# MID COURSE PROJECT

Server used: MySQL Community Server.

Client used: MySQL Workbench.

Database used: Maven Movies SQL database created using the SQL script “create\_mavenmovies.sql”.

The Situation: The company’s insurance policy is up for renewal and the insurance company’s underwriters need some updated information from us before they will issue a new policy.

The Objective: Use MySQL to:

Leverage your SQL skills to extract and analyse data from various tables in the Maven

Movies database to answer the underwriters’ questions. Each question can be answered

by querying just one table. Part of your job as an Analyst is figuring out which table to use.

The Letter: Dear Maven Movies Management,

In our review of your policy renewal application, we have realized that your business information has not been updated for several years.

To accurately assess the risk and approve your policy renewal, we will need you to provide all of the following information.

Sincerely,

Joe Scardycat, Lead Underwriter

Mid-Course Project Questions and Answers:

Q1. We will need a list of all staff members, including their first and last names, email addresses, and the store identification number where they work.

### SQL Code:

SELECT

first\_name,

last\_name,

email,

store\_id

FROM staff;

### Screenshot Of Output:

Graphical user interface, application

Description automatically generated

Q2. We will need separate counts of inventory items held at each of your two stores.

### SQL Code:

SELECT

store\_id,

COUNT(inventory\_id) AS Inventory\_Count

FROM

inventory

GROUP BY

store\_id

ORDER BY

store\_id;

### Screenshot Of Output:

Graphical user interface, application, table

Description automatically generated

Q3. We will need a count of active customers for each of your stores. Separately, please.

### SQL Code:

SELECT

store\_id,

COUNT(customer\_id) AS Active\_Customer\_Count

FROM

customer

WHERE

active = 1

GROUP BY

store\_id

ORDER BY

store\_id;

### Screenshot Of Output:

Graphical user interface, application, table

Description automatically generated

Q4. To assess the liability of a data breach, we will need you to provide a count of all customer email addresses stored in the database.

### SQL Code:

SELECT

COUNT(email) AS Number\_Of\_Email\_Addresses

FROM

customer;

### Screenshot Of Output:

Graphical user interface, application

Description automatically generated

Q5. We are interested in how diverse your film offering is as a means of understanding how likely you are to keep customers engaged in the future. Please provide a count of unique film titles you have in inventory at each store and then provide a count of the unique categories of films you provide.

### Part-I:

### SQL Code:

SELECT

store\_id,

COUNT(DISTINCT film\_id) AS Unique\_Films

FROM

inventory

GROUP BY

store\_id

ORDER BY

store\_id;

### Screenshot Of Output:

Graphical user interface, text, application

Description automatically generated

### Part-II:

### SQL Code:

SELECT

COUNT(DISTINCT category\_id) AS Unique\_Categories

FROM

category;

### Screenshot Of Output:

Graphical user interface, application

Description automatically generated

Q6. We would like to understand the replacement cost of your films. Please provide the replacement cost for the film that is least expensive to replace, the most expensive to replace, and the average of all films you carry.

### SQL Code:

SELECT

MIN(replacement\_cost) AS "Least expensive film to replace",

MAX(replacement\_cost) AS "Least expensive film to replace",

AVG(replacement\_cost) AS "Least expensive film to replace"

FROM

film;

### Screenshot Of Output:

Graphical user interface, application

Description automatically generated

Q7. We are interested in having you put payment monitoring systems and maximum payment processing restrictions in place to minimize the future risk of fraud by your staff. Please provide the average payment you process, as well as the maximum payment you have processed.

### SQL Code:

SELECT

AVG(amount) AS "Average Payment processed",

MAX(amount) AS "Maximum Payment processed"

FROM payment;

### Screenshot Of Output:

Graphical user interface, application, table

Description automatically generated

Q8. We would like to better understand what your customer base looks like. Please provide a list of all customer identification values, with a count of rentals they have made all-time, with your highest volume customers at the top of the list.

### SQL Code:

SELECT

customer\_id,

COUNT(rental\_id) AS "Rental\_Count"

FROM rental

GROUP BY customer\_id

ORDER BY Rental\_Count DESC;

### Screenshot Of Output:

Table

Description automatically generated with medium confidence

Q9. I would like to know which store each customer goes to, and whether they are active or not.

Could you pull a list of first and last names of all customers, and label them as either ‘store 1 active’, ‘store 1 inactive’, ‘store 2 active’, or ‘store 2 inactive’?”

### SQL Code:

SELECT

first\_name,

last\_name,

CASE

WHEN store\_id=1 AND active=1 THEN "store 1 active"

WHEN store\_id=1 AND active=0 THEN "store 1 inactive"

WHEN store\_id=2 AND active=1 THEN "store 2 active"

WHEN store\_id=2 AND active=0 THEN "store 2 inactive"

ELSE "It is not captured"

END AS "Store\_and\_Status"

FROM customer;

### Screenshot Of Output:

Graphical user interface, table, Excel

Description automatically generated

Q10. “I’m curious how many inactive customers we have at each store.

Could you please create a table to count the number of customers broken down by store\_id (in rows), and active status (in columns)?”

### SQL Code:

SELECT

store\_id,

COUNT(CASE WHEN active=1 THEN customer\_id ELSE NULL END) AS "Active\_Customer\_Count",

COUNT(CASE WHEN active=0 THEN customer\_id ELSE NULL END) AS "InActive\_Customer\_Count"

FROM customer

GROUP BY store\_id

### Screenshot Of Output:

Graphical user interface, application, Word

Description automatically generated

Lessons Learnt:

1. “Big 6” Statement and Clauses of SQL queries
   1. SELECT (Mandatory)
   2. FROM (Mandatory)
   3. WHERE (Optional by default)
   4. GROUP BY (Optional by default but will be required if any aggregation is carried out)
   5. HAVING (Optional by default but will can be used only if GROUP BY is present)
   6. ORDER BY (Optional by default)
2. SELECT DISTINCT
3. AS (used to provide an alias to the columns being queried)
4. WHERE (with Logical operators like AND & OR)
5. WHERE (with IN, BETWEEN, LIKE)
6. Grouping with Aggregate Functions
   1. COUNT
   2. COUNT(DISTINCT)
   3. SUM
   4. AVG
   5. MIN
   6. MAX
7. CASE
8. “Pivoting” with COUNT & CASE