**UNIVERSITY OF RWANDA**

**COLLEGE OF BUSINESS AND ECONOMICS**

**African Center of Excellence in Data Science(ACE DS)**

**DATA MINING**

**Advanced Database Technology**

**REG NO 223028000**

**NGO PROJECT FUNDING AND EXPENDITURE TRACKING**

**CREATING TABLES**

Donor Table

CREATE TABLE Donor (

DonorID NUMBER PRIMARY KEY, -- Unique identifier for each donor; must be a number and cannot be null

FullName VARCHAR2(100) NOT NULL, -- Donor's full name; up to 100 characters and required

Organization VARCHAR2(100), -- Name of the donor's organization; optional field

Contact VARCHAR2(20) NOT NULL, -- Contact number; up to 20 characters and required

Email VARCHAR2(100) NOT NULL UNIQUE -- Donor's email address; required and must be unique across all donors

);

Project Table

CREATE TABLE Project (

ProjectID NUMBER PRIMARY KEY, -- Unique identifier for each project

Title VARCHAR2(150) NOT NULL, -- Title of the project; required and up to 150 characters

Description VARCHAR2(1000), -- Optional detailed description of the project

Budget NUMBER(12,2) NOT NULL -- Project budget with up to 12 digits, 2 decimal places; required

CONSTRAINT chk\_budget CHECK (Budget >= 0), -- Ensures the budget is non-negative

StartDate DATE NOT NULL, -- Start date of the project; required

EndDate DATE, -- Optional end date of the project

DonorID NUMBER NOT NULL, -- Foreign key referencing the donor who funds the project; required

Status VARCHAR2(50) DEFAULT 'Ongoing', -- Current status of the project; defaults to 'Ongoing'

CONSTRAINT chk\_status CHECK (Status IN -- Restricts status to predefined values

('Planned', 'Ongoing', 'Completed', 'Cancelled')),

CONSTRAINT fk\_donor\_project FOREIGN KEY (DonorID) -- Foreign key constraint linking to Donor table

REFERENCES Donor(DonorID)

);

Staff Table

CREATE TABLE Staff (

StaffID NUMBER PRIMARY KEY, -- Unique identifier for each staff member

FullName VARCHAR2(100) NOT NULL, -- Staff member's full name; required and up to 100 characters

Role VARCHAR2(50) NOT NULL -- Role assigned to the staff member; required

CHECK (Role IN ('Manager', 'Coordinator', 'Volunteer', 'Analyst', 'Support')), -- Restricts role to predefined values

Contact VARCHAR2(20) NOT NULL, -- Contact number; required and up to 20 characters

ProjectID NUMBER NOT NULL, -- Foreign key linking staff to a specific project; required

CONSTRAINT fk\_project\_staff FOREIGN KEY (ProjectID) -- Foreign key constraint referencing Project table

REFERENCES Project(ProjectID)

);

Activity Table

CREATE TABLE Activity (

ActivityID NUMBER PRIMARY KEY, -- Unique identifier for each activity

ProjectID NUMBER NOT NULL, -- Foreign key linking the activity to a specific project; required

Title VARCHAR2(150) NOT NULL, -- Title of the activity; required and up to 150 characters

ScheduleDate DATE NOT NULL, -- Scheduled date for the activity; required

Status VARCHAR2(50) NOT NULL -- Current status of the activity; required

CHECK (Status IN ('Planned', 'Ongoing', 'Completed', 'Cancelled')), -- Restricts status to predefined values

CONSTRAINT fk\_project\_activity FOREIGN KEY (ProjectID) -- Foreign key constraint referencing Project table

REFERENCES Project(ProjectID)

);

Expense Table

CREATE TABLE Expense (

ExpenseID NUMBER PRIMARY KEY, -- Unique identifier for each expense record

ProjectID NUMBER NOT NULL, -- Foreign key linking the expense to a specific project; required

Description VARCHAR2(1000) NOT NULL, -- Detailed description of the expense; required and up to 1000 characters

Cost NUMBER(10,2) NOT NULL -- Expense amount with up to 10 digits, 2 decimal places; required

CHECK (Cost >= 0), -- Ensures the cost is non-negative

DateIncurred DATE NOT NULL -- Date when the expense was incurred; required

CHECK (DateIncurred >= TO\_DATE('2000-01-01', 'YYYY-MM-DD')), -- Ensures the date is not before the year 2000

CONSTRAINT fk\_project\_expense FOREIGN KEY (ProjectID) -- Foreign key constraint referencing the Project table

REFERENCES Project(ProjectID) ON DELETE CASCADE -- Automatically deletes related expenses if the project is deleted

);

Report Table

CREATE TABLE Report (

ReportID NUMBER PRIMARY KEY, -- Unique identifier for each report

ProjectID NUMBER NOT NULL, -- Foreign key linking the report to a specific project; required

SubmittedBy VARCHAR2(100) NOT NULL, -- Name of the person who submitted the report; required

DateSubmitted DATE NOT NULL -- Date the report was submitted; required

CHECK (DateSubmitted >= TO\_DATE('2000-01-01', 'YYYY-MM-DD')), -- Ensures the submission date is not before the year 2000

Summary VARCHAR2(4000) NOT NULL, -- Detailed summary of the report; required and up to 4000 characters

CONSTRAINT fk\_project\_report FOREIGN KEY (ProjectID) -- Foreign key constraint referencing the Project table

REFERENCES Project(ProjectID) ON DELETE CASCADE -- Automatically deletes related reports if the project is deleted

);

3. Inserting data into tables

INSERT INTO Project (ProjectID, Title, Description, Budget, StartDate, EndDate, DonorID, Status)

VALUES (101, 'Clean Water Initiative', 'Providing clean water to rural communities.', 50000.00, TO\_DATE('2025-01-15', 'YYYY-MM-DD'), TO\_DATE('2025-06-15', 'YYYY-MM-DD'), 1, 'Ongoing');

INSERT INTO Project (ProjectID, Title, Description, Budget, StartDate, EndDate, DonorID, Status)

VALUES (102, 'Youth Empowerment Program', 'Training youth in entrepreneurship and leadership.', 75000.00, TO\_DATE('2025-02-01', 'YYYY-MM-DD'), NULL, 2, 'Planned');

INSERT INTO Project (ProjectID, Title, Description, Budget, StartDate, EndDate, DonorID, Status)

VALUES (103, 'Health Outreach Campaign', 'Mobile clinics and health education.', 60000.00, TO\_DATE('2025-03-10', 'YYYY-MM-DD'), TO\_DATE('2025-08-10', 'YYYY-MM-DD'), 1, 'Ongoing');

INSERT INTO Project (ProjectID, Title, Description, Budget, StartDate, EndDate, DonorID, Status)

VALUES (104, 'Digital Literacy for Women', 'Teaching basic computer skills to women.', 40000.00, TO\_DATE('2025-04-05', 'YYYY-MM-DD'), NULL, 3, 'Planned');

b.

INSERT INTO Donor (DonorID, FullName, Organization, Contact, Email)

VALUES (1, 'Alice Mugenzi', 'Mugenzi Foundation', '0788123456', 'alice@mugenzi.org');

INSERT INTO Donor (DonorID, FullName, Organization, Contact, Email)

VALUES (2, 'Jean Bosco', 'Bosco Group', '0788345678', 'jean.bosco@bosco.com');

INSERT INTO Donor (DonorID, FullName, Organization, Contact, Email)

VALUES (3, 'Clara Uwase', NULL, '0788567890', 'clara.uwase@gmail.com');

INSERT INTO Project (ProjectID, Title, Description, Budget, StartDate, EndDate, DonorID, Status)

VALUES (105, 'Environmental Awareness Drive', 'Workshops and campaigns on climate change.', 30000.00, TO\_DATE('2025-05-20', 'YYYY-MM-DD'), TO\_DATE('2025-09-20', 'YYYY-MM-DD'), 2, 'Ongoing');

4. Retrieve all projects and their total expenses

SELECT

p.ProjectID, -- Select the unique ID of each project

p.Title, -- Select the title of the project

SUM(e.Cost) AS TotalExpense -- Calculate the total expense for each project

FROM

Project p -- Use the Project table as the main source

LEFT JOIN

Expense e -- Join with the Expense table to link expenses to projects

ON p.ProjectID = e.ProjectID -- Match expenses to their corresponding project

GROUP BY

p.ProjectID, p.Title; -- Group results by project to aggregate expenses correctly

5. Update the status of a project after its final report is submitted

UPDATE Project

SET Status = 'Completed' -- Set the new status

WHERE ProjectID IN (

SELECT ProjectID

FROM Report

WHERE Summary LIKE '%final report%' -- Identify reports marked as final

);

6. Identify donors funding more than one project

SELECT DonorID, COUNT(ProjectID) AS ProjectCount

FROM Project

GROUP BY DonorID

HAVING COUNT(ProjectID) > 1;

7. Create a view showing total spending per project.

CREATE VIEW ProjectSpending AS

SELECT

p.ProjectID,

p.Title,

COALESCE(SUM(e.Cost), 0) AS TotalSpending

FROM

Project p

LEFT JOIN

Expense e ON p.ProjectID = e.ProjectID

GROUP BY

p.ProjectID, p.Title;.

SELECT \* FROM ProjectSpending;

8. Creating trigger

CREATE OR REPLACE TRIGGER prevent\_over\_budget\_expense

BEFORE INSERT ON Expense

FOR EACH ROW

DECLARE

total\_spent NUMBER;

project\_budget NUMBER;

BEGIN

-- Get total expenses already recorded for the project

SELECT COALESCE(SUM(Cost), 0)

INTO total\_spent

FROM Expense

WHERE ProjectID = :NEW.ProjectID;

-- Get the project's budget

SELECT Budget

INTO project\_budget

FROM Project

WHERE ProjectID = :NEW.ProjectID;

-- Check if new expense would exceed the budget

IF total\_spent + :NEW.Cost > project\_budget THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Expense exceeds project budget.');

END IF;

END;