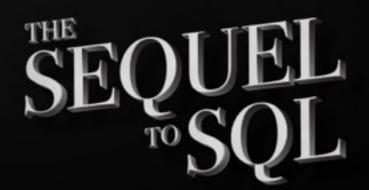
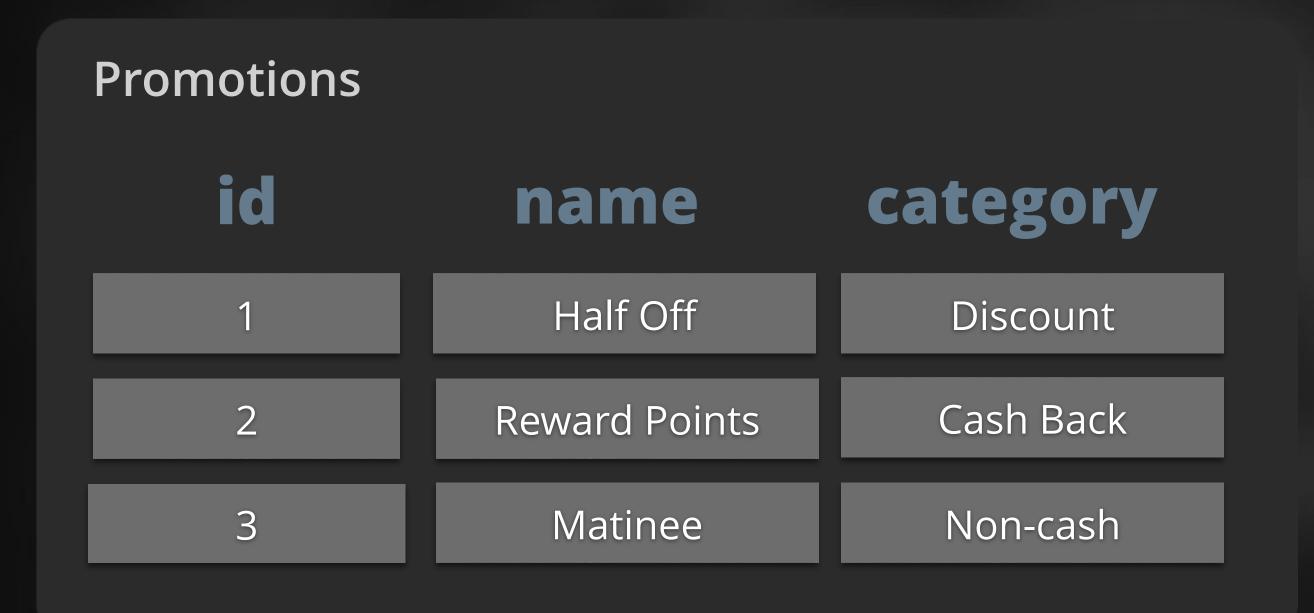
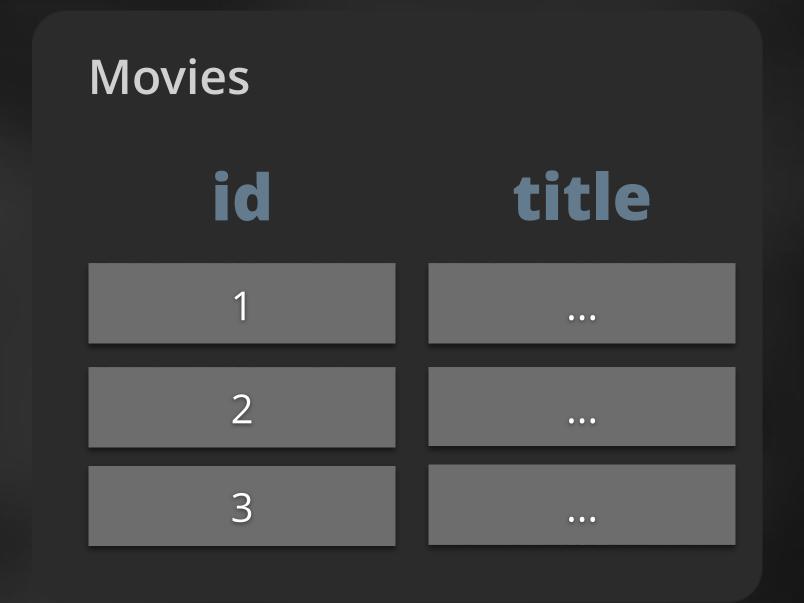
The Sequel to SQL Level 2 – Section 2 Value Constraints



Creating Valid References Between 2 Tables

How might we start associating particular promotions with specific movies?





We could store the movie title inside the promotions table, but then we'd be repeating ourselves.

Referencing the Movies Primary Key

Promotions

id movie_id

The movie_id column references the id column in the Movies table.

id	movie_id	name	category
1	1	Half Off	Discount
2	2	Reward Points	Cash Back
3	3	Matinee	Non-cash



Common naming convention

movie_id

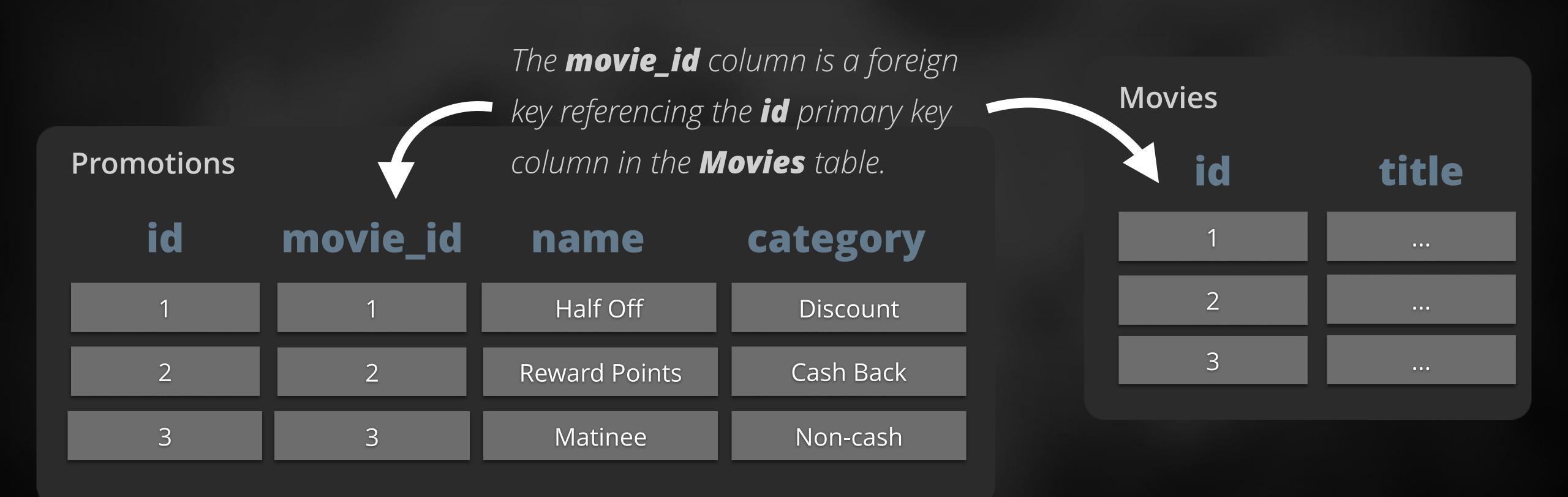
An underscore followed by the column name

Singular version of the table you're referencing

Introducing the Foreign Key

movie_id is a foreign key.

A foreign key is a column in 1 table that references the primary key column of another table.



Querying for Relationship Data

How would we find the promotions for the movie Gone With the Wind?

SELECT id
FROM Movies
WHERE title = 'Gone With the Wind';

SELECT name, category
FROM Promotions
WHERE movie_id = 2;



Returns the value 2

Promotions

id	movie_id	name	category
1	1	Half Off	Discount
2	2	Reward Points	Cash Back
3	3	Matinee	Non-cash

Movies

id	title
1	•••
2	Gone With the Wind
3	•••

Querying for Relationship Data

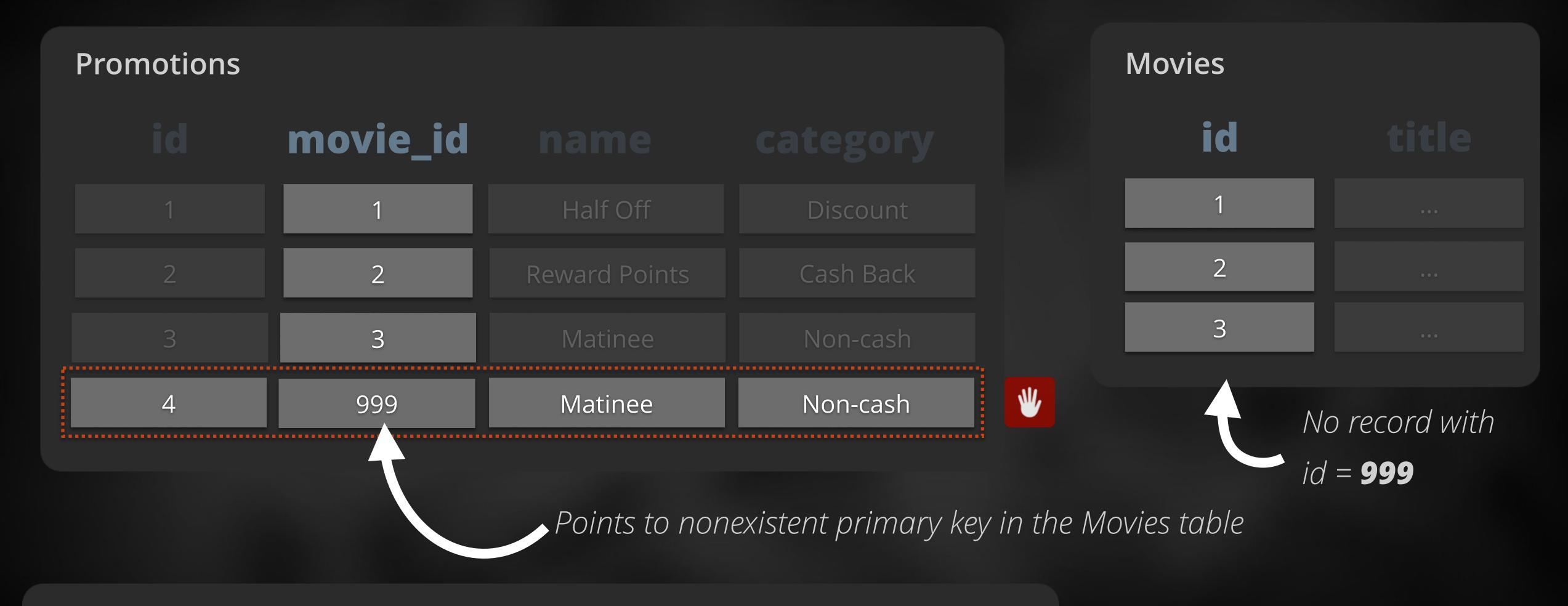
Promotions

id	movie_id	name	category
1	1	Half Off	Discount
2	2	Reward Points	Cash Back
3	3	Matinee	Non-cash

Movies

id	title	
1	•••	
2	Gone With the Wind	
3	•••	

Inserting Invalid Data for movie_id



INSERT INTO Promotions (id, movie_id, name, category)
VALUES (4, 999, 'Fake Promotion', 'Hoax');

Creating a FOREIGN KEY Constraint

The REFERENCES keyword can be used to make a FOREIGN KEY constraint.

业

```
CREATE TABLE Movies
(
  id int PRIMARY KEY,
  title varchar(20) NOT NULL UNIQUE
);
```

The table being referenced must be created **first**.

Notice we've added a primary key!

```
CREATE TABLE Promotions
(
   id int PRIMARY KEY,
   movie_id int,
   name varchar(50),
   category varchar(15)
);
```

Adding constraint

CREATE TABLE Promotions
(
 id int PRIMARY KEY,
 movie_id int REFERENCES movies(id),
 name varchar(50),
 category varchar(15)
);

Preventing Inconsistent Relationships

The foreign key in the second table **must match** a primary key in the table being referenced.

Promotionsidmovie_idnamecategory11Half OffDiscount22Reward PointsCash Back33MatineeNon-cash

INSERT INTO Promotions (id, movie_id, name, category)
VALUES (4, 999, 'Fake Promotion', 'Hoax');



The FOREIGN KEY constraint will generate errors upon invalid data inserts.

ERROR: insert or update on table "promotions" violates foreign key constraint
"promotions_movie_id_fkey"

DETAIL: Key (movie_id)=(999) is not present in table "movies".



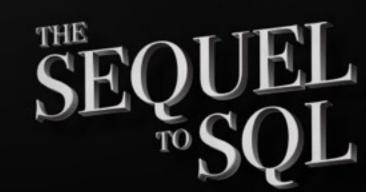
Using a Shorter FOREIGN KEY Constraint Syntax

```
CREATE TABLE Promotions
(
   id int PRIMARY KEY,
   movie_id int REFERENCES movies(id),
   name varchar(50),
   category varchar(15),
);
```

```
CREATE TABLE Promotions
(
  id int PRIMARY KEY,
  movie_id int REFERENCES movies ,
  name varchar(50),
  category varchar(15),
);
```

Same thing

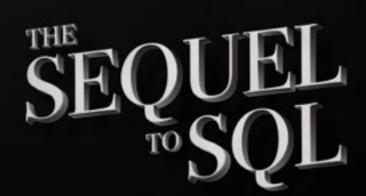
In absence of a column, the **primary key** of the referenced table is used.



Using Table Constraint Syntax

```
CREATE TABLE Promotions
  id int PRIMARY KEY,
  movie_id int REFERENCES movies
  name varchar(50),
  category varchar(15),
);
CREATE TABLE Promotions
  id int PRIMARY KEY,
  movie_id int,
  name varchar(50),
  category varchar(15),
  FOREIGN KEY (movie_id) REFERENCES movies
);
```

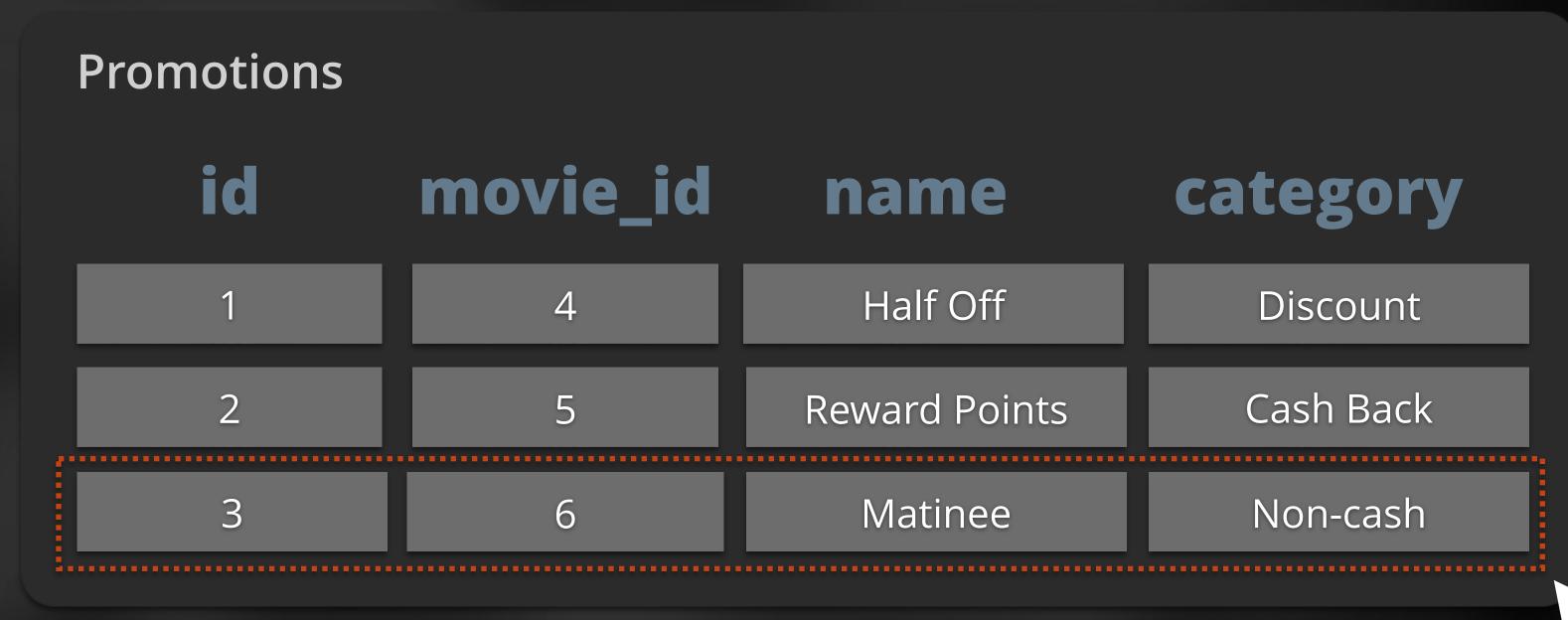
Same behavior



Orphan Records

Orphan records are child records with a foreign key to a parent record that has been deleted.







...this row becomes **orphan**, since there's no movie row with **id** = **6**.

Preventing Orphan Records

The FOREIGN KEY constraint helps avoid orphan records.

DELETE FROM Movies WHERE id = 6;

Statements that would otherwise result in orphan records will now generate errors.

ERROR: update or delete on table "movies" violates foreign key constraint "promotions_movie_id_fkey" on table "promotions" DETAIL: Key (id)=(6) is still referenced from table "promotions".

DELETE FROM Promotions WHERE movie_id = 6;

DELETE FROM Movies WHERE id = 6;



Preventing Orphan Records When Dropping Tables

Tables must be dropped in the correct order.

```
DROP TABLE Movies;
```

```
ERROR: cannot drop table movies because other objects depend on it DETAIL: constraint promotions_movie_id_fkey on table promotions depends on table movies
```

DROP TABLE Promotions;

DROP TABLE Movies;



Validating Data Insertion

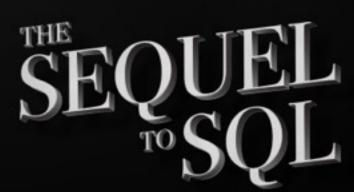
We want to make sure no film has a duration of less than 10 minutes.

Movies			
id	title	genre	duration
1	Don Juan	Romance	110
2	Peter Pan	Adventure	120
3	The Lost World	Fantasy	105
4	Wolfman Lives	Horror	-10

It's currently possible to set

a -10 minute duration, which
is clearly wrong.

INSERT INTO Movies (id, title, genre, duration)
VALUES (4, 'Wolfman Lives', 'Horror', -10);



Adding CHECK Constraint

The CHECK constraint is used to validate the value that can be placed in a column.

```
CREATE TABLE Movies
(
  id int PRIMARY KEY,
  title varchar(20) NOT NULL UNIQUE,
  genre varchar(100),
  duration int CHECK (duration > 0)
);
```

```
INSERT INTO Movies (id, title, genre, duration)
VALUES (4, 'Wolfman Lives', 'Horror', -10);
```

Attempts to insert invalid data on the **DURATION** column will now raise **errors**.

ERROR: new row for relation "movies" violates check constraint "movies_duration_check" DETAIL: Failing row contains (4, Wolfman Lives, Horror, -10).