To implement the triggers for your Corporate\_vendor\_management\_system database, we'll focus on ensuring data integrity, security (such as password hashing), and enforcing business logic in different operations like inserts and updates. Here's how you can write the necessary triggers for your DDL schema:

**1. Trigger: Hash User Password Before Insert**

This trigger ensures that the password is hashed (using SHA-256) before inserting a new user into the User table. This will help ensure that user passwords are stored securely.

DELIMITER $$

CREATE TRIGGER HashUserPasswordBeforeInsert

BEFORE INSERT ON User

FOR EACH ROW

BEGIN

-- Hash the password before insertion

SET NEW.Password = SHA2(NEW.Password, 256); -- Hashing the password using SHA-256

END $$

DELIMITER ;

**2. Trigger: Hash User Password Before Update**

This trigger hashes the password only when it is being updated in the User table. It will avoid unnecessary hashing when the password remains the same.

DELIMITER $$

CREATE TRIGGER HashUserPasswordBeforeUpdate

BEFORE UPDATE ON User

FOR EACH ROW

BEGIN

-- Only hash the password if it has changed

IF NEW.Password != OLD.Password THEN

SET NEW.Password = SHA2(NEW.Password, 256); -- Hashing the password using SHA-256

END IF;

END $$

DELIMITER ;

**3. Trigger: Validate Expenditure Doesn't Exceed Total Budget**

This trigger ensures that the Expenditure in the Budget table does not exceed the TotalBudget for any department. If an insert violates this rule, it will raise an error.

DELIMITER $$

CREATE TRIGGER ValidateBudgetExpenditure

BEFORE INSERT ON Budget

FOR EACH ROW

BEGIN

IF NEW.Expenditure > NEW.TotalBudget THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Expenditure cannot exceed the total budget!';

END IF;

END $$

DELIMITER ;

**4. Trigger: Default Task Status to "Pending"**

This trigger sets the default Status of a task to 'Pending' if the Status is not provided during an insert into the Task table.

DELIMITER $$

CREATE TRIGGER SetDefaultTaskStatus

BEFORE INSERT ON Task

FOR EACH ROW

BEGIN

IF NEW.Status IS NULL THEN

SET NEW.Status = 'Pending'; -- Default status is 'Pending' if not provided

END IF;

END $$

DELIMITER ;

**5. Trigger: Log User Role Changes**

This trigger logs the changes made to the Role field in the User table. When a user's role is updated, a log entry is inserted into a UserRoleChangeLog table (which should be created separately if you don't have it yet).

DELIMITER $$

CREATE TRIGGER LogUserRoleChange

AFTER UPDATE ON User

FOR EACH ROW

BEGIN

-- Log the role change if the Role field has changed

IF OLD.Role != NEW.Role THEN

INSERT INTO UserRoleChangeLog (UserID, OldRole, NewRole, ChangeDate)

VALUES (OLD.UserID, OLD.Role, NEW.Role, NOW());

END IF;

END $$

DELIMITER ;

* **Note**: The UserRoleChangeLog table should look something like this:

CREATE TABLE UserRoleChangeLog (

ID INT AUTO\_INCREMENT PRIMARY KEY,

UserID INT NOT NULL,

OldRole VARCHAR(255) NOT NULL,

NewRole VARCHAR(255) NOT NULL,

ChangeDate TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (UserID) REFERENCES User(UserID) ON DELETE CASCADE

);

**6. Trigger: Default Contract Status to "Active"**

This trigger ensures that a contract status is set to 'Active' if no status is provided during an insert into the Contract table.

DELIMITER $$

CREATE TRIGGER SetDefaultContractStatus

BEFORE INSERT ON Contract

FOR EACH ROW

BEGIN

IF NEW.Status IS NULL THEN

SET NEW.Status = 'Active'; -- Default status is 'Active' if not provided

END IF;

END $$

DELIMITER ;

**7. Trigger: Validate Contract End Date is After Start Date**

This trigger ensures that the EndDate of the contract is always greater than the StartDate. This is enforced with a CHECK constraint in your DDL, but adding a trigger provides an extra layer of validation in case the constraint is removed or ignored.

DELIMITER $$

CREATE TRIGGER ValidateContractDates

BEFORE INSERT ON Contract

FOR EACH ROW

BEGIN

IF NEW.EndDate <= NEW.StartDate THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Contract End Date must be after the Start Date!';

END IF;

END $$

DELIMITER ;

**8. Trigger: Prevent Negative Purchase Order Total Cost**

This trigger ensures that the TotalCost for a purchase order cannot be negative when a new purchase order is inserted into the PurchaseOrder table.

DELIMITER $$

CREATE TRIGGER ValidatePurchaseOrderTotalCost

BEFORE INSERT ON PurchaseOrder

FOR EACH ROW

BEGIN

IF NEW.TotalCost < 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Total Cost for Purchase Order cannot be negative!';

END IF;

END $$

DELIMITER ;

**9. Trigger: Default Vendor Rating to 0.00**

This trigger sets the default performance Rating to 0.00 if no rating is provided during an insert into the Performance table.

DELIMITER $$

CREATE TRIGGER SetDefaultPerformanceRating

BEFORE INSERT ON Performance

FOR EACH ROW

BEGIN

IF NEW.Rating IS NULL THEN

SET NEW.Rating = 0.00; -- Default rating is 0 if not provided

END IF;

END $$

DELIMITER ;