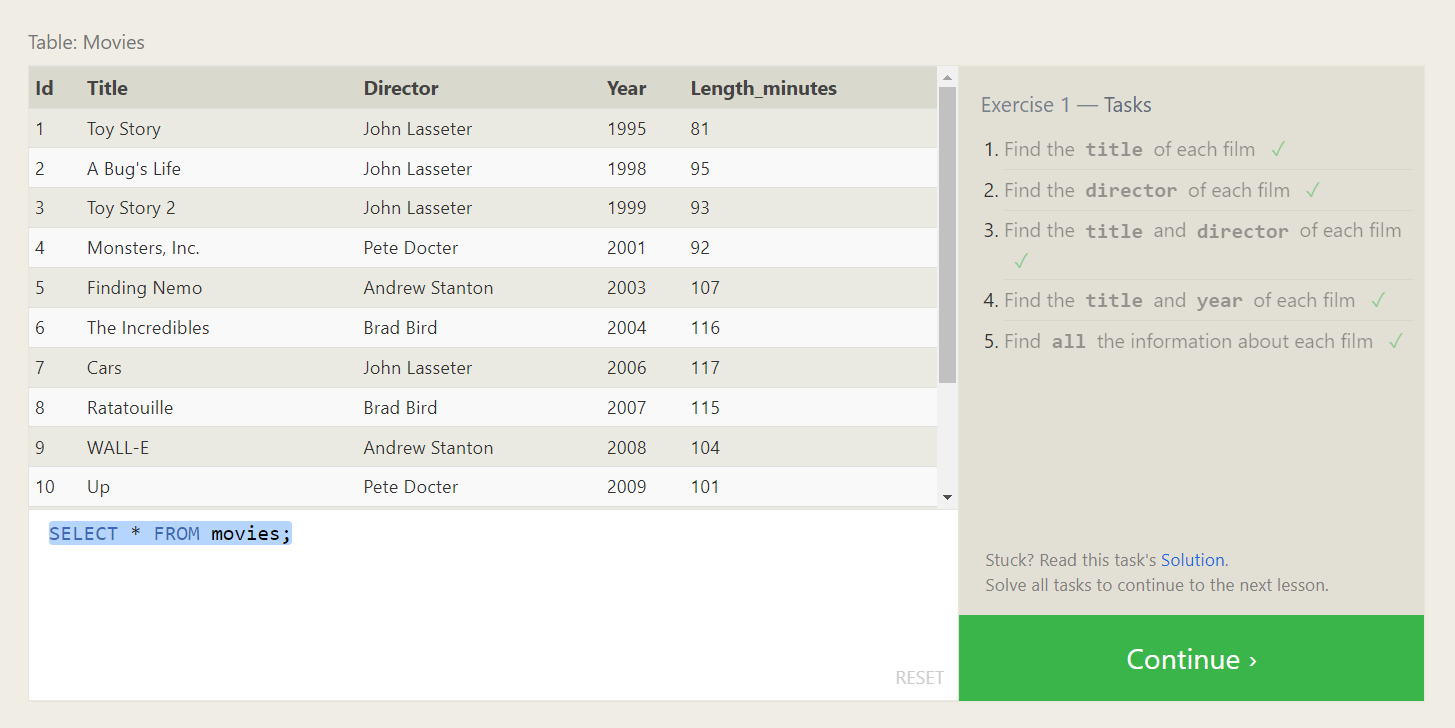
**QUERIES**

**SQL Lesson 1: SELECT queries 101**

1. SELECT title FROM movies;
2. SELECT director FROM movies;
3. SELECT title,director FROM movies;
4. SELECT title,year FROM movies;
5. SELECT \* FROM movies;

**Solution:**



**SQL Lesson 2: Queries with constraints**

1. SELECT \* FROM movies where id=6;
2. SELECT \* FROM movies where year between 2000 and 2010;
3. SELECT \* FROM movies where year not between 2000 and 2010;
4. SELECT \* FROM movies limit 5;
5. **Solution:**

A screenshot of a computer

Description automatically generated

**SQL Lesson 3: Queries with constraints (Pt. 2)**

1. SELECT \* FROM movies where title like '%toy story%';
2. SELECT \* FROM movies where director='John Lasseter';
3. SELECT title,director FROM movies where not director='John Lasseter';
4. SELECT \* FROM movies where title like "WALL-%";

**Solution:**

A screenshot of a computer

Description automatically generated

**SQL Lesson 4: Filtering and sorting Query results**

1. SELECT DISTINCT director FROM movies ORDER BY director;
2. SELECT DISTINCT title FROM movies ORDER BY year DESC LIMIT 4;
3. SELECT title FROM movies ORDER BY title LIMIT 5;
4. SELECT title FROM movies ORDER BY title LIMIT 5 OFFSET 5;
5. **Solution:**

A screenshot of a computer

Description automatically generated

**SQL Review: Simple SELECT Queries**

1. SELECT city, population FROM north\_american\_cities WHERE country = "Canada";
2. SELECT city FROM north\_american\_cities WHERE country = "United States"ORDER BY latitude DESC;
3. SELECT city FROM north\_american\_cities WHERE longitude < -87.629798 ORDER BY longitude;
4. SELECT city FROM north\_american\_cities WHERE country = "Mexico" ORDER BY population DESC LIMIT 2;
5. SELECT city FROM north\_american\_cities WHERE country = "United States" ORDER BY population DESC LIMIT 2 OFFSET 2;

**Solution:**

A screenshot of a computer

Description automatically generated

**SQL Lesson 6: Multi-table queries with JOINs**

1. SELECT title, domestic\_sales, international\_sales FROM movies INNER JOIN boxoffice ON movies.id = boxoffice.movie\_id;

2. SELECT title, domestic\_sales, international\_sales FROM movies INNER JOIN boxofficeON movies.id = boxoffice.movie\_id WHERE international\_sales > domestic\_sales;

3. SELECT title, rating FROM movies INNER JOIN boxoffice ON movies.id = boxoffice.movie\_id ORDER BY rating DESC;

**Solution:**

A screenshot of a computer

Description automatically generated

**SQL Lesson 7: OUTER JOINs**

1. SELECT DISTINCT building FROM employees;
2. SELECT \* FROM buildings;
3. SELECT DISTINCT building\_name, role

FROM buildings LEFT JOIN employees ON building\_name = employees.building;

**Solution:**

A screenshot of a computer

Description automatically generated

**SQL Lesson 8: A short note on NULLs**

1. SELECT name, role FROM employees WHERE building IS NULL;
2. SELECT DISTINCT building\_name FROM buildings LEFT JOIN employees ON building\_name = employees.building

WHERE employees.building IS NULL;

**Solution:**

A screenshot of a computer

Description automatically generated

**SQL Lesson 9: Queries with expressions**

1. SELECT DISTINCT

title,

(domestic\_sales + international\_sales) / 1000000 AS sales

FROM movies

INNER JOIN boxoffice

ON movies.id = boxoffice.movie\_id;

2. SELECT DISTINCT

title,

(rating \* 10) AS rate\_percent

FROM movies

INNER JOIN boxoffice

ON movies.id = boxoffice.movie\_id;

1. SELECT title FROM movies WHERE year % 2 = 0;

**Solution:**

A screenshot of a computer

Description automatically generated

**SQL Lesson 10: Queries with aggregates (Pt. 1)**

1. SELECT name, MAX(years\_employed) FROM employees;
2. SELECT role, AVG(years\_employed) as Average\_years\_employed FROM employees GROUP BY role;
3. SELECT building, SUM(years\_employed) FROM employees GROUP BY building;

**Solution:**

A screenshot of a computer

Description automatically generated

**SQL Lesson 11: Queries with aggregates (Pt. 2)**

1. SELECT COUNT(\*) FROM employees WHERE role LIKE 'artist';
2. SELECT role, COUNT(name) FROM employees GROUP BY role;
3. SELECT role, SUM(years\_employed) FROM employees GROUP BY role HAVING role LIKE 'engineer';

**Solution:**

A screenshot of a computer

Description automatically generated

**SQL Lesson 12: Order of execution of a Query**

1.SELECT director, COUNT(\*) FROM movies GROUP BY director;

2. SELECT director, SUM(domestic\_sales) + SUM(international\_sales) AS Total FROM movies LEFT JOIN boxoffice ON movies.id = boxoffice.movie\_id GROUP BY director;

**Solution:**

A screenshot of a computer

Description automatically generated

**SQL Lesson 13: Inserting rows**

1. INSERT INTO movies (title, director, year, length\_minutes)

VALUES ('Toy Story 4', 'John Lasseter', 2019, 123);

2. INSERT INTO boxoffice (movie\_id, rating, domestic\_sales, international\_sales)

VALUES (15, 8.7, 340000000, 270000000);

**Solution:**

A screenshot of a computer

Description automatically generated

**SQL Lesson 14: Updating rows**

1. Update Movies set Director = "John Lasseter" where Title = "A Bug's Life";
2. Update Movies set Year = 1999 where Title = "Toy Story 2";
3. Update Movies set Title = "Toy Story 3", Director = "Lee Unkrich"
4. where Title = "Toy Story 8";

**Solution:**

A screenshot of a computer

Description automatically generated

**SQL Lesson 15: Deleting rows**

1. DELETE FROM movies WHERE year < 2005
2. DELETE FROM movies WHERE director = 'Andrew Stanton'

**Solution:**

A screenshot of a computer

Description automatically generated

**SQL Lesson 16: Creating tables**

1. CREATE TABLE database (

name TEXT,

version INTEGER,

download\_count INTEGER

)

**Solution:**

A screenshot of a computer

Description automatically generated

**SQL Lesson 17: Altering tables**

1. ALTER TABLE movies ADD aspect\_ratio FLOAT
2. ALTER TABLE movies ADD language TEXT DEFAULT 'English'

**Solution:**

A screenshot of a computer

Description automatically generated

**SQL Lesson 18: Dropping tables**

1.Drop table movies;

2.Drop table BoxOffice;

**Solution:**

A screenshot of a computer

Description automatically generated