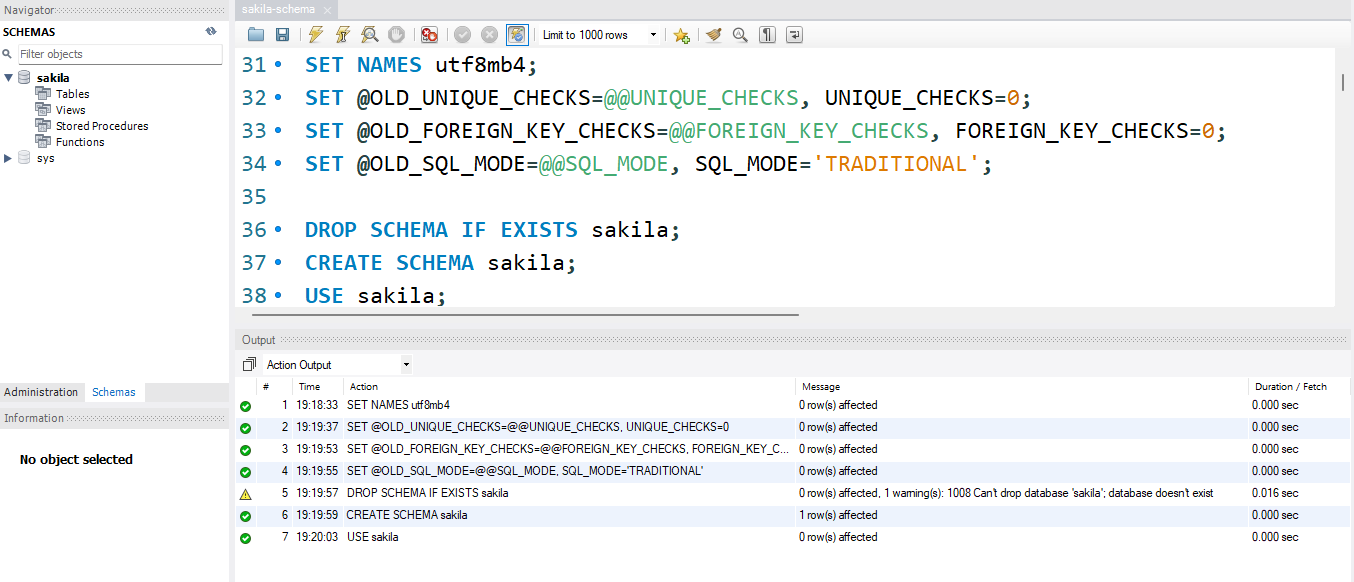
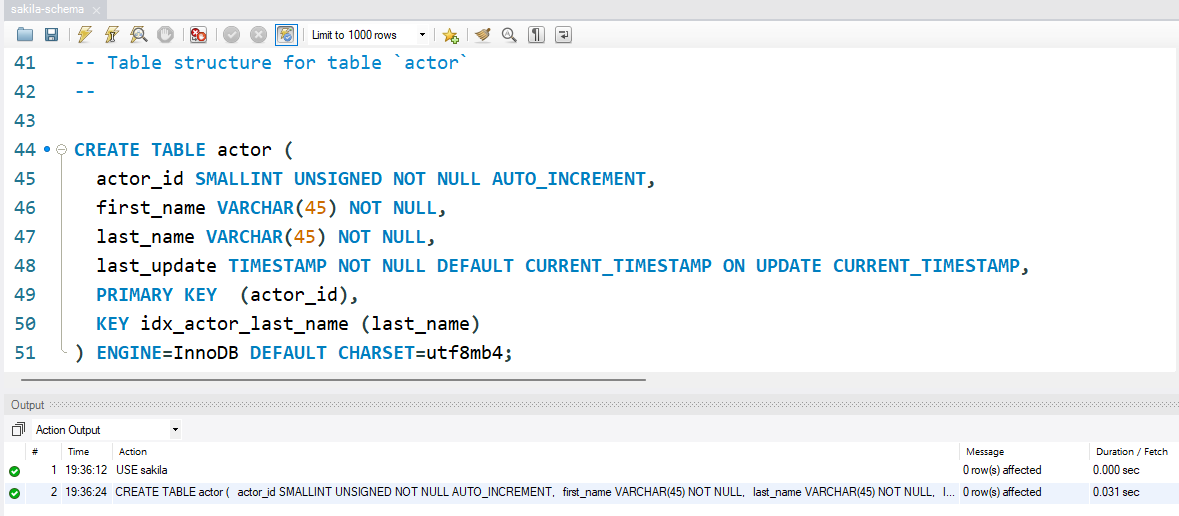
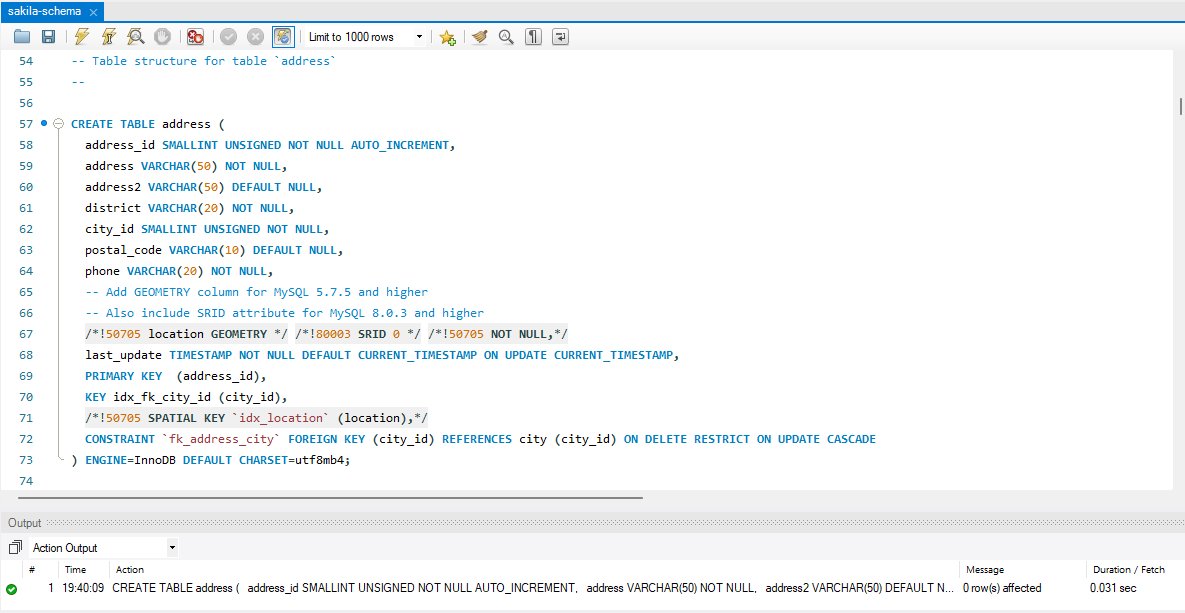
Creating and using database (schema):



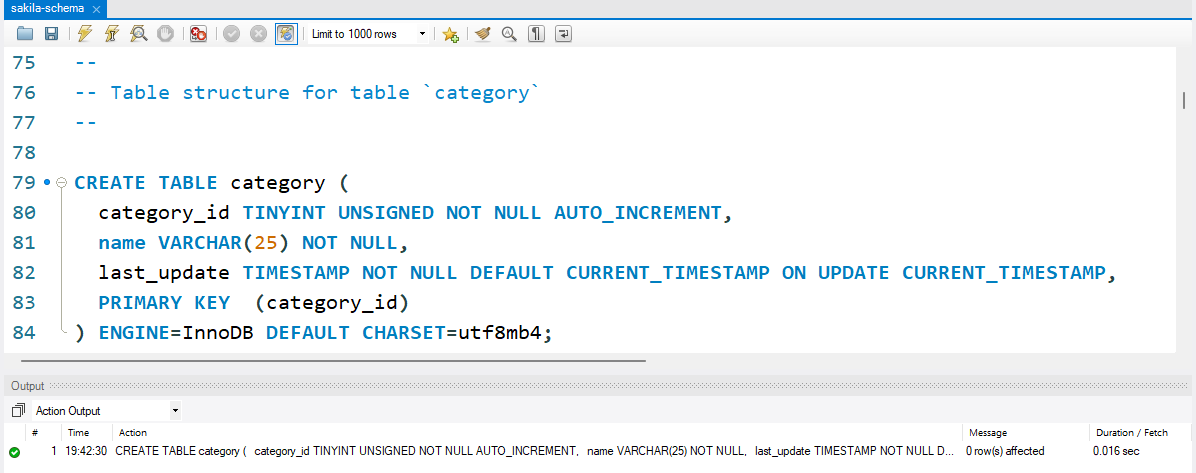
Creating table “actor”:



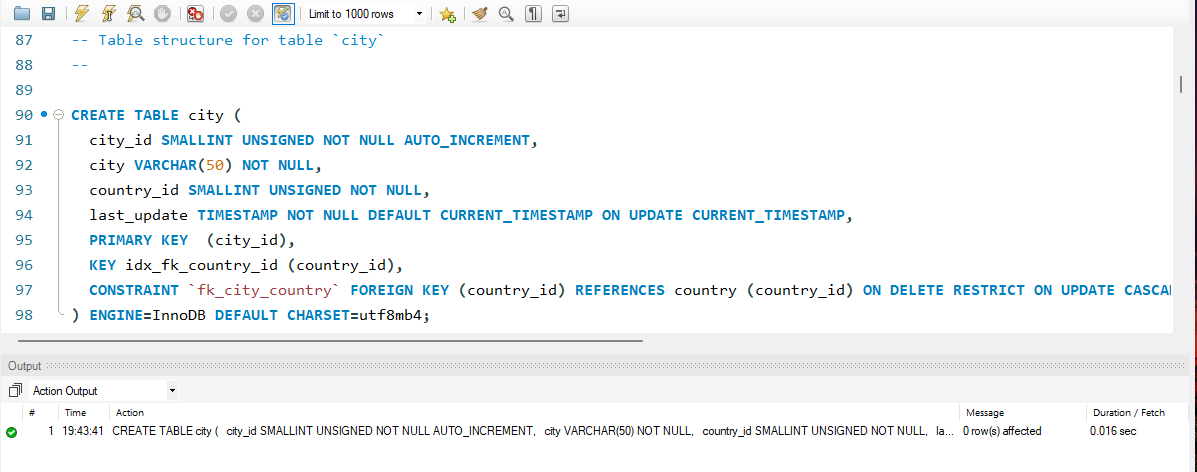
Creating table “address”:



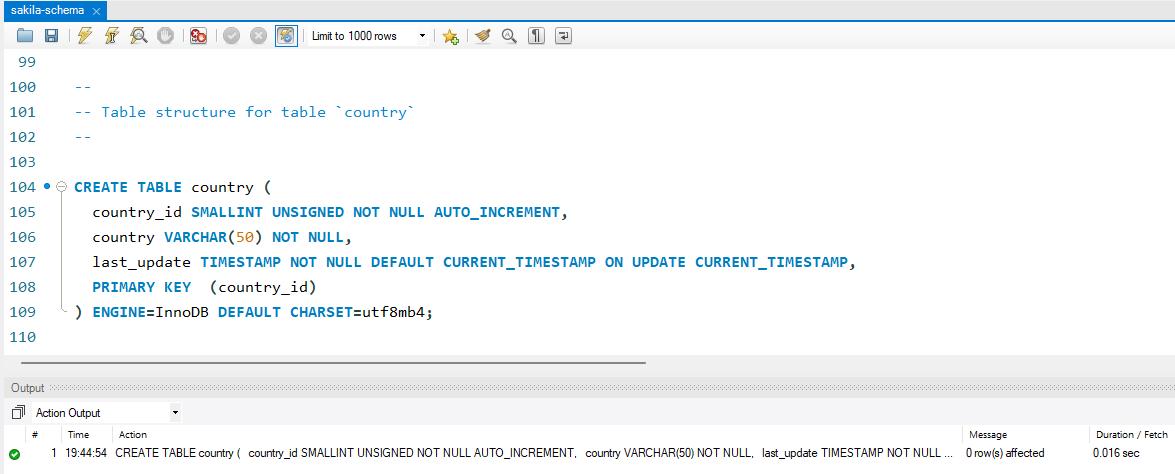
Creating table “category”:



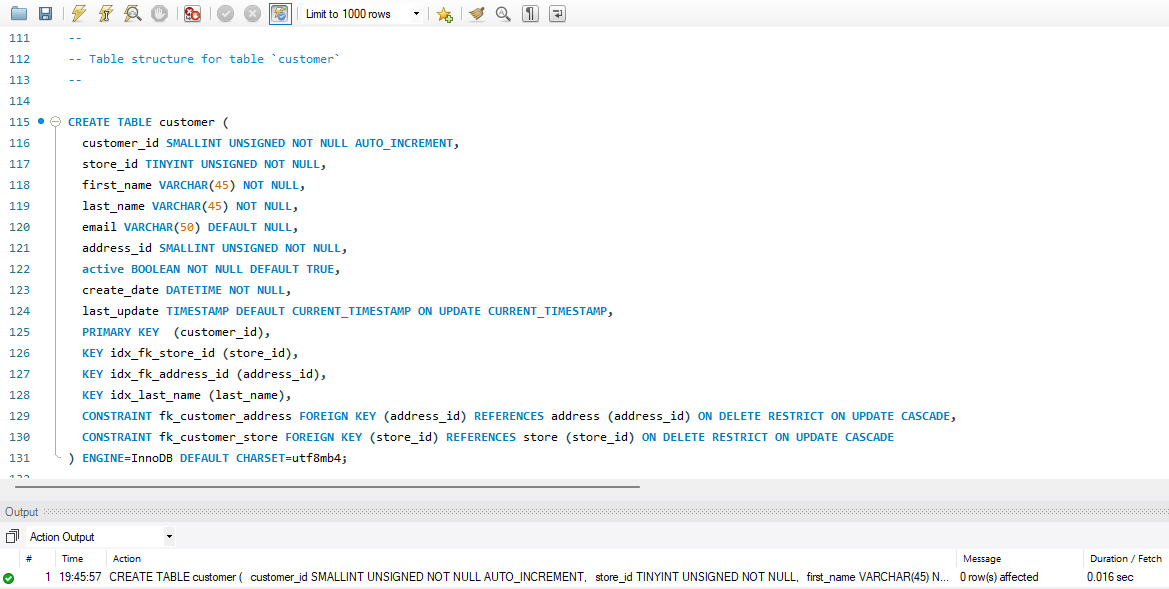
Creating table “city”:



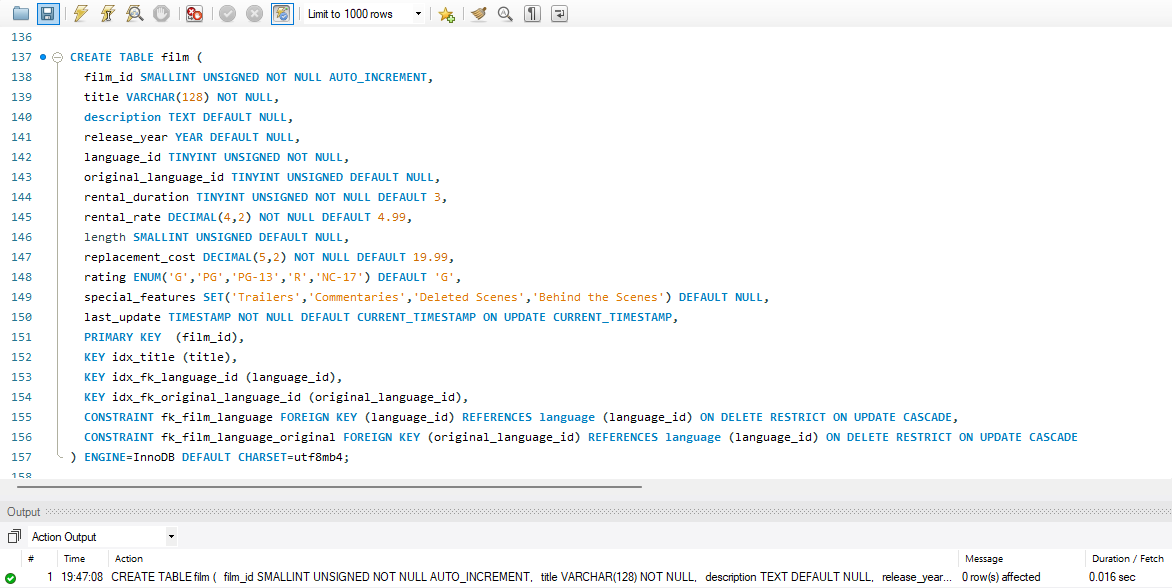
Creating table “country”:



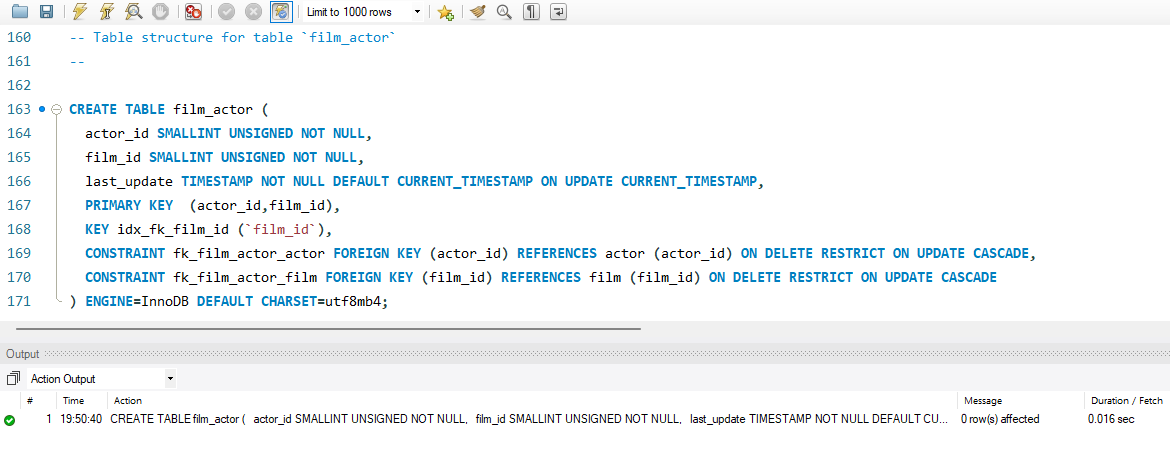
Creating table “customer”:



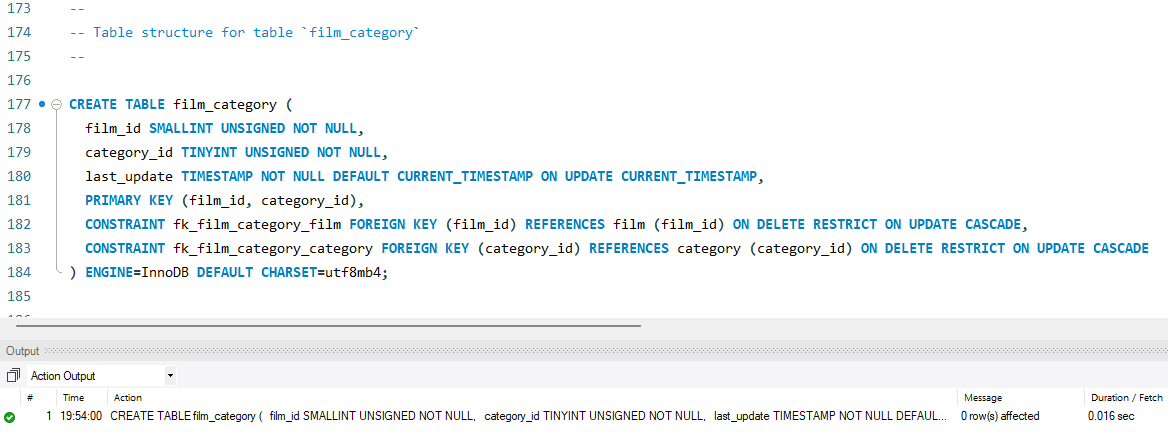
Creating table “film”:



Create table “film\_actor”:

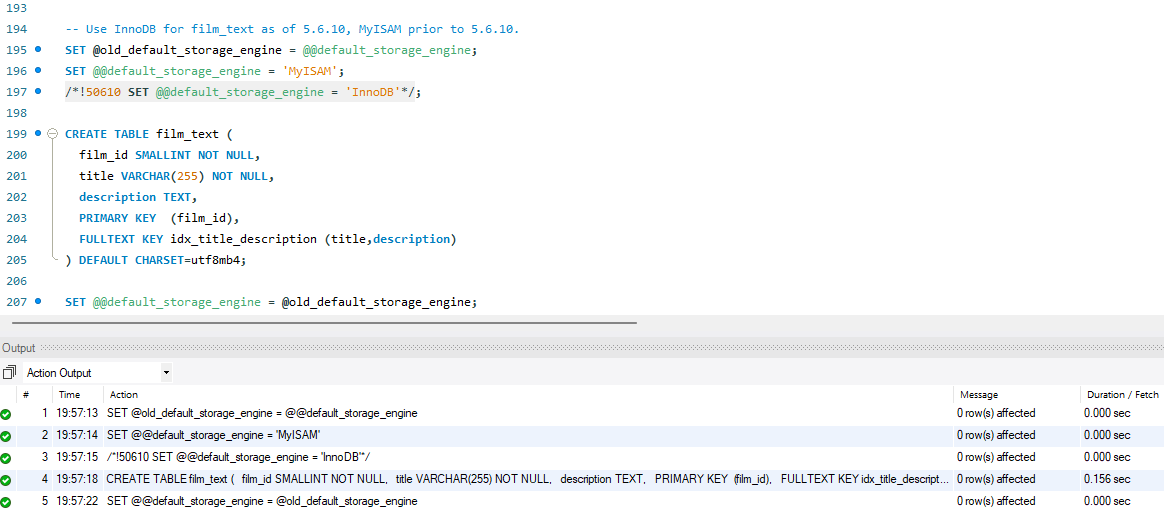


Create table “film\_category”:



Use InnoDB for film\_text as of 5.6.10, MyISAM prior to 5.6.10.

Create table “film\_text”:



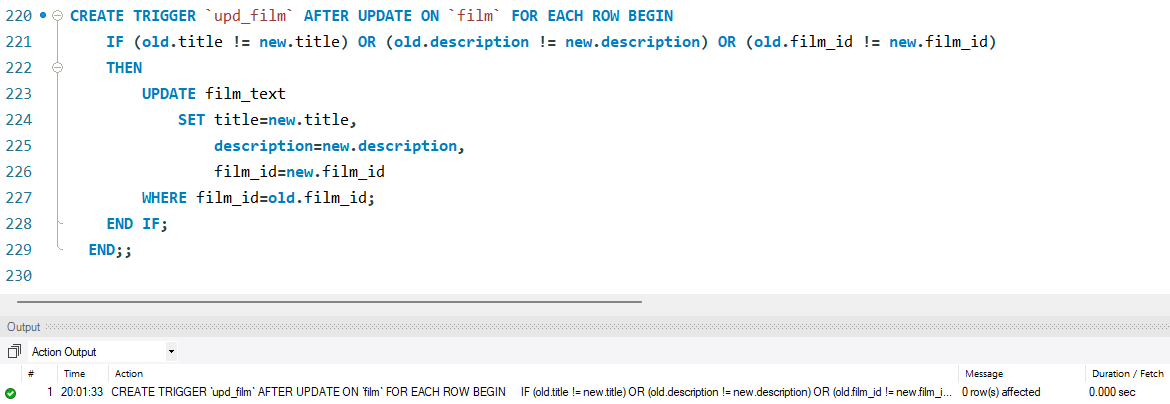


Triggers for loading film\_text from film:

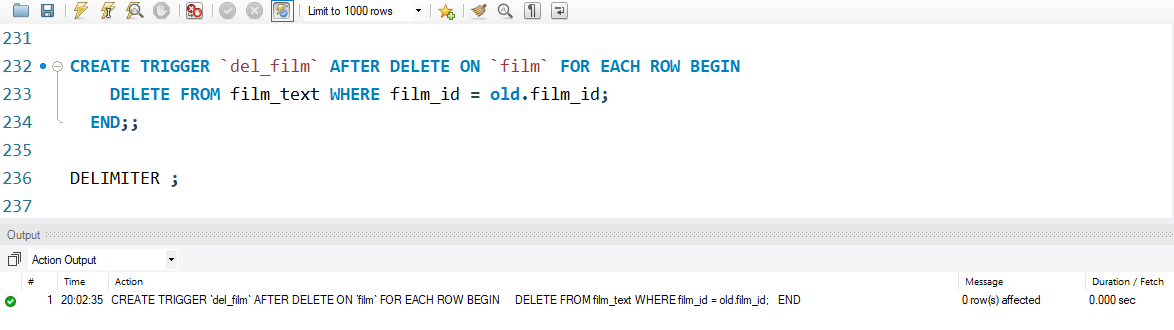
Creating trigger `ins\_film`:



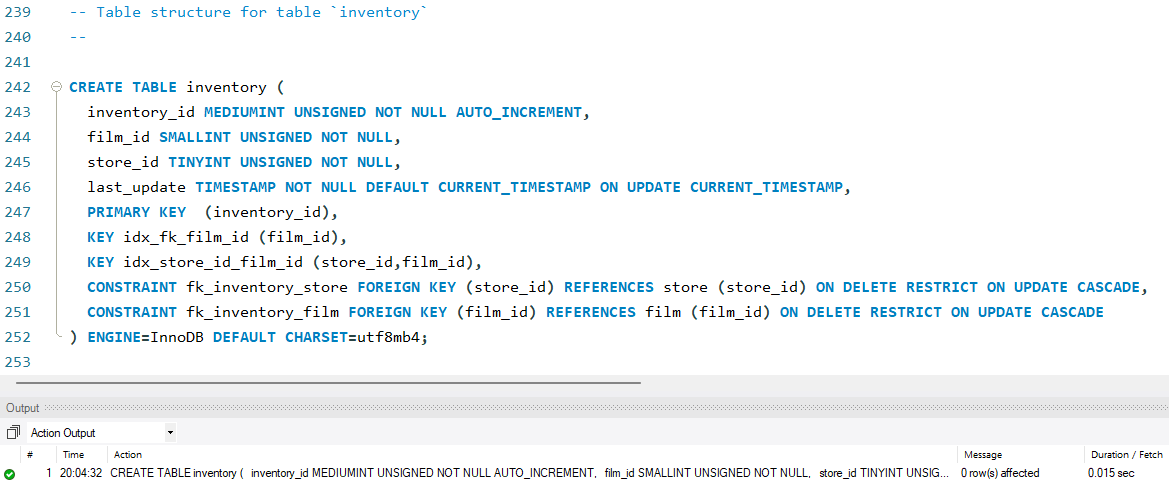
Creating trigger `upd\_film`:



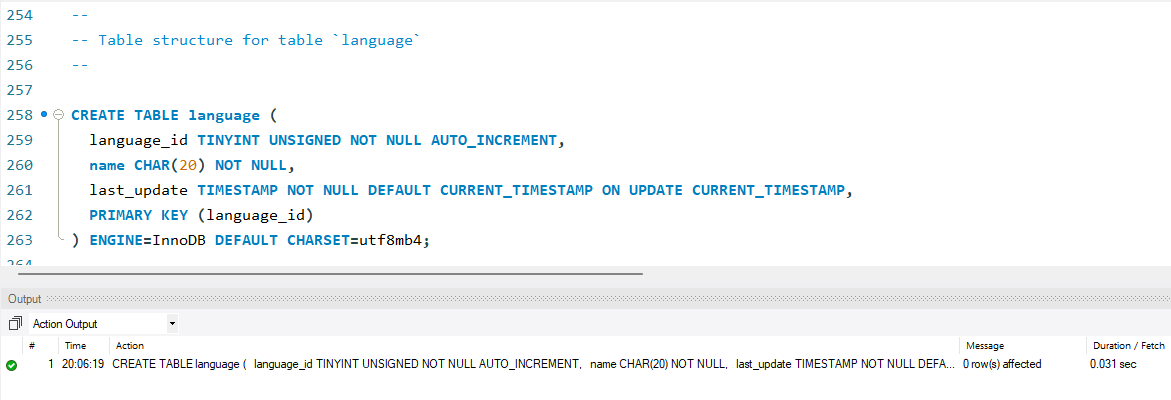
Creating trigger `del\_film`:



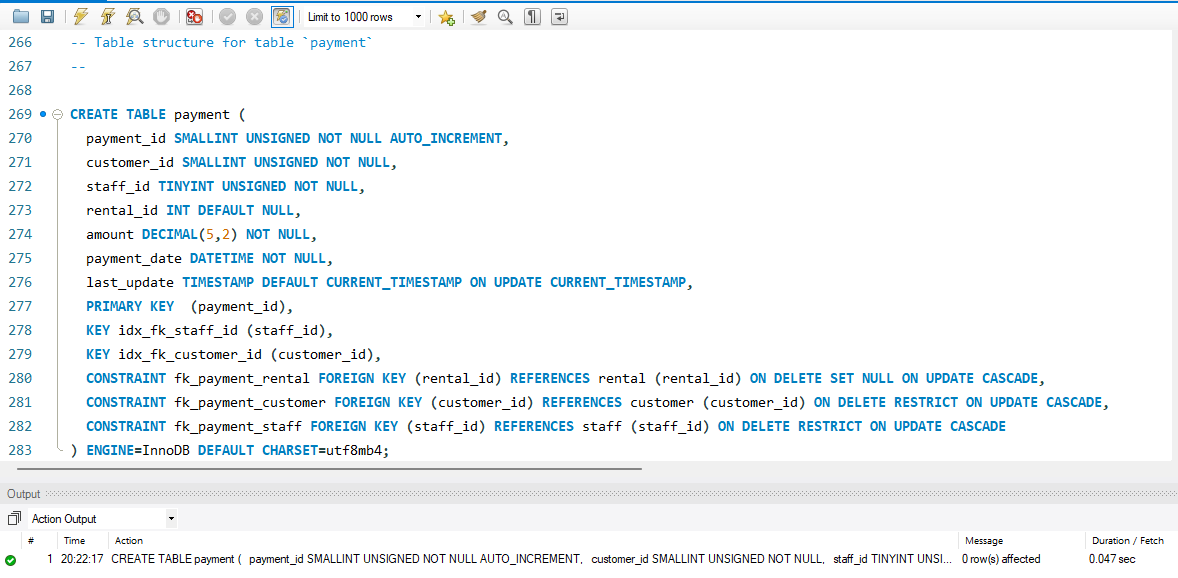
Creating table “inventory”:



Create table “language”:



Create table “payment”:



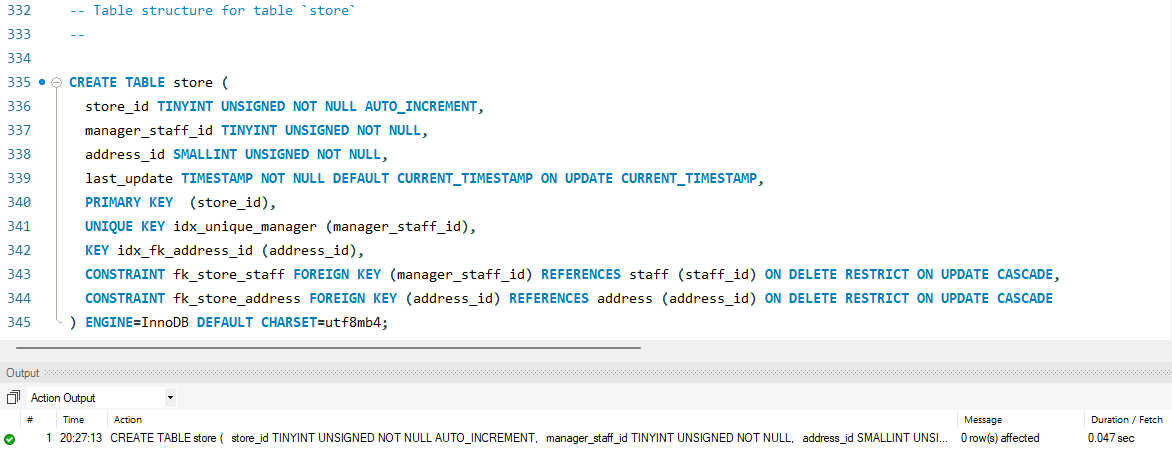
Create table “rental”:



Create table “staff”:



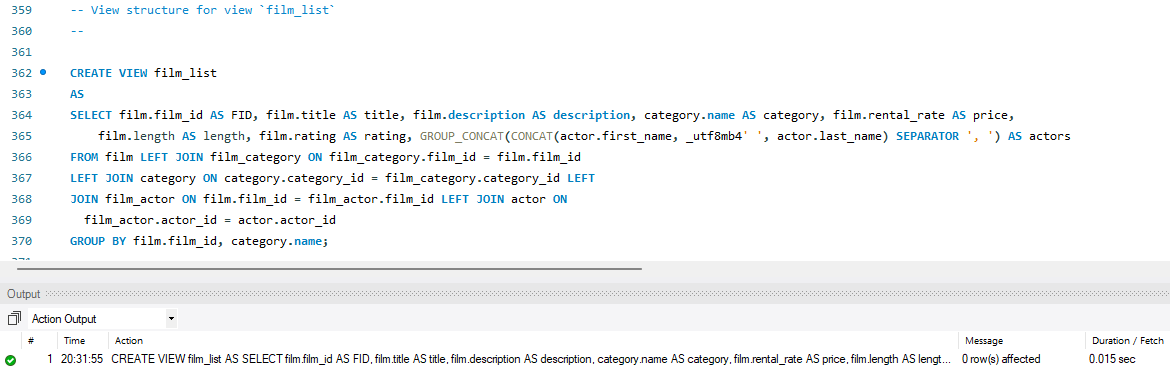
Create table “store”:



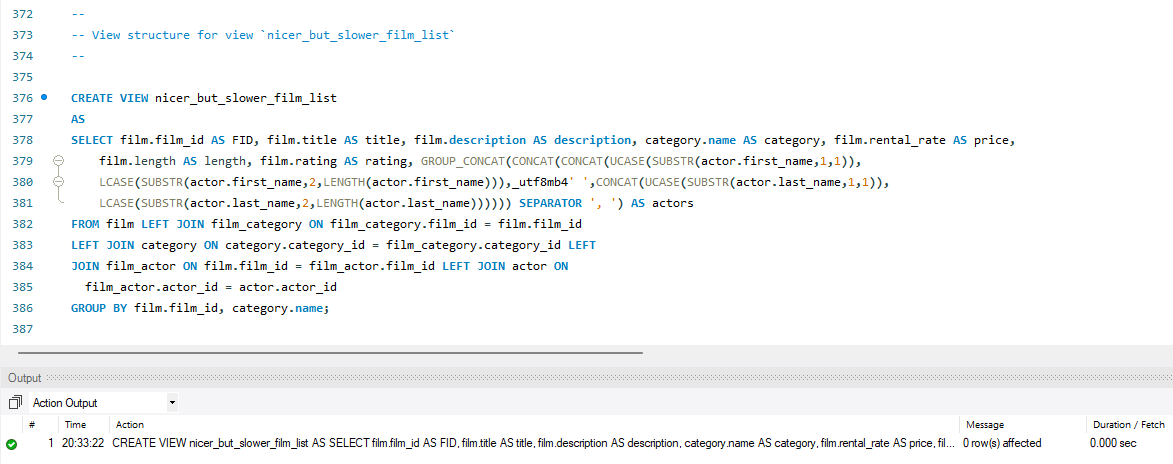
View structure for view “customer\_list”:



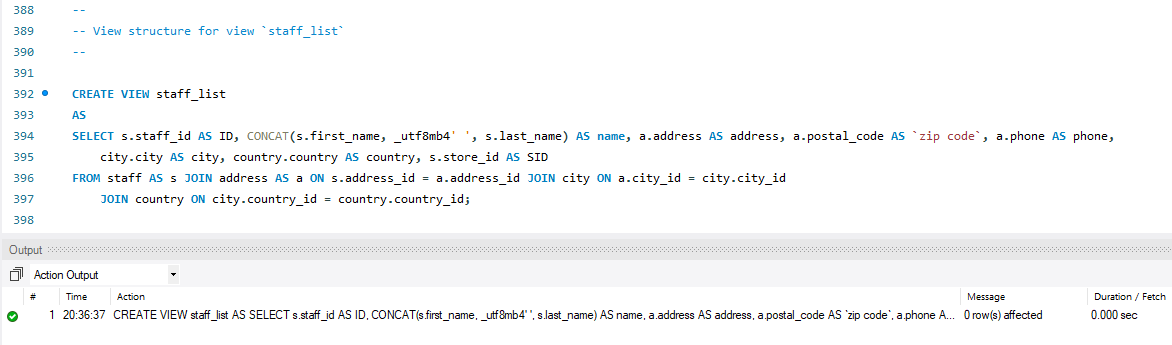
View structure for view “film\_list”:



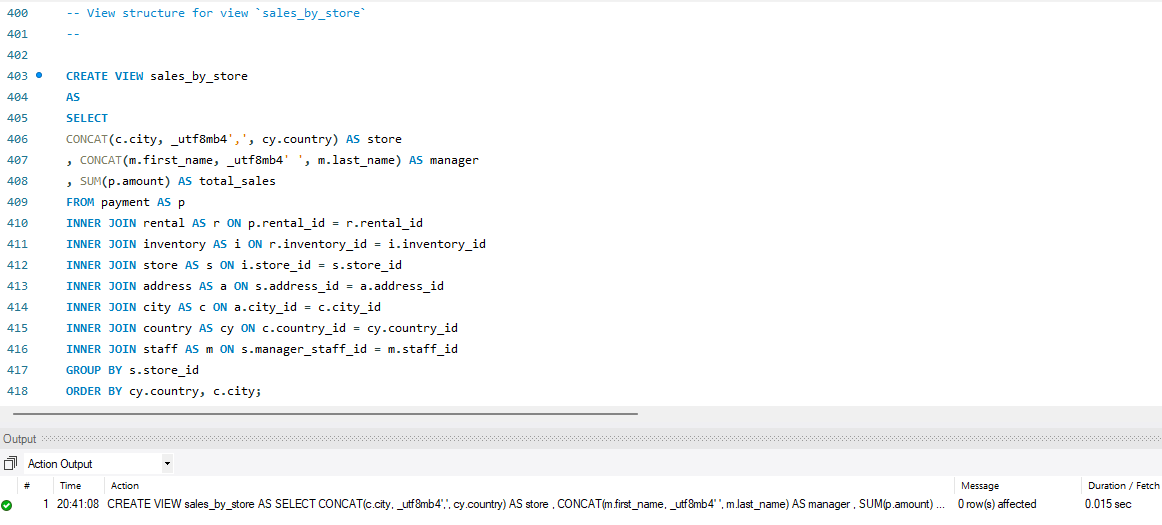
View structure for view “Nicer\_but\_slower\_film\_list”:



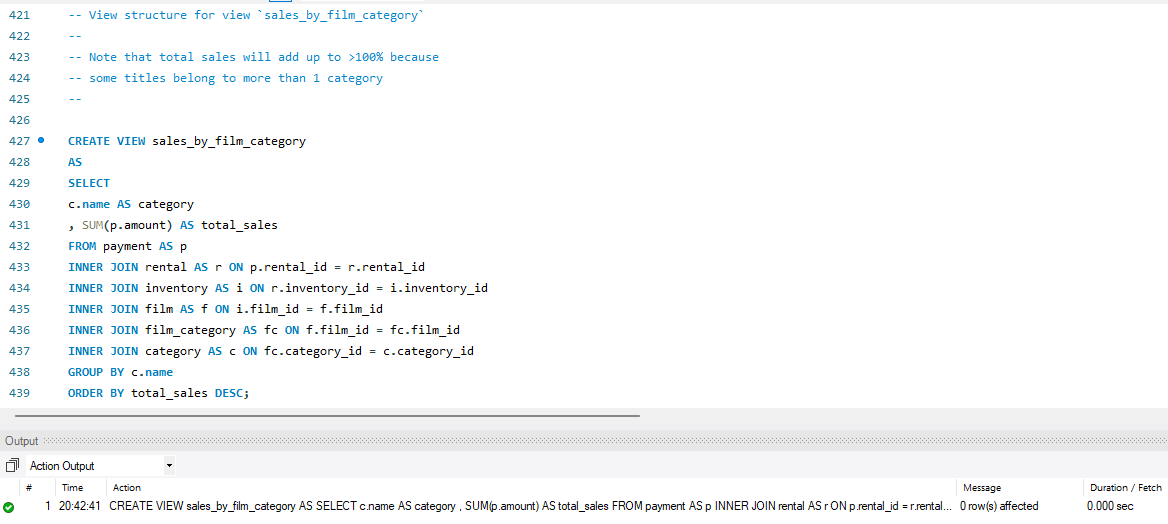
View structure for view “staff\_list”:



View structure for view “sales\_by\_store”:

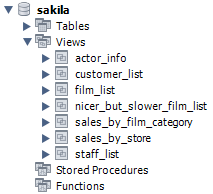


View structure for view “sales\_by\_film\_category”:



Creating view structure for view “actor\_info”:





Procedure structures:

--

-- Procedure structure for procedure `rewards\_report`

--

DELIMITER //

CREATE PROCEDURE rewards\_report (

IN min\_monthly\_purchases TINYINT UNSIGNED

, IN min\_dollar\_amount\_purchased DECIMAL(10,2)

, OUT count\_rewardees INT

)

LANGUAGE SQL

NOT DETERMINISTIC

READS SQL DATA

SQL SECURITY DEFINER

COMMENT 'Provides a customizable report on best customers'

proc: BEGIN

DECLARE last\_month\_start DATE;

DECLARE last\_month\_end DATE;

/\* Some sanity checks... \*/

IF min\_monthly\_purchases = 0 THEN

SELECT 'Minimum monthly purchases parameter must be > 0';

LEAVE proc;

END IF;

IF min\_dollar\_amount\_purchased = 0.00 THEN

SELECT 'Minimum monthly dollar amount purchased parameter must be > $0.00';

LEAVE proc;

END IF;

/\* Determine start and end time periods \*/

SET last\_month\_start = DATE\_SUB(CURRENT\_DATE(), INTERVAL 1 MONTH);

SET last\_month\_start = STR\_TO\_DATE(CONCAT(YEAR(last\_month\_start),'-',MONTH(last\_month\_start),'-01'),'%Y-%m-%d');

SET last\_month\_end = LAST\_DAY(last\_month\_start);

/\*

Create a temporary storage area for

Customer IDs.

\*/

CREATE TEMPORARY TABLE tmpCustomer (customer\_id SMALLINT UNSIGNED NOT NULL PRIMARY KEY);

/\*

Find all customers meeting the

monthly purchase requirements

\*/

INSERT INTO tmpCustomer (customer\_id)

SELECT p.customer\_id

FROM payment AS p

WHERE DATE(p.payment\_date) BETWEEN last\_month\_start AND last\_month\_end

GROUP BY customer\_id

HAVING SUM(p.amount) > min\_dollar\_amount\_purchased

AND COUNT(customer\_id) > min\_monthly\_purchases;

/\* Populate OUT parameter with count of found customers \*/

SELECT COUNT(\*) FROM tmpCustomer INTO count\_rewardees;

/\*

Output ALL customer information of matching rewardees.

Customize output as needed.

\*/

SELECT c.\*

FROM tmpCustomer AS t

INNER JOIN customer AS c ON t.customer\_id = c.customer\_id;

/\* Clean up \*/

DROP TABLE tmpCustomer;

END //

DELIMITER ;

DELIMITER $$

CREATE FUNCTION get\_customer\_balance(p\_customer\_id INT, p\_effective\_date DATETIME) RETURNS DECIMAL(5,2)

DETERMINISTIC

READS SQL DATA

BEGIN

#OK, WE NEED TO CALCULATE THE CURRENT BALANCE GIVEN A CUSTOMER\_ID AND A DATE

#THAT WE WANT THE BALANCE TO BE EFFECTIVE FOR. THE BALANCE IS:

# 1) RENTAL FEES FOR ALL PREVIOUS RENTALS

# 2) ONE DOLLAR FOR EVERY DAY THE PREVIOUS RENTALS ARE OVERDUE

# 3) IF A FILM IS MORE THAN RENTAL\_DURATION \* 2 OVERDUE, CHARGE THE REPLACEMENT\_COST

# 4) SUBTRACT ALL PAYMENTS MADE BEFORE THE DATE SPECIFIED

DECLARE v\_rentfees DECIMAL(5,2); #FEES PAID TO RENT THE VIDEOS INITIALLY

DECLARE v\_overfees INTEGER; #LATE FEES FOR PRIOR RENTALS

DECLARE v\_payments DECIMAL(5,2); #SUM OF PAYMENTS MADE PREVIOUSLY

SELECT IFNULL(SUM(film.rental\_rate),0) INTO v\_rentfees

FROM film, inventory, rental

WHERE film.film\_id = inventory.film\_id

AND inventory.inventory\_id = rental.inventory\_id

AND rental.rental\_date <= p\_effective\_date

AND rental.customer\_id = p\_customer\_id;

SELECT IFNULL(SUM(IF((TO\_DAYS(rental.return\_date) - TO\_DAYS(rental.rental\_date)) > film.rental\_duration,

((TO\_DAYS(rental.return\_date) - TO\_DAYS(rental.rental\_date)) - film.rental\_duration),0)),0) INTO v\_overfees

FROM rental, inventory, film

WHERE film.film\_id = inventory.film\_id

AND inventory.inventory\_id = rental.inventory\_id

AND rental.rental\_date <= p\_effective\_date

AND rental.customer\_id = p\_customer\_id;

SELECT IFNULL(SUM(payment.amount),0) INTO v\_payments

FROM payment

WHERE payment.payment\_date <= p\_effective\_date

AND payment.customer\_id = p\_customer\_id;

RETURN v\_rentfees + v\_overfees - v\_payments;

END $$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE film\_in\_stock(IN p\_film\_id INT, IN p\_store\_id INT, OUT p\_film\_count INT)

READS SQL DATA

BEGIN

SELECT inventory\_id

FROM inventory

WHERE film\_id = p\_film\_id

AND store\_id = p\_store\_id

AND inventory\_in\_stock(inventory\_id);

SELECT COUNT(\*)

FROM inventory

WHERE film\_id = p\_film\_id

AND store\_id = p\_store\_id

AND inventory\_in\_stock(inventory\_id)

INTO p\_film\_count;

END $$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE film\_not\_in\_stock(IN p\_film\_id INT, IN p\_store\_id INT, OUT p\_film\_count INT)

READS SQL DATA

BEGIN

SELECT inventory\_id

FROM inventory

WHERE film\_id = p\_film\_id

AND store\_id = p\_store\_id

AND NOT inventory\_in\_stock(inventory\_id);

SELECT COUNT(\*)

FROM inventory

WHERE film\_id = p\_film\_id

AND store\_id = p\_store\_id

AND NOT inventory\_in\_stock(inventory\_id)

INTO p\_film\_count;

END $$

DELIMITER ;

DELIMITER $$

CREATE FUNCTION inventory\_held\_by\_customer(p\_inventory\_id INT) RETURNS INT

READS SQL DATA

BEGIN

DECLARE v\_customer\_id INT;

DECLARE EXIT HANDLER FOR NOT FOUND RETURN NULL;

SELECT customer\_id INTO v\_customer\_id

FROM rental

WHERE return\_date IS NULL

AND inventory\_id = p\_inventory\_id;

RETURN v\_customer\_id;

END $$

DELIMITER ;

DELIMITER $$

CREATE FUNCTION inventory\_in\_stock(p\_inventory\_id INT) RETURNS BOOLEAN

READS SQL DATA

BEGIN

DECLARE v\_rentals INT;

DECLARE v\_out INT;

#AN ITEM IS IN-STOCK IF THERE ARE EITHER NO ROWS IN THE rental TABLE

#FOR THE ITEM OR ALL ROWS HAVE return\_date POPULATED

SELECT COUNT(\*) INTO v\_rentals

FROM rental

WHERE inventory\_id = p\_inventory\_id;

IF v\_rentals = 0 THEN

RETURN TRUE;

END IF;

SELECT COUNT(rental\_id) INTO v\_out

FROM inventory LEFT JOIN rental USING(inventory\_id)

WHERE inventory.inventory\_id = p\_inventory\_id

AND rental.return\_date IS NULL;

IF v\_out > 0 THEN

RETURN FALSE;

ELSE

RETURN TRUE;

END IF;

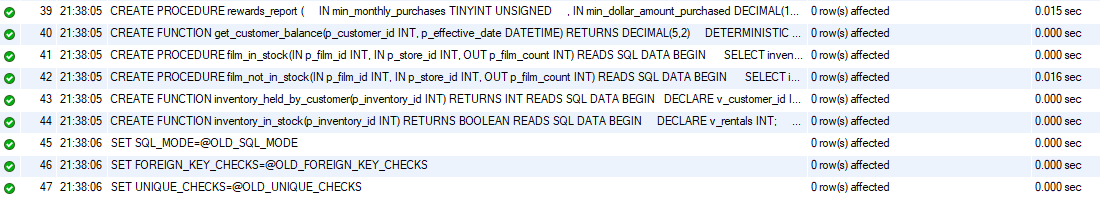
END $$

DELIMITER ;

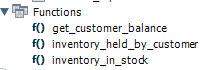
SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

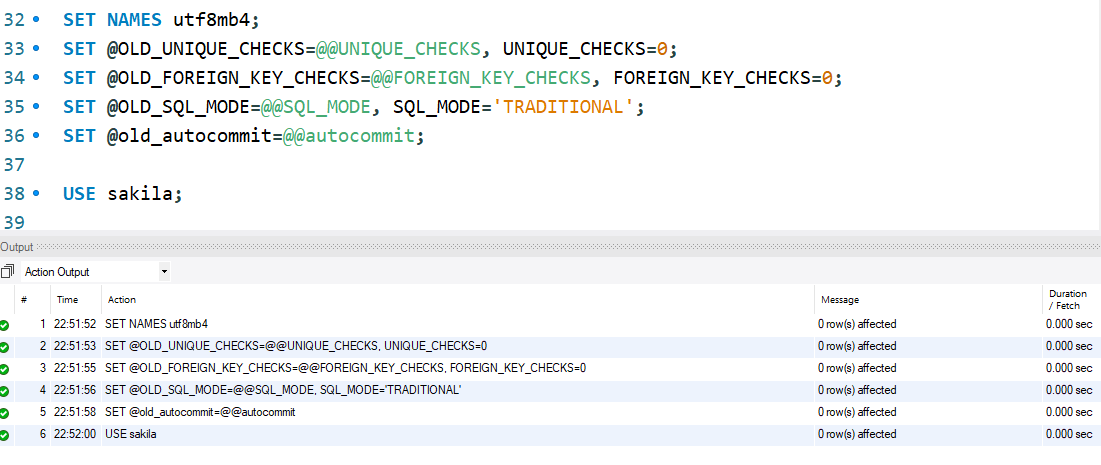
SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

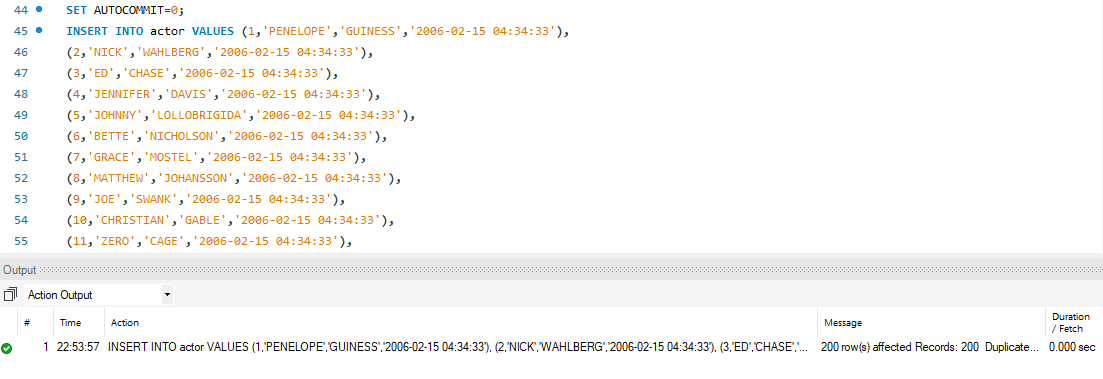




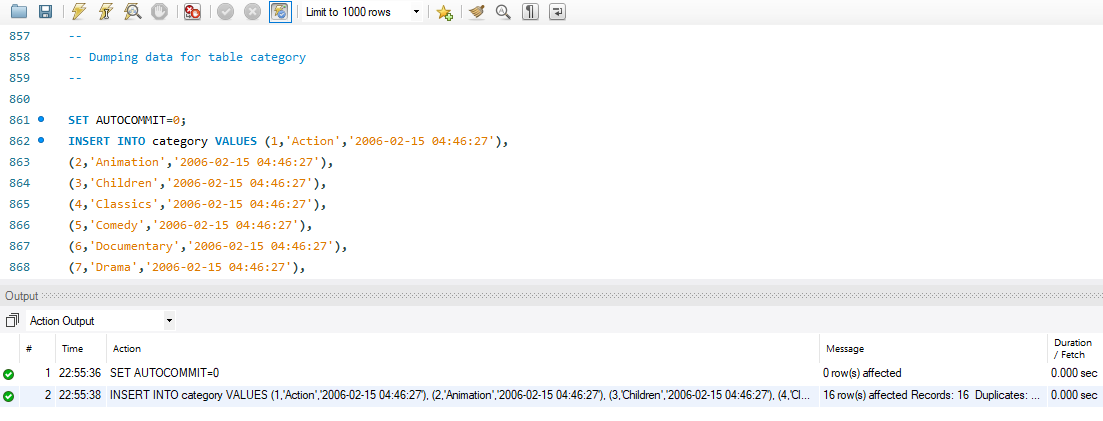


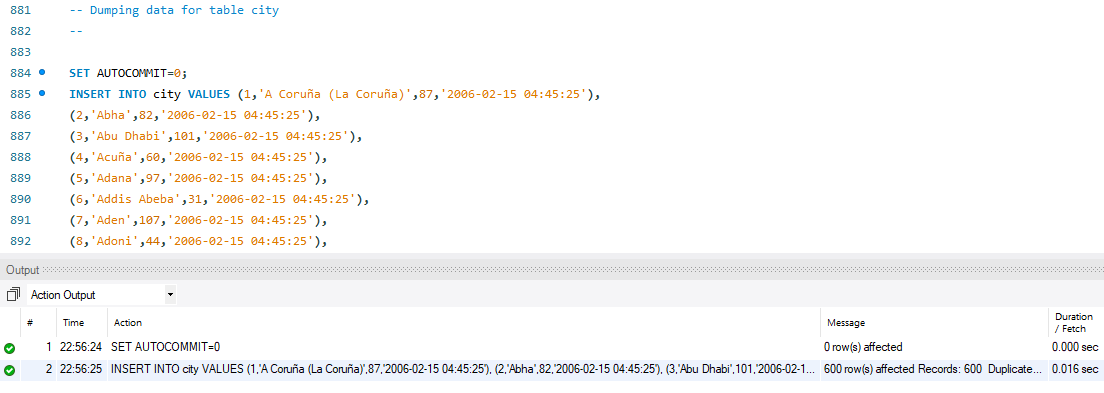
Populating the tables:

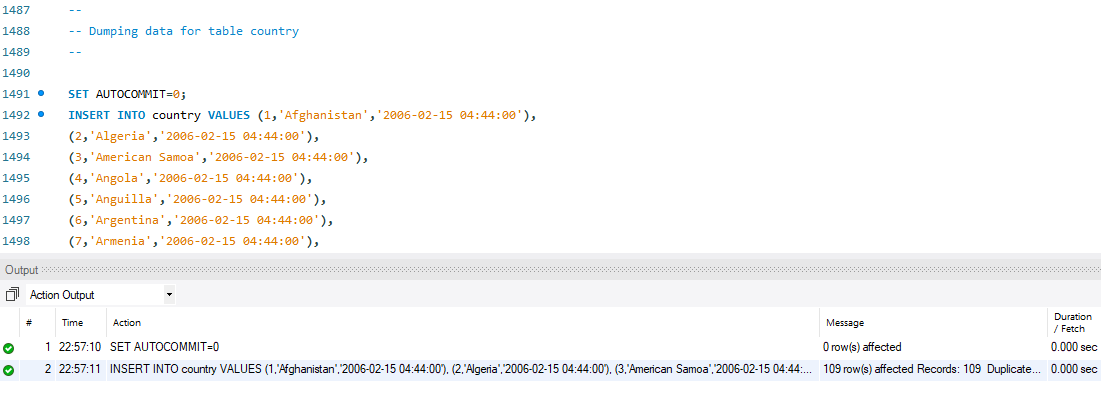




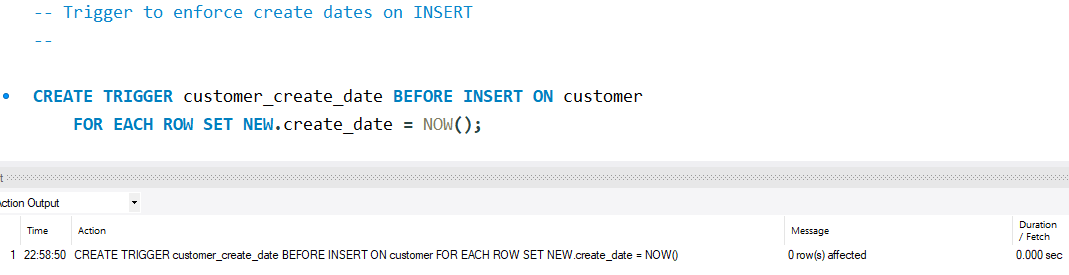


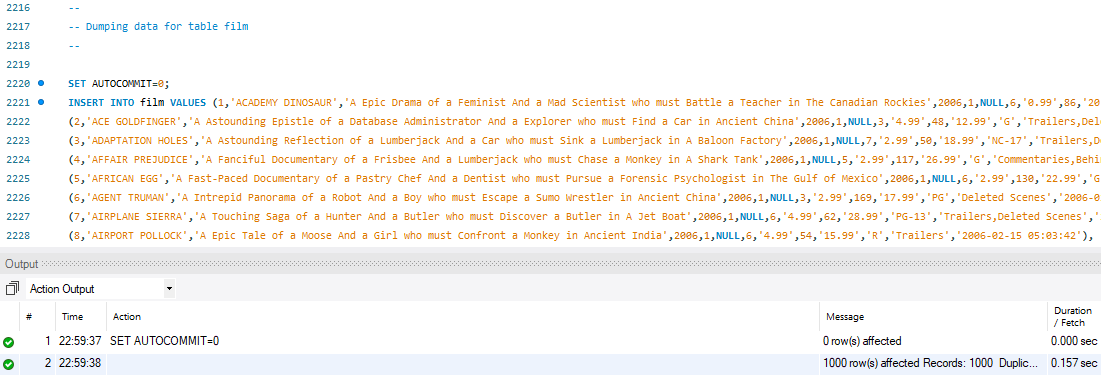


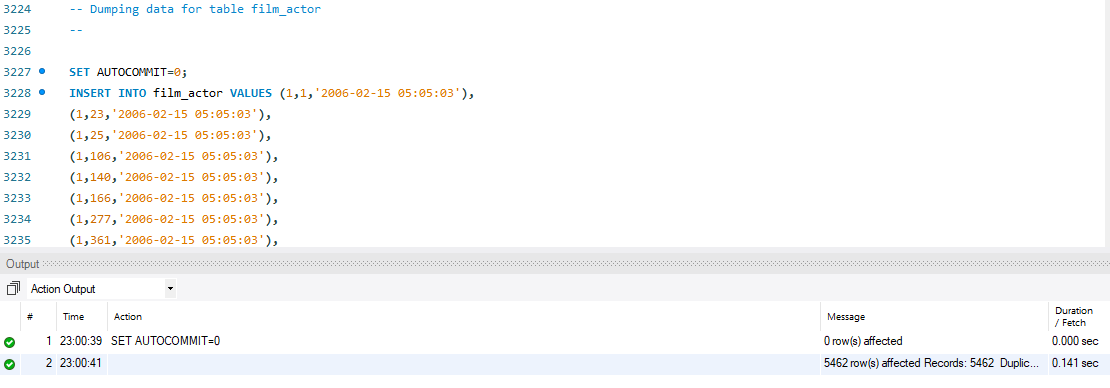




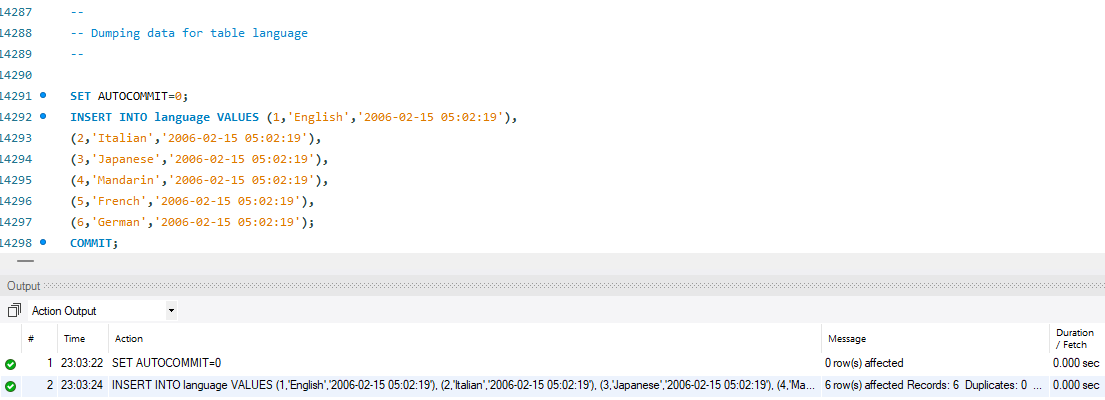


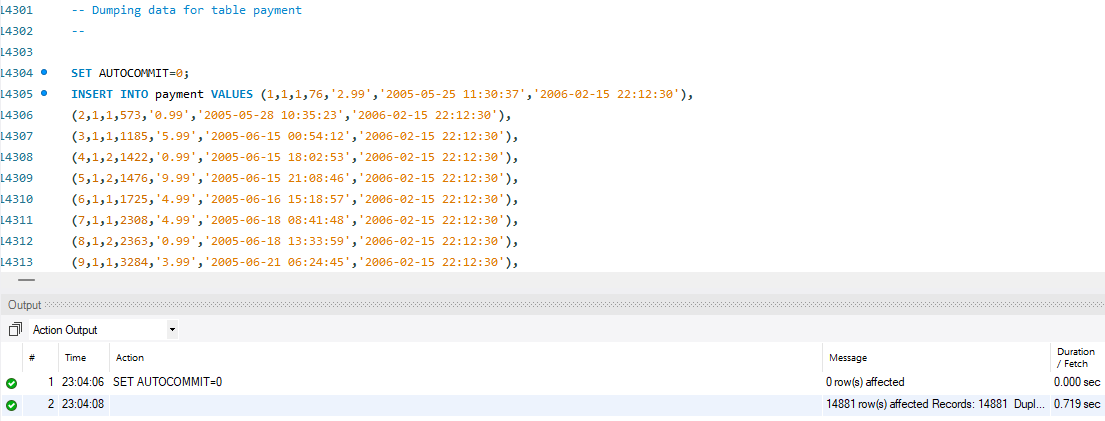


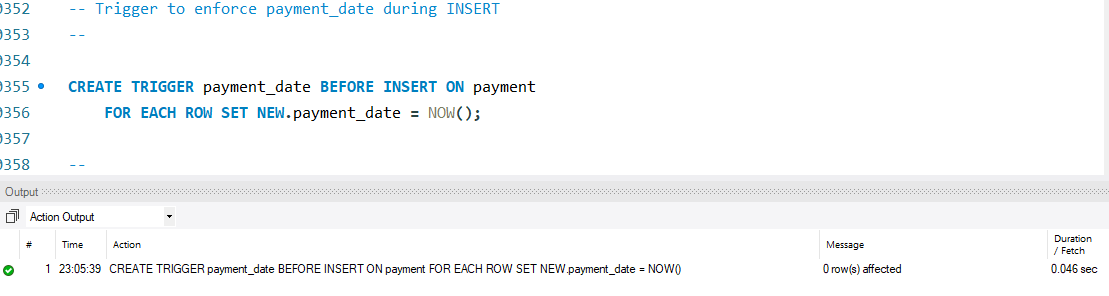


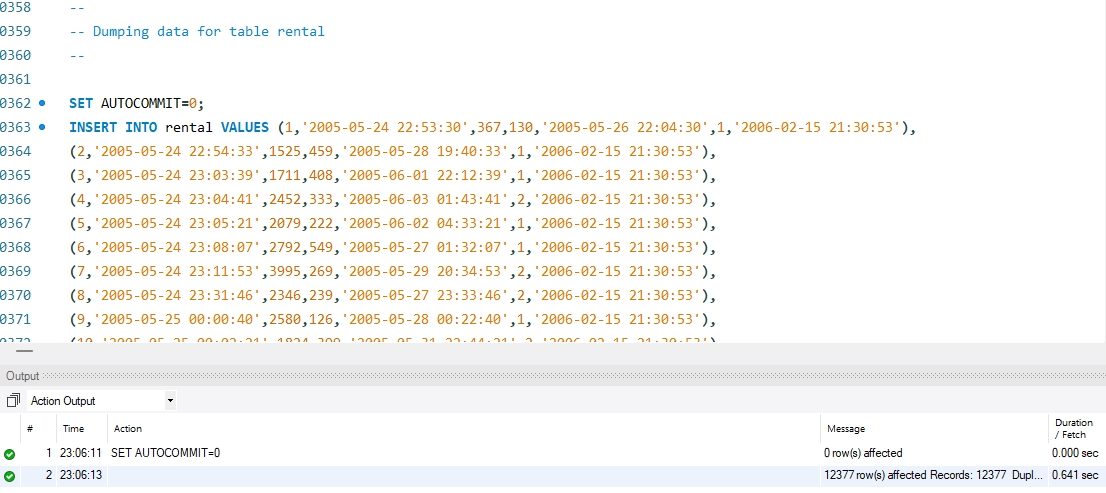


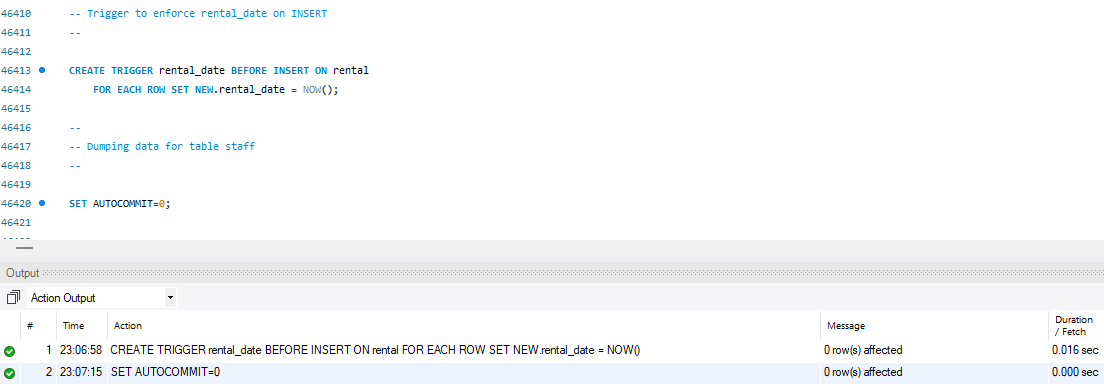




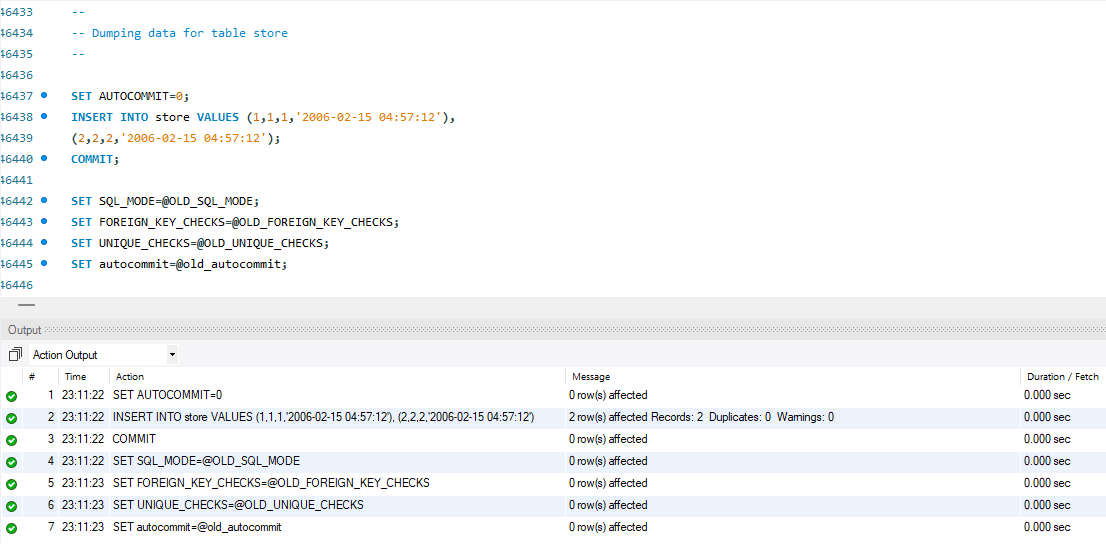






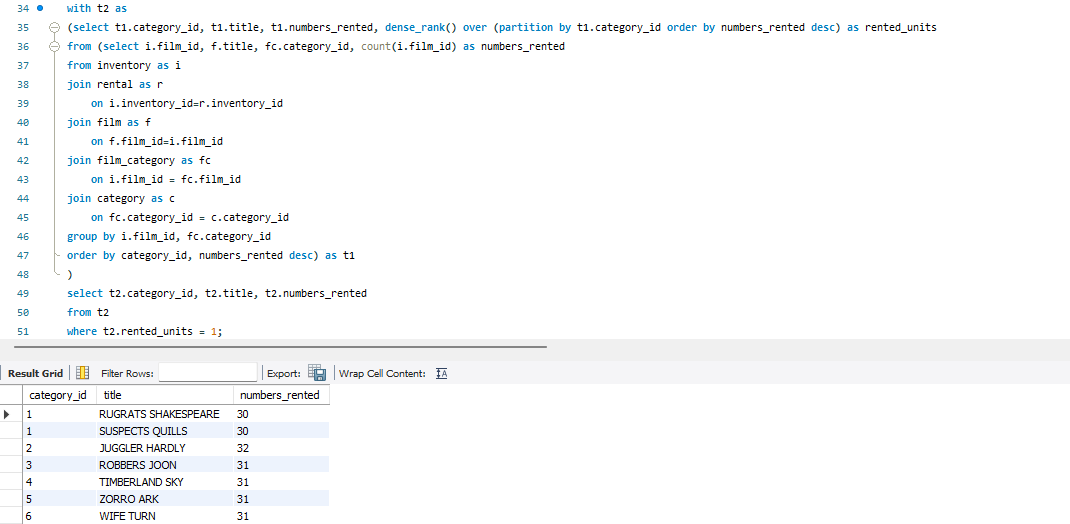




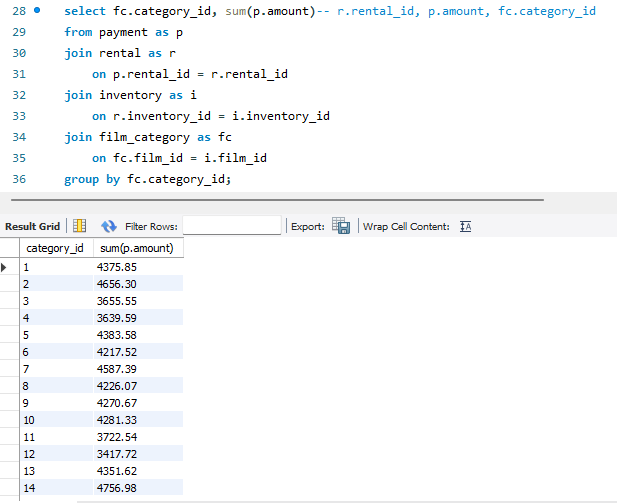


Solutions:

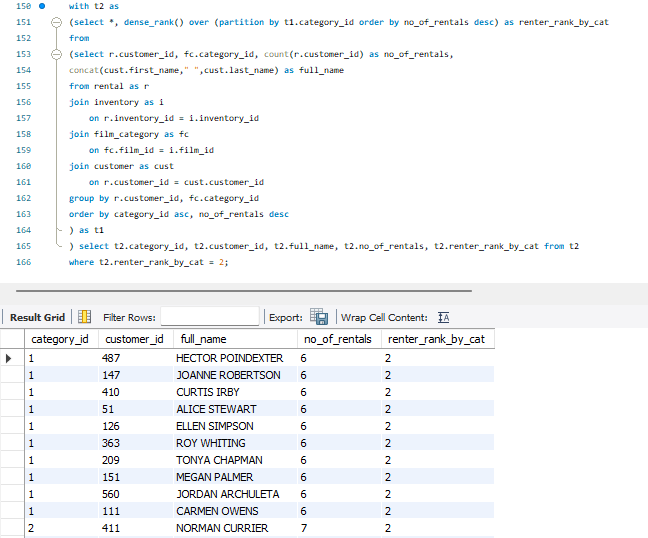
1. What is the most watched film (number of rentals) by each category?



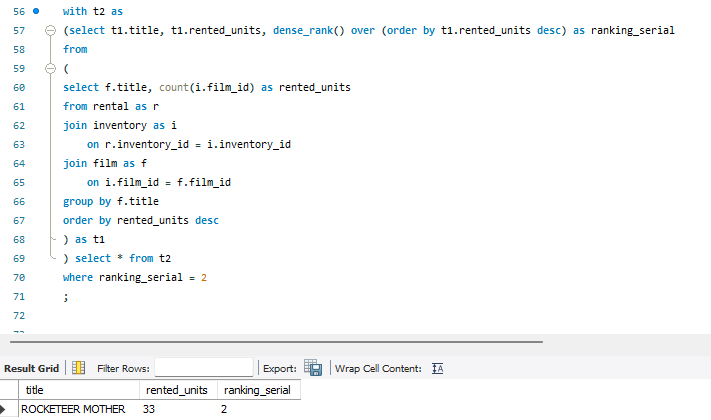
1. What is the total rental amount per category?



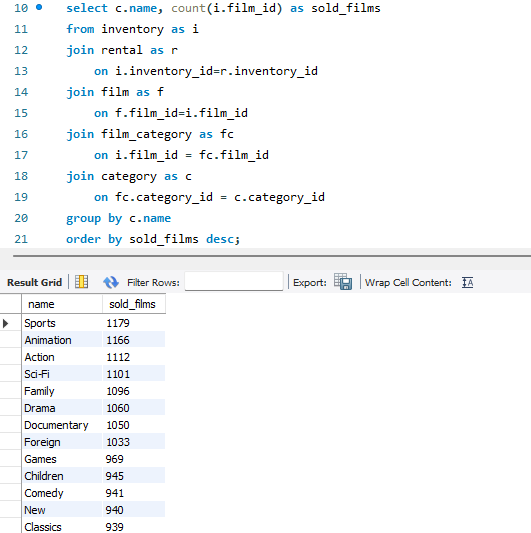
1. For each category, list the customer who is the second most renter.



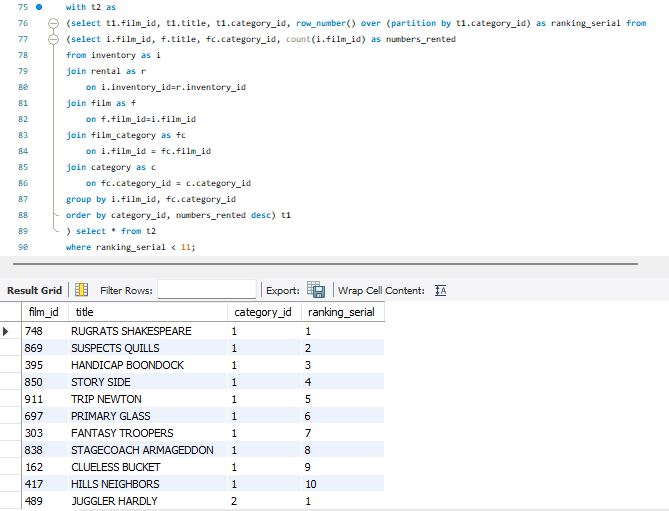
1. What is the second most popular movie title?



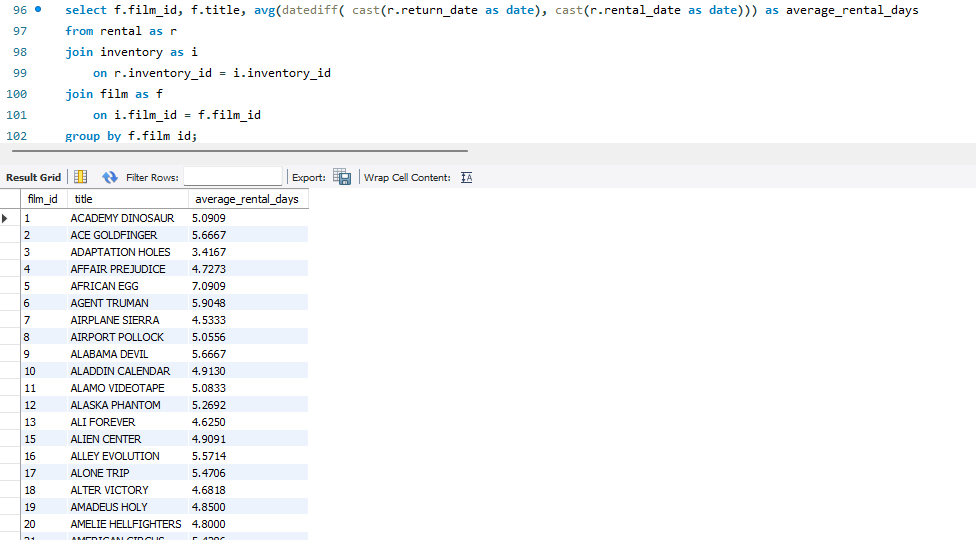
1. What are the most popular categories?



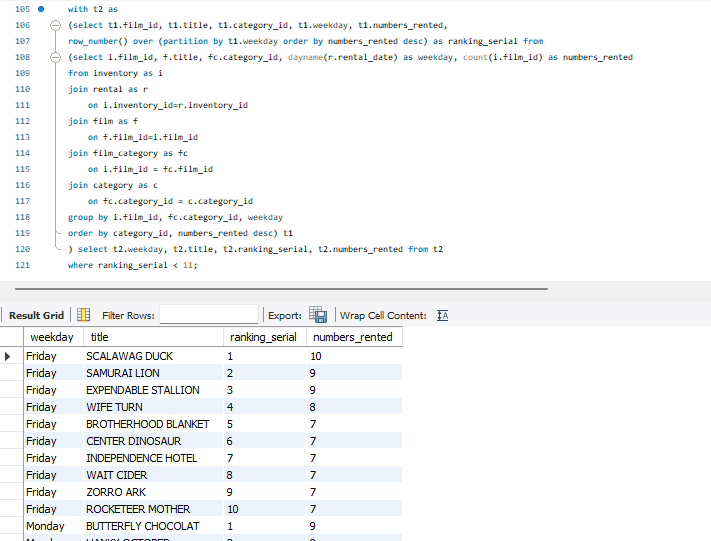
1. What are the top 10 most rented movies by category?



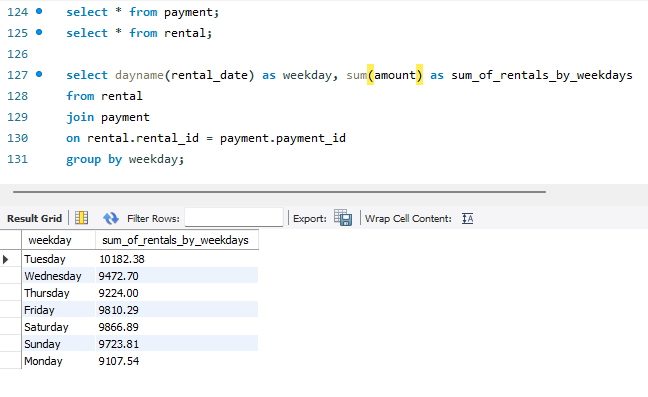
1. What are the average rental days for each movie?



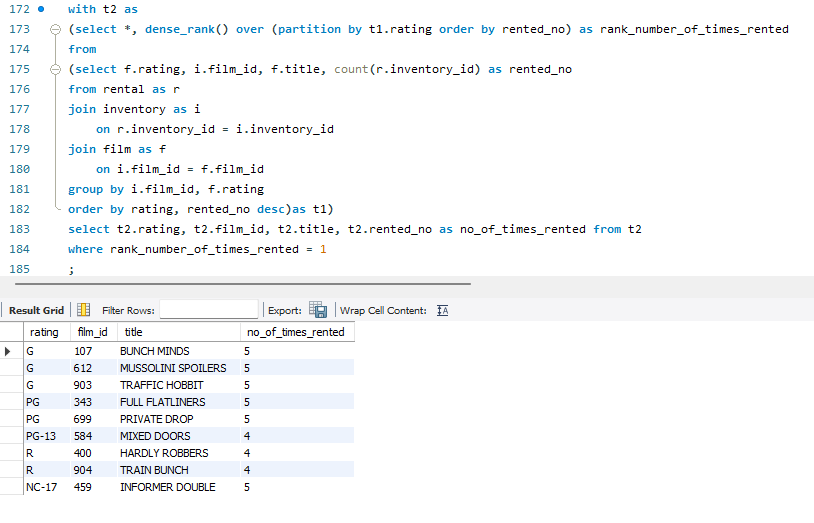
1. What are the top 10 most rented movies by day of the week?



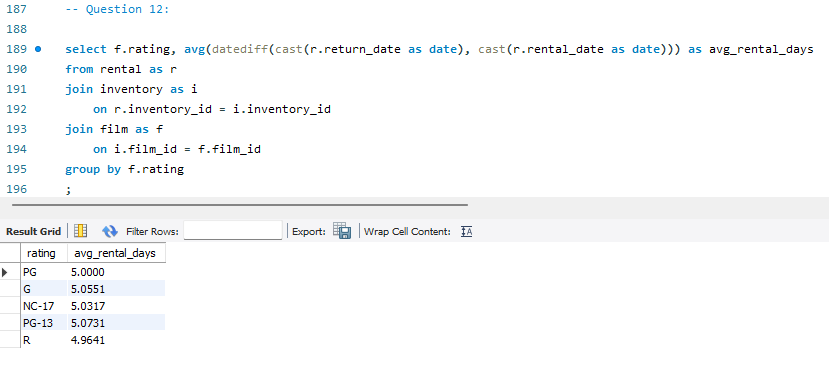
1. What is the total rental revenue (total amount of all rentals) by day of the week?



1. What are the most popular movies for each reward tier (rating)?

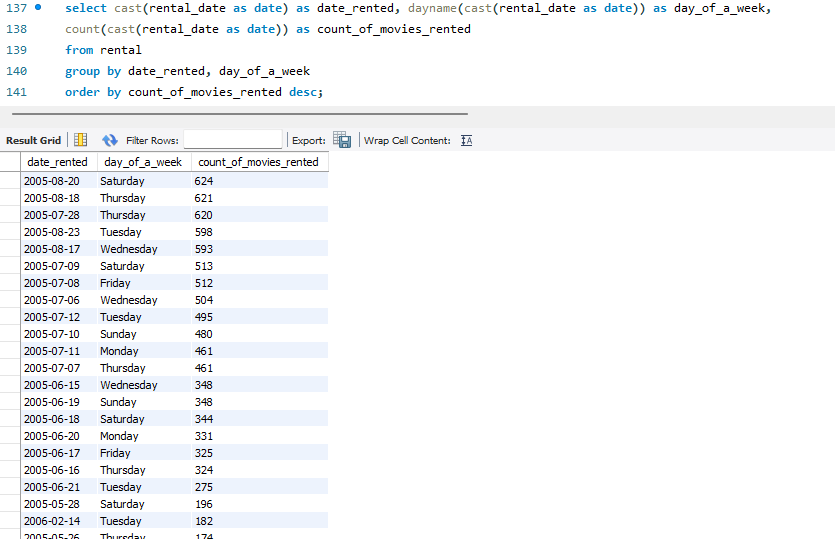


1. What are the average rental days by reward tier (rating)?

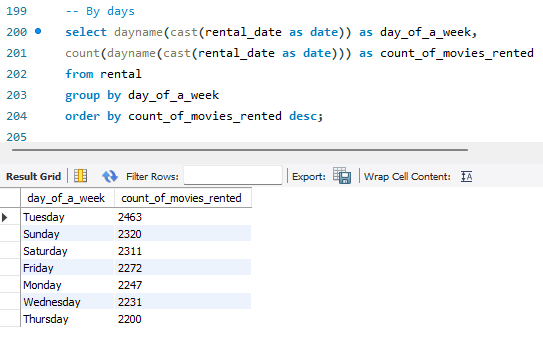


1. What are the most popular days to rent movies?

By date

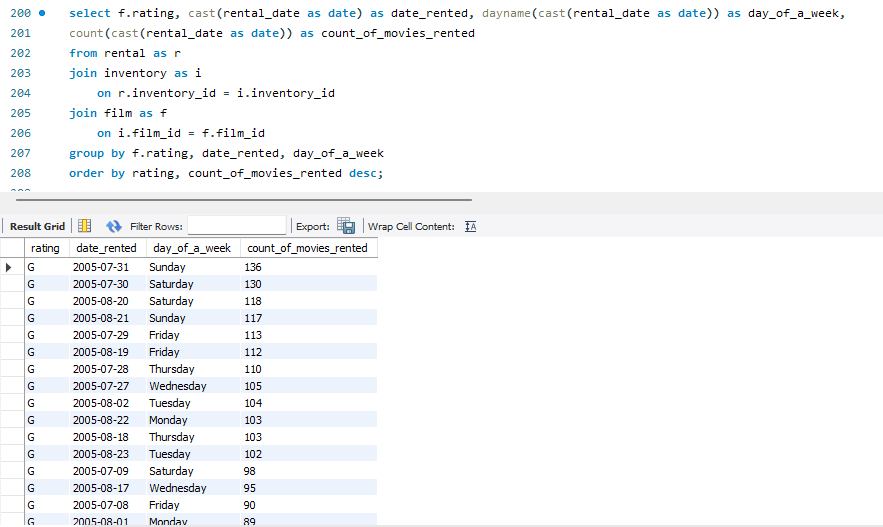


By weekday



1. What are the most popular days to rent movies by reward tier (rating)?

By date



By weekday

