Here's how you can implement PL/SQL scenarios with cursors and triggers for the given commands ALTER, DROP, RENAME, MERGE, ROLLBACK, GRANT, and REVOKE in the context of your Store Management System:

**### SCENARIO 1: UPDATE SALARY WITH EXPLICIT CURSOR**

\*\*Objective:\*\* Update the salary of all staff members by a certain percentage.

```sql

-- PL/SQL Procedure with Cursor to update staff salaries

-- Anonymous PL/SQL block to update staff salaries

DECLARE

CURSOR staff\_cursor IS

SELECT Staff\_ID, Salary FROM Staff;

BEGIN

FOR rec IN staff\_cursor LOOP

UPDATE Staff

SET Salary = rec.Salary \* (1 + p\_percentage / 100)

WHERE Staff\_ID = rec.Staff\_ID;

END LOOP;

COMMIT;

END;

/

-- Example Usage: Increase all staff salaries by 10%

BEGIN

EXECUTE update\_staff\_salary(10);

END;

/```

**### SCENARIO 2: DROP COMMAND WITH ROW LEVEL TRIGGER**

\*\*Objective:\*\* When a store is dropped, delete all associated data such as bills, purchase registers, sales registers, etc.

```sql

-- Trigger to automatically delete associated data when a store is dropped

CREATE OR REPLACE TRIGGER drop\_store\_trigger

BEFORE DELETE ON Store

FOR EACH ROW

BEGIN

DELETE FROM Bills WHERE Bill\_no IN (SELECT Bill\_no FROM Bills WHERE Store\_ID = :OLD.Store\_ID);

DELETE FROM Purchase\_register WHERE Prod\_ID IN (SELECT Prod\_ID FROM Purchase\_register WHERE Store\_ID = :OLD.Store\_ID);

DELETE FROM Sales\_register WHERE Bill\_no IN (SELECT Bill\_no FROM Sales\_register WHERE Store\_ID = :OLD.Store\_ID);

-- Add more deletes for other associated data

END;

/

-- Example Usage: Drop a store

DELETE FROM Store WHERE Store\_ID = 1; -- This will automatically trigger the deletion of associated data

```

**### SCENARIO 3: RENAME COMMAND WITH PL/SQL**

\*\*Objective:\*\* Rename a store manager.

```sql

-- PL/SQL Procedure to rename a store manager

CREATE OR REPLACE PROCEDURE rename\_manager(p\_manager\_id NUMBER, p\_new\_name VARCHAR2) IS

BEGIN

UPDATE Store\_Manager

SET Manager\_Name = p\_new\_name

WHERE Manager\_ID = p\_manager\_id;

COMMIT;

END;

/

-- Example Usage: Rename a store manager

BEGIN

rename\_manager(1, 'New Manager Name');

END;

/

```

**### SCENARIO 4: MERGE COMMAND WITH PL/SQL**

\*\*Objective:\*\* Merge feedback from customers into a single table.

```sql

-- PL/SQL Procedure to merge customer feedback into a single table

CREATE OR REPLACE PROCEDURE merge\_feedback IS

BEGIN

MERGE INTO Customer\_Feedback tgt

USING (SELECT \* FROM Other\_Customer\_Feedback) src

ON (tgt.Feedback\_ID = src.Feedback\_ID)

WHEN MATCHED THEN

UPDATE SET tgt.Feedback\_text = src.Feedback\_text

WHEN NOT MATCHED THEN

INSERT (Feedback\_ID, Cust\_ID, Feedback\_text)

VALUES (src.Feedback\_ID, src.Cust\_ID, src.Feedback\_text);

COMMIT;

END;

/

-- Example Usage: Merge customer feedback

BEGIN

merge\_feedback;

END;

/

```

**### SCENARIO 5: ROLLBACK COMMAND WITH PL/SQL**

\*\*Objective:\*\* Rollback a transaction if an error occurs while updating staff details.

```sql

-- PL/SQL Procedure to update staff details and rollback if an error occurs

CREATE OR REPLACE PROCEDURE update\_staff\_details(p\_staff\_id NUMBER, p\_new\_phone VARCHAR2) IS

BEGIN

UPDATE Staff

SET Staff\_ph1 = p\_new\_phone

WHERE Staff\_ID = p\_staff\_id;

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

RAISE;

END;

/

-- Example Usage: Update staff details with a possible rollback

BEGIN

update\_staff\_details(1, '1234567890');

END;

/

```

**### SCENARIO 6: GRANT AND REVOKE COMMANDS**

\*\*Objective:\*\* Grant and revoke privileges to a specific user.

```sql

-- Grant SELECT privilege on Store table to a user

GRANT SELECT ON Store TO username;

-- Revoke SELECT privilege on Store table from a user

REVOKE SELECT ON Store FROM username;

```

These scenarios provide examples of how you can use PL/SQL with cursors and triggers to perform various operations like ALTER, DROP, RENAME, MERGE, ROLLBACK, GRANT, and REVOKE in your Store Management System.

X-X-X  
  
Sure, let's add a scenario to use SAVEPOINT and ROLLBACK commands to handle a specific situation in the context of updating staff details.

**### SCENARIO 7: SAVEPOINT AND ROLLBACK COMMANDS**

\*\*Objective:\*\* Use SAVEPOINT to mark a point in a transaction and ROLLBACK to undo changes up to that point.

```sql

-- PL/SQL Procedure to update staff details with SAVEPOINT and ROLLBACK

CREATE OR REPLACE PROCEDURE update\_staff\_details\_with\_rollback(p\_staff\_id NUMBER, p\_new\_phone VARCHAR2) IS

BEGIN

-- Set a savepoint

SAVEPOINT before\_update;

-- Update staff details

UPDATE Staff

SET Staff\_ph1 = p\_new\_phone

WHERE Staff\_ID = p\_staff\_id;

-- Check if the phone number format is valid (for demonstration purposes)

IF LENGTH(p\_new\_phone) <> 10 THEN

-- If phone number format is not valid, rollback to the savepoint

ROLLBACK TO before\_update;

RAISE\_APPLICATION\_ERROR(-20001, 'Invalid phone number format. Rollback applied.');

END IF;

-- If phone number format is valid, commit the transaction

COMMIT;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK TO before\_update;

RAISE;

END;

/

```

### Example Usage:

```sql

-- Attempt to update staff details with invalid phone number format

BEGIN

update\_staff\_details\_with\_rollback(1, '123'); -- Invalid phone number format

END;

/

```

\*\*Output:\*\* The above code will raise an exception due to the invalid phone number format and rollback the transaction to the savepoint `before\_update`. The changes made to the staff details will be undone.

This scenario demonstrates how to use SAVEPOINT and ROLLBACK commands to handle specific situations within a transaction and ensure data integrity.

X-X-X  
  
**### SCENARIO 8: ADD COLUMN IN STAFF USING ALTER**

Sure, let's add a scenario where the `ALTER` command is used to modify the structure of an existing table. For example, let's say we want to add a new column named `Hire\_Date` to the `Staff` table:

```sql

-- Scenario: Use ALTER command to add a new column to the Staff table

ALTER TABLE Staff

ADD Hire\_Date DATE;

```

In this scenario, the `ALTER TABLE` command is used to modify the structure of the `Staff` table by adding a new column named `Hire\_Date` of data type `DATE`. This allows us to store the hire date of each staff member in the table.  
  
X-X-X  
  
**REGULAR QUERIES**Sure, here are simple MySQL commands for performing various operations on the given tables:

**### UPDATE Command:**

To update data in a table, you can use the `UPDATE` command. For example, to update the salary of a staff member with Staff\_ID = 1:

```sql

UPDATE Staff

SET Salary = 35000

WHERE Staff\_ID = 1;

```

**### DELETE Command:**

To delete data from a table, you can use the `DELETE` command. For example, to delete a customer with Cust\_ID = 3:

```sql

DELETE FROM Customer

WHERE Cust\_ID = 3;

```

**### ALTER Command:**

To modify the structure of a table, you can use the `ALTER TABLE` command. For example, to add a new column named `Email` to the Staff table:

```sql

ALTER TABLE Staff

ADD Email VARCHAR(255);

```

**### DROP Command:**

To delete a table or a column from a table, you can use the `DROP` command. For example, to drop the `Email` column from the Staff table:

```sql

ALTER TABLE Staff

DROP COLUMN Email;

```

**### TRUNCATE Command:**

To remove all rows from a table, you can use the `TRUNCATE` command. For example, to remove all data from the Bills table:

```sql

TRUNCATE TABLE Bills;

```

**### RENAME Command:**

To rename a table, you can use the `RENAME TABLE` command. For example, to rename the Customer table to Customer\_Info:

```sql

RENAME TABLE Customer TO Customer\_Info;

```

**### MERGE Command:**

MySQL does not have a direct `MERGE` command like some other database systems. However, you can achieve similar functionality using `INSERT INTO ... ON DUPLICATE KEY UPDATE` syntax. For example, to merge data into the Inventory table:

```sql

INSERT INTO Inventory (Prod\_ID, Prod\_type, Quantity, Exp\_date)

VALUES (1, 'Electronics', 100, '2024-12-31')

ON DUPLICATE KEY UPDATE Quantity = Quantity + VALUES(Quantity);

```

**### SAVEPOINT and ROLLBACK Commands:**

MySQL supports transactions and savepoints, which you can use to control transactions. For example, to set a savepoint:

```sql

SAVEPOINT before\_update;

```

And to rollback to a savepoint:

```sql

ROLLBACK TO before\_update;

```

**### GRANT Command:**

To grant privileges to a user, you can use the `GRANT` command. For example, to grant SELECT privilege on the Store table to a user:

```sql

GRANT SELECT ON Store TO username;

```

**### REVOKE Command:**

To revoke privileges from a user, you can use the `REVOKE` command. For example, to revoke SELECT privilege on the Store table from a user:

```sql

REVOKE SELECT ON Store FROM username;

```

These are basic MySQL commands for performing various operations on the given tables. Make sure to adjust the table names and column names according to your database schema.