# 1. Project Proposal / Synopsis

**Project Title:**
**SQL-Based Data Exploration and Analysis on Maven Movies Dataset**

**Objective:**
To utilize SQL to design, manipulate, and analyze a relational database (Maven Movies) to derive actionable insights like top customers, film rentals, category-wise performance, and revenue.

**Scope:**

- Use SQL DDL and DML
- Work with relational joins, constraints, views, and functions
- Perform normalization and query optimization

**Tools Used:**

- MySQL
- SQL Workbench / CLI
- Graphviz (for ER diagrams)

# 2. SRS – Software Requirements Specification

## 1. Introduction

- **Purpose:** Understand and analyze rental data using SQL
- **Dataset:** Sakila (Maven Movies)
- **Users:** Analysts, Admins

## 2. Functional Requirements

- Create and modify relational tables

- Run joins, aggregations, filters
- Normalize tables (1NF, 2NF)
- Create views, procedures, functions

### 3. Non-Functional Requirements

- Queries should return results within 2 seconds
- Database must maintain referential integrity
- All table definitions should include meaningful constraints

# 3. Database Design Document

### Key Tables:

- **Customer:** Stores customer info
- **Film:** Stores movie data
- **Rental:** Track movie rentals
- **Payment:** Payment history
- **Inventory:** Copies of films at each store

### Example: `customer`

```
CREATE TABLE customer (
  customer_id SMALLINT PRIMARY KEY AUTO_INCREMENT,
  first_name VARCHAR(45),
  last_name VARCHAR(45),
  email VARCHAR(50),
  address_id SMALLINT,
  create_date DATETIME,
  ...
);
```

### Relationships:

- `customer` → `rental` → `inventory` → `film`
- `payment` → `customer`, `rental`

## 4. ER Diagram

Included in diagram (see earlier). Describes relationships between:

- Films ↔ Actors
- Films ↔ Categories
- Rentals ↔ Inventory ↔ Films
- Customers ↔ Rentals ↔ Payments

## 5. SQL Queries Document

### Example Queries:

```
-- Top 5 customers by total payment
SELECT customer_id, SUM(amount) as total_paid
FROM payment
GROUP BY customer_id
ORDER BY total_paid DESC
LIMIT 5;

-- Number of rentals per film
SELECT f.title, COUNT(r.rental_id) as rentals
FROM film f
JOIN inventory i ON f.film_id = i.film_id
JOIN rental r ON i.inventory_id = r.inventory_id
GROUP BY f.title;

-- Films with no rentals
SELECT f.title
FROM film f
LEFT JOIN inventory i ON f.film_id = i.film_id
LEFT JOIN rental r ON i.inventory_id = r.inventory_id
```

```
WHERE r.rental_id IS NULL;
```

# 6. Normalization Report

### Example Violation of 1NF:

A table with multiple phone columns:

```
CREATE TABLE customer (
  customer_id INT,
  name VARCHAR(50),
  phone1 VARCHAR(15),
  phone2 VARCHAR(15)
);
```

### Normalized:

```
CREATE TABLE customer (
  customer_id INT,
  name VARCHAR(50)
);
CREATE TABLE customer_phone (
  customer_id INT,
  phone VARCHAR(15)
);
```

# 7. CTE and Window Function Report

### CTE Example:

```
WITH top_customers AS (
  SELECT customer_id, SUM(amount) AS total
  FROM payment
  GROUP BY customer_id
)
SELECT * FROM top_customers WHERE total > 100;
```

### Window Function Example:

```
SELECT customer_id, amount,
```

```
  RANK() OVER (ORDER BY amount DESC) AS rank
FROM payment;
```

# 8. Testing Report

| Test Case | SQL Used | Expected Output | Result |
|---|---|---|---|
| Insert customer with no store_id | FK constraint triggers | Error | ✅ |
| Query rentals per film | Uses JOIN, GROUP BY | List of counts | ✅ |
| Create default constraint | Set salary default to 30000 | Inserts succeed | ✅ |
| Invalid email uniqueness test | Duplicate email insert fails | Error | ✅ |

# 9. Results & Analysis

- **Top 5 customers** contribute ~25% of all payments
- **Action & Comedy** dominate rentals
- Stores in urban cities perform better
- Some films are never rented — inventory optimization needed

# 10. Conclusion & Future Scope

### Key Takeaways:

- SQL is powerful for structured data analysis
- ER design simplifies complex relationships
- Views & CTEs improve readability and maintainability

### Future Scope:

- Integrate with BI tools (Power BI, Tableau)
- Build a Streamlit or Flask dashboard for reports
- Add stored procedures for automation