

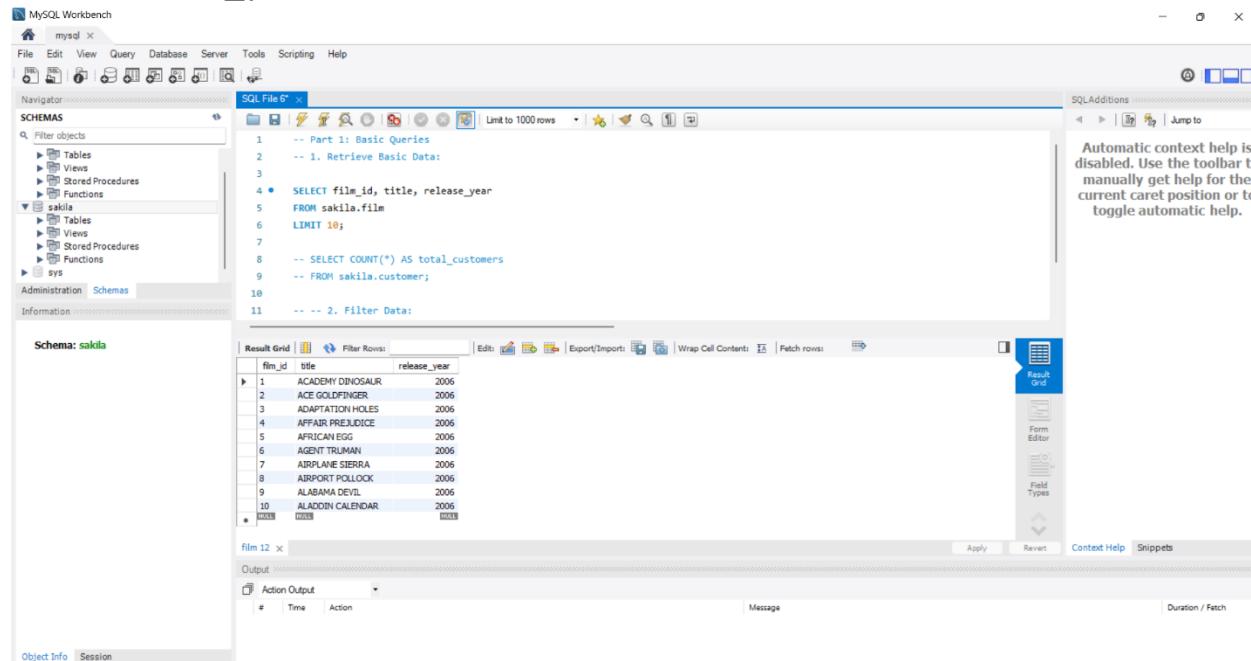
Name: Yeab Chhunsann

Class: M1

## Part: 1

### 1. Retrieve Basic Data:

- Write a query to list the first 10 films, displaying the film\_id, title, and release\_year.



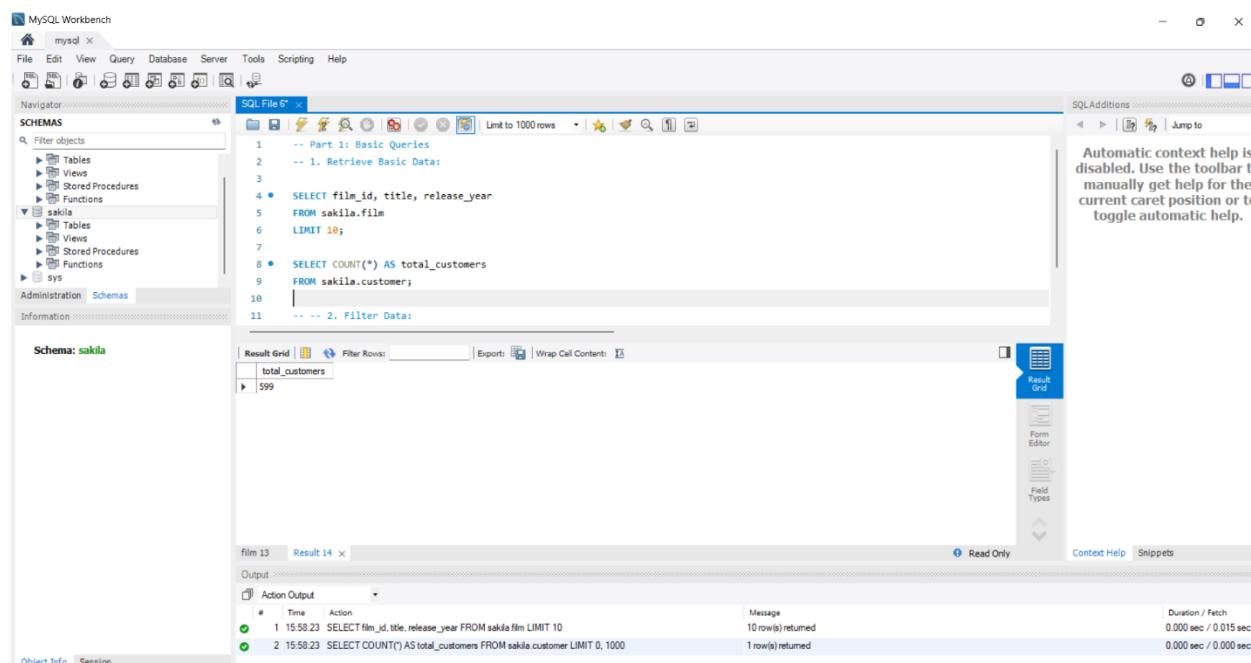
The screenshot shows the MySQL Workbench interface with the Sakila database selected. In the SQL Editor tab, the following query is run:

```
-- Part 1: Basic Queries
-- 1. Retrieve Basic Data:
SELECT film_id, title, release_year
FROM sakila.film
LIMIT 10;
```

The Result Grid displays the first 10 rows of film data:

film_id	title	release_year
1	ACADEMY DINOSAUR	2006
2	ACE GOLDFINGER	2006
3	ADAPTATION HOLES	2006
4	AFFAIR PREJUDICE	2006
5	AFRICAN EGG	2006
6	AGENT TRUMAN	2006
7	AIRPLANE SIERRA	2006
8	AIRPORT POLLOCK	2006
9	ALABAMA DEVIL	2006
10	ALADDIN CALENDAR	2006

- Find the total number of customers in the database.



The screenshot shows the MySQL Workbench interface with the Sakila database selected. In the SQL Editor tab, the following query is run:

```
-- Part 1: Basic Queries
-- 1. Retrieve Basic Data:
SELECT film_id, title, release_year
FROM sakila.film
LIMIT 10;

-- 2. Find the total number of customers in the database.
SELECT COUNT(*) AS total_customers
FROM sakila.customer;
```

The Result Grid displays the total number of customers:

total_customers
599

The Output pane shows the execution details:

Action	Time	Message	Duration / Fetch
1	15:58:23	SELECT film_id, title, release_year FROM sakila.film LIMIT 10	0.000 sec / 0.015 sec
2	15:58:23	SELECT COUNT() AS total_customers FROM sakila.customer LIMIT 0, 1000	0.000 sec / 0.000 sec

## 2. Filter Data:

- List all movies in the film table released in the year 2006.

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
5 FROM sakila.film
6 LIMIT 10;
7
8 * SELECT COUNT(*) AS total_customers
9 FROM sakila.customer;
10
11 -- 2. Filter Data:
12 * SELECT title
13 FROM sakila.film
14 WHERE release_year = 2006;
15
```

The Result Grid shows the results of the second part of the query:

title
ACADEMY DINOSAUR
ACE GOLDFINGER
ADAPTATION HOLES
AFFAIR PREJUDICE
AFRICAN EGG
AGBETI TRUMAN
AVIPLANE SIERRA
AIRPORT POLLOCK
ALABAMA DEVIL
ALADDIN CALENDAR
ALAMO VIDEOTAPE
ALASKA PHANTOM

The Output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
4	15:59:04	SELECT COUNT(*) AS total_customers FROM sakila.customer LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
5	15:59:04	SELECT title FROM sakila.film WHERE release_year = 2006 LIMIT 0, 1000	1000 row(s) returned	0.000 sec / 0.000 sec

- Retrieve all customers whose last name starts with the letter “S”.

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
10
11 -- 2. Filter Data:
12 * SELECT title
13 FROM sakila.film
14 WHERE release_year = 2006;
15
16 * SELECT customer_id, first_name, last_name
17 FROM sakila.customer
18 WHERE last_name LIKE 'S%';
19
20
```

The Result Grid shows the results of the third part of the query:

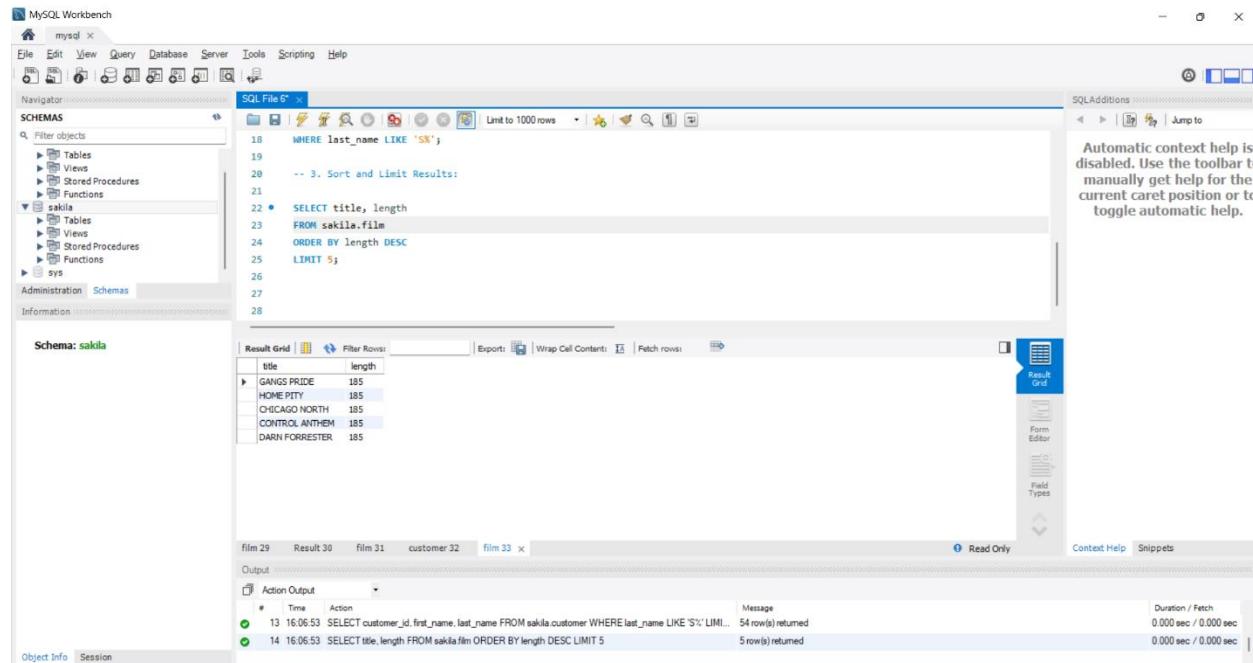
customer_id	first_name	last_name
347	RYAN	SALSBURY
498	GENE	SANBORN
52	JULIE	SANCHEZ
75	TAMMY	SANDERS
303	WILLIAM	SATTERFIELD
471	DEAN	SAUER
353	JONATHAN	SCARBOROUGH
387	JESSE	SOHLLING
204	ROSEMARY	SCHMIDT
405	LEONARD	SCHOFIELD
397	DMYTRY	SCHRAEDER
321	KEVIN	SCHULER

The Output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
3	16:01:09	SELECT title FROM sakila.film WHERE release_year = 2006 LIMIT 0, 1000	1000 row(s) returned	0.000 sec / 0.000 sec
4	16:01:09	SELECT customer_id, first_name, last_name FROM sakila.customer WHERE last_name LIKE 'S%' LIMIT 0, 1000	54 row(s) returned	0.000 sec / 0.000 sec

### 3. Sort and Limit Results:

- Find the top 5 longest movies (length), displaying title and length.



The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema **sakila** selected, with tables **customer**, **film**, and **rental** listed.
- SQL Editor:** Contains the following SQL code:

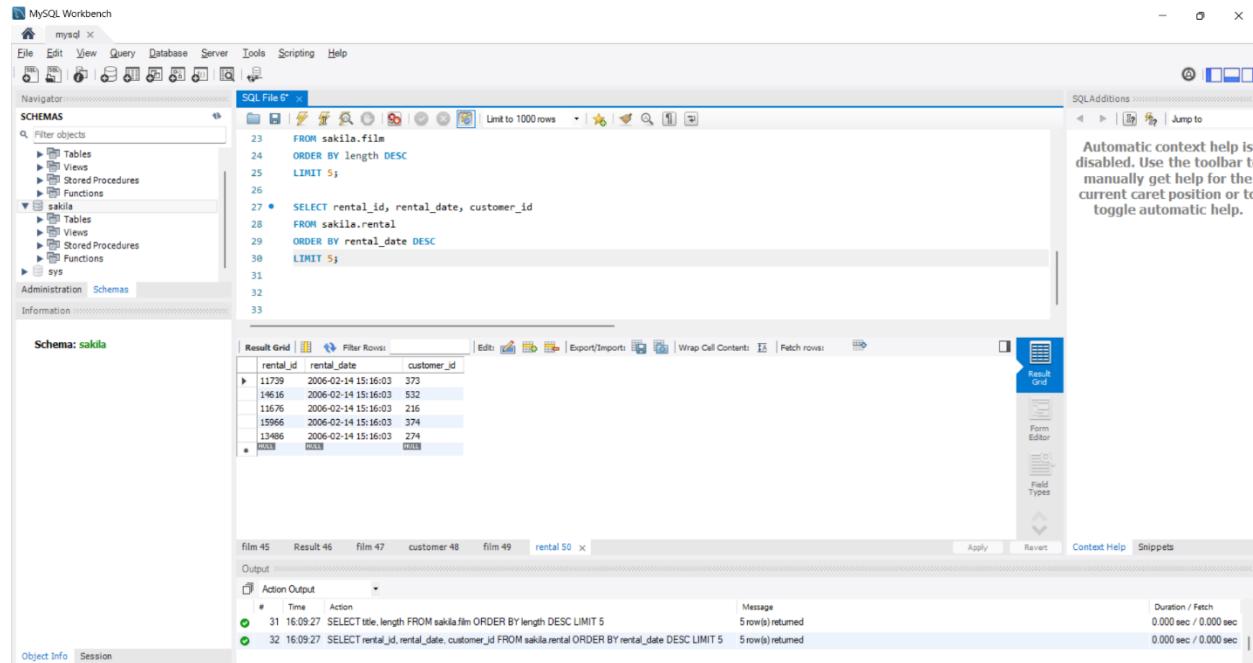
```

18 WHERE last_name LIKE '%S%';
19
20 -- 3. Sort and Limit Results:
21
22 • SELECT title, length
23 FROM sakila.film
24 ORDER BY length DESC
25 LIMIT 5;
26
27
28

```
- Result Grid:** Displays the results of the query:

title	length
GANGS PRIDE	185
HOME PITY	185
CHICAGO NORTH	185
CONTROL ANTHEM	185
DARN FORBESTER	185
- Action Output:** Shows two log entries:
  - 13 16:06:53 SELECT customer\_id, first\_name, last\_name FROM sakila.customer WHERE last\_name LIKE '%S%' LIMIT 5 row(s) returned
  - 14 16:06:53 SELECT title, length FROM sakila.film ORDER BY length DESC LIMIT 5 row(s) returned

- List the 5 most recent rentals, showing rental\_id, rental\_date, and customer\_id.



The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema **sakila** selected, with tables **customer**, **film**, and **rental** listed.
- SQL Editor:** Contains the following SQL code:

```

23 FROM sakila.film
24 ORDER BY length DESC
25 LIMIT 5;
26
27 • SELECT rental_id, rental_date, customer_id
28 FROM sakila.rental
29 ORDER BY rental_date DESC
30 LIMIT 5;
31
32
33

```
- Result Grid:** Displays the results of the query:

rental_id	rental_date	customer_id
11739	2006-02-14 15:16:03	373
14616	2006-02-14 15:16:03	532
11676	2006-02-14 15:16:03	216
15966	2006-02-14 15:16:03	374
13486	2006-02-14 15:16:03	274
- Action Output:** Shows two log entries:
  - 31 16:09:27 SELECT title, length FROM sakila.film ORDER BY length DESC LIMIT 5 row(s) returned
  - 32 16:09:27 SELECT rental\_id, rental\_date, customer\_id FROM sakila.rental ORDER BY rental\_date DESC LIMIT 5 row(s) returned

## Part 2: Advanced Queries

### 4. Joins:

- Write a query to find the names of actors who appeared in the movie “Academy Dinosaur”.

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is `sakila`.
- Query Editor:** The code entered is:

```
-- LIMIT 5;
38
39
40 -- Part 2: Advanced Queries
41 -- 4. Joins:
42 -- Write a query to find the names of actors who appeared in the movie "Academy Dinosaur".
43
44 • SELECT a.first_name, a.last_name
45   FROM actor a
46   JOIN film_actor fa ON a.actor_id = fa.actor_id
47   JOIN film f ON fa.film_id = f.film_id
48   WHERE f.title = 'Academy Dinosaur';
49
50
51 -- Retrieve a list of customers along with the titles of movies they rented. Include customer_id, first_name, last_name, and title.
```

**Result Grid:**

first_name	last_name
PENELOPE	GUINNESS
CHRISTIAN	GABLE
LUCILLE	TRACY
SANDRA	PECK
JOHNNY	CAGE
MENA	TEMPLE
WARREN	NOLTE

**Action Output:**

```
1 17:53:37 SELECT a.first_name, a.last_name FROM actor a JOIN film_actor fa ON a.actor_id = fa.actor_id JOIN film f ON ... 10 row(s) returned
Message
Duration / Fetch
0.047 sec / 0.000 sec
```

- Retrieve a list of customers along with the titles of movies they rented. Include `customer_id`, `first_name`, `last_name`, and `title`.

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is `sakila`.
- Query Editor:** The code entered is:

```
-- JOIN film_actor fa ON a.actor_id = fa.actor_id
-- JOIN film f ON fa.film_id = f.film_id
-- WHERE f.title = 'Academy Dinosaur';
50
51 -- Retrieve a list of customers along with the titles of movies they rented. Include customer_id, first_name, last_name, and title.
52
53 • SELECT c.customer_id, c.first_name, c.last_name, f.title
54   FROM customer c
55   JOIN rental r ON c.customer_id = r.customer_id
56   JOIN inventory i ON r.inventory_id = i.inventory_id
57   JOIN film f ON i.film_id = f.film_id;
58
59
60 -- 5. Aggregations:
```

**Result Grid:**

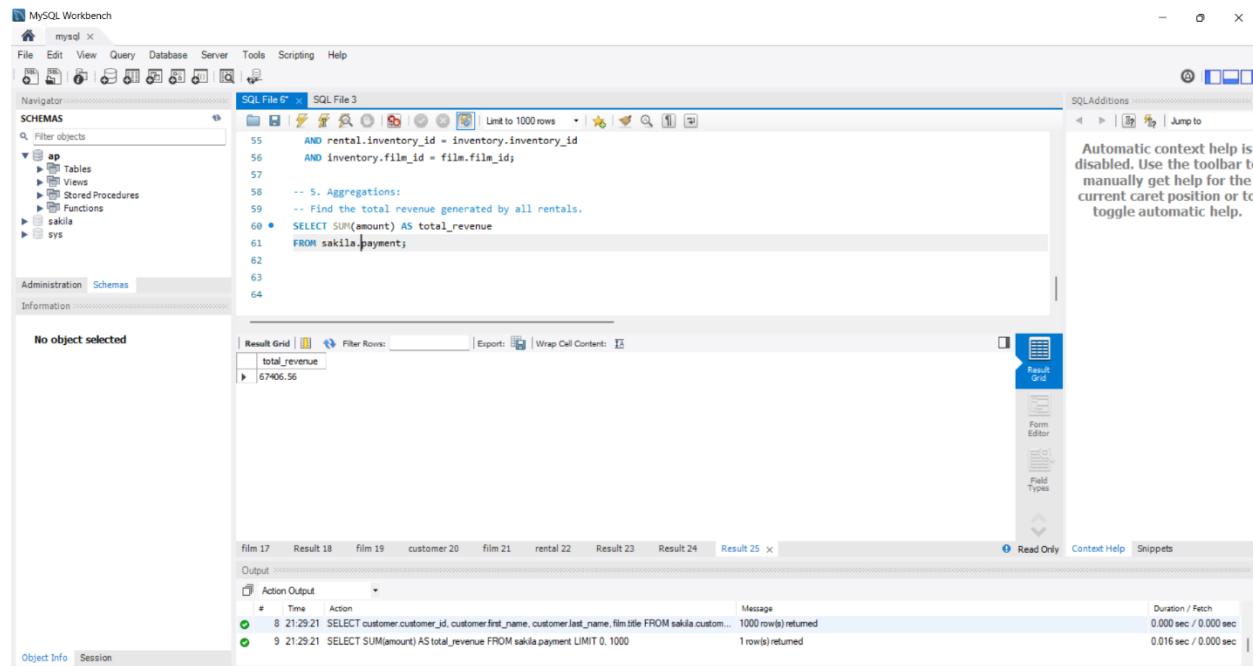
customer_id	first_name	last_name	title
431	JOEL	FRANCISCO	ACADEMY DINOSAUR
518	GERALD	HARDER	ACADEMY DINOSAUR
279	DIMMIE	SHELTON	ACADEMY DINOSAUR
411	NORMAN	DURRER	ACADEMY DINOSAUR
170	BEATRICE	AUGOLD	ACADEMY DINOSAUR
161	GERALDINE	PERKINS	ACADEMY DINOSAUR
581	VIRGIL	WOFFORD	ACADEMY DINOSAUR

**Action Output:**

```
1 17:53:37 SELECT a.first_name, a.last_name FROM actor a JOIN film_actor fa ON a.actor_id = fa.actor_id JOIN film f ON ... 10 row(s) returned
Message
Duration / Fetch
0.047 sec / 0.000 sec
2 18:04:59 SELECT c.customer_id, c.first_name, c.last_name, f.title FROM customer c JOIN rental r ON c.customer_id = r.customer_id ... 1000 row(s) returned
Message
Duration / Fetch
0.032 sec / 0.000 sec
```

## 5. Aggregations:

- Find the total revenue generated by all rentals.



The screenshot shows the MySQL Workbench interface with a query editor window titled "SQL File 6". The code is as follows:

```

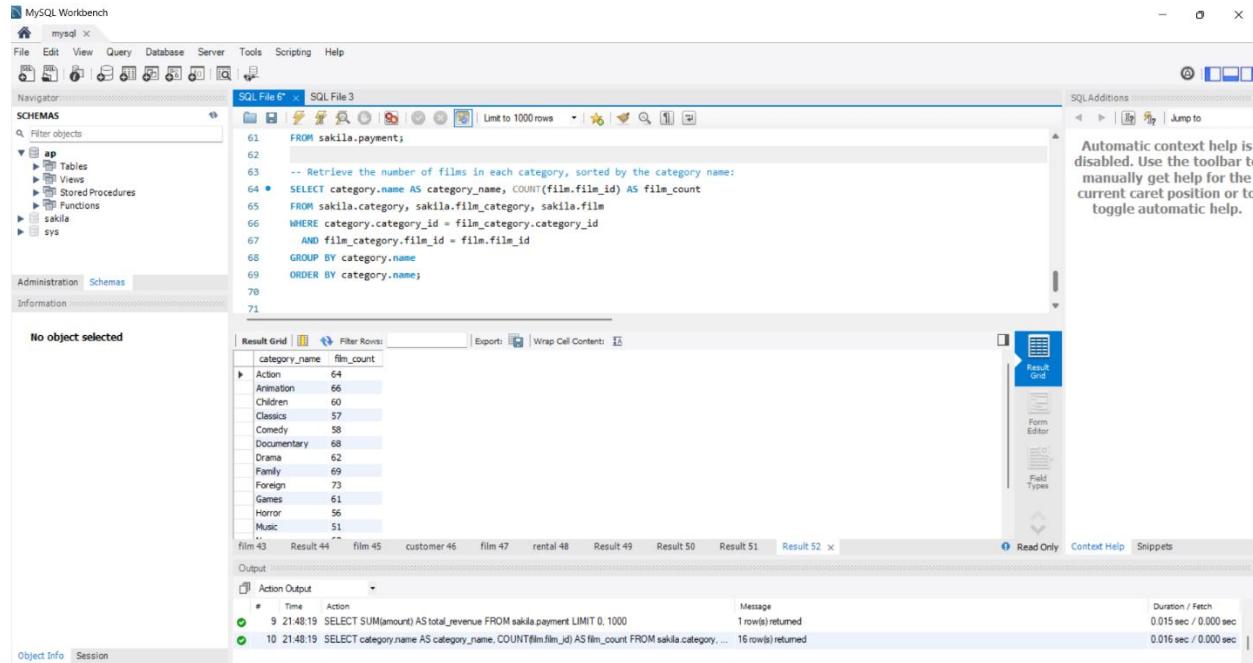
55     AND rental.inventory_id = inventory.inventory_id
56     AND inventory.film_id = film.film_id;
57
58 -- 5. Aggregations:
59 -- Find the total revenue generated by all rentals.
60 •   SELECT SUM(amount) AS total_revenue
61   FROM sakila.payment;
62
63
64

```

The result grid shows one row with the value 67406.56. The output pane shows two rows of execution logs:

Action	Time	Action	Message	Duration / Fetch
8	21:29:21	SELECT customer.customer_id, customer.first_name, customer.last_name, film.title FROM sakila.customer	1000 row(s) returned	0.000 sec / 0.000 sec
9	21:29:21	SELECT SUM(amount) AS total_revenue FROM sakila.payment LIMIT 0, 1000	1row(s) returned	0.016 sec / 0.000 sec

- Retrieve the number of films in each category, sorted by the category name.



The screenshot shows the MySQL Workbench interface with a query editor window titled "SQL File 6". The code is as follows:

```

61   FROM sakila.payment;
62
63 -- Retrieve the number of films in each category, sorted by the category name:
64 •   SELECT category.name AS category_name, COUNT(film.film_id) AS film_count
65   FROM sakila.category, sakila.film_category, sakila.film
66   WHERE category.category_id = film_category.category_id
67     AND film_category.film_id = film.film_id
68   GROUP BY category_name
69   ORDER BY category_name;
70
71

```

The result grid shows the following data:

category_name	film_count
Action	64
Animation	66
Children	60
Classics	57
Comedy	58
Documentary	68
Drama	62
Family	69
Foreign	73
Games	61
Horror	56
Music	51

The output pane shows two rows of execution logs:

Action	Time	Action	Message	Duration / Fetch
9	21:48:19	SELECT SUM(amount) AS total_revenue FROM sakila.payment LIMIT 0, 1000	1row(s) returned	0.015 sec / 0.000 sec
10	21:48:19	SELECT category.name AS category_name, COUNT(film.film_id) AS film_count FROM sakila.category, ...	16 row(s) returned	0.016 sec / 0.000 sec

## 6. Subqueries:

- List the names of customers who have rented more than 30 movies.

The screenshot shows the MySQL Workbench interface with a query editor window titled "SQL File 6". The code is as follows:

```
-- 6. Subqueries:  
-- List the names of customers who have rented more than 30 movies.  
74 • SELECT first_name, last_name  
    FROM sakila.customer  
    WHERE customer_id IN (  
        SELECT customer_id  
        FROM sakila.rental  
        GROUP BY customer_id  
        HAVING COUNT(*) > 30  
    );  
81  
82
```

The results grid shows the following data:

first_name	last_name
MARY	SMITH
ELIZABETH	BROWN
MARIA	MILLER
HELEN	HARRIS
MICHELLE	CLARK
JESSICA	HALL
SHIRLEY	ALLEN
CYNTHIA	YOUNG
ANGELA	HERNANDEZ
MELISSA	KING
VIRGINIA	GREEN
MARTHA	GONZALEZ

The output pane shows the execution log:

```
# Time Action Message  
10 21:52:16 SELECT category_name AS category_name, COUNT(film.film_id) AS film_count FROM sakila.category ... 16 row(s) returned  
11 21:52:16 SELECT first_name, last_name FROM sakila.customer WHERE customer_id IN ( ... ) 134 row(s) returned
```

- Find all films that have never been rented.

The screenshot shows the MySQL Workbench interface with a query editor window titled "SQL File 6". The code is as follows:

```
-- Find all films that have never been rented.  
83 • SELECT title  
    FROM sakila.film  
    WHERE film_id NOT IN (  
        SELECT film_id  
        FROM sakila.inventory  
        WHERE inventory_id IN (  
            SELECT inventory_id  
            FROM sakila.rental  
        )  
    );  
93  
94
```

The results grid shows the following data:

title
ALICE FANTASIA
APOLLO TEEN
ARMCHAIR TOWN
ARK RIDEABOUT
ARMSTRONG INDEPENDENCE
BODDOCK BALROOM
BUTCH PANTHER
CATCH AMISTAD
CHINATOWN GLADIATOR
CHOCOLATE DUCK
COMMANDMENTS EXPRESS

The output pane shows the execution log:

```
# Time Action Message  
22 21:54:55 SELECT first_name, last_name FROM sakila.customer WHERE customer_id IN ( ... ) 134 row(s) returned  
23 21:54:55 SELECT title FROM sakila.film WHERE film_id NOT IN ( ... ) 42 row(s) returned
```

## 7. Case Statements

- Categorize movies based on their length

The screenshot shows the MySQL Workbench interface. In the SQL editor tab, a query is written to categorize movie lengths:

```
97 • SELECT title, length,
98   CASE
99     WHEN length < 60 THEN 'Short'
100    WHEN length BETWEEN 60 AND 120 THEN 'Medium'
101    ELSE 'Long'
102  END AS length_category
103 FROM sakila.film;
```

The results grid displays the title, length, and length\_category for various movies. The output pane shows the execution log:

```
12 21:57:56 SELECT title, length, CASE WHEN length < 60 THEN 'Short' WHEN length BETWEEN 60 ... 42 row(s) returned
13 21:57:56 SELECT title, length, CASE WHEN length < 60 THEN 'Short' WHEN length BETWEEN 60 ... 1000 row(s) returned
```

## Part 3: Database Management

### 8. Insert Data

- Add a new customer to the customer table with relevant information.

The screenshot shows the MySQL Workbench interface. In the SQL editor tab, a query is used to insert a new customer record:

```
-- Part 3: Database Management
-- 8. Insert Data
-- Add a new customer to the customer table with relevant information.
110 • INSERT INTO customer (store_id, first_name, last_name, email, address_id, active, create_date)
VALUES (1, 'Yeab', 'Chhunsann', 'chhunsann.hd@gmail.com', 5, 1, NOW());
```

The results grid shows the newly inserted customer record. The output pane shows the execution log:

```
61 23:06:41 INSERT INTO customer (store_id, first_name, last_name, email, address_id, active, create_date) VALUES ('1', 'Yeab', 'Chhunsann', 'chhunsann.hd@gmail.com', 5, 1, '2024-12-22 23:06:41') 1 row(s) affected
62 23:06:41 SELECT * FROM sakila.customer LIMIT 0, 1000 600 row(s) returned
```

- Insert a new film into the film table, specifying appropriate values for all columns.

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following code:

```

112
113 -- SELECT * FROM sakila.customer;
114 -- Insert a new film into the film table, specifying appropriate values for all columns.
115
116 • INSERT INTO sakila.film (title, description, release_year, language_id, rental_duration, rental_rate, length, replacement_cost, rating, special_features, last_update)
VALUES ('Zyber Zu!', 'An exciting new movie.', 2024, 1, 3, 4.99, 130, 19.99, 'PG', 'Trailers,Deleted Scenes', NOW());
117
118 • SELECT * FROM sakila.film LIMIT 0, 1010;
119
120
121
122
123

```

The Result Grid shows the first few rows of the film table:

film_id	title	description	release_year	language_id	original_language_id	rental_duration	rental_rate	length	replacement_cost	rating	special_features	last_update
993	WRONG BEHAVIOR	A Emotional Saga of a Crocodile And a Sumo Wr...	2006	1		6	2.99					2006-02-14 22:04:37
994	WYOMING STORM	A Awe-Inspiring Panorama of a Robot And a Bo...	2006	1		6	4.99					2006-02-14 22:04:37
995	YENTL DAHO	A Amazing Display of a Robot And a Astronaut ...	2006	1		5	4.99					2006-02-14 22:04:37
996	YOUNG LANGUAGE	A Unbelievable Yarn of a Boat And a Database...	2006	1		6	0.99					2006-02-14 22:04:37
997	YOUTH KICK	A Touching Drama of a Teacher And a Cat who ...	2006	1		4	0.99					2006-02-14 22:04:37
998	ZHIVAGO CORE	A Fateful Yarn of a Composer And a Man who m...	2006	1		6	0.99					2006-02-14 22:04:37
999	ZOOLANDER FICTION	A Fateful Reflection of a Waitress And a boat w...	2006	1		5	2.99					2006-02-14 22:04:37
1000	ZORRO ARK	A Intrepid Panorama of a Mad Scientist And a B...	2006	1		3	4.99					2006-02-14 22:04:37
1001	Tom Teav	An exciting new movie.	2024	1		3	4.99					2024-12-22 23:06:41

The Action Output shows two recent actions:

- 227 23:41:15 DELETE FROM film WHERE title = 'Tom Teav' and last\_update ... 1 row(s) affected
- 228 23:41:15 SELECT \* FROM sakila.film WHERE title = 'Tom Teav' and last\_update ... 0 row(s) returned

## 9. Update Data:

- Update the email of a specific customer in the customer table.

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following code:

```

124 -- Update the email of a specific customer in the customer table.
125
126 • UPDATE sakila.customer
127 SET email = 'bongsann@gmail.com'
128 WHERE customer_id = 603; -- The customer that just added.
129
130 • select * from sakila.customer;
131
132
133

```

The Result Grid shows the customer table:

customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
591	1	KENT	ARSENAULT	KENT.ARSENAULT@sakilacustomer.org	597	1	2006-02-14 22:04:37	2006-02-15 04:57:20
592	1	TERRANCE	ROUSH	TERRANCE.ROUSH@sakilacustomer.org	598	0	2006-02-14 22:04:37	2006-02-15 04:57:20
593	2	RENE	MCALISTER	RENE.MCALISTER@sakilacustomer.org	599	1	2006-02-14 22:04:37	2006-02-15 04:57:20
594	1	EDUARDO	HIATT	EDUARDO.HIATT@sakilacustomer.org	600	1	2006-02-14 22:04:37	2006-02-15 04:57:20
595	1	TERRENCE	GUNDERSON	TERRENCE.GUNDERSON@sakilacustome...	601	1	2006-02-14 22:04:37	2006-02-15 04:57:20
596	1	ENRIQUE	FORSYTHE	ENRIQUE.FORSYTHE@sakilacustomer.org	602	1	2006-02-14 22:04:37	2006-02-15 04:57:20
597	1	FREDIE	DUGGAN	FREDIE.DUGGAN@sakilacustomer.org	603	1	2006-02-14 22:04:37	2006-02-15 04:57:20
598	1	WADE	DELVALLE	WADE.DELVALLE@sakilacustomer.org	604	1	2006-02-14 22:04:37	2006-02-15 04:57:20
599	2	AUSTIN	CINTRON	AUSTIN.CINTRON@sakilacustomer.org	605	1	2006-02-14 22:04:37	2006-02-15 04:57:20
603	1	Yeab	Chhunsoen	bongsann@gmail.com	5	1	2024-12-22 23:06:41	2024-12-23 16:55:58

The Action Output shows two recent actions:

- 30 16:56:32 UPDATE sakila.customer SET email = 'bongsann@gmail.com' WHERE customer\_id = 603 0 rows affected Rows matched: 1 Changed: 0 Warnings: 0
- 31 16:56:32 select \* from sakila.customer LIMIT 0, 1000 600 row(s) returned

- Change the rental rate for all movies in the film table to 4.99 where the length is greater than 120 minutes

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator Schemas SQL File 1* film customer
Filter objects
SCHEMAS ap sakila
Tables actor address category city country customer film film_actor film_category film_text inventory language payment rental staff store
Views Stored Procedures Functions sys
Administration Schemas Information
No object selected
File 28 x
Output
Action Output
# Time Action Message Duration / Fetch
1 17:08:44 SET SQL_SAFE_UPDATES = 0 0 row(s) affected 0.000 sec
2 17:08:44 UPDATE film SET rental_rate = 4.99 WHERE length > 120 300 row(s) affected Rows matched: 458 Changed: 300 Warnings: 0 0.078 sec
3 17:08:44 SET SQL_SAFE_UPDATES = 1 0 row(s) affected 0.000 sec
4 17:08:44 select * from sakila.film where length > 120; 458 row(s) returned 0.000 sec / 0.000 sec
Object Info Session
Query Completed

```

The screenshot shows the MySQL Workbench interface. The SQL Editor tab has the following content:

```

130 -- select * from sakila.customer;
131
132 -- Change the rental rate for all movies in the film table to 4.99 where the length is greater than 120 minutes
133 • SET SQL_SAFE_UPDATES = 0;
134
135 • UPDATE film
136     SET rental_rate = 4.99
137     WHERE length > 120;
138
139 • SET SQL_SAFE_UPDATES = 1; -- Re-enable safe update mode after the operation
140
141
142 • select * from sakila.film where length > 120;
143

```

The Result Grid shows the following data:

film_id	title	description	release_year	language_id	original_language_id	rental_duration	rental_rate	length	replacement_cost
5	AFRICAN EGG	A Fast-Paced Documentary of a Pastry Chef An...	2006	1	MAX	6	4.99	130	22.99
6	AGENT TRUMAN	A Intrepid Panorama of a Robot And a Boy who...	2006	1	MAX	3	4.99	169	17.99
11	ALAMO VIDEOTAPE	A Boring Esquire of a Butler And a Cat who ...	2006	1	MAX	6	4.99	126	16.99
12	ALASKA PHANTOM	A Farcical Saga of a Hunter And a Pastry Chef...	2006	1	MAX	6	4.99	136	22.99

## 10. Delete Data:

- Delete a customer from the customer table who hasn't rented any movies

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator Schemas SQL File 1* film customer SQL File 4* rental
Filter objects
SCHEMAS ap sakila
Tables actor address category city country customer film film_actor film_category film_text inventory language payment rental staff store
Views Stored Procedures Functions sys
Administration Schemas Information
No object selected
File 29 x
Output
Action Output
# Time Action Message Duration / Fetch
1 17:22:03 SELECT * FROM sakila.rental LIMIT 0, 1000 1000 row(s) returned 0.000 sec / 0.000 sec
2 17:25:00 SET SQL_SAFE_UPDATES = 0 0 row(s) affected 0.000 sec
3 17:25:00 DELETE FROM customer WHERE customer_id NOT IN ( SELECT DISTINCT customer_id FROM rental ) 1 row(s) affected 0.016 sec
4 17:25:00 SET SQL_SAFE_UPDATES = 1 0 row(s) affected 0.000 sec
Object Info Session
Query Completed

```

The screenshot shows the MySQL Workbench interface. The SQL Editor tab has the following content:

```

141 -- select * from sakila.film where length > 120;
142
143
144 -- 10. Delete Data:
145 -- Delete a customer from the customer table who hasn't rented any movies.
146
147 • SET SQL_SAFE_UPDATES = 0;
148 • DELETE FROM customer
149     WHERE customer_id NOT IN (
150         SELECT DISTINCT customer_id
151         FROM rental
152     );
153 • SET SQL_SAFE_UPDATES = 1;
154
155
156
157
158
159
160
161

```

- The verification after delete a customer table who hasn't rented any movies

The screenshot shows the MySQL Workbench interface. In the top navigation bar, the database is set to 'mysql'. The left sidebar shows the schema 'sakila' with various tables listed. The main area has four tabs: 'SQL File 6\*', 'film', 'customer', and 'SQL File 4\*' which contains the following SQL code:

```

1 • SELECT *
2   FROM customer
3   WHERE customer_id NOT IN (
4     SELECT DISTINCT customer_id
5       FROM rental
6   );
7

```

Below the code, there is a 'Result Grid' table with columns: customer\_id, store\_id, first\_name, last\_name, email, address\_id, active, create\_date, last\_update. The grid is currently empty.

In the bottom right corner of the main window, there is an 'Output' section titled 'Action Output' showing the execution history:

#	Time	Action	Message	Duration / Fetch
9	17:25:00	DELETE FROM customer WHERE customer_id NOT IN ( SELECT DISTINCT customer_id FROM rental )	1 row(s) affected	0.016 sec
10	17:25:00	SET SQL_SAFE_UPDATES = 1	0 row(s) affected	0.000 sec
11	17:25:49	SELECT c.customer_id, c.first_name, c.last_name, c.email FROM customer c LEFT JOIN rental r ON c.customer_id = r.customer_id WHERE r.rental_date < '2005-05-22' AND r.returned_date IS NULL	0 row(s) returned	0.015 sec / 0.000 sec
12	17:30:39	SELECT * FROM customer WHERE customer_id NOT IN ( SELECT DISTINCT customer_id FROM rental )	0 row(s) returned	0.016 sec / 0.000 sec

The status bar at the bottom indicates 'Query Completed'.

- Remove all movies in the film table that were released before 2000

The screenshot shows the MySQL Workbench interface. In the top navigation bar, the database is set to 'mysql'. The left sidebar shows the schema 'sakila' with various tables listed. The main area has four tabs: 'SQL File 6\*', 'film', 'customer', and 'SQL File 4\*' which contains the following SQL code:

```

155 -- Remove all movies in the film table that were released before 2000
156 • set SQL_SAFE_UPDATES = 0;
157
158 • DELETE FROM film
159   WHERE release_year < 2000;
160
161 • SET SQL_SAFE_UPDATES = 1;
162
163
164 • select * from film where release_year < 2000;
165
166
167
168

```

Below the code, there is a 'Result Grid' table with columns: film\_id, title, description, release\_year, language\_id, original\_language\_id, rental\_duration, rental\_rate, length, replacement\_cost, rating, special\_features, last\_update. The grid is currently empty.

In the bottom right corner of the main window, there is an 'Output' section titled 'Action Output' showing the execution history:

#	Time	Action	Message	Duration / Fetch
13	17:34:09	set SQL_SAFE_UPDATES = 0	0 row(s) affected	0.000 sec
14	17:34:09	DELETE FROM film WHERE release_year < 2000	0 row(s) affected	0.000 sec
15	17:34:09	SET SQL_SAFE_UPDATES = 1	0 row(s) affected	0.000 sec
16	17:34:09	select * from film where release_year < 2000 LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec

The status bar at the bottom indicates 'Query Completed'.

## 11. Create a Report:

- Generate a report showing the top 10 customers who rented the most movies. Include customer\_id, first\_name, last\_name, and the number of movies rented.

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema **sakila** with various tables like actor, address, category, city, country, customer, film, film\_actor, film\_category, film\_text, inventory, language, payment, rental, staff, and store.
- SQL Editor:** Contains the following SQL query:

```
-- 11. Create a Report:  
-- Generate a report showing the top 10 customers who rented the most movies. Include customer_id, first_name, last_name, and the number of movies rented.  
SELECT customer_id, first_name, last_name, COUNT(*) AS total_rentals  
FROM rental  
JOIN customer USING (customer_id)  
GROUP BY customer_id  
ORDER BY total_rentals DESC  
LIMIT 10;
```
- Result Grid:** Displays the results of the query:

customer_id	first_name	last_name	total_rentals
148	ELEANOR	HUNT	46
526	KARL	SEAL	45
144	CLARA	SHAW	42
236	MARICIA	DEAN	42
75	TAMMY	SANDERS	41
197	SUE	PETERS	40
469	WESLEY	BULL	40
178	MARION	SNYDER	39
137	RHONDA	KENNEDY	39
468	TM	CARY	39
- Action Output:** Shows a single delete operation:

```
14 17:34:09 DELETE FROM film WHERE release_year < 2000
```
- Object Info:** Session
- Session Status:** Read Only
- Message:** 0 row(s) affected
- Duration / Fetch:** 0.000 sec

## 12. Revenue Analysis:

- Calculate the total revenue generated per category. Display the category name and total revenue.

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema **sakila** with various tables like actor, address, category, city, country, customer, film, film\_actor, film\_category, film\_text, inventory, language, payment, rental, staff, and store.
- SQL Editor:** Contains the following SQL query:

```
-- 12. Revenue Analysis:  
-- Calculate the total revenue generated per category. Display the category name and total revenue.  
SELECT name AS category_name, SUM(amount) AS total_revenue  
FROM payment  
JOIN rental USING (rental_id)  
JOIN inventory USING (inventory_id)  
JOIN film USING (film_id)  
JOIN film_category USING (film_id)  
JOIN category USING (category_id)  
GROUP BY category_id  
ORDER BY total_revenue DESC;
```
- Result Grid:** Displays the results of the query:

category_name	total_revenue
Sports	5314.21
Sci-Fi	4756.98
Animation	4656.30
Drama	4587.39
Comedy	4383.58
Action	4375.85
New	4351.62
Games	4281.33
Foreign	4270.67
Family	4226.07
Documentary	4217.52
- Action Output:** Shows a single select operation:

```
17 17:46:37 SELECT customer_id, first_name, last_name, COUNT(*) AS total_rentals FROM rental JOIN customer USING (...
```
- Object Info:** Session
- Session Status:** Read Only
- Message:** 10 row(s) returned
- Duration / Fetch:** 0.015 sec / 0.000 sec

## 13. Rental Trends:

- Identify the busiest rental month. Display the month and the total number of rentals during that month.

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
189
190 -- 13. Rental Trends:
191 -- Identify the busiest rental month. Display the month and the total number of rentals during that month.
192
193 • SELECT MONTHNAME(rental_date) AS rental_month, COUNT(*) AS total_rentals
194   FROM rental
195   GROUP BY rental_month
196   ORDER BY total_rentals DESC
197   LIMIT 1;
198
199
```

The Result Grid shows the output:

rental_month	total_rentals
July	6709

The Action Output pane shows the execution details:

#	Time	Action
1	18:00:39	SELECT MONTHNAME(rental_date) AS rental_month, COUNT(*) AS total_rentals FROM rental GROUP BY rental ORDER BY total_rentals DESC LIMIT 1;

Message: 1 row(s) returned

Duration / Fetch: 0.016 sec / 0.000 sec

## Part 5: Views, Indexes, and Stored Procedures

### 14. Create Views:

- Create a view customer\_rentals that displays customer information along with the titles of movies they rented.

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query to create a view:

```
201 -- Create a view customer_rentals that displays customer information along with the titles of movies they rented.
202 CREATE VIEW customer_rentals AS
203   SELECT
204     c.customer_id,
205     c.first_name,
206     c.last_name,
207     f.title
208   FROM customer c
209   JOIN rental r ON c.customer_id = r.customer_id
210   JOIN inventory i ON r.inventory_id = i.inventory_id
211   JOIN film f ON i.film_id = f.film_id;
212
213 -- This is for displaying the customer information along with the titles of movies they rented.
214 • SELECT * FROM sakila.customer_rentals;
215
```

The Result Grid shows the output:

customer_id	first_name	last_name	title
431	JOEL	FRANCISCO	ACADEMY DINOSAUR
518	GABRIEL	HARDER	ACADEMY DINOSAUR
279	DIANNE	SHELTON	ACADEMY DINOSAUR
411	NORMAN	CURRIER	ACADEMY DINOSAUR
170	BEATRICE	ARNOLD	ACADEMY DINOSAUR
161	GERALDINE	PERKINS	ACADEMY DINOSAUR
581	VIRGIL	WOPFORD	ACADEMY DINOSAUR
359	WILLIE	MARSHAM	ACADEMY DINOSAUR
no record	NAOMI	ACADEMY DINOSAUR	

The Action Output pane shows the execution details:

#	Time	Action
1	21:37:13	CREATE VIEW customer_rentals AS SELECT c.customer_id, c.first_name, c.last_name, f.title FROM customer c JOIN rental r ON c.customer_id = r.customer_id JOIN inventory i ON r.inventory_id = i.inventory_id JOIN film f ON i.film_id = f.film_id;
2	21:37:13	SELECT * FROM sakila.customer_rentals LIMIT 0, 1000

Message: 0 row(s) affected

Message: 1000 row(s) returned

Duration / Fetch: 0.062 sec / 0.000 sec

- Create a view category\_revenue that shows the total revenue generated by each category.

The screenshot shows the MySQL Workbench interface with the SQL Editor tab active. The code pane contains the following SQL:

```

216 -- Create a view category_revenue that shows the total revenue generated by each category.
217 CREATE VIEW category_revenue AS
218     SELECT
219         c.name AS category_name,
220         SUM(p.amount) AS total_revenue
221     FROM payment p
222     JOIN rental r USING (rental_id)
223     JOIN inventory i USING (inventory_id)
224     JOIN film f USING (film_id)
225     JOIN film_category fc USING (film_id)
226     JOIN category c USING (category_id)
227     GROUP BY c.category_id;
228
229 -- This is for displaying the total revenue generated by each category.
230 • SELECT * FROM sakila.category_revenue;

```

The Result Grid shows the data from the view:

category_name	total_revenue
Action	4375.85
Animation	4656.30
Children	3655.55
Classics	3639.59
Comedy	4383.58
Documentary	4217.52
Drama	4587.39
Family	4226.07
Thriller	4777.47

The Output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
1	21:42:23	CREATE VIEW category_revenue AS SELECT c.name AS category_name, SUM(p.amount) AS total_revenue FROM payment p JOIN rental r USING (rental_id) JOIN inventory i USING (inventory_id) JOIN film f USING (film_id) JOIN film_category fc USING (film_id) JOIN category c USING (category_id) GROUP BY c.category_id;	0 row(s) affected	0.094 sec
2	21:42:24	SELECT * FROM sakila.category_revenue	16 row(s) returned	0.187 sec / 0.000 sec

## 15. Create Indexes:

- Create an index on the title column of the film table to optimize search queries.

The screenshot shows the MySQL Workbench interface with the SQL Editor tab active. The code pane contains the following SQL:

```

225 -- JOIN film_category fc USING (film_id)
226 -- JOIN category c USING (category_id);
227 -- GROUP BY c.category_id;
228
229 -- This is for displaying the total revenue generated by each category.
230 -- SELECT * FROM sakila.category_revenue;
231
232 -- 15. Create Indexes
233 -- Create an index on the title column of the film table to optimize search queries.
234 • CREATE INDEX idx_film_title ON film(title);
235
236 -- To verify we created successful.
237 • SHOW INDEX FROM film;
238
239

```

The Result Grid shows the index information:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible
film	0	PRIMARY	1	film_id	A	1000				BTREE			YES
film	1	idx_film	1	title	A	1000				BTREE			YES
film	1	idx_film_language_id	1	language_id	A	1				BTREE			YES
film	1	idx_film_original_language_id	1	original_language_id	A	1			YES	BTREE			YES
film	1	idx_film_title	1	title	A	1000				BTREE			YES

The Output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
3	21:49:03	CREATE INDEX idx_film_title ON film(title)	0 row(s) affected, 1 warning(s), 1831 Duplicate index 'idx_film_title' defined on the table 'sakila.film'. This is degree	0.125 sec
4	21:49:03	SHOW INDEX FROM film	5 row(s) returned	0.016 sec / 0.000 sec

- Add an index to the rental\_date column of the rental table to speed up date-based searches.

The screenshot shows the MySQL Workbench interface with the SQL tab selected. The code editor contains the following SQL script:

```

234 -- CREATE INDEX idx_file_title ON film(title);
235
236 -- To verify that we created successful.
237 -- SHOW INDEX FROM film;
238
239
240 -- Add an index to the rental_date column of the rental table to speed up date-based searches.
241 • CREATE INDEX idx_rental_date ON rental(rental_date);
242
243 -- To verify that we created successful.
244 • SHOW INDEX FROM rental;
245
246
247
248

```

The Result Grid shows the execution of the last two commands:

Table	Non_Unique	Key_name	Seq_in_Index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Express
rental	0	rental_date	1	rental_date	A	15815			NULL	BTREE		YES	NULL	
rental	0	rental_date	2	inventory_id	A	16012			NULL	BTREE		YES	NULL	
rental	0	rental_date	3	customer_id	A	16012			NULL	BTREE		YES	NULL	
rental	1	idx_fk_inventory_id	1	inventory_id	A	4580			NULL	BTREE		YES	NULL	
rental	1	idx_fk_customer_id	1	customer_id	A	599			NULL	BTREE		YES	NULL	
rental	1	idx_fk_staff_id	1	staff_id	A	2			NULL	BTREE		YES	NULL	
rental	1	idx_rental_date	1	rental_date	A	15815			NULL	BTREE		YES	NULL	

The Action Output pane shows the execution results:

```

# Time Action
5 21:59:18 CREATE INDEX idx_rental_date ON rental(rental_date)
6 21:59:18 SHOW INDEX FROM rental

```

## 16. Stored Procedure:

- Write a stored procedure get\_customer\_rentals that accepts a customer\_id as input and returns the list of movies rented by that customer.

The screenshot shows the MySQL Workbench interface with the SQL tab selected. The code editor contains the following SQL script:

```

246 -- 16. Stored Procedure.
247 -- Write a stored procedure get_customer_rentals that accepts a customer_id as input and returns the list of movies rented by that customer.
248 DELIMITER $$ 
249 • CREATE PROCEDURE get_customer_rentals(IN customer_id INT)
250 BEGIN
251   SELECT
252     f.title
253   FROM rental r
254   JOIN inventory i ON r.inventory_id = i.inventory_id
255   JOIN film f ON i.film_id = f.film_id
256   WHERE r.customer_id = customer_id;
257 END $$
258 DELIMITER ;
259 • -- To Execute the Procedure:
260 CALL get_customer_rentals(1); -- Replace '1' with the desired customer_id

```

The Result Grid shows the output of the SELECT statement:

title
PATIENT SISTER
TALENTED HOMICIDE
MUSKeteERS WAIT
DETECTIVE VISION
FERRIS MOTHER
CLOSER BANG
ATTACKS HATE
SAVANNAH TOWN

The Action Output pane shows the execution results:

```

# Time Action
12 22:12:20 CREATE PROCEDURE get_customer_rentals(IN customer_id INT) BEGIN SELECT f.title FROM r...
13 22:12:20 - To Execute the Procedure: CALL get_customer_rentals(1)

```