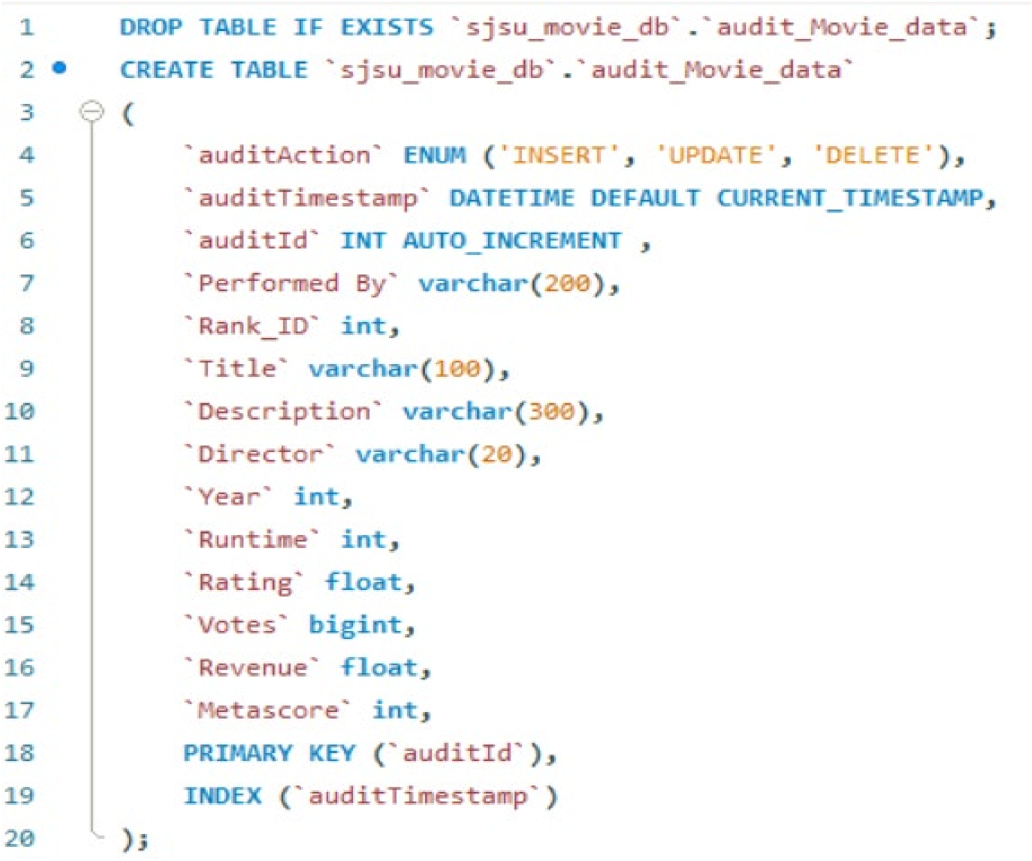
AUDIT TABLES, TRIGGERS , PROCEDURES – QUERIES & SCREENSHOTS

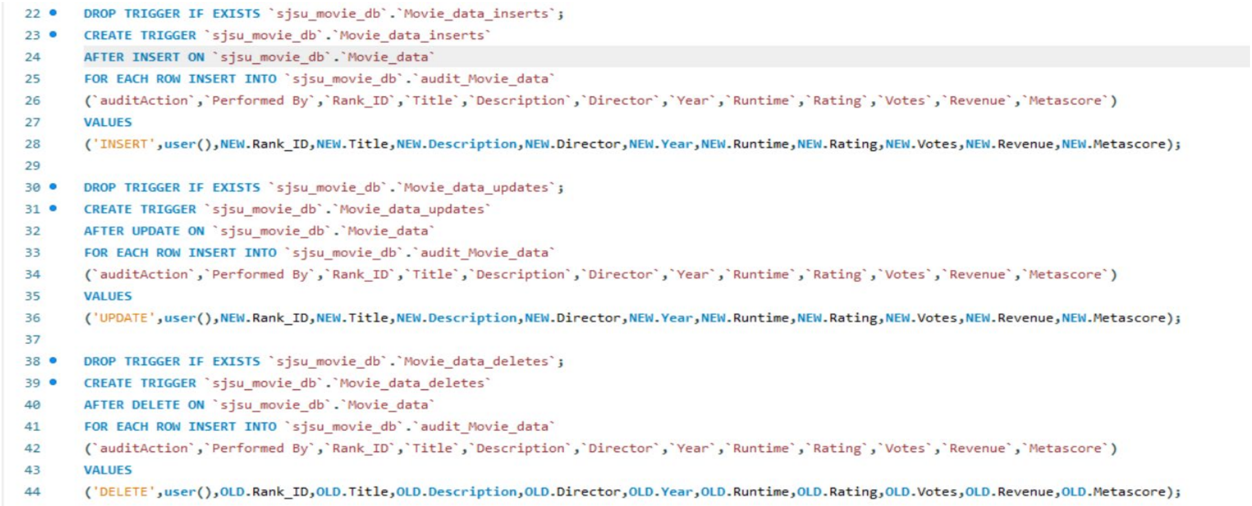
AUDIT TABLES WITH RESPECTIVE TRIGGERS

AUDIT MOVIE DATA:

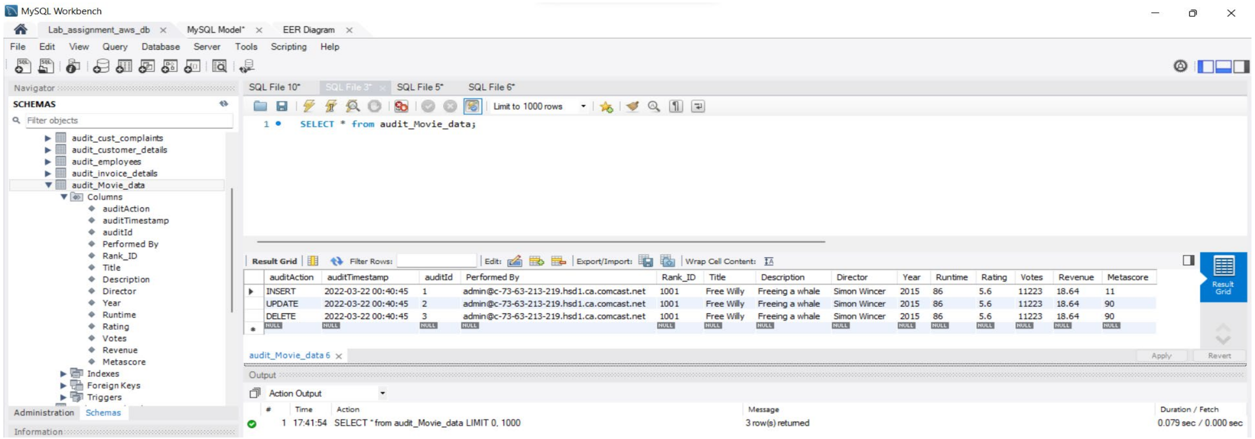
```
DROP TABLE IF EXISTS `sjsu_movie_db`.`audit_Movie_data`;
CREATE TABLE `sjsu_movie_db`.`audit_Movie_data`(
`auditAction` ENUM ('INSERT', 'UPDATE', 'DELETE'),
`auditTimestamp` DATETIME DEFAULT CURRENT_TIMESTAMP,
`auditId` INT AUTO_INCREMENT ,
`Performed By` varchar(200),
`Rank_ID` int,
`Title` varchar(100),
`Description` varchar(300),
`Director` varchar(20),
`Year` int, `Runtime` int,
`Rating` float,
`Votes` bigint,
`Revenue` float,
`Metascore` int,
PRIMARY KEY (`auditId`),
INDEX (`auditTimestamp`));

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`Movie_data_inserts`;
CREATE TRIGGER `sjsu_movie_db`.`Movie_data_inserts`
AFTER INSERT ON `sjsu_movie_db`.`Movie_data`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_Movie_data`
(`auditAction`,`Performed
By`,`Rank_ID`,`Title`,`Description`,`Director`,`Year`,`Runtime`,`Rating`,`Votes`,`Revenue`,`Metascore`)
VALUES
('INSERT',user(),NEW.Rank_ID,NEW.Title,NEW.Description,NEW.Director,NEW.Year,NEW.Runtime,NEW.Rating,NE
W.Votes,NEW.Revenue,NEW.Metascore);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`Movie_data_updates`;
CREATE TRIGGER `sjsu_movie_db`.`Movie_data_updates`
AFTER UPDATE ON `sjsu_movie_db`.`Movie_data`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_Movie_data`
(`auditAction`,`Performed
By`,`Rank_ID`,`Title`,`Description`,`Director`,`Year`,`Runtime`,`Rating`,`Votes`,`Revenue`,`Metascore`)
VALUES
('UPDATE',user(),NEW.Rank_ID,NEW.Title,NEW.Description,NEW.Director,NEW.Year,NEW.Runtime,NEW.Rating,NE
W.Votes,NEW.Revenue,NEW.Metascore);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`Movie_data_deletes`;
CREATE TRIGGER `sjsu_movie_db`.`Movie_data_deletes`
AFTER DELETE ON `sjsu_movie_db`.`Movie_data`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_Movie_data`
(`auditAction`,`Performed
By`,`Rank_ID`,`Title`,`Description`,`Director`,`Year`,`Runtime`,`Rating`,`Votes`,`Revenue`,`Metascore`)
VALUES
('DELETE',user(),OLD.Rank_ID,OLD.Title,OLD.Description,OLD.Director,OLD.Year,OLD.Runtime,OLD.Rating,OLD.Vote
s,OLD.Revenue,OLD.Metascore);
```



```sql
DROP TABLE IF EXISTS `sjsu_movie_db`.`audit_Movie_data`;
CREATE TABLE `sjsu_movie_db`.`audit_Movie_data`
(
    `auditAction` ENUM ('INSERT', 'UPDATE', 'DELETE'),
    `auditTimestamp` DATETIME DEFAULT CURRENT_TIMESTAMP,
    `auditId` INT AUTO_INCREMENT ,
    `Performed By` varchar(200),
    `Rank_ID` int,
    `Title` varchar(100),
    `Description` varchar(300),
    `Director` varchar(20),
    `Year` int,
    `Runtime` int,
    `Rating` float,
    `Votes` bigint,
    `Revenue` float,
    `Metascore` int,
    PRIMARY KEY (`auditId`),
    INDEX (`auditTimestamp`)
);
```



```sql
DROP TRIGGER IF EXISTS `sjsu_movie_db`.`Movie_data_inserts`;
CREATE TRIGGER `sjsu_movie_db`.`Movie_data_inserts`
AFTER INSERT ON `sjsu_movie_db`.`Movie_data`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_Movie_data`
(`auditAction`,`Performed By`,`Rank_ID`,`Title`,`Description`,`Director`,`Year`,`Runtime`,`Rating`,`Votes`,`Revenue`,`Metascore`)
VALUES
('INSERT',user(),NEW.Rank_ID,NEW.Title,NEW.Description,NEW.Director,NEW.Year,NEW.Runtime,NEW.Rating,NEW.Votes,NEW.Revenue,NEW.Metascore);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`Movie_data_updates`;
CREATE TRIGGER `sjsu_movie_db`.`Movie_data_updates`
AFTER UPDATE ON `sjsu_movie_db`.`Movie_data`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_Movie_data`
(`auditAction`,`Performed By`,`Rank_ID`,`Title`,`Description`,`Director`,`Year`,`Runtime`,`Rating`,`Votes`,`Revenue`,`Metascore`)
VALUES
('UPDATE',user(),NEW.Rank_ID,NEW.Title,NEW.Description,NEW.Director,NEW.Year,NEW.Runtime,NEW.Rating,NEW.Votes,NEW.Revenue,NEW.Metascore);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`Movie_data_deletes`;
CREATE TRIGGER `sjsu_movie_db`.`Movie_data_deletes`
AFTER DELETE ON `sjsu_movie_db`.`Movie_data`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_Movie_data`
(`auditAction`,`Performed By`,`Rank_ID`,`Title`,`Description`,`Director`,`Year`,`Runtime`,`Rating`,`Votes`,`Revenue`,`Metascore`)
VALUES
('DELETE',user(),OLD.Rank_ID,OLD.Title,OLD.Description,OLD.Director,OLD.Year,OLD.Runtime,OLD.Rating,OLD.Votes,OLD.Revenue,OLD.Metascore);
```



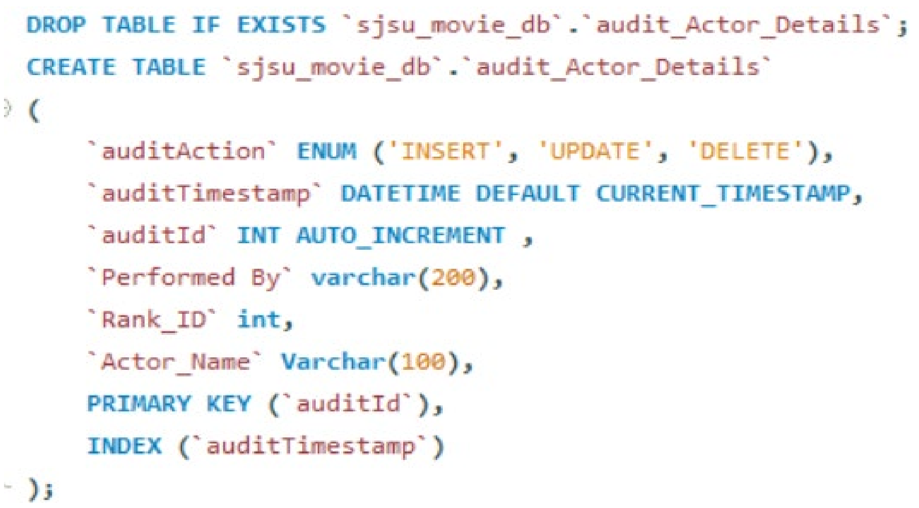
```sql
SELECT * from audit_Movie_data;
```

auditAction	auditTimestamp	auditId	Performed By	Rank_ID	Title	Description	Director	Year	Runtime	Rating	Votes	Revenue	Metascore
INSERT	2022-03-22 00:40:45	1	admin@c-73-63-213-219.hsd1.ca.comcast.net	1001	Free Willy	Freeing a whale	Simon Wincer	2015	86	5.6	11223	18.64	11
UPDATE	2022-03-22 00:40:45	2	admin@c-73-63-213-219.hsd1.ca.comcast.net	1001	Free Willy	Freeing a whale	Simon Wincer	2015	86	5.6	11223	18.64	90
DELETE	2022-03-22 00:40:45	3	admin@c-73-63-213-219.hsd1.ca.comcast.net	1001	Free Willy	Freeing a whale	Simon Wincer	2015	86	5.6	11223	18.64	90
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

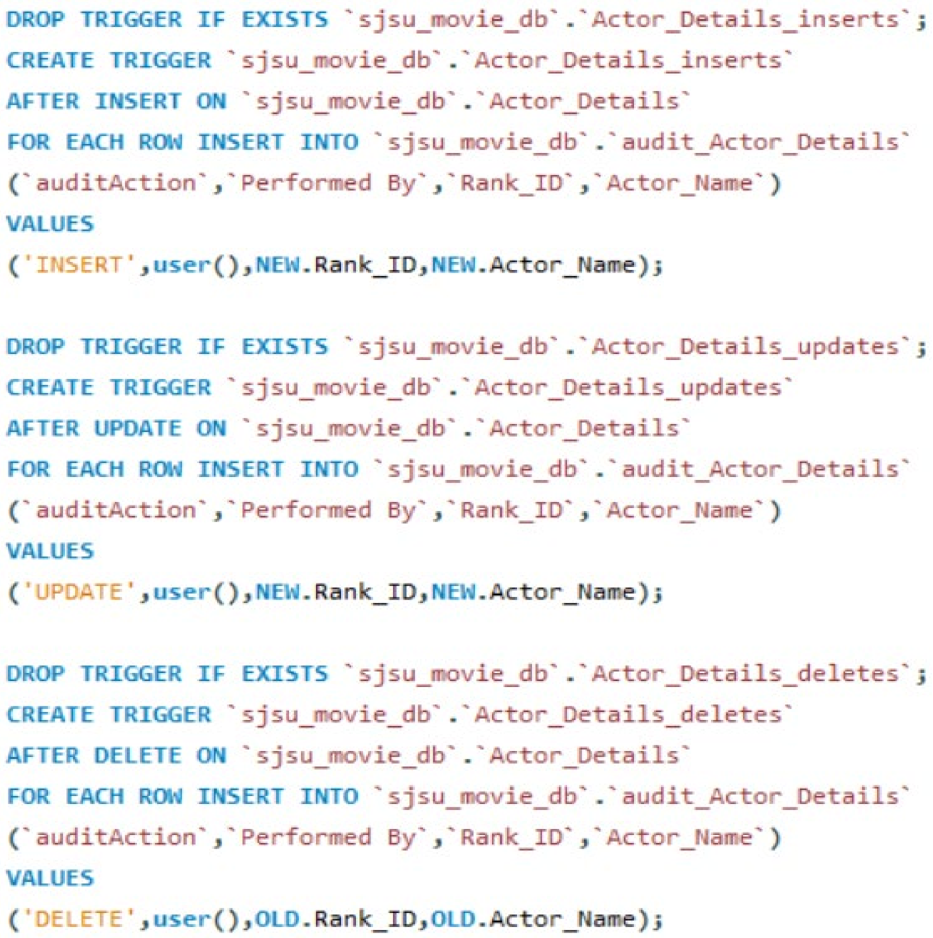
#	Time	Action	Message	Duration / Fetch
1	17:41:54	SELECT * from audit_Movie_data LIMIT 0, 1000	3 row(s) returned	0.079 sec / 0.000 sec

AUDIT ACTOR DETAILS:

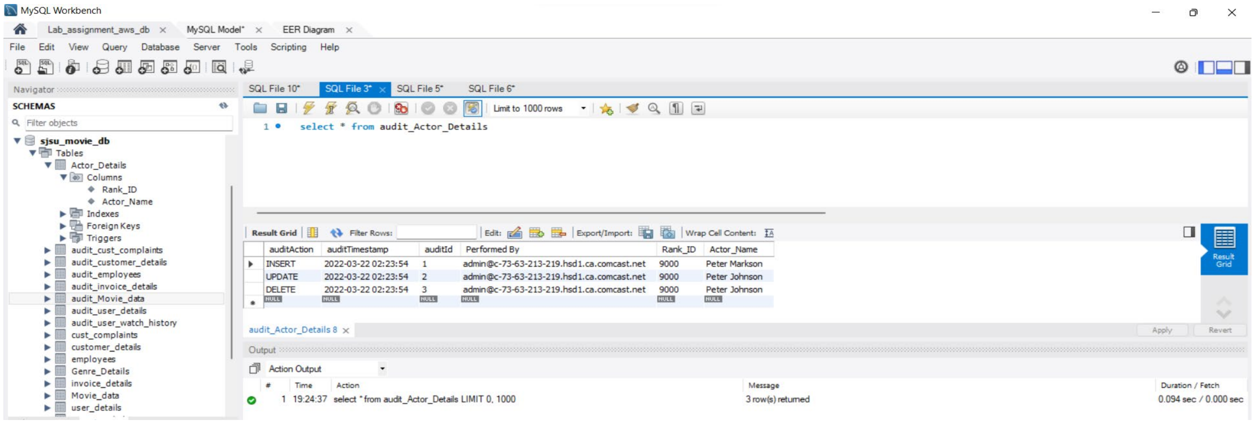
```
DROP TABLE IF EXISTS `sjsu_movie_db`.`audit_Actor_Details`;
CREATE TABLE `sjsu_movie_db`.`audit_Actor_Details`
(
`auditAction` ENUM ('INSERT', 'UPDATE', 'DELETE'),
`auditTimestamp` DATETIME DEFAULT CURRENT_TIMESTAMP,
`auditId` INT AUTO_INCREMENT ,
`Performed By` varchar(200),
`Rank_ID` int,
`Actor_Name` Varchar(100),
PRIMARY KEY (`auditId`),
INDEX (`auditTimestamp`)
);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`Actor_Details_inserts`;
CREATE TRIGGER `sjsu_movie_db`.`Actor_Details_inserts`
AFTER INSERT ON `sjsu_movie_db`.`Actor_Details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_Actor_Details`
(`auditAction`,`Performed By`,`Rank_ID`,`Actor_Name`)
VALUES
('INSERT',user(),NEW.Rank_ID,NEW.Actor_Name);


DROP TRIGGER IF EXISTS `sjsu_movie_db`.`Actor_Details_updates`;
CREATE TRIGGER `sjsu_movie_db`.`Actor_Details_updates`
AFTER UPDATE ON `sjsu_movie_db`.`Actor_Details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_Actor_Details`
(`auditAction`,`Performed By`,`Rank_ID`,`Actor_Name`)
VALUES
('UPDATE',user(),NEW.Rank_ID,NEW.Actor_Name);


DROP TRIGGER IF EXISTS `sjsu_movie_db`.`Actor_Details_deletes`;
CREATE TRIGGER `sjsu_movie_db`.`Actor_Details_deletes`
AFTER DELETE ON `sjsu_movie_db`.`Actor_Details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_Actor_Details`
(`auditAction`,`Performed By`,`Rank_ID`,`Actor_Name`)
VALUES
('DELETE',user(),OLD.Rank_ID,OLD.Actor_Name);
```



```
DROP TABLE IF EXISTS `sjsu_movie_db`.`audit_Actor_Details`;
CREATE TABLE `sjsu_movie_db`.`audit_Actor_Details`
(
     `auditAction` ENUM ('INSERT', 'UPDATE', 'DELETE'),
     `auditTimestamp` DATETIME DEFAULT CURRENT_TIMESTAMP,
     `auditId` INT AUTO_INCREMENT ,
     `Performed By` varchar(200),
     `Rank_ID` int,
     `Actor_Name` Varchar(100),
     PRIMARY KEY (`auditId`),
     INDEX (`auditTimestamp`)
);
```



```
DROP TRIGGER IF EXISTS `sjsu_movie_db`.`Actor_Details_inserts`;
CREATE TRIGGER `sjsu_movie_db`.`Actor_Details_inserts`
AFTER INSERT ON `sjsu_movie_db`.`Actor_Details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_Actor_Details`
(`auditAction`,`Performed By`,`Rank_ID`,`Actor_Name`)
VALUES
('INSERT',user(),NEW.Rank_ID,NEW.Actor_Name);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`Actor_Details_updates`;
CREATE TRIGGER `sjsu_movie_db`.`Actor_Details_updates`
AFTER UPDATE ON `sjsu_movie_db`.`Actor_Details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_Actor_Details`
(`auditAction`,`Performed By`,`Rank_ID`,`Actor_Name`)
VALUES
('UPDATE',user(),NEW.Rank_ID,NEW.Actor_Name);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`Actor_Details_deletes`;
CREATE TRIGGER `sjsu_movie_db`.`Actor_Details_deletes`
AFTER DELETE ON `sjsu_movie_db`.`Actor_Details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_Actor_Details`
(`auditAction`,`Performed By`,`Rank_ID`,`Actor_Name`)
VALUES
('DELETE',user(),OLD.Rank_ID,OLD.Actor_Name);
```



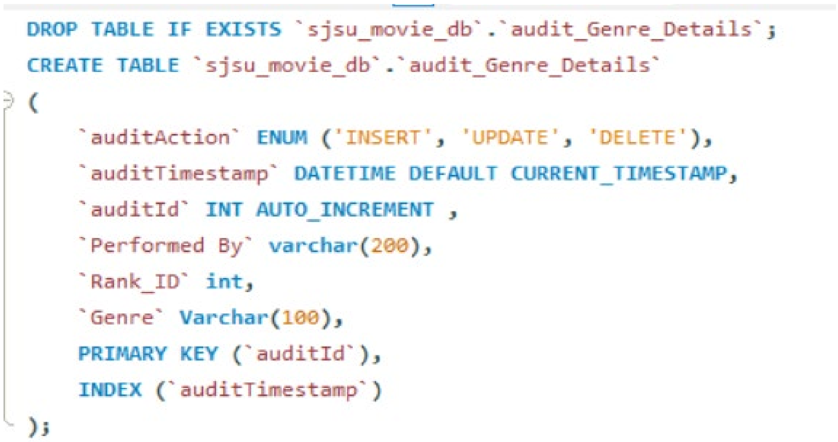
```
select * from audit_Actor_Details
```

auditAction	auditTimestamp	auditId	Performed By	Rank_ID	Actor_Name
INSERT	2022-03-22 02:23:54	1	admin@c-73-63-213-219.hsd1.ca.comcast.net	9000	Peter Markson
UPDATE	2022-03-22 02:23:54	2	admin@c-73-63-213-219.hsd1.ca.comcast.net	9000	Peter Johnson
DELETE	2022-03-22 02:23:54	3	admin@c-73-63-213-219.hsd1.ca.comcast.net	9000	Peter Johnson
NULL	NULL	NULL	NULL	NULL	NULL

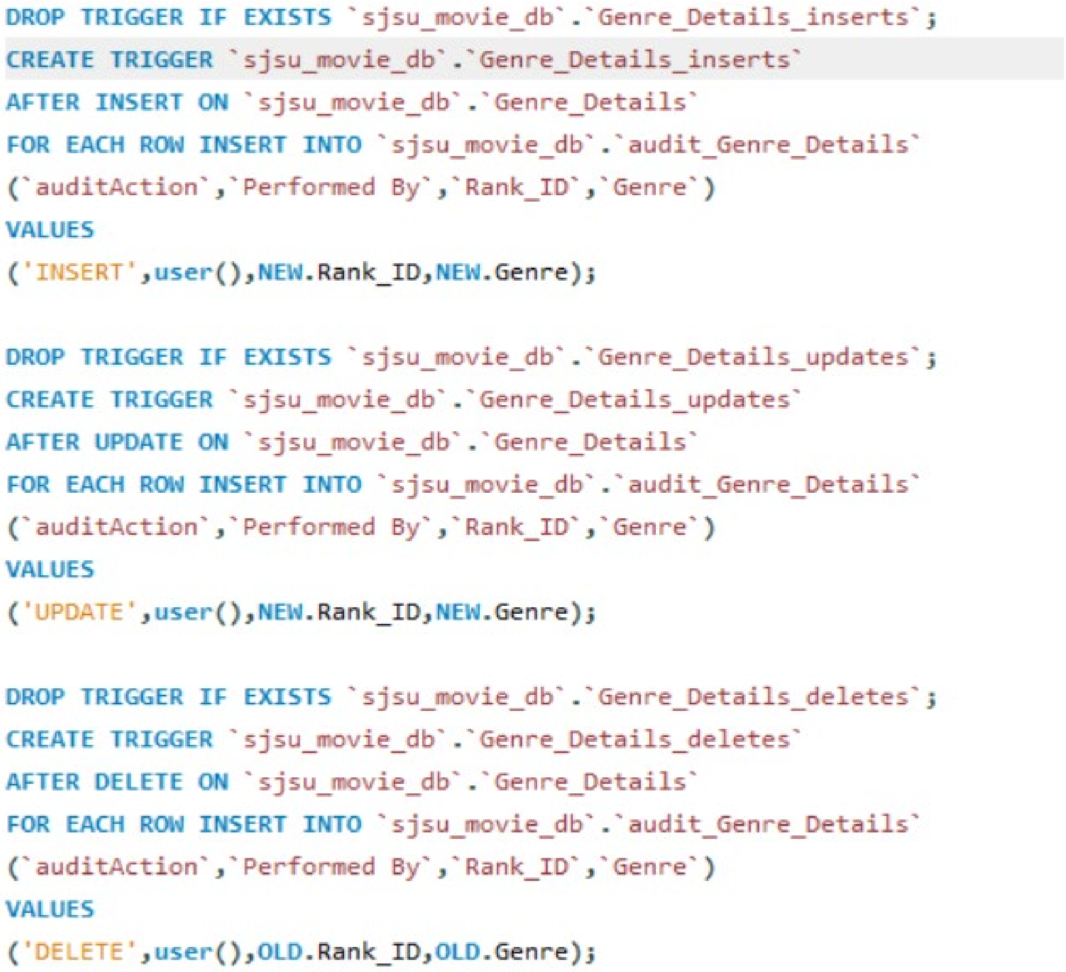
#	Time	Action	Message	Duration / Fetch
1	19:24:37	select * from audit_Actor_Details LIMIT 0, 1000	3 row(s) returned	0.094 sec / 0.000 sec

AUDIT GENRE DETAILS:

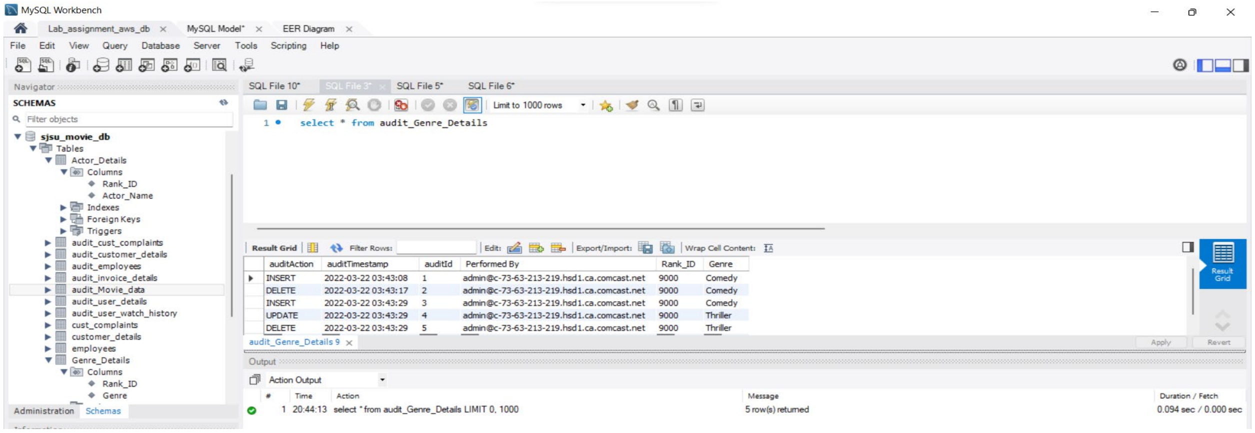
```
DROP TABLE IF EXISTS `sjsu_movie_db`.`audit_Genre_Details`;
CREATE TABLE `sjsu_movie_db`.`audit_Genre_Details`
(
`auditAction` ENUM ('INSERT', 'UPDATE', 'DELETE'),
`auditTimestamp` DATETIME DEFAULT CURRENT_TIMESTAMP,
`auditId` INT AUTO_INCREMENT ,
`Performed By` varchar(200),
`Rank_ID` int,
`Genre` Varchar(100),
PRIMARY KEY (`auditId`),
INDEX (`auditTimestamp`)
);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`Genre_Details_inserts`;
CREATE TRIGGER `sjsu_movie_db`.`Genre_Details_inserts`
AFTER INSERT ON `sjsu_movie_db`.`Genre_Details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_Genre_Details`
(`auditAction`,`Performed By`,`Rank_ID`,`Genre`)
VALUES
('INSERT',user(),NEW.Rank_ID,NEW.Genre);


DROP TRIGGER IF EXISTS `sjsu_movie_db`.`Genre_Details_updates`;
CREATE TRIGGER `sjsu_movie_db`.`Genre_Details_updates`
AFTER UPDATE ON `sjsu_movie_db`.`Genre_Details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_Genre_Details`
(`auditAction`,`Performed By`,`Rank_ID`,`Genre`)
VALUES
('UPDATE',user(),NEW.Rank_ID,NEW.Genre);


DROP TRIGGER IF EXISTS `sjsu_movie_db`.`Genre_Details_deletes`;
CREATE TRIGGER `sjsu_movie_db`.`Genre_Details_deletes`
AFTER DELETE ON `sjsu_movie_db`.`Genre_Details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_Genre_Details`
(`auditAction`,`Performed By`,`Rank_ID`,`Genre`)
VALUES
('DELETE',user(),OLD.Rank_ID,OLD.Genre);
```



```
DROP TABLE IF EXISTS `sjsu_movie_db`.`audit_Genre_Details`;
CREATE TABLE `sjsu_movie_db`.`audit_Genre_Details`
(
    `auditAction` ENUM ('INSERT', 'UPDATE', 'DELETE'),
    `auditTimestamp` DATETIME DEFAULT CURRENT_TIMESTAMP,
    `auditId` INT AUTO_INCREMENT ,
    `Performed By` varchar(200),
    `Rank_ID` int,
    `Genre` Varchar(100),
    PRIMARY KEY (`auditId`),
    INDEX (`auditTimestamp`)
);
```



```
DROP TRIGGER IF EXISTS `sjsu_movie_db`.`Genre_Details_inserts`;
CREATE TRIGGER `sjsu_movie_db`.`Genre_Details_inserts`
AFTER INSERT ON `sjsu_movie_db`.`Genre_Details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_Genre_Details`
(`auditAction`,`Performed By`,`Rank_ID`,`Genre`)
VALUES
('INSERT',user(),NEW.Rank_ID,NEW.Genre);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`Genre_Details_updates`;
CREATE TRIGGER `sjsu_movie_db`.`Genre_Details_updates`
AFTER UPDATE ON `sjsu_movie_db`.`Genre_Details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_Genre_Details`
(`auditAction`,`Performed By`,`Rank_ID`,`Genre`)
VALUES
('UPDATE',user(),NEW.Rank_ID,NEW.Genre);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`Genre_Details_deletes`;
CREATE TRIGGER `sjsu_movie_db`.`Genre_Details_deletes`
AFTER DELETE ON `sjsu_movie_db`.`Genre_Details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_Genre_Details`
(`auditAction`,`Performed By`,`Rank_ID`,`Genre`)
VALUES
('DELETE',user(),OLD.Rank_ID,OLD.Genre);
```



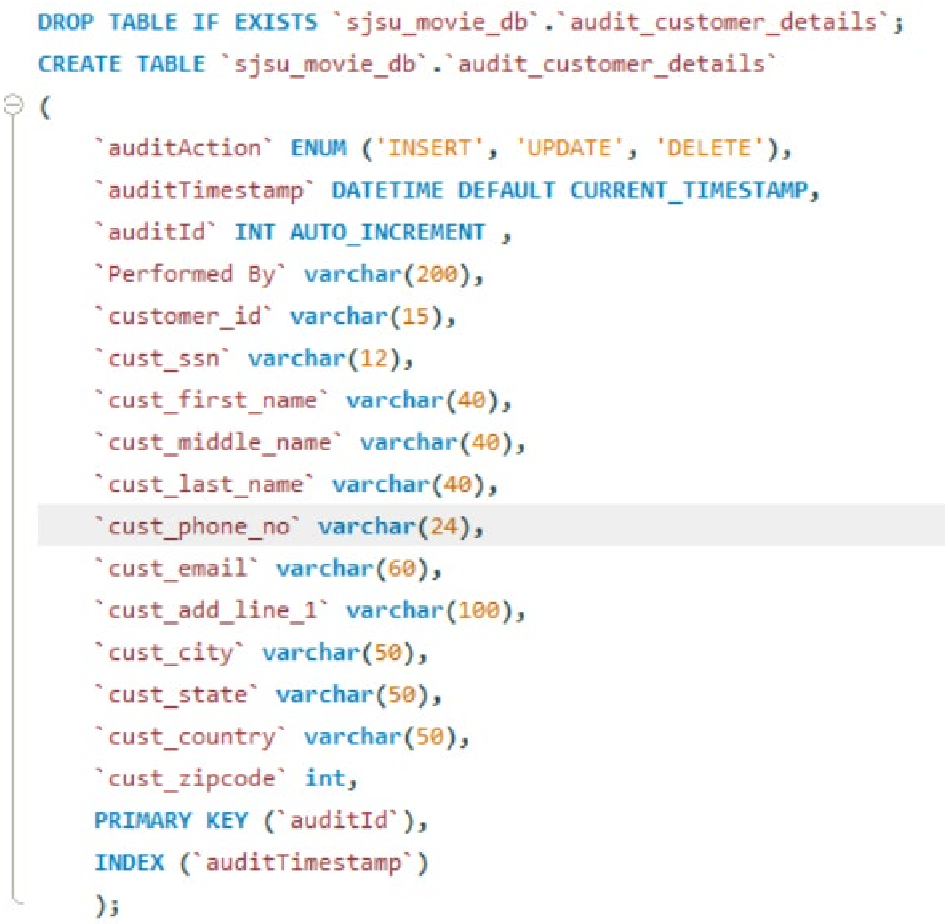
```
select * from audit_Genre_Details
```

auditAction	auditTimestamp	auditId	Performed By	Rank_ID	Genre
INSERT	2022-03-22 03:43:08	1	admin@c-73-63-213-219.hsd1.ca.comcast.net	9000	Comedy
DELETE	2022-03-22 03:43:17	2	admin@c-73-63-213-219.hsd1.ca.comcast.net	9000	Comedy
INSERT	2022-03-22 03:43:29	3	admin@c-73-63-213-219.hsd1.ca.comcast.net	9000	Comedy
UPDATE	2022-03-22 03:43:29	4	admin@c-73-63-213-219.hsd1.ca.comcast.net	9000	Thriller
DELETE	2022-03-22 03:43:29	5	admin@c-73-63-213-219.hsd1.ca.comcast.net	9000	Thriller

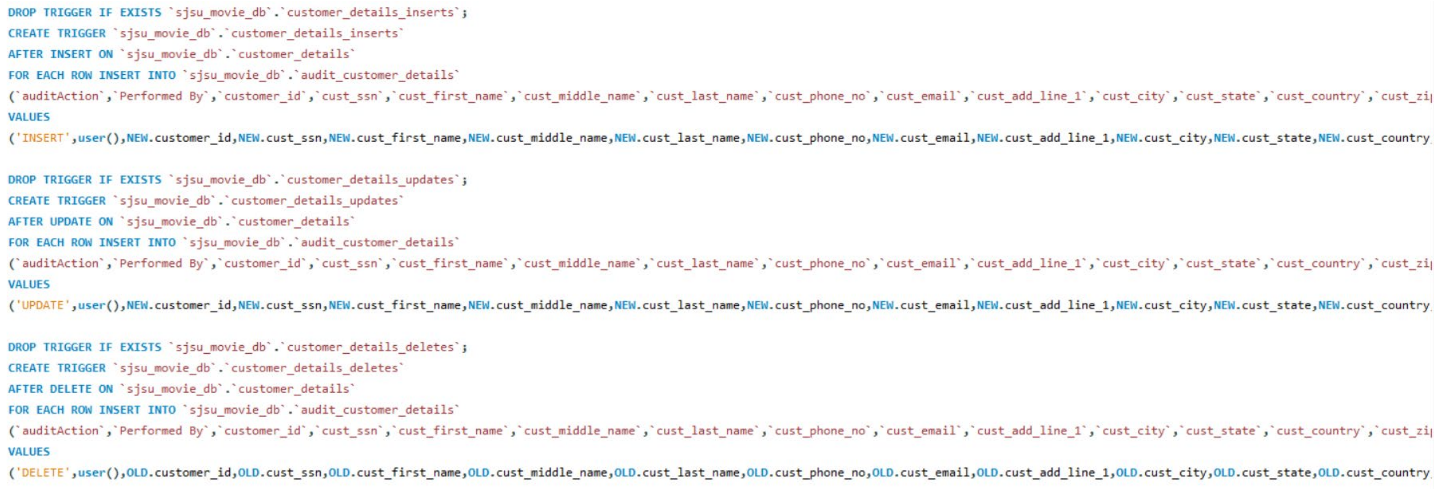
#	Time	Action	Message	Duration / Fetch
1	20:44:13	select * from audit_Genre_Details LIMIT 0, 1000	5 row(s) returned	0.094 sec / 0.000 sec

AUDIT CUSTOMER DETAILS:

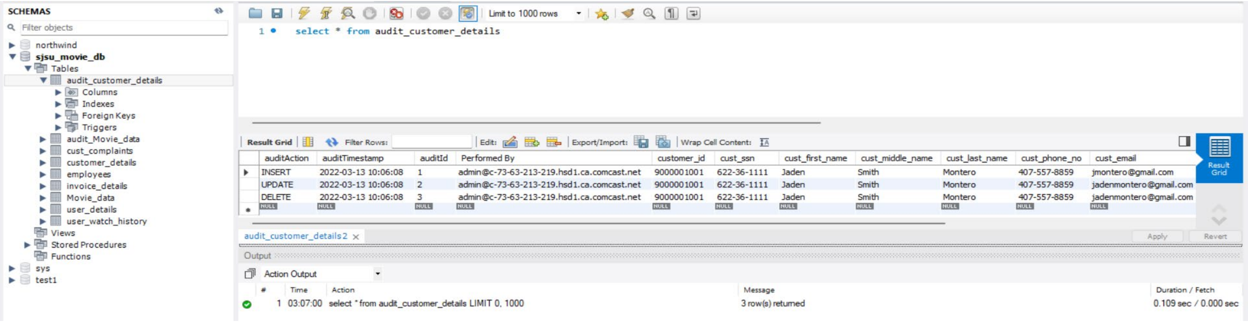
```
DROP TABLE IF EXISTS `sjsu_movie_db`.`audit_customer_details`;
CREATE TABLE `sjsu_movie_db`.`audit_customer_details`
(
`auditAction` ENUM ('INSERT', 'UPDATE', 'DELETE'),
`auditTimestamp` DATETIME DEFAULT CURRENT_TIMESTAMP,
`auditId` INT AUTO_INCREMENT ,
`Performed By` varchar(200),
`customer_id` varchar(15),  `cust_ssn` varchar(12),
`cust_first_name` varchar(40), `cust_middle_name` varchar(40),
`cust_last_name` varchar(40), `cust_phone_no` varchar(24),
`cust_email` varchar(60), `cust_add_line_1` varchar(100),
`cust_city` varchar(50),
`cust_state` varchar(50),
`cust_country` varchar(50),
`cust_zipcode` int,
PRIMARY KEY (`auditId`),
INDEX (`auditTimestamp`)
);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`customer_details_inserts`;
CREATE TRIGGER `sjsu_movie_db`.`customer_details_inserts`
AFTER INSERT ON `sjsu_movie_db`.`customer_details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_customer_details`
(`auditAction`,`Performed By`,`customer_id`,`cust_ssn`,`cust_first_name`,`cust_middle_name`,
`cust_last_name`,`cust_phone_no`,`cust_email`,`cust_add_line_1`,`cust_city`,`cust_state`,`cust_country`,`cust_zipcode`)
VALUES
('INSERT',user(),NEW.customer_id,NEW.cust_ssn,NEW.cust_first_name,NEW.cust_middle_name,NEW.cust_last_name,
NEW.cust_phone_no,NEW.cust_email,NEW.cust_add_line_1,NEW.cust_city,NEW.cust_state,NEW.cust_country
,NEW.cust_zipcode);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`customer_details_updates`;
CREATE TRIGGER `sjsu_movie_db`.`customer_details_updates`
AFTER UPDATE ON `sjsu_movie_db`.`customer_details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_customer_details`
(`auditAction`,`Performed By`,`customer_id`,`cust_ssn`,`cust_first_name`,`cust_middle_name`,
`cust_last_name`,`cust_phone_no`,`cust_email`,`cust_add_line_1`,`cust_city`,`cust_state`,`cust_country`,`cust_zipcode`)
VALUES
('UPDATE',user(),NEW.customer_id,NEW.cust_ssn,NEW.cust_first_name,
NEW.cust_middle_name,NEW.cust_last_name,NEW.cust_phone_no,NEW.cust_email,NEW.cust_add_line_1,
NEW.cust_city,NEW.cust_state,NEW.cust_country,NEW.cust_zipcode);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`customer_details_deletes`;
CREATE TRIGGER `sjsu_movie_db`.`customer_details_deletes`
AFTER DELETE ON `sjsu_movie_db`.`customer_details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_customer_details`
(`auditAction`,`Performed By`,`customer_id`,`cust_ssn`,`cust_first_name`,`cust_middle_name`,
`cust_last_name`,`cust_phone_no`,`cust_email`,`cust_add_line_1`,`cust_city`,`cust_state`,`cust_country`,`cust_zipcode`)
VALUES
('DELETE',user(),OLD.customer_id,OLD.cust_ssn,OLD.cust_first_name
,OLD.cust_middle_name,OLD.cust_last_name,OLD.cust_phone_no,OLD.cust_email,OLD.cust_add_line_1,
OLD.cust_city,OLD.cust_state,OLD.cust_country,OLD.cust_zipcode);
```



```sql
DROP TABLE IF EXISTS `sjsu_movie_db`.`audit_customer_details`;
CREATE TABLE `sjsu_movie_db`.`audit_customer_details`
(
    `auditAction` ENUM ('INSERT', 'UPDATE', 'DELETE'),
    `auditTimestamp` DATETIME DEFAULT CURRENT_TIMESTAMP,
    `auditId` INT AUTO_INCREMENT ,
    `Performed By` varchar(200),
    `customer_id` varchar(15),
    `cust_ssn` varchar(12),
    `cust_first_name` varchar(40),
    `cust_middle_name` varchar(40),
    `cust_last_name` varchar(40),
    `cust_phone_no` varchar(24),
    `cust_email` varchar(60),
    `cust_add_line_1` varchar(100),
    `cust_city` varchar(50),
    `cust_state` varchar(50),
    `cust_country` varchar(50),
    `cust_zipcode` int,
    PRIMARY KEY (`auditId`),
    INDEX (`auditTimestamp`)
    );
```



```sql
DROP TRIGGER IF EXISTS `sjsu_movie_db`.`customer_details_inserts`;
CREATE TRIGGER `sjsu_movie_db`.`customer_details_inserts`
AFTER INSERT ON `sjsu_movie_db`.`customer_details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_customer_details`
(`auditAction`,`Performed By`,`customer_id`,`cust_ssn`,`cust_first_name`,`cust_middle_name`,`cust_last_name`,`cust_phone_no`,`cust_email`,`cust_add_line_1`,`cust_city`,`cust_state`,`cust_country`,`cust_zi
VALUES
('INSERT',user(),NEW.customer_id,NEW.cust_ssn,NEW.cust_first_name,NEW.cust_middle_name,NEW.cust_last_name,NEW.cust_phone_no,NEW.cust_email,NEW.cust_add_line_1,NEW.cust_city,NEW.cust_state,NEW.cust_country

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`customer_details_updates`;
CREATE TRIGGER `sjsu_movie_db`.`customer_details_updates`
AFTER UPDATE ON `sjsu_movie_db`.`customer_details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_customer_details`
(`auditAction`,`Performed By`,`customer_id`,`cust_ssn`,`cust_first_name`,`cust_middle_name`,`cust_last_name`,`cust_phone_no`,`cust_email`,`cust_add_line_1`,`cust_city`,`cust_state`,`cust_country`,`cust_zi
VALUES
('UPDATE',user(),NEW.customer_id,NEW.cust_ssn,NEW.cust_first_name,NEW.cust_middle_name,NEW.cust_last_name,NEW.cust_phone_no,NEW.cust_email,NEW.cust_add_line_1,NEW.cust_city,NEW.cust_state,NEW.cust_country

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`customer_details_deletes`;
CREATE TRIGGER `sjsu_movie_db`.`customer_details_deletes`
AFTER DELETE ON `sjsu_movie_db`.`customer_details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_customer_details`
(`auditAction`,`Performed By`,`customer_id`,`cust_ssn`,`cust_first_name`,`cust_middle_name`,`cust_last_name`,`cust_phone_no`,`cust_email`,`cust_add_line_1`,`cust_city`,`cust_state`,`cust_country`,`cust_zi
VALUES
('DELETE',user(),OLD.customer_id,OLD.cust_ssn,OLD.cust_first_name,OLD.cust_middle_name,OLD.cust_last_name,OLD.cust_phone_no,OLD.cust_email,OLD.cust_add_line_1,OLD.cust_city,OLD.cust_state,OLD.cust_country
```



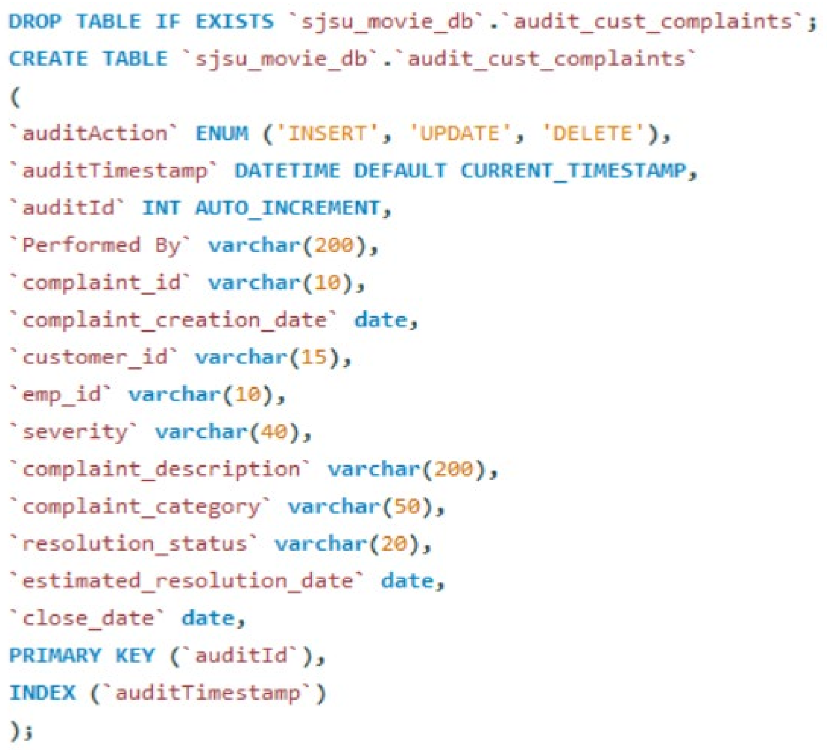
```sql
select * from audit_customer_details
```

auditAction	auditTimestamp	auditId	Performed By	customer_id	cust_ssn	cust_first_name	cust_middle_name	cust_last_name	cust_phone_no	cust_email
INSERT	2022-03-13 10:06:08	1	admin@c-73-63-213-219.hsd1.ca.comcast.net	9000001001	622-36-1111	Jaden	Smith	Montero	407-557-8859	jmontero@gmail.com
UPDATE	2022-03-13 10:06:08	2	admin@c-73-63-213-219.hsd1.ca.comcast.net	9000001001	622-36-1111	Jaden	Smith	Montero	407-557-8859	jadenmontero@gmail.com
DELETE	2022-03-13 10:06:08	3	admin@c-73-63-213-219.hsd1.ca.comcast.net	9000001001	622-36-1111	Jaden	Smith	Montero	407-557-8859	jadenmontero@gmail.com
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

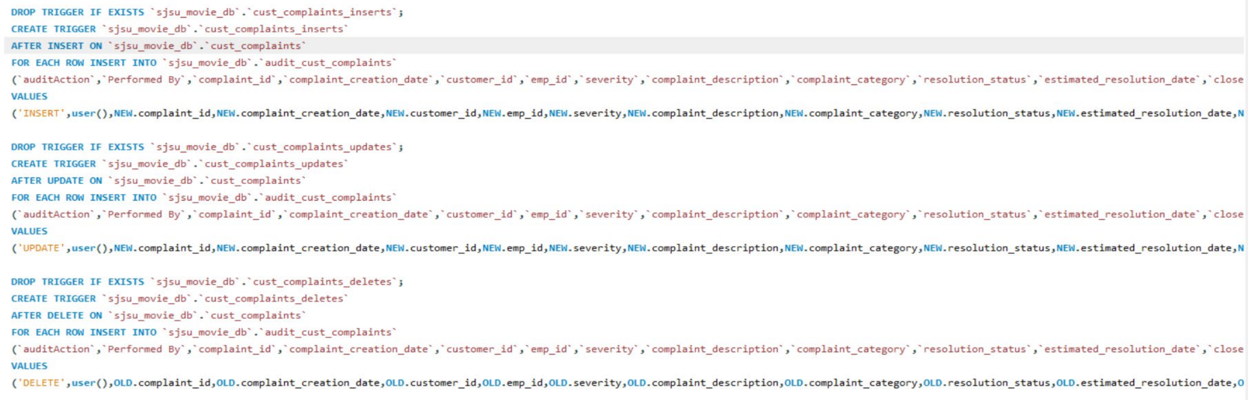
#	Time	Action	Message	Duration / Fetch
1	03:07:00	select * from audit_customer_details LIMIT 0, 1000	3 row(s) returned	0.109 sec / 0.000 sec

AUDIT CUSTOMER COMPLAINTS:

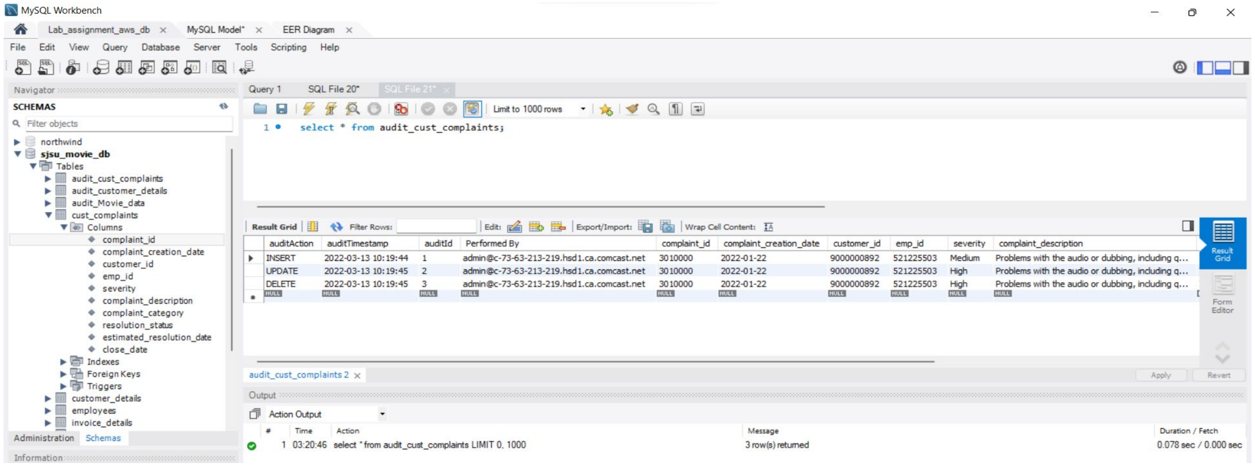
```
DROP TABLE IF EXISTS `sjsu_movie_db`.`audit_cust_complaints`;
CREATE TABLE `sjsu_movie_db`.`audit_cust_complaints`(
`auditAction` ENUM ('INSERT', 'UPDATE', 'DELETE'),
`auditTimestamp` DATETIME DEFAULT CURRENT_TIMESTAMP,
`auditId` INT AUTO_INCREMENT,
`Performed By` varchar(200), `complaint_id` varchar(10),
`complaint_creation_date` date, `customer_id` varchar(15),
`emp_id` varchar(10), `severity` varchar(40),
`complaint_description` varchar(200), `complaint_category` varchar(50),
`resolution_status` varchar(20), `estimated_resolution_date` date,
`close_date` date, PRIMARY KEY (`auditId`), INDEX (`auditTimestamp`));

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`cust_complaints_inserts`;
CREATE TRIGGER `sjsu_movie_db`.`cust_complaints_inserts`
AFTER INSERT ON `sjsu_movie_db`.`cust_complaints`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_cust_complaints`
(`auditAction`,`Performed
By`,`complaint_id`,`complaint_creation_date`,`customer_id`,`emp_id`,`severity`,`complaint_description`,`
complaint_category`,`resolution_status`,`estimated_resolution_date`,`close_date`)
VALUES
('INSERT',user(),NEW.complaint_id,NEW.complaint_creation_date,NEW.customer_id,NEW.emp_id,NEW.s
everity,NEW.complaint_description,NEW.complaint_category,NEW.resolution_status,NEW.estimated_res
olution_date,NEW.close_date);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`cust_complaints_updates`;
CREATE TRIGGER `sjsu_movie_db`.`cust_complaints_updates`
AFTER UPDATE ON `sjsu_movie_db`.`cust_complaints`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_cust_complaints`
(`auditAction`,`Performed
By`,`complaint_id`,`complaint_creation_date`,`customer_id`,`emp_id`,`severity`,`complaint_description`,`
complaint_category`,`resolution_status`,`estimated_resolution_date`,`close_date`)
VALUES
('UPDATE',user(),NEW.complaint_id,NEW.complaint_creation_date,NEW.customer_id,NEW.emp_id,NEW.
severity,NEW.complaint_description,NEW.complaint_category,NEW.resolution_status,NEW.estimated_re
solution_date,NEW.close_date);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`cust_complaints_deletes`;
CREATE TRIGGER `sjsu_movie_db`.`cust_complaints_deletes`
AFTER DELETE ON `sjsu_movie_db`.`cust_complaints`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_cust_complaints`
(`auditAction`,`Performed
By`,`complaint_id`,`complaint_creation_date`,`customer_id`,`emp_id`,`severity`,`complaint_description`,`
complaint_category`,`resolution_status`,`estimated_resolution_date`,`close_date`)
VALUES
('DELETE',user(),OLD.complaint_id,OLD.complaint_creation_date,OLD.customer_id,OLD.emp_id,OLD.seve
rity,OLD.complaint_description,OLD.complaint_category,OLD.resolution_status,OLD.estimated_resolutio
n_date,OLD.close_date);
```



```
DROP TABLE IF EXISTS `sjsu_movie_db`.`audit_cust_complaints`;
CREATE TABLE `sjsu_movie_db`.`audit_cust_complaints`
(
`auditAction` ENUM ('INSERT', 'UPDATE', 'DELETE'),
`auditTimestamp` DATETIME DEFAULT CURRENT_TIMESTAMP,
`auditId` INT AUTO_INCREMENT,
`Performed By` varchar(200),
`complaint_id` varchar(10),
`complaint_creation_date` date,
`customer_id` varchar(15),
`emp_id` varchar(10),
`severity` varchar(40),
`complaint_description` varchar(200),
`complaint_category` varchar(50),
`resolution_status` varchar(20),
`estimated_resolution_date` date,
`close_date` date,
PRIMARY KEY (`auditId`),
INDEX (`auditTimestamp`)
);
```



```
DROP TRIGGER IF EXISTS `sjsu_movie_db`.`cust_complaints_inserts`;
CREATE TRIGGER `sjsu_movie_db`.`cust_complaints_inserts`
AFTER INSERT ON `sjsu_movie_db`.`cust_complaints`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_cust_complaints`
(`auditAction`,`Performed By`,`complaint_id`,`complaint_creation_date`,`customer_id`,`emp_id`,`severity`,`complaint_description`,`complaint_category`,`resolution_status`,`estimated_resolution_date`,`close
VALUES
('INSERT',user(),NEW.complaint_id,NEW.complaint_creation_date,NEW.customer_id,NEW.emp_id,NEW.severity,NEW.complaint_description,NEW.complaint_category,NEW.resolution_status,NEW.estimated_resolution_date,N

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`cust_complaints_updates`;
CREATE TRIGGER `sjsu_movie_db`.`cust_complaints_updates`
AFTER UPDATE ON `sjsu_movie_db`.`cust_complaints`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_cust_complaints`
(`auditAction`,`Performed By`,`complaint_id`,`complaint_creation_date`,`customer_id`,`emp_id`,`severity`,`complaint_description`,`complaint_category`,`resolution_status`,`estimated_resolution_date`,`close
VALUES
('UPDATE',user(),NEW.complaint_id,NEW.complaint_creation_date,NEW.customer_id,NEW.emp_id,NEW.severity,NEW.complaint_description,NEW.complaint_category,NEW.resolution_status,NEW.estimated_resolution_date,N

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`cust_complaints_deletes`;
CREATE TRIGGER `sjsu_movie_db`.`cust_complaints_deletes`
AFTER DELETE ON `sjsu_movie_db`.`cust_complaints`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_cust_complaints`
(`auditAction`,`Performed By`,`complaint_id`,`complaint_creation_date`,`customer_id`,`emp_id`,`severity`,`complaint_description`,`complaint_category`,`resolution_status`,`estimated_resolution_date`,`close
VALUES
('DELETE',user(),OLD.complaint_id,OLD.complaint_creation_date,OLD.customer_id,OLD.emp_id,OLD.severity,OLD.complaint_description,OLD.complaint_category,OLD.resolution_status,OLD.estimated_resolution_date,O
```



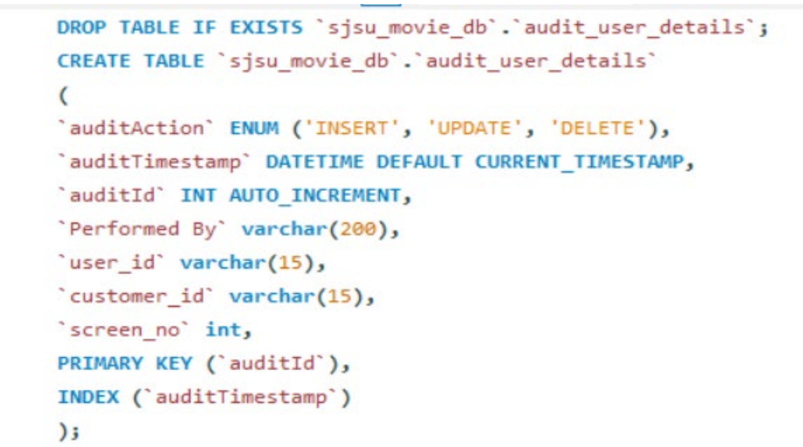
```
select * from audit_cust_complaints;
```

auditAction	auditTimestamp	auditId	Performed By	complaint_id	complaint_creation_date	customer_id	emp_id	severity	complaint_description
INSERT	2022-03-13 10:19:44	1	admin@c-73-63-213-219.hsd1.ca.comcast.net	3010000	2022-01-22	9000000892	521225503	Medium	Problems with the audio or dubbing, including q...
UPDATE	2022-03-13 10:19:45	2	admin@c-73-63-213-219.hsd1.ca.comcast.net	3010000	2022-01-22	9000000892	521225503	High	Problems with the audio or dubbing, including q...
DELETE	2022-03-13 10:19:45	3	admin@c-73-63-213-219.hsd1.ca.comcast.net	3010000	2022-01-22	9000000892	521225503	High	Problems with the audio or dubbing, including q...
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

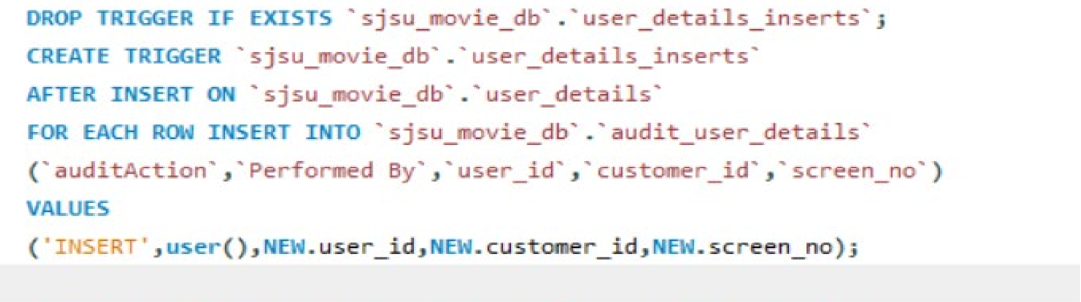
#	Time	Action	Message	Duration / Fetch
1	03:20:46	select * from audit_cust_complaints LIMIT 0, 1000	3 row(s) returned	0.078 sec / 0.000 sec

AUDIT USER DETAILS:

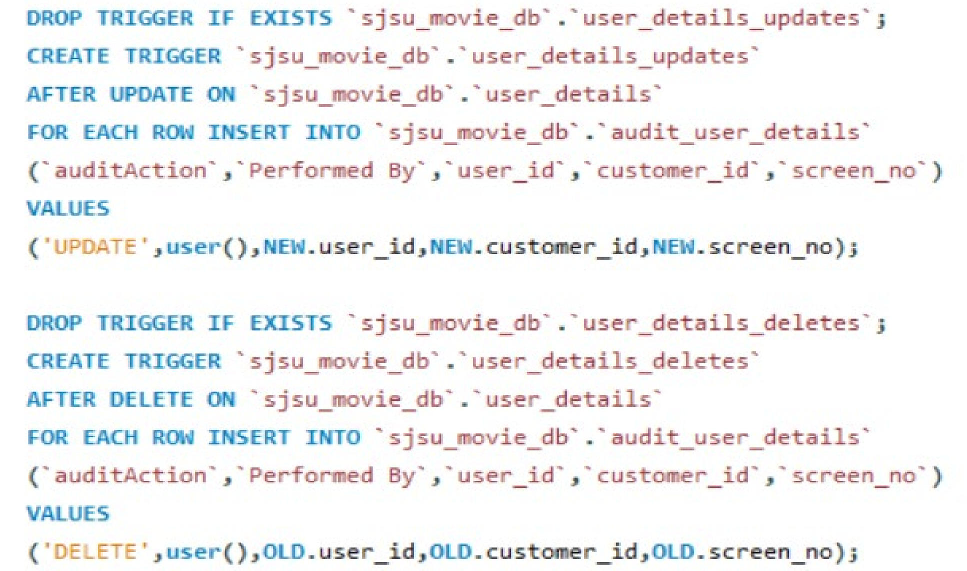
```
DROP TABLE IF EXISTS `sjsu_movie_db`.`audit_user_details`;
CREATE TABLE `sjsu_movie_db`.`audit_user_details`
(
`auditAction` ENUM ('INSERT', 'UPDATE', 'DELETE'),
`auditTimestamp` DATETIME DEFAULT CURRENT_TIMESTAMP,
`auditId` INT AUTO_INCREMENT,
`Performed By` varchar(200),
`user_id` varchar(15),
`customer_id` varchar(15),
`screen_no` int,
PRIMARY KEY (`auditId`),
INDEX (`auditTimestamp`)
);


DROP TRIGGER IF EXISTS `sjsu_movie_db`.`user_details_inserts`;
CREATE TRIGGER `sjsu_movie_db`.`user_details_inserts`
AFTER INSERT ON `sjsu_movie_db`.`user_details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_user_details`
(`auditAction`,`Performed By`,`user_id`,`customer_id`,`screen_no`)
VALUES
('INSERT',user(),NEW.user_id,NEW.customer_id,NEW.screen_no);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`user_details_updates`;
CREATE TRIGGER `sjsu_movie_db`.`user_details_updates`
AFTER UPDATE ON `sjsu_movie_db`.`user_details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_user_details`
(`auditAction`,`Performed By`,`user_id`,`customer_id`,`screen_no`)
VALUES
('UPDATE',user(),NEW.user_id,NEW.customer_id,NEW.screen_no);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`user_details_deletes`;
CREATE TRIGGER `sjsu_movie_db`.`user_details_deletes`
AFTER DELETE ON `sjsu_movie_db`.`user_details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_user_details`
(`auditAction`,`Performed By`,`user_id`,`customer_id`,`screen_no`)
VALUES
('DELETE',user(),OLD.user_id,OLD.customer_id,OLD.screen_no);
```



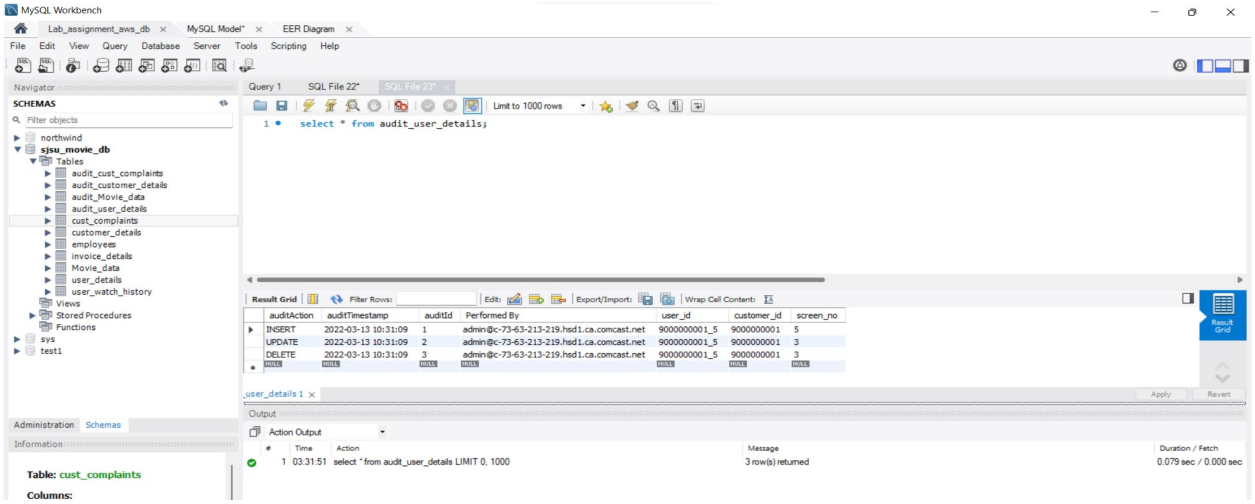
```
DROP TABLE IF EXISTS `sjsu_movie_db`.`audit_user_details`;
CREATE TABLE `sjsu_movie_db`.`audit_user_details`
(
`auditAction` ENUM ('INSERT', 'UPDATE', 'DELETE'),
`auditTimestamp` DATETIME DEFAULT CURRENT_TIMESTAMP,
`auditId` INT AUTO_INCREMENT,
`Performed By` varchar(200),
`user_id` varchar(15),
`customer_id` varchar(15),
`screen_no` int,
PRIMARY KEY (`auditId`),
INDEX (`auditTimestamp`)
);
```



```
DROP TRIGGER IF EXISTS `sjsu_movie_db`.`user_details_inserts`;
CREATE TRIGGER `sjsu_movie_db`.`user_details_inserts`
AFTER INSERT ON `sjsu_movie_db`.`user_details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_user_details`
(`auditAction`,`Performed By`,`user_id`,`customer_id`,`screen_no`)
VALUES
('INSERT',user(),NEW.user_id,NEW.customer_id,NEW.screen_no);
```



```
DROP TRIGGER IF EXISTS `sjsu_movie_db`.`user_details_updates`;
CREATE TRIGGER `sjsu_movie_db`.`user_details_updates`
AFTER UPDATE ON `sjsu_movie_db`.`user_details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_user_details`
(`auditAction`,`Performed By`,`user_id`,`customer_id`,`screen_no`)
VALUES
('UPDATE',user(),NEW.user_id,NEW.customer_id,NEW.screen_no);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`user_details_deletes`;
CREATE TRIGGER `sjsu_movie_db`.`user_details_deletes`
AFTER DELETE ON `sjsu_movie_db`.`user_details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_user_details`
(`auditAction`,`Performed By`,`user_id`,`customer_id`,`screen_no`)
VALUES
('DELETE',user(),OLD.user_id,OLD.customer_id,OLD.screen_no);
```



```
select * from audit_user_details;
```

	auditAction	auditTimestamp	auditId	Performed By	user_id	customer_id	screen_no
▶	INSERT	2022-03-13 10:31:09	1	admin@c-73-63-213-219.hsd1.ca.comcast.net	9000000001_5	9000000001	5
	UPDATE	2022-03-13 10:31:09	2	admin@c-73-63-213-219.hsd1.ca.comcast.net	9000000001_5	9000000001	3
	DELETE	2022-03-13 10:31:09	3	admin@c-73-63-213-219.hsd1.ca.comcast.net	9000000001_5	9000000001	3
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

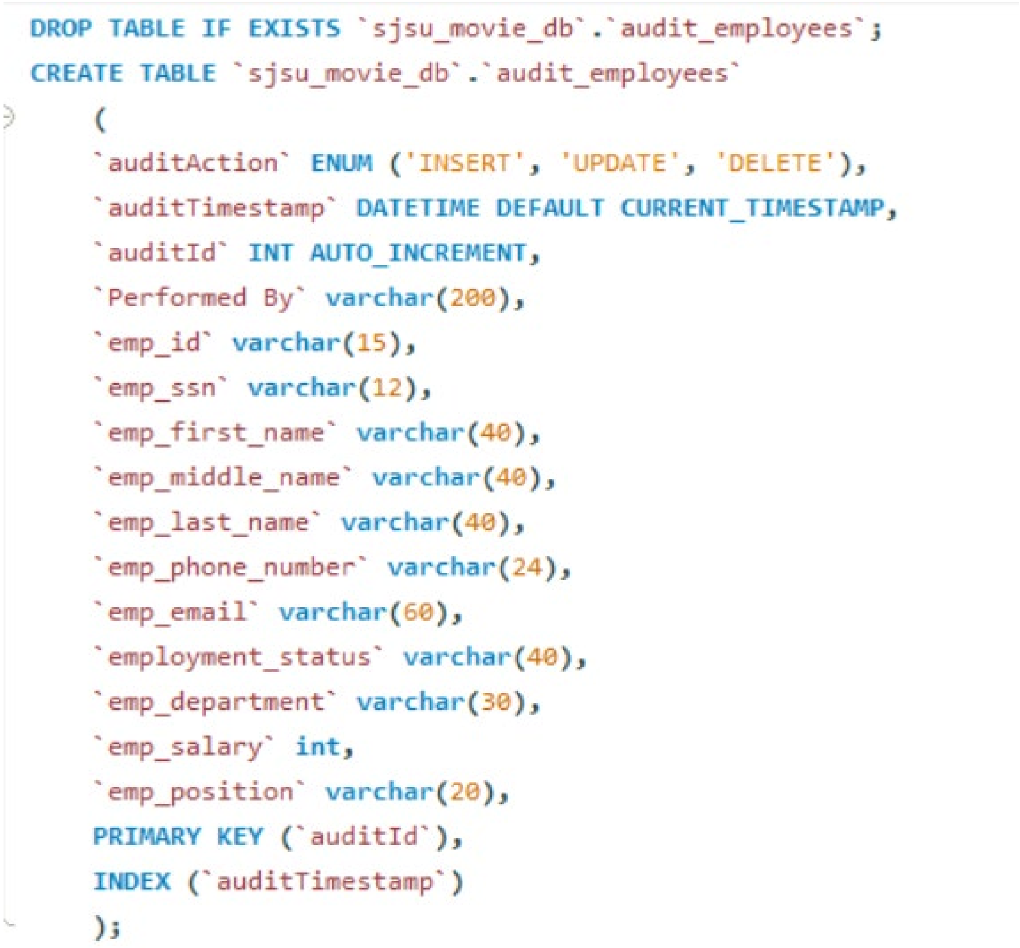
	#	Time	Action	Message	Duration / Fetch
✓	1	03:31:51	select * from audit_user_details LIMIT 0, 1000	3 row(s) returned	0.079 sec / 0.000 sec

Table: cust_complaints

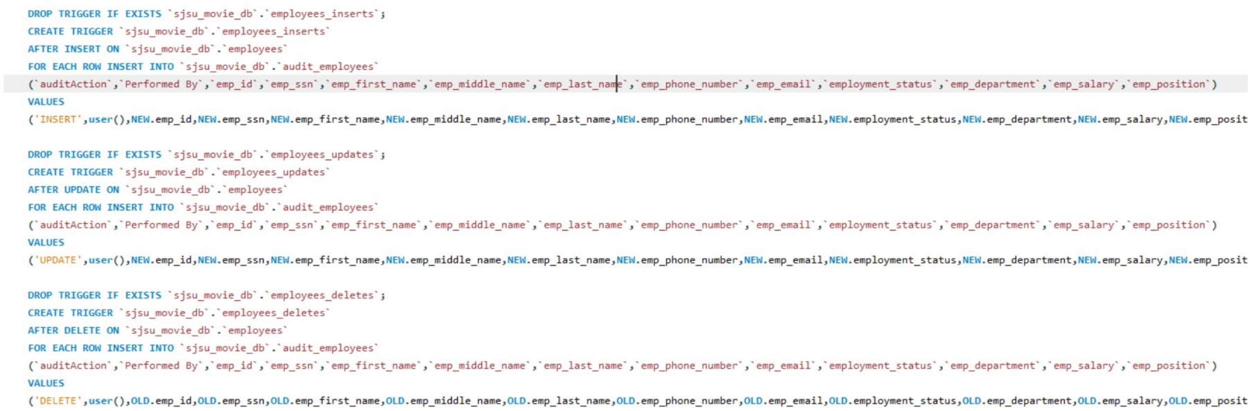
Columns:

AUDIT EMPLOYEES:

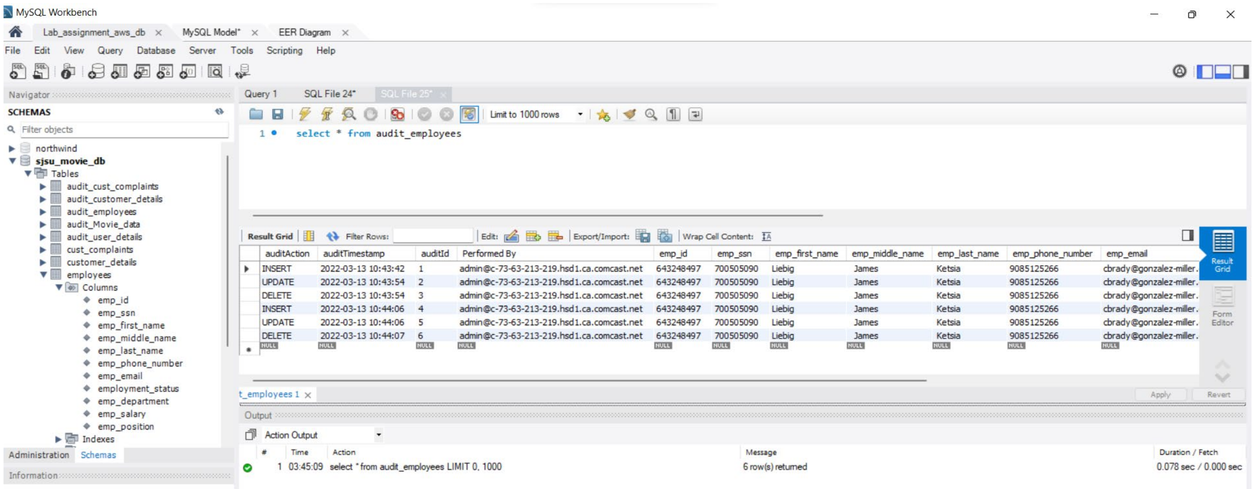
```
DROP TABLE IF EXISTS `sjsu_movie_db`.`audit_employees`;
CREATE TABLE `sjsu_movie_db`.`audit_employees`(
`auditAction` ENUM ('INSERT', 'UPDATE', 'DELETE'),
`auditTimestamp` DATETIME DEFAULT CURRENT_TIMESTAMP,
`auditId` INT AUTO_INCREMENT, `Performed By` varchar(200), `emp_id` varchar(15),
`emp_ssn` varchar(12), `emp_first_name` varchar(40),
`emp_middle_name` varchar(40), `emp_last_name` varchar(40),
`emp_phone_number` varchar(24), `emp_email` varchar(60),
`employment_status` varchar(40), `emp_department` varchar(30),
`emp_salary` int, `emp_position` varchar(20),
PRIMARY KEY (`auditId`), INDEX (`auditTimestamp`));

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`employees_inserts`;
CREATE TRIGGER `sjsu_movie_db`.`employees_inserts`
AFTER INSERT ON `sjsu_movie_db`.`employees`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_employees`
(`auditAction`,`Performed
By`,`emp_id`,`emp_ssn`,`emp_first_name`,`emp_middle_name`,`emp_last_name`,`emp_phone_number`,
`emp_email`,`employment_status`,`emp_department`,`emp_salary`,`emp_position`)
VALUES
('INSERT',user(),NEW.emp_id,NEW.emp_ssn,NEW.emp_first_name,NEW.emp_middle_name,NEW.emp_l
ast_name,NEW.emp_phone_number,NEW.emp_email,NEW.employment_status,NEW.emp_department,
NEW.emp_salary,NEW.emp_position);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`employees_updates`;
CREATE TRIGGER `sjsu_movie_db`.`employees_updates`
AFTER UPDATE ON `sjsu_movie_db`.`employees`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_employees`
(`auditAction`,`Performed
By`,`emp_id`,`emp_ssn`,`emp_first_name`,`emp_middle_name`,`emp_last_name`,`emp_phone_number`,
`emp_email`,`employment_status`,`emp_department`,`emp_salary`,`emp_position`)
VALUES
('UPDATE',user(),NEW.emp_id,NEW.emp_ssn,NEW.emp_first_name,NEW.emp_middle_name,NEW.emp_
last_name,NEW.emp_phone_number,NEW.emp_email,NEW.employment_status,NEW.emp_department,
NEW.emp_salary,NEW.emp_position);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`employees_deletes`;
CREATE TRIGGER `sjsu_movie_db`.`employees_deletes`
AFTER DELETE ON `sjsu_movie_db`.`employees`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_employees`
(`auditAction`,`Performed
By`,`emp_id`,`emp_ssn`,`emp_first_name`,`emp_middle_name`,`emp_last_name`,`emp_phone_number`,
`emp_email`,`employment_status`,`emp_department`,`emp_salary`,`emp_position`)
VALUES
('DELETE',user(),OLD.emp_id,OLD.emp_ssn,OLD.emp_first_name,OLD.emp_middle_name,OLD.emp_last_
name,OLD.emp_phone_number,OLD.emp_email,OLD.employment_status,OLD.emp_department,OLD.em
p_salary,OLD.emp_position);
```



```
DROP TABLE IF EXISTS `sjsu_movie_db`.`audit_employees`;
CREATE TABLE `sjsu_movie_db`.`audit_employees`
    (
    `auditAction` ENUM ('INSERT', 'UPDATE', 'DELETE'),
    `auditTimestamp` DATETIME DEFAULT CURRENT_TIMESTAMP,
    `auditId` INT AUTO_INCREMENT,
    `Performed By` varchar(200),
    `emp_id` varchar(15),
    `emp_ssn` varchar(12),
    `emp_first_name` varchar(40),
    `emp_middle_name` varchar(40),
    `emp_last_name` varchar(40),
    `emp_phone_number` varchar(24),
    `emp_email` varchar(60),
    `employment_status` varchar(40),
    `emp_department` varchar(30),
    `emp_salary` int,
    `emp_position` varchar(20),
    PRIMARY KEY (`auditId`),
    INDEX (`auditTimestamp`)
    );
```



```
DROP TRIGGER IF EXISTS `sjsu_movie_db`.`employees_inserts`;
CREATE TRIGGER `sjsu_movie_db`.`employees_inserts`
AFTER INSERT ON `sjsu_movie_db`.`employees`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_employees`
(`auditAction`,`Performed By`,`emp_id`,`emp_ssn`,`emp_first_name`,`emp_middle_name`,`emp_last_name`,`emp_phone_number`,`emp_email`,`employment_status`,`emp_department`,`emp_salary`,`emp_position`)
VALUES
('INSERT',user(),NEW.emp_id,NEW.emp_ssn,NEW.emp_first_name,NEW.emp_middle_name,NEW.emp_last_name,NEW.emp_phone_number,NEW.emp_email,NEW.employment_status,NEW.emp_department,NEW.emp_salary,NEW.emp_posit

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`employees_updates`;
CREATE TRIGGER `sjsu_movie_db`.`employees_updates`
AFTER UPDATE ON `sjsu_movie_db`.`employees`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_employees`
(`auditAction`,`Performed By`,`emp_id`,`emp_ssn`,`emp_first_name`,`emp_middle_name`,`emp_last_name`,`emp_phone_number`,`emp_email`,`employment_status`,`emp_department`,`emp_salary`,`emp_position`)
VALUES
('UPDATE',user(),NEW.emp_id,NEW.emp_ssn,NEW.emp_first_name,NEW.emp_middle_name,NEW.emp_last_name,NEW.emp_phone_number,NEW.emp_email,NEW.employment_status,NEW.emp_department,NEW.emp_salary,NEW.emp_posit

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`employees_deletes`;
CREATE TRIGGER `sjsu_movie_db`.`employees_deletes`
AFTER DELETE ON `sjsu_movie_db`.`employees`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_employees`
(`auditAction`,`Performed By`,`emp_id`,`emp_ssn`,`emp_first_name`,`emp_middle_name`,`emp_last_name`,`emp_phone_number`,`emp_email`,`employment_status`,`emp_department`,`emp_salary`,`emp_position`)
VALUES
('DELETE',user(),OLD.emp_id,OLD.emp_ssn,OLD.emp_first_name,OLD.emp_middle_name,OLD.emp_last_name,OLD.emp_phone_number,OLD.emp_email,OLD.employment_status,OLD.emp_department,OLD.emp_salary,OLD.emp_posit
```



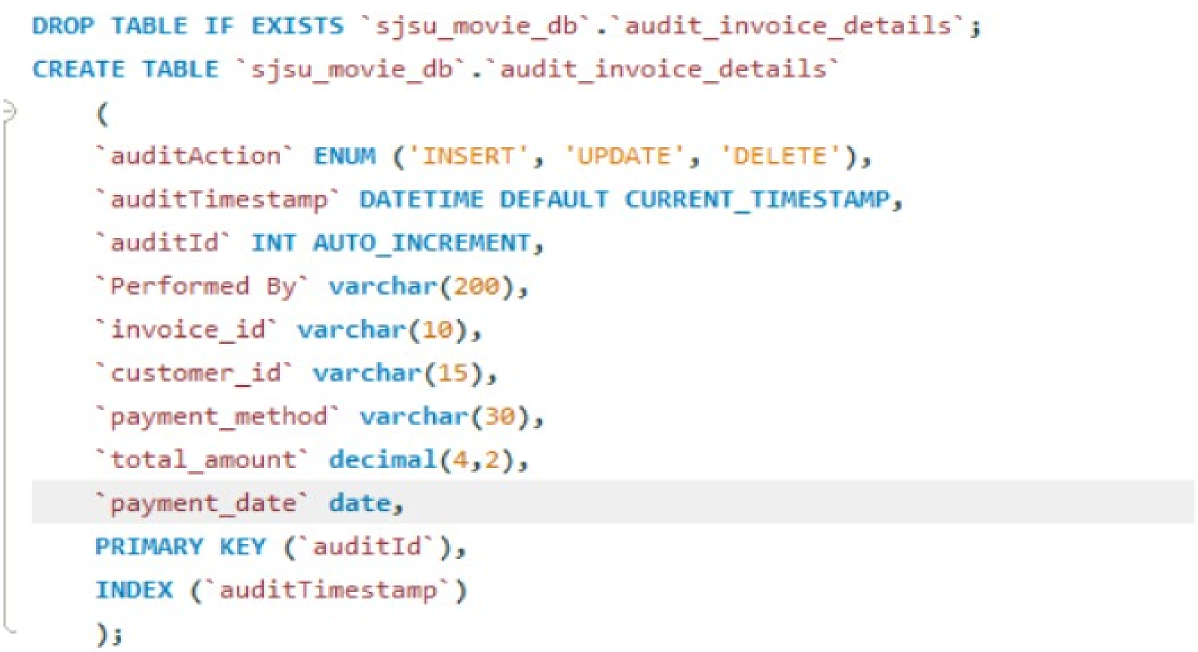
```
select * from audit_employees
```

auditAction	auditTimestamp	auditId	Performed By	emp_id	emp_ssn	emp_first_name	emp_middle_name	emp_last_name	emp_phone_number	emp_email
INSERT	2022-03-13 10:43:42	1	admin@c-73-63-213-219.hsd1.ca.comcast.net	643248497	700505090	Liebig	James	Ketsia	9085125266	cbrady@gonzalez-miller.
UPDATE	2022-03-13 10:43:54	2	admin@c-73-63-213-219.hsd1.ca.comcast.net	643248497	700505090	Liebig	James	Ketsia	9085125266	cbrady@gonzalez-miller.
DELETE	2022-03-13 10:43:54	3	admin@c-73-63-213-219.hsd1.ca.comcast.net	643248497	700505090	Liebig	James	Ketsia	9085125266	cbrady@gonzalez-miller.
INSERT	2022-03-13 10:44:06	4	admin@c-73-63-213-219.hsd1.ca.comcast.net	643248497	700505090	Liebig	James	Ketsia	9085125266	cbrady@gonzalez-miller.
UPDATE	2022-03-13 10:44:06	5	admin@c-73-63-213-219.hsd1.ca.comcast.net	643248497	700505090	Liebig	James	Ketsia	9085125266	cbrady@gonzalez-miller.
DELETE	2022-03-13 10:44:07	6	admin@c-73-63-213-219.hsd1.ca.comcast.net	643248497	700505090	Liebig	James	Ketsia	9085125266	cbrady@gonzalez-miller.
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

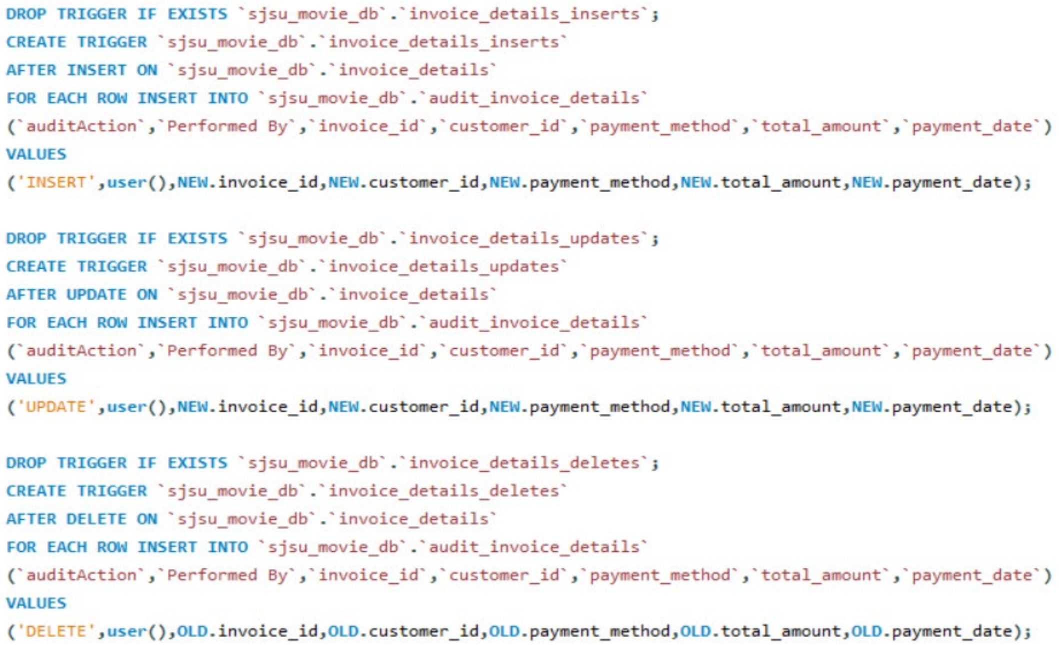
#	Time	Action	Message	Duration / Fetch
1	03:45:09	select * from audit_employees LIMIT 0, 1000	6 row(s) returned	0.078 sec / 0.000 sec

AUDIT INVOICE DETAILS:

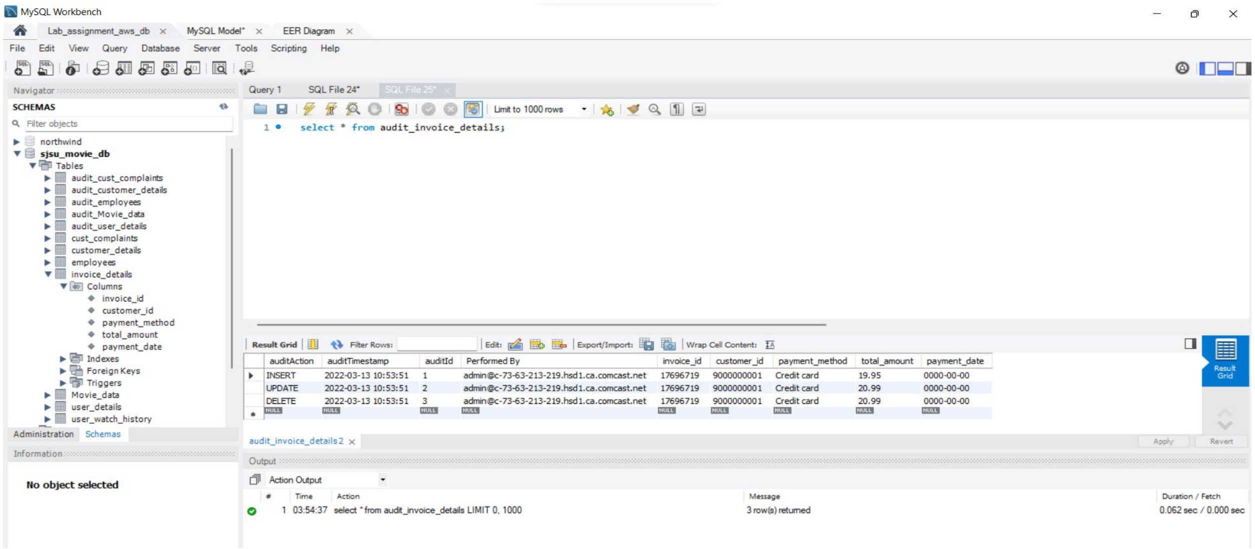
```
DROP TABLE IF EXISTS `sjsu_movie_db`.`audit_invoice_details`;
CREATE TABLE `sjsu_movie_db`.`audit_invoice_details`
(
`auditAction` ENUM ('INSERT', 'UPDATE', 'DELETE'),
`auditTimestamp` DATETIME DEFAULT CURRENT_TIMESTAMP,
`auditId` INT AUTO_INCREMENT,
`Performed By` varchar(200),
`invoice_id` varchar(10),
`customer_id` varchar(15),
`payment_method` varchar(30),
`total_amount` decimal(4,2),
`payment_date` date,
PRIMARY KEY (`auditId`),
INDEX (`auditTimestamp`)
);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`invoice_details_inserts`;
CREATE TRIGGER `sjsu_movie_db`.`invoice_details_inserts`
AFTER INSERT ON `sjsu_movie_db`.`invoice_details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_invoice_details`
(`auditAction`,`Performed
By`,`invoice_id`,`customer_id`,`payment_method`,`total_amount`,`payment_date`)
VALUES
('INSERT',user(),NEW.invoice_id,NEW.customer_id,NEW.payment_method,NEW.total_amount,NEW.pay
ment_date);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`invoice_details_updates`;
CREATE TRIGGER `sjsu_movie_db`.`invoice_details_updates`
AFTER UPDATE ON `sjsu_movie_db`.`invoice_details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_invoice_details`
(`auditAction`,`Performed
By`,`invoice_id`,`customer_id`,`payment_method`,`total_amount`,`payment_date`)
VALUES
('UPDATE',user(),NEW.invoice_id,NEW.customer_id,NEW.payment_method,NEW.total_amount,NEW.pay
ment_date);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`invoice_details_deletes`;
CREATE TRIGGER `sjsu_movie_db`.`invoice_details_deletes`
AFTER DELETE ON `sjsu_movie_db`.`invoice_details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_invoice_details`
(`auditAction`,`Performed
By`,`invoice_id`,`customer_id`,`payment_method`,`total_amount`,`payment_date`)
VALUES
('DELETE',user(),OLD.invoice_id,OLD.customer_id,OLD.payment_method,OLD.total_amount,OLD.paymen
t_date);
```



```sql
DROP TABLE IF EXISTS `sjsu_movie_db`.`audit_invoice_details`;
CREATE TABLE `sjsu_movie_db`.`audit_invoice_details`
    (
    `auditAction` ENUM ('INSERT', 'UPDATE', 'DELETE'),
    `auditTimestamp` DATETIME DEFAULT CURRENT_TIMESTAMP,
    `auditId` INT AUTO_INCREMENT,
    `Performed By` varchar(200),
    `invoice_id` varchar(10),
    `customer_id` varchar(15),
    `payment_method` varchar(30),
    `total_amount` decimal(4,2),
    `payment_date` date,
    PRIMARY KEY (`auditId`),
    INDEX (`auditTimestamp`)
    );
```



```sql
DROP TRIGGER IF EXISTS `sjsu_movie_db`.`invoice_details_inserts`;
CREATE TRIGGER `sjsu_movie_db`.`invoice_details_inserts`
AFTER INSERT ON `sjsu_movie_db`.`invoice_details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_invoice_details`
(`auditAction`,`Performed By`,`invoice_id`,`customer_id`,`payment_method`,`total_amount`,`payment_date`)
VALUES
('INSERT',user(),NEW.invoice_id,NEW.customer_id,NEW.payment_method,NEW.total_amount,NEW.payment_date);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`invoice_details_updates`;
CREATE TRIGGER `sjsu_movie_db`.`invoice_details_updates`
AFTER UPDATE ON `sjsu_movie_db`.`invoice_details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_invoice_details`
(`auditAction`,`Performed By`,`invoice_id`,`customer_id`,`payment_method`,`total_amount`,`payment_date`)
VALUES
('UPDATE',user(),NEW.invoice_id,NEW.customer_id,NEW.payment_method,NEW.total_amount,NEW.payment_date);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`invoice_details_deletes`;
CREATE TRIGGER `sjsu_movie_db`.`invoice_details_deletes`
AFTER DELETE ON `sjsu_movie_db`.`invoice_details`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_invoice_details`
(`auditAction`,`Performed By`,`invoice_id`,`customer_id`,`payment_method`,`total_amount`,`payment_date`)
VALUES
('DELETE',user(),OLD.invoice_id,OLD.customer_id,OLD.payment_method,OLD.total_amount,OLD.payment_date);
```



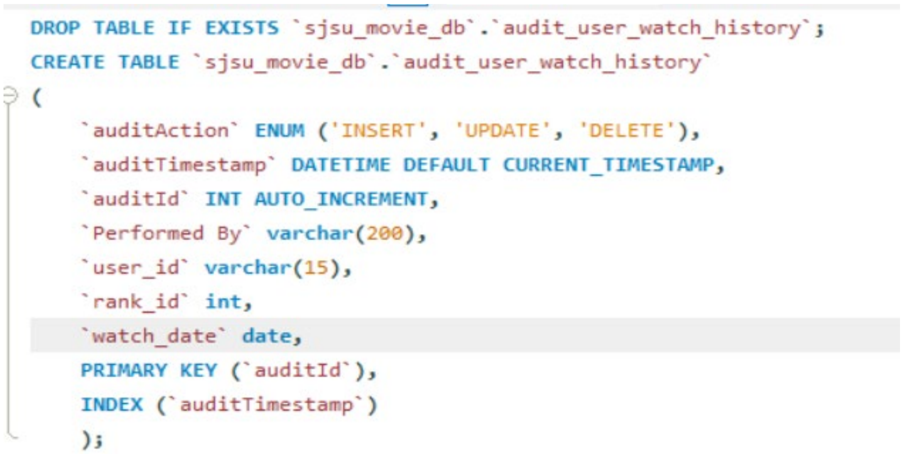
```sql
select * from audit_invoice_details;
```

auditAction	auditTimestamp	auditId	Performed By	invoice_id	customer_id	payment_method	total_amount	payment_date
INSERT	2022-03-13 10:53:51	1	admin@c-73-63-213-219.hsd1.ca.comcast.net	17696719	9000000001	Credit card	19.95	0000-00-00
UPDATE	2022-03-13 10:53:51	2	admin@c-73-63-213-219.hsd1.ca.comcast.net	17696719	9000000001	Credit card	20.99	0000-00-00
DELETE	2022-03-13 10:53:51	3	admin@c-73-63-213-219.hsd1.ca.comcast.net	17696719	9000000001	Credit card	20.99	0000-00-00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

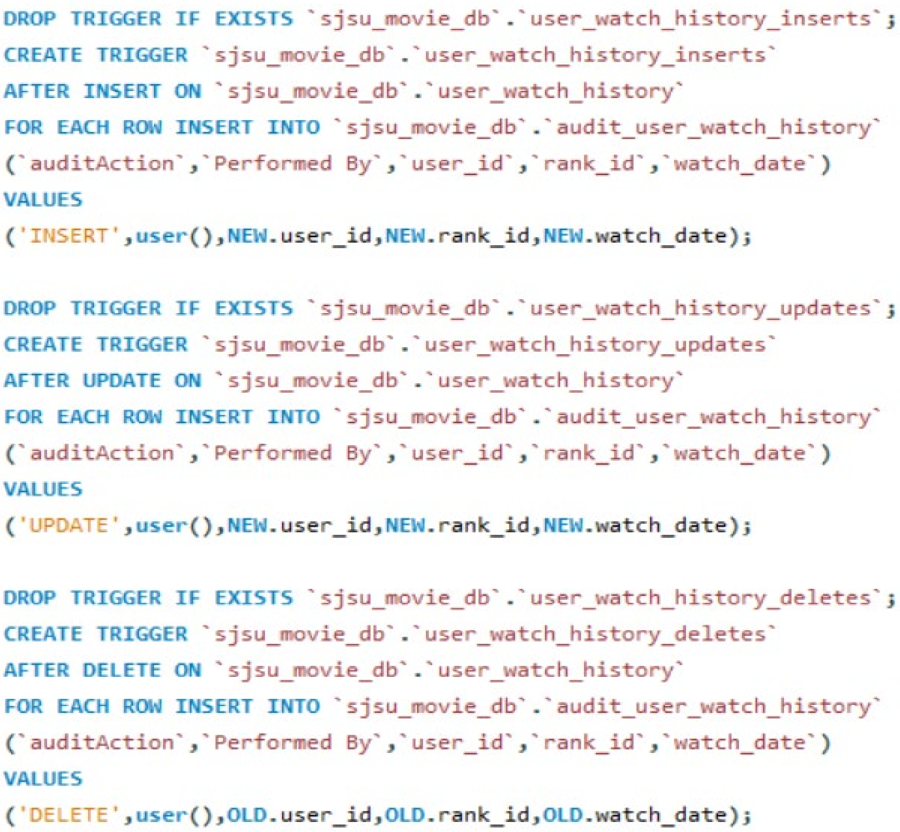
```
#  Time      Action                                           Message              Duration / Fetch
1  03:54:37  select * from audit_invoice_details LIMIT 0, 1000  3 row(s) returned  0.062 sec / 0.000 sec
```

AUDIT USER WATCH HISTORY:

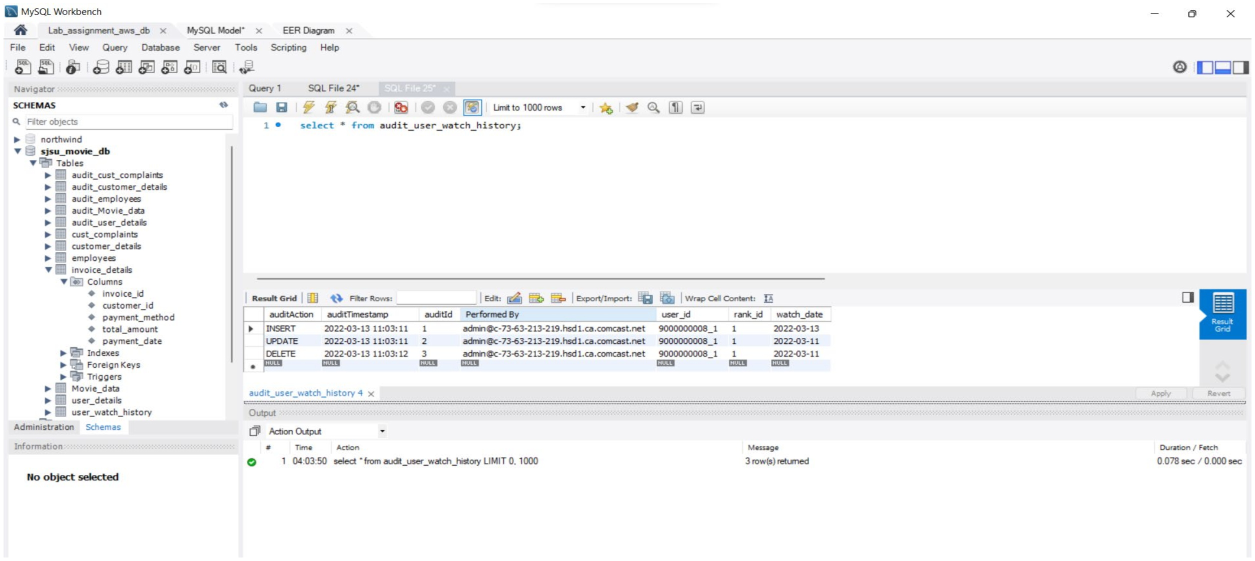
```sql
DROP TABLE IF EXISTS `sjsu_movie_db`.`audit_user_watch_history`;
CREATE TABLE `sjsu_movie_db`.`audit_user_watch_history`
(
`auditAction` ENUM ('INSERT', 'UPDATE', 'DELETE'),
`auditTimestamp` DATETIME DEFAULT CURRENT_TIMESTAMP,
`auditId` INT AUTO_INCREMENT,
`Performed By` varchar(200),
`user_id` varchar(15),
`rank_id` int,
`watch_date` date,
PRIMARY KEY (`auditId`),
INDEX (`auditTimestamp`)
);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`user_watch_history_inserts`;
CREATE TRIGGER `sjsu_movie_db`.`user_watch_history_inserts`
AFTER INSERT ON `sjsu_movie_db`.`user_watch_history`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_user_watch_history`
(`auditAction`,`Performed By`,`user_id`,`rank_id`,`watch_date`)
VALUES
('INSERT',user(),NEW.user_id,NEW.rank_id,NEW.watch_date);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`user_watch_history_updates`;
CREATE TRIGGER `sjsu_movie_db`.`user_watch_history_updates`
AFTER UPDATE ON `sjsu_movie_db`.`user_watch_history`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_user_watch_history`
(`auditAction`,`Performed By`,`user_id`,`rank_id`,`watch_date`)
VALUES
('UPDATE',user(),NEW.user_id,NEW.rank_id,NEW.watch_date);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`user_watch_history_deletes`;
CREATE TRIGGER `sjsu_movie_db`.`user_watch_history_deletes`
AFTER DELETE ON `sjsu_movie_db`.`user_watch_history`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_user_watch_history`
(`auditAction`,`Performed By`,`user_id`,`rank_id`,`watch_date`)
VALUES
('DELETE',user(),OLD.user_id,OLD.rank_id,OLD.watch_date);
```



```sql
DROP TABLE IF EXISTS `sjsu_movie_db`.`audit_user_watch_history`;
CREATE TABLE `sjsu_movie_db`.`audit_user_watch_history`
(
    `auditAction` ENUM ('INSERT', 'UPDATE', 'DELETE'),
    `auditTimestamp` DATETIME DEFAULT CURRENT_TIMESTAMP,
    `auditId` INT AUTO_INCREMENT,
    `Performed By` varchar(200),
    `user_id` varchar(15),
    `rank_id` int,
    `watch_date` date,
    PRIMARY KEY (`auditId`),
    INDEX (`auditTimestamp`)
    );
```



```sql
DROP TRIGGER IF EXISTS `sjsu_movie_db`.`user_watch_history_inserts`;
CREATE TRIGGER `sjsu_movie_db`.`user_watch_history_inserts`
AFTER INSERT ON `sjsu_movie_db`.`user_watch_history`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_user_watch_history`
(`auditAction`,`Performed By`,`user_id`,`rank_id`,`watch_date`)
VALUES
('INSERT',user(),NEW.user_id,NEW.rank_id,NEW.watch_date);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`user_watch_history_updates`;
CREATE TRIGGER `sjsu_movie_db`.`user_watch_history_updates`
AFTER UPDATE ON `sjsu_movie_db`.`user_watch_history`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_user_watch_history`
(`auditAction`,`Performed By`,`user_id`,`rank_id`,`watch_date`)
VALUES
('UPDATE',user(),NEW.user_id,NEW.rank_id,NEW.watch_date);

DROP TRIGGER IF EXISTS `sjsu_movie_db`.`user_watch_history_deletes`;
CREATE TRIGGER `sjsu_movie_db`.`user_watch_history_deletes`
AFTER DELETE ON `sjsu_movie_db`.`user_watch_history`
FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_user_watch_history`
(`auditAction`,`Performed By`,`user_id`,`rank_id`,`watch_date`)
VALUES
('DELETE',user(),OLD.user_id,OLD.rank_id,OLD.watch_date);
```



```sql
select * from audit_user_watch_history;
```

auditAction	auditTimestamp	auditId	Performed By	user_id	rank_id	watch_date
INSERT	2022-03-13 11:03:11	1	admin@c-73-63-213-219.hsd1.ca.comcast.net	9000000008_1	1	2022-03-13
UPDATE	2022-03-13 11:03:11	2	admin@c-73-63-213-219.hsd1.ca.comcast.net	9000000008_1	1	2022-03-11
DELETE	2022-03-13 11:03:12	3	admin@c-73-63-213-219.hsd1.ca.comcast.net	9000000008_1	1	2022-03-11
NULL	NULL	NULL	NULL	NULL	NULL	NULL

#	Time	Action	Message	Duration / Fetch
1	04:03:50	select * from audit_user_watch_history LIMIT 0, 1000	3 row(s) returned	0.078 sec / 0.000 sec

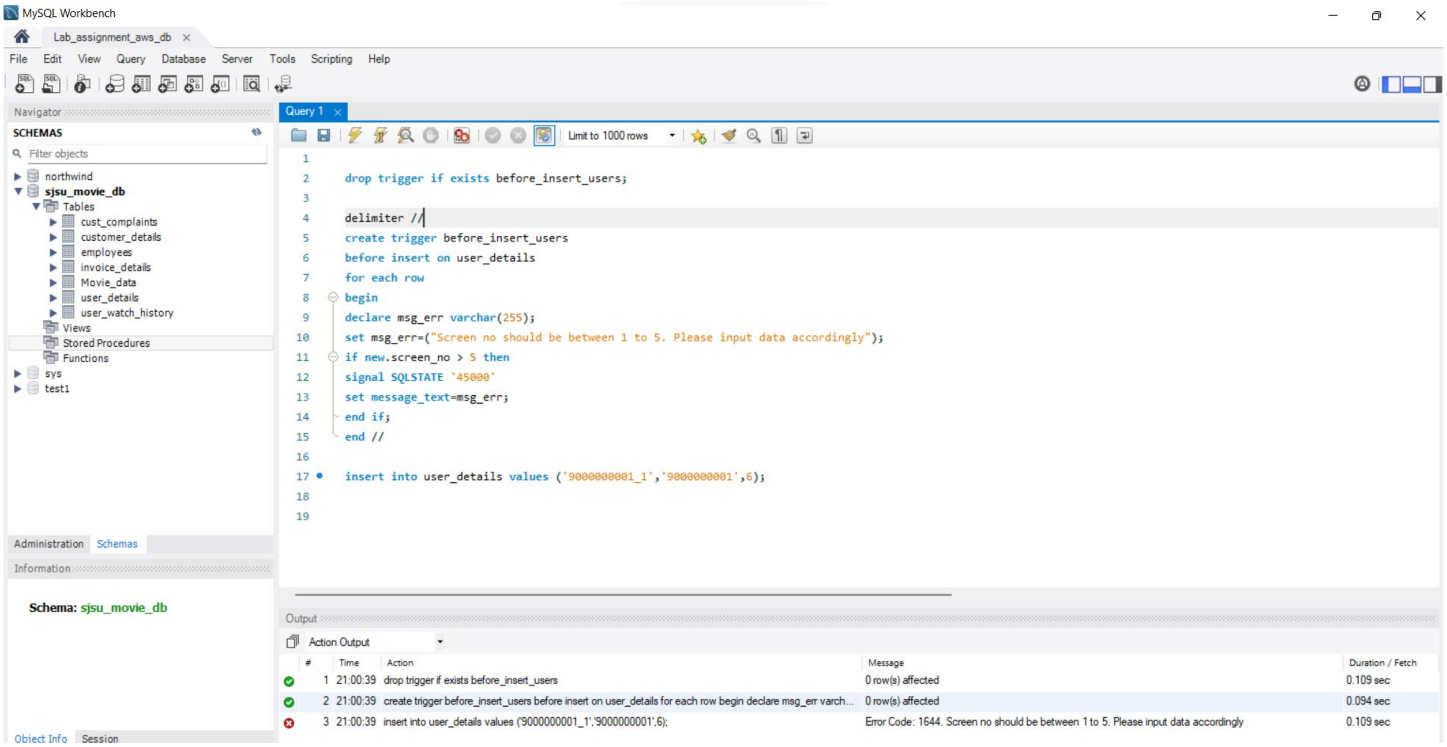
OTHER GENERAL TRIGGERS

TRIGGER FOR SCREEN NUMBER:

PURPOSE:

Total no of screens per customer is 5. In case if any record with screen no more than 5 is entered, this trigger will catch it and print an error message.

```
drop trigger if exists before_insert_users;

delimiter //
create trigger before_insert_users
before insert on user_details
for each row
begin
declare msg_err varchar(255);
set msg_err=("Screen no should be between 1 to 5. Please input data accordingly");
if new.screen_no > 5 then
signal SQLSTATE '45000'
set message_text=msg_err;
end if;
end //

insert into user_details values ('9000000001_1','9000000001',6);
```



```

drop trigger if exists before_insert_users;

delimiter //
create trigger before_insert_users
before insert on user_details
for each row
begin
declare msg_err varchar(255);
set msg_err=("Screen no should be between 1 to 5. Please input data accordingly");
if new.screen_no > 5 then
signal SQLSTATE '45000'
set message_text=msg_err;
end if;
end //

insert into user_details values ('9000000001_1','9000000001',6);


```

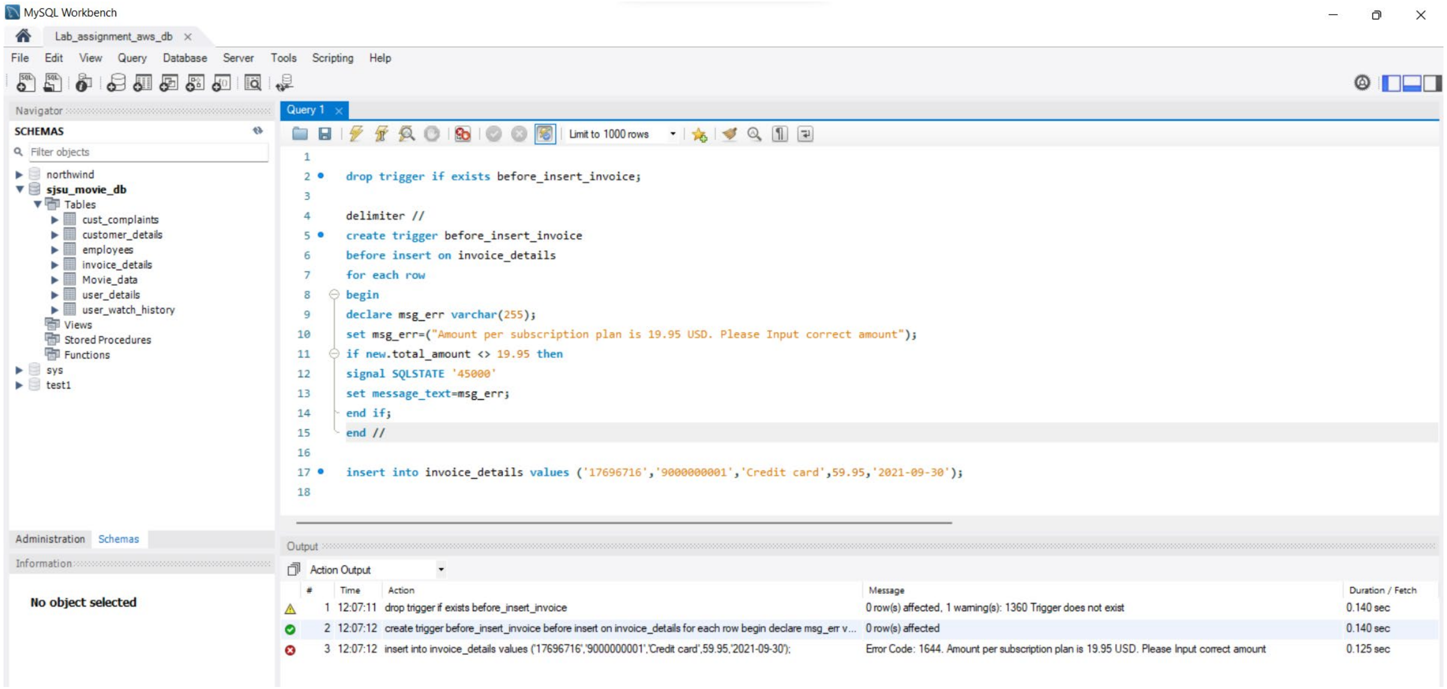
#	Time	Action	Message	Duration / Fetch
1	21:00:39	drop trigger if exists before_insert_users	0 row(s) affected	0.109 sec
2	21:00:39	create trigger before_insert_users before insert on user_details for each row begin declare msg_err varch...	0 row(s) affected	0.094 sec
3	21:00:39	insert into user_details values ('9000000001_1','9000000001',6);	Error Code: 1644. Screen no should be between 1 to 5. Please input data accordingly	0.109 sec

TRIGGER TO CHECK THE SUBSCRIPITION AMOUNT:

PURPOSE:

Amount per subscription plan is $19.95. If any record is inserted with different amount, this trigger will catch it and print an error message.

```
drop trigger if exists before_insert_invoice;

delimiter //
create trigger before_insert_invoice
before insert on invoice_details
for each row
begin
declare msg_err varchar(255);
set msg_err=("Amount per subscription plan is 19.95 USD. Please Input correct amount");
if new.total_amount <> 19.95 then
signal SQLSTATE '45000'
set message_text=msg_err;
end if;
end //

insert into invoice_details values ('17696716','9000000001','Credit card',59.95,'2021-09-30');
```



```
drop trigger if exists before_insert_invoice;

delimiter //
create trigger before_insert_invoice
before insert on invoice_details
for each row
begin
declare msg_err varchar(255);
set msg_err=("Amount per subscription plan is 19.95 USD. Please Input correct amount");
if new.total_amount <> 19.95 then
signal SQLSTATE '45000'
set message_text=msg_err;
end if;
end //

insert into invoice_details values ('17696716','9000000001','Credit card',59.95,'2021-09-30');
```

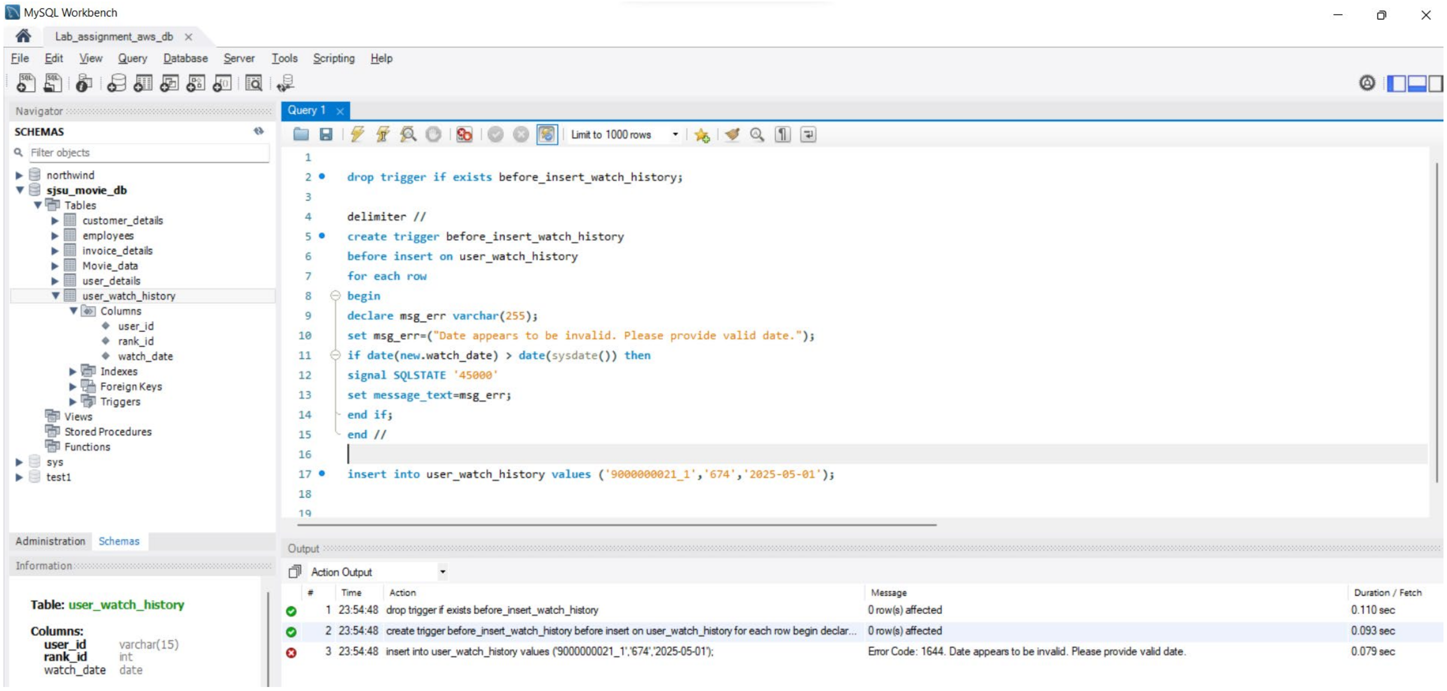
	#	Time	Action	Message	Duration / Fetch
⚠	1	12:07:11	drop trigger if exists before_insert_invoice	0 row(s) affected, 1 warning(s): 1360 Trigger does not exist	0.140 sec
✓	2	12:07:12	create trigger before_insert_invoice before insert on invoice_details for each row begin declare msg_err v...	0 row(s) affected	0.140 sec
✗	3	12:07:12	insert into invoice_details values ('17696716','9000000001','Credit card',59.95,'2021-09-30');	Error Code: 1644. Amount per subscription plan is 19.95 USD. Please Input correct amount	0.125 sec

TRIGGER TO CHECK THE DATE OF USER WATCH HISTORY:

PURPOSE:

This trigger checks if any date for the watch history is wrongly entered. If so, it will display an error message

```
drop trigger if exists before_insert_watch_history;

delimiter //
create trigger before_insert_watch_history
before insert on user_watch_history
for each row
begin
declare msg_err varchar(255);
set msg_err=("Date appears to be invalid. Please provide valid date.");
if date(new.watch_date) > date(sysdate()) then
signal SQLSTATE '45000'
set message_text=msg_err;
end if;
end //

insert into user_watch_history values ('9000000021_1','674','2025-05-01');
```



```
drop trigger if exists before_insert_watch_history;

delimiter //
create trigger before_insert_watch_history
before insert on user_watch_history
for each row
begin
declare msg_err varchar(255);
set msg_err=("Date appears to be invalid. Please provide valid date.");
if date(new.watch_date) > date(sysdate()) then
signal SQLSTATE '45000'
set message_text=msg_err;
end if;
end //

insert into user_watch_history values ('9000000021_1','674','2025-05-01');
```

Table: user_watch_history

Columns:

user_id	varchar(15)
rank_id	int
watch_date	date

	#	Time	Action	Message	Duration / Fetch
✓	1	23:54:48	drop trigger if exists before_insert_watch_history	0 row(s) affected	0.110 sec
✓	2	23:54:48	create trigger before_insert_watch_history before insert on user_watch_history for each row begin declar...	0 row(s) affected	0.093 sec
✗	3	23:54:48	insert into user_watch_history values ('9000000021_1','674','2025-05-01');	Error Code: 1644. Date appears to be invalid. Please provide valid date.	0.079 sec

TRIGGER TO CHECK THE VALIDITY OF DATES IN CUSTOMER COMPLAINTS:

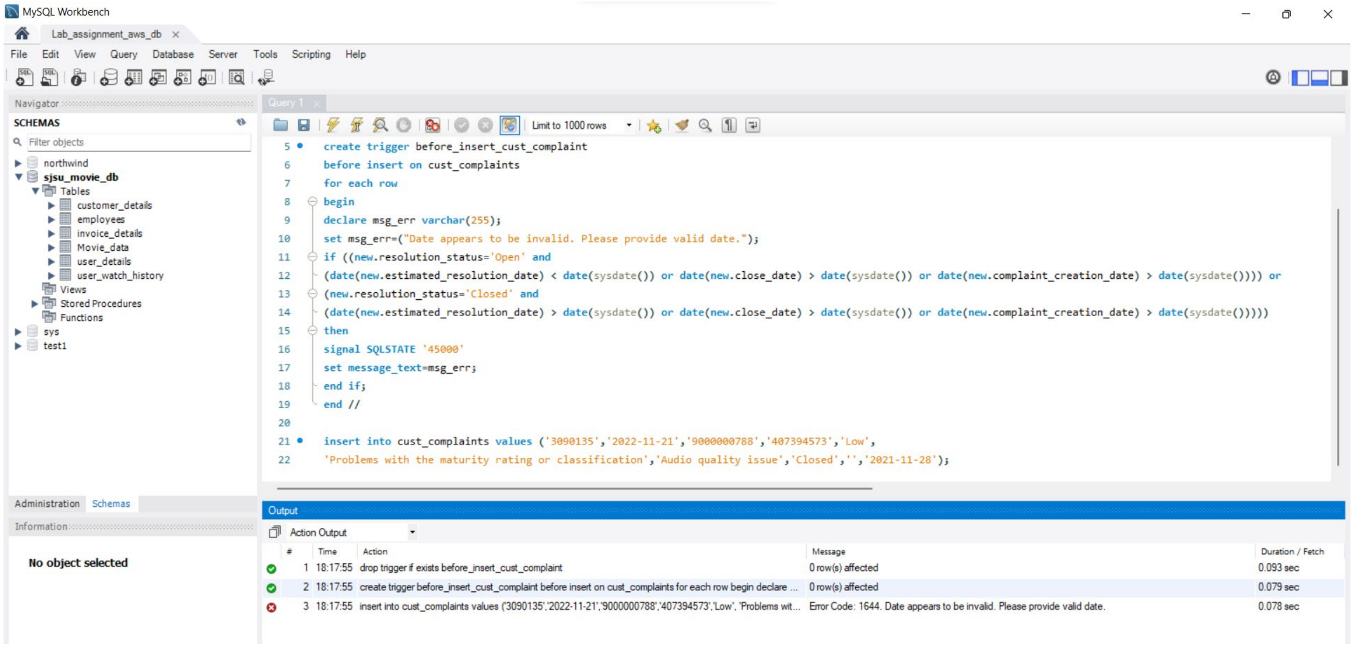
PURPOSE:

This trigger checks the following scenarios,

1. For an open ticket, the estimated resolution date should be in future
2. For closed tickets, the creation date, estimated closure and closure date should not be greater than the current date

If any data fails these scenarios, then an error message will be printed.

```
drop trigger if exists before_insert_cust_complaint;

delimiter //
create trigger before_insert_cust_complaint
before insert on cust_complaints
for each row
begin
declare msg_err varchar(255);
set msg_err=("Date appears to be invalid. Please provide valid date.");
if ((new.resolution_status='Open' and
(date(new.estimated_resolution_date) < date(sysdate()) or date(new.close_date) > date(sysdate()) or
date(new.complaint_creation_date) > date(sysdate()))) or
(new.resolution_status='Closed' and
(date(new.estimated_resolution_date) > date(sysdate()) or date(new.close_date) > date(sysdate()) or
date(new.complaint_creation_date) > date(sysdate()))))
then
signal SQLSTATE '45000'
set message_text=msg_err;
end if;
end //

insert into cust_complaints values ('3090135','2022-11-21','9000000788','407394573','Low',
'Problems with the maturity rating or classification','Audio quality issue','Closed','','2021-11-28');
```



```
create trigger before_insert_cust_complaint
before insert on cust_complaints
for each row
begin
declare msg_err varchar(255);
set msg_err=("Date appears to be invalid. Please provide valid date.");
if ((new.resolution_status='Open' and
(date(new.estimated_resolution_date) < date(sysdate()) or date(new.close_date) > date(sysdate()) or date(new.complaint_creation_date) > date(sysdate()))) or
(new.resolution_status='Closed' and
(date(new.estimated_resolution_date) > date(sysdate()) or date(new.close_date) > date(sysdate()) or date(new.complaint_creation_date) > date(sysdate()))))
then
signal SQLSTATE '45000'
set message_text=msg_err;
end if;
end //

insert into cust_complaints values ('3090135','2022-11-21','9000000788','407394573','Low',
'Problems with the maturity rating or classification','Audio quality issue','Closed','','2021-11-28');
```

```
#  Time      Action                                                                                          Message                                                          Duration / Fetch
1  18:17:55  drop trigger if exists before_insert_cust_complaint                                              0 row(s) affected                                                0.093 sec
2  18:17:55  create trigger before_insert_cust_complaint before insert on cust_complaints for each row begin declare ...  0 row(s) affected                                  0.079 sec
3  18:17:55  insert into cust_complaints values ('3090135','2022-11-21','9000000788','407394573','Low', 'Problems wit...  Error Code: 1644. Date appears to be invalid. Please provide valid date.  0.078 sec
```

TOTAL TRIGGERS CREATED

Trigger	Event	Table	Statement	Timing	Created	sql_mode	Definer	character_set_client	collation_connection	Database Collation
Actor_Details_inserts	INSERT	Actor_Details	INSERT INTO `sjsu_movi	AFTER	2022-03-22 02:20:17.77	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
Actor_Details_updates	UPDATE	Actor_Details	INSERT INTO `sjsu_movi	AFTER	2022-03-22 02:20:17.97	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
Actor_Details_deletes	DELETE	Actor_Details	INSERT INTO `sjsu_movi	AFTER	2022-03-22 02:20:18.15	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
Genre_Details_inserts	INSERT	Genre_Details	INSERT INTO `sjsu_movi	AFTER	2022-03-22 03:40:31.40	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
Genre_Details_updates	UPDATE	Genre_Details	INSERT INTO `sjsu_movi	AFTER	2022-03-22 03:40:31.59	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
Genre_Details_deletes	DELETE	Genre_Details	INSERT INTO `sjsu_movi	AFTER	2022-03-22 03:40:31.78	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
Movie_data_inserts	INSERT	Movie_data	INSERT INTO `sjsu_movi	AFTER	2022-03-22 02:18:16.98	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
Movie_data_updates	UPDATE	Movie_data	INSERT INTO `sjsu_movi	AFTER	2022-03-22 02:18:17.16	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
Movie_data_deletes	DELETE	Movie_data	INSERT INTO `sjsu_movi	AFTER	2022-03-22 02:18:17.34	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
before_insert_cust_com	INSERT	cust_complaints	begin declare msg_err va	BEFORE	2022-03-13 02:17:55.89	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
cust_complaints_inserts	INSERT	cust_complaints	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:16:39.16	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
cust_complaints_update	UPDATE	cust_complaints	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:16:39.34	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
cust_complaints_deletes	DELETE	cust_complaints	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:16:39.52	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
customer_details_inserts	INSERT	customer_details	INSERT INTO `sjsu_movi	AFTER	2022-03-13 09:57:35.86	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
customer_details_updat	UPDATE	customer_details	INSERT INTO `sjsu_movi	AFTER	2022-03-13 09:57:36.04	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
customer_details_delete	DELETE	customer_details	INSERT INTO `sjsu_movi	AFTER	2022-03-13 09:57:36.22	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
employees_inserts	INSERT	employees	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:40:40.51	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
employees_updates	UPDATE	employees	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:40:40.72	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
employees_deletes	DELETE	employees	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:40:40.91	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
before_insert_invoice	INSERT	invoice_details	begin declare msg_err va	BEFORE	2022-03-09 20:07:12.09	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
invoice_details_inserts	INSERT	invoice_details	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:50:49.08	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
invoice_details_updates	UPDATE	invoice_details	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:50:49.26	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
invoice_details_deletes	DELETE	invoice_details	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:50:49.44	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
before_insert_users	INSERT	user_details	begin declare msg_err va	BEFORE	2022-03-09 06:10:02.57	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
user_details_inserts	INSERT	user_details	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:27:16.79	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
user_details_updates	UPDATE	user_details	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:27:16.97	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
user_details_deletes	DELETE	user_details	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:27:17.15	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
before_insert_watch_his	INSERT	user_watch_history	begin declare msg_err va	BEFORE	2022-03-09 07:54:50.31	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
user_watch_history_inse	INSERT	user_watch_history	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:59:58.05	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
user_watch_history_upd	UPDATE	user_watch_history	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:59:58.24	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
user_watch_history_dele	DELETE	user_watch_history	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:59:58.41	NO_ENGINE_SUBSTITUT	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci

STORED PROCEDURES

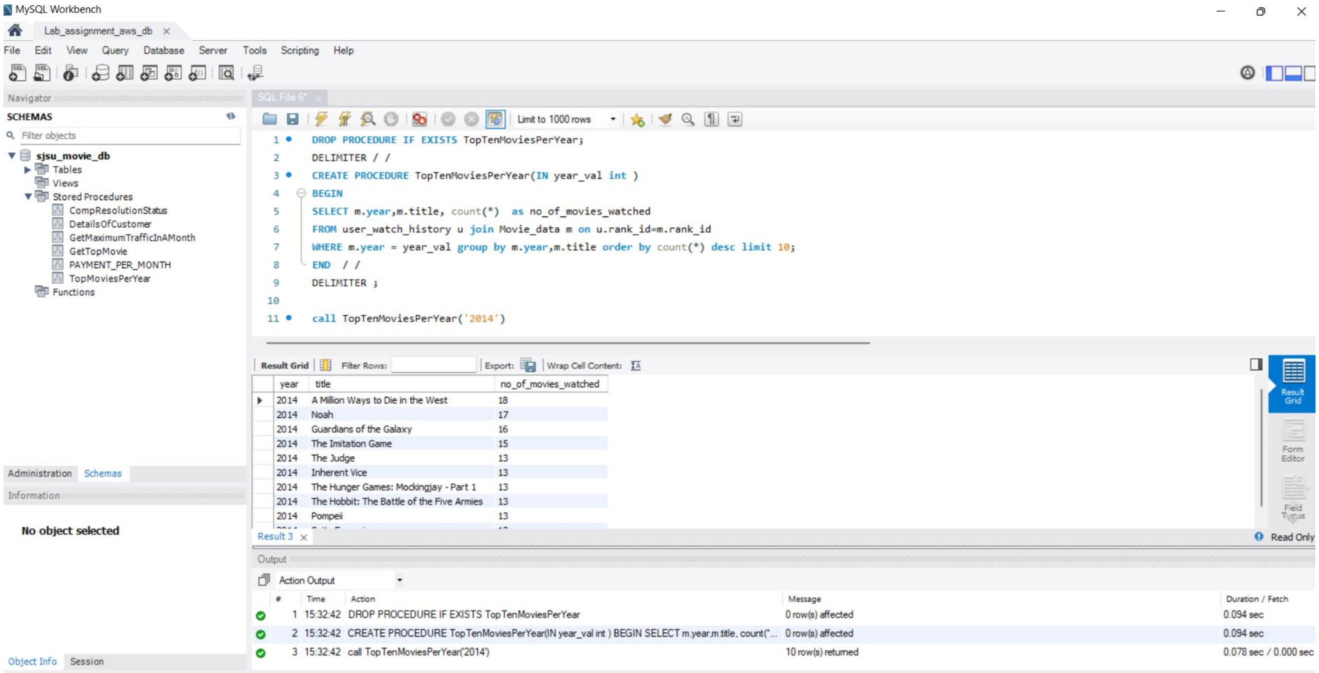
TopTenMoviesPerYear:

SQL QUERY

```
DROP PROCEDURE IF EXISTS TopTenMoviesPerYear;
DELIMITER / /
CREATE PROCEDURE TopTenMoviesPerYear (IN year_val int )
BEGIN
SELECT m.year,m.title, count(*)  as no_of_movies_watched
FROM user_watch_history u join Movie_data m on u.rank_id=m.rank_id
WHERE m.year = year_val group by m.year,m.title order by count(*) desc limit 10;
END  / /
DELIMITER ;

call TopTenMoviesPerYear('2014')
```

EMBEDDED SQL

```
from mysql.connector import connect, Error
try:
with connect(host="lab-assignment-225.cibzfcia066j.us-east-1.rds.amazonaws.com",
user=usrnm, password=pwd, database="sjsu_movie_db"
) as connection:
a=input("Enter the year (2006 - 2016) for which top 10 movies need to be displayed:\t")
sp3 = "call TopTenMoviesPerYear('{}')".format(a)
with connection.cursor() as cursp3:
cursp3.execute(sp3)
sp3_result = pd. DataFrame (cursp3. fetchall ())
except Error as e:
print(e)
sp3_result.columns = ['Year','Title','No of times watched']
sp3_result
```



```sql
DROP PROCEDURE IF EXISTS TopTenMoviesPerYear;
DELIMITER / /
CREATE PROCEDURE TopTenMoviesPerYear(IN year_val int )
BEGIN
SELECT m.year,m.title, count(*)  as no_of_movies_watched
FROM user_watch_history u join Movie_data m on u.rank_id=m.rank_id
WHERE m.year = year_val group by m.year,m.title order by count(*) desc limit 10;
END  / /
DELIMITER ;

call TopTenMoviesPerYear('2014')
```

year	title	no_of_movies_watched
2014	A Million Ways to Die in the West	18
2014	Noah	17
2014	Guardians of the Galaxy	16
2014	The Imitation Game	15
2014	The Judge	13
2014	Inherent Vice	13
2014	The Hunger Games: Mockingjay - Part 1	13
2014	The Hobbit: The Battle of the Five Armies	13
2014	Pompeii	13

#	Time	Action	Message	Duration / Fetch
1	15:32:42	DROP PROCEDURE IF EXISTS TopTenMoviesPerYear	0 row(s) affected	0.094 sec
2	15:32:42	CREATE PROCEDURE TopTenMoviesPerYear(IN year_val int) BEGIN SELECT m.year,m.title, count(*...	0 row(s) affected	0.094 sec
3	15:32:42	call TopTenMoviesPerYear('2014')	10 row(s) returned	0.078 sec / 0.000 sec

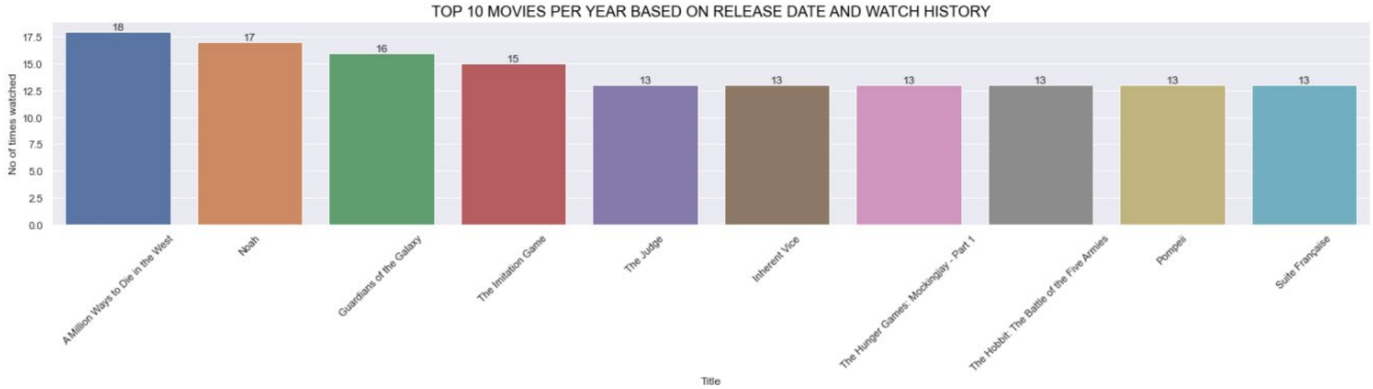
Display top 10 movies and No of times it is watched for the 'year' read as input

```python
from mysql.connector import connect, Error

try:
    with connect(host="lab-assignment-225.cibzfcia066j.us-east-1.rds.amazonaws.com",
                 user=usrnm,
                 password=pwd,
                 database="sjsu_movie_db"
    ) as connection:
        a=input("Enter the year (2006 - 2016) for which top 10 movies need to be displayed:\t")
        sp3 = "call TopTenMoviesPerYear('{}')".format(a)
        with connection.cursor() as cursp3:
            cursp3.execute(sp3)
            sp3_result = pd. DataFrame (cursp3. fetchall ())
except Error as e:
    print(e)

sp3_result.columns = ['Year','Title','No of times watched']
sp3_result
```

```
Enter the year (2006 - 2016) for which top 10 movies need to be displayed:      2009
```

0]:

	Year	Title	No of times watched
0	2009	Pandorum	13
1	2009	The Ugly Truth	13
2	2009	2012	12
3	2009	Watchmen	12
4	2009	Avatar	11
5	2009	Up	11
6	2009	The Lovely Bones	11
7	2009	Terminator Salvation	11
8	2009	Underworld: Rise of the Lycans	10
9	2009	Harry Potter and the Half-Blood Prince	10



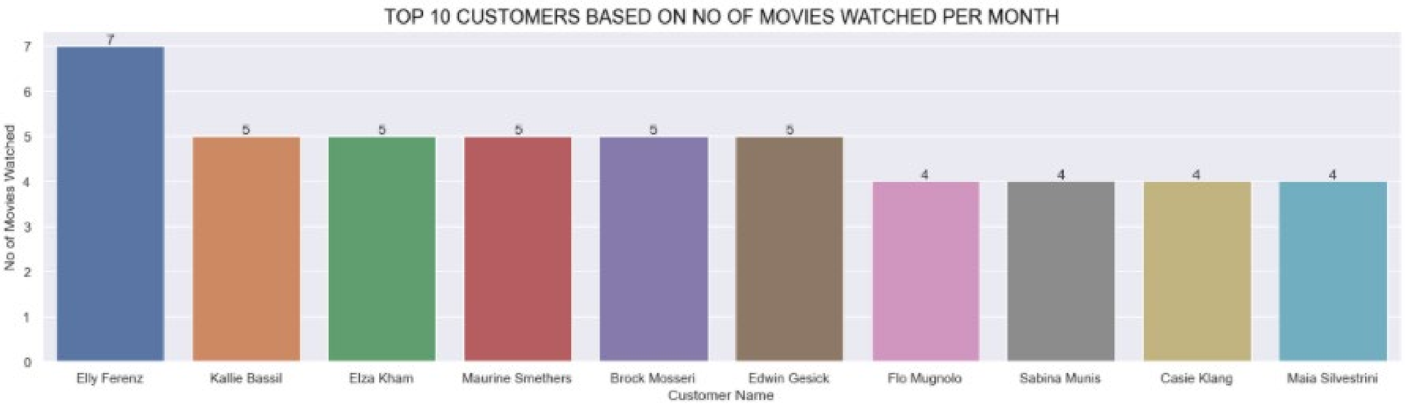
GetMaximumTrafficInAMonth:

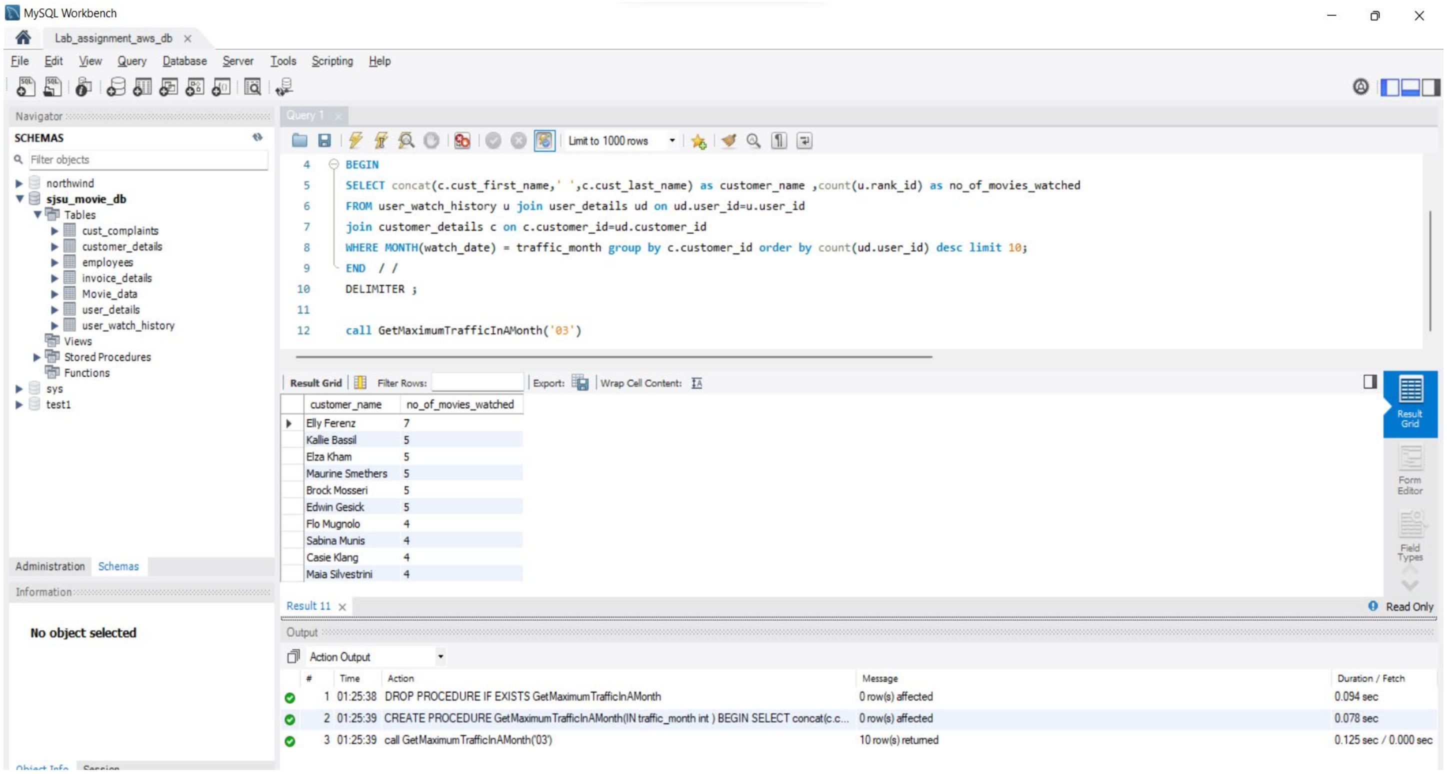
SQL QUERY

```
DROP PROCEDURE IF EXISTS GetMaximumTrafficInAMonth;
DELIMITER / /
CREATE PROCEDURE GetMaximumTrafficInAMonth(IN traffic_month int )
BEGIN
SELECT concat(c.cust_first_name,' ',c.cust_last_name) as customer_name ,count(u.rank_id) as
no_of_movies_watched
FROM user_watch_history u join user_details ud on ud.user_id=u.user_id
join customer_details c on c.customer_id=ud.customer_id
WHERE MONTH(watch_date) = traffic_month group by c.customer_id order by count(ud.user_id) desc;
END  / /
DELIMITER ;

call GetMaximumTrafficInAMonth('01')
```

EMBEDDED SQL

```
from mysql.connector import connect, Error

try:
with connect(host="lab-assignment-225.cibzfcia066j.us-east-1.rds.amazonaws.com",
user=usrnm,
password=pwd,
database="sjsu_movie_db"
) as connection:
a=input("Enter Month number:\t")
sp2 = "call GetMaximumTrafficInAMonth('{}')".format(a)
with connection.cursor() as cursp2:
cursp2.execute(sp2)
sp2_result = pd. DataFrame (cursp2. fetchall ())
except Error as e:
print(e)

sp2_result.columns = ['Customer Name','No of Movies Watched']
sp2_result
```





```sql
BEGIN
SELECT concat(c.cust_first_name,' ',c.cust_last_name) as customer_name ,count(u.rank_id) as no_of_movies_watched
FROM user_watch_history u join user_details ud on ud.user_id=u.user_id
join customer_details c on c.customer_id=ud.customer_id
WHERE MONTH(watch_date) = traffic_month group by c.customer_id order by count(ud.user_id) desc limit 10;
END  / /
DELIMITER ;

call GetMaximumTrafficInAMonth('03')
```

customer_name	no_of_movies_watched
Elly Ferenz	7
Kallie Bassil	5
Elza Kham	5
Maurine Smethers	5
Brock Mosseri	5
Edwin Gesick	5
Flo Mugnolo	4
Sabina Munis	4
Casie Klang	4
Maia Silvestrini	4

#	Time	Action	Message	Duration / Fetch
1	01:25:38	DROP PROCEDURE IF EXISTS GetMaximumTrafficInAMonth	0 row(s) affected	0.094 sec
2	01:25:39	CREATE PROCEDURE GetMaximumTrafficInAMonth(IN traffic_month int) BEGIN SELECT concat(c.c...	0 row(s) affected	0.078 sec
3	01:25:39	call GetMaximumTrafficInAMonth('03')	10 row(s) returned	0.125 sec / 0.000 sec

Display the TOP 10 Customers who watched most movies on the month read as Input

```python
from mysql.connector import connect, Error

try:
    with connect(host="lab-assignment-225.cibzfcia066j.us-east-1.rds.amazonaws.com",
                 user=usrnm,
                 password=pwd,
                 database="sjsu_movie_db"
    ) as connection:
        a=input("Enter Month number:\t")
        sp2 = "call GetMaximumTrafficInAMonth('{}')".format(a)
        with connection.cursor() as cursp2:
            cursp2.execute(sp2)
            sp2_result = pd. DataFrame (cursp2. fetchall ())
except Error as e:
    print(e)

sp2_result.columns = ['Customer Name','No of Movies Watched']
sp2_result
```

```
Enter Month number:	3
```

3]:

	Customer Name	No of Movies Watched
0	Elly Ferenz	7
1	Kallie Bassil	5
2	Elza Kham	5
3	Maurine Smethers	5
4	Brock Mosseri	5
5	Edwin Gesick	5
6	Flo Mugnolo	4
7	Sabina Munis	4
8	Casie Klang	4
9	Maia Silvestrini	4

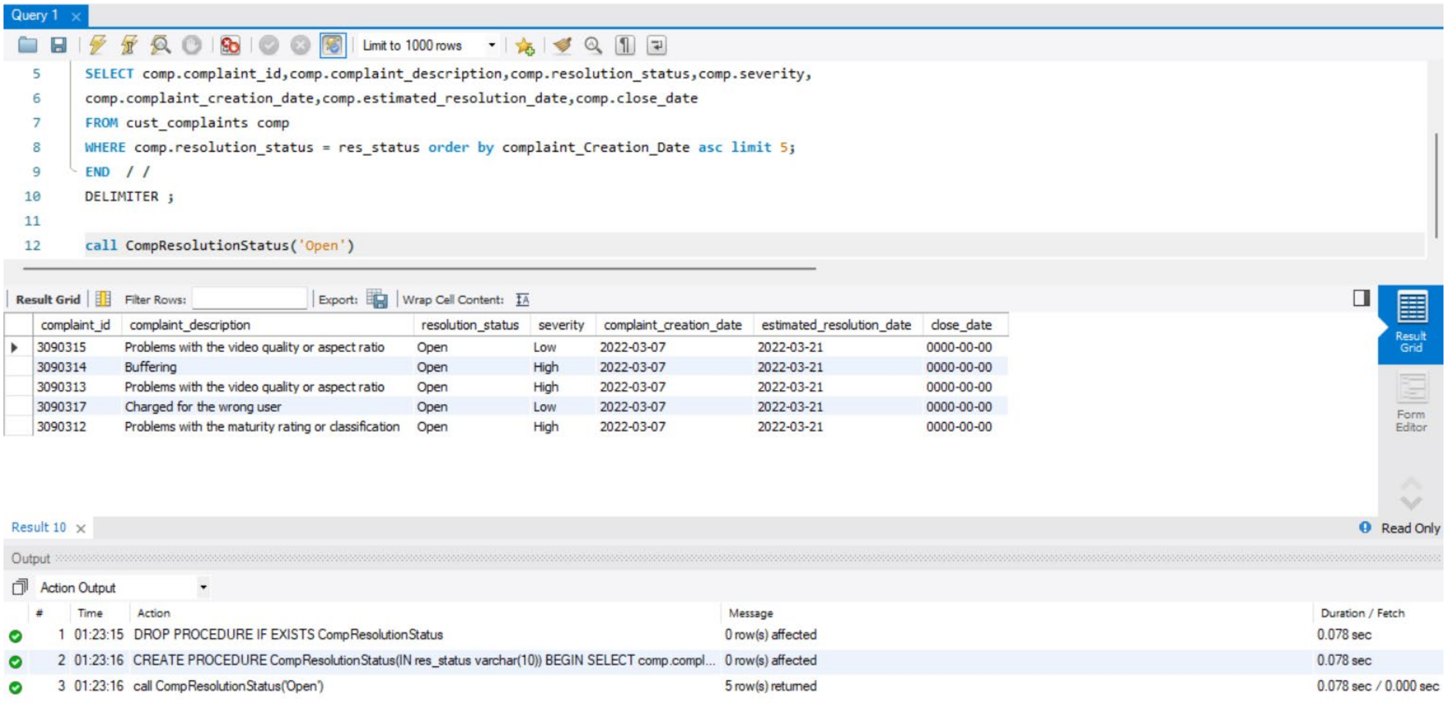
CompResolutionStatus:

SQL QUERY

```
DROP PROCEDURE IF EXISTS CompResolutionStatus;
DELIMITER / /
CREATE PROCEDURE CompResolutionStatus(IN res_status varchar(10))
BEGIN
SELECT comp.complaint_id,comp.complaint_description,comp.resolution_status,comp.severity,
comp.complaint_creation_date,comp.estimated_resolution_date,comp.close_date
FROM cust_complaints comp
WHERE comp.resolution_status = res_status order by complaint_Creation_Date asc limit 5;
END  / /
DELIMITER ;

call CompResolutionStatus('Closed')
```

EMBEDDED SQL

```
from mysql.connector import connect, Error

try:
with connect(host="lab-assignment-225.cibzfcia066j.us-east-1.rds.amazonaws.com",
user=usrnm,
password=pwd,
database="sjsu_movie_db"
) as connection:
sp1 = "call CompResolutionStatus('Open')"
with connection.cursor() as cursp1:
cursp1.execute(sp1)
sp1_result = pd. DataFrame (cursp1. fetchall ())
except Error as e:
print(e)

sp1_result.columns = ['Complaint ID','Complaint Description','Resolution Status','Severity',
'Complaint Creation Date','Estimated Resolution Date','Close Date']
sp1_result
```

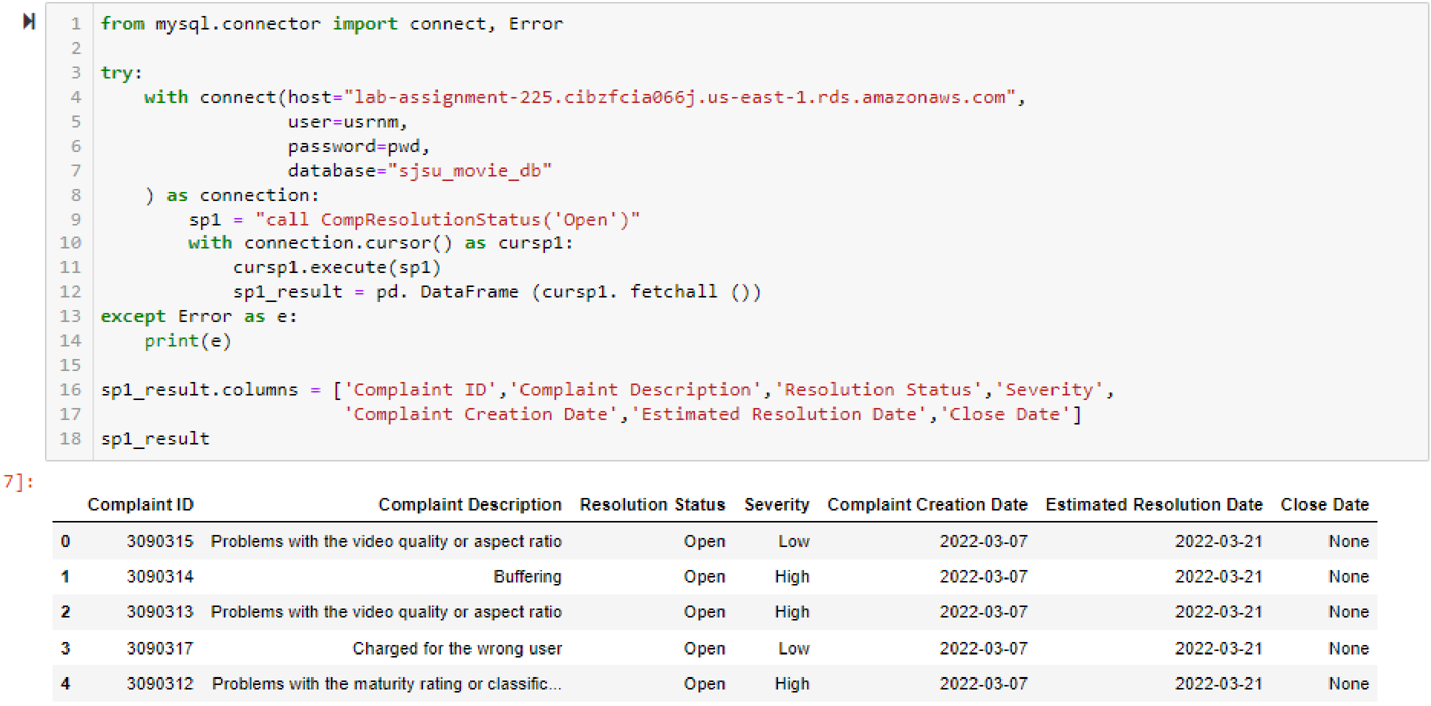


```sql
SELECT comp.complaint_id,comp.complaint_description,comp.resolution_status,comp.severity,
comp.complaint_creation_date,comp.estimated_resolution_date,comp.close_date
FROM cust_complaints comp
WHERE comp.resolution_status = res_status order by complaint_Creation_Date asc limit 5;
END //
DELIMITER ;

call CompResolutionStatus('Open')
```

complaint_id	complaint_description	resolution_status	severity	complaint_creation_date	estimated_resolution_date	close_date
3090315	Problems with the video quality or aspect ratio	Open	Low	2022-03-07	2022-03-21	0000-00-00
3090314	Buffering	Open	High	2022-03-07	2022-03-21	0000-00-00
3090313	Problems with the video quality or aspect ratio	Open	High	2022-03-07	2022-03-21	0000-00-00
3090317	Charged for the wrong user	Open	Low	2022-03-07	2022-03-21	0000-00-00
3090312	Problems with the maturity rating or classification	Open	High	2022-03-07	2022-03-21	0000-00-00

#	Time	Action	Message	Duration / Fetch
1	01:23:15	DROP PROCEDURE IF EXISTS CompResolutionStatus	0 row(s) affected	0.078 sec
2	01:23:16	CREATE PROCEDURE CompResolutionStatus(IN res_status varchar(10)) BEGIN SELECT comp.compl...	0 row(s) affected	0.078 sec
3	01:23:16	call CompResolutionStatus('Open')	5 row(s) returned	0.078 sec / 0.000 sec

Execution of stored procedure created to list the 5 oldest open complaints



```python
from mysql.connector import connect, Error

try:
    with connect(host="lab-assignment-225.cibzfcia066j.us-east-1.rds.amazonaws.com",
                 user=usrnm,
                 password=pwd,
                 database="sjsu_movie_db"
    ) as connection:
        sp1 = "call CompResolutionStatus('Open')"
        with connection.cursor() as cursp1:
            cursp1.execute(sp1)
            sp1_result = pd. DataFrame (cursp1. fetchall ())
except Error as e:
    print(e)

sp1_result.columns = ['Complaint ID','Complaint Description','Resolution Status','Severity',
                      'Complaint Creation Date','Estimated Resolution Date','Close Date']
sp1_result
```

7]:

	Complaint ID	Complaint Description	Resolution Status	Severity	Complaint Creation Date	Estimated Resolution Date	Close Date
0	3090315	Problems with the video quality or aspect ratio	Open	Low	2022-03-07	2022-03-21	None
1	3090314	Buffering	Open	High	2022-03-07	2022-03-21	None
2	3090313	Problems with the video quality or aspect ratio	Open	High	2022-03-07	2022-03-21	None
3	3090317	Charged for the wrong user	Open	Low	2022-03-07	2022-03-21	None
4	3090312	Problems with the maturity rating or classific...	Open	High	2022-03-07	2022-03-21	None

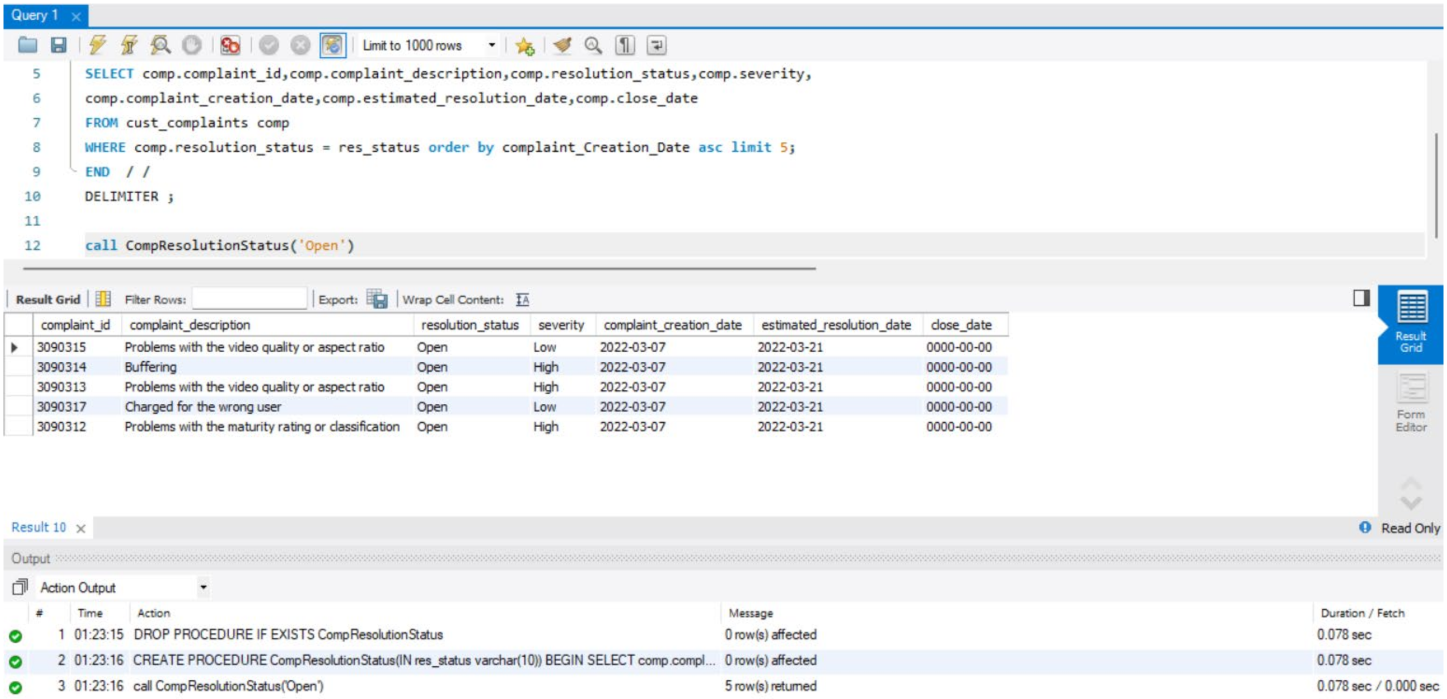
DetailsOfCustomer:

SQL QUERY

```
DROP PROCEDURE IF EXISTS DetailsOfCustomer;

DELIMITER //
CREATE PROCEDURE DetailsOfCustomer(IN input_id varchar(15),IN in1 varchar(10))
BEGIN
if in1='personal' then
Select * from customer_details where customer_id=input_id;
else
SELECT c.customer_id,c.cust_first_name,c.cust_last_name,i.payment_method,i.payment_date,i.total_amount
FROM customer_details c join invoice_details i on i.customer_id=c.customer_id
WHERE c.customer_id= input_id ;
END IF;
END //
DELIMITER ;

call DetailsOfCustomer('9000000025','invoice')
```

EMBEDDED SQL

```
from mysql.connector import connect, Error

try:
with connect(host="lab-assignment-225.cibzfcia066j.us-east-1.rds.amazonaws.com",
user=usrnm,
password=pwd,
database="sjsu_movie_db"
) as connection:
a=input("Enter customer id:\t")
b=input("Personal or Invoice details needed? Please enter input \t")
inp_var="'"+a+"','"+b+"'"
sp4 = "call DetailsOfCustomer({})".format(inp_var)
with connection.cursor() as cursp4:
cursp4.execute(sp4)
sp4_result = pd. DataFrame (cursp4. fetchall ())
except Error as e:
print(e)

if(b.upper()=="INVOICE"):
sp4_result.columns = ['ID','First Name','Last Name','Payment Method','Total Amount','Payment Date']
else:
sp4_result.columns =['ID','SSN','First Name','Middle Name','Last Name','Phone NO','Email','Address Line 1','City','State','Country','Zipcode']
sp4_result
```



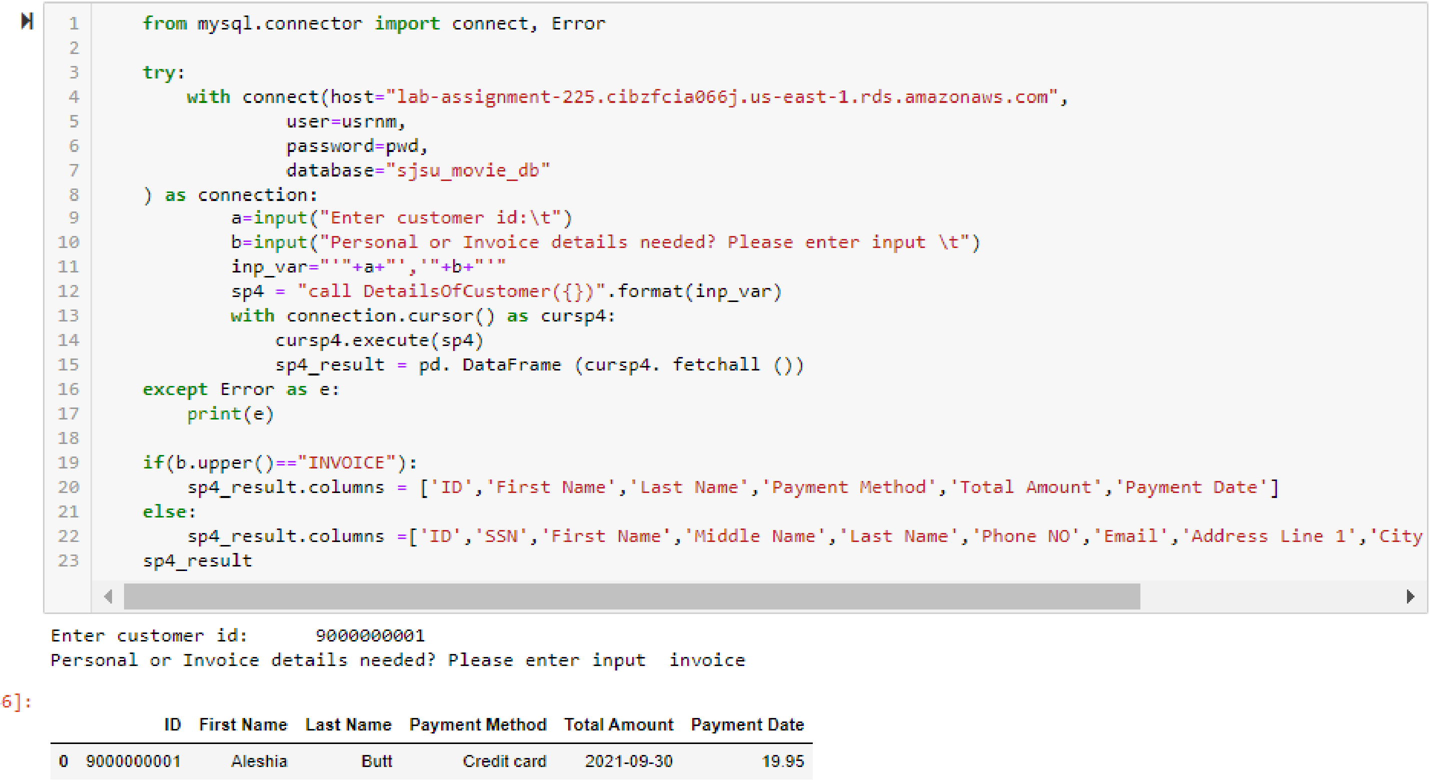
```sql
SELECT comp.complaint_id,comp.complaint_description,comp.resolution_status,comp.severity,
comp.complaint_creation_date,comp.estimated_resolution_date,comp.close_date
FROM cust_complaints comp
WHERE comp.resolution_status = res_status order by complaint_Creation_Date asc limit 5;
END / /
DELIMITER ;

call CompResolutionStatus('Open')
```

complaint_id	complaint_description	resolution_status	severity	complaint_creation_date	estimated_resolution_date	close_date
3090315	Problems with the video quality or aspect ratio	Open	Low	2022-03-07	2022-03-21	0000-00-00
3090314	Buffering	Open	High	2022-03-07	2022-03-21	0000-00-00
3090313	Problems with the video quality or aspect ratio	Open	High	2022-03-07	2022-03-21	0000-00-00
3090317	Charged for the wrong user	Open	Low	2022-03-07	2022-03-21	0000-00-00
3090312	Problems with the maturity rating or classification	Open	High	2022-03-07	2022-03-21	0000-00-00

#	Time	Action	Message	Duration / Fetch
1	01:23:15	DROP PROCEDURE IF EXISTS CompResolutionStatus	0 row(s) affected	0.078 sec
2	01:23:16	CREATE PROCEDURE CompResolutionStatus(IN res_status varchar(10)) BEGIN SELECT comp.compl...	0 row(s) affected	0.078 sec
3	01:23:16	call CompResolutionStatus('Open')	5 row(s) returned	0.078 sec / 0.000 sec

STORED PROCEDURE 4:

Read customer id and 'personal' or 'invoice' as customer input.
If personal selected, personal details of the customer will be displayed.
If invoice selected, invoice details will be selected.



```python
from mysql.connector import connect, Error

try:
    with connect(host="lab-assignment-225.cibzfcia066j.us-east-1.rds.amazonaws.com",
                 user=usrnm,
                 password=pwd,
                 database="sjsu_movie_db"
) as connection:
        a=input("Enter customer id:\t")
        b=input("Personal or Invoice details needed? Please enter input \t")
        inp_var="'"+a+"','"+b+"'"
        sp4 = "call DetailsOfCustomer({})".format(inp_var)
        with connection.cursor() as cursp4:
            cursp4.execute(sp4)
            sp4_result = pd. DataFrame (cursp4. fetchall ())
except Error as e:
    print(e)

if(b.upper()=="INVOICE"):
    sp4_result.columns = ['ID','First Name','Last Name','Payment Method','Total Amount','Payment Date']
else:
    sp4_result.columns =['ID','SSN','First Name','Middle Name','Last Name','Phone NO','Email','Address Line 1','City
sp4_result
```

```
Enter customer id:	9000000001
Personal or Invoice details needed? Please enter input 	invoice
```

46]:

	ID	First Name	Last Name	Payment Method	Total Amount	Payment Date
0	9000000001	Aleshia	Butt	Credit card	2021-09-30	19.95

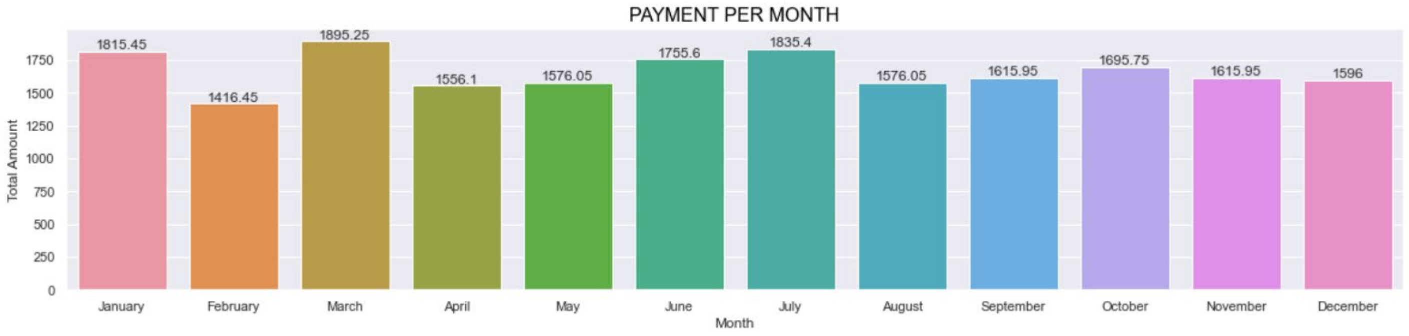
payment_per_month:

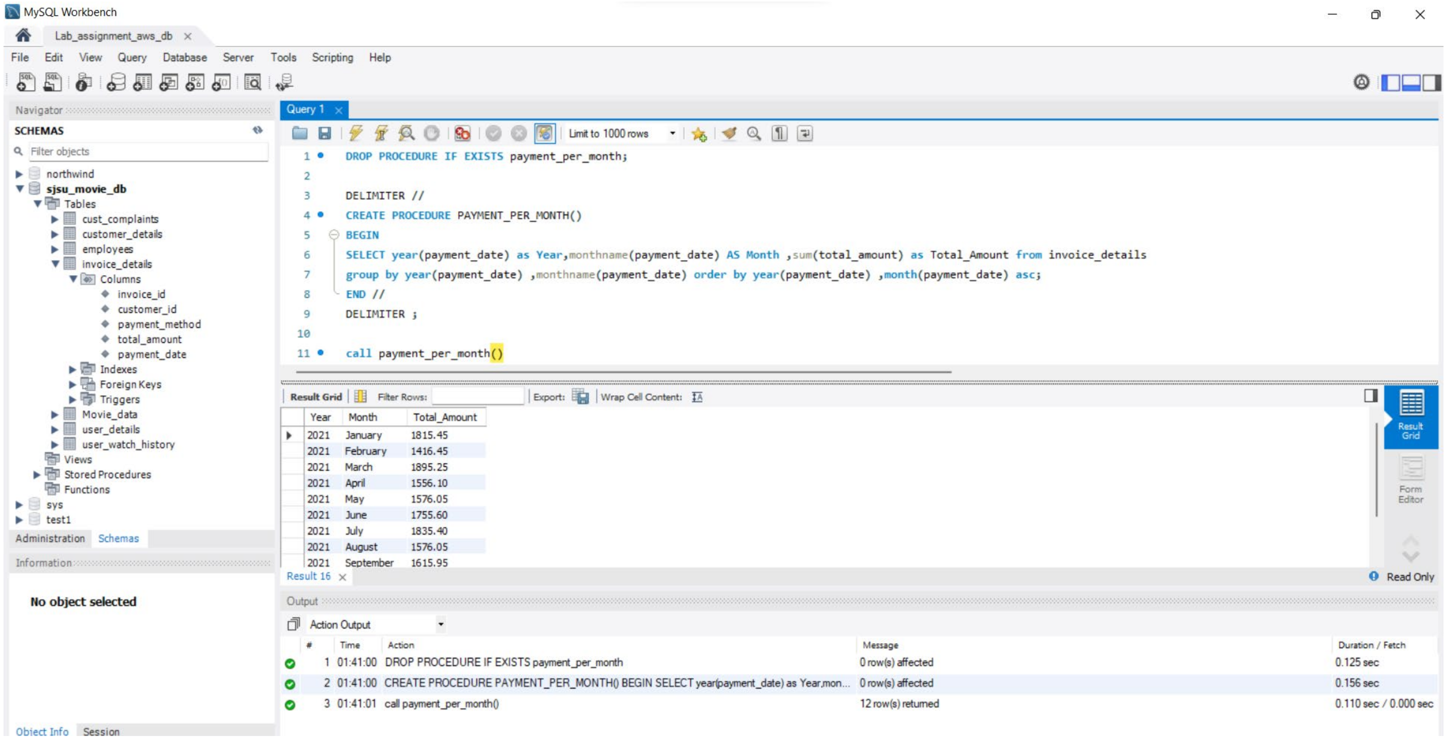
SQL QUERY

```
DROP PROCEDURE IF EXISTS payment_per_month;

DELIMITER //
CREATE PROCEDURE PAYMENT_PER_MONTH()
BEGIN
SELECT year(payment_date) as Year,monthname(payment_date) AS Month ,sum(total_amount) as Total_Amount
from invoice_details
group by year(payment_date) ,monthname(payment_date) order by year(payment_date) ,month(payment_date) asc;
END //
DELIMITER ;

call payment_per_month()
```

EMBEDDED SQL

```
from mysql.connector import connect, Error

try:
with connect(host="lab-assignment-225.cibzfcia066j.us-east-1.rds.amazonaws.com",
user=usrnm,
password=pwd,
database="sjsu_movie_db"
) as connection:
sp5 = "call payment_per_month()"
with connection.cursor() as cursp5:
cursp5.execute(sp5)
sp5_result = pd. DataFrame (cursp5. fetchall ())
except Error as e:
print(e)

sp5_result.columns = ['Year','Month','Total Amount']
sp5_result
```





```sql
DROP PROCEDURE IF EXISTS payment_per_month;

DELIMITER //
CREATE PROCEDURE PAYMENT_PER_MONTH()
BEGIN
SELECT year(payment_date) as Year,monthname(payment_date) AS Month ,sum(total_amount) as Total_Amount from invoice_details
group by year(payment_date) ,monthname(payment_date) order by year(payment_date) ,month(payment_date) asc;
END //
DELIMITER ;

call payment_per_month()
```

Year	Month	Total_Amount
2021	January	1815.45
2021	February	1416.45
2021	March	1895.25
2021	April	1556.10
2021	May	1576.05
2021	June	1755.60
2021	July	1835.40
2021	August	1576.05
2021	September	1615.95

#	Time	Action	Message	Duration / Fetch
1	01:41:00	DROP PROCEDURE IF EXISTS payment_per_month	0 row(s) affected	0.125 sec
2	01:41:00	CREATE PROCEDURE PAYMENT_PER_MONTH() BEGIN SELECT year(payment_date) as Year,mon...	0 row(s) affected	0.156 sec
3	01:41:01	call payment_per_month()	12 row(s) returned	0.110 sec / 0.000 sec

Payment based on year and month

```python
from mysql.connector import connect, Error

try:
    with connect(host="lab-assignment-225.cibzfcia066j.us-east-1.rds.amazonaws.com",
                 user=usrnm,
                 password=pwd,
                 database="sjsu_movie_db"
    ) as connection:
        sp5 = "call payment_per_month()"
        with connection.cursor() as cursp5:
            cursp5.execute(sp5)
            sp5_result = pd. DataFrame (cursp5. fetchall ())
except Error as e:
    print(e)

sp5_result.columns = ['Year','Month','Total Amount']
sp5_result
```

3]:

	Year	Month	Total Amount
0	2021	January	1815.45
1	2021	February	1416.45
2	2021	March	1895.25
3	2021	April	1556.10
4	2021	May	1576.05
5	2021	June	1755.60
6	2021	July	1835.40
7	2021	August	1576.05
8	2021	September	1615.95
9	2021	October	1695.75
10	2021	November	1615.95
11	2021	December	1596.00