Module 2-3

Joins

Keys

In a relational database, all rows must be unique. The column or combination of columns that make it unique are referred to as **key(s)**.

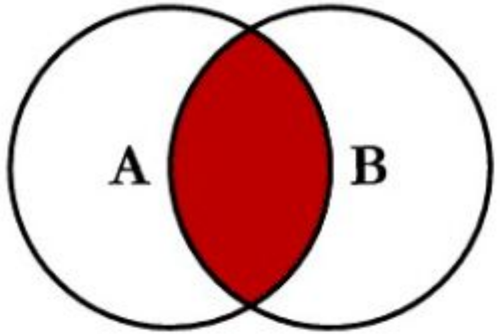
- **Primary Key**: column or columns in a table that uniquely identify the row. These cannot be duplicated.
  - If you say that SSN is your key, there cannot be more than one row with the same SSN.
- **Foreign Key**: A primary key present on another table.

Joins

Joins in SQL allow us to pull in data from several tables.

Joins : Inner Join

An inner join returns the rows in Table A that has a matching key value in Table B, the Venn Diagram representation is as follows:

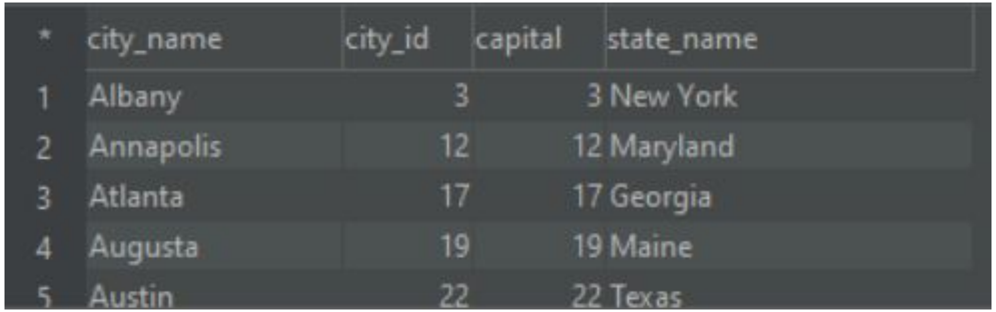


```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```

Joins : Inner Join Example

Consider the following example: **I want the city name along with the state_name.** The challenge is that these pieces of information are on two different tables. However, the capital column from state are city id's so we can bridge these two tables together:

```
SELECT
A.city_name,
A.city_id,
B.capital,
B.state_name
FROM city A
JOIN state B ON A.city_id = B.capital;
```

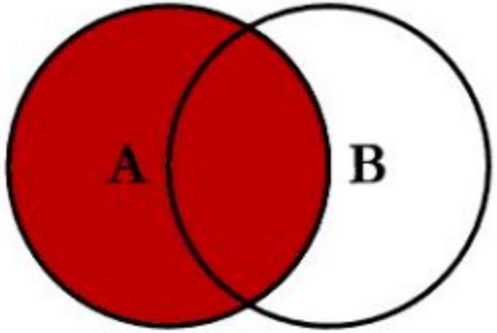


*	city_name	city_id	capital	state_name
1	Albany	3	3	New York
2	Annapolis	12	12	Maryland
3	Atlanta	17	17	Georgia
4	Augusta	19	19	Maine
5	Austin	22	22	Texas

Let’s write some joins!

Joins : Left Outer Join

The Left Outer Join returns all the rows on the “left” side table of the join, it will attempt to match to the right side. If there is match… If it can’t find a match it includes it in the result, but with NULL values.



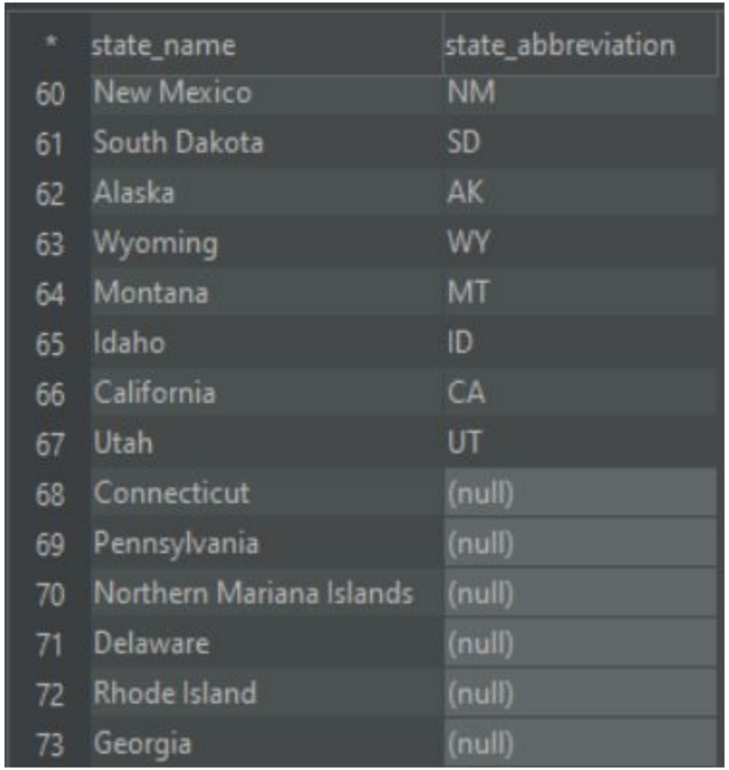
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

Joins : Left Outer Join Example

```
SELECT
state_name,
park_state.state_abbreviation
FROM state
LEFT OUTER JOIN park_state ON state.state_abbreviation = park_state.state_abbreviation;
```

With the Left Outer Join, SQL will try to join state_abbreviation from state and state_abbreviation from park_state.

If it can't find a match, the row from state will still be included but with a corresponding null value for the state_abbreviation it couldn't match.

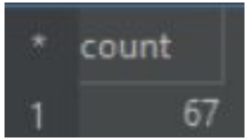


*	state_name	state_abbreviation
60	New Mexico	NM
61	South Dakota	SD
62	Alaska	AK
63	Wyoming	WY
64	Montana	MT
65	Idaho	ID
66	California	CA
67	Utah	UT
68	Connecticut	(null)
69	Pennsylvania	(null)
70	Northern Mariana Islands	(null)
71	Delaware	(null)
72	Rhode Island	(null)
73	Georgia	(null)

Joins : Left Outer Join vs Inner Join

With the same data set as the previous slide, let’s compare the LEFT OUTER vs an INNER.

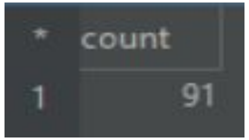
```
SELECT
COUNT(*)
FROM state
JOIN park_state ON state.state_abbreviation = park_state.state_abbreviation;
```



```
*  count
1     67
```

Note that since the LOJ will never drop any rows, it will have a higher row count.

```
SELECT
COUNT(*)
FROM state
LEFT OUTER JOIN park_state
ON state.state_abbreviation = park_state.state_abbreviation;
```



```
*  count
1     91
```

Unions

A union is a combination of two result sets. The following pattern is used:

[SQL Query 1]

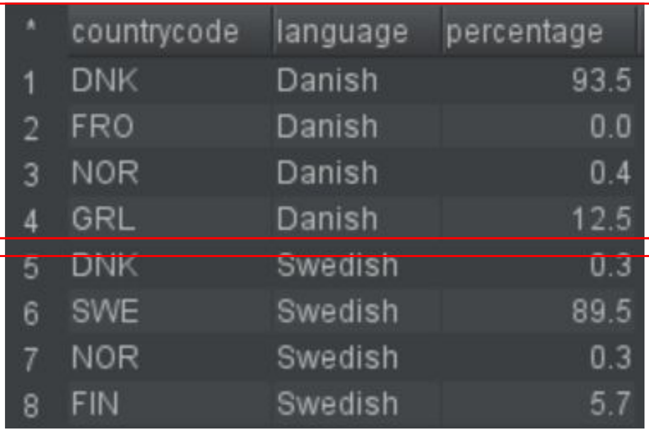
UNION

[SQL Query 2]

Unions Example

Consider the following query:

```
SELECT countrycode, language, percentage FROM countrylanguage where language = 'Danish'
UNION
SELECT countrycode, language, percentage FROM countrylanguage where language = 'Swedish'
ORDER BY countrycode;
```



	countrycode	language	percentage
1	DNK	Danish	93.5
2	FRO	Danish	0.0
3	NOR	Danish	0.4
4	GRL	Danish	12.5
5	DNK	Swedish	0.3
6	SWE	Swedish	89.5
7	NOR	Swedish	0.3
8	FIN	Swedish	5.7

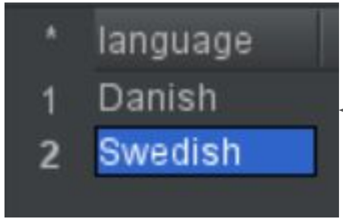
Result of query first query (language = ‘Danish’)

Result of query second query (language = ‘Danish’)

Union All

Suppose we changed the previous to only return the language instead:

```
SELECT language FROM countrylanguage where language = 'Danish'
UNION
SELECT language FROM countrylanguage where language = 'Swedish'
ORDER BY language
```



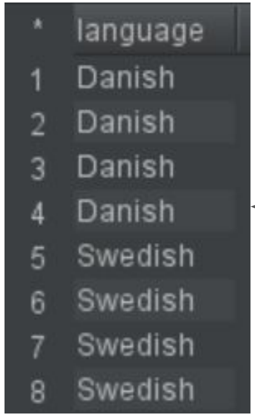
```
  language
1 Danish
2 Swedish
```

Note that the query only returns the unique values associated with countrylanguage

Union All

In situations like this, we can override this behavior by specifying UNION ALL instead:

```
SELECT language FROM countrylanguage where language = 'Danish'
UNION ALL
SELECT language FROM countrylanguage where language = 'Swedish'
ORDER BY language
```



```
*  language
1  Danish
2  Danish
3  Danish
4  Danish
5  Swedish
6  Swedish
7  Swedish
8  Swedish
```

Duplicates are now not taken into account.