SQL

1. **SELECT - Part 1:**

    - Specific columns: **SELECT coln_name1, coln_name2, .. FROM table_name;**
    - All columns: **SELECT * FROM table_name;**

2. **Exercises:**

Table: Movies

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93
4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107
6	The Incredibles	Brad Bird	2004	116
7	Cars	John Lasseter	2006	117
8	Ratatouille	Brad Bird	2007	115
9	WALL-E	Andrew Stanton	2008	104
10	Up	Pete Docter	2009	101
11	Toy Story 3	Lee Unkrich	2010	103
12	Cars 2	John Lasseter	2011	120
13	Brave	Brenda Chapman	2012	102
14	Monsters University	Dan Scanlon	2013	110

1. Find the title of each film. **SELECT Title FROM movies;**
2. Find the director of each film. **SELECT Director FROM movies;**
3. Find the title and director of each film. **SELECT Title, Director FROM movies;**
4. Find the title and year of each film. **SELECT Title, Year FROM movies;**
5. Find all the information about each film. **SELECT * FROM movies;**

3. SELECT - Part 2:

- WHERE: To filter certain results.
- **SELECT * FROM table_name WHERE condn;**
- Operator1: =, !=, >, >=, <, <= Ex: **coln_name != num_val;**
- Operator2: BETWEEN..AND.. - Number is within range of two values (inclusive) Ex: **coln_name BETWEEN value1 AND value2;**
- Operator3: NOT BETWEEN..AND.. - Number is not within range of two values (inclusive) Ex: **coln_name NOT BETWEEN value1 AND value2;**
- Operator4: IN - Number exists in a list. Ex: **coln_name IN (value1, value2, ...);**
- Operator5: NOT IN - Number not exists in a list. Ex: **coln_name NOT IN (value1, value2, ...);**

4. Exercises:

Table: Movies

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93
4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107
6	The Incredibles	Brad Bird	2004	116
7	Cars	John Lasseter	2006	117
8	Ratatouille	Brad Bird	2007	115
9	WALL-E	Andrew Stanton	2008	104
10	Up	Pete Docter	2009	101
11	Toy Story 3	Lee Unkrich	2010	103
12	Cars 2	John Lasseter	2011	120
13	Brave	Brenda Chapman	2012	102
14	Monsters University	Dan Scanlon	2013	110

1. Find the movie with a row id of 6. **SELECT * FROM movies WHERE Id = 6;**

2. Find the movies released in the years between 2000 and 2010. **SELECT * FROM movies WHERE Year BETWEEN 2000 AND 2010;**
3. Find the movies not released in the years between 2000 and 2010. **SELECT * FROM movies WHERE Year NOT BETWEEN 2000 AND 2010;**
4. Find the first 5 Pixar movies and their release year. **SELECT * FROM movies WHERE Id BETWEEN 1 AND 5;**

5. **SELECT - Part 3:**

    - Operator1: = Case sensitive exact string comparison Ex: **coln_name = “str_val”;**
    - Operator2: != Case sensitive exact string inequality comparison Ex: **coln_name != “str_val”;**
    - Operator3: LIKE - Case insensitive exact string comparison Ex: **coln_name LIKE “str_val”;**
    - Operator4: NOT LIKE - Case insensitive exact string inequality comparison Ex: **coln_name NOT LIKE “str_val”;**
    - Operator5: % - Used anywhere in a string to match a sequence of zero or more characters Ex: **coln_name LIKE “%str_val%”;**
    - Operator6: _ - Used anywhere in a string to match a single character Ex: **coln_name LIKE “strval_”;**
    - Operator7: IN - String exists in a list Ex: **coln_name IN (“str_value1”, “str_value2”, “str_value3”...);**
    - Operator8: NOT IN - String not exists in a list Ex: **coln_name NOT IN (“str_value1”, “str_value2”, “str_value3”...);**

6. Exercises:

Table: Movies

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93
4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107
6	The Incredibles	Brad Bird	2004	116
7	Cars	John Lasseter	2006	117
8	Ratatouille	Brad Bird	2007	115
9	WALL-E	Andrew Stanton	2008	104
10	Up	Pete Docter	2009	101
11	Toy Story 3	Lee Unkrich	2010	103
12	Cars 2	John Lasseter	2011	120
13	Brave	Brenda Chapman	2012	102
14	Monsters University	Dan Scanlon	2013	110
87	WALL-G	Brenda Chapman	2042	97

1. Find all the Toy Story movies. **SELECT * FROM movies WHERE Title LIKE "Toy Story%";**
2. Find all the movies directed by John Lasseter. **SELECT * FROM movies WHERE Director = "John Lasseter";**
3. Find all the movies (and director) not directed by John Lasseter. **SELECT * FROM movies WHERE Director != "John Lasseter";**
4. Find all the WALL-* movies. **SELECT * FROM movies WHERE Title LIKE "WALL-_";**

7. SELECT - Part 4:

- DISTINCT: Remove duplicate rows.
- ORDER BY: To sort the results by given column in ascending or descending order.
- **SELECT * FROM table_name WHERE condn ORDER BY coln_name ASC/DESC;**

- LIMIT: Reduce the number of rows to return.
- OFFSET: Specify where to begin counting the number of rows.
- **SELECT * FROM table_name WHERE condn ORDER BY coln_name ASC/DESC LIMIT limit_num OFFSET offset_num;**

8. Exercises:

Table: Movies

Id	Title	Director	Year	Length_minutes
1	Toy Story 2	John Lasseter	1999	93
2	Monsters University	Dan Scanlon	2013	110
3	Brave	Brenda Chapman	2012	102
4	Monsters, Inc.	Pete Docter	2001	92
5	Cars	John Lasseter	2006	117
6	A Bug's Life	John Lasseter	1998	95
7	Up	Pete Docter	2009	101
8	The Incredibles	Brad Bird	2004	116
9	Finding Nemo	Andrew Stanton	2003	107
10	WALL-E	Andrew Stanton	2008	104
11	Cars 2	John Lasseter	2011	120
12	Toy Story 3	Lee Unkrich	2010	103
13	Toy Story	John Lasseter	1995	81
14	Ratatouille	Brad Bird	2007	115

1. List all directors of Pixar movies (alphabetically), without duplicates.
   SELECT DISTINCT Director FROM movies ORDER BY Director;
2. List the last four Pixar movies released (ordered from most recent to least).
   SELECT * FROM movies ORDER BY Year DESC LIMIT 4;
3. List the first five Pixar movies sorted alphabetically.
   SELECT * FROM movies ORDER BY Title ASC LIMIT 5;
4. List the next five Pixar movies sorted alphabetically.
   SELECT * FROM movies ORDER BY Title ASC LIMIT 5 OFFSET 5;

9. Review Exercises:

Table: North_american_cities

City	Country	Population	Latitude	Longitude
Guadalajara	Mexico	1500800	20.659699	-103.349609
Toronto	Canada	2795060	43.653226	-79.383184
Houston	United States	2195914	29.760427	-95.369803
New York	United States	8405837	40.712784	-74.005941
Philadelphia	United States	1553165	39.952584	-75.165222
Havana	Cuba	2106146	23.05407	-82.345189
Mexico City	Mexico	8555500	19.432608	-99.133208
Phoenix	United States	1513367	33.448377	-112.074037
Los Angeles	United States	3884307	34.052234	-118.243685
Ecatepec de Morelos	Mexico	1742000	19.601841	-99.050674
Montreal	Canada	1717767	45.501689	-73.567256
Chicago	United States	2718782	41.878114	-87.629798

1. List all the Canadian cities and their populations.
   SELECT City, Country, Population FROM north_american_cities WHERE Country = "Canada";
2. Order all the cities in the United States by their latitude from north to south.
   SELECT City, Country, Latitude FROM north_american_cities WHERE Country = "United States" ORDER BY Latitude DESC;
3. List the two largest cities in Mexico (by population).
   SELECT City, Country, Population FROM north_american_cities WHERE Country = "Mexico" ORDER BY Population DESC LIMIT 2;
4. List the third and fourth largest cities (by population) in the United States and their population.
   SELECT City, Country, Population FROM north_american_cities WHERE Country = "United States" ORDER BY Population DESC LIMIT 2 OFFSET 2;

10. Multiple Table With Joins:

- JOIN: Combine row data across two separate tables using primary key.
- INNER JOIN: Matches rows from the first table and the second table which have the same key to create a result row with the combined columns from both tables.
- **SELECT * FROM table_name1 INNER JOIN table_name2 ON table_name1.id = table_name2.id WHERE condn ORDER BY coln_name LIMIT limit_val OFFSET offset_val;**
- INNER JOIN = JOIN

11. Exercises:

Table: Movies (Read-Only)

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93
4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107
6	The Incredibles	Brad Bird	2004	116
7	Cars	John Lasseter	2006	117
8	Ratatouille	Brad Bird	2007	115
9	WALL-E	Andrew Stanton	2008	104
10	Up	Pete Docter	2009	101
11	Toy Story 3	Lee Unkrich	2010	103
12	Cars 2	John Lasseter	2011	120
13	Brave	Brenda Chapman	2012	102
14	Monsters University	Dan Scanlon	2013	110

Table: Boxoffice (Read-Only)

Movie_id	Rating	Domestic_sales	International_sales
5	8.2	380843261	555900000
14	7.4	268492764	475066843
8	8	206445654	417277164
12	6.4	191452396	368400000
3	7.9	245852179	239163000
6	8	261441092	370001000
9	8.5	223808164	297503696
11	8.4	415004880	648167031
1	8.3	191796233	170162503
7	7.2	244082982	217900167
10	8.3	293004164	438338580
4	8.1	289916256	272900000
2	7.2	162798565	200600000
13	7.2	237283207	301700000

Id	Title	Director	Year	Length_minutes	Movie_id	Rating	Domestic_sales	International_sa
5	Finding Nemo	Andrew Stanton	2003	107	5	8.2	380843261	555900000
14	Monsters University	Dan Scanlon	2013	110	14	7.4	268492764	475066843
8	Ratatouille	Brad Bird	2007	115	8	8	206445654	417277164
12	Cars 2	John Lasseter	2011	120	12	6.4	191452396	368400000
3	Toy Story 2	John Lasseter	1999	93	3	7.9	245852179	239163000
6	The Incredibles	Brad Bird	2004	116	6	8	261441092	370001000
9	WALL-E	Andrew Stanton	2008	104	9	8.5	223808164	297503696
11	Toy Story 3	Lee Unkrich	2010	103	11	8.4	415004880	648167031
1	Toy Story	John Lasseter	1995	81	1	8.3	191796233	170162503
7	Cars	John Lasseter	2006	117	7	7.2	244082982	217900167
10	Up	Pete Docter	2009	101	10	8.3	293004164	438338580
4	Monsters, Inc.	Pete Docter	2001	92	4	8.1	289916256	272900000
2	A Bug's Life	John Lasseter	1998	95	2	7.2	162798565	200600000
13	Brave	Brenda Chapman	2012	102	13	7.2	237283207	301700000

1. Find the domestic and international sales for each movie.
   SELECT * FROM Movies INNER JOIN Boxoffice ON Movies.Id = Boxoffice.Movie_id;
2. Show the sales numbers for each movie that did better internationally rather than domestically.
   SELECT * FROM Movies INNER JOIN Boxoffice ON Movies.Id = Boxoffice.Movie_id WHERE Domestic_sales < International_sales;

3. List all the movies by their ratings in descending order.
   SELECT * FROM Movies INNER JOIN Boxoffice ON Movies.Id = Boxoffice.Movie_id ORDER BY Rating DESC;

12. Outer Joins:

- **SELECT * FROM table_name1 INNER/LEFT/RIGHT/FULL JOIN table_name2 ON table_name1.id = table_name2.id WHERE condn ORDER BY coln_name ASC/DESC LIMIT limit_num OFFSET offset_num;**
- LEFT JOIN: When joining table A to table B, a LEFT JOIN simply includes rows from A regardless of whether a matching row is found in B.
- RIGHT JOIN: Keeping rows in B regardless of whether a match is found in A.
- FULL JOIN: Rows from both tables are kept, regardless of whether a matching row exists in the other table.
- LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN = LEFT JOIN, RIGHT JOIN, FULL JOIN.
- OUTER keyword is really kept for SQL-92 compatibility.

13. Exercises:

Table: Buildings (Read-Only)

Building_name	Capacity
1e	24
1w	32
2e	16
2w	20

Table: Employees (Read-Only)

Role	Name	Building	Years_employed
Engineer	Becky A.	1e	4
Engineer	Dan B.	1e	2
Engineer	Sharon F.	1e	6
Engineer	Dan M.	1e	4
Engineer	Malcom S.	1e	1
Artist	Tylar S.	2w	2
Artist	Sherman D.	2w	8
Artist	Jakob J.	2w	6
Artist	Lillia A.	2w	7
Artist	Brandon J.	2w	7
Manager	Scott K.	1e	9
Manager	Shirlee M.	1e	3
Manager	Daria O.	2w	6

1. Find the list of all buildings that have employees.
   SELECT DISTINCT Building FROM Employees LEFT JOIN Buildings ON Employees.Building = Buildings.Building_name;
2. Find the list of all buildings and their capacity.
   SELECT * FROM Buildings LEFT JOIN Employees ON Buildings.Building_name = Employees.Building;
3. List all buildings and the distinct employee roles in each building (including empty buildings).
   SELECT DISTINCT building_name, role FROM buildings LEFT JOIN employees ON building_name = building;

14. NULLs:

- **SELECT * FROM table_name WHERE coln_name IS / IS NOT NULL;**

15. Exercises:

Table: Buildings (Read-Only)

Building_name	Capacity
1e	24
1w	32
2e	16
2w	20

Table: Employees (Read-Only)

Role	Name	Building	Years_employed
Engineer	Becky A.	1e	4
Engineer	Dan B.	1e	2
Engineer	Sharon F.	1e	6
Engineer	Dan M.	1e	4
Engineer	Malcom S.	1e	1
Artist	Tylar S.	2w	2
Artist	Sherman D.	2w	8
Artist	Jakob J.	2w	6
Artist	Lillia A.	2w	7
Artist	Brandon J.	2w	7
Manager	Scott K.	1e	9
Manager	Shirlee M.	1e	3
Manager	Daria O.	2w	6
Engineer	Yancy I.		0
Artist	Oliver P.		0

1. Find the name and role of all employees who have not been assigned to a building.
   SELECT Role, Name FROM employees WHERE Building IS NULL;
2. Find the names of the buildings that hold no employees.
   SELECT Building_name FROM Buildings LEFT JOIN Employees ON Buildings.Building_name = Employees.Building WHERE Name IS NULL;

16. Queries With Expressions:

- **SELECT column_expression AS expression_description FROM table_name;**

17. Exercises:

Table: Movies (Read-Only)

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93
4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107
6	The Incredibles	Brad Bird	2004	116
7	Cars	John Lasseter	2006	117
8	Ratatouille	Brad Bird	2007	115
9	WALL-E	Andrew Stanton	2008	104
10	Up	Pete Docter	2009	101
11	Toy Story 3	Lee Unkrich	2010	103
12	Cars 2	John Lasseter	2011	120
13	Brave	Brenda Chapman	2012	102
14	Monsters University	Dan Scanlon	2013	110

Table: Boxoffice (Read-Only)

Movie_id	Rating	Domestic_sales	International_sales
5	8.2	380843261	555900000
14	7.4	268492764	475066843
8	8	206445654	417277164
12	6.4	191452396	368400000
3	7.9	245852179	239163000
6	8	261441092	370001000
9	8.5	223808164	297503696
11	8.4	415004880	648167031
1	8.3	191796233	170162503
7	7.2	244082982	217900167
10	8.3	293004164	438338580
4	8.1	289916256	272900000
2	7.2	162798565	200600000
13	7.2	237283207	301700000

1. List all movies and their combined sales in millions of dollars.
   SELECT title, (domestic_sales + international_sales) / 1000000 AS total_sales_millions FROM movies INNER JOIN boxoffice ON movies.id = boxoffice.movie_id;
2. List all movies and their ratings in percent.
   SELECT title, (rating * 10) AS rating_percent FROM movies INNER JOIN boxoffice ON movies.id = boxoffice.movie_id;
3. List all movies that were released on even number years.
   SELECT * FROM Movies WHERE Year % 2 == 0;

18. Queries With Aggregates - Part 1:

- Aggregation Function Or Expression: Summarize information about a group of rows of data.
- **SELECT Aggregate_Func(Column_Expression) AS Aggregate_Description FROM table_name WHERE condn;**
- COUNT(*) : Counts the number of rows in the group if no column name is specified.
- COUNT(column_name) : Count the number of rows in the group with non-NULL values in the specified column.
- MIN(column_name) : Finds the smallest numerical value in the specified column for all rows in the group.
- MAX(column_name) : Finds the largest numerical value in the specified column for all rows in the group.
- AVG(column_name) : Finds the average numerical value in the specified column for all rows in the group.
- SUM(column_name) : Finds the sum of all numerical values in the specified column for the rows in the group.
- GROUP BY : Instead of aggregating across all the rows in a table, we can apply the aggregate functions to individual groups of data within that group.
- GROUP BY : Grouping rows that have the same value in the column specified.
- **SELECT Aggregate_Func(Column_Expression) AS Aggregate_Description FROM table_name WHERE condn GROUP BY coln_name;**

19. Exercises:

Table: Employees

Role	Name	Building	Years_employed
Engineer	Becky A.	1e	4
Engineer	Dan B.	1e	2
Engineer	Sharon F.	1e	6
Engineer	Dan M.	1e	4
Engineer	Malcom S.	1e	1
Artist	Tylar S.	2w	2
Artist	Sherman D.	2w	8
Artist	Jakob J.	2w	6
Artist	Lillia A.	2w	7
Artist	Brandon J.	2w	7
Manager	Scott K.	1e	9
Manager	Shirlee M.	1e	3
Manager	Daria O.	2w	6

1. Find the longest time that an employee has been at the studio.
   SELECT MAX(Years_employed) AS Senior_Employee, Role, Name FROM employees;
2. For each role, find the average number of years employed by employees in that role.
   SELECT Role, AVG(Years_employed) AS Avg_Years FROM Employees GROUP BY Role;
3. Find the total number of employee years worked in each building.
   SELECT Building, SUM(Years_employed) AS Sum_Years FROM Employees GROUP BY Building;

20. Queries With Aggregates - Part 2:

- HAVING : HAVING clause which is used specifically with the GROUP BY clause to allow us to filter grouped rows from the result set.
- **SELECT group_by_coln Aggregate_Func(Column_Expression) AS Aggregate_Description FROM table_name WHERE condn GROUP BY coln_name HAVING group_condn;**

21. Exercises:

Table: Employees

Role	Name	Building	Years_employed
Engineer	Becky A.	1e	4
Engineer	Dan B.	1e	2
Engineer	Sharon F.	1e	6
Engineer	Dan M.	1e	4
Engineer	Malcom S.	1e	1
Artist	Tylar S.	2w	2
Artist	Sherman D.	2w	8
Artist	Jakob J.	2w	6
Artist	Lillia A.	2w	7
Artist	Brandon J.	2w	7
Manager	Scott K.	1e	9
Manager	Shirlee M.	1e	3
Manager	Daria O.	2w	6

1. Find the number of Artists in the studio (without a HAVING clause).
   SELECT COUNT(*) AS No_Of_Artists FROM employees WHERE Role == "Artist";
2. Find the number of Employees of each role in the studio.
   SELECT Role, COUNT(Role) AS No_Of_Employees FROM employees GROUP BY Role;
3. Find the total number of years employed by all Engineers.
   SELECT SUM(Years_employed) AS Total_Years_Engineer FROM employees GROUP BY Role HAVING Role = "Engineer";

22. Order Of Execution Of A Query :

- **SELECT DISTINCT column, AGG_FUNC(column_or_expression) FROM table_name1 JOIN table_name2 ON table_name1.column = table_name2.column WHERE condn GROUP BY column HAVING condn ORDER BY column ASC/DESC LIMIT limit_val OFFSET offset_val;**

23. Exercises:

Table: Movies (Read-Only)

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93
4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107
6	The Incredibles	Brad Bird	2004	116
7	Cars	John Lasseter	2006	117
8	Ratatouille	Brad Bird	2007	115
9	WALL-E	Andrew Stanton	2008	104
10	Up	Pete Docter	2009	101
11	Toy Story 3	Lee Unkrich	2010	103
12	Cars 2	John Lasseter	2011	120
13	Brave	Brenda Chapman	2012	102
14	Monsters University	Dan Scanlon	2013	110

Table: Boxoffice (Read-Only)

Movie_id	Rating	Domestic_sales	International_sales
5	8.2	380843261	555900000
14	7.4	268492764	475066843
8	8	206445654	417277164
12	6.4	191452396	368400000
3	7.9	245852179	239163000
6	8	261441092	370001000
9	8.5	223808164	297503696
11	8.4	415004880	648167031
1	8.3	191796233	170162503
7	7.2	244082982	217900167
10	8.3	293004164	438338580
4	8.1	289916256	272900000
2	7.2	162798565	200600000
13	7.2	237283207	301700000

1. Find the number of movies each director has directed.
   SELECT COUNT(Title) AS No_Of_Movies, Director FROM movies GROUP BY Director;
2. Find the total domestic and international sales that can be attributed to each director.
   SELECT Director, SUM(Domestic_sales + International_sales) AS Total_sales FROM Movies INNER JOIN Boxoffice ON Movies.Id = Boxoffice.Movie_id GROUP BY Director

24. Inserting Rows:

- Insert Statement With Values For All Columns:
  INSERT INTO table_name VALUES (value_or_expr1, another_value_or_expr1, …), …(value_or_expr2, another_value_or_expr2, …);
- Insert Statement With Values For Specific Columns:
  INSERT INTO table_name (coln_name1, coln_name2 …) VALUES (value_or_expr1, another_value_or_expr1, …), …(value_or_expr2, another_value_or_expr2, …);

25. Exercises:

Table: Movies (Read-Only)

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93

Table: Boxoffice (Read-Only)

Movie_id	Rating	Domestic_sales	International_sales
3	7.9	245852179	239163000
1	8.3	191796233	170162503
2	7.2	162798565	200600000

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93
4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107
6	The Incredibles	Brad Bird	2004	116
7	Cars	John Lasseter	2006	117
8	Ratatouille	Brad Bird	2007	115
9	WALL-E	Andrew Stanton	2008	104
10	Up	Pete Docter	2009	101
11	Toy Story 3	Lee Unkrich	2010	103
12	Cars 2	John Lasseter	2011	120
13	Brave	Brenda Chapman	2012	102
14	Monsters University	Dan Scanlon	2013	110

1. Add the studio's new production, Toy Story 4 to the list of movies (you can use any director).
   INSERT INTO Movies VALUES(15, "Toy Story 4", "John Lasseter", 2024, 120);
2. Toy Story 4 has been released to critical acclaim! It had a rating of 8.7, and made 340 million domestically and 270 million internationally. Add the record to the BoxOffice table.
   INSERT INTO Boxoffice VALUES(15, 8.7, 340, 270);

26. Updating Rows:

- **UPDATE table_name SET coln_name = value_or_exp, other_coln_name = value_or_exp … WHERE cond;**

27. Exercises:

Table: Movies

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	El Directore	1998	95
3	Toy Story 2	John Lasseter	1899	93
4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107
6	The Incredibles	Brad Bird	2004	116
7	Cars	John Lasseter	2006	117
8	Ratatouille	Brad Bird	2007	115
9	WALL-E	Andrew Stanton	2008	104
10	Up	Pete Docter	2009	101
11	Toy Story 8	El Directore	2010	103
12	Cars 2	John Lasseter	2011	120
13	Brave	Brenda Chapman	2012	102
14	Monsters University	Dan Scanlon	2013	110

1. The director for A Bug's Life is incorrect, it was actually directed by John Lasseter.
   UPDATE Movies SET Director = "John Lasseter" WHERE Title = "A Bug's Life";
2. The year that Toy Story 2 was released is incorrect, it was actually released in 1999.
   UPDATE Movies SET Year = 1999 WHERE Title = "Toy Story 2";
3. Both the title and director for Toy Story 8 is incorrect! The title should be "Toy Story 3" and it was directed by Lee Unkrich.
   UPDATE Movies SET Title = "Toy Story 3", Director = "Lee Unkrich" WHERE Title = "Toy Story 8";

28. Deleting Rows:

- **DELETE FROM table_name WHERE condn;**

29. Exercises:

Table: Movies

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93
4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107
6	The Incredibles	Brad Bird	2004	116
7	Cars	John Lasseter	2006	117
8	Ratatouille	Brad Bird	2007	115
9	WALL-E	Andrew Stanton	2008	104
10	Up	Pete Docter	2009	101
11	Toy Story 3	Lee Unkrich	2010	103
12	Cars 2	John Lasseter	2011	120
13	Brave	Brenda Chapman	2012	102
14	Monsters University	Dan Scanlon	2013	110

1. This database is getting too big, lets remove all movies that were released before 2005.
   DELETE FROM Movies WHERE Year<2005;
2. Andrew Stanton has also left the studio, so please remove all movies directed by him.
   DELETE FROM Movies WHERE Director = "Andrew Stanton";

30. Altering Tables:

- ALTER TABLE: To add, remove, or modify columns and table constraints.
- Adding Columns: **ALTER TABLE table_name ADD coln_name data_type DEFAULT default_value;**
- Removing Columns: **ALTER TABLE table_name DROP coln_name_to_be_deleted;**
- Renaming Table: **ALTER TABLE table_name RENAME TO new_table_name;**

31. Exercises:

Table: Movies

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93
4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107
6	The Incredibles	Brad Bird	2004	116
7	Cars	John Lasseter	2006	117
8	Ratatouille	Brad Bird	2007	115
9	WALL-E	Andrew Stanton	2008	104
10	Up	Pete Docter	2009	101
11	Toy Story 3	Lee Unkrich	2010	103
12	Cars 2	John Lasseter	2011	120
13	Brave	Brenda Chapman	2012	102
14	Monsters University	Dan Scanlon	2013	110

1. Add a column named Aspect_ratio with a FLOAT data type to store the aspect-ratio each movie was released in.
   ALTER TABLE Movies ADD Aspect_ratio FLOAT DEFAULT 6;
2. Add another column named Language with a TEXT data type to store the language that the movie was released in. Ensure that the default for this language is English.
   ALTER TABLE Movies ADD Language TEXT DEFAULT "English";

32. Dropping Tables:

- **DROP TABLE IF EXISTS table_name;**

33. Exercises:

Table: Movies (Read-Only)

Id	Title	Director	Year	Length_minutes
1	Toy Story	John Lasseter	1995	81
2	A Bug's Life	John Lasseter	1998	95
3	Toy Story 2	John Lasseter	1999	93
4	Monsters, Inc.	Pete Docter	2001	92
5	Finding Nemo	Andrew Stanton	2003	107
6	The Incredibles	Brad Bird	2004	116
7	Cars	John Lasseter	2006	117
8	Ratatouille	Brad Bird	2007	115
9	WALL-E	Andrew Stanton	2008	104
10	Up	Pete Docter	2009	101
11	Toy Story 3	Lee Unkrich	2010	103
12	Cars 2	John Lasseter	2011	120
13	Brave	Brenda Chapman	2012	102
14	Monsters University	Dan Scanlon	2013	110

Table: Boxoffice (Read-Only)

Movie_id	Rating	Domestic_sales	International_sales
5	8.2	380843261	555900000
14	7.4	268492764	475066843
8	8	206445654	417277164
12	6.4	191452396	368400000
3	7.9	245852179	239163000
6	8	261441092	370001000
9	8.5	223808164	297503696
11	8.4	415004880	648167031
1	8.3	191796233	170162503
7	7.2	244082982	217900167
10	8.3	293004164	438338580
4	8.1	289916256	272900000
2	7.2	162798565	200600000
13	7.2	237283207	301700000

1. We've sadly reached the end of our lessons, lets clean up by removing the Movies table.
   DROP TABLE IF EXISTS Movies;
2. And drop the BoxOffice table as well.
   DROP TABLE IF EXISTS BoxOffice;

34. Creating Tables:

- **CREATE TABLE IF NOT EXISTS table_name(coln_name1 data_type, coln_name2 data_type…);**
- Datatypes: INTEGER, BOOLEAN, FLOAT, DOUBLE, REAL, VARCHAR, TEXT, DATE, DATETIME, BLOB (Binary data).
- Table Constraints: PRIMARY KEY, UNIQUE, NOT NULL, FOREIGN KEY.
- PRIMARY KEY: Values in this column are unique, and each value can be used to identify a single row in this table.
- UNIQUE: This means that the values in this column have to be unique, so you can't insert another row with the same value in this column as another row in the table. Differs from the `PRIMARY KEY` in that it doesn't have to be a key for a row in the table.
- FOREIGN KEY: Ensures that each value in this column corresponds to another value in a column in another table.

35. Exercises:

- Create a new table named Database with the following columns: Name A string (text) describing the name of the database ; Version A number (floating point) of the latest version of this database ; Download_count An integer count of the number of times this database was downloaded. This table has no constraints.
- **CREATE TABLE Database(Name TEXT, Version FLOAT, Download_count INTEGER);**
  SELECT * FROM Database;