Create necessary tables:

CREATE TABLE donors (

donor\_id NUMBER(10) PRIMARY KEY,

first\_name VARCHAR2(50) NOT NULL,

last\_name VARCHAR2(50) NOT NULL,

email VARCHAR2(50) NOT NULL,

phone\_number VARCHAR2(20) NOT NULL,

address VARCHAR2(200) NOT NULL

);

CREATE TABLE donations (

donation\_id NUMBER(10) PRIMARY KEY,

donor\_id NUMBER(10) NOT NULL,

donation\_date DATE NOT NULL,

amount NUMBER(10, 2) NOT NULL,

payment\_method VARCHAR2(20) NOT NULL,

CONSTRAINT fk\_donors FOREIGN KEY (donor\_id) REFERENCES donors(donor\_id)

);

CREATE TABLE projects (

project\_id NUMBER(10) PRIMARY KEY,

name VARCHAR2(50) NOT NULL,

description VARCHAR2(200) NOT NULL,

target\_amount NUMBER(10, 2) NOT NULL,

start\_date DATE NOT NULL,

end\_date DATE NOT NULL,

status VARCHAR2(20) NOT NULL

);

CREATE TABLE project\_allocations (

allocation\_id NUMBER(10) PRIMARY KEY,

donation\_id NUMBER(10) NOT NULL,

project\_id NUMBER(10) NOT NULL,

allocation\_date DATE NOT NULL,

amount NUMBER(10, 2) NOT NULL,

CONSTRAINT fk\_donations FOREIGN KEY (donation\_id) REFERENCES donations(donation\_id),

CONSTRAINT fk\_projects FOREIGN KEY (project\_id) REFERENCES projects(project\_id)

);

-- Inserting data into the donors table

INSERT INTO donors (donor\_id, first\_name, last\_name, email, phone\_number, address)

VALUES (1, 'John', 'Doe', 'johndoe@example.com', '1234567890', '123 Main St');

INSERT INTO donors (donor\_id, first\_name, last\_name, email, phone\_number, address)

VALUES (2, 'Jane', 'Smith', 'janesmith@example.com', '9876543210', '456 Elm St');

-- Inserting data into the donations table

INSERT INTO donations (donation\_id, donor\_id, donation\_date, amount, payment\_method)

VALUES (1, 1, TO\_DATE('2023-06-01', 'YYYY-MM-DD'), 100.00, 'Credit Card');

INSERT INTO donations (donation\_id, donor\_id, donation\_date, amount, payment\_method)

VALUES (2, 2, TO\_DATE('2023-06-02', 'YYYY-MM-DD'), 250.00, 'PayPal');

-- Inserting data into the projects table

INSERT INTO projects (project\_id, name, description, target\_amount, start\_date, end\_date, status)

VALUES (1, 'Project A', 'Description of Project A', 1000.00, TO\_DATE('2023-06-01', 'YYYY-MM-DD'), TO\_DATE('2023-07-01', 'YYYY-MM-DD'), 'Active');

INSERT INTO projects (project\_id, name, description, target\_amount, start\_date, end\_date, status)

VALUES (2, 'Project B', 'Description of Project B', 500.00, TO\_DATE('2023-06-15', 'YYYY-MM-DD'), TO\_DATE('2023-07-15', 'YYYY-MM-DD'), 'Active');

-- Inserting data into the project\_allocations table

INSERT INTO project\_allocations (allocation\_id, donation\_id, project\_id, allocation\_date, amount)

VALUES (1, 1, 1, TO\_DATE('2023-06-03', 'YYYY-MM-DD'), 50.00);

INSERT INTO project\_allocations (allocation\_id, donation\_id, project\_id, allocation\_date, amount)

VALUES (2, 2, 2, TO\_DATE('2023-06-04', 'YYYY-MM-DD'), 200.00);

1. Create a trigger that automatically updates the status of a project to "Funded" when the total amount allocated to that project reaches or exceeds the target amount. Name the trigger as "project\_funded\_trigger".

```sql

CREATE OR REPLACE TRIGGER project\_funded\_trigger

AFTER INSERT OR UPDATE ON project\_allocations

FOR EACH ROW

DECLARE

total\_allocated\_amount NUMBER(10, 2);

BEGIN

SELECT SUM(amount) INTO total\_allocated\_amount

FROM project\_allocations

WHERE project\_id = :NEW.project\_id;

IF total\_allocated\_amount >= (SELECT target\_amount FROM projects WHERE project\_id = :NEW.project\_id) THEN

UPDATE projects SET status = 'Funded' WHERE project\_id = :NEW.project\_id;

END IF;

END;

/

```

2. Create a cursor that retrieves all the donors who have made donations above a specified amount. Name the cursor as "high\_value\_donors\_cur".

```sql

CREATE OR REPLACE PROCEDURE retrieve\_high\_value\_donors(

high\_amount IN NUMBER,

cur OUT SYS\_REFCURSOR

)

AS

BEGIN

OPEN cur FOR

SELECT d.donor\_id, d.first\_name, d.last\_name

FROM donors d

JOIN donations dn ON d.donor\_id = dn.donor\_id

WHERE dn.amount > high\_amount;

END;

/

```

3. Create a procedure to delete a donor and their associated donation records. Name the procedure as "delete\_donor".

```sql

CREATE OR REPLACE PROCEDURE delete\_donor(

donor\_id IN NUMBER

)

AS

BEGIN

DELETE FROM donations WHERE donor\_id = donor\_id;

DELETE FROM donors WHERE donor\_id = donor\_id;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

RAISE;

END;

/

```

4. Create a function to calculate the total amount donated to a specific project. Name the function as "calculate\_total\_donation\_for\_project".

```sql

CREATE OR REPLACE FUNCTION calculate\_total\_donation\_for\_project(

project\_id IN NUMBER

)

RETURN NUMBER

AS

total\_donation NUMBER(10, 2);

BEGIN

SELECT COALESCE(SUM(amount), 0) INTO total\_donation

FROM project\_allocations

WHERE project\_id = project\_id;

RETURN total\_donation;

END;

/

```

5. Create an exception handler for the procedure that records a new donation, to handle the case when the donation amount is negative or exceeds the remaining target amount for the project.

```sql

CREATE OR REPLACE PROCEDURE record\_donation(

donor\_id IN NUMBER,

donation\_date IN DATE,

amount IN NUMBER,

payment\_method IN VARCHAR2,

project\_id IN NUMBER

)

AS

remaining\_amount NUMBER(10, 2);

BEGIN

SELECT target\_amount - calculate\_total\_donation\_for\_project(project\_id)

INTO remaining\_amount

FROM projects

WHERE project\_id = project\_id;

IF amount <= 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Invalid donation amount.');

ELSIF amount > remaining\_amount THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Donation amount exceeds the remaining target amount for the project.');

ELSE

INSERT INTO donations (donation\_id, donor\_id, donation\_date, amount, payment\_method)

VALUES (donation\_id\_seq.NEXTVAL, donor\_id, donation\_date, amount, payment\_method);

INSERT INTO project\_allocations (allocation\_id, donation\_id, project\_id, allocation\_date, amount)

VALUES (allocation\_id\_seq.NEXTVAL, donation\_id\_seq.CURRVAL, project\_id, SYSD

ATE, amount);

COMMIT;

END IF;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

RAISE;

END;

/

```

Webapi:

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Web.Http;

// Define the Hospital entity

public class Hospital

{

public int HospitalId { get; set; }

public string Name { get; set; }

public string Address { get; set; }

public string ContactNumber { get; set; }

public int NumberOfBeds { get; set; }

public List<string> Departments { get; set; }

}

// Define the database context

public class HospitalContext : DbContext

{

public DbSet<Hospital> Hospitals { get; set; }

}

// Define the API controller

public class HospitalsController : ApiController

{

private readonly HospitalContext \_context;

public HospitalsController()

{

\_context = new HospitalContext();

}

// GET /api/hospitals

public IHttpActionResult GetHospitals()

{

List<Hospital> hospitals = \_context.Hospitals.ToList();

return Ok(hospitals);

}

// GET /api/hospitals/{id}

public IHttpActionResult GetHospital(int id)

{

Hospital hospital = \_context.Hospitals.FirstOrDefault(h => h.HospitalId == id);

if (hospital == null)

{

return NotFound();

}

return Ok(hospital);

}

// POST /api/hospitals

public IHttpActionResult PostHospital(Hospital hospital)

{

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

\_context.Hospitals.Add(hospital);

\_context.SaveChanges();

return CreatedAtRoute("DefaultApi", new { id = hospital.HospitalId }, hospital);

}

// PUT /api/hospitals/{id}

public IHttpActionResult PutHospital(int id, Hospital hospital)

{

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

if (id != hospital.HospitalId)

{

return BadRequest();

}

\_context.Entry(hospital).State = EntityState.Modified;

\_context.SaveChanges();

return StatusCode(HttpStatusCode.NoContent);

}

// DELETE /api/hospitals/{id}

public IHttpActionResult DeleteHospital(int id)

{

Hospital hospital = \_context.Hospitals.FirstOrDefault(h => h.HospitalId == id);

if (hospital == null)

{

return NotFound();

}

\_context.Hospitals.Remove(hospital);

\_context.SaveChanges();

return Ok(hospital);

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

\_context.Dispose();

}

base.Dispose(disposing);

}

}

// Configure routing

public static class WebApiConfig

{

public static void Register(HttpConfiguration config)

{

config.Routes.MapHttpRoute(

name: "DefaultApi",

routeTemplate: "api/{controller}/{id}",

defaults: new { id = RouteParameter.Optional }

);

}

}

// In the Global.asax.cs file:

public class WebApiApplication : System.Web.HttpApplication

{

protected void Application\_Start()

{

GlobalConfiguration.Configure(WebApiConfig.Register);

}