

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);
```

```

1 DROP TABLE IF EXISTS `sjsu_movie_db`.`audit_Movie_data`;
2 • CREATE TABLE `sjsu_movie_db`.`audit_Movie_data`
3 (
4     `auditAction` ENUM ('INSERT', 'UPDATE', 'DELETE'),
5     `auditTimestamp` DATETIME DEFAULT CURRENT_TIMESTAMP,
6     `auditId` INT AUTO_INCREMENT ,
7     `Performed By` varchar(200),
8     `Rank_ID` int,
9     `Title` varchar(100),
10    `Description` varchar(300),
11    `Director` varchar(20),
12    `Year` int,
13    `Runtime` int,
14    `Rating` float,
15    `Votes` bigint,
16    `Revenue` float,
17    `Metascore` int,
18    PRIMARY KEY (`auditId`),
19    INDEX (`auditTimestamp`)
20 );

22 • DROP TRIGGER IF EXISTS `sjsu_movie_db`.`Movie_data_inserts`;
23 • CREATE TRIGGER `sjsu_movie_db`.`Movie_data_inserts`
24 AFTER INSERT ON `sjsu_movie_db`.`Movie_data`
25 FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_Movie_data`
26 ('auditAction', 'Performed By', 'Rank_ID', 'Title', 'Description', 'Director', 'Year', 'Runtime', 'Rating', 'Votes', 'Revenue', 'Metascore')
27 VALUES
28 ('INSERT', user(), NEW.Rank_ID, NEW.Title, NEW.Description, NEW.Director, NEW.Year, NEW.Runtime, NEW.Rating, NEW.Votes, NEW.Revenue, NEW.Metascore);
29
30 • DROP TRIGGER IF EXISTS `sjsu_movie_db`.`Movie_data_updates`;
31 • CREATE TRIGGER `sjsu_movie_db`.`Movie_data_updates`
32 AFTER UPDATE ON `sjsu_movie_db`.`Movie_data`
33 FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_Movie_data`
34 ('auditAction', 'Performed By', 'Rank_ID', 'Title', 'Description', 'Director', 'Year', 'Runtime', 'Rating', 'Votes', 'Revenue', 'Metascore')
35 VALUES
36 ('UPDATE', user(), NEW.Rank_ID, NEW.Title, NEW.Description, NEW.Director, NEW.Year, NEW.Runtime, NEW.Rating, NEW.Votes, NEW.Revenue, NEW.Metascore);
37
38 • DROP TRIGGER IF EXISTS `sjsu_movie_db`.`Movie_data_deletes`;
39 • CREATE TRIGGER `sjsu_movie_db`.`Movie_data_deletes`
40 AFTER DELETE ON `sjsu_movie_db`.`Movie_data`
41 FOR EACH ROW INSERT INTO `sjsu_movie_db`.`audit_Movie_data`
42 ('auditAction', 'Performed By', 'Rank_ID', 'Title', 'Description', 'Director', 'Year', 'Runtime', 'Rating', 'Votes', 'Revenue', 'Metascore')
43 VALUES
44 ('DELETE', user(), OLD.Rank_ID, OLD.Title, OLD.Description, OLD.Director, OLD.Year, OLD.Runtime, OLD.Rating, OLD.Votes, OLD.Revenue, OLD.Metascore);

```

MySQL Workbench

Lab_assignment_ews_db x MySQL Model x EER Diagram x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- audit_cust_complaints
- audit_customer_details
- audit_employees
- audit_invoice_details
- audit_Movie_data
 - Columns
 - auditAction
 - auditTimestamp
 - auditId
 - Performed By
 - Rank_ID
 - Title
 - Description
 - Director
 - Year
 - Runtime
 - Rating
 - Votes
 - Revenue
 - Metascore
 - Indexes
 - Foreign Keys
 - Triggers

Administration Schemas Information

SQL File 10* x SQL File 3* x SQL File 6* x

1 • SELECT * from audit_Movie_data;

Result Grid

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@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@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@73-63-213-219.hsd1.ca.comcast.net	1001	Free Willy	Freeing a whale	Simon Wincer	2015	86	5.6	11223	18.64	90

audit_Movie_data 6 x

Output

Action Output

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);

```

MySQL Workbench

Lab_assignment_aws_db x MySQL Model x EER Diagram x

File Edit View Query Database Server Tools Scripting Help

Navigator

Filter objects

SCHEMAS

sjsu_movie_db

Tables

Actor_Details

Columns

Rank_ID

Actor_Name

Indexes

Foreign Keys

Triggers

audit_cust_complaints

audit_customer_details

audit_employees

audit_invoice_details

audit_Movie_data

audit_user_details

audit_user_watch_history

cust_complaints

customer_details

employees

Genre_Details

invoice_details

Movie_data

user_details

SQL File 10* SQL File 3* SQL File 5* SQL File 6*

1 select * from audit_Actor_Details

Result Grid

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

audit_Actor_Details 8 x

Output

Action Output

Time Action

1 19:24:37 select * from audit_Actor_Details LIMIT 0, 1000

Message

3 row(s) returned

Duration / Fech

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);

```

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'sjsu_movie_db' expanded, showing tables like 'Actor_Details', 'audit_cust_complaints', 'audit_customer_details', 'audit_invoice_details', 'audit_movie_data', 'audit_user_details', 'audit_user_watch_history', 'cust_complaints', 'customer_details', 'employees', 'Genre_Details', and 'Rank_ID'. The main window shows a query result grid for the query 'select * from audit_Genre_Details'. The grid contains 5 rows of data, including INSERT, DELETE, and UPDATE actions performed by 'admin@73-63-213-219.hsdl.ca.comcast.net' on '9000' ranked 'Comedy' and 'Thriller' genres. The bottom status bar indicates the query was executed at 20:44:13, returning 5 rows in 0.094 seconds.

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

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);
```

```

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',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_zipcode',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_zipcode',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);

```

SCHEMAS

- Filter objects
- northwind
- sjsu_movie_db
 - Tables
 - audit_customer_details
 - audit_movie_data
 - customer_details
 - employees
 - invoice_details
 - movie_data
 - user_details
 - user_watch_history
 - Views
 - Stored Procedures
 - Functions
 - sys
 - test1

1 • select * from audit_customer_details

Result Grid

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

audit_customer_details2 x

Output

Action Output

#	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,0

```

MySQL Workbench

Lab_assignment_aws_db x MySQL Model x EER Diagram x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHMAS

Filter objects

northwind

sjsu_movie_db

Tables

audit_cust_complaints

audit_customer_details

audit_Movie_data

cust_complaints

Columns

complaint_id

complaint_creation_date

customer_id

emp_id

severity

complaint_description

complaint_category

resolution_status

estimated_resolution_date

close_date

Indexes

Foreign Keys

Triggers

customer_details

employees

invoice_details

Administration Schemas

Information

Query 1 SQL File 20" SQL File 21"

Limit to 1000 rows

1 • select * from audit_cust_complaints;

Result Grid

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...

audit_cust_complaints 2 x

Output

Action Output

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);

```

The screenshot shows the MySQL Workbench interface. The 'Query' tab is active, displaying a SQL query: `select * from audit_user_details;`. The 'Results' tab shows the output of the query, which is a table with 7 columns: `auditAction`, `auditTimestamp`, `auditId`, `Performed By`, `user_id`, `customer_id`, and `screen_no`. The table contains 3 rows of data, corresponding to INSERT, UPDATE, and DELETE operations. The 'Messages' tab at the bottom shows a message: '1 03:31:51 select * from audit_user_details LIMIT 0, 1000 3 row(s) returned'.

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

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

```

MySQL Workbench

Lab_assignment_aws_db x MySQL Model x EER Diagram x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHMAS

Filter objects

northwind

sjsu_movie_db

Tables

- audit_cust_complaints
- audit_customer_details
- audit_employees
- audit_Movie_data
- audit_user_details
- customer_details
- cust_complaints
- employees

Columns

- 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

Indexes

Administration Schemas Information

Query 1 SQL File 24* SQL File 25*

Limit to 1000 rows

1 select * from audit_employees

Result Grid

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

t_employees1 x

Output

Action Output

#	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);
```

```

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.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);

```

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows the 'sjsu_movie_db' database with various tables and columns. The 'audit_invoice_details' table is highlighted. The main pane shows the 'Query 1' window with the SQL statement: `select * from audit_invoice_details;`. The 'Result Grid' shows the results of the query, which are the audit log entries. The 'Output' pane shows the 'Action Output' for the query, indicating that 3 rows were returned.

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.hed1.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.hed1.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.hed1.ca.comcast.net	17696719	9000000001	Credit card	20.99	0000-00-00

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

AUDIT USER WATCH HISTORY:

```
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);
```



```

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);

```

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'sjsu_movie_db' selected. The main area shows the 'Query 1' window with the SQL query 'select * from audit_user_watch_history;'. Below the query, the 'Result Grid' displays the following data:

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

Below the result grid, the 'Output' window shows the message: 'select * from audit_user_watch_history LIMIT 0, 1000' and '3 row(s) returned'.

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);
```

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'northwind' and 'sjsu_movie_db' expanded. The 'sjsu_movie_db' schema contains tables like 'cust_complaints', 'customer_details', 'employees', 'invoice_details', 'Movie_data', 'user_details', and 'user_watch_history'. The main editor shows a SQL query with line numbers 1 through 19. The query includes a 'drop trigger if exists' statement, a 'delimiter //', a 'create trigger' statement for 'before_insert_users', and an 'insert into user_details' statement. The 'Output' tab at the bottom shows the execution results:

#	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 varchar...	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 SUBSCRIPTION 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');
```

The screenshot displays the MySQL Workbench interface. The left sidebar shows the 'SCHEMAS' tree with 'northwind' and 'sjsu_movie_db' expanded. The 'Query' window contains the SQL code from the previous block. The 'Output' window at the bottom shows the execution results:

#	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');
```

The screenshot shows the MySQL Workbench interface. The left sidebar displays the database schema for 'sjsu_movie_db', including tables like 'customer_details', 'employees', 'invoice_details', 'Movie_data', 'user_details', and 'user_watch_history'. The 'user_watch_history' table is selected, showing columns: 'user_id' (varchar(15)), 'rank_id' (int), and 'watch_date' (date). The main query window contains the SQL code for dropping the trigger, creating it, and inserting a row. The 'Output' window at the bottom shows the execution results:

#	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');
```

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' pane with a tree view of databases including 'northwind', 'sjsu_movie_db', and 'sys'. The main editor window shows the SQL script from the previous block. The 'Output' pane at the bottom displays the execution results:

#	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 with the maturity rating or classification','Audio quality issue','Closed','', '2021-11-28');	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_movie' (actor_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-22 02:20:17.77	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
Actor_Details_updates	UPDATE	Actor_Details	INSERT INTO 'sjsu_movie' (actor_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-22 02:20:17.97	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
Actor_Details_deletes	DELETE	Actor_Details	INSERT INTO 'sjsu_movie' (actor_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-22 02:20:18.15	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
Genre_Details_inserts	INSERT	Genre_Details	INSERT INTO 'sjsu_movie' (genre_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-22 03:40:31.40	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
Genre_Details_updates	UPDATE	Genre_Details	INSERT INTO 'sjsu_movie' (genre_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-22 03:40:31.59	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
Genre_Details_deletes	DELETE	Genre_Details	INSERT INTO 'sjsu_movie' (genre_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-22 03:40:31.78	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
Movie_data_inserts	INSERT	Movie_data	INSERT INTO 'sjsu_movie' (movie_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-22 02:18:16.98	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
Movie_data_updates	UPDATE	Movie_data	INSERT INTO 'sjsu_movie' (movie_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-22 02:18:17.16	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
Movie_data_deletes	DELETE	Movie_data	INSERT INTO 'sjsu_movie' (movie_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-22 02:18:17.34	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
before_insert_cust_complaints	INSERT	cust_complaints	begin declare msg_err varchar(255);	BEFORE	2022-03-13 02:17:55.89	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
cust_complaints_inserts	INSERT	cust_complaints	INSERT INTO 'sjsu_movie' (movie_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-13 10:16:39.16	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
cust_complaints_updates	UPDATE	cust_complaints	INSERT INTO 'sjsu_movie' (movie_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-13 10:16:39.34	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
cust_complaints_deletes	DELETE	cust_complaints	INSERT INTO 'sjsu_movie' (movie_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-13 10:16:39.52	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
customer_details_inserts	INSERT	customer_details	INSERT INTO 'sjsu_movie' (movie_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-13 09:57:35.86	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
customer_details_updates	UPDATE	customer_details	INSERT INTO 'sjsu_movie' (movie_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-13 09:57:36.04	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
customer_details_deletes	DELETE	customer_details	INSERT INTO 'sjsu_movie' (movie_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-13 09:57:36.22	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
employees_inserts	INSERT	employees	INSERT INTO 'sjsu_movie' (movie_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-13 10:40:40.51	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
employees_updates	UPDATE	employees	INSERT INTO 'sjsu_movie' (movie_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-13 10:40:40.72	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
employees_deletes	DELETE	employees	INSERT INTO 'sjsu_movie' (movie_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-13 10:40:40.91	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
before_insert_invoice_details	INSERT	invoice_details	begin declare msg_err varchar(255);	BEFORE	2022-03-09 20:07:12.09	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
invoice_details_inserts	INSERT	invoice_details	INSERT INTO 'sjsu_movie' (movie_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-13 10:50:49.08	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
invoice_details_updates	UPDATE	invoice_details	INSERT INTO 'sjsu_movie' (movie_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-13 10:50:49.26	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
invoice_details_deletes	DELETE	invoice_details	INSERT INTO 'sjsu_movie' (movie_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-13 10:50:49.44	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
before_insert_user_details	INSERT	user_details	begin declare msg_err varchar(255);	BEFORE	2022-03-09 06:10:02.57	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
user_details_inserts	INSERT	user_details	INSERT INTO 'sjsu_movie' (movie_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-13 10:27:16.79	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
user_details_updates	UPDATE	user_details	INSERT INTO 'sjsu_movie' (movie_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-13 10:27:16.97	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
user_details_deletes	DELETE	user_details	INSERT INTO 'sjsu_movie' (movie_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-13 10:27:17.15	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
before_insert_watch_history	INSERT	user_watch_history	begin declare msg_err varchar(255);	BEFORE	2022-03-09 07:54:50.31	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
user_watch_history_inserts	INSERT	user_watch_history	INSERT INTO 'sjsu_movie' (movie_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-13 10:59:58.05	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
user_watch_history_updates	UPDATE	user_watch_history	INSERT INTO 'sjsu_movie' (movie_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-13 10:59:58.24	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
user_watch_history_deletes	DELETE	user_watch_history	INSERT INTO 'sjsu_movie' (movie_id, title, year, rank_id) VALUES (1, 'The Godfather', 1972, 1)	AFTER	2022-03-13 10:59:58.41	NO_ENGINE_SUBSTITUTION	admin@%	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci

STORED PROCEDURES

TopTenMoviesPerYear:

SQL QUERY
<pre> 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')</pre>

EMBEDDED SQL
<pre> from mysql.connector import connect, Error try: with connect(host="lab-assignment-225.cibzfcia066j.us-east-1.rds.amazonaws.com", user=usnm, 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</pre>

MySQL Workbench

Lab_assignment_aws_db

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHMAS

Filter objects

sjsu_movie_db

Tables

Views

Stored Procedures

CompResolutionStatus

DetailsOfCustomer

GetMaximumTrafficInAMonth

GetTopMovie

PAYMENT_PER_MONTH

TopMoviesPerYear

Functions

Administration Schemas

Information

No object selected

Object Info Session

SQL File 6'

Limit to 1000 rows

```

1 DROP PROCEDURE IF EXISTS TopTenMoviesPerYear;
2 DELIMITER //
3 CREATE PROCEDURE TopTenMoviesPerYear(IN year_val INT)
4 BEGIN
5     SELECT m.year,m.title, count(*) as no_of_movies_watched
6     FROM user_watch_history u join Movie_data m on u.rank_id=m.rank_id
7     WHERE m.year = year_val group by m.year,m.title order by count(*) desc limit 10;
8 END //
9 DELIMITER ;
10
11 call TopTenMoviesPerYear('2014')

```

Result Grid

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

Result 3 x

Output

Action Output

#	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

```

1 from mysql.connector import connect, Error
2
3 try:
4     with connect(host="lab-assignment-225.cibzfcia066j.us-east-1.rds.amazonaws.com",
5                 user=usnm,
6                 password=pwd,
7                 database="sjsu_movie_db"
8             ) as connection:
9         a=input("Enter the year (2006 - 2016) for which top 10 movies need to be displayed:\t")
10        sp3 = "call TopTenMoviesPerYear('{}').format(a)
11        with connection.cursor() as cursp3:
12            cursp3.execute(sp3)
13            sp3_result = pd.DataFrame (cursp3. fetchall ())
14 except Error as e:
15     print(e)
16
17 sp3_result.columns = ['Year','Title','No of times watched']
18 sp3_result

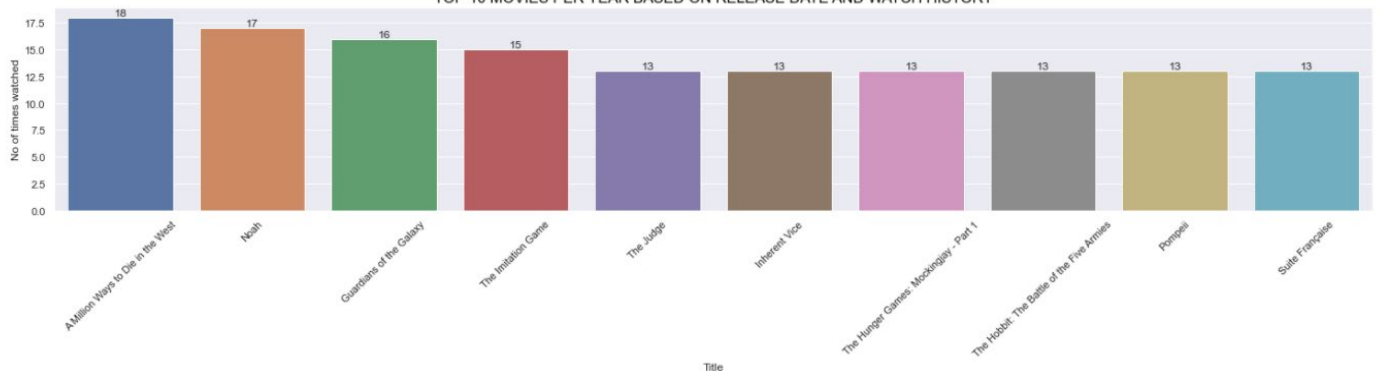
```

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

TOP 10 MOVIES PER YEAR BASED ON RELEASE DATE AND WATCH HISTORY



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')
```

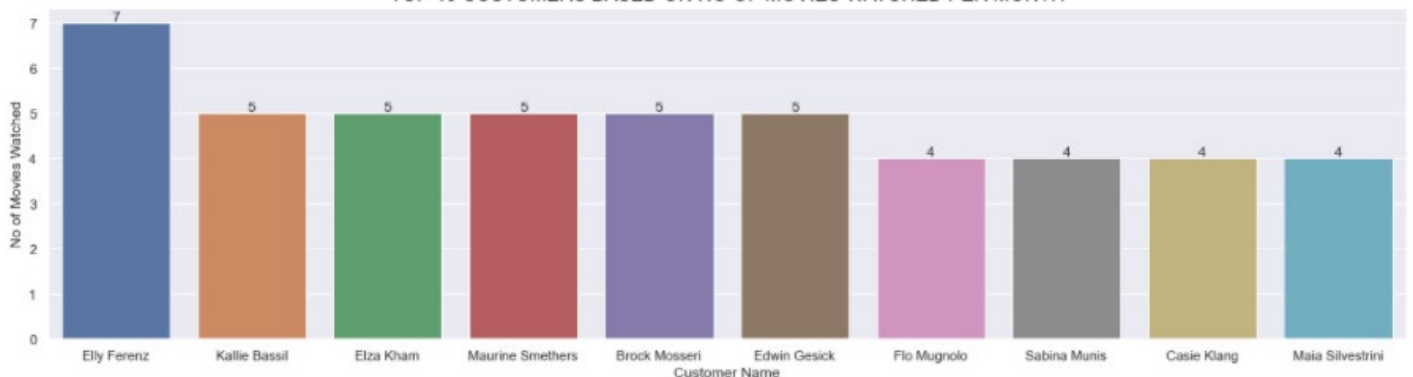
EMBEDDED SQL

```
from mysql.connector import connect, Error

try:
with connect(host="lab-assignment-225.cibzfcia066j.us-east-1.rds.amazonaws.com",
            user=usnm,
            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
```

TOP 10 CUSTOMERS BASED ON NO OF MOVIES WATCHED PER MONTH



MySQL Workbench

Lab_assignment_aws_db

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

northwind

sjsu_movie_db

Tables

cust_complaints

customer_details

employees

invoice_details

Movie_data

user_details

user_watch_history

Views

Stored Procedures

Functions

sys

test1

Query 1

```

4 BEGIN
5 SELECT concat(c.cust_first_name,' ',c.cust_last_name) as customer_name ,count(u.rank_id) as no_of_movies_watched
6 FROM user_watch_history u join user_details ud on ud.user_id=u.user_id
7 join customer_details c on c.customer_id=ud.customer_id
8 WHERE MONTH(watch_date) = traffic_month group by c.customer_id order by count(ud.user_id) desc limit 10;
9 END //
10 DELIMITER ;
11
12 call GetMaximumTrafficInAMonth('03')

```

Result Grid

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

Administration Schemas

Information

No object selected

Result 11

Output

Action Output

#	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

```

1 from mysql.connector import connect, Error
2
3 try:
4     with connect(host="lab-assignment-225.cibzfcia066j.us-east-1.rds.amazonaws.com",
5                 user=usnm,
6                 password=pwd,
7                 database="sjsu_movie_db"
8             ) as connection:
9         a=input("Enter Month number:\t")
10        sp2 = "call GetMaximumTrafficInAMonth('{}').format(a)
11        with connection.cursor() as cursp2:
12            cursp2.execute(sp2)
13            sp2_result = pd.DataFrame (cursp2. fetchall ())
14 except Error as e:
15     print(e)
16
17 sp2_result.columns = ['Customer Name','No of Movies Watched']
18 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=usnm,
            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
```

```

Query 1 x
5 SELECT comp.complaint_id,comp.complaint_description,comp.resolution_status,comp.severity,
6 comp.complaint_creation_date,comp.estimated_resolution_date,comp.close_date
7 FROM cust_complaints comp
8 WHERE comp.resolution_status = res_status order by complaint_Creation_Date asc limit 5;
9 END //
10 DELIMITER ;
11
12 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

```

1 from mysql.connector import connect, Error
2
3 try:
4     with connect(host="lab-assignment-225.cibzfcia066j.us-east-1.rds.amazonaws.com",
5                 user=usnm,
6                 password=pwd,
7                 database="sjsu_movie_db"
8             ) as connection:
9         sp1 = "call CompResolutionStatus('Open')"
10        with connection.cursor() as cursp1:
11            cursp1.execute(sp1)
12            sp1_result = pd.DataFrame (cursp1. fetchall ())
13 except Error as e:
14     print(e)
15
16 sp1_result.columns = ['Complaint ID','Complaint Description','Resolution Status','Severity',
17                      'Complaint Creation Date','Estimated Resolution Date','Close Date']
18 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=usnm,
        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
```

```

Query 1 x
Limit to 1000 rows
5 SELECT comp.complaint_id,comp.complaint_description,comp.resolution_status,comp.severity,
6 comp.complaint_creation_date,comp.estimated_resolution_date,comp.close_date
7 FROM cust_complaints comp
8 WHERE comp.resolution_status = res_status order by complaint_Creation_Date asc limit 5;
9 END //
10 DELIMITER ;
11
12 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.

```

1 from mysql.connector import connect, Error
2
3 try:
4     with connect(host="lab-assignment-225.cibzfcia066j.us-east-1.rds.amazonaws.com",
5                 user=usnm,
6                 password=pwd,
7                 database="sjsu_movie_db"
8     ) as connection:
9         a=input("Enter customer id:\t")
10        b=input("Personal or Invoice details needed? Please enter input \t")
11        inp_var=" "+a+" "+b+" "
12        sp4 = "call DetailsOfCustomer({})".format(inp_var)
13        with connection.cursor() as cursp4:
14            cursp4.execute(sp4)
15            sp4_result = pd. DataFrame (cursp4. fetchall ())
16    except Error as e:
17        print(e)
18
19    if(b.upper()=="INVOICE"):
20        sp4_result.columns = ['ID','First Name','Last Name','Payment Method','Total Amount','Payment Date']
21    else:
22        sp4_result.columns =['ID','SSN','First Name','Middle Name','Last Name','Phone NO','Email','Address Line 1','City']
23    sp4_result

```

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

!6]:

	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=usnm,
            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
```



MySQL Workbench

Lab_assignment_aws_db

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

northwind

sjsu_movie_db

Tables

cust_complaints

customer_details

employees

invoice_details

Columns

invoice_id

customer_id

payment_method

total_amount

payment_date

Indexes

Foreign Keys

Triggers

Movie_data

user_details

user_watch_history

Views

Stored Procedures

Functions

sys

test1

Administration Schemas

Information

No object selected

Object Info Session

Query 1

```

1 DROP PROCEDURE IF EXISTS payment_per_month;
2
3 DELIMITER //
4 CREATE PROCEDURE PAYMENT_PER_MONTH()
5 BEGIN
6 SELECT year(payment_date) as Year, monthname(payment_date) AS Month, sum(total_amount) as Total_Amount from invoice_details
7 group by year(payment_date), monthname(payment_date) order by year(payment_date), month(payment_date) asc;
8 END //
9 DELIMITER ;
10
11 call payment_per_month();

```

Result Grid

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

Result 16

Output

Action Output

#	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

```

1 from mysql.connector import connect, Error
2
3 try:
4     with connect(host="lab-assignment-225.cibzfcia066j.us-east-1.rds.amazonaws.com",
5                 user=usnm,
6                 password=pwd,
7                 database="sjsu_movie_db"
8             ) as connection:
9         sp5 = "call payment_per_month()"
10        with connection.cursor() as cursp5:
11            cursp5.execute(sp5)
12            sp5_result = pd.DataFrame (cursp5. fetchall ())
13 except Error as e:
14     print(e)
15
16 sp5_result.columns = ['Year', 'Month', 'Total Amount']
17 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