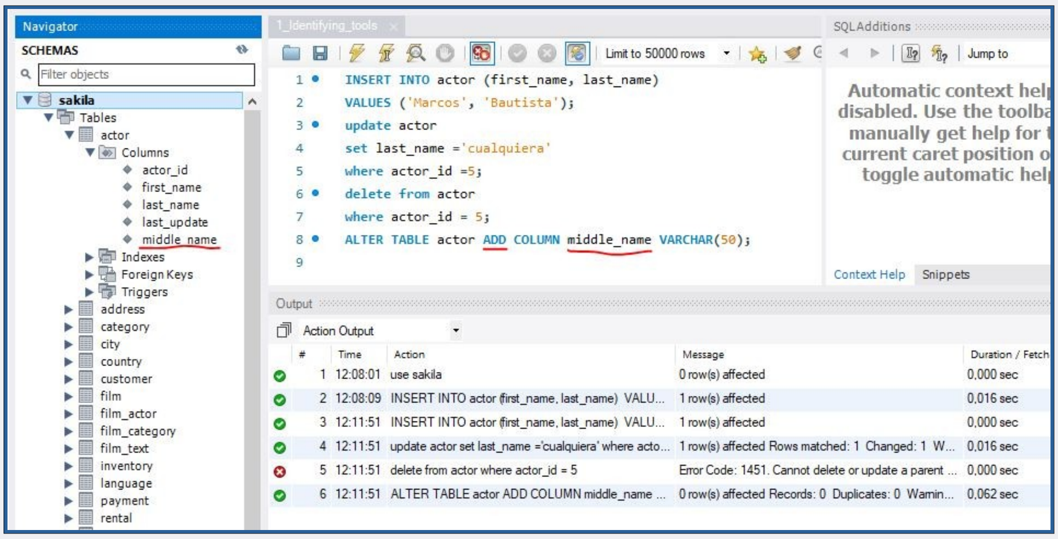
Marcos Bautista

Final report

1. Identifying Tools and Statements for Modifying Database Content

Main SQL statements for modify database content:

- INSERT: Adds new record to a table.
- UPDATE: Updates existing records in the table.
- DELETE : Deletes records from a table.
- ALTER: Modifies structure of a table; adds new columns.



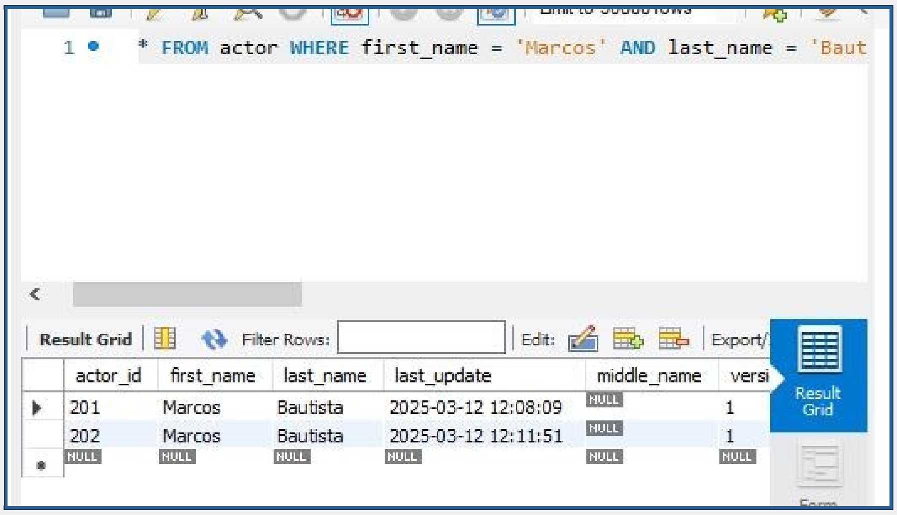
```sql
INSERT INTO actor (first_name, last_name)
VALUES ('Marcos', 'Bautista');
update actor
set last_name ='cualquiera'
where actor_id =5;
delete from actor
where actor_id = 5;
ALTER TABLE actor ADD COLUMN middle_name VARCHAR(50);
```

Automatic context help disabled. Use the toolba manually get help for t current caret position o toggle automatic hel

Output

Action Output

```
#  Time      Action                                                Message                                                 Duration / Fetch
1  12:08:01  use sakila                                            0 row(s) affected                                       0,000 sec
2  12:08:09  INSERT INTO actor (first_name, last_name)  VALU...    1 row(s) affected                                       0,016 sec
3  12:11:51  INSERT INTO actor (first_name, last_name)  VALU...    1 row(s) affected                                       0,000 sec
4  12:11:51  update actor set last_name ='cualquiera' where acto... 1 row(s) affected Rows matched: 1  Changed: 1  W...  0,016 sec
5  12:11:51  delete from actor where actor_id = 5                  Error Code: 1451. Cannot delete or update a parent ...  0,000 sec
6  12:11:51  ALTER TABLE actor ADD COLUMN middle_name ...          0 row(s) affected Records: 0  Duplicates: 0  Warnin...  0,062 sec
```

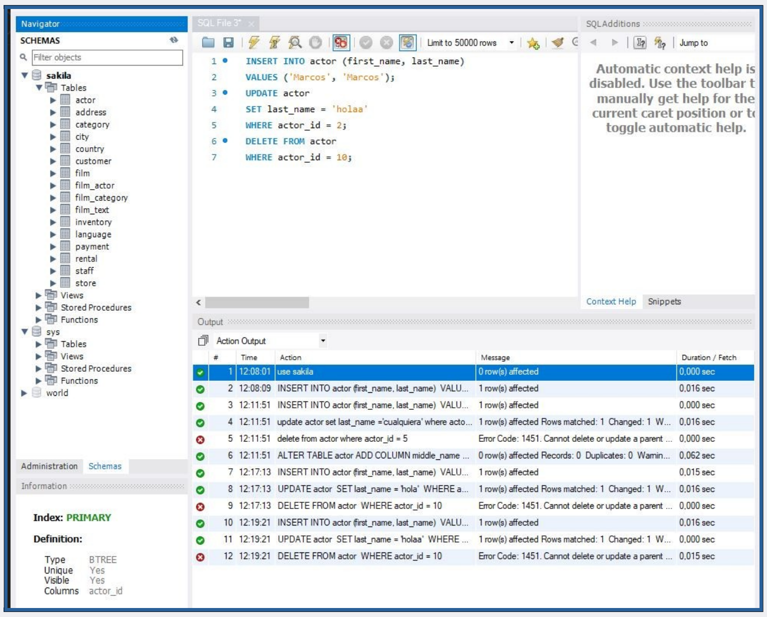


```sql
* FROM actor WHERE first_name = 'Marcos' AND last_name = 'Baut
```

```
actor_id  first_name  last_name  last_update          middle_name  versi
201       Marcos      Bautista   2025-03-12 12:08:09  NULL         1
202       Marcos      Bautista   2025-03-12 12:11:51  NULL         1
NULL      NULL        NULL       NULL                 NULL         NULL
```

2. Data Insertion, Deletion, and Update

Insert new record, update existing record, delete record...
As in the previous point, we have commands to perform these actions

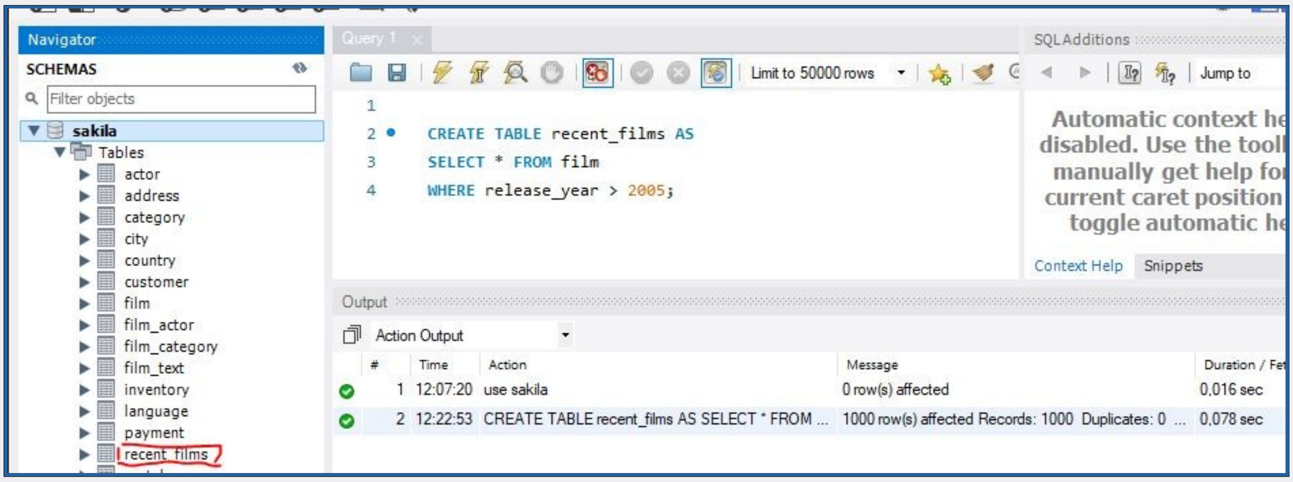


```sql
INSERT INTO actor (first_name, last_name)
VALUES ('Marcos', 'Marcos');
UPDATE actor
SET last_name = 'holaa'
WHERE actor_id = 2;
DELETE FROM actor
WHERE actor_id = 10;
```

```
 #   Time      Action                                                 Message                                                    Duration / Fetch
 1   12:08:01  use sakila                                             0 row(s) affected                                          0,000 sec
 2   12:08:09  INSERT INTO actor (first_name, last_name)  VALU...     1 row(s) affected                                          0,016 sec
 3   12:11:51  INSERT INTO actor (first_name, last_name)  VALU...     1 row(s) affected                                          0,000 sec
 4   12:11:51  update actor set last_name ='cualquiera' where acto... 1 row(s) affected Rows matched: 1  Changed: 1  W...        0,016 sec
 5   12:11:51  delete from actor where actor_id = 5                   Error Code: 1451. Cannot delete or update a parent ...     0,000 sec
 6   12:11:51  ALTER TABLE actor ADD COLUMN middle_name ...           0 row(s) affected Records: 0  Duplicates: 0  Warnin...     0,062 sec
 7   12:17:13  INSERT INTO actor (first_name, last_name)  VALU...     1 row(s) affected                                          0,015 sec
 8   12:17:13  UPDATE actor  SET last_name = 'hola'  WHERE a...       1 row(s) affected Rows matched: 1  Changed: 1  W...        0,016 sec
 9   12:17:13  DELETE FROM actor  WHERE actor_id = 10                 Error Code: 1451. Cannot delete or update a parent ...     0,000 sec
10   12:19:21  INSERT INTO actor (first_name, last_name)  VALU...     1 row(s) affected                                          0,016 sec
11   12:19:21  UPDATE actor  SET last_name = 'holaa'  WHERE ...       1 row(s) affected Rows matched: 1  Changed: 1  W...        0,000 sec
12   12:19:21  DELETE FROM actor  WHERE actor_id = 10                 Error Code: 1451. Cannot delete or update a parent ...     0,015 sec
```

3. Creating a Table from a Query Result

Create table ___ as - select * from & where, for filter as we like



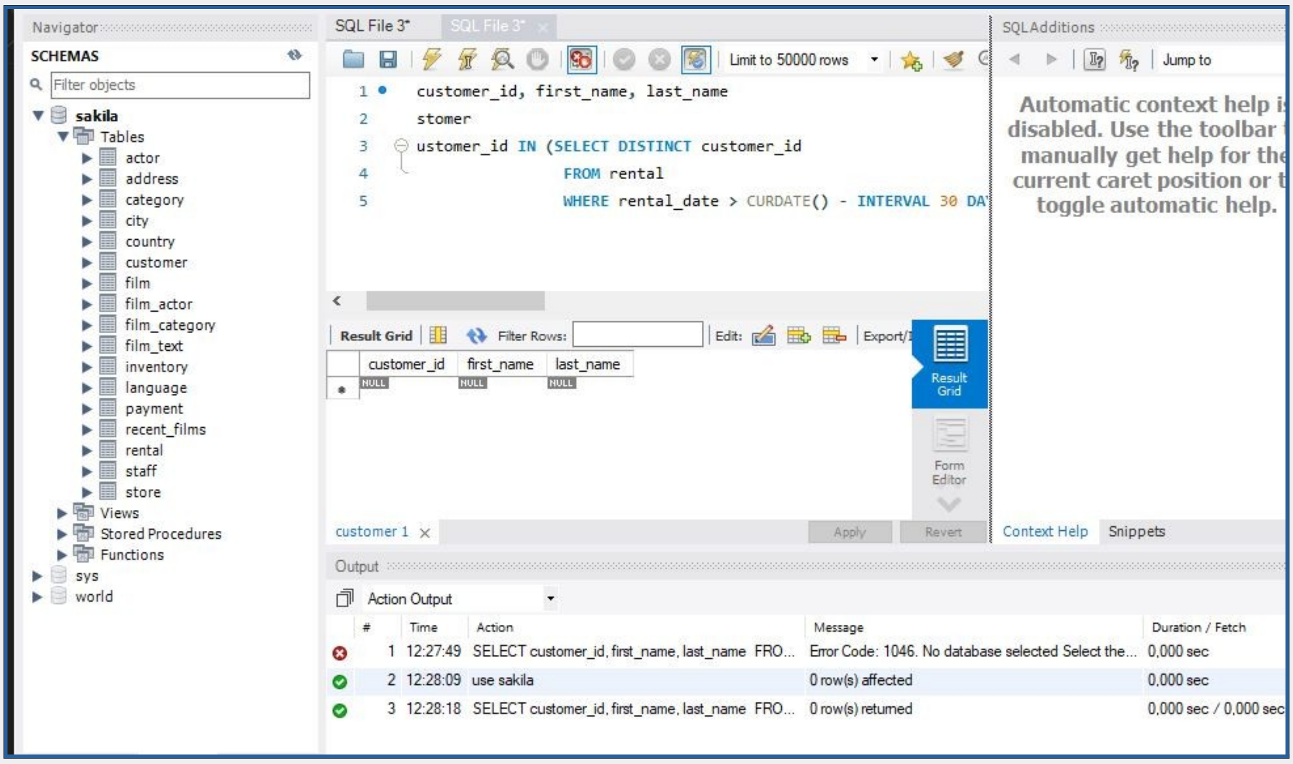
```sql
CREATE TABLE recent_films AS
SELECT * FROM film
WHERE release_year > 2005;
```

```
#   Time      Action                                         Message                                                     Duration / Fe
1   12:07:20  use sakila                                     0 row(s) affected                                           0,016 sec
2   12:22:53  CREATE TABLE recent_films AS SELECT * FROM ... 1000 row(s) affected Records: 1000  Duplicates: 0  ...      0,078 sec
```

The new table was successfully created

4. Designing Complex SQL Scripts

List customers who have rented a film in the last 30 days

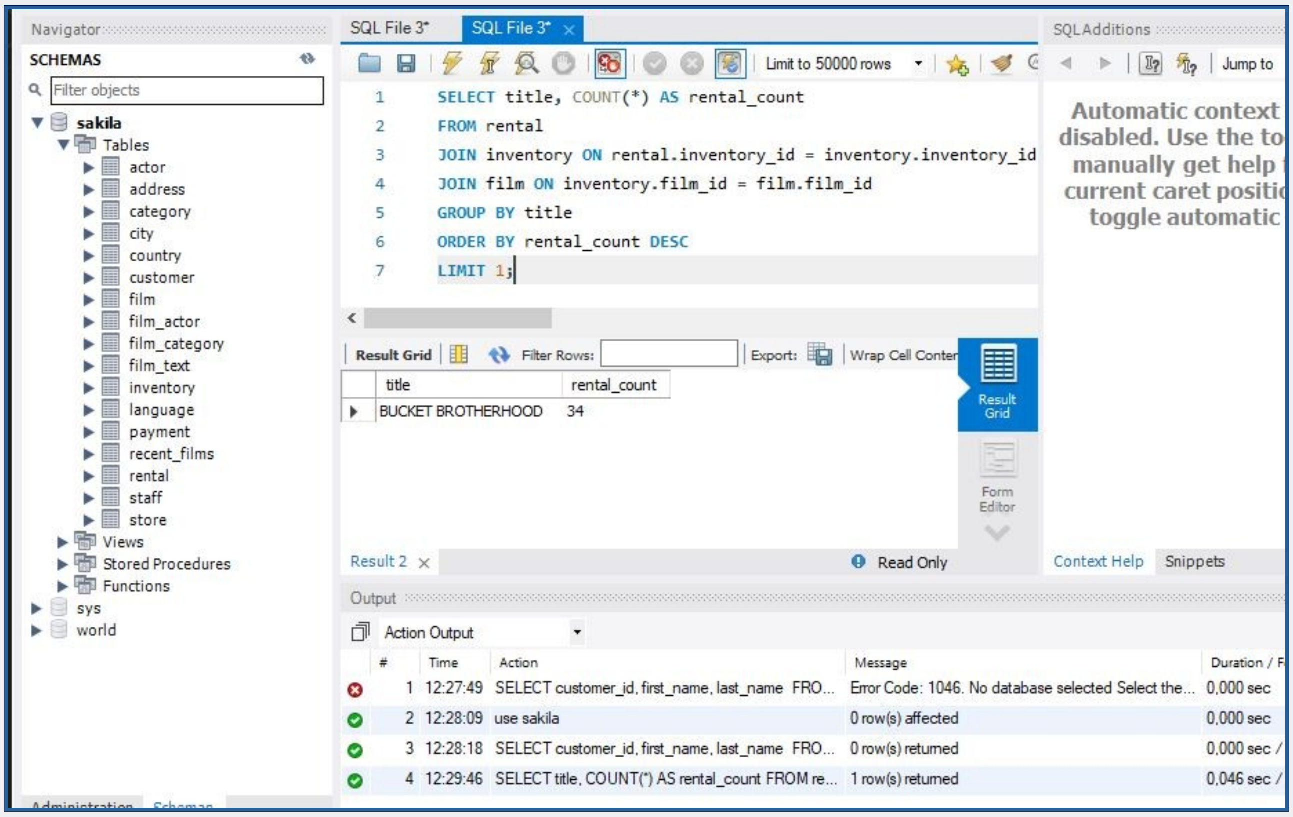


```sql
customer_id, first_name, last_name
stomer
ustomer_id IN (SELECT DISTINCT customer_id
               FROM rental
               WHERE rental_date > CURDATE() - INTERVAL 30 DA
```

	customer_id	first_name	last_name
*	NULL	NULL	NULL

	#	Time	Action	Message	Duration / Fetch
✗	1	12:27:49	SELECT customer_id, first_name, last_name FRO...	Error Code: 1046. No database selected Select the...	0,000 sec
✓	2	12:28:09	use sakila	0 row(s) affected	0,000 sec
✓	3	12:28:18	SELECT customer_id, first_name, last_name FRO...	0 row(s) returned	0,000 sec / 0,000 sec

Identify the most rented film

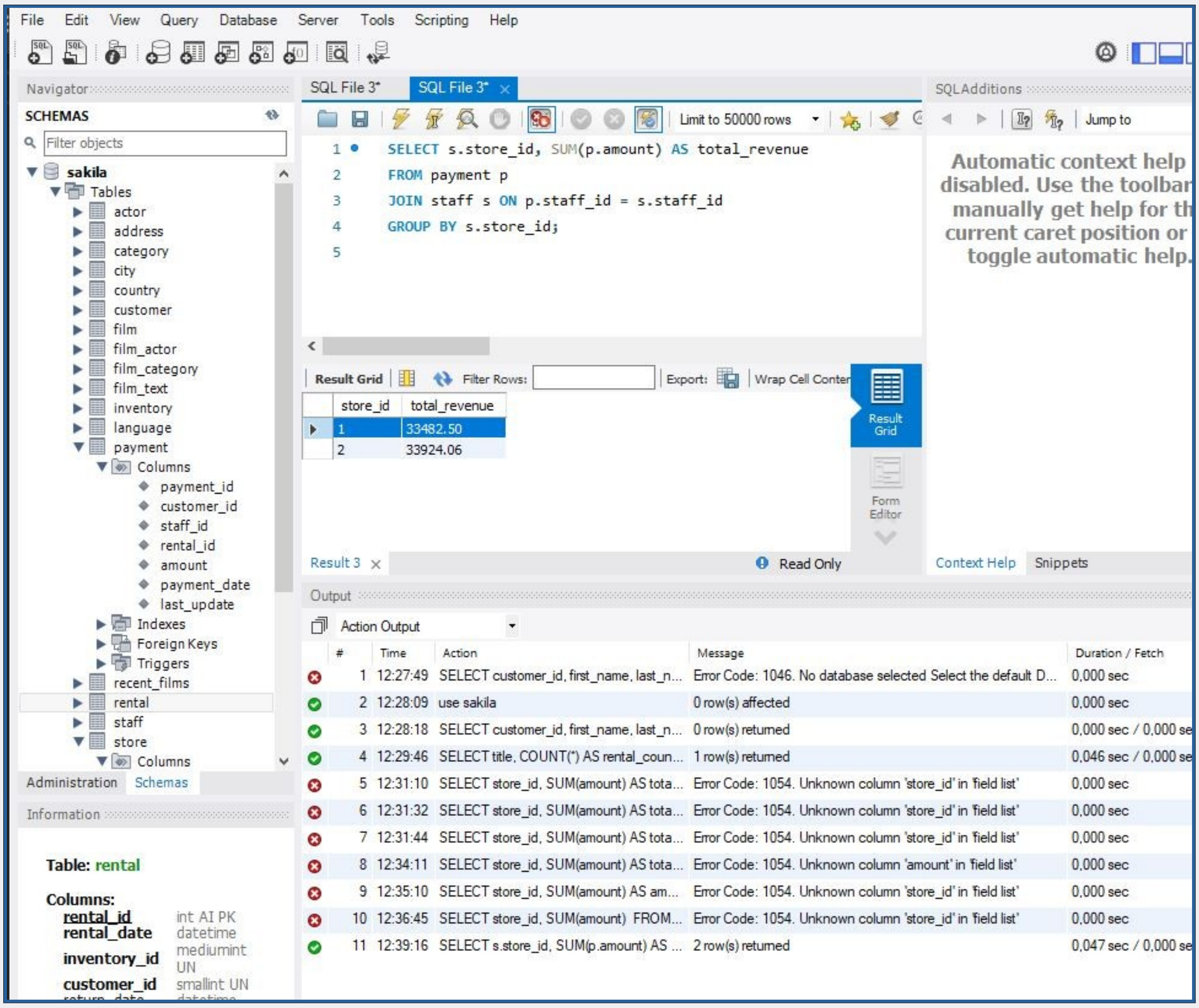


```sql
SELECT title, COUNT(*) AS rental_count
FROM rental
JOIN inventory ON rental.inventory_id = inventory.inventory_id
JOIN film ON inventory.film_id = film.film_id
GROUP BY title
ORDER BY rental_count DESC
LIMIT 1;
```

title	rental_count
BUCKET BROTHERHOOD	34

	#	Time	Action	Message	Duration / F
✗	1	12:27:49	SELECT customer_id, first_name, last_name FRO...	Error Code: 1046. No database selected Select the...	0,000 sec
✓	2	12:28:09	use sakila	0 row(s) affected	0,000 sec
✓	3	12:28:18	SELECT customer_id, first_name, last_name FRO...	0 row(s) returned	0,000 sec /
✓	4	12:29:46	SELECT title, COUNT(*) AS rental_count FROM re...	1 row(s) returned	0,046 sec /

Display the total revenue per store



```sql
SELECT s.store_id, SUM(p.amount) AS total_revenue
FROM payment p
JOIN staff s ON p.staff_id = s.staff_id
GROUP BY s.store_id;
```

store_id	total_revenue
1	33482.50
2	33924.06

#	Time	Action	Message	Duration / Fetch
1	12:27:49	SELECT customer_id, first_name, last_n...	Error Code: 1046. No database selected Select the default D...	0,000 sec
2	12:28:09	use sakila	0 row(s) affected	0,000 sec
3	12:28:18	SELECT customer_id, first_name, last_n...	0 row(s) returned	0,000 sec / 0,000 se
4	12:29:46	SELECT title, COUNT(*) AS rental_coun...	1 row(s) returned	0,046 sec / 0,000 se
5	12:31:10	SELECT store_id, SUM(amount) AS tota...	Error Code: 1054. Unknown column 'store_id' in 'field list'	0,000 sec
6	12:31:32	SELECT store_id, SUM(amount) AS tota...	Error Code: 1054. Unknown column 'store_id' in 'field list'	0,000 sec
7	12:31:44	SELECT store_id, SUM(amount) AS tota...	Error Code: 1054. Unknown column 'store_id' in 'field list'	0,000 sec
8	12:34:11	SELECT store_id, SUM(amount) AS tota...	Error Code: 1054. Unknown column 'amount' in 'field list'	0,000 sec
9	12:35:10	SELECT store_id, SUM(amount) AS am...	Error Code: 1054. Unknown column 'store_id' in 'field list'	0,000 sec
10	12:36:45	SELECT store_id, SUM(amount) FROM...	Error Code: 1054. Unknown column 'store_id' in 'field list'	0,000 sec
11	12:39:16	SELECT s.store_id, SUM(p.amount) AS ...	2 row(s) returned	0,047 sec / 0,000 se

Automatic context help disabled. Use the toolbar manually get help for th current caret position or toggle automatic help.

Table: rental

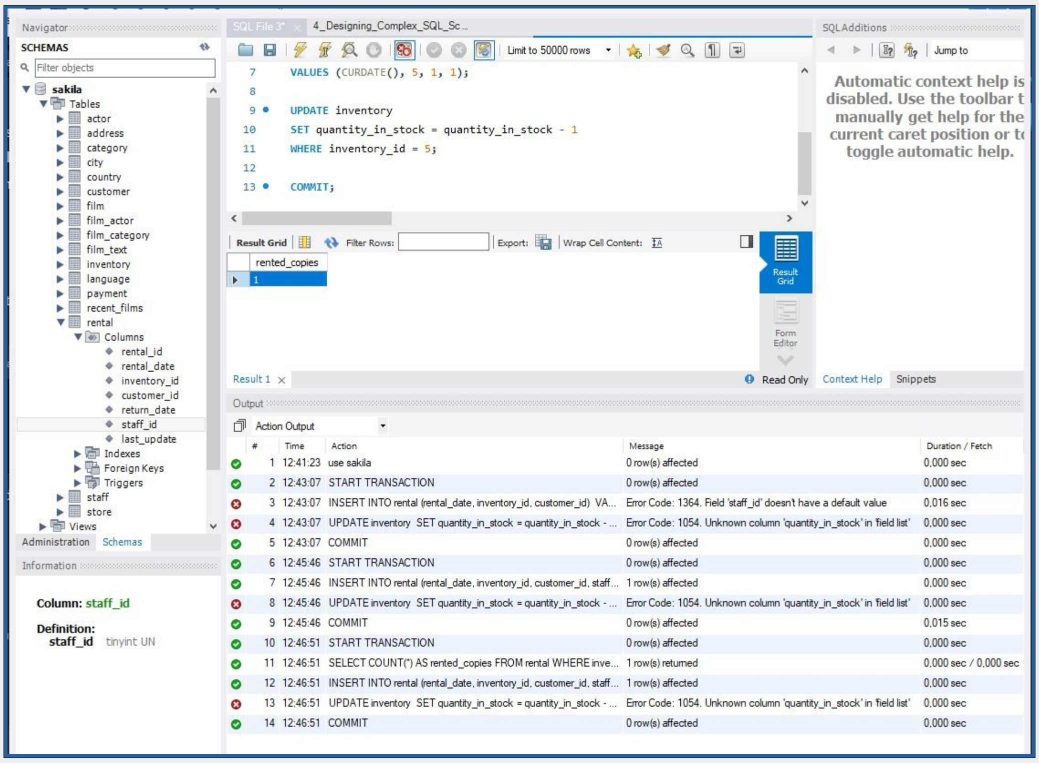
Columns:

rental_id	int AI PK
rental_date	datetime
inventory_id	mediumint UN
customer_id	smallint UN

5. Understanding transactions

A transaction ensures that a group of operations (inserts, updates, deletes) is completed as a unit, meaning either all changes are applied, or none are if there’s an error.

Therefore: start a transaction, perform actions, like inserting a new rental record, update inventory to reflect the rental , commit the transaction to save changes



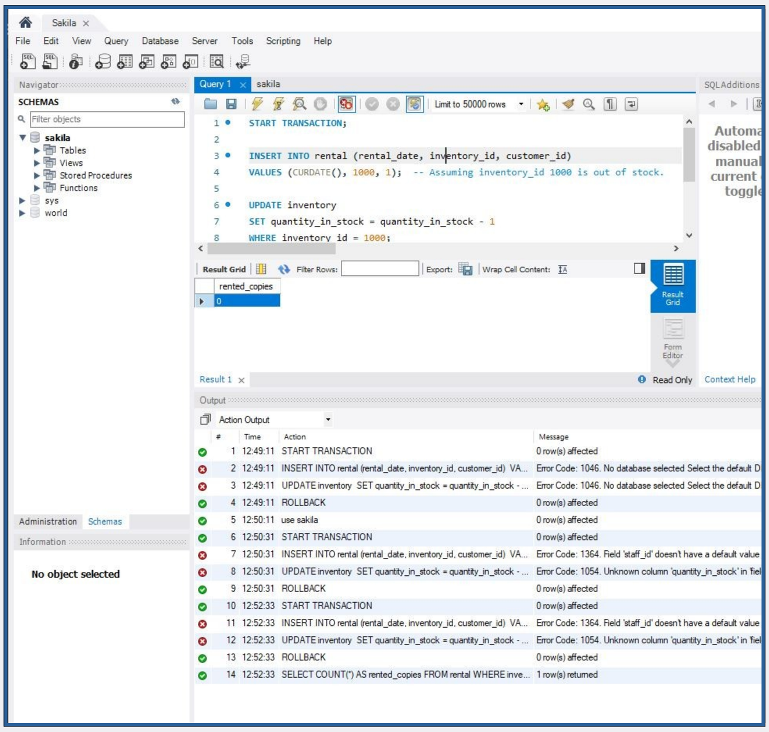
```
VALUES (CURDATE(), 5, 1, 1);

UPDATE inventory
SET quantity_in_stock = quantity_in_stock - 1
WHERE inventory_id = 5;

COMMIT;
```

```
rented_copies
1
```

```
#   Time      Action                                                          Message                                                              Duration / Fetch
1   12:41:23  use sakila                                                      0 row(s) affected                                                    0,000 sec
2   12:43:07  START TRANSACTION                                               0 row(s) affected                                                    0,000 sec
3   12:43:07  INSERT INTO rental (rental_date, inventory_id, customer_id) VA...  Error Code: 1364. Field 'staff_id' doesn't have a default value      0,016 sec
4   12:43:07  UPDATE inventory  SET quantity_in_stock = quantity_in_stock - ...  Error Code: 1054. Unknown column 'quantity_in_stock' in 'field list'  0,000 sec
5   12:43:07  COMMIT                                                          0 row(s) affected                                                    0,000 sec
6   12:45:46  START TRANSACTION                                               0 row(s) affected                                                    0,000 sec
7   12:45:46  INSERT INTO rental (rental_date, inventory_id, customer_id, staff...  1 row(s) affected                                                 0,000 sec
8   12:45:46  UPDATE inventory  SET quantity_in_stock = quantity_in_stock - ...  Error Code: 1054. Unknown column 'quantity_in_stock' in 'field list'  0,000 sec
9   12:45:46  COMMIT                                                          0 row(s) affected                                                    0,015 sec
10  12:46:51  START TRANSACTION                                               0 row(s) affected                                                    0,000 sec
11  12:46:51  SELECT COUNT(*) AS rented_copies FROM rental WHERE inve...      1 row(s) returned                                                    0,000 sec / 0,000 sec
12  12:46:51  INSERT INTO rental (rental_date, inventory_id, customer_id, staff...  1 row(s) affected                                                 0,000 sec
13  12:46:51  UPDATE inventory  SET quantity_in_stock = quantity_in_stock - ...  Error Code: 1054. Unknown column 'quantity_in_stock' in 'field list'  0,000 sec
14  12:46:51  COMMIT                                                          0 row(s) affected                                                    0,000 sec
```

6. Rolling Back Transactions

A rollback undoes changes if something goes wrong during the transaction.



```
START TRANSACTION;

INSERT INTO rental (rental_date, inventory_id, customer_id)
VALUES (CURDATE(), 1000, 1);  -- Assuming inventory_id 1000 is out of stock.

UPDATE inventory
SET quantity_in_stock = quantity_in_stock - 1
WHERE inventory_id = 1000;
```

```
rented_copies
0
```

```
#   Time      Action                                                          Message
1   12:49:11  START TRANSACTION                                               0 row(s) affected
2   12:49:11  INSERT INTO rental (rental_date, inventory_id, customer_id) VA...  Error Code: 1046. No database selected Select the default D
3   12:49:11  UPDATE inventory  SET quantity_in_stock = quantity_in_stock - ...  Error Code: 1046. No database selected Select the default D
4   12:49:11  ROLLBACK                                                        0 row(s) affected
5   12:50:11  use sakila                                                      0 row(s) affected
6   12:50:31  START TRANSACTION                                               0 row(s) affected
7   12:50:31  INSERT INTO rental (rental_date, inventory_id, customer_id) VA...  Error Code: 1364. Field 'staff_id' doesn't have a default value
8   12:50:31  UPDATE inventory  SET quantity_in_stock = quantity_in_stock - ...  Error Code: 1054. Unknown column 'quantity_in_stock' in 'fiel
9   12:50:31  ROLLBACK                                                        0 row(s) affected
10  12:52:33  START TRANSACTION                                               0 row(s) affected
11  12:52:33  INSERT INTO rental (rental_date, inventory_id, customer_id) VA...  Error Code: 1364. Field 'staff_id' doesn't have a default value
12  12:52:33  UPDATE inventory  SET quantity_in_stock = quantity_in_stock - ...  Error Code: 1054. Unknown column 'quantity_in_stock' in 'fiel
13  12:52:33  ROLLBACK                                                        0 row(s) affected
14  12:52:33  SELECT COUNT(*) AS rented_copies FROM rental WHERE inve...      1 row(s) returned
```

This initiates a transaction

rental

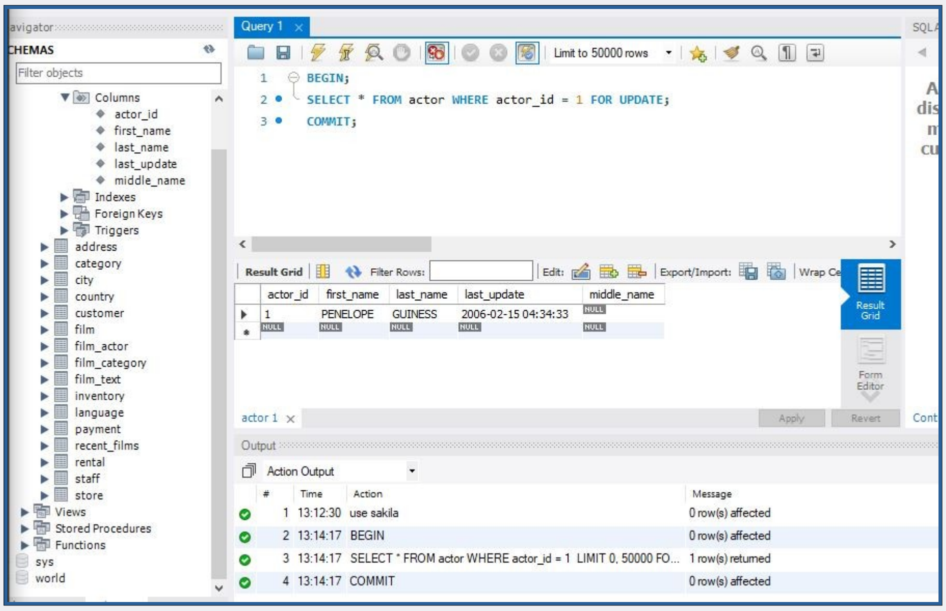
inventory

quantity_in_stock *inventory_id*

ROLLBACK

inventory

7. Understanding Record Locking Policies

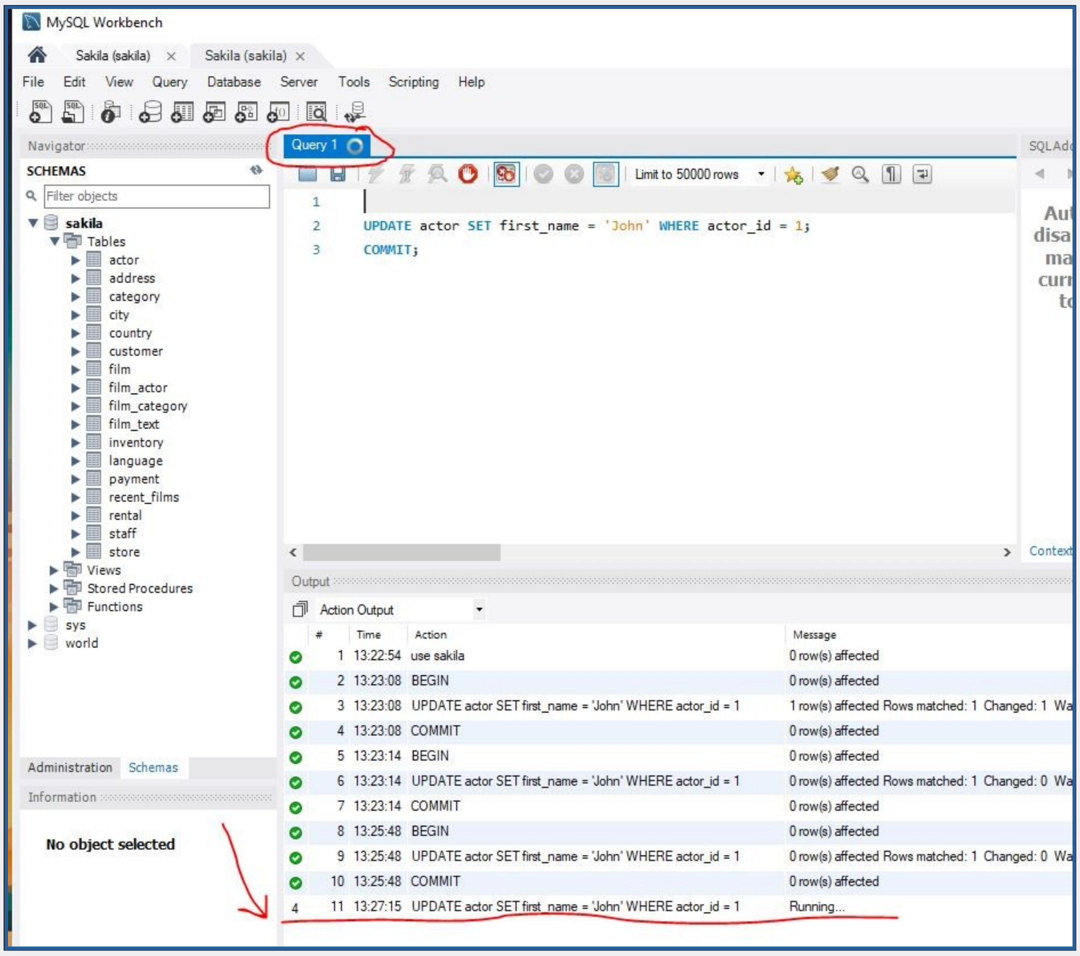


```
BEGIN;
SELECT * FROM actor WHERE actor_id = 1 FOR UPDATE;
COMMIT;
```

actor_id	first_name	last_name	last_update	middle_name
1	PENELOPE	GUINESS	2006-02-15 04:34:33	NULL
NULL	NULL	NULL	NULL	NULL

```
#   Time      Action                                                Message
1   13:12:30  use sakila                                            0 row(s) affected
2   13:14:17  BEGIN                                                 0 row(s) affected
3   13:14:17  SELECT * FROM actor WHERE actor_id = 1 LIMIT 0, 50000 FO...  1 row(s) returned
4   13:14:17  COMMIT                                                0 row(s) affected
```

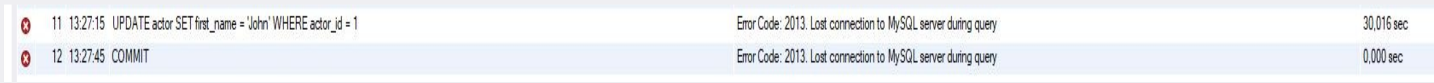
When a user locks a record to prevent other users from accessing it, assuming a conflict will happen. This is done using `SELECT ... FOR UPDATE`:



```
UPDATE actor SET first_name = 'John' WHERE actor_id = 1;
COMMIT;
```

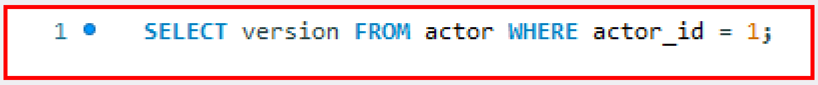
```
#   Time      Action                                                Message
1   13:22:54  use sakila                                            0 row(s) affected
2   13:23:08  BEGIN                                                 0 row(s) affected
3   13:23:08  UPDATE actor SET first_name = 'John' WHERE actor_id = 1   1 row(s) affected Rows matched: 1  Changed: 1  Wa
4   13:23:08  COMMIT                                                0 row(s) affected
5   13:23:14  BEGIN                                                 0 row(s) affected
6   13:23:14  UPDATE actor SET first_name = 'John' WHERE actor_id = 1   0 row(s) affected Rows matched: 1  Changed: 0  Wa
7   13:23:14  COMMIT                                                0 row(s) affected
8   13:25:48  BEGIN                                                 0 row(s) affected
9   13:25:48  UPDATE actor SET first_name = 'John' WHERE actor_id = 1   0 row(s) affected Rows matched: 1  Changed: 0  Wa
10  13:25:48  COMMIT                                                0 row(s) affected
11  13:27:15  UPDATE actor SET first_name = 'John' WHERE actor_id = 1   Running...
```

This is :<u>Pessimistic locking</u>



```
11  13:27:15  UPDATE actor SET first_name = 'John' WHERE actor_id = 1    Error Code: 2013. Lost connection to MySQL server during query    30,016 sec
12  13:27:45  COMMIT                                                    Error Code: 2013. Lost connection to MySQL server during query    0,000 sec
```

Optimistic Locking: Assumes no conflict, but checks before committing the changes. You would need a version column to track changes:

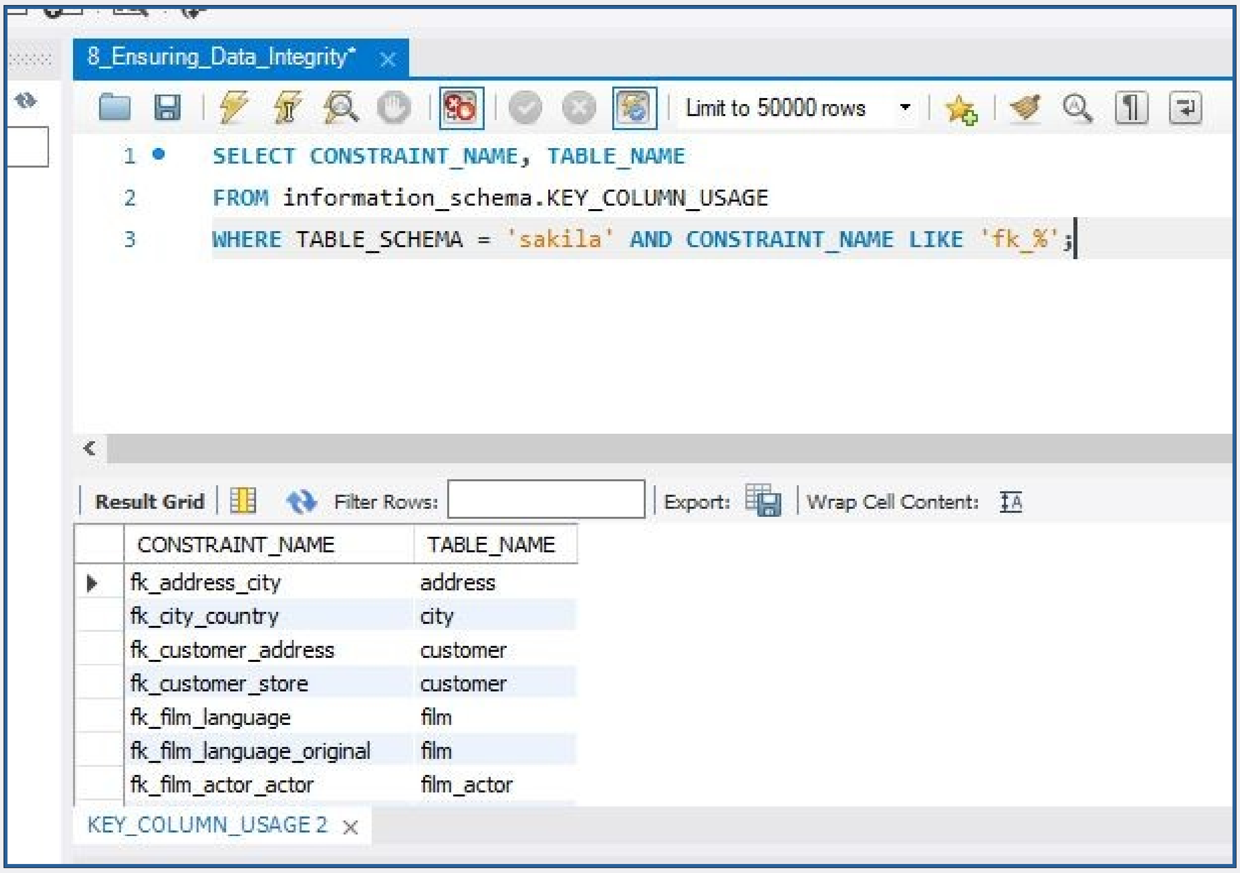


```
SELECT version FROM actor WHERE actor_id = 1;
```

8. Ensuring Data Integrity and Consistency

To ensure data integrity, you can:

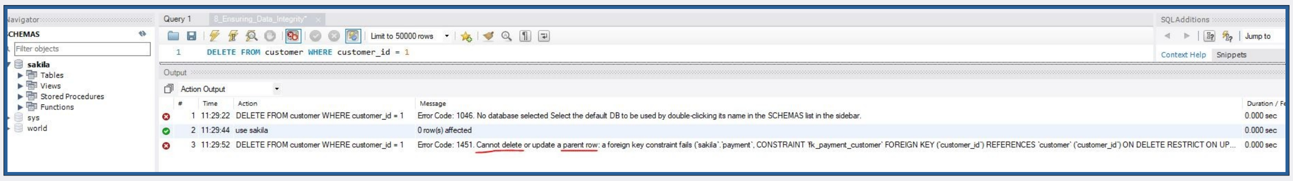
Foreign Key Constraints: Ensure data consistency between related tables (e.g., making sure no orders exist without a valid customer).


8_Ensuring_Data_Integrity*

```sql
SELECT CONSTRAINT_NAME, TABLE_NAME
FROM information_schema.KEY_COLUMN_USAGE
WHERE TABLE_SCHEMA = 'sakila' AND CONSTRAINT_NAME LIKE 'fk_%';
```

CONSTRAINT_NAME	TABLE_NAME
fk_address_city	address
fk_city_country	city
fk_customer_address	customer
fk_customer_store	customer
fk_film_language	film
fk_film_language_original	film
fk_film_actor_actor	film_actor

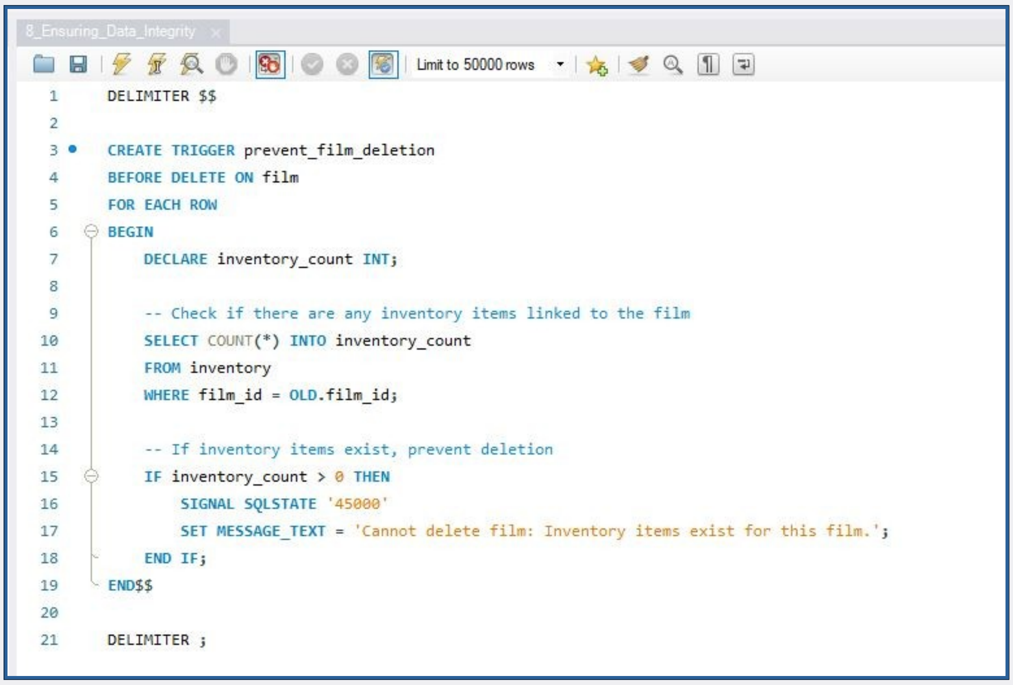
KEY_COLUMN_USAGE 2



```sql
DELETE FROM customer WHERE customer_id = 1
```

```
#  Time      Action                                    Message                                                                                                                                                                                     Duration / F
1  11:29:22  DELETE FROM customer WHERE customer_id = 1  Error Code: 1046. No database selected Select the default DB to be used by double-clicking its name in the SCHEMAS list in the sidebar.                                          0.000 sec
2  11:29:44  use sakila                                  0 row(s) affected                                                                                                                                                                          0.000 sec
3  11:29:52  DELETE FROM customer WHERE customer_id = 1  Error Code: 1451. Cannot delete or update a parent row: a foreign key constraint fails ('sakila'.'payment', CONSTRAINT 'fk_payment_customer' FOREIGN KEY ('customer_id') REFERENCES 'customer' ('customer_id') ON DELETE RESTRICT ON UP...  0.000 sec
```

Triggers: Automatically check or enforce rules on insert, update, or delete. For example, preventing a rental from being deleted if it's still active:



```sql
DELIMITER $$

CREATE TRIGGER prevent_film_deletion
BEFORE DELETE ON film
FOR EACH ROW
BEGIN
    DECLARE inventory_count INT;

    -- Check if there are any inventory items linked to the film
    SELECT COUNT(*) INTO inventory_count
    FROM inventory
    WHERE film_id = OLD.film_id;

    -- If inventory items exist, prevent deletion
    IF inventory_count > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cannot delete film: Inventory items exist for this film.';
    END IF;
END$$

DELIMITER ;
```