**Stored Procedure**

**1. AddBudget Procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `AddBudget`(

IN departmentID INT,

IN totalBudget DECIMAL(10, 2),

IN expenditure DECIMAL(10, 2)

)

BEGIN

INSERT INTO Budget (DepartmentID, TotalBudget, Expenditure)

VALUES (departmentID, totalBudget, expenditure);

END;

**Explanation:** This procedure inserts a new budget record for a department. It takes three parameters:

* departmentID: The ID of the department for which the budget is being created.
* totalBudget: The total allocated budget for the department.
* expenditure: The current expenditure for the department.

The procedure inserts these values into the Budget table, associating the budget with the specific department.

**2. AddContract Procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `AddContract`(

IN vendorID INT,

IN contractTerms TEXT,

IN startDate DATE,

IN endDate DATE,

IN renewalDate DATE,

IN status VARCHAR(50)

)

BEGIN

INSERT INTO Contract (VendorID, Terms, StartDate, EndDate, RenewalDate, Status)

VALUES (vendorID, contractTerms, startDate, endDate, renewalDate, Status);

END;

**Explanation:** This procedure adds a new contract between the vendor and the company. It takes the following parameters:

* vendorID: The ID of the vendor who the contract is with.
* contractTerms: The text details of the contract terms.
* startDate: The start date of the contract.
* endDate: The end date of the contract.
* renewalDate: The date when the contract should be renewed.
* status: The current status of the contract (e.g., Active, Pending).

The procedure inserts these values into the Contract table.

**3. AddPerformance Procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `AddPerformance`(

IN vendorID INT,

IN rating DECIMAL(3, 2),

IN feedback TEXT

)

BEGIN

INSERT INTO Performance (VendorID, Rating, Feedback)

VALUES (vendorID, rating, feedback);

END;

**Explanation:** This procedure inserts a performance record for a vendor. It takes three parameters:

* vendorID: The ID of the vendor being evaluated.
* rating: The performance rating of the vendor (a decimal value).
* feedback: Additional comments or feedback regarding the vendor's performance.

It inserts this information into the Performance table, which records the vendor’s performance evaluation.

**4. AddPurchaseOrder Procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `AddPurchaseOrder`(

IN contractID INT,

IN departmentID INT,

IN items TEXT,

IN totalCost DECIMAL(10, 2),

IN status VARCHAR(50)

)

BEGIN

INSERT INTO PurchaseOrder (ContractID, DepartmentID, Items, TotalCost, Status)

VALUES (contractID, departmentID, items, totalCost, status);

END;

**Explanation:** This procedure adds a purchase order related to a contract and department. It takes the following parameters:

* contractID: The ID of the contract associated with the purchase order.
* departmentID: The department responsible for the purchase order.
* items: A description of the items being purchased.
* totalCost: The total cost of the purchase order.
* status: The status of the purchase order (e.g., Fulfilled, Pending).

This procedure inserts the data into the PurchaseOrder table.

**5. AddUser Procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `AddUser`(

IN userRole ENUM('Admin', 'Procurement Manager', 'Department Head'),

IN userName VARCHAR(100),

IN userEmail VARCHAR(255)

)

BEGIN

INSERT INTO User (Role, Name, Email)

VALUES (userRole, userName, userEmail);

END;

**Explanation:** This procedure adds a new user to the User table. It accepts the following parameters:

* userRole: The role of the user, which can be 'Admin', 'Procurement Manager', or 'Department Head'.
* userName: The name of the user.
* userEmail: The email address of the user.

The procedure inserts these values into the User table, effectively adding a new user.

**6. AddVendor Procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `AddVendor`(

IN vendorName VARCHAR(100),

IN contactDetails VARCHAR(255),

IN serviceCategory VARCHAR(100),

IN complianceCert VARCHAR(255)

)

BEGIN

INSERT INTO Vendor (Name, ContactDetails, ServiceCategory, ComplianceCertification)

VALUES (vendorName, contactDetails, serviceCategory, complianceCert);

END;

**Explanation:** This procedure adds a new vendor to the Vendor table. It takes the following parameters:

* vendorName: The name of the vendor.
* contactDetails: The contact details for the vendor (e.g., phone number, email).
* serviceCategory: The category of services that the vendor provides.
* complianceCert: Any compliance certification the vendor holds (e.g., ISO certification).

The procedure inserts this data into the Vendor table, effectively creating a new vendor record.

**7. AssignTask Procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `AssignTask`(

IN assignedTo INT,

IN departmentID INT,

IN taskDescription TEXT,

IN taskStatus VARCHAR(50),

IN dueDate DATE

)

BEGIN

INSERT INTO Task (AssignedTo, DepartmentID, Description, Status, DueDate)

VALUES (assignedTo, departmentID, taskDescription, taskStatus, dueDate);

END;

**Explanation:** This procedure assigns a task to a user. It takes the following parameters:

* assignedTo: The ID of the user to whom the task is assigned.
* departmentID: The department that is responsible for the task.
* taskDescription: A textual description of the task.
* taskStatus: The current status of the task (e.g., Pending, In Progress, Completed).
* dueDate: The due date for completing the task.

The procedure inserts this task into the Task table, assigning it to the specified user and department.

**8. MarkNotificationAsRead Procedure**

CREATE DEFINER=`root`@`localhost` PROCEDURE `MarkNotificationAsRead`(

IN notificationID INT

)

BEGIN

UPDATE Notification

SET IsRead = TRUE

WHERE NotificationID = notificationID;

END;

**Explanation:** This procedure updates a notification’s status to “read”. It takes one parameter:

* notificationID: The ID of the notification to be marked as read.

When executed, the procedure updates the IsRead field in the Notification table, setting it to TRUE for the specified notification.

**How to Use This SQL Code:**

* **Copy and paste** the SQL code for each procedure into your SQL client (such as MySQL Workbench or phpMyAdmin) to create these stored procedures in your database.
* **Parameters**: Each procedure requires specific input parameters. Ensure that the correct data types and values are passed when calling these procedures.
* **Execution**: You can execute these procedures by calling them in your SQL client, for example:
* CALL AddVendor('Tech Supplies Co.', '123-456-7890', 'IT Equipment', 'ISO 9001');