**Real-Time Stock Analysis and Prediction**

**Using Python and Yahoo Finance**

**A Project Report**

Submitted in partial fulfillment of the requirements for the   
**Award of the degree of**

**Master of Computer Application**

**By**

**Md. Tarique Anwer**

**(322201297)**

****

**Centre for Distance and Online Education**

**LOVELY PROFESSIONAL UNIVERSITY**

**PHAGWARA, PUNJAB**

**2024**

# Declaration by the Student

**To whom-so-ever it may concern**

I, **MD TARIQUE ANWER, 322201297,** hereby declare that the work done by me on “**Real-Time Stock Analysis and Prediction using python and Yahoo finance**”, is a record of original work for the partial fulfillment of the requirements for the award of the degree, **Master of Computer Application.**

**MD TARIQUE ANWER (322201297)**

Signature of the student

Dated: 09/02/2021

# Acknowledgement

I would like to express my sincere gratitude to everyone who contributed to the successful completion of my project, "Real-Time Stock Analysis and Prediction using Python and Yahoo Finance."

I am grateful to the faculty members of the **Online Master of Computer Application**, **Lovely Professional University**, for providing me with the knowledge and resources necessary for this project. Their support and the academic environment they created have been crucial to my learning and development.

Lastly, I would like to extend my heartfelt thanks to my family for their unwavering support and patience, which gave me the strength to stay focused and complete the project.

This project has been a valuable learning experience, and I am truly thankful to everyone who made it possible.

**MD TARIQUE ANWER (322201297)**

**Lovely Professional University, Phagwara**

***Tarique***

# Contents

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Topic** | **Page No.** |
| 1 | Cover Page | 1 |
| 2 | Student Declaration | 2 |
| 3 | Acknowledgement | 3 |
| 4 | Contents | 4 |
| 5 | List of Tables/Figures | 5 |
| 6 | List of Abbreviations | 6 |
| 7 | Abstract | 7 |
| 9 | **Chapter 1 –** Introduction | 8 - 10 |
| 10 |  Overview of Training | 8 - 9 |
| 11 |  Overview of Project | 9 - 10 |
| 12 | **Chapter 2 –** Reason for Choosing this  Technology | 10 - 11 |
| 13 | **Chapter 3 –** Project: Movies Data Analysis | 11 - 43 |
| 14 |  Profile of the Project | 11 |
| 15 |  System Requirements and Specifications | 12 |
| 16 |  Technologies and Tools Used | 13 |
| 17 |  Design and Charts | 14 - 15 |
| 18 |  Project Source Code | 16 - 43 |
| 19 | **Chapter 4 –** Timeline of Training and Project  Work | 44 |
| 20 | **Chapter 5 –** Skills Gained from Training and Project | 45 |
| 20 | **Final Chapter –** Conclusion and Future  Outlook of Big Data, SQL Databases, and Data Analysis | 46 - 47 |
| 21 | Summary of Report | 48 |
| 22 | References | 49 |

**List of Tables and Figures**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Table Name** | **Page No.** |
| 3.1 | ER Diagram – Created using Creately | 14 |
| 3.2 | ER Diagram – Created using MySQL Workbench | 15 |

# List of Abbreviations

* **SQL –** Structured Query Language
* **DBMS –** Database Management System
* **RDBMS –** Relational Database Management System
* **CPU –** central processing unit
* **GB –** Gigabyte
* **RAM –** Random-access memory

# Abstract

In this report, I have shared a project where I have done data analysis of a movie's data set. This report also presents my learning and contributions during my summer training. This project, **“Real-Time Stock Analysis and Prediction Using Python and Yahoo Finance”,** aims to assist investors in making informed decisions by providing timely insights into stock market trends. In today's fast-paced financial environment, analyzing real-time stock data is crucial for individuals and organizations to stay ahead in investments. This project leverages Python for data extraction, processing, and visualization, using Yahoo Finance as the primary data source for live market data.

Our approach integrates data analysis and machine learning to predict short-term price movements and identify potential investment opportunities. Key tools and libraries used include pandas for data manipulation, matplotlib for visualization, and various predictive models for forecasting stock prices. By analyzing historical stock data and real-time trends, we generate insights on stock performance that can be used to predict future price movements.

Through this project, we demonstrate a practical application of machine learning in finance, highlighting the use of web scraping, data analysis, and predictive modeling to support investment decisions. This report details the methodologies, tools, and outcomes, providing a comprehensive view of how Python can be used to deliver robust stock analysis and predictive insights for real-time decision-making in the stock market.

# CHAPTER-1

# Introduction

The project, **“Real-Time Stock Analysis and Prediction Using Python and Yahoo Finance”,** is designed to provide investors with accurate, data-driven insights into stock market performance. This project retrieves both historical and real-time stock data from Yahoo Finance using web scraping techniques, with an alternative option to use the YFinance API if web scraping becomes unavailable.

By integrating machine learning techniques and leveraging key Python libraries—such as **Pandas** for efficient data manipulation, **NumPy** for complex numerical computations, and **Matplotlib** for data visualization—this project transforms raw market data into actionable insights. The primary objective is to predict future stock trends, empowering investors to make informed decisions about which stocks to buy. Through real-time analysis and forecasting, this project demonstrates the potential of Python in financial analytics, offering a robust tool for stock prediction and decision-making.

# Overview of the Project

**“The Real-Time Stock Analysis and Prediction Using Python and Yahoo Finance”** project is designed to empower investors with data-driven insights into stock performance, using a combination of real-time and historical data from Yahoo Finance. By extracting and analyzing this data, the project provides a basis for predicting stock trends and informing investment decisions.

**Data Retrieval and Processing**

Data is sourced from Yahoo Finance through web scraping techniques, or alternatively through the YFinance API if web scraping becomes inaccessible. This dual approach ensures a reliable and continuous flow of market data for analysis. Key libraries such as Pandas and NumPy play a central role in structuring, cleaning, and manipulating the data, transforming raw inputs into meaningful insights.

**Analytical Approach and Machine Learning**

Using Python’s extensive data science ecosystem, the project explores historical stock prices to identify patterns and predict future stock performance. Various machine learning models are applied to this dataset, aiming to predict short-term stock price movements based on historical trends and market behavior. This predictive modeling enables a comparative analysis of stocks, identifying those with promising trends for potential investments.

**Visualization and User Insights**

Data visualization is achieved through the Matplotlib library, providing graphical insights that help investors quickly understand trends and patterns. By combining real-time data analysis with predictive analytics, the project delivers a user-friendly platform where users can visualize potential stock movements and make informed investment decisions.

**Goal and Significance**

The primary objective of this project is to enable informed, data-driven decision-making for investors looking to maximize returns. By providing an accessible, Python-based solution for stock analysis and prediction, the project serves as a practical tool for anyone interested in leveraging real-time financial data to anticipate market movements. This project not only demonstrates the application of Python in financial analytics but also showcases the role of machine learning in shaping modern investment strategies.

# CHAPTER-2

# System Requirement Specification

In this chapter, the requirements for the **Real-Time Stock Analysis and Prediction Using Python and Yahoo Finance** project are analyzed in terms of system, software, and data requirements. A thorough understanding of these requirements ensures that the project is developed within the scope and performs as expected, providing accurate predictions and insights for investors.

The system requirements for the project focus on ensuring that the hardware and software infrastructure is capable of handling real-time stock data retrieval, data processing, and machine learning model training.

**3.1 Hardware Requirements**

The hardware requirements are moderate, as the project involves data analysis, web scraping, and machine learning model prediction, which are computationally intensive but not resource-heavy for small to medium-scale data.

* **Processor:** A minimum of Intel Core i3 or equivalent (Quad-core preferred for better performance during data processing and model training).
* **RAM:** 8 GB or higher to handle data manipulation and machine learning tasks.
* **Storage:** At least 50 GB of free disk space for storing datasets, CSV files, and model output.
* **Internet Connection:** A stable connection for web scraping or API requests to Yahoo Finance.

**3.2 Software Requirements**

The software environment is crucial for running the project smoothly, especially for Python-based development. Several software tools and platforms will be required for development, testing, and deployment.

* **Operating System:** Windows 10 or higher, macOS, or any Linux-based OS.
* **IDE (Integrated Development Environment):** Recommended IDEs include Visual Studio Code, Google Colab, or Jupyter Notebook for interactive coding and analysis.
* **Version Control System:** Git for managing project code versions, collaborating with teams, and maintaining code history.
* **Internet Browser:** Google Chrome, Firefox, Safari, Edge, Opera or any other latest version of modern internet browsers.
* **Python:** Version 3.11 as it is compatible with the necessary libraries and frameworks.
* **Web Framework:** Flask for backend development.
* **Python Libraries:**
  + **Pandas:** For data manipulation and analysis.
  + **NumPy:** For numerical computations.
  + **Plotly:** For data visualization using graph
  + **Matplotlib:** For data visualization.
  + **YFinance:** For accessing Yahoo Finance data through API calls.
  + **Scikit-learn:** For implementing machine learning algorithms.
  + **Beautiful Soup:** For web scraping (if needed).
  + **Flask:** For web application and API’s
* **JavaScript Libraries and Frameworks:** 
  + **Jquery 3.6:**
  + **Bootstrap 5**: For UI
  + **jQuery DataTable:** For datatable
  + **Moment.js:** For Date and time formatting
* **Microsoft Excel:** Microsoft Excel for reading csv (comma separated value) files.

**3.3 Data Requirements**

Data plays a central role in this project, and it is vital that the data is reliable, accurate, and up-to-date. The following data requirements have been identified for effective stock prediction:

**Stock Data:** The project will rely on historical and real-time stock data from Yahoo Finance, including stock prices, market trends, trading volume, and other relevant metrics.

**Data Storage:** CSV files will be used to store historical stock data retrieved from Yahoo Finance. These files will include columns such as stock symbols, date, opening price, closing price, volume, and other relevant indicators.

**Data Accuracy and Completeness:** The stock data needs to be accurate and free from missing values to ensure the quality of analysis. Incomplete or erroneous data will affect prediction accuracy.

**Real-Time Data:** The project will need to pull real-time stock data for ongoing analysis. This will be achieved through web scraping or the YFinance API, which requires a continuous internet connection for fetching the latest market data

# CHAPTER-3

# Technology Stack

In developing the **“Real-Time Stock Analysis and Prediction Using Python and Yahoo Finance”** project, a variety of technologies were carefully selected to ensure robust data handling, efficient processing, and effective visualization. Each chosen technology contributes uniquely to the project's requirements, facilitating a streamlined approach to analyzing and predicting stock trends.

**2.1 Python**

Python was chosen as the primary programming language due to its extensive ecosystem of libraries and tools designed for data science, machine learning, and financial analysis. Python's syntax is simple and readable, making it highly accessible and suitable for rapid development. Additionally, Python offers a range of libraries tailored for data manipulation, visualization, and machine learning, making it an ideal choice for building an end-to-end stock prediction model.

**2.2 Pandas**

The Pandas library is pivotal for data manipulation in this project. It provides robust data structures, like Data Frames, which allow for easy manipulation, cleaning, and transformation of large datasets. Stock data analysis often requires processing extensive historical datasets, and Pandas simplifies the process of importing, organizing, and preparing this data for analysis. Its integration with other Python libraries makes it highly versatile for building comprehensive data workflows.

**2.3 NumPy**

NumPy is essential for performing numerical computations efficiently. Since stock prediction models rely on complex mathematical operations and calculations, NumPy’s fast array operations and mathematical functions allow for efficient handling of large datasets. This library complements Pandas, enabling streamlined computations necessary for data preparation and modeling in real-time and historical data analysis.

**2.4 Matplotlib**

Matplotlib was selected for its powerful data visualization capabilities. Visualizing stock trends is critical for understanding and presenting market insights. Matplotlib enables the creation of charts and graphs that help illustrate stock performance, historical trends, and predictions. This visual clarity enhances the project by providing users with an intuitive way to comprehend data patterns, assisting investors in making informed decisions.

**2.5 Web Scraping and YFinance API**

Yahoo Finance is a reliable source of both real-time and historical stock data, making it a valuable resource for this project. Web scraping is used to retrieve data directly from Yahoo Finance, providing flexibility in obtaining various types of financial information. However, to ensure data access continuity, the YFinance API serves as a backup in cases where web scraping may be limited or unavailable. This dual approach ensures consistent data flow, allowing the project to function effectively without disruptions.

**2.6 CSV (Comma-Separated Values)**

The CSV (Comma-Separated Values) format is commonly used for storing and exchanging data. In this project, CSV files are used to store historical stock data retrieved from Yahoo Finance. The CSV format provides a simple and widely-supported method of saving data, making it easy to import, export, and work with data in Python using Pandas. By leveraging CSV files, the project ensures that data can be easily loaded, saved, and shared across different components and systems, maintaining flexibility in data processing.

**2.7 Machine Learning Algorithms**

Integrating machine learning algorithms enables this project to move beyond mere data analysis and into prediction. Machine learning algorithms can analyze patterns within historical stock data and generate forecasts on future stock prices. By using Python’s machine learning ecosystem, which includes libraries like scikit-learn, the project can implement models that provide actionable insights for investors. This predictive aspect is the cornerstone of the project, offering users a data-driven basis for their investment decisions.

**2.8 Plotly**

Plotly was selected for its advanced data visualization capabilities that extend beyond the basic plotting functionalities offered by libraries such as Matplotlib. Plotly’s interactive charts provide users with a more engaging experience, allowing them to hover over data points for detailed information and manipulate the visualization for deeper insights. In the context of stock analysis and prediction, Plotly’s interactivity enhances the user’s ability to explore and interpret data trends, aiding in a comprehensive understanding of market patterns. This interactive approach makes it easier for users to make well-informed decisions by observing trends and predictions in a dynamic format.

**2.9 Flask**

Flask was chosen as the web framework for this project due to its lightweight and flexible nature. Flask's simplicity allows for rapid development and integration with various Python libraries, making it ideal for building web-based applications that require custom functionality. For this project, Flask acts as the backbone of the web interface, facilitating seamless communication between the frontend and backend, and ensuring a smooth user experience. Flask’s modular structure supports scalability, which is advantageous for projects that may require expansion in the future. Additionally, its integration with Jinja2 templating engine aids in rendering dynamic HTML content, enabling interactive and responsive web pages.

**2.10 HTML**

HTML (Hypertext Markup Language) was chosen as the structural foundation for the web application. It is essential for defining the content and layout of the web pages, ensuring that information is organized in a meaningful way. HTML provides the basic elements that structure the application, allowing for the seamless integration of data visualization tools, interactive charts, and user input forms. Its compatibility with various web technologies makes it a versatile choice for building the framework of the project’s user interface.

**2.11 CSS**

CSS (Cascading Style Sheets) was used to style the web pages and enhance the visual appeal of the application. With CSS, the project achieves a modern and user-friendly design that improves the overall user experience. CSS allows for customization of elements such as fonts, colors, and layout, ensuring that the interface is both attractive and responsive. By leveraging CSS, the project maintains consistency in design across different devices, enhancing accessibility and usability.

**2.12 JavaScript**

JavaScript was employed to add interactivity and dynamic behavior to the web application. This scripting language enables real-time updates, client-side data validation, and interactive features that improve user engagement. JavaScript’s ability to manipulate the Document Object Model (DOM) allows the project to respond to user actions effectively, creating a more seamless and responsive user experience. Integrating JavaScript ensures that users can interact with the stock analysis and prediction platform in real-time, enhancing the practicality and functionality of the application.

**2.13 jQuery 3.6**

A lightweight, fast, and feature-rich JavaScript library used for simplifying HTML document traversal, event handling, animation, and AJAX interactions. Version 3.6 ensures compatibility and modern web practices.

**2.14 Bootstrap 5**

A popular CSS framework that provides pre-designed components and responsive design utilities to create modern, mobile-first, and aesthetically pleasing web applications without extensive custom CSS.

**2.15 jQuery DataTable**

A plugin for jQuery that adds advanced interaction controls to HTML tables, including search, sorting, pagination, and data export features, enhancing the usability of data displays in web applications.

**2.15 Moment.js**

A JavaScript library designed for parsing, validating, manipulating, and formatting dates and times. It's particularly useful for handling complex date operations in client-side scripting.

# CHAPTER-4

# System Design

The system design phase is critical to ensuring that the “**Real-Time Stock Analysis and Prediction Using Python and Yahoo Finance”** project operates efficiently and meets its objectives. This chapter outlines the architecture, components, and workflow of the system, detailing how each element interacts to create a cohesive stock analysis and prediction tool.

**4.1 Architecture Overview**

The system architecture is designed to handle data retrieval, processing, analysis, and prediction efficiently. The project follows a modular structure, where each module has a distinct function that contributes to the overall workflow. The architecture can be broken down into the following main components:

* **Data Collection Module:** Responsible for retrieving real-time and historical stock data from Yahoo Finance using web scraping and the YFinance API.
* **Data Preprocessing Module:** Cleans and formats the retrieved data for analysis.
* **Analysis and Prediction Module:** Uses machine learning models to analyze historical data and predict future stock trends.
* **Visualization Module:** Presents data and prediction results using charts and graphs created with Matplotlib.
* **User Interface Module:** Provides a user-friendly interface for users to input stock symbols and view analysis and predictions.

**4.2 Data Flow Diagram (DFD)**

The Data Flow Diagram illustrates the flow of information through the system, providing a clear overview of how data is processed from input to output.

**Level 0:**

* **User Input:** Users provide the stock symbol they are interested in analyzing and a start and end date to analyze from historical data.
* **System Output:** The system displays analysis and prediction results.

**Level 1:**

* **Data Input:** Stock symbol and date range are input by the user.
* **Data Collection Module:** Retrieves data from Yahoo Finance via web scraping or the YFinance API.
* **Data Preprocessing Module:** Formats and cleans the data for analysis.
* **Analysis and Prediction Module:** Applies machine learning models to generate predictions.
* **Visualization Module:** Creates visual representations of stock trends and predictions.
* **User Output:** Final analysis and prediction results are displayed.

**4.3 System Components**

Each system component plays a specific role in the workflow. The following sections describe these components in detail:

**4.3.1 Data Collection Module**

This module is designed to interact with Yahoo Finance to fetch both historical and real-time stock data. The data is retrieved using:

* **Web Scraping:** For more flexible data extraction when needed.
* **YFinance API:** Provides a straightforward method for accessing structured stock data.

**4.3.2 Data Preprocessing Module**

The preprocessing module ensures the data is cleaned and ready for analysis. This involves:

* Removing null or missing values.
* Formatting data into a structured CSV or DataFrame.
* Calculating additional features if necessary (e.g., moving averages, volatility).

**4.3.3 Analysis and Prediction Module**

This module is the core of the system where machine learning algorithms are applied. The steps include:

* **Feature Selection:** Choosing relevant features from the dataset to improve prediction accuracy.
* **Model Training:** Using algorithms such as linear regression or decision trees to train the model.
* **Prediction:** Running the trained model on the prepared dataset to forecast future stock prices.

**4.3.4 Visualization Module**

The visualization module employs Matplotlib to create graphs that help in understanding stock trends and predictions. Common visual outputs include:

* Time series plots for stock prices.
* Prediction overlay charts showing actual vs. predicted data.

**4.3.5 User Interface Module**

A simple user interface allows users to input stock symbols and view the results. This module may include:

* **Web-based Interface:** Built using frameworks like Flask.
* **Command-line Interface (CLI):** For developers and advanced users who prefer using the terminal.

**4.4 User Interface Design**

A user-friendly interface is essential for an effective project. The design prioritizes simplicity and clarity, ensuring that even non-technical users can navigate and use the tool efficiently. The main features of the interface include:

**Search Box:** For users to enter the stock symbol and select the date range.

**Search Button:** To initiate data retrieval and prediction.

**Overview Section:** To view stock current closing price, change and change percentage and prediction recommendation.

**Graph Display Tab:** To present the analysis and prediction results visually.

**Historical Data Tab:** To present the analysis and prediction results visually.

**Finance News Tab:** To present the analysis and prediction results visually.

**4.5 Workflow Diagram**

A workflow diagram shows the step-by-step process that data goes through in the system:

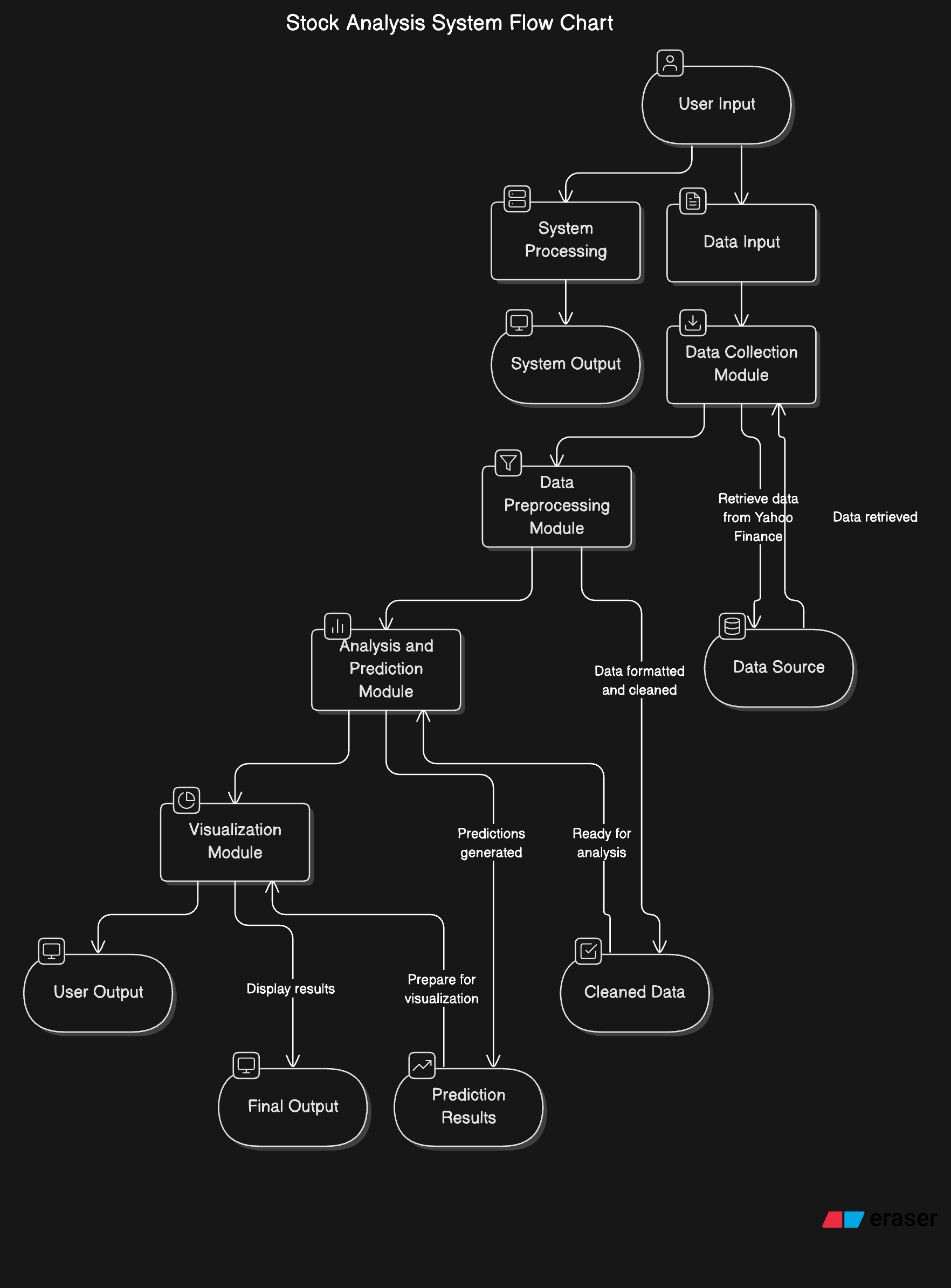
* **User Input:** Stock symbol and date range.
* **Data Collection:** Data is retrieved from Yahoo Finance.
* **Data Preprocessing:** Data is cleaned and formatted.
* **Analysis and Prediction:** Machine learning models process the data.
* **Visualization:** Results are plotted.
* **User Output:** Predictions are displayed to the user.

**4.6 System Design Considerations**

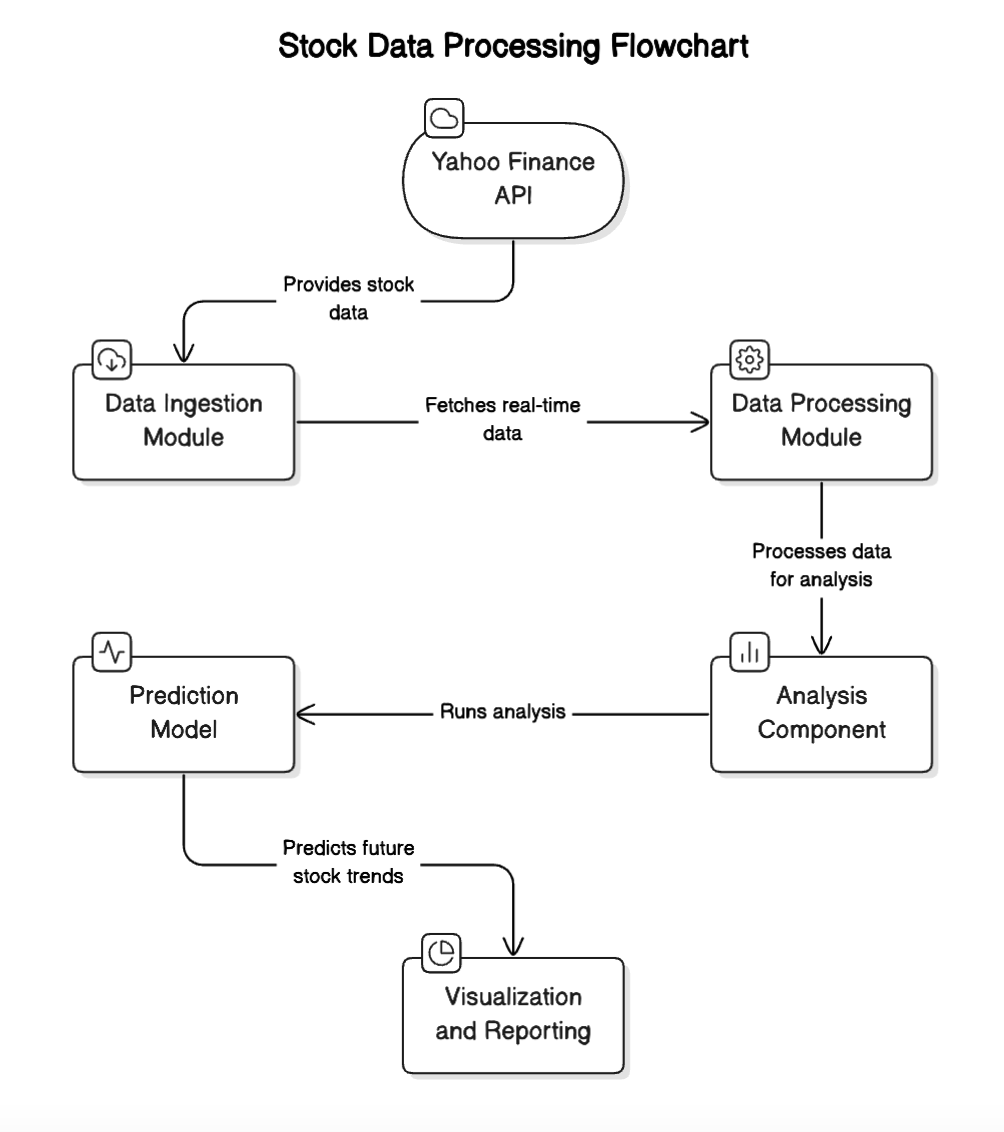
* **Modularity:** Each component is independent, making the system easy to modify and update.
* **Scalability:** The design supports scaling up to include more data sources or more advanced machine learning models.
* **Error Handling:** The system includes mechanisms to handle errors in data retrieval or inconsistencies.

# Design and Charts

**Figure 3.1 - Stock Analysis System Flowchart**



**Figure 3.2 – Stock Data Processing Flowchart**



# Project Source Code



**CREATE** DATABASE movies;

**USE** movies;

**CREATE TABLE** films (

id INTEGER,

title VARCHAR(50),

release\_year INTEGER,

country VARCHAR(30),

duration INTEGER,

language VARCHAR(30),

certification VARCHAR(25),

gross BIGINT,

budget BIGINT,

**CONSTRAINT** films\_pk **PRIMARY KEY**(id)

);

**CREATE TABLE** people ( id

name birthdate deathdate

);

INTEGER **PRIMARY KEY**, VARCHAR(30),

DATE, DATE

**CREATE TABLE** reviews ( id

film\_id num\_user num\_critic imdb\_score num\_votes facebook\_likes

);

INTEGER INTEGER, INTEGER, INTEGER, REAL, INTEGER, INTEGER

**PRIMARY KEY**,

**CREATE TABLE** roles (

id film\_id person\_id role

);

INTEGER **PRIMARY KEY**, INTEGER,

INTEGER, VARCHAR(30)



**CREATE TABLE** roles (

id film\_id person\_id role

);

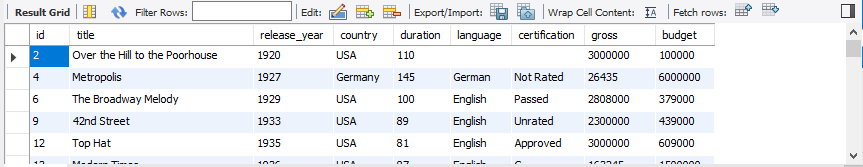
INTEGER **PRIMARY KEY**, INTEGER,

INTEGER, VARCHAR(30)



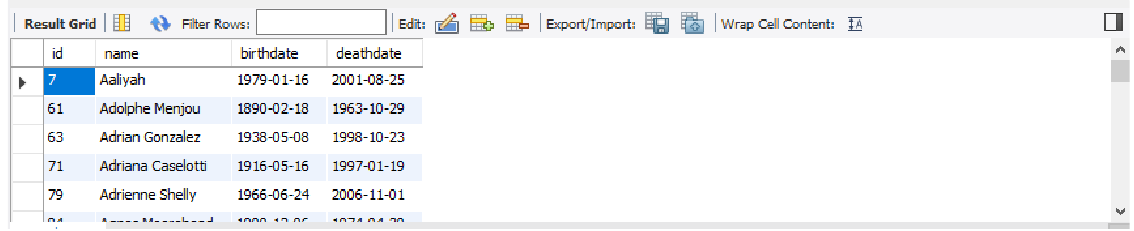
-- Show All tables

**SHOW TABLES**;



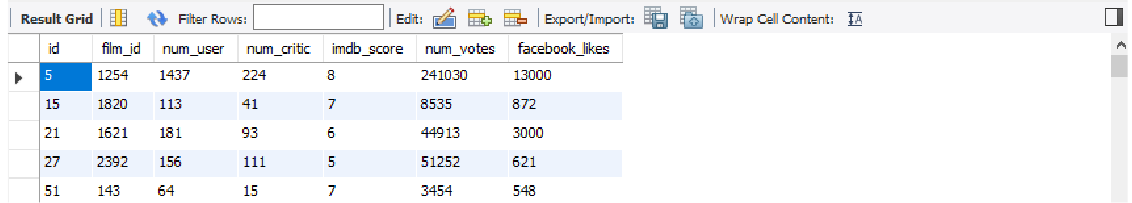
-- Show content of films table

**SELECT** \* **FROM** films;



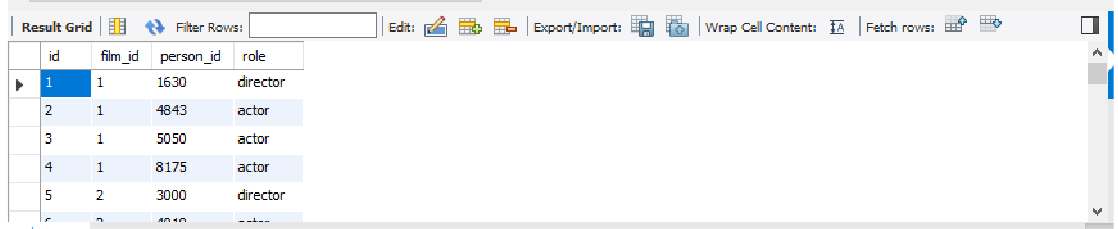
-- Show content of people table

**SELECT** \* **FROM** people;



-- Show content of reviews table

**SELECT** \* **FROM** reviews;

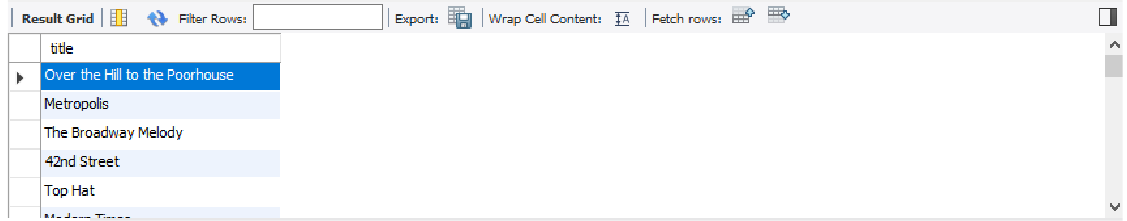


-- Show content of roles table

**SELECT** \* **FROM** roles;



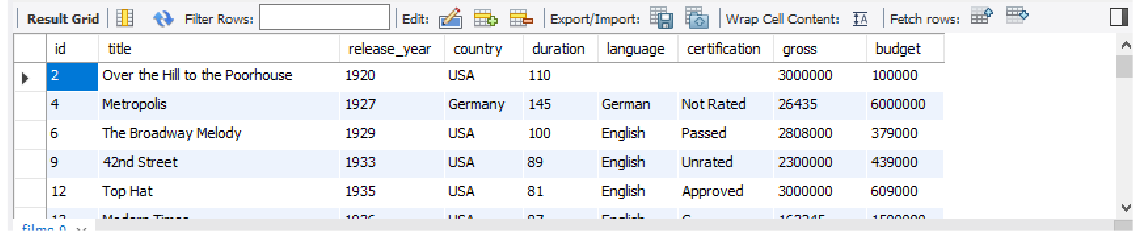
**-- Selecting Columns: SELECT, SELECT DISTINCT**



-- Get the title of every film

**SELECT** title

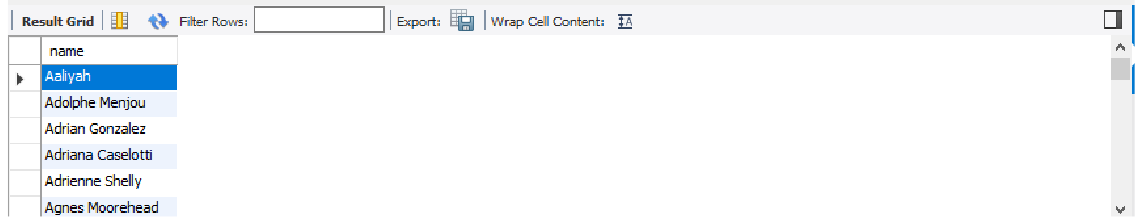
**FROM** films;



-- Get all details for every film

**SELECT** \*

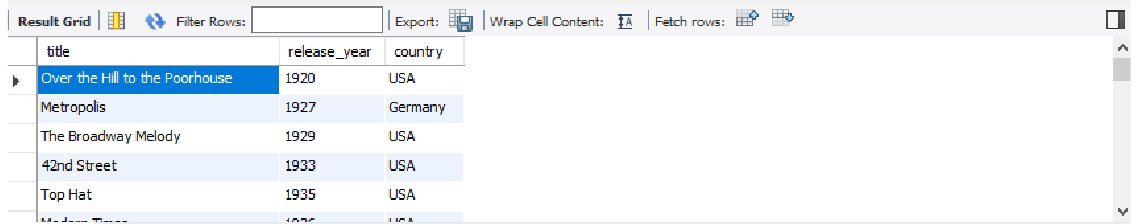
**FROM** films;



-- Get the names of everyone involved in working on the films

**SELECT** name

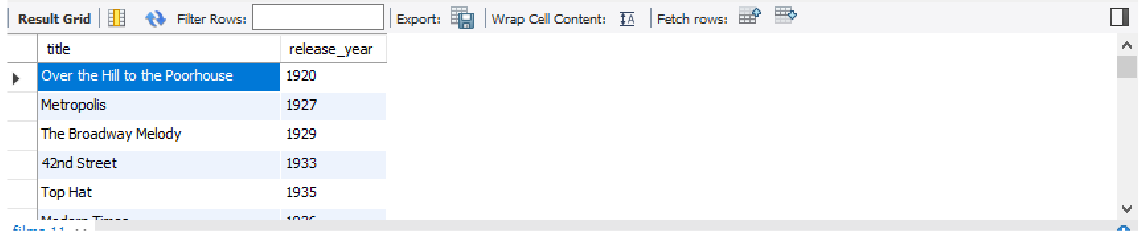
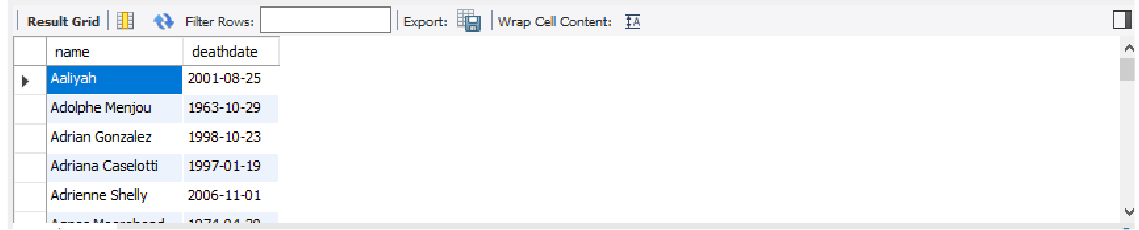
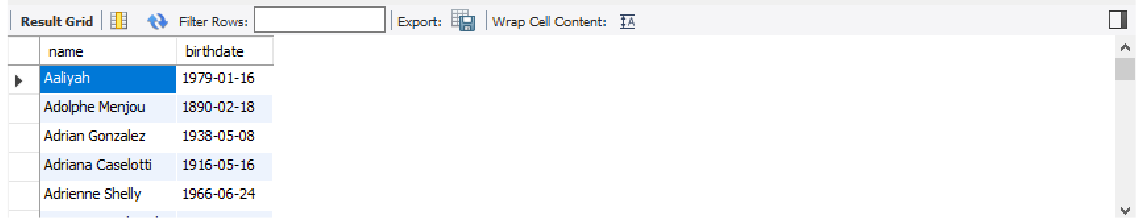
**FROM** people;



-- Get the title, release year and country for every film

**SELECT** title, release\_year, country

**FROM** films;



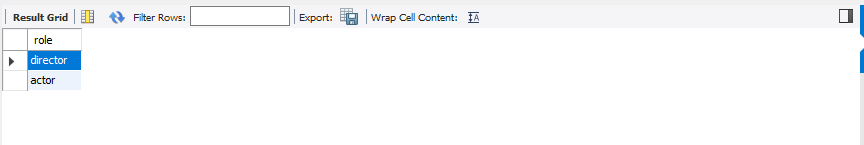
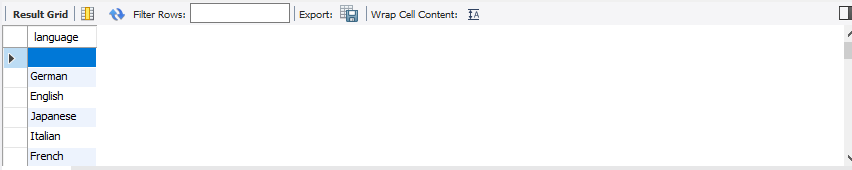
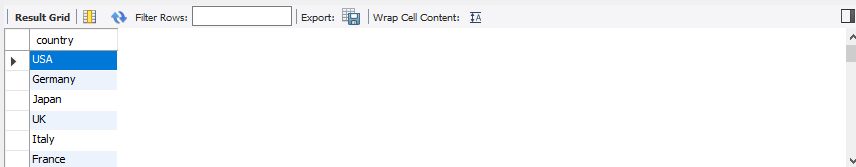
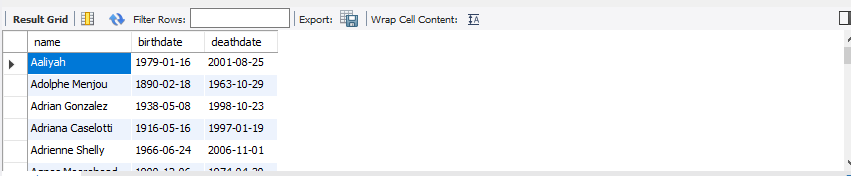
-- Get the title and release year of every film

**SELECT** title, release\_year

**FROM** films;

|  |
| --- |
| -- Get every person's name and their date of birth where possible  **SELECT** name, birthdate  **FROM** people; |
|  |
| -- Get every person name and their date of death where possible  **SELECT** name, deathdate  **FROM** people; |

-- Get everyone's name, date of birth, and date of death (where possible)



**SELECT** name, birthdate, deathdate

**FROM** people;

- Get all the different countries

**SELECT DISTINCT** country

**FROM** films;

-- Get all the different film languages

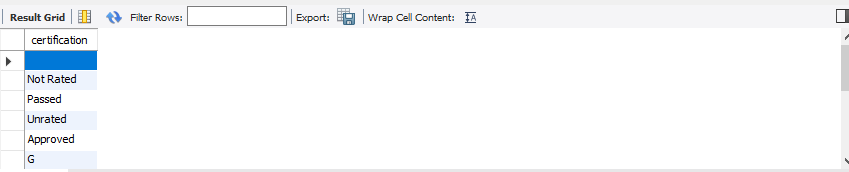
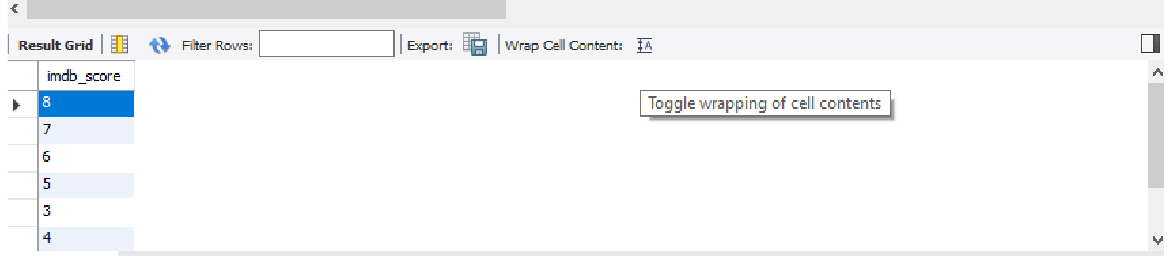
**SELECT DISTINCT** language

**FROM** films;

-- Get the different types of film roles

**SELECT DISTINCT** role

**FROM** roles;



-- Get all the different certification categories

**SELECT DISTINCT** certification

**FROM** films;

-- Get all the different IMDB scores

**SELECT DISTINCT** imdb\_score

**FROM** reviews;



**-- Aggregate Functions: COUNT, SUM, AVG, MIN, MAX**



-- Count the number of rows in the people table

**SELECT** COUNT(\*)

**FROM** people;

-- Result 781

-- Count the number of birthdate entries in the people table

**SELECT** COUNT(birthdate)

**FROM** people;

-- Result 781

-- Count the number of unique birthdate entries in the people table

**SELECT** COUNT(**DISTINCT** birthdate)

**FROM** people;

-- Result 757

-- Count the number of unique languages



**SELECT** COUNT(**DISTINCT** language)

**FROM** films;

-- Result 39

-- Count the number of unique countries

**SELECT** COUNT(**DISTINCT** country)

**FROM** films;

-- Result 46

-- Count the number of people who have died

**SELECT** COUNT(deathdate)

**FROM** people;

-- Result 781

-- Count the number of years the dataset covers

**SELECT** COUNT(**DISTINCT** release\_year)

**FROM** films;

-- Result 75

-- Get the total duration of all films

**SELECT** SUM(duration)

**FROM** films;

-- Result 426426

-- Get the average duration of all films

**SELECT** AVG(duration)

**FROM** films;

-- Result 109.9319

-- Get the duration of the shortest film

**SELECT** MIN(duration)

**FROM** films;

-- Result 37

-- Get the amount made by the highest grossing film

**SELECT** MAX(gross)

**FROM** films;

-- Result 936627416

-- Get the amount made by the lowest grossing film

**SELECT** MIN(gross)

**From** films;

-- Result 162



-- Get both the lowest and highest grossing films, for comparision

**SELECT** MIN(gross), MAX(gross)

**FROM** films;

-- Get the highest number of Facebook likes for any film

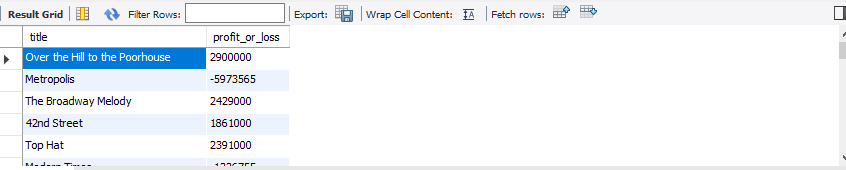
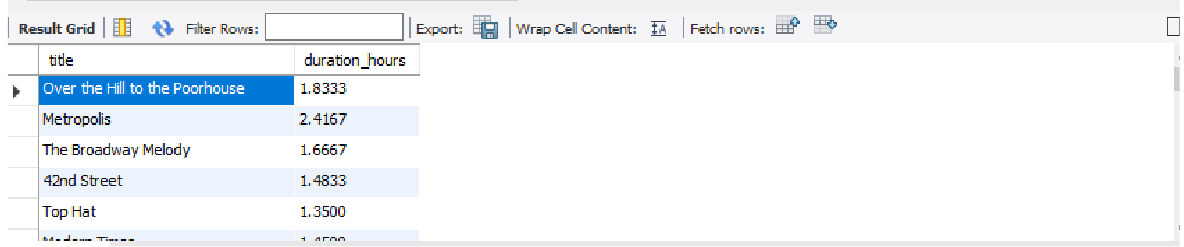
**SELECT** MAX(facebook\_likes)

**FROM** reviews;

-- Result 150000



**-- Aliasing and Basic Arithmetic**



-- Get the profit (or loss) for each movie, where possible

**SELECT** title, gross - budget

**AS** profit\_or\_loss

**FROM** films;

-- Get the duration in hours for each film **SELECT** title, duration / 60.0 **AS** duration\_hours **FROM** films;

-- Get the average film duration in hours



**SELECT** AVG(duration) / 60.0

**AS** duration\_hours

**FROM** films;

-- Result 1.83219902

-- Get the percentage of people who have died

**SELECT** COUNT(deathdate) \* 100 / COUNT(\*)

**AS** percentage\_dead

**FROM** people;

-- Result 100.0000

-- Check if there's an even number of unique languages

**SELECT** COUNT(**DISTINCT** language) % 2

**AS** result

**FROM** films;

-- Result 1 (0 = yes, 1 = no)

-- Get the of years between the oldest film and newest film

**SELECT** MAX(release\_year) - MIN(release\_year)

**AS** difference

**FROM** films;

-- Result 96

-- Get the number of decades this dataset covers **SELECT** (MAX(release\_year) - MIN(release\_year)) / 10 **AS** number\_of\_decades

**FROM** films;

-- Result 9.6000



**-- Rounding Functions: ROUND, FLOOR, CEILING**



-- Get the average duration of all films, rounded to the nearest minute

**SELECT** ROUND(AVG(duration)) **AS** rounded\_avg\_run\_time **FROM** films;

-- Result 110



-- Get the average duration of all films, rounded down to nearest minute

**SELECT** FLOOR(AVG(duration)) **AS** floored\_avg\_run\_time **FROM** films;

-- Result 109

-- Get the average duration of all films, rounded up to the nearest minute

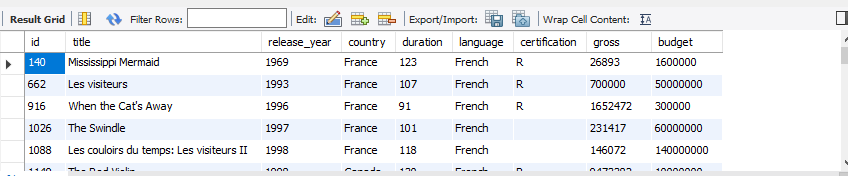
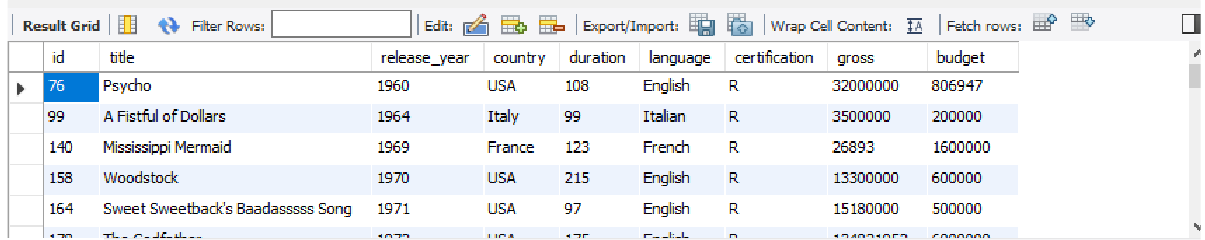
**SELECT** CEILING(AVG(duration))

**FROM** films;

-- Result 110



**-- Filtering: WHERE, =, <>, <, <=, >, >=, AND, OR**



-- Get all French language films

**SELECT** \*

**FROM** films

**WHERE** language = 'French';

**SELECT** COUNT(\*)

**FROM** films

**WHERE** language = 'Hindi';

-- Result 10

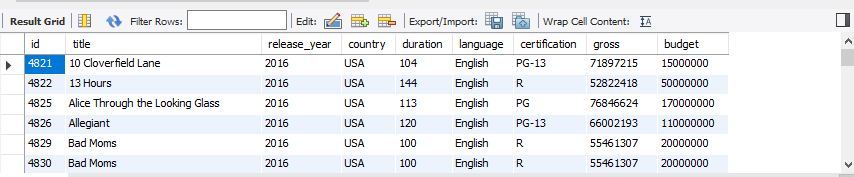
-- Get all movies with an R certification

**SELECT** \*

**FROM** films

**WHERE** certification = 'R';

-- Get all films released in 2016



**SELECT** \*

**FROM** films

**WHERE** release\_year = 2016;

-- Count of actors

**SELECT** COUNT(\*)

**FROM** roles

**WHERE** role = 'actor';

-- Result 14862

-- Count of directors

**SELECT** COUNT(\*)

**FROM** roles

**WHERE** role = 'director';

-- Result 4929

-- Count of movies not rated

**SELECT** COUNT(\*)

**FROM** films

**WHERE** certification = 'Not Rated' OR certification

**IS** NULL;

-- Result 42

-- Count of movies not in English

**SELECT** COUNT(\*)

**FROM** films

**WHERE** language <> 'English';

-- Result 184

-- Get the number of films released before 2000

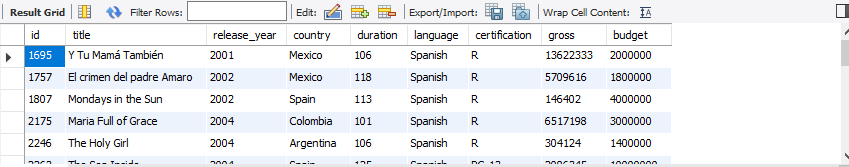
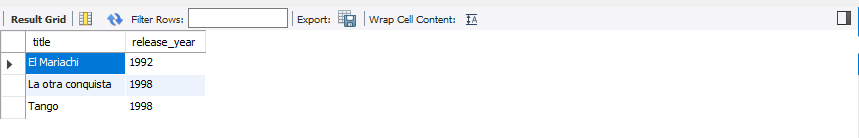
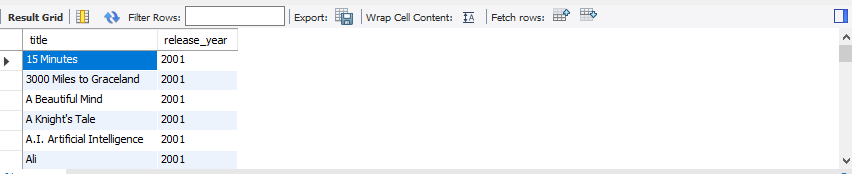
**SELECT** COUNT(\*)

**FROM** films

**WHERE** release\_year < 2000;

-- Result 1050

-- Get the title and release year of films released since 2000



**SELECT** title, release\_year

**FROM** films

**WHERE** release\_year > 2000;

-- Get all Spanish films released before 2000

**SELECT** title, release\_year

**FROM** films

**WHERE** release\_year < 2000 AND language = 'Spanish';

-- Get the all Spanish films released since 2000

**SELECT** \*

**FROM** films

**WHERE** release\_year > 2000 AND language = 'Spanish';

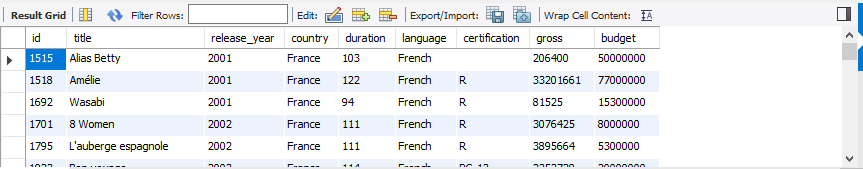
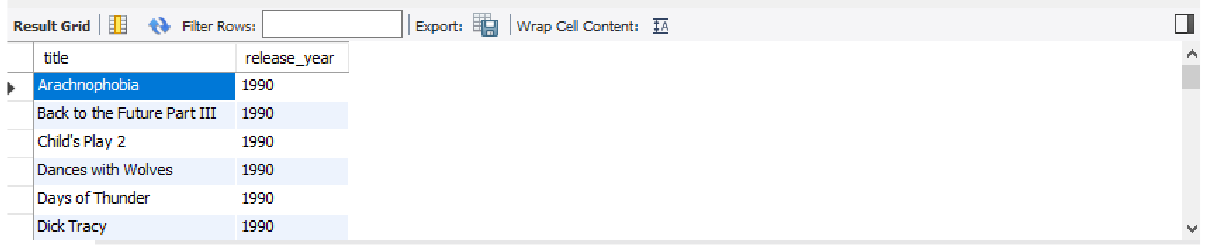
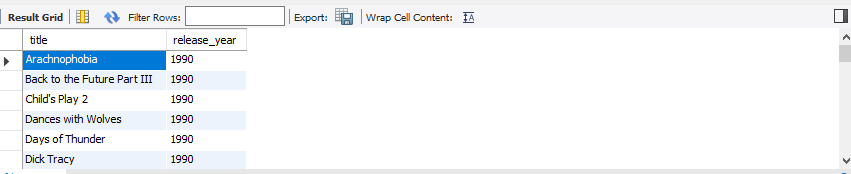
-- Get average duration for films released in France in 1993

**SELECT** AVG(duration)

**FROM** films

**WHERE** release\_year = 1993 AND country = 'France';

-- Result 103.0000



-- Get films released in 1990 or released in 2000 in French or Spanish

**SELECT** title, release\_year

**FROM** films

**WHERE** release\_year = 1990 OR release\_year = 2000 AND language = 'French' OR language = 'Spanish';

-- Get films released since 2000 that are in French or Spanish, and made more than $20m

**SELECT** \*

**FROM** films

**WHERE** release\_year > 2000

AND language = 'French' OR language = 'Spanish' AND gross > 20000000;

-- Get films released in the 90s

**SELECT** title, release\_year

**FROM** films

**WHERE** release\_year >= 1990 AND release\_year <= 2000;



-- Get average duration for films released in the UK or which were released in 2012

**SELECT** AVG(duration)

**FROM** films

**AS** average\_duration

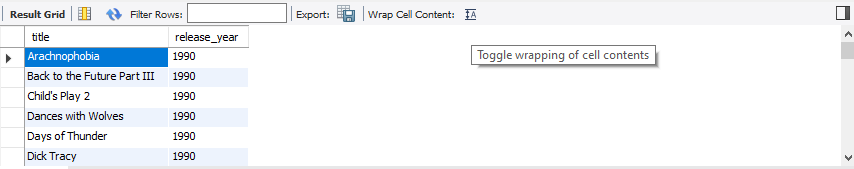
**WHERE** release\_year = 2012 OR COUNTRY = 'UK';

-- Result 111.3194



**-- Advanced Filtering: BETWEEN, IN, IS NULL, IS NOT**

**NULL, LIKE, NOT LIKE**



-- Get films released in the 90s

**SELECT** title, release\_year

**FROM** films

**WHERE** release\_year **BETWEEN** 1990 AND 2000;

-- Get the number of films released in the 90s

**SELECT** COUNT(\*)

**FROM** films

**WHERE** release\_year **BETWEEN** 1990 AND 2000;

-- Result 111.3194

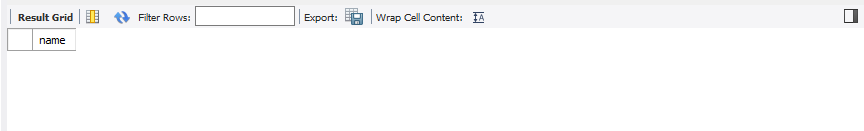
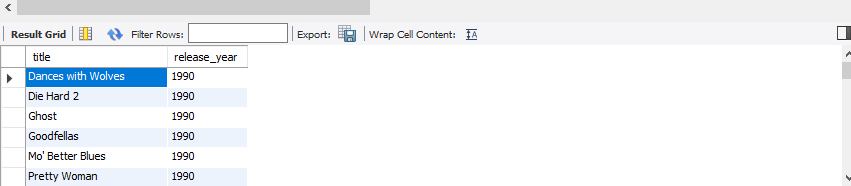
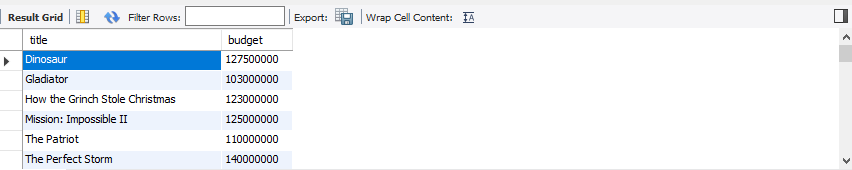
-- Get the number of films released between 2000 and 2015 that were longer than two hours

**SELECT** COUNT(\*)

**FROM** films

**WHERE** release\_year **BETWEEN** 2000 AND 2015 AND duration > 120;

-- Result 587



-- Get the number of films made between 2000 and 2015 with budgets over $100 million

**SELECT** title, budget

**FROM** films

**WHERE** release\_year **BETWEEN** 2000 AND 2015 AND budget > 100000000;

-- Get films released in in 1990 or released in 2000 that were longer than two hours

**SELECT** title, release\_year

**FROM** films

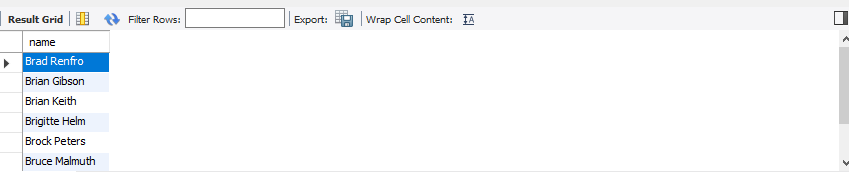
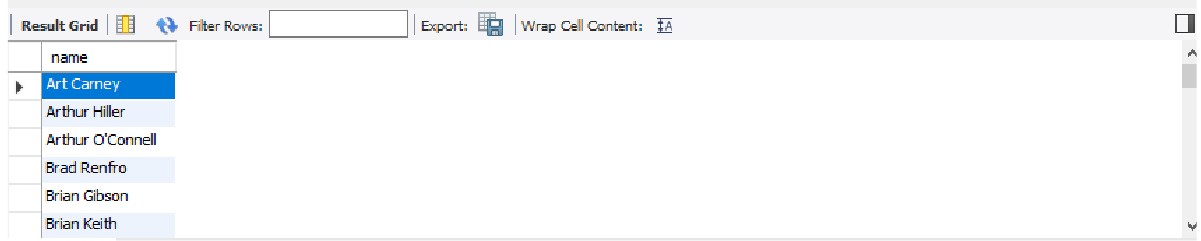
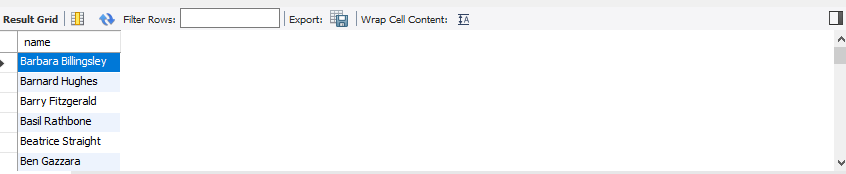
**WHERE** release\_year **IN** (1990, 2000) AND duration > 120;

-- Get the names of people who are still alive

**SELECT** name

**FROM** people

**WHERE** deathdate **IS** NULL OR deathdate = '';



-- Get people whose names begin with 'B'

**SELECT** name

**FROM** people

**WHERE** name LIKE 'B%';

-- Get people whose names begin with 'Br'

**SELECT** name

**FROM** people

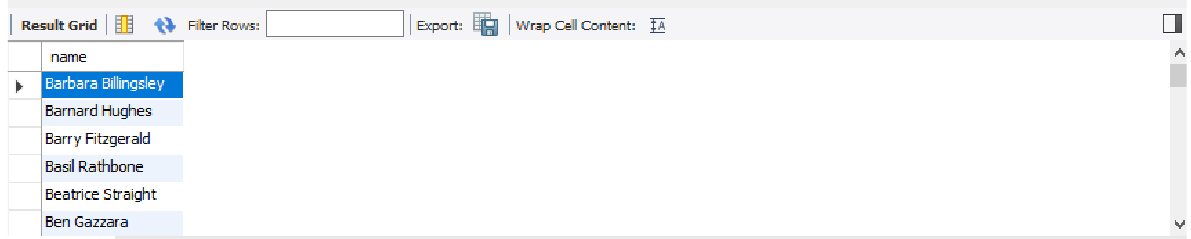
**WHERE** name LIKE 'Br%';

-- Get people whose names have 'r' as the second letter

**SELECT** name

**FROM** people

**WHERE** name LIKE '\_r%';



-- Get people whose names don't start with A

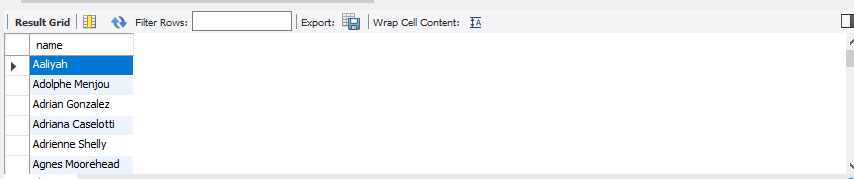
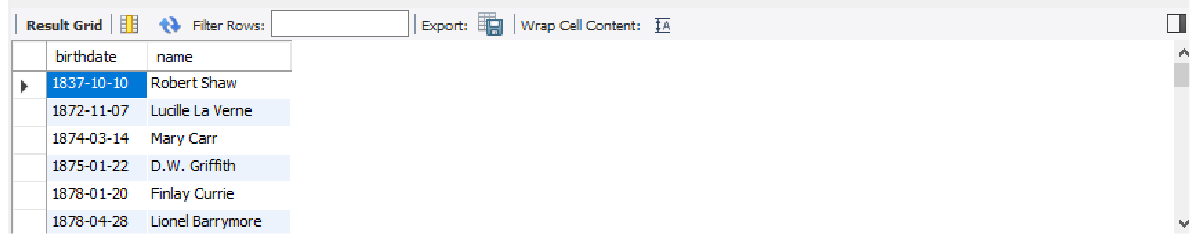
**SELECT** name

**FROM** people

**WHERE** name NOT LIKE 'A%';



**-- Sorting and Grouping**



-- Get people, sort by name

**SELECT** name

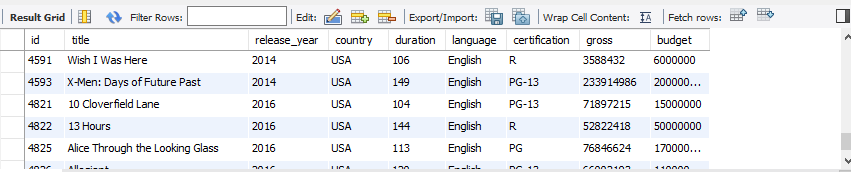
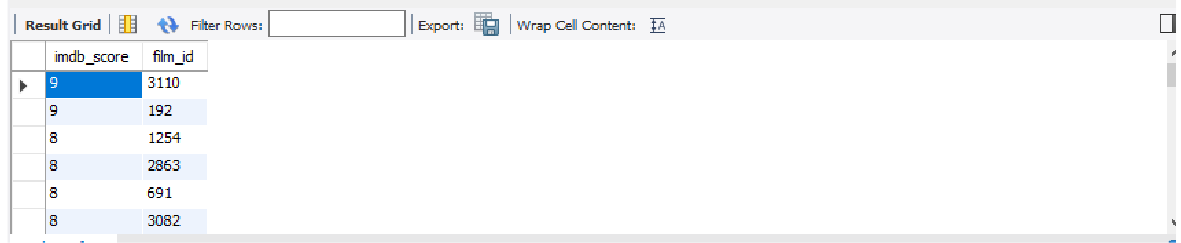
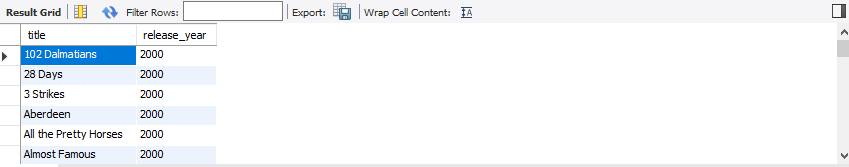
**FROM** people **ORDER BY** name;

-- Get people, in order of when they were born

**SELECT** birthdate, name

**FROM** people

**ORDER BY** birthdate;



-- Get films released in 2000 or 2015, in the order they were released

**SELECT** title, release\_year

**FROM** films

**WHERE** release\_year **in** (2000, 2015)

**ORDER BY** release\_year;

-- Get all films except those released in 2015, and order them

**SELECT** \*

**FROM** films

**WHERE** release\_year <> 2015

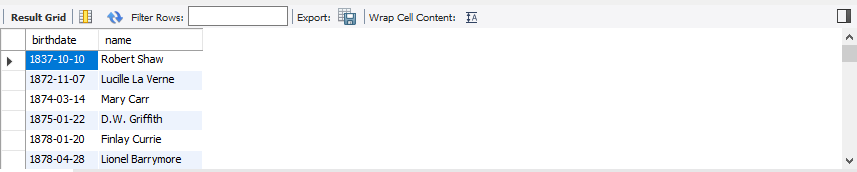
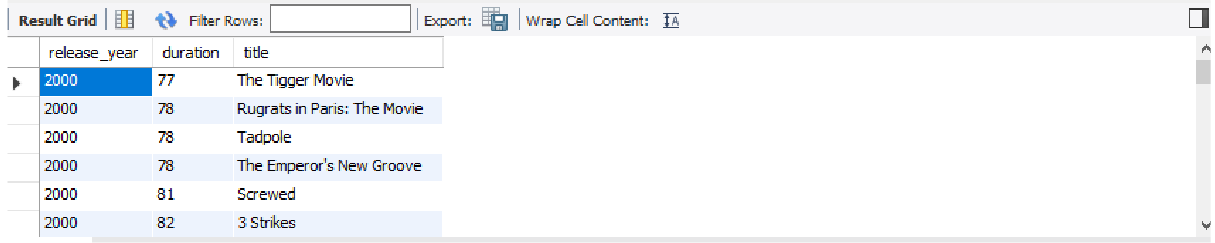
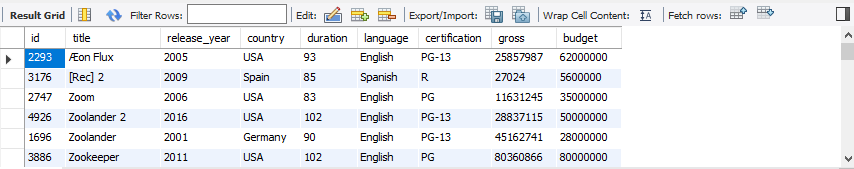
**ORDER BY** release\_year;

-- Get the score and film id for every film, from highest to lowest

**SELECT** imdb\_score, film\_id

**FROM** reviews

**ORDER BY** imdb\_score **DESC**;



-- Get the titles of films in reverse order

**SELECT** \*

**FROM** films

**ORDER BY** title **DESC**;

-- Get people, in order of when they were born, and alphabetical order

**SELECT** birthdate, name

**FROM** people

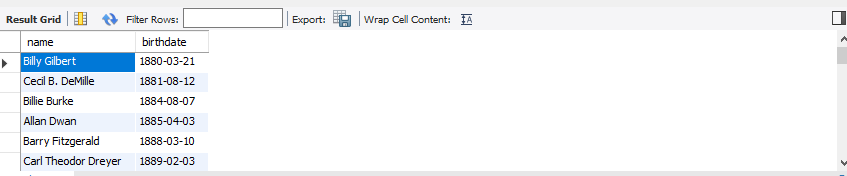
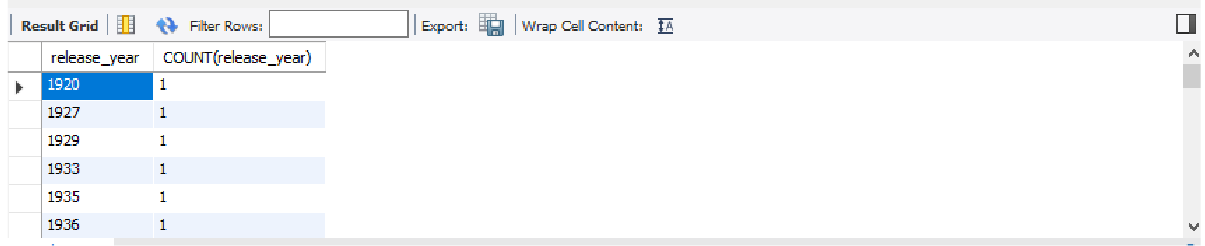
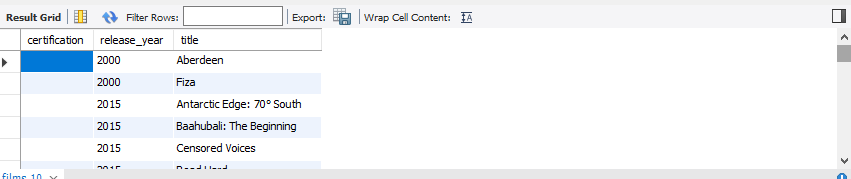
**ORDER BY** birthdate, name;

-- Get films from in 2000 or 2015, sorted in the order they were released, and how long they were **SELECT** release\_year, duration, title

**FROM** films

**WHERE** release\_year **IN** (2000, 2015)

**ORDER BY** release\_year, duration;



-- Get films between 2000 and 2015, sorted by certification and the year they were released **SELECT** certification, release\_year, title **FROM** films

**WHERE** release\_year **IN** (2000, 2015)

**ORDER BY** certification, release\_year;

-- Get people whose names start with A, B or C, (redundantly) ordered

**SELECT** name, birthdate

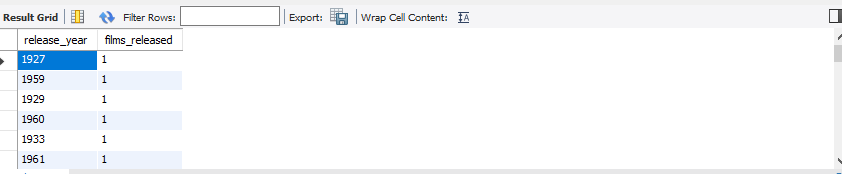
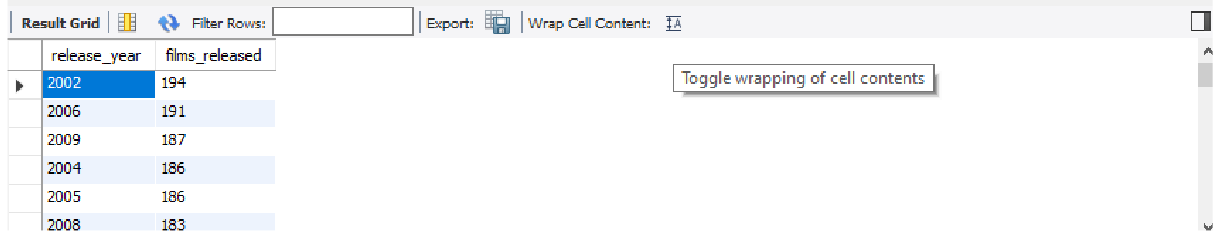
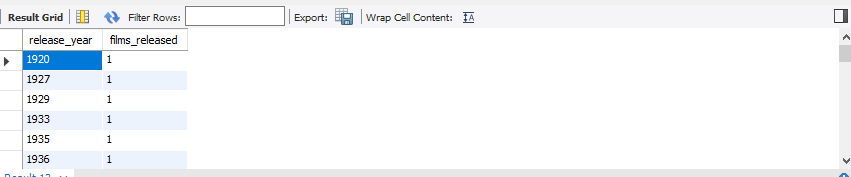
**FROM** people

**WHERE** name LIKE 'A%' OR name LIKE 'B%' OR name LIKE 'C%'

**ORDER BY** birthdate;

-- Get count of films made in each year **SELECT** release\_year, COUNT(release\_year) **FROM** films

**GROUP BY** release\_year;



-- Get count of films, group by release year then order by release year

**SELECT** release\_year, COUNT(title) **as** films\_released

**FROM** films

**GROUP BY** release\_year

**ORDER BY** release\_year;

-- Get count of films released in each year, ordered by count, lowest to highest

**SELECT** release\_year, COUNT(title) **AS** films\_released

**FROM** films

**GROUP BY** release\_year

**ORDER BY** films\_released;

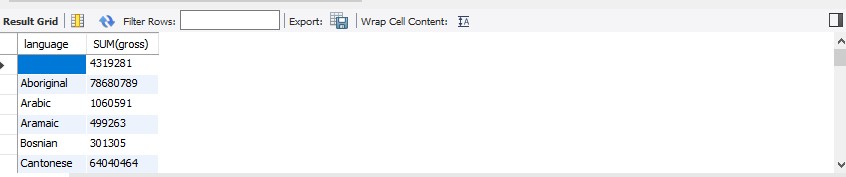
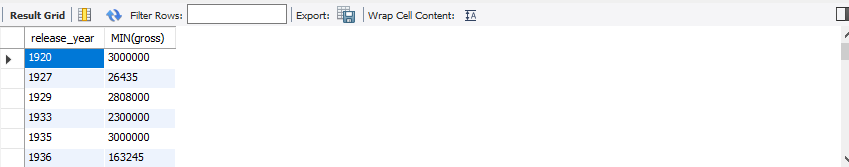
-- Get count of films released in each year, ordered by count highest to lowest

**SELECT** release\_year, COUNT(title) **AS** films\_released

**FROM** films

**GROUP BY** release\_year

**ORDER BY** films\_released **DESC**;



-- Get lowest box office earnings per year

**SELECT** release\_year, MIN(gross)

**FROM** films

**GROUP BY** release\_year

**ORDER BY** release\_year;

-- Get the total amount made in each language

**SELECT** language, SUM(gross)

**FROM** films

**GROUP BY** language;

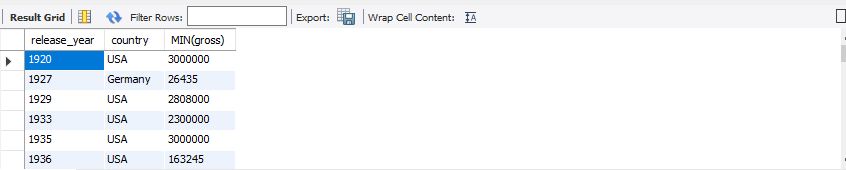
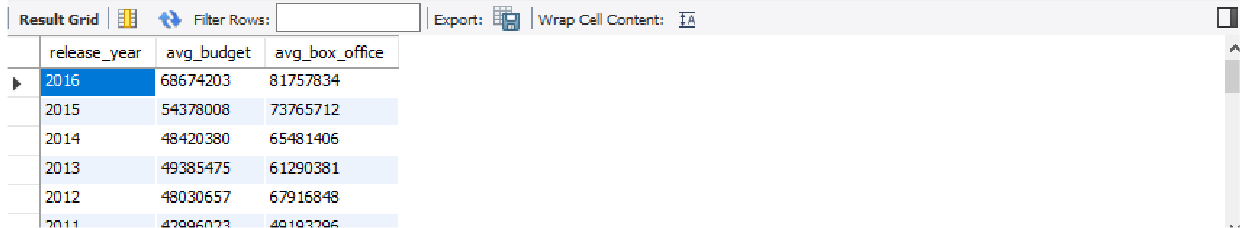
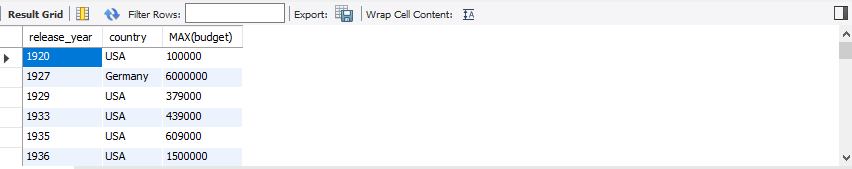
-- Get the total amount spent by each country

**SELECT** country, SUM(gross)

**FROM** films

**GROUP BY** country;

-- Get the max budget spent in making film for each year, for each country



**SELECT** release\_year, country, MAX(budget)

**FROM** films

**GROUP BY** release\_year, country

**ORDER BY** release\_year, country;

-- Get the lowest box office made by each country in each year

**SELECT** release\_year, country, MIN(gross)

**FROM** films

**GROUP BY** release\_year, country

**ORDER BY** release\_year, country;

-- Get the rounded average budget and average box office earnings for movies since 1990, but only if the average budget was greater than $60m in that year **SELECT** release\_year, ROUND(AVG(budget)) **AS** avg\_budget, ROUND(AVG(gross)) **AS** avg\_box\_office

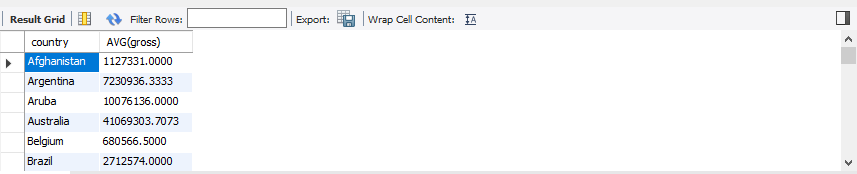
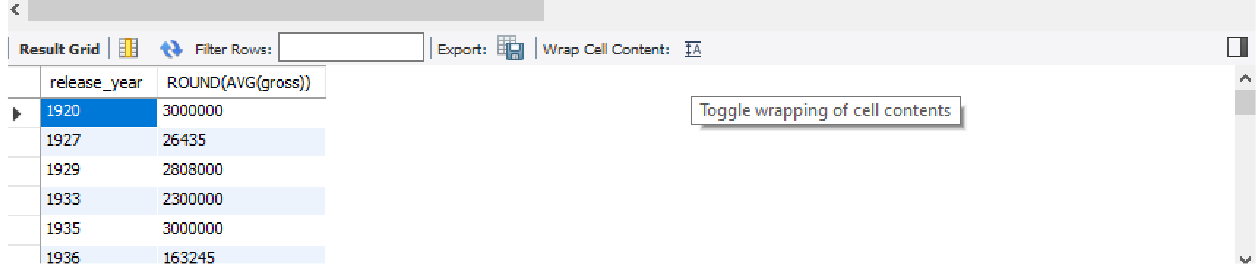
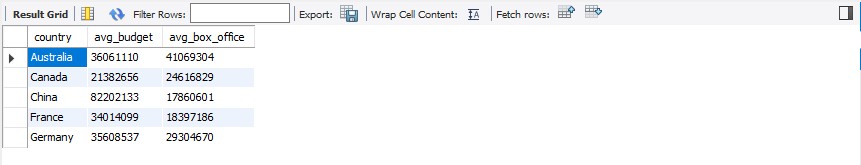
**FROM** films

**WHERE** release\_year > 1990

**GROUP BY** release\_year

**HAVING** AVG(budget) > 20000000

**ORDER BY** release\_year **DESC**;



-- Get the name, average budget, average box office take of countries who have made more than 10 films. Order by name, and get the top five

**SELECT** country, ROUND(AVG(budget)) **AS** avg\_budget, ROUND(AVG(gross)) **AS** avg\_box\_office

**FROM** films

**GROUP BY** country

**HAVING** COUNT(title) > 10

**ORDER BY** country

**LIMIT** 5;

-- Get the average amount made by each country

**SELECT** country, AVG(gross)

**FROM** films

**GROUP BY** country;

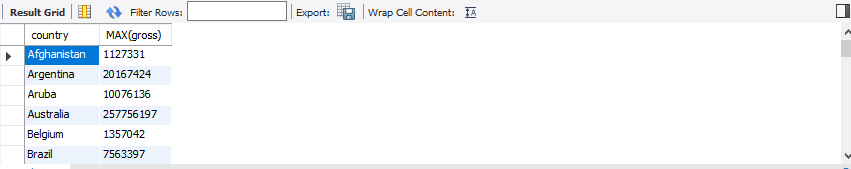
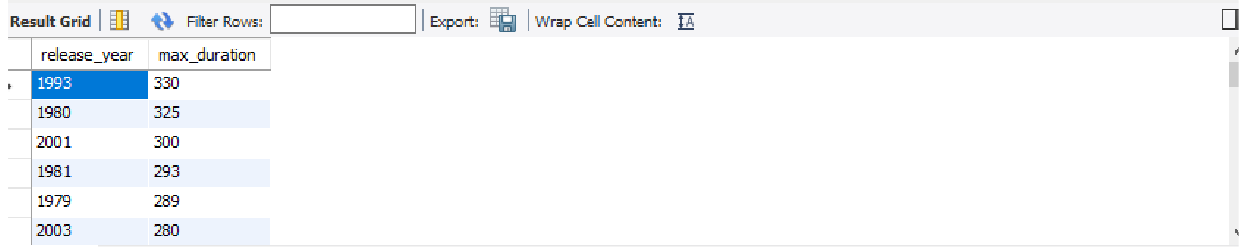
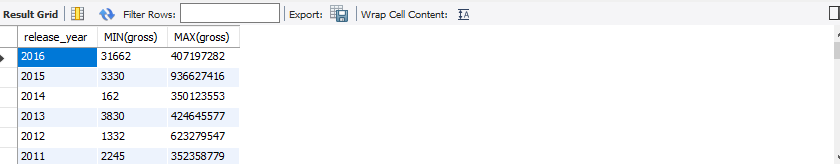
-- Get rounded average box office earnings per year

**SELECT** release\_year, ROUND(AVG(gross))

**FROM** films

**GROUP BY** release\_year

**ORDER BY** release\_year;



-- Get lowest and highest box office earnings per year

**SELECT** release\_year, MIN(gross), MAX(gross)

**FROM** films

**GROUP BY** release\_year

**ORDER BY** release\_year **DESC**;

-- Get the highest box office take per country

**SELECT** country, MAX(gross)

**FROM** films

**GROUP BY** country;

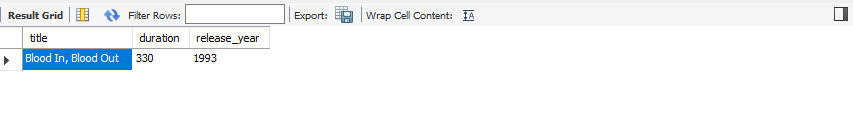
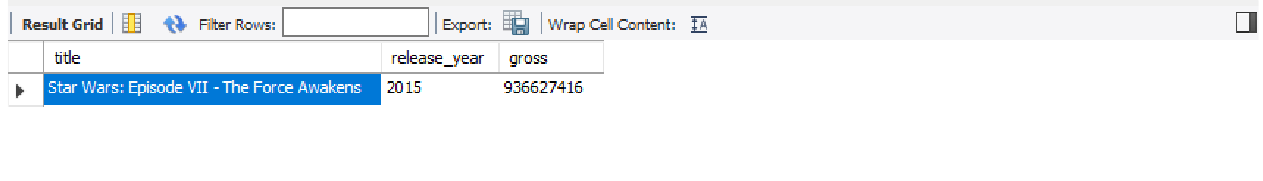
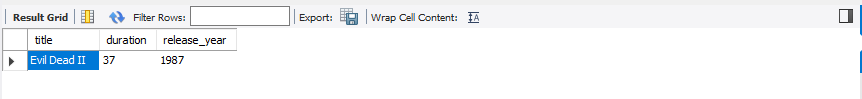
-- Longest duration per year

**SELECT** release\_year, MAX(duration) **AS** max\_duration

**FROM** films

**GROUP BY** release\_year

**ORDER BY** max\_duration **DESC**;



**-- Subqueries**

-- Get the title, duration and release year of the shortest film(s)

**SELECT** title, duration, release\_year

**FROM** films

**WHERE** duration = (

**SELECT** MIN(duration) **FROM** films

);

-- Get the title, duration and release year of the longest film(s)

**SELECT** title, duration, release\_year

**FROM** films

**WHERE** duration = ( **SELECT** MAX(duration) **FROM** films

);

-- Get the title, release\_year and box office take for the highest grossing film

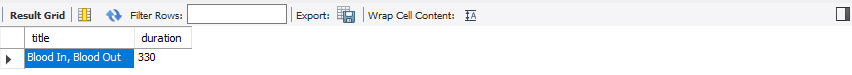
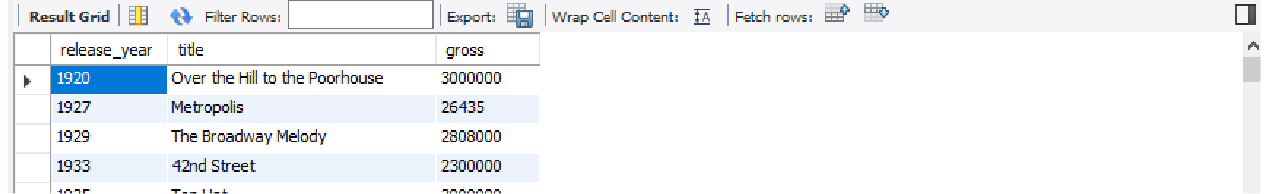
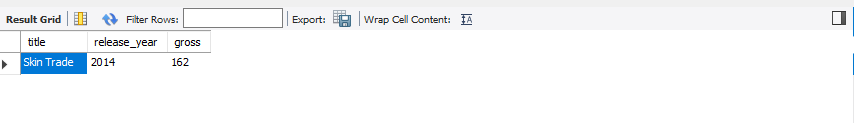
**SELECT** title, release\_year, gross

**FROM** films

**WHERE** gross = ( **SELECT** MAX(gross) **FROM** films

);

-- Get the title, release\_year and box office take for the lowest grossing film



**SELECT** title, release\_year, gross

**FROM** films

**WHERE** gross = ( **SELECT** MIN(gross) **FROM** films

);

-- Get the duration of the longest movie made in the USA

**SELECT** title, duration

**FROM** films

**WHERE** duration = ( **SELECT** MAX(duration) **FROM** films

**WHERE** country = 'USA'

);

-- Get details for the film with the lowest box office earnings per year

**SELECT** release\_year, title, gross

**FROM** films

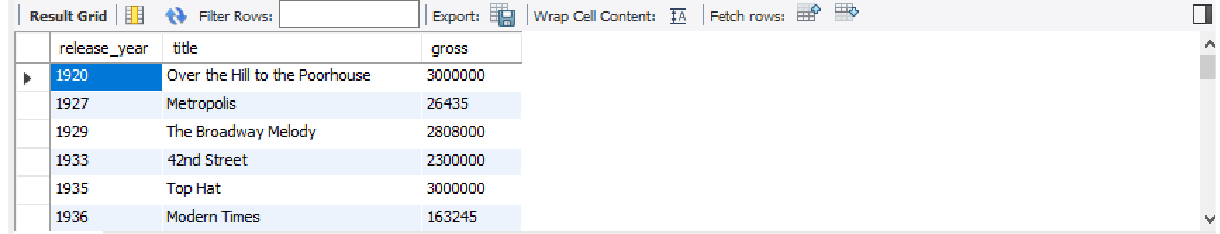
**WHERE** release\_year **IN** ( **SELECT** release\_year **FROM** films

**WHERE** gross **IN** ( **SELECT** MIN(gross) **FROM** films

**GROUP BY** release\_year

)

);



-- Get details for the film with the highest box office earnings per year

**SELECT** release\_year, title, gross

**FROM** films

**WHERE** release\_year **IN** ( **SELECT** release\_year **FROM** films

**WHERE** gross **IN** ( **SELECT** MAX(gross) **FROM** films

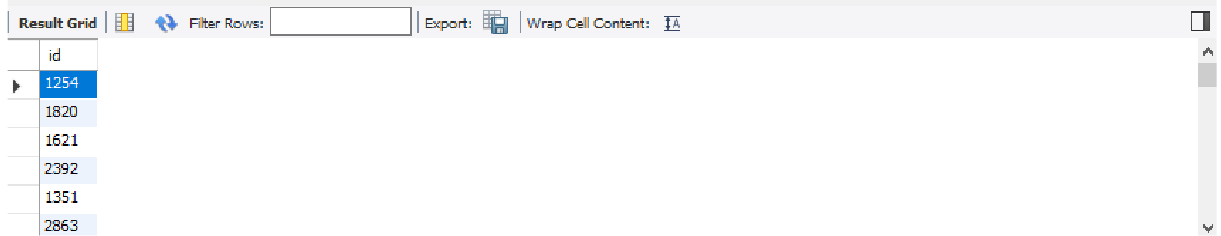
**GROUP BY** release\_year

)

);



**-- Joins**



-- Get film id of films whose review exist

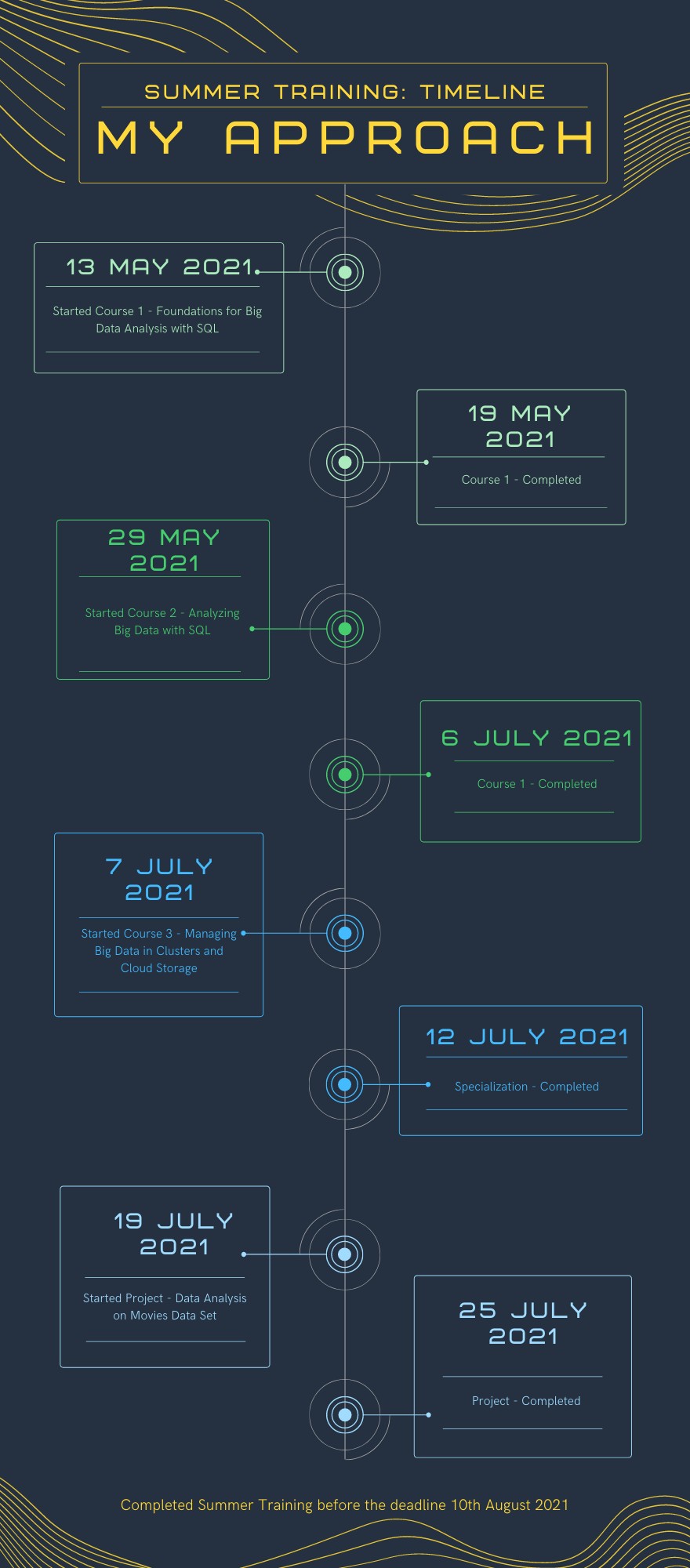
**SELECT** films.id

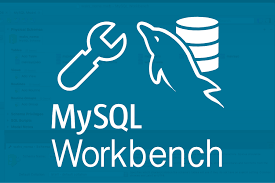
**FROM** films

**INNER JOIN** reviews

**ON** films.id = reviews.film\_id;

# Chapter 4 – Timeline of Training and Project



**Chapter 5 – Skills Gained from Training and**

# C:\Users\HP\Downloads\Summer Training\download-removebg-preview.pngProject



**Final Chapter – Conclusion and Future Outlook of Big Data, SQL Databases, and Data Analysis**

The future of SQL Server will depend on the future of the use of SQL as a query language. Relational Database Management Systems as we know them have not really changed much over the last two decades while just about every other subject relating to computing has. The success of SQL is in its simplicity and at lower levels of abstraction, we will always need a technology like this.

However, there are a handful of needs that SQL and RDBMS' simply don't provide in their current form.

Complex Interface – SQL has a difficult interface that makes few users uncomfortable while dealing with the database.

Cost – Some versions are costly and hence, programmers cannot access them.

Partial Control – Due to hidden business rules, complete control is not given to the database.

Almost every business problem or need involves the use and maintenance of data. Data is virtually the lifeblood of a business so it will always be important. After all, data – and big data – are just point-in-time recordings of business or operational events (something a person, machine, or business did.

Up until perhaps five years ago, most business data was still at a very coarse level - representing discrete transactions (e.g. purchases, trades, orders, line items, travel segments, etc.). With big data, we now have the ability to capture and analyze transactions that are happening at a finer, more granular level, so we're moving from transactional to behavioral understanding. A good example of this centers around e-commerce, and the contrast between tracking and analyzing purchases on the one hand, and the measurement and analysis of clickstream data, to understand customer behavior, on the

other. Big data analytics have gained popularity over the past decade, and many experts see the same to continue for the next decade.

Big data analytics is going to be mainstream with increased adoption among every industry and forms a virtuous cycle with more people wanting access to even bigger data.

However, often the requirements for big data analysis are really not well understood by the developers and business owners, thus creating an undesirable product.

For organizations to not waste precious time and money and manpower over these issues, there is a need to develop expertise and process of creating small-scale prototypes quickly and test them to demonstrate their correctness, matching with business goals.

A survey by Gartner found that 48% of the companies invested in big data in 2016, and nearly three-quarters of those surveyed had already invested, or were planning to invest in data analytics. Big data is helping companies in different sectors, from marketing to pharmaceutical companies to third sector organizations. By 2023, it is predicted that the amount of data that is worthy of being analyzed, will surprisingly triple.

Seeing and analyzing the applications of big data analytics, and the huge support that it provides to companies, it is clear that it is here to stay. It is efficient and predicts most of the data right and saves time and cost. Therefore, for every area touched by big data analytics the word "better" can be added in front of it, that is, better security, better training, better education, better business, etc. That is the potential of this technology.

# Summary of Report

In this report, I have shared my journey and learning throughout my summer training. I have discussed what I learnt from all the three courses of the specialization, what skills I have gained, and timeline of the summer training.

I have discussed in detail my entire data analysis using SQL project where I have calculated results using different SQL functions. In this project, I have calculated various results from this large database, like how many French movies are there, which movie earned the highest profit in the ’90s, which actor acted in most of the movies, what was the average duration of movies, which was the longest English movie, etc.

I have also discussed in brief the future of relational databases and Big Data Analytics.

# References

* <https://www.wikipedia.org/>. (Accessed on 6th Sept 2021).
* <https://www.quora.com/>. (Accessed on 10th Sept 2021).
* <https://www.datacamp.com/courses/introduction-to-sql> . Downloaded data sets used in this project. (Accessed on 11th Sept 2021).
* Kerry Koitzsch, Future of Big Data Analysis, published in book Pro Hadoop Data Analytics, December 2017, pp. 263-273.
* Ali Salehnia – South Dakota State University, Comparisons of Relational Databases with Big Data, 2015, pp. 01-17.
* Yasin N. Silva, Isadora Almeida, Michelle Queiroz, SQL from Traditional Databases to Big Data, published in the 47th ACM Technical Symposium, February 2016, pp. 01-03.
* Modern Big Data Analysis with SQL Specialization offered by CLOUDERA through Coursera.