPLACE PACKAGE BODY EducationPackage AS
  FUNCTION GetTotalBudget(Dept_id IN NUMBER) RETURN DECIMAL IS
    Total_Budget DECIMAL(15, 2);
  BEGIN
    SELECT SUM(Allocate_Budget)
    INTO Total_Budget
    FROM Edu_Schemes
    WHERE Dept_id = GetTotalBudget.Dept_id;

    RETURN Total_Budget;
  END GetTotalBudget;

  FUNCTION IsStateEligible(Schemes_id IN NUMBER) RETURN BOOLEAN IS
    Is_Eligible BOOLEAN := FALSE;
  BEGIN
    SELECT COUNT(*)
    INTO Is_Eligible
    FROM tate_Government
    WHERE Schemes_id = IsStateEligible.Schemes_id;

    RETURN Is_Eligible > 0;
  END IsStateEligible;
END EducationPackage;
/

CREATE OR REPLACE TRIGGER CheckSchemeBudget
BEFORE INSERT OR UPDATE ON Edu_Schemes
FOR EACH ROW
DECLARE
```

```
    Total_Budget DECIMAL(15, 2);

BEGIN

    Total_budget:=EducationPackage.GetTotalBudget(:NEW.Dept_id);

    If EducationPackage.IsStateEligible(:New.Schemes_id) then
    IF Total_Budget + :NEW.Allocate_Budget > 1000000 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Cannot allocate more
budget. Total budget limit exceeded.');
    END IF;

    END IF;
END;
/
```



TRIGGERS

- Write a trigger which would be executed when the insertion is attempted after the deadline date of tender bid.

```
create or replace trigger check_under_valid_date before insert on
tender_company_bid

for each row
   declare
       last_date  construction.last_date_for_apply%type;
  begin
       select last_date_for_apply into last_date from construction c
        where :new.tender_id=c.tender_id;
       dbms_output.put_line(last_date||'is last date');
       if (last_date-sysdate)<0 then dbms_output.put_line('you are
       late');
       end if;

exception
   when no_data_found then dbms_output.put_line('you made some
   mistake');
   when others then dbms_output.put_line('you done something
   wrong');
end;
```

- Write a trigger which will be executed when one tender or project is completed then it will tranfer into work_done table.

```
create or replace trigger go_to_work_done after update on
construction
for each row
 declare
    cursor c1 is select * from construction;
    r1 c1%rowtype;
    price number;
 begin
        for r1 in c1
          loop
           if(r1.status='Completed') then
            select Bid_price into price from tender_company_bid where
        company_id=r1.company_id and tender_id=r1.tender_id;

            insert into work_done
        values(r1.company_id,r1.tender_id,price,sysdate,'4.5');

            delete from construction c where r1.tender_id=c.tender_id;
          end if;
         end loop;
end go_to_work_done;
/
```

- Write a trigger which will update the project rating after the insertion occurs in Tourist_attraction table.

```
CREATE OR REPLACE TRIGGER
UpdateProjectRatingOnAttractionInsert
AFTER INSERT ON Tourist_attraction
FOR EACH ROW
DECLARE
   project_id NUMBER;
   category VARCHAR2(50);
BEGIN
   project_id := :new.project_id;

   category := :new.category;

   IF category = 'Historical' THEN
      UPDATE Tourism_Development_Project
      SET rating = rating + 1
      WHERE project_id = project_id;
   ELSIF category = 'Adventure' THEN
      UPDATE Tourism_Development_Project
      SET rating = rating + 2
      WHERE project_id = project_id;
   ELSIF category = 'Natural' THEN
      UPDATE Tourism_Development_Project
      SET rating = rating + 3
      WHERE project_id = project_id;
   END IF;
END;
/
```

- Write a trigger which checks the equipment which is inserted in equipment_list are associated for the mission or not, if it is then change resource_allocated for that mission based on the number of equipment assigned to it.

```
CREATE OR REPLACE TRIGGER
UpdateMissionsOnEquipmentAssignment
AFTER INSERT ON Equipment_list
FOR EACH ROW
DECLARE
  mission_id NUMBER;
BEGIN
  mission_id := GetMissionID(:new.equipment_id);
    IF mission_id IS NOT NULL THEN
    UpdateMission(mission_id);
  END IF;
END;
/

CREATE OR REPLACE FUNCTION GetMissionID(p_equipment_id IN
NUMBER) RETURN NUMBER
IS
  missions_id NUMBER;
BEGIN
  SELECT mission_id INTO missions_id
  FROM Missions
  WHERE equipment_id = p_equipment_id;

  RETURN missions_id;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
```

```
    RETURN NULL;
END;
CREATE OR REPLACE PROCEDURE UpdateMission(p_mission_id IN
NUMBER)
IS
  resources_allocated NUMBER;
BEGIN
  SELECT COUNT(*) INTO resources_allocated
  FROM Equipment_list
  WHERE mission_id = p_mission_id;

  UPDATE Missions

  WHERE mission_id = p_mission_id;
END;
/
```