# FINAL 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: You and your business partner were recently approached by another local business owner who is interested in purchasing Maven Movies. He primarily owns restaurants and bars, so he

has lots of questions for you about your business and the rental business in general. His offer

seems very generous, so you are going to entertain his questions.

The Objective: Use MySQL to:

Leverage your SQL skills to extract and analyze data from various tables in the Maven Movies database to answer your potential Acquirer’s questions. Each question will require you to write a multi-table SQL query, joining at least two tables.

The Letter: Dear Maven Movies Management,

I am excited about the potential acquisition and learning more about your rental business.

Please bear with me as I am new to the industry, but I have several questions for you. Assuming you can answer them all, and that there are no major surprises, we should be able to move forward with the purchase.

Best,

Martin Moneybags

Final Project Questions and Answers:

Q1. My partner and I want to come by each of the stores in person and meet the managers.

Please send over the managers’ names at each store, with the full address of each property (street address, district, city, and country please).

### SQL Code:

SELECT

staff.first\_name AS "Manager\_First\_Name",

staff.last\_name AS "Manager\_Last\_Name",

address.address AS "Full\_Address",

address.district AS "District",

city.city AS "City",

country.country AS "Country"

FROM store

LEFT JOIN staff

ON store.manager\_staff\_id = staff.staff\_id

LEFT JOIN address

ON staff.address\_id = address.address\_id

LEFT JOIN city

ON address.city\_id = city.city\_id

LEFT JOIN country

ON city.country\_id = country.country\_id

### Screenshot Of Output:

Graphical user interface, application

Description automatically generated

Q2. I would like to get a better understanding of all the inventory that would come along with the business. Please pull together a list of each inventory item you have stocked, including the store\_id number, the inventory\_id, the name of the film, the film’s rating, its rental rate and replacement cost.

### SQL Code:

SELECT

I.store\_id,

I.inventory\_id,

F.title,

F.rating,

F.replacement\_cost

FROM inventory AS I

LEFT JOIN film AS F

ON I.film\_id = F.film\_ID

LIMIT 5000;

### Screenshot Of Output:

Graphical user interface, text, application

Description automatically generated

Q3. From the same list of films, you just pulled, please roll that data up and provide a summary level overview of your inventory. We would like to know how many inventory items you have with each rating at each store.

### SQL Code:

SELECT

I.store\_id,

F.rating,

COUNT(I.inventory\_id) AS "Number\_Of\_Inventory\_Items"

FROM inventory AS I

LEFT JOIN film AS F

ON I.film\_id = F.film\_ID

GROUP BY

I.store\_id,

F.rating;

### Screenshot Of Output:

Table

Description automatically generated

Q4. Similarly, we want to understand how diversified the inventory is in terms of replacement cost. We want to see how big of a hit it would be if a certain category of film became unpopular at a certain store. We would like to see the number of films, as well as the average replacement cost, and total replacement cost, sliced by store and film category.

### SQL Code:

SELECT

store\_id,

category.name AS "Category\_Name",

COUNT(inventory.inventory\_id) AS "Number\_Of\_Films",

AVG(film.replacement\_cost) AS "Average\_Replacement\_Cost",

SUM(film.replacement\_cost) AS "Total\_Replacement\_Cost"

FROM inventory

LEFT JOIN film

ON inventory.film\_id = film.film\_id

LEFT JOIN film\_category

ON film.film\_id = film\_category.film\_id

LEFT JOIN category

ON category.category\_id = film\_category.category\_id

GROUP BY

store\_id,

category.name

ORDER BY

SUM(film.replacement\_cost) DESC;

### Screenshot Of Output:

Table

Description automatically generated

Q5. We want to make sure you folks have a good handle on who your customers are. Please provide a list of all customer names, which store they go to, whether or not they are currently active, and their full addresses – street address, city, and country.

### SQL Code:

SELECT

customer.first\_name AS "Customer\_First\_Name",

customer.last\_name AS "Customer\_Last\_Name",

customer.store\_id AS "Store\_Name",

customer.active AS "Active\_Status",

address.address AS "Full\_Address",

city.city "City\_Name",

country.country AS "Country\_Name"

FROM customer

LEFT JOIN address

ON customer.address\_id = address.address\_id

LEFT JOIN city

ON address.city\_id = city.city\_id

LEFT JOIN country

ON city.country\_id = country.country\_id

### Screenshot Of Output:

Table

Description automatically generated

Q6. We would like to understand how much your customers are spending with you, and to know who your most valuable customers are. Please pull together a list of customer names, their total lifetime rentals, and the sum of all payments you have collected from them. It would be great to see this ordered on total lifetime value, with the most valuable customers at the top of the list.

### SQL Code:

SELECT

customer.first\_name AS "Customer\_First\_Name",

customer.last\_name AS "Customer\_Last\_Name",

COUNT(rental.rental\_id) AS "Total\_Number\_Of\_Rentals",

SUM(payment.amount) AS "Total\_Amount\_Collected"

FROM customer

LEFT JOIN rental

ON customer.customer\_id = rental.customer\_id

LEFT JOIN payment

ON rental.rental\_id = payment.rental\_id

GROUP BY

customer.first\_name,

customer.last\_name

ORDER BY

SUM(payment.amount) DESC;

### Screenshot Of Output:

Table

Description automatically generated

Q7. My partner and I would like to get to know your board of advisors and any current investors.

Could you please provide a list of advisor and investor names in one table?

Could you please note whether they are an investor or an advisor, and for the investors, it would be good to include which company they work with.

### SQL Code:

SELECT

"Investor" as "Type",

first\_name AS "First\_Name",

last\_name AS "Last\_Name",

company\_name AS "Company\_Name"

FROM investor

UNION

SELECT

"Advisor" as "Type",

first\_name AS "First\_Name",

last\_name AS "Last\_Name",

NULL

FROM advisor;

### Screenshot Of Output:

Graphical user interface, table

Description automatically generated with medium confidence

Q8. We are interested in how well you have covered the most-awarded actors. Of all the actors with three types of awards, for what % of them do we carry a film?

And how about for actors with two types of awards? Same questions.

Finally, how about actors with just one award?

### SQL Code:

SELECT

CASE

WHEN actor\_award.awards = 'Emmy, Oscar, Tony ' THEN '3 Awards'

WHEN actor\_award.awards IN ('Emmy, Oscar', 'Emmy, Tony', 'Oscar, Tony') THEN '2 Awards'

ELSE '1 Award'

END AS Number\_Of\_Awards,

AVG(CASE WHEN actor\_award.actor\_id is NULL THEN 0 ELSE 1 END) \* 100 AS "Percentage\_With\_Atleast\_1\_Film"

FROM actor\_award

GROUP BY

CASE

WHEN actor\_award.awards = 'Emmy, Oscar, Tony ' THEN '3 Awards'

WHEN actor\_award.awards IN ('Emmy, Oscar', 'Emmy, Tony', 'Oscar, Tony') THEN '2 Awards'

ELSE '1 Award'

END

### Screenshot Of Output:

Graphical user interface, application, table

Description automatically generated

Lessons Learnt:

1. Concepts of Normalization and Cardinality
2. Relationship Diagrams
3. Various types of JOINS (INNER, LEFT, RIGHT, FULL OUTER JOIN)
4. Bridging unrelated tables
5. Multi-condition JOINS
6. UNION and UNION ALL