

Instructions



SQL 2: **INSERT** and **SELECT** Data

COMP1048: Databases and Interfaces (2024-2025)

Matthew Pike and Yuan Yao

Overview



- In the previous lecture, we learned how to create tables using the **CREATE** statement.
- In this lecture, we will learn how to:
 - Insert data into a table using the **INSERT** statement.
 - Retrieve data from a table using the **SELECT** statement.
 - Filter data using the **WHERE** clause.
 - Sort data using the **ORDER BY** clause.

Inserting Data

```
INSERT INTO  
    table_name (column1, ...)  
VALUES  
    (value1, ...);
```

- **INSERT** adds data to a table as a new row(s).
- It's a Data Manipulation Language (DML) command.
- **INSERT INTO** specifies the table and optionally columns.
 - If columns aren't specified, values must cover all columns.
- **VALUES** provides the values to insert.
 - The number of values must match the columns.
 - Multiple rows can be inserted with comma-separated values.

Student Table Definition

We will use the following definition for the **Student** table in the coming examples:

```
CREATE TABLE Student (  
    sID INTEGER PRIMARY KEY,  
    sName VARCHAR(50) NOT NULL,  
    sAddress VARCHAR(255),  
    sYear INTEGER DEFAULT 1  
);
```


We can add a student to the **Student** table using **INSERT**:

```
INSERT INTO
  Student (sID, sName, sAddress,
    sYear)
VALUES
  (1, 'John S', '1 Sun St', 1);
```

Which means that the **Student** table now contains the following data:

sID	sName	sAddress	sYear
1	John S	1 Sun St	1

Table 1: There is now one row in our **Student** table.

Specifying Primary Keys

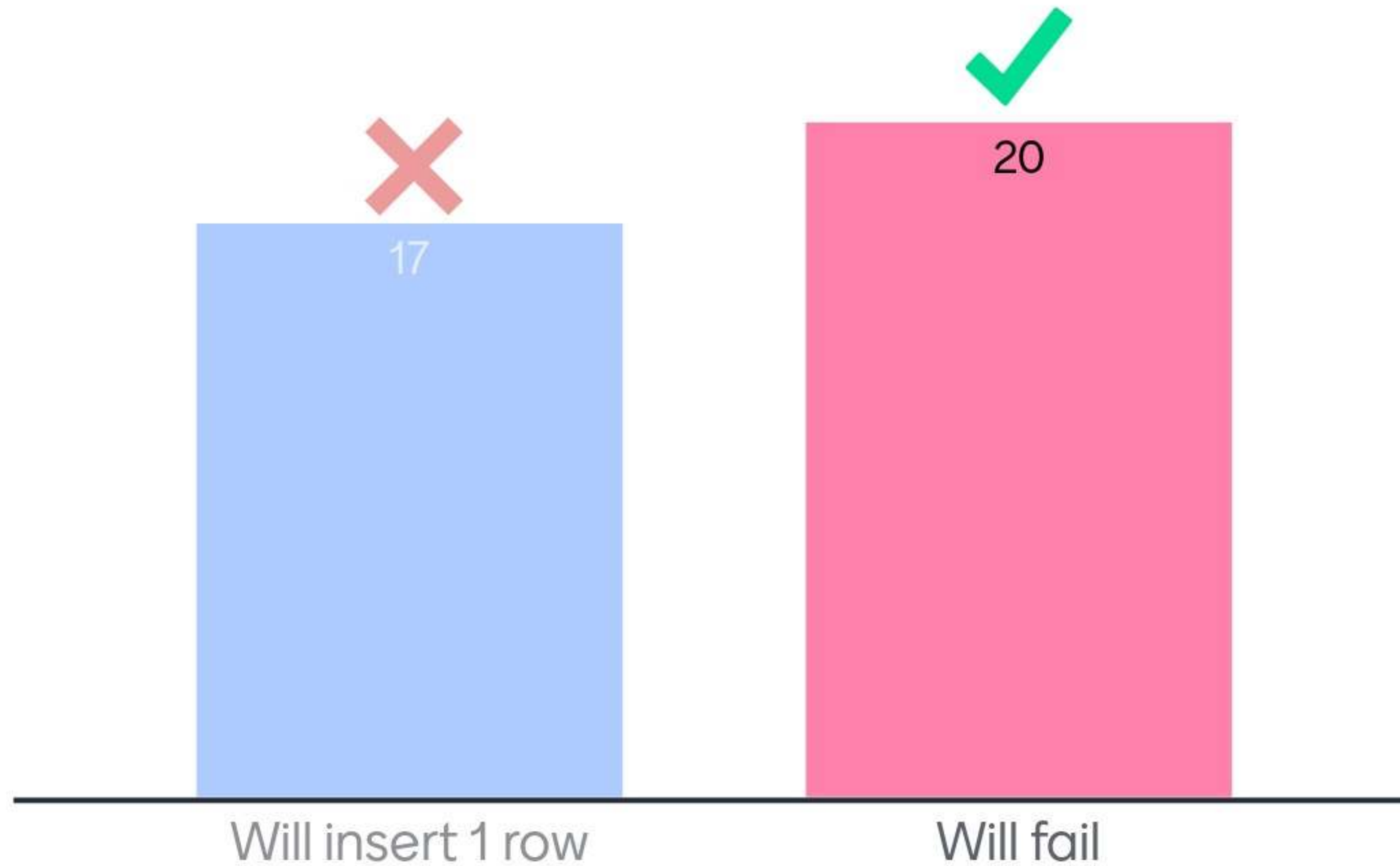
Notice, in this example, we did not specify a value for the **sID** column. If not specified, primary keys are automatically generated by the DBMS, and are guaranteed to be unique, but not necessarily sequential.

```
INSERT INTO Student
  (sName, sAddress, sYear)
VALUES
  ('Joe B', '2 Bay St', 2),
  ('Jane D', '3 Elm Rd', 3);
```

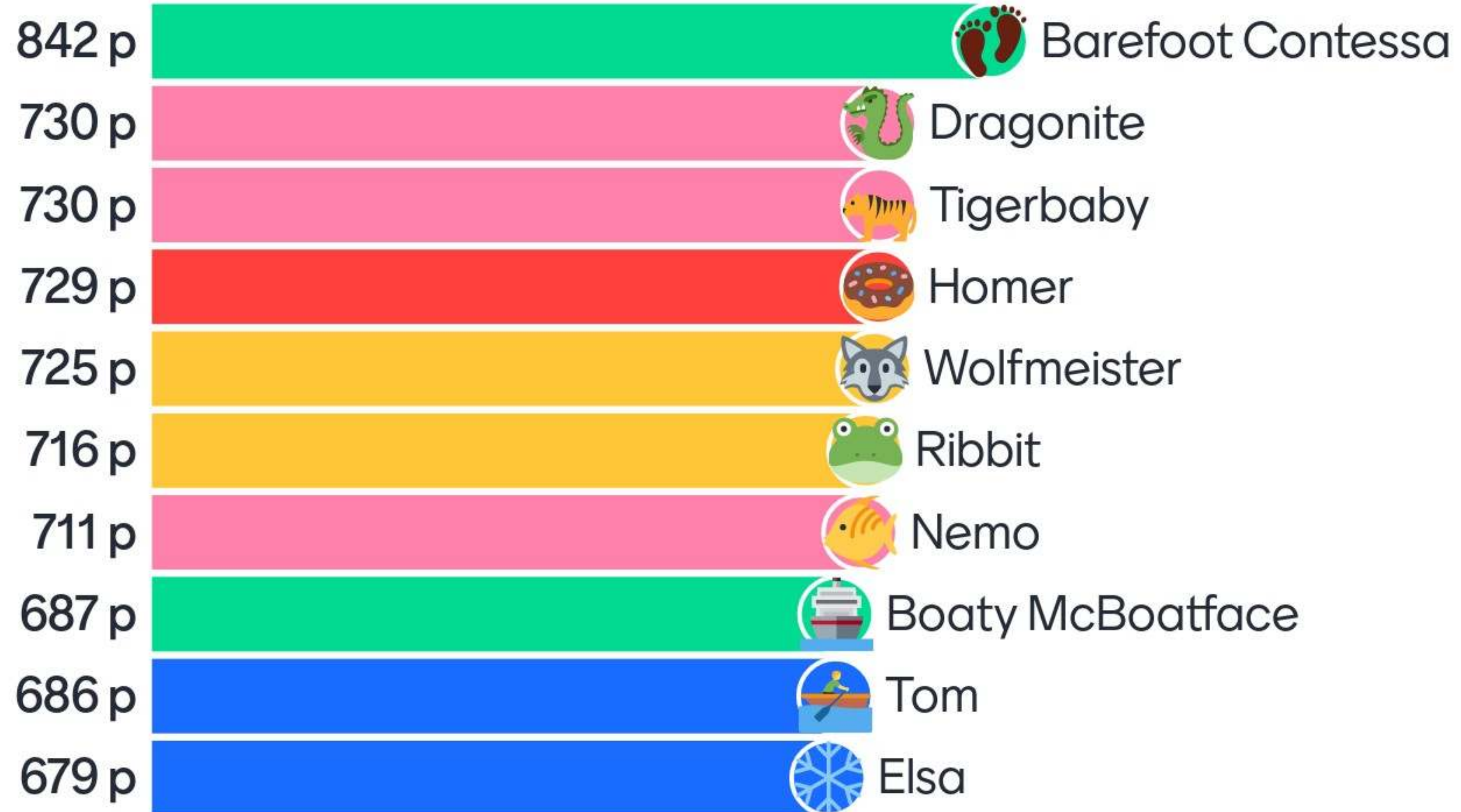
sID	sName	sAddress	sYear
1	John S	1 Sun St	1
2	Joe B	2 Bay St	2
3	Jane D	3 Elm Rd	3

Table 2: Including the previous entries, there are now three entries in the **Student** table.

INSERT INTO Student VALUES (NULL, NULL, NULL, NULL);



Quiz leaderboard



If a column has a **DEFAULT** value, then we do not need to specify a value for that column when inserting a new row.

```
INSERT INTO
    Student (sName, sAddress)
VALUES
    ('Jack T', '4 Bus Rd');
```

sID	sName	sAddress	sYear
1	John S	1 Sun St	1
2	Joe B	2 Bay St	2
3	Jane D	3 Elm Rd	3
4	Jack T	4 Bus Rd	1

Table 3: Including the previous entries, there are now four entries in the **Student** table.

! Specifying PRIMARY KEY Values

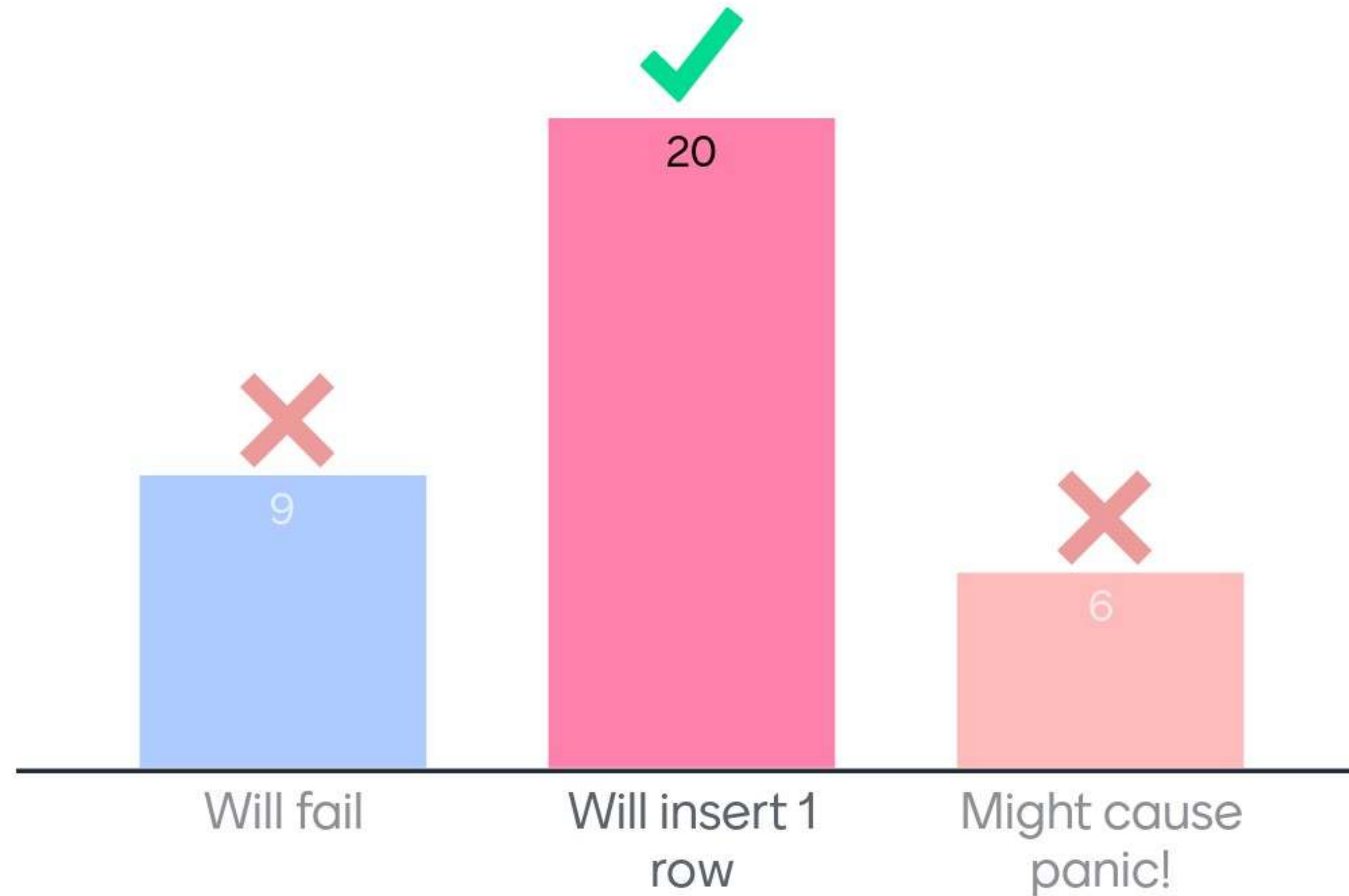
The following **INSERT** statement will result in an error, because the **sID** (**PRIMARY KEY**) column is not specified. Remember, if we do not specify which columns we're inserting into, we must provide values for all columns.

```
INSERT INTO Student VALUES ('Jess Y', '5 Oak St', 3);
```

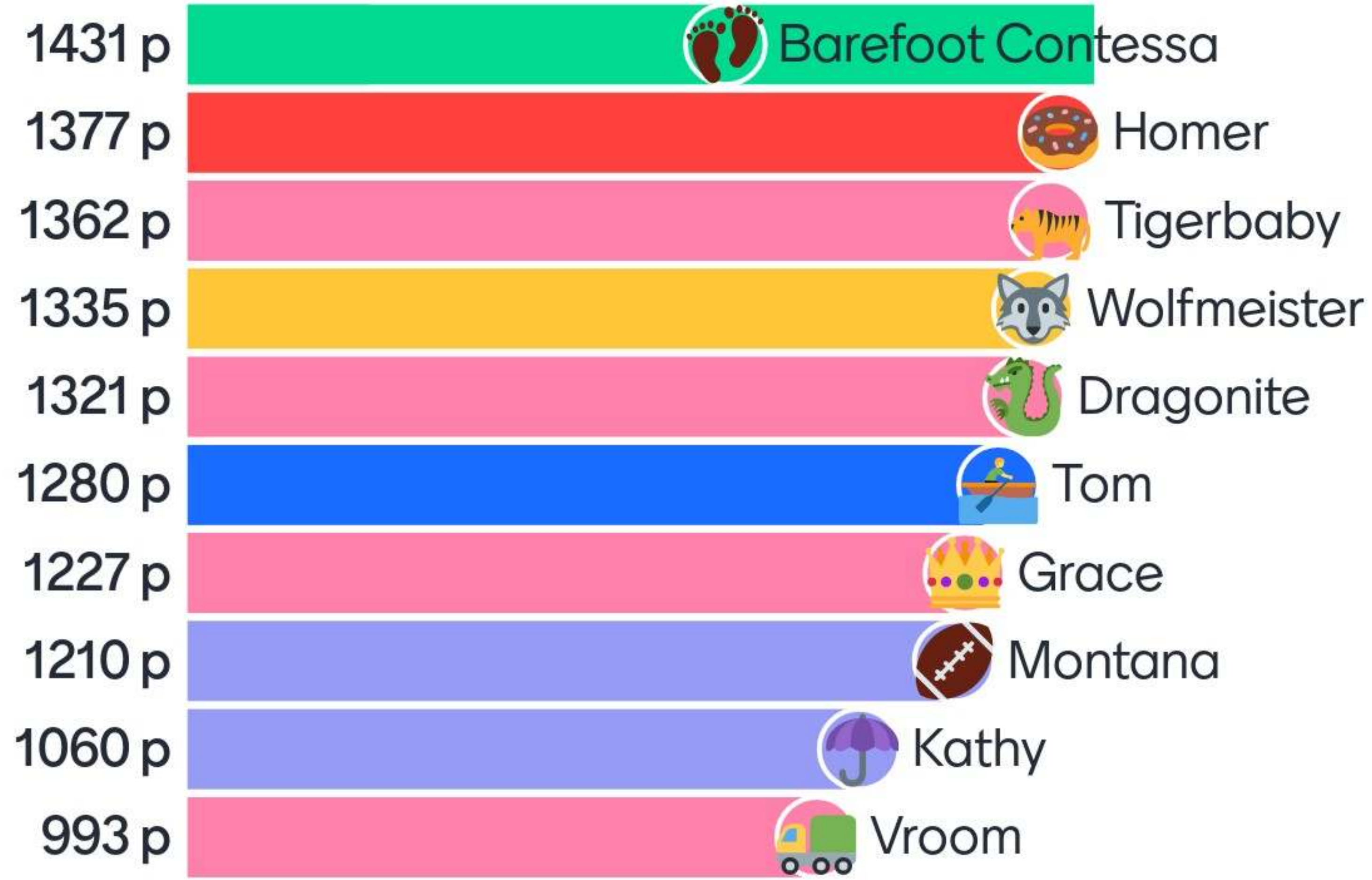
The following statement is valid, since the DBMS will generate a unique value in place of the **NULL** value for the **sID** column.

```
INSERT INTO Student VALUES (NULL, 'Jess Y', '5 Oak St', 3);
```

INSERT INTO Student(sName) VALUES ('Bob');



Quiz leaderboard



Selecting Data

```
SELECT
    column1, ...
FROM
    table_name
WHERE
    condition;
```

- The **SELECT** statement is a **DML** command for retrieving data from a table. The syntax definition given here is simplified, and does not include all possible clauses.
 - **column1, ...**: the names of the columns to include in the result set.
 - **table_name**: the name of the table to get data from.
 - **WHERE condition**: a condition that must be true for a row to be selected.

- Next, we will use **SELECT** to get data from the **Student** table.
- For reference, here is the table and its contents we will be using in the following examples:

sID	sName	sAddress	sYear
1	John S	1 Sun St	1
2	Joe B	2 Bay St	2
3	Jane D	3 Elm Rd	3
4	Jack T	4 Bus Rd	1
5	Jess Y	5 Oak St	3

Table 4: We will use this **Student** table in the following examples.

- The `*` operator is used to select all columns from a table.
- To retrieve all students from the **Student** table, we can use the following **SELECT** statement:

```
SELECT * FROM Student;
```

- We can read this statement as:
 - “Select all columns from the **Student** table”.

sID	sName	sAddress	sYear
1	John S	1 Sun St	1
2	Joe B	2 Bay St	2
3	Jane D	3 Elm Rd	3
4	Jack T	4 Bus Rd	1
5	Jess Y	5 Oak St	3

Table 5: Retrieving all columns and rows from the **Student** table.

- We can select specific columns to be returned by the **SELECT** statement.
- One or more columns can be specified, separated by commas.

```
SELECT  
    sName, sAddress  
  
FROM  
    Student;
```

sName	sAddress
John S	1 Sun St
Joe B	2 Bay St
Jane D	3 Elm Rd
Jack T	4 Bus Rd
Jess Y	5 Oak St

Table 6: Retrieving the **sName** and **sAddress** columns from the **Student** table.

- We can use **WHERE** to select only rows that meet a condition.
- For example, to get the names of students in year 2:
-

```
SELECT sName  
FROM Student  
WHERE sYear = 2;
```

sName

Joe B

Table 7: Names of students in year 2.

- Example conditions:
 - `sYear > 1`
 - `sName = 'John Smith'`
 - `sName <> 'John Smith'`
 - `sYear >= 2 AND sYear <= 3`
 - `sYear = 2 OR sYear = 3`

Combining Multiple Conditions using AND and OR

AND Operator

```
SELECT sID
FROM Student
WHERE
    sYear = 2
    AND
    sName = 'John S';
```

—
sID
—
—

Table 8: 0 records

OR Operator

```
SELECT sID
FROM Student
WHERE
    sYear = 2
    OR
    sName = 'John S';
```

—
sID
—
1
2
—

Table 9: 2 records

- Often, we want to remove duplicate rows from the result set. We can use the **DISTINCT** keyword to do this.

```
SELECT DISTINCT sYear FROM Student;
```

sYear
1
2
3

Table 10: The distinct values stored in the Year column.

Using **ORDER BY** to sort data

```
SELECT *  
FROM Student  
ORDER BY sYear;
```

sID	sName	sAddress	sYear
1	John S	1 Sun St	1
4	Jack T	4 Bus Rd	1
2	Joe B	2 Bay St	2
3	Jane D	3 Elm Rd	3
5	Jess Y	5 Oak St	3

Table 11: All student data, ordered by year.

- The **ORDER BY** clause is used to sort the result set by a column.
- The default sort order is ascending (**ASC**).
- To sort in descending order, use **DESC** after the column name.
- Unless explicitly specified, there is no guarantee on the order of rows with the same value in the sorted column; they may be returned in an arbitrary order.

```
SELECT * FROM Student
ORDER BY
    sYear DESC,
    sAddress ASC;
```

sID	sName	sAddress	sYear
3	Jane D	3 Elm Rd	3
5	Jess Y	5 Oak St	3
2	Joe B	2 Bay St	2
1	John S	1 Sun St	1
4	Jack T	4 Bus Rd	1

Table 12: All student data, ordered by year and address.

- We can sort by multiple columns
- The first column is used to sort the rows, and then the second column is used to sort the rows that have the same value in the first column

Reference




```
INSERT INTO  
    table_name (column1, ...)  
VALUES  
    (value1, ...);
```

- **INSERT** is a command to put data into a table
- **INTO** is a keyword that tells SQL where to put the data
- **table_name** the name of the table you want to put data into
- **column1, ...** are the names of the columns you want to put data into
- **VALUES** is a keyword that tells SQL what data to put into the table
- **value1, ...** are the values you want to put into the table

```
SELECT
    [DISTINCT] col1, ...
FROM
    table_name
WHERE
    condition
[ORDER BY
    column1 [ASC | DESC],
[GROUP BY
    column1, ...]
[HAVING
    condition]
```

- **SELECT** is a command to get data out of a table
- **DISTINCT** is a keyword that tells SQL to remove duplicate rows from the result set
- **FROM** is a keyword that tells SQL where to get the data from
- **WHERE** is a keyword that tells SQL which rows to get data from
- **ORDER BY** is a keyword that tells SQL how to sort the result set
- We haven't covered **GROUP BY** and **HAVING** yet, but we will cover them in a later lecture


```
SELECT
    column1, ...
FROM
    table_name
WHERE
    condition
ORDER BY
    column1, ... ASC|DESC;
```

- **ORDER BY** is a keyword that tells SQL to sort the data
- **column1, ...** are the names of the columns you want to sort by
- **ASC** is an optional keyword that tells SQL to sort in ascending order (default)
- **DESC** is an optional keyword that tells SQL to sort in descending order