Learn SQL

Queries

Print cheatsheet

`SELECT` Statement

The `SELECT *` statement returns all columns from the provided table in the result set. The given query will fetch all columns and records (rows) from the `movies` table.

```
SELECT *
FROM movies;
```

`AS` Clause

Columns or tables in SQL can be *aliased* using the `AS` clause. This allows columns or tables to be specifically renamed in the returned result set. The given query will return a result set with the column for `name` renamed to `movie_title`.

```
SELECT name AS 'movie_title'
FROM movies;
```

`DISTINCT` Clause

Unique values of a column can be selected using a `DISTINCT` query. For a table `contact_details` having five rows in which the `city` column contains Chicago, Madison, Boston, Madison, and Denver, the given query would return:

- `Chicago`
- `Madison`
- `Boston`
- `Denver`

```
SELECT DISTINCT city
FROM contact_details;
```

`WHERE` Clause

The `WHERE` clause is used to filter records (rows) that match a certain condition. The given query will select all records where the `pub_year` equals `2017`.

```
SELECT title
FROM library
WHERE pub_year = 2017;
```

`LIKE` Operator

The `LIKE` operator can be used inside of a `WHERE` clause to match a specified pattern. The given query will match any movie that begins with `Star` in its title.

```
SELECT name
FROM movies
WHERE name LIKE 'Star%';
```

`_` Wildcard

The `_` wildcard can be used in a `LIKE` operator pattern to match any single

unspecified character. The given query will match any movie which begins with a single character, followed by `ove`.

```
SELECT name
FROM movies
WHERE name LIKE '_ove';
```

`%` Wildcard

The `%` wildcard can be used in a `LIKE` operator pattern to match zero or more unspecified character(s). The example query will match any movie that begins with `The`, followed by zero or more of any characters.

```
SELECT name
FROM movies
WHERE name LIKE 'The%';
```

`NULL` Values

Column values in SQL records can be `NULL`, or have no value. These records can be matched (or not matched) using the `IS NULL` and `IS NOT NULL` operators in combination with the `WHERE` clause. The given query will match all addresses where the address has a value or is not `NULL`.

```
SELECT address
FROM records
WHERE address IS NOT NULL;
```

`BETWEEN` Operator

The `BETWEEN` operator can be used to filter by a *range* of values. The range of

values can be text, numbers or date data. The given query will match any movie made between the years 1980 and 1990, inclusive.

```
SELECT *
FROM movies
WHERE year BETWEEN 1980 AND 1990;
```

`AND` Operator

The `AND` operator allows multiple conditions to be combined. Records must match both conditions that are joined by `AND` to be included in the result set. The example query will match any car that is blue and made after 2014.

```
SELECT model
FROM cars
WHERE color = 'blue'
  AND year > 2014;
```

`OR` Operator

The `OR` operator allows multiple conditions to be combined. Records matching either condition joined by the `OR` are included in the result set. The given query will match customers whose state is either `ca` or `ny`.

```
SELECT name
FROM customers
WHERE state = "ca"
   OR state = "ny";
```

`ORDER BY` Clause

The `ORDER BY` clause can be used to sort the result set by a particular column either alphabetically or numerically. It can be ordered in ascending (default) or descending order with `ASC` / `DESC`. In the example, all the rows of the `contacts` table will be ordered by the `birth_date` column in descending order.

```
SELECT *
FROM contacts
ORDER BY birth_date DESC;
```

`LIMIT` Clause

The `LIMIT` clause is used to narrow, or *limit*, a result set to the specified number of rows. The given query will limit the result set to 5 rows.

```
SELECT *
FROM movies
LIMIT 5;
```

Related Courses

Course

Learn SQL

Learn to communicate with databases using SQL, the standard