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)

LPUOnline

Same Degree, Now Online.

Entitled by UGC

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

List of Tables and Figures

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

"Real-Time Stock Analysis and Prediction Using Python and Yahoo Finance" retrieves stock data from Yahoo Finance using web scraping or the YFinance API (only if web scraping becomes unavailable). By leveraging historical and real-time stock data, along with machine learning techniques in Python, the project utilizes libraries such as Pandas for data manipulation, NumPy for numerical operations, and Matplotlib for data visualization. The goal is to predict future stock performance, enabling investors to make informed decisions on which stocks to buy today.

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 – Reason for Choosing this Technology

My main reason behind choosing this technology was the exponential increase in the use of databases nowadays. In today's world, it won't be wrong to say that we are surrounded by data, data is everywhere from the marks in our report card, our bank account statements, what movies we have downloaded to binge-watch this weekend, etc.

But what actually is data? Data is unstructured facts and figures. Facts and figures relay something specific, but which are not organized in any way and which provide no further information regarding patterns, context, and other details.

But what is the use of unstructured data, we can get information from the data only if our data is contextualized, categorized, calculated, and condensed. This is why it is so important to arrange, calculate,

condense, filter, and optimize the data. And this is where databases come into the picture.

To say that the databases are everywhere would be an understatement. They virtually permeate our lives: Online stores, health care providers, clubs, libraries, video stores, beauty salons, travel agencies, phone companies, government agencies like FBI, INS, IRS, and

NASA, giant businesses like Netflix, Microsoft, Google they all use database.
So, I thought its the best time to get hands-on experience on the most demanded skill i.e. data analysis. Data analysis is the future, and the future will demand skills for jobs as functional analysts, data engineers, data scientists, and advanced analysts. I believed that after this training I will have the ability to analyze data and make informed recommendations to drive effective decision-making and this will turn me into an indispensable member of any team.

Chapter 3 – Project: Movies Data Analysis

Profile of the Project

A movie database finds its real application in online movie streaming sites, but the movies data set involved in this picture consists of records of more than 20000 movies of different languages, made in different countries, directed by different directors over a range of 100 years. This data set is very useful in calculating results from past movies to analyze trends in the movie industries across the globe and use the past experiences and trends as samples for working on newer projects.

The movies database consists of 4 tables, films, people, reviews, and roles. All the data sets combined consist of more than 20000 entries. The movie's database contains information regarding the name of the movies, the year when they were released, country of origin, duration of films, the language of films, the budget of the movie, and the profit earned. It also contains the name of people who were involved in the films, what was their roles like actors or directors, when they were born, and when they died. It also gives details about the number of reviews a movie received, what was the response it received from the audience, how was the response of the movie over social media, etc.

System Requirements Specification

The environmental specification specifies the hardware and software requirements for carrying out this project. The following are the hardware and software requirements.

Hardware –

	Minimum	Recommended
CPU	64bit x86 CPU	Multi Core 64bit x86 CPU, 8 GB RAM
RAM	4 GB	8 GB or higher
Display	1024×768	1920×1200 or higher

Software -

Operating System	Architecture
Oracle Linux 8 / Red Hat Enterprise Linux 8	x86_64
Ubuntu 21.04	x86_64
Ubuntu 20.04 LTS	x86_64
Windows Server 2019	x86_64
Windows 10	x86_64
macOS 11	x86_64

- Microsoft .NET Framework 4.5.2
- Microsoft Visual C++ Redistributable for Visual Studio 2019
- MySQL server
- MySQL Workbench

Technologies and Tools Used

MySQL Workbench - MySQL Workbench is a visual database design tool that integrates SQL development, administration, database design, creation and maintenance into a single integrated development environment for the MySQL database system.

SQL - SQL is a domain-specific language used in programming and designed for managing data held in a relational database management system, or for stream processing in a relational data stream management system.

CSV File - A comma-separated values file is a delimited text file that uses a comma to separate values. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. The use of the comma as a field separator is the source of the name for this file format.

Creately - Creately is a SaaS visual collaboration tool with diagramming and design capabilities designed by Cinergix. Creately has two versions: an online cloud edition and a downloadable offline edition for desktop which is compatible with Windows, Mac and Linux.

Design and Charts

Figure 3.1 - ER Diagram – Created using Creately

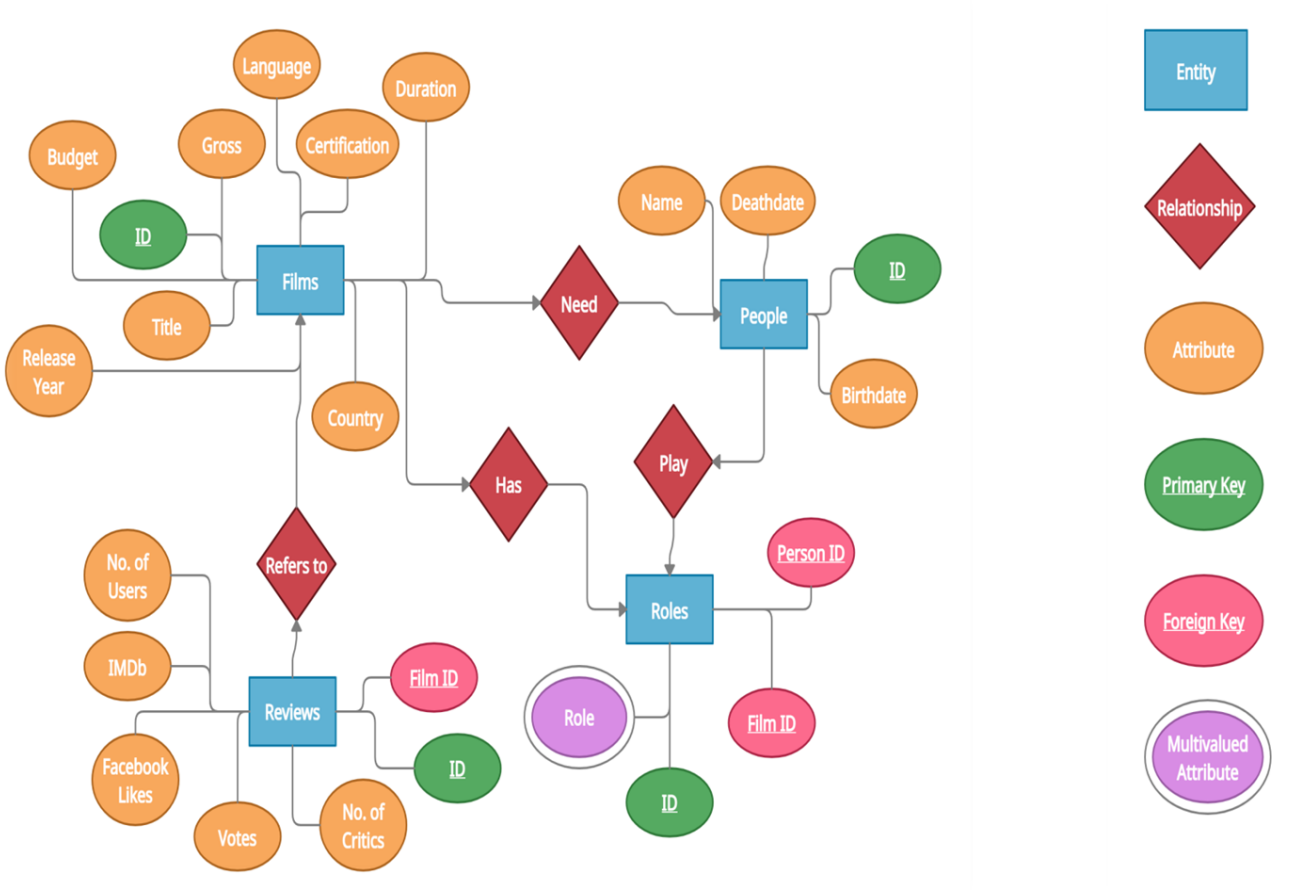
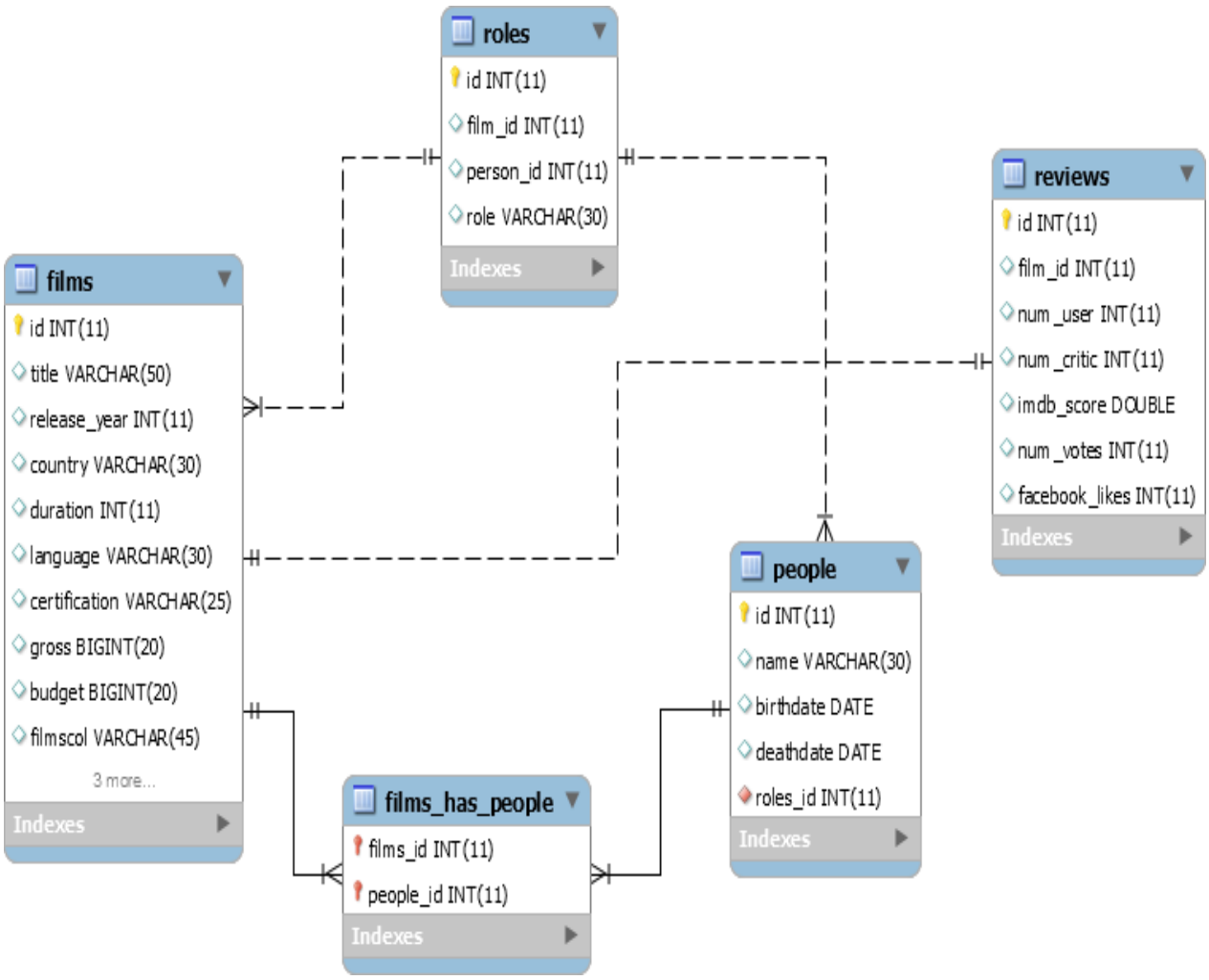


Figure 3.2 - ER Diagram – Created using MySQL Workbench



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                    INTEGER     PRIMARY KEY,
  name                  VARCHAR(30),
  birthdate             DATE,
  deathdate             DATE
);

CREATE TABLE reviews (
  id                    INTEGER     PRIMARY KEY,
  film_id               INTEGER,
  num_user              INTEGER,
  num_critic            INTEGER,
  imdb_score            REAL,
  num_votes             INTEGER,
  facebook_likes        INTEGER
);

CREATE TABLE roles (
  id                    INTEGER     PRIMARY KEY,
  film_id               INTEGER,
  person_id             INTEGER,
  role                  VARCHAR(30)
);
```

```sql
CREATE TABLE roles (
  id                    INTEGER       PRIMARY KEY,
  film_id               INTEGER,
  person_id             INTEGER,
  role                  VARCHAR(30)
);
```

```sql
-- Show All tables
SHOW TABLES;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Tables_in_movies
films
people
reviews
roles

```sql
-- Show content of films table
SELECT * FROM films;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

id	title	release_year	country	duration	language	certification	gross	budget
2	Over the Hill to the Poorhouse	1920	USA	110			3000000	100000
4	Metropolis	1927	Germany	145	German	Not Rated	26435	6000000
6	The Broadway Melody	1929	USA	100	English	Passed	2808000	379000
9	42nd Street	1933	USA	89	English	Unrated	2300000	439000
12	Top Hat	1935	USA	81	English	Approved	3000000	609000

```sql
-- Show content of people table
SELECT * FROM people;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

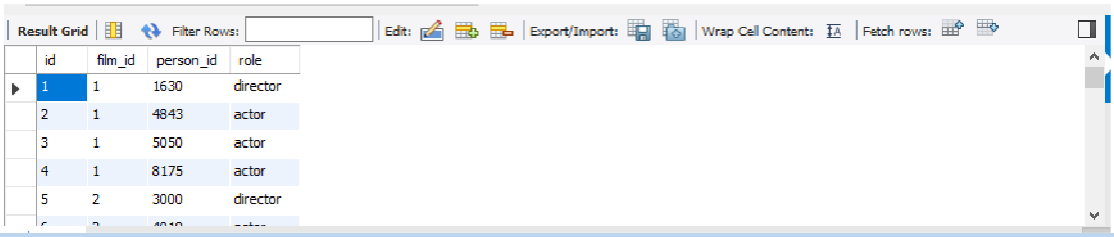
id	name	birthdate	deathdate
7	Aaliyah	1979-01-16	2001-08-25
61	Adolphe Menjou	1890-02-18	1963-10-29
63	Adrian Gonzalez	1938-05-08	1998-10-23
71	Adriana Caselotti	1916-05-16	1997-01-19
79	Adrienne Shelly	1966-06-24	2006-11-01

```sql
-- Show content of reviews table
SELECT * FROM reviews;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	film_id	num_user	num_critic	imdb_score	num_votes	facebook_likes
5	1254	1437	224	8	241030	13000
15	1820	113	41	7	8535	872
21	1621	181	93	6	44913	3000
27	2392	156	111	5	51252	621
51	143	64	15	7	3454	548

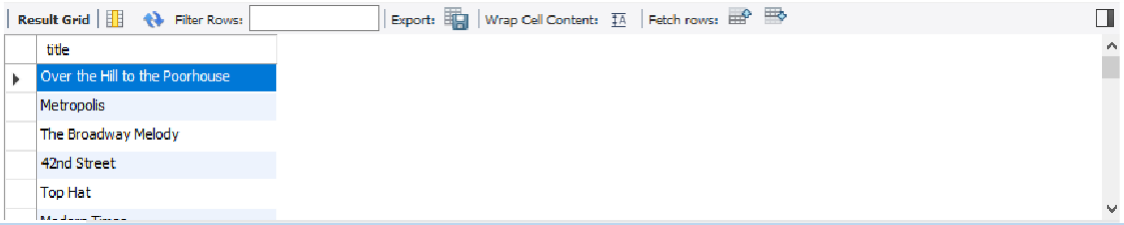
```sql
-- Show content of roles table
SELECT * FROM roles;
```



id	film_id	person_id	role
1	1	1630	director
2	1	4843	actor
3	1	5050	actor
4	1	8175	actor
5	2	3000	director

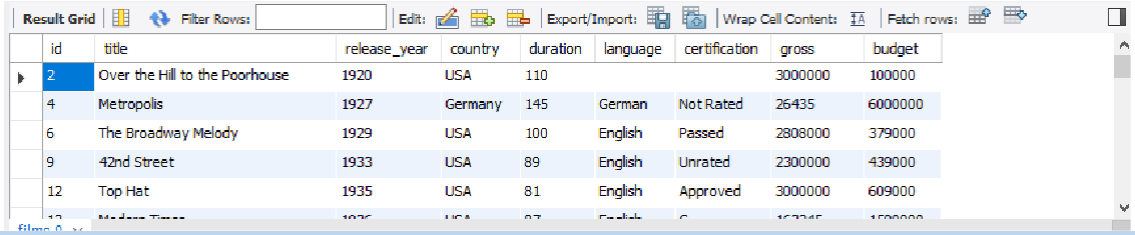
```sql
-- Selecting Columns: SELECT, SELECT DISTINCT
```

```sql
-- Get the title of every film
SELECT title
FROM films;
```



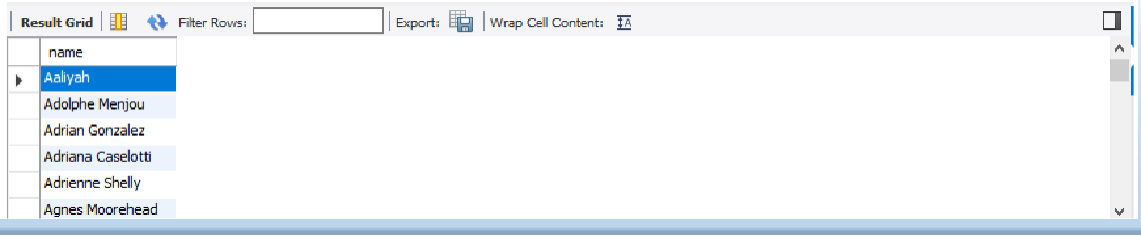
title
Over the Hill to the Poorhouse
Metropolis
The Broadway Melody
42nd Street
Top Hat

```sql
-- Get all details for every film
SELECT *
FROM films;
```



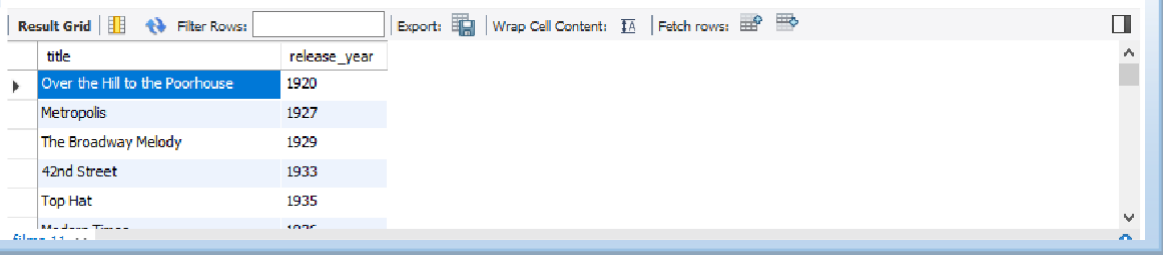
id	title	release_year	country	duration	language	certification	gross	budget
2	Over the Hill to the Poorhouse	1920	USA	110			3000000	100000
4	Metropolis	1927	Germany	145	German	Not Rated	26435	6000000
6	The Broadway Melody	1929	USA	100	English	Passed	2808000	379000
9	42nd Street	1933	USA	89	English	Unrated	2300000	439000
12	Top Hat	1935	USA	81	English	Approved	3000000	609000

```sql
-- Get the names of everyone involved in working
on the films
SELECT name
FROM people;
```



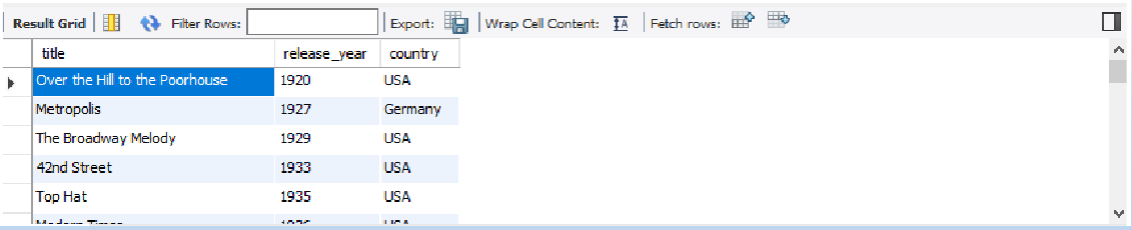
name
Aaliyah
Adolphe Menjou
Adrian Gonzalez
Adriana Caselotti
Adrienne Shelly
Agnes Moorehead

```sql
-- Get the title and release year of every film
SELECT title, release_year
FROM films;
```



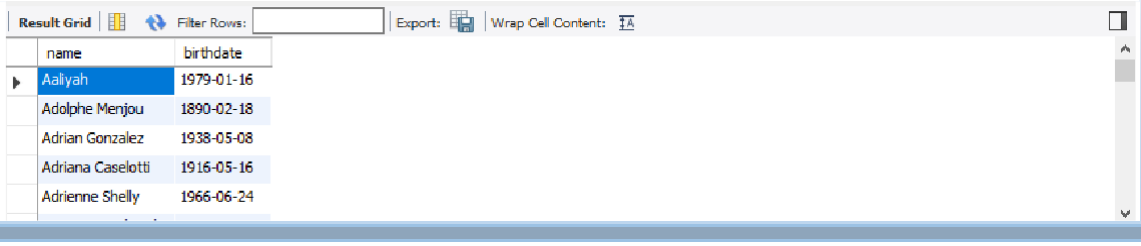
title	release_year
Over the Hill to the Poorhouse	1920
Metropolis	1927
The Broadway Melody	1929
42nd Street	1933
Top Hat	1935

```sql
-- Get the title, release year and country for every film
SELECT title, release_year, country
FROM films;
```



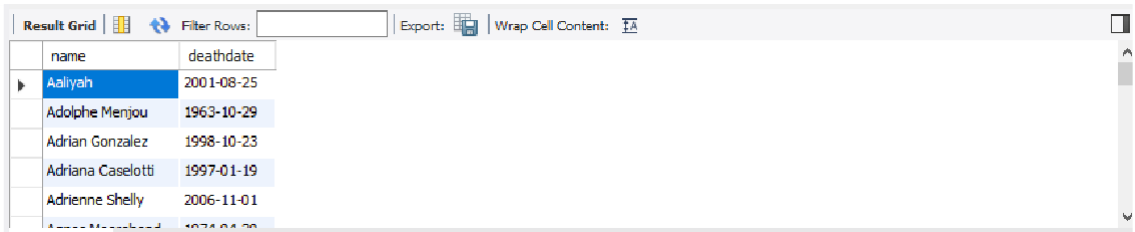
title	release_year	country
Over the Hill to the Poorhouse	1920	USA
Metropolis	1927	Germany
The Broadway Melody	1929	USA
42nd Street	1933	USA
Top Hat	1935	USA

```sql
-- Get every person's name and their date of birth where possible
SELECT name, birthdate
FROM people;
```



name	birthdate
Aaliyah	1979-01-16
Adolphe Menjou	1890-02-18
Adrian Gonzalez	1938-05-08
Adriana Caselotti	1916-05-16
Adrienne Shelly	1966-06-24

```sql
-- Get every person name and their date of death where possible
SELECT name, deathdate
FROM people;
```

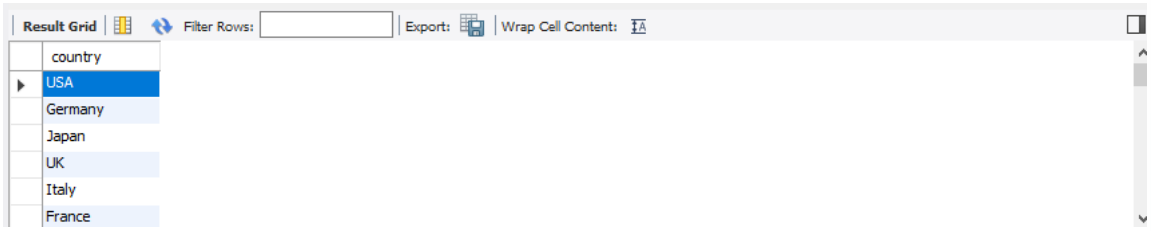


name	deathdate
Aaliyah	2001-08-25
Adolphe Menjou	1963-10-29
Adrian Gonzalez	1998-10-23
Adriana Caselotti	1997-01-19
Adrienne Shelly	2006-11-01

```sql
-- Get everyone's name, date of birth, and date of death (where possible)
SELECT name, birthdate, deathdate
FROM people;
```

name	birthdate	deathdate
Aaliyah	1979-01-16	2001-08-25
Adolphe Menjou	1890-02-18	1963-10-29
Adrian Gonzalez	1938-05-08	1998-10-23
Adriana Caselotti	1916-05-16	1997-01-19
Adrienne Shelly	1966-06-24	2006-11-01

```sql
- Get all the different countries
SELECT DISTINCT country
FROM films;
```

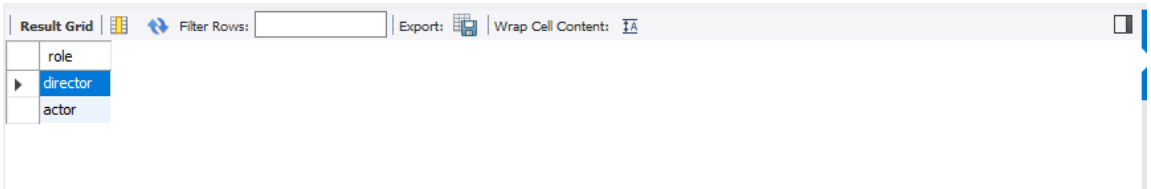


country
USA
Germany
Japan
UK
Italy
France

```sql
-- Get all the different film languages
SELECT DISTINCT language
FROM films;
```

language
German
English
Japanese
Italian
French

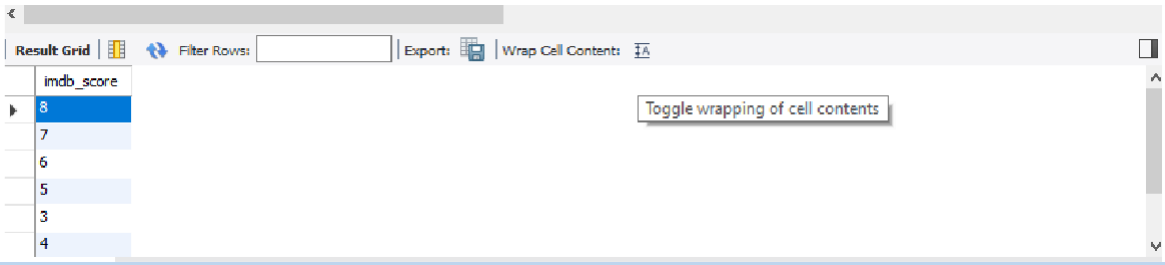
```sql
-- Get the different types of film roles
SELECT DISTINCT role
FROM roles;
```



role
director
actor

```sql
-- Get all the different certification categories
SELECT DISTINCT certification
FROM films;
```


```sql
-- Get all the different IMDB scores
SELECT DISTINCT imdb_score
FROM reviews;
```




```sql
-- Aggregate Functions: COUNT, SUM, AVG, MIN, MAX
```

```sql
-- 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
```

```sql
-- 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
```

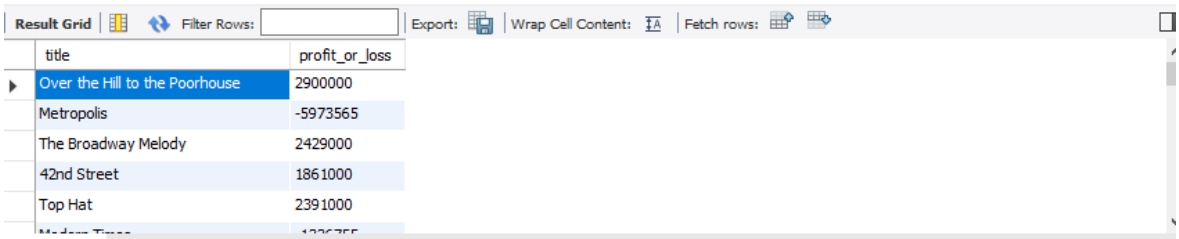
```sql
-- Get both the lowest and highest grossing films,
for comparision
SELECT MIN(gross), MAX(gross)
FROM films;
```

MIN(gross)	MAX(gross)
162	936627416

```sql
-- Get the highest number of Facebook likes for any
film
SELECT MAX(facebook_likes)
FROM reviews;
-- Result 150000
```

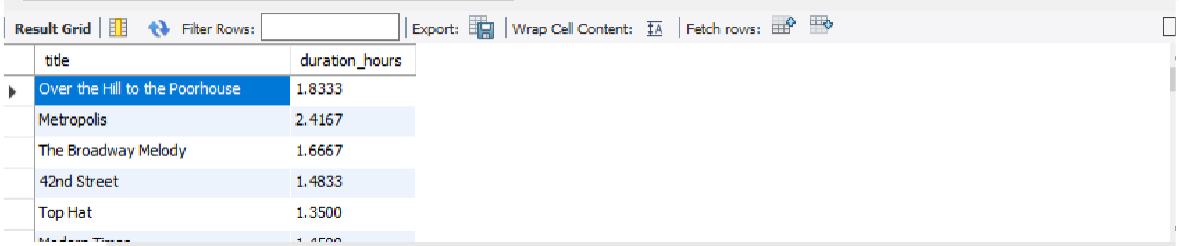
-- Aliasing and Basic Arithmetic

```sql
-- Get the profit (or loss) for each movie, where
possible
SELECT title, gross - budget
AS profit_or_loss
FROM films;
```



title	profit_or_loss
Over the Hill to the Poorhouse	2900000
Metropolis	-5973565
The Broadway Melody	2429000
42nd Street	1861000
Top Hat	2391000

```sql
-- Get the duration in hours for each film
SELECT title, duration / 60.0 AS duration_hours
FROM films;
```



title	duration_hours
Over the Hill to the Poorhouse	1.8333
Metropolis	2.4167
The Broadway Melody	1.6667
42nd Street	1.4833
Top Hat	1.3500

```sql
-- 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
```

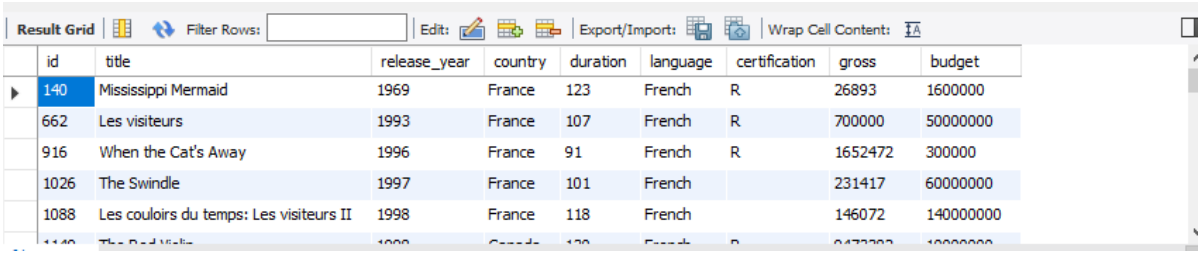
```sql
-- Rounding Functions: ROUND, FLOOR, CEILING
```

```sql
-- 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
```

```sql
-- 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
```

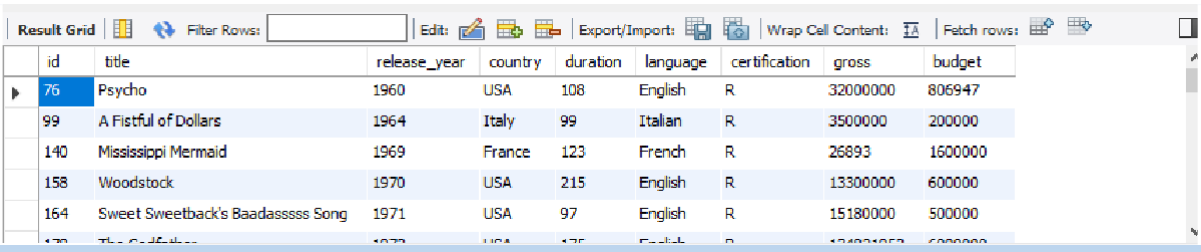
```sql
-- Filtering: WHERE, =, <>, <, <=, >, >=, AND, OR
```

```sql
-- Get all French language films
SELECT *
FROM films
WHERE language = 'French';
```



id	title	release_year	country	duration	language	certification	gross	budget
140	Mississippi Mermaid	1969	France	123	French	R	26893	1600000
662	Les visiteurs	1993	France	107	French	R	700000	50000000
916	When the Cat's Away	1996	France	91	French	R	1652472	300000
1026	The Swindle	1997	France	101	French		231417	60000000
1088	Les couloirs du temps: Les visiteurs II	1998	France	118	French		146072	140000000

```sql
SELECT COUNT(*)
FROM films
WHERE language = 'Hindi';
-- Result 10

-- Get all movies with an R certification
SELECT *
FROM films
WHERE certification = 'R';
```



id	title	release_year	country	duration	language	certification	gross	budget
76	Psycho	1960	USA	108	English	R	32000000	806947
99	A Fistful of Dollars	1964	Italy	99	Italian	R	3500000	200000
140	Mississippi Mermaid	1969	France	123	French	R	26893	1600000
158	Woodstock	1970	USA	215	English	R	13300000	600000
164	Sweet Sweetback's Baadasssss Song	1971	USA	97	English	R	15180000	500000

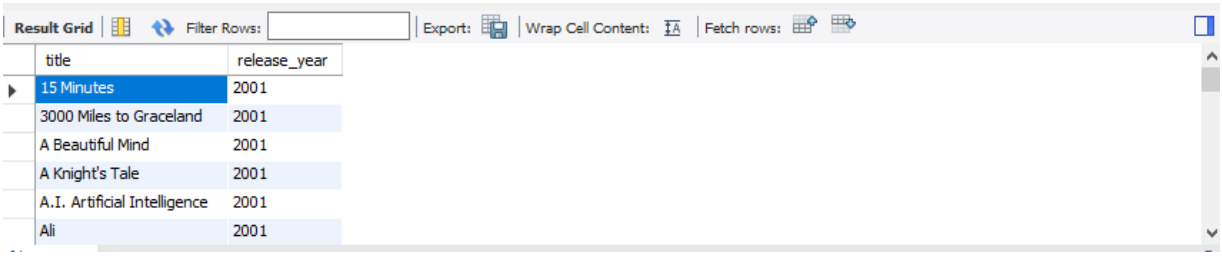
```sql
-- Get all films released in 2016
SELECT *
FROM films
WHERE release_year = 2016;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	title	release_year	country	duration	language	certification	gross	budget
4821	10 Cloverfield Lane	2016	USA	104	English	PG-13	71897215	15000000
4822	13 Hours	2016	USA	144	English	R	52822418	50000000
4825	Alice Through the Looking Glass	2016	USA	113	English	PG	76846624	170000000
4826	Allegiant	2016	USA	120	English	PG-13	66002193	110000000
4829	Bad Moms	2016	USA	100	English	R	55461307	20000000
4830	Bad Moms	2016	USA	100	English	R	55461307	20000000

```sql
-- 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
```

```sql
-- Get the title and release year of films released since 2000
SELECT title, release_year
FROM films
WHERE release_year > 2000;
```



title	release_year
15 Minutes	2001
3000 Miles to Graceland	2001
A Beautiful Mind	2001
A Knight's Tale	2001
A.I. Artificial Intelligence	2001
Ali	2001

```sql
-- Get all Spanish films released before 2000
SELECT title, release_year
FROM films
WHERE release_year < 2000
AND language = 'Spanish';
```

title	release_year
El Mariachi	1992
La otra conquista	1998
Tango	1998

```sql
-- Get the all Spanish films released since 2000
SELECT *
FROM films
WHERE release_year > 2000
AND language = 'Spanish';
```

id	title	release_year	country	duration	language	certification	gross	budget
1695	Y Tu Mamá También	2001	Mexico	106	Spanish	R	13622333	2000000
1757	El crimen del padre Amaro	2002	Mexico	118	Spanish	R	5709616	1800000
1807	Mondays in the Sun	2002	Spain	113	Spanish	R	146402	4000000
2175	Maria Full of Grace	2004	Colombia	101	Spanish	R	6517198	3000000
2246	The Holy Girl	2004	Argentina	106	Spanish	R	304124	1400000

```sql
-- 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
```

```sql
-- 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';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

title	release_year
Arachnophobia	1990
Back to the Future Part III	1990
Child's Play 2	1990
Dances with Wolves	1990
Days of Thunder	1990
Dick Tracy	1990

```sql
-- 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;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	title	release_year	country	duration	language	certification	gross	budget
1515	Alias Betty	2001	France	103	French		206400	50000000
1518	Amélie	2001	France	122	French	R	33201661	77000000
1692	Wasabi	2001	France	94	French	R	81525	15300000
1701	8 Women	2002	France	111	French	R	3076425	8000000
1795	L'auberge espagnole	2002	France	111	French	R	3895664	5300000

```sql
-- Get films released in the 90s
SELECT title, release_year
FROM films
WHERE release_year >= 1990 AND release_year <= 2000;
```

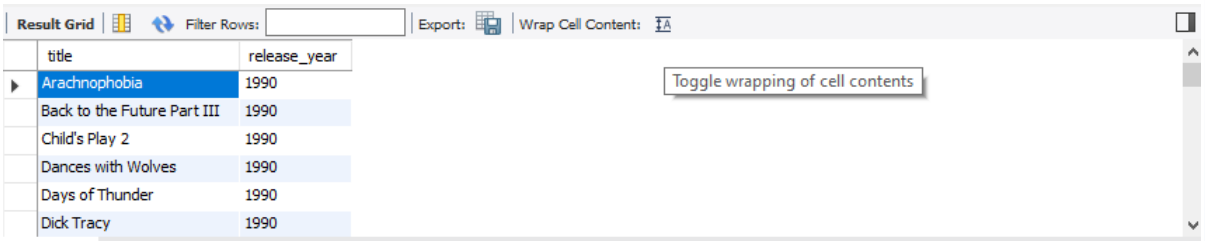
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

title	release_year
Arachnophobia	1990
Back to the Future Part III	1990
Child's Play 2	1990
Dances with Wolves	1990
Days of Thunder	1990
Dick Tracy	1990

```sql
-- 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
```

```sql
-- Advanced Filtering: BETWEEN, IN, IS NULL, IS NOT
NULL, LIKE, NOT LIKE
```

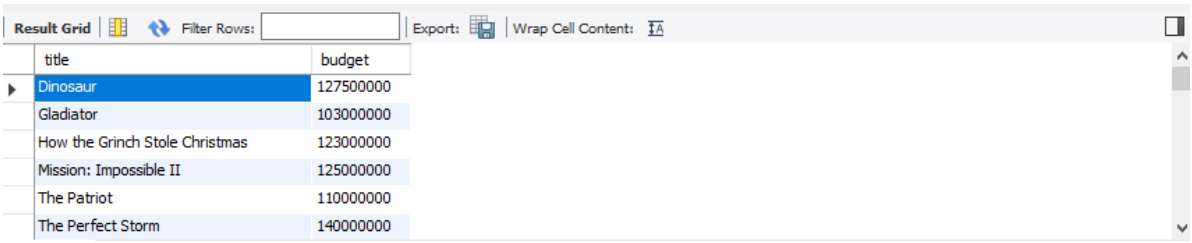
```sql
-- Get films released in the 90s
SELECT title, release_year
FROM films
WHERE release_year BETWEEN 1990 AND 2000;
```



title	release_year
Arachnophobia	1990
Back to the Future Part III	1990
Child's Play 2	1990
Dances with Wolves	1990
Days of Thunder	1990
Dick Tracy	1990

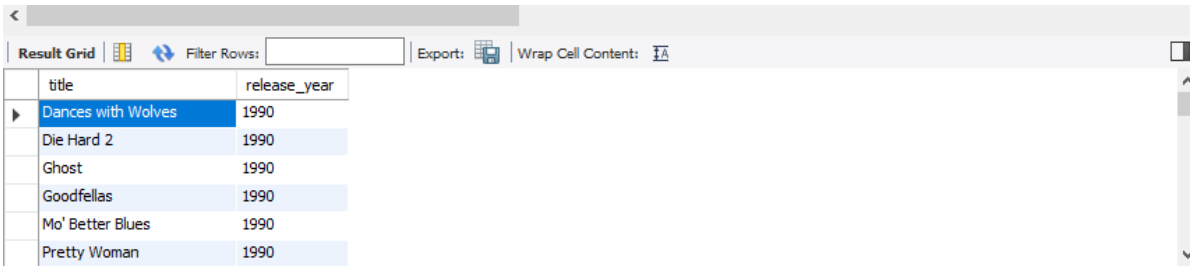
```sql
-- Get the number of films released in the 90s
SELECT COUNT(*)
FROM films
WHERE release_year BETWEEN 1990 AND 2000;
-- Result 111.3194

-- 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;
```



title	budget
Dinosaur	127500000
Gladiator	103000000
How the Grinch Stole Christmas	123000000
Mission: Impossible II	125000000
The Patriot	110000000
The Perfect Storm	140000000

```
-- 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;
```

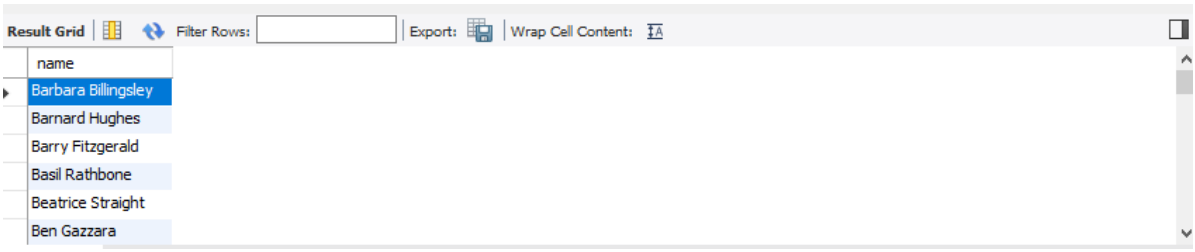


title	release_year
Dances with Wolves	1990
Die Hard 2	1990
Ghost	1990
Goodfellas	1990
Mo' Better Blues	1990
Pretty Woman	1990

```
-- Get the names of people who are still alive
SELECT name
FROM people
WHERE deathdate IS NULL OR deathdate = '';
```

name

```sql
-- Get people whose names begin with 'B'
SELECT name
FROM people
WHERE name LIKE 'B%';
```

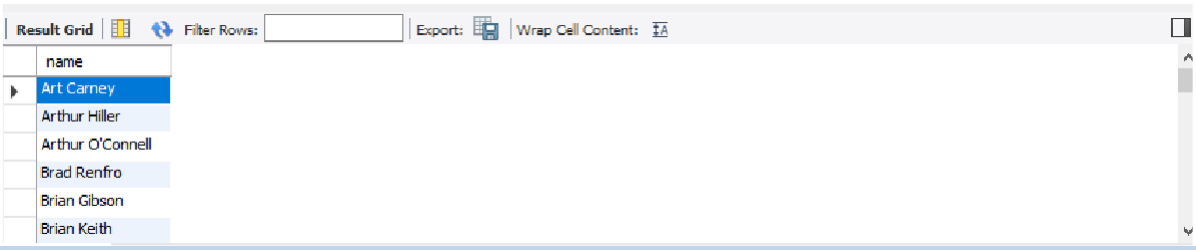


name
Barbara Billingsley
Barnard Hughes
Barry Fitzgerald
Basil Rathbone
Beatrice Straight
Ben Gazzara

```sql
-- Get people whose names begin with 'Br'
SELECT name
FROM people
WHERE name LIKE 'Br%';
```

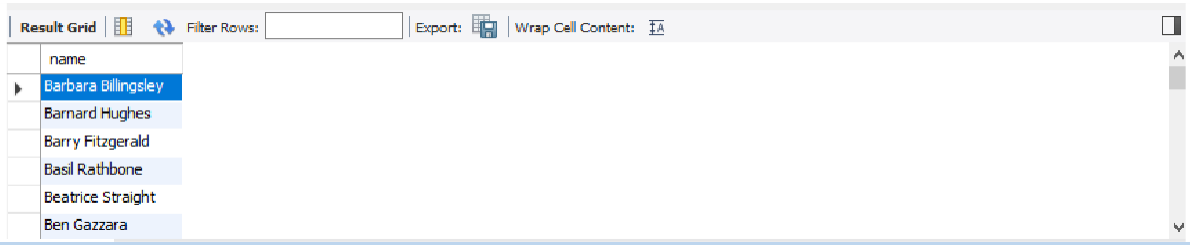
name
Brad Renfro
Brian Gibson
Brian Keith
Brigitte Helm
Brock Peters
Bruce Malmuth

```sql
-- Get people whose names have 'r' as the second
letter
SELECT name
FROM people
WHERE name LIKE '_r%';
```



name
Art Carney
Arthur Hiller
Arthur O'Connell
Brad Renfro
Brian Gibson
Brian Keith

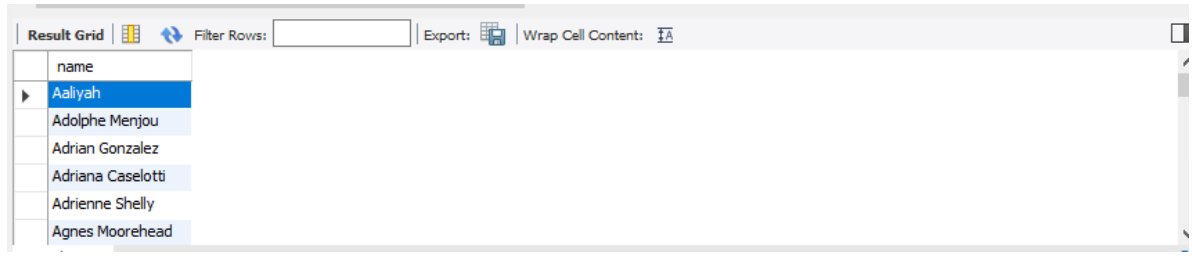
```sql
-- Get people whose names don't start with A
SELECT name
FROM people
WHERE name NOT LIKE 'A%';
```



name
Barbara Billingsley
Barnard Hughes
Barry Fitzgerald
Basil Rathbone
Beatrice Straight
Ben Gazzara

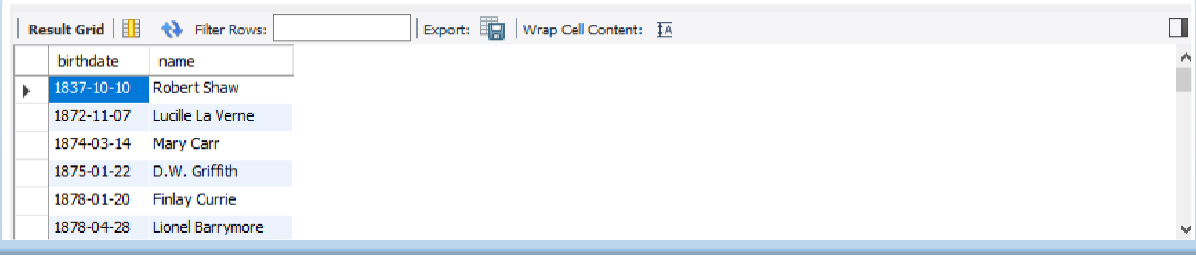
-- Sorting and Grouping

```sql
-- Get people, sort by name
SELECT name
FROM people ORDER BY name;
```



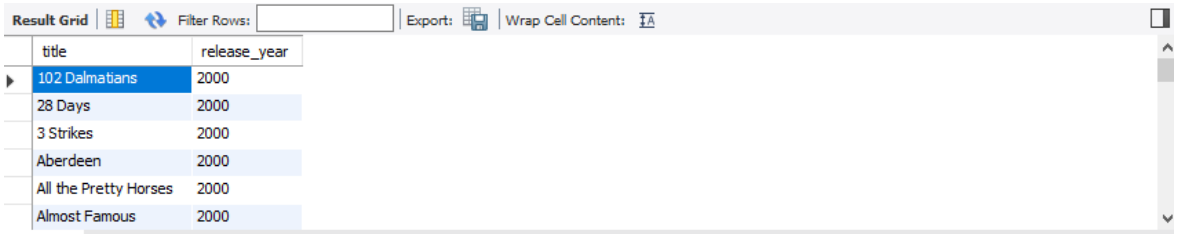
name
Aaliyah
Adolphe Menjou
Adrian Gonzalez
Adriana Caselotti
Adrienne Shelly
Agnes Moorehead

```sql
-- Get people, in order of when they were born
SELECT birthdate, name
FROM people
ORDER BY birthdate;
```



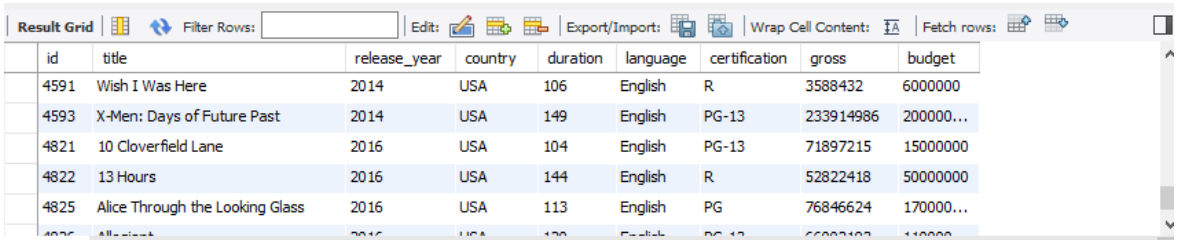
birthdate	name
1837-10-10	Robert Shaw
1872-11-07	Lucille La Verne
1874-03-14	Mary Carr
1875-01-22	D.W. Griffith
1878-01-20	Finlay Currie
1878-04-28	Lionel Barrymore

```sql
-- 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;
```



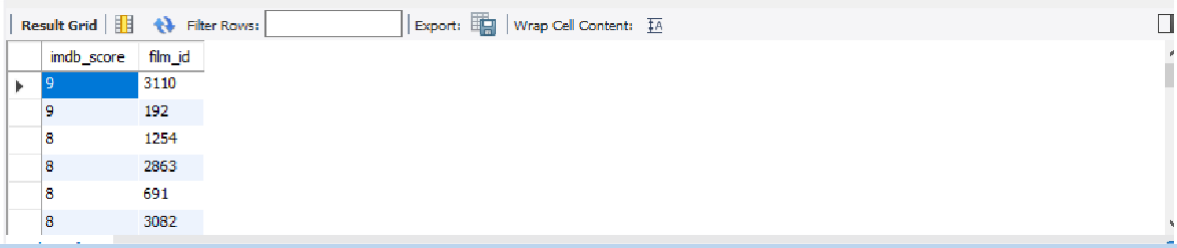
title	release_year
102 Dalmatians	2000
28 Days	2000
3 Strikes	2000
Aberdeen	2000
All the Pretty Horses	2000
Almost Famous	2000

```sql
-- Get all films except those released in 2015, and order them
SELECT *
FROM films
WHERE release_year <> 2015
ORDER BY release_year;
```



id	title	release_year	country	duration	language	certification	gross	budget
4591	Wish I Was Here	2014	USA	106	English	R	3588432	6000000
4593	X-Men: Days of Future Past	2014	USA	149	English	PG-13	233914986	200000...
4821	10 Cloverfield Lane	2016	USA	104	English	PG-13	71897215	15000000
4822	13 Hours	2016	USA	144	English	R	52822418	50000000
4825	Alice Through the Looking Glass	2016	USA	113	English	PG	76846624	170000...

```sql
-- 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;
```



imdb_score	film_id
9	3110
9	192
8	1254
8	2863
8	691
8	3082

```sql
-- Get the titles of films in reverse order
SELECT *
FROM films
ORDER BY title DESC;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

id	title	release_year	country	duration	language	certification	gross	budget
2293	Æon Flux	2005	USA	93	English	PG-13	25857987	62000000
3176	[Rec] 2	2009	Spain	85	Spanish	R	27024	5600000
2747	Zoom	2006	USA	83	English	PG	11631245	35000000
4926	Zoolander 2	2016	USA	102	English	PG-13	28837115	50000000
1696	Zoolander	2001	Germany	90	English	PG-13	45162741	28000000
3886	Zookeeper	2011	USA	102	English	PG	80360866	80000000

```sql
-- Get people, in order of when they were born, and alphabetical order
SELECT birthdate, name
FROM people
ORDER BY birthdate, name;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

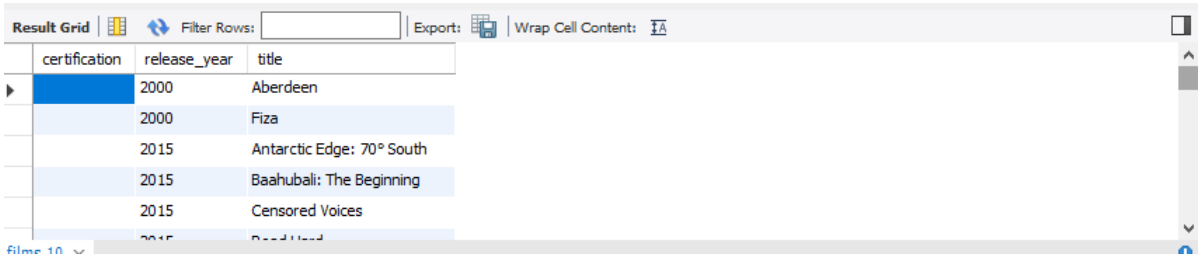
birthdate	name
1837-10-10	Robert Shaw
1872-11-07	Lucille La Verne
1874-03-14	Mary Carr
1875-01-22	D.W. Griffith
1878-01-20	Finlay Currie
1878-04-28	Lionel Barrymore

```sql
-- 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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

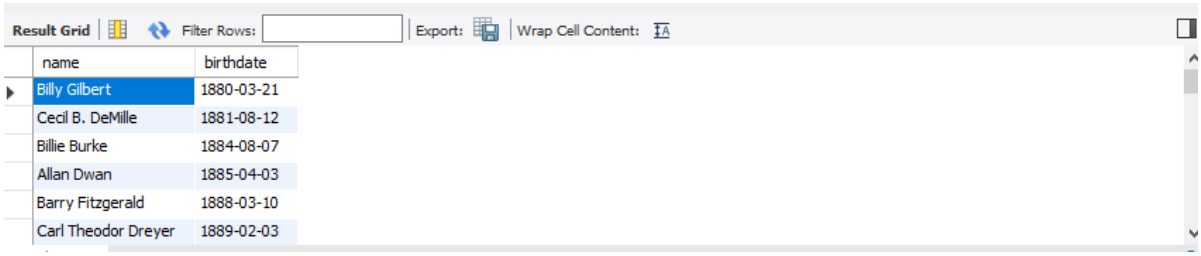
release_year	duration	title
2000	77	The Tigger Movie
2000	78	Rugrats in Paris: The Movie
2000	78	Tadpole
2000	78	The Emperor's New Groove
2000	81	Screwed
2000	82	3 Strikes

```
-- 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;
```



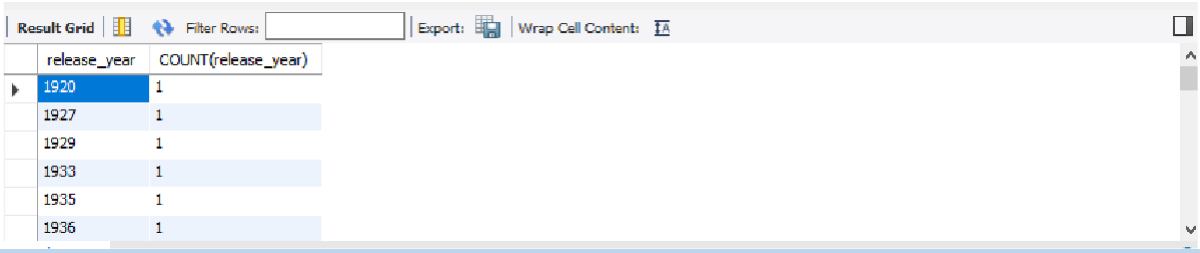
certification	release_year	title
	2000	Aberdeen
	2000	Fiza
	2015	Antarctic Edge: 70° South
	2015	Baahubali: The Beginning
	2015	Censored Voices

```
-- 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;
```



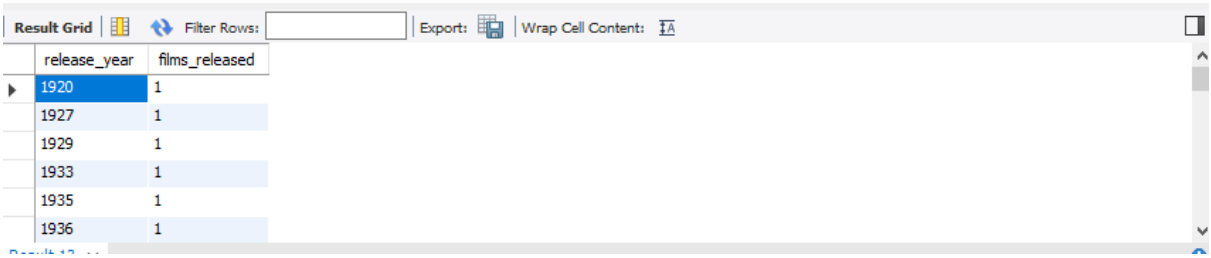
name	birthdate
Billy Gilbert	1880-03-21
Cecil B. DeMille	1881-08-12
Billie Burke	1884-08-07
Allan Dwan	1885-04-03
Barry Fitzgerald	1888-03-10
Carl Theodor Dreyer	1889-02-03

```
-- Get count of films made in each year
SELECT release_year, COUNT(release_year)
FROM films
GROUP BY release_year;
```



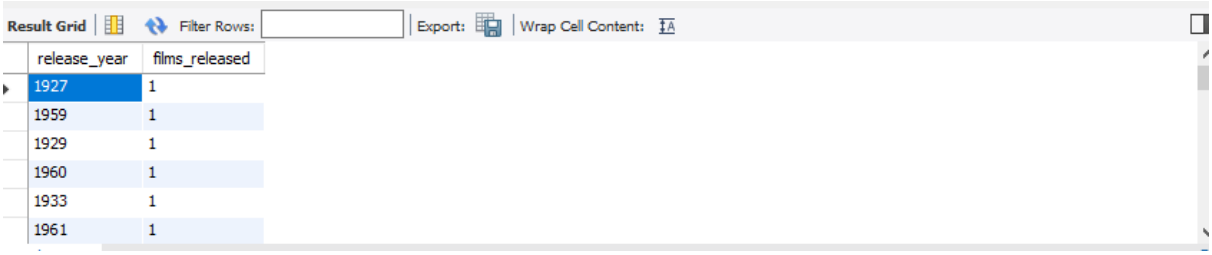
release_year	COUNT(release_year)
1920	1
1927	1
1929	1
1933	1
1935	1
1936	1

```sql
-- 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;
```



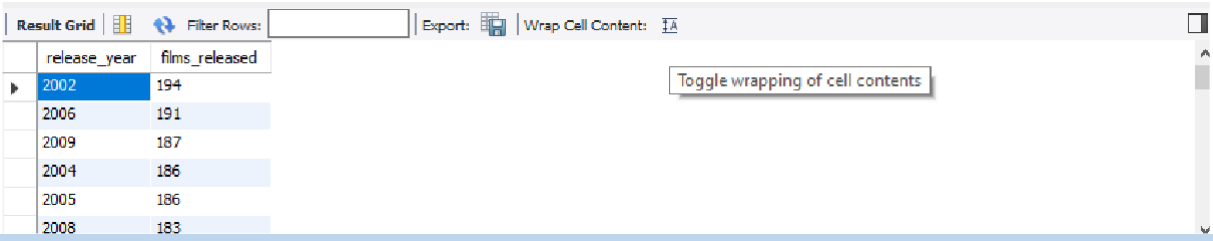
release_year	films_released
1920	1
1927	1
1929	1
1933	1
1935	1
1936	1

```sql
-- 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;
```



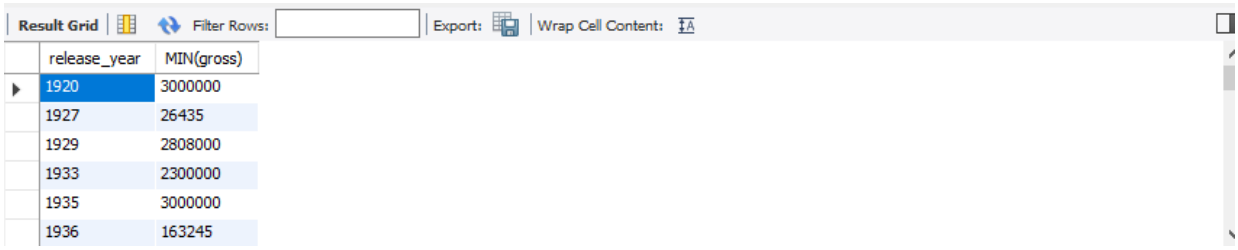
release_year	films_released
1927	1
1959	1
1929	1
1960	1
1933	1
1961	1

```sql
-- 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;
```



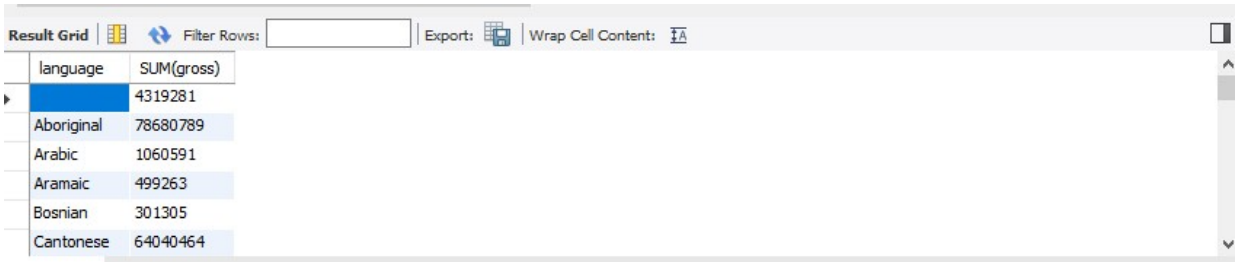
release_year	films_released
2002	194
2006	191
2009	187
2004	186
2005	186
2008	183

```sql
-- Get lowest box office earnings per year
SELECT release_year, MIN(gross)
FROM films
GROUP BY release_year
ORDER BY release_year;
```



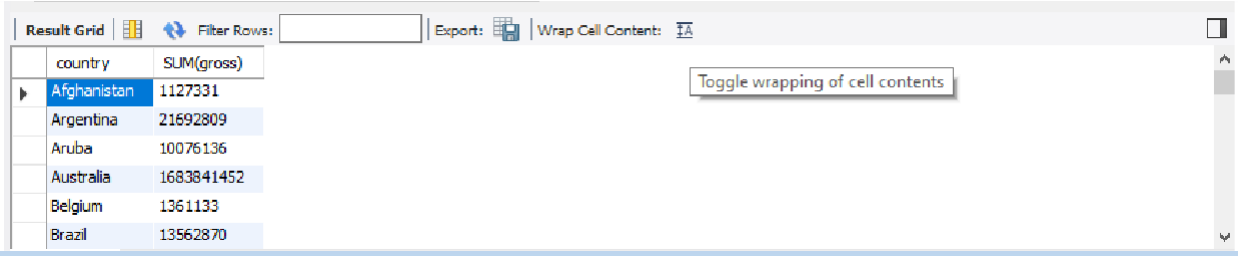
release_year	MIN(gross)
1920	3000000
1927	26435
1929	2808000
1933	2300000
1935	3000000
1936	163245

```sql
-- Get the total amount made in each language
SELECT language, SUM(gross)
FROM films
GROUP BY language;
```



language	SUM(gross)
	4319281
Aboriginal	78680789
Arabic	1060591
Aramaic	499263
Bosnian	301305
Cantonese	64040464

```sql
-- Get the total amount spent by each country
SELECT country, SUM(gross)
FROM films
GROUP BY country;
```



country	SUM(gross)
Afghanistan	1127331
Argentina	21692809
Aruba	10076136
Australia	1683841452
Belgium	1361133
Brazil	13562870

```sql
-- 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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

release_year	country	MAX(budget)
1920	USA	100000
1927	Germany	6000000
1929	USA	379000
1933	USA	439000
1935	USA	609000
1936	USA	1500000

```sql
-- 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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

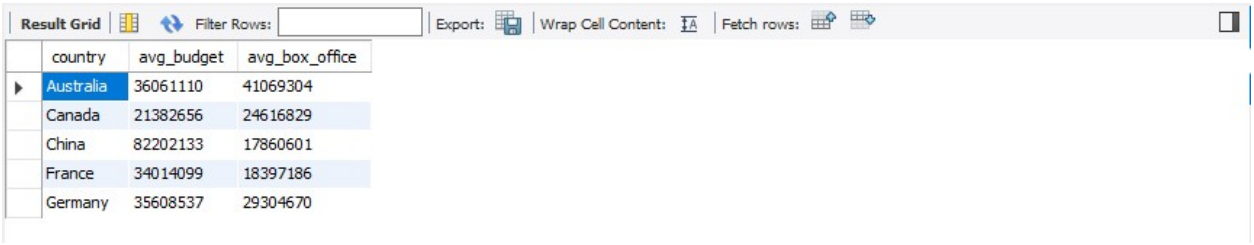
release_year	country	MIN(gross)
1920	USA	3000000
1927	Germany	26435
1929	USA	2808000
1933	USA	2300000
1935	USA	3000000
1936	USA	163245

```sql
-- 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;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

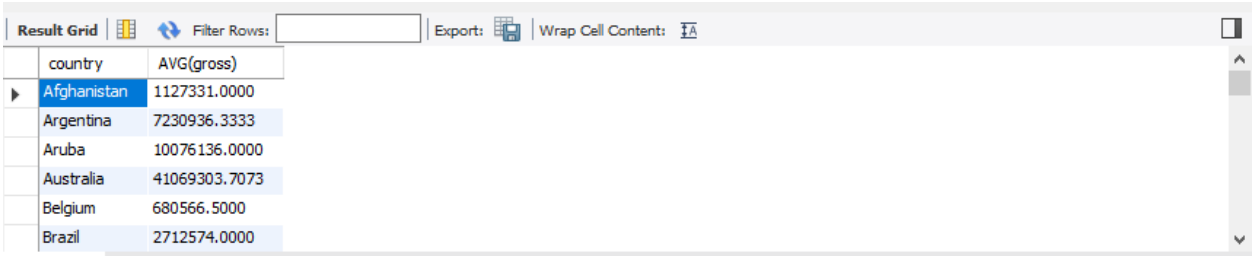
release_year	avg_budget	avg_box_office
2016	68674203	81757834
2015	54378008	73765712
2014	48420380	65481406
2013	49385475	61290381
2012	48030657	67916848
2011	42996023	49183796

```sql
-- 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;
```



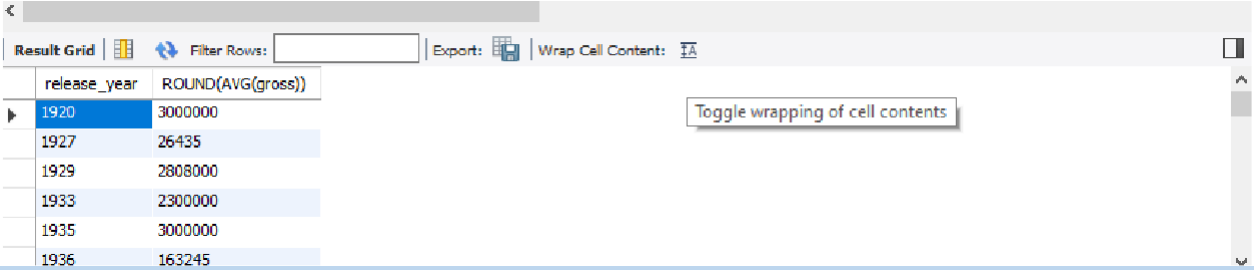
country	avg_budget	avg_box_office
Australia	36061110	41069304
Canada	21382656	24616829
China	82202133	17860601
France	34014099	18397186
Germany	35608537	29304670

```sql
-- Get the average amount made by each country
SELECT country, AVG(gross)
FROM films
GROUP BY country;
```



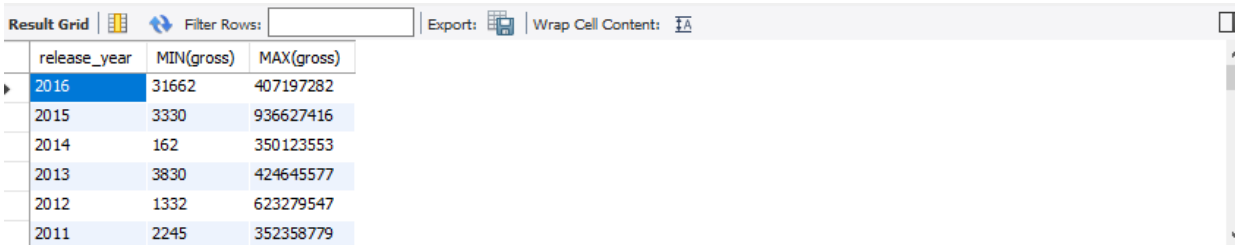
country	AVG(gross)
Afghanistan	1127331.0000
Argentina	7230936.3333
Aruba	10076136.0000
Australia	41069303.7073
Belgium	680566.5000
Brazil	2712574.0000

```sql
-- Get rounded average box office earnings per year
SELECT release_year, ROUND(AVG(gross))
FROM films
GROUP BY release_year
ORDER BY release_year;
```



release_year	ROUND(AVG(gross))
1920	3000000
1927	26435
1929	2808000
1933	2300000
1935	3000000
1936	163245

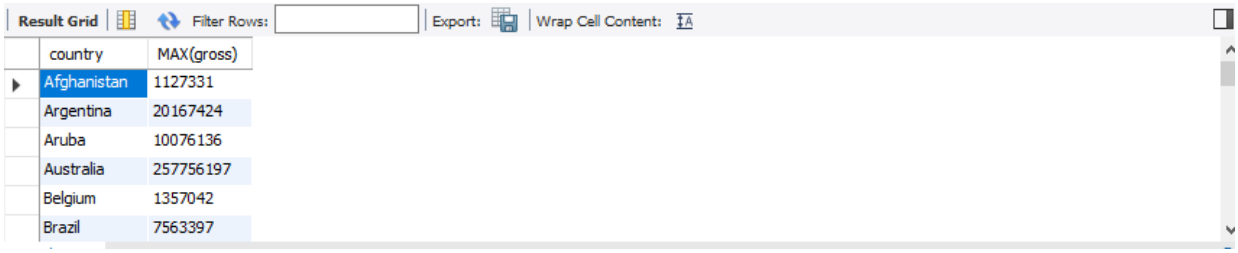
```sql
-- Get lowest and highest box office earnings per year
SELECT release_year, MIN(gross), MAX(gross)
FROM films
GROUP BY release_year
ORDER BY release_year DESC;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

release_year	MIN(gross)	MAX(gross)
2016	31662	407197282
2015	3330	936627416
2014	162	350123553
2013	3830	424645577
2012	1332	623279547
2011	2245	352358779

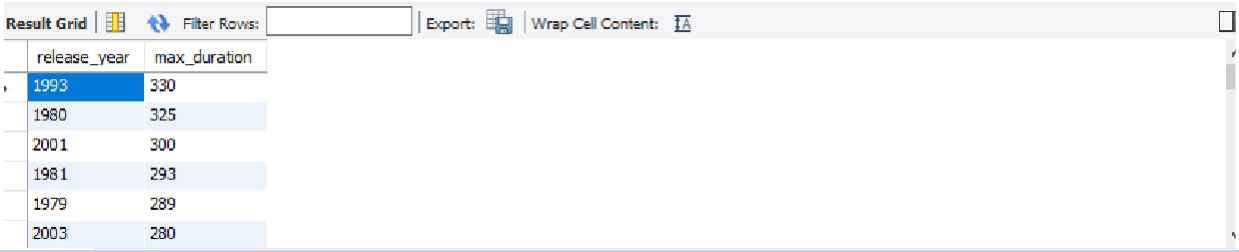
```sql
-- Get the highest box office take per country
SELECT country, MAX(gross)
FROM films
GROUP BY country;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

country	MAX(gross)
Afghanistan	1127331
Argentina	20167424
Aruba	10076136
Australia	257756197
Belgium	1357042
Brazil	7563397

```sql
-- Longest duration per year
SELECT release_year, MAX(duration) AS max_duration
FROM films
GROUP BY release_year
ORDER BY max_duration DESC;
```

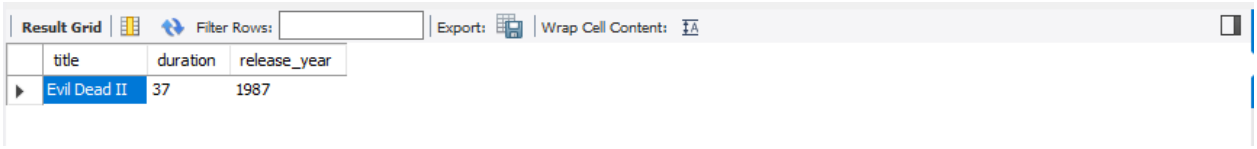


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

release_year	max_duration
1993	330
1980	325
2001	300
1981	293
1979	289
2003	280

```
-- Subqueries
```

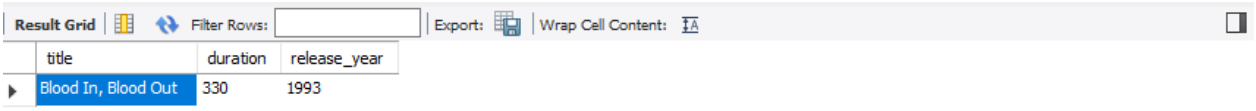
```
-- Get the title, duration and release year of the
shortest film(s)
SELECT title, duration, release_year
FROM films
WHERE duration = (
  SELECT MIN(duration) FROM films
);
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

title	duration	release_year
Evil Dead II	37	1987

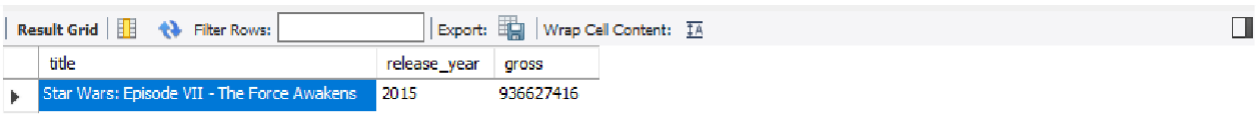
```
-- Get the title, duration and release year of the
longest film(s)
SELECT title, duration, release_year
FROM films
WHERE duration = (
  SELECT MAX(duration)
  FROM films
);
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

title	duration	release_year
Blood In, Blood Out	330	1993

```
-- 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
);
```



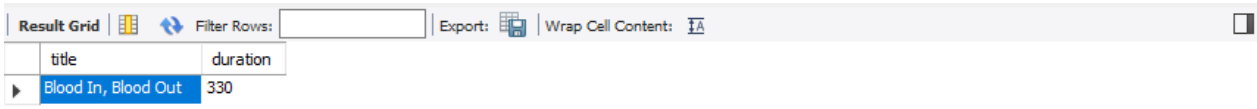
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

title	release_year	gross
Star Wars: Episode VII - The Force Awakens	2015	936627416

```sql
-- 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
);
```

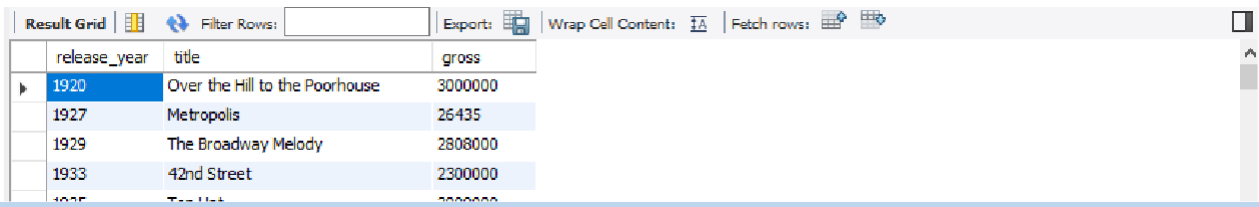
title	release_year	gross
Skin Trade	2014	162

```sql
-- 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'
);
```



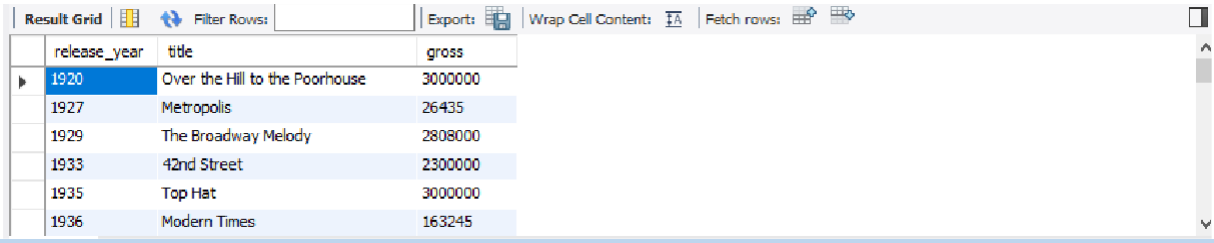
title	duration
Blood In, Blood Out	330

```sql
-- 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
  )
);
```



release_year	title	gross
1920	Over the Hill to the Poorhouse	3000000
1927	Metropolis	26435
1929	The Broadway Melody	2808000
1933	42nd Street	2300000

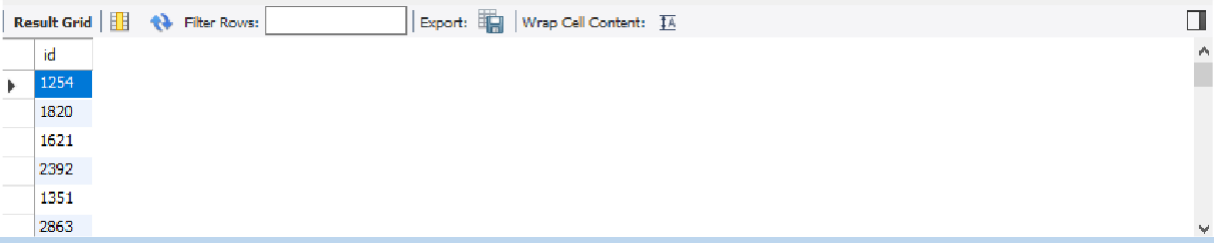
```sql
-- Get details for the film with the highest box
office earnings per year
SELECT release_year, title, gross
FROM films
WHERE release_year IN (
  SELECT release_year
  FROM films
  WHERE gross IN (
    SELECT MAX(gross)
    FROM films
    GROUP BY release_year
  )
);
```



release_year	title	gross
1920	Over the Hill to the Poorhouse	3000000
1927	Metropolis	26435
1929	The Broadway Melody	2808000
1933	42nd Street	2300000
1935	Top Hat	3000000
1936	Modern Times	163245

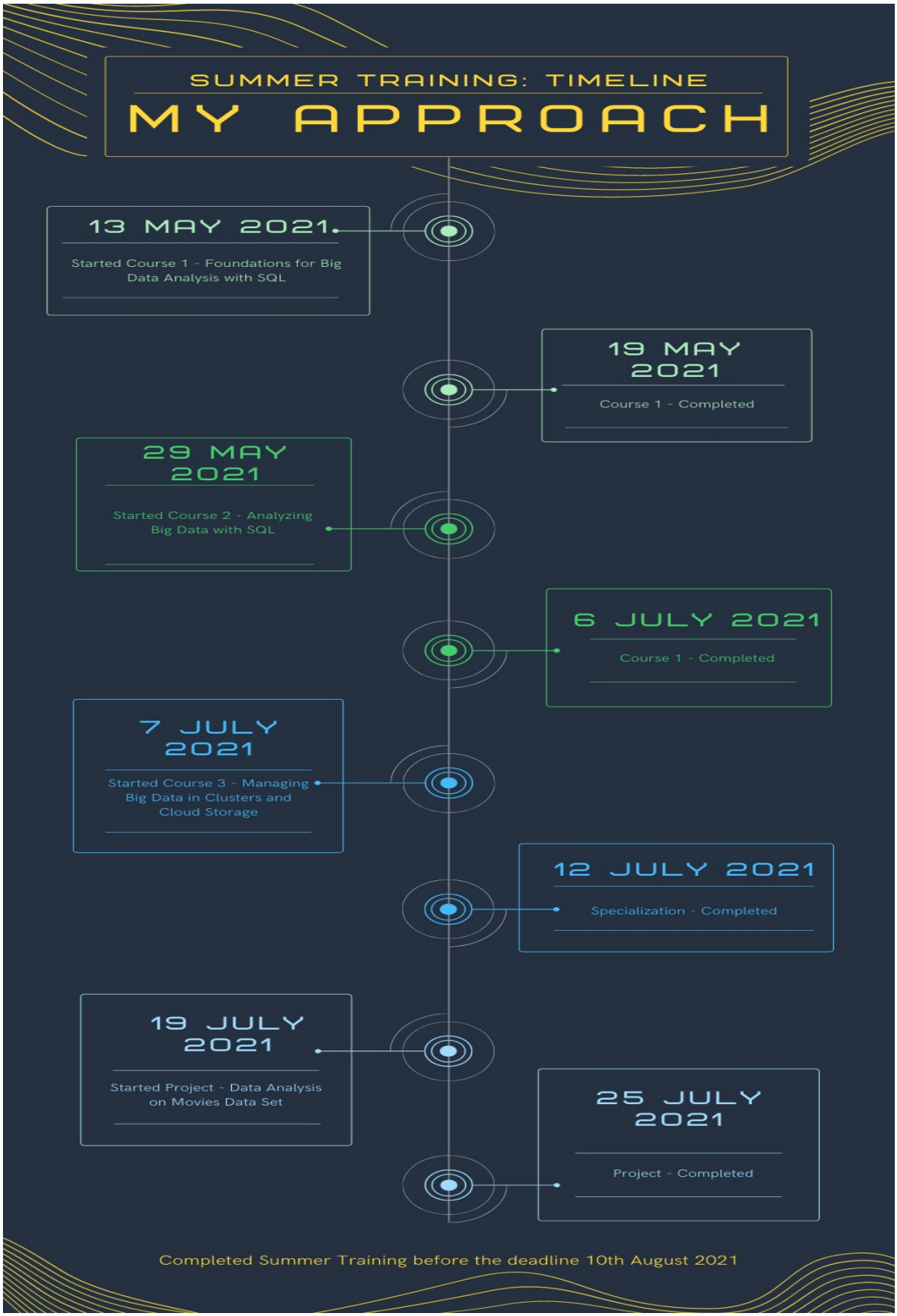
```sql
-- Joins
```

```sql
-- Get film id of films whose review exist
SELECT films.id
FROM films
INNER JOIN reviews
ON films.id = reviews.film_id;
```

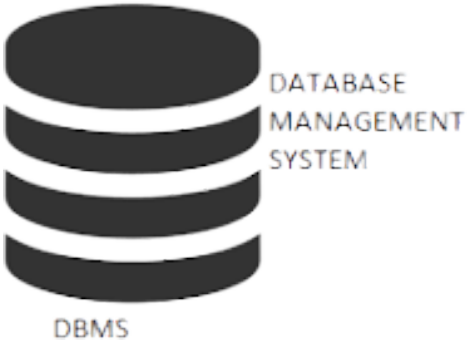


id
1254
1820
1621
2392
1351
2863

Chapter 4 – Timeline of Training and Project



Chapter 5 – Skills Gained from Training and Project



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.