What's SQL all about?

SQL, or Structured Query Language, is a language for talking to databases. It lets you select specific data and build complex reports. Today, SQL is a universal language of data, used in practically all technologies that process data.

PRAGMATIC WORKS

AGGREGATIONS

SUM(col1)	Sum a column
COUNT(*)	Count all rows
COUNT (DISTINCT col1)	Count unique rows
AVG(col1)	Average a column
MIN(col1)	Smallest column value
MAX(col1)	Largest column value

BEGINNER SYNTAX

SELECT	Select the columns
FROM	Which table to poll from
WHERE	Filter the data
GROUP BY	Aggregate the data
HAVING	Filter an aggregate
ORDER BY	Sort the data

COMPARISON OPERATORS

=	Equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
<>	Not equal to

INTERMEDIATE SYNTAX

LIKE	Match on a pattern
AND	Both criteria
OR	One or the other
CASE WHEN	If then logic
IN	Filter by a list
UNION ALL	Append data
BETWEEN	Between two items
CAST	Change data type

EXAMPLES

Return all columns with filter

```
SELECT *
FROM [TABLE_NAME]
```

Return specific columns with multiple filters

```
SELECT [COLUMN_1], [COLUMN_2]
FROM [TABLE_NAME]
WHERE [COLUMN_1] >= [VALUE_1]
AND [COLUMN_2]< [VALUE_2]
```

Return a range of values using

```
BETWEEN SELECT [COLUMN_1],
[COLUMN_2]
FROM [TABLE_NAME]
WHERE [COLUMN_1] BETWEEN
[VALUE_1] AND [VALUE_2]
```

Return specific columns and sort by a column in descending order

```
SELECT [COLUMN_1], [COLUMN_2]
FROM [TABLE_NAME]
ORDER BY [COLUMN_1] DESC
```

Return a summarization of a column and group by the others

```
SELECT [COLUMN_1], [COLUMN_2],
SUM([COLUMN_3]) AS [COLUMN_3]
FROM [TABLE_NAME]
GROUP BY [COLUMN_1], [COLUMN_2]
```

Update rows in a table

```
UPDATE [TABLE_NAME]
SET [COLUMN_1] = [VALUE_1]
WHERE [COLUMN_2] >= [VALUE_2]
```

Insert rows into a table

```
INSERT INTO [TABLE_NAME]
([COLUMN_1],
[COLUMN_2]) VALUES ([VALUE_1],
[VALUE_2])
```

Delete rows from a table

```
DELETE FROM [TABLE_NAME] WHERE
[COLUMN_2] > [VALUE_1]
```

Want to learn more?

Learn T-SQL, the primary language for data management. Build strong fundamentals with Mitchell Pearson's session. Quick and efficient T-SQL startup. Learn database concepts, query writing, enhance your database expertise.

PRAGMATIC WORKS

JOIN EXAMPLES

LEFT JOIN returns all of the rows from Table A and only the matching rows from Table B

```
SELECT *
FROM [TABLE_NAME_A]
LEFT JOIN [TABLE_NAME_B] ON
[TABLE_NAME_A].[COLUMN_1] = [TABLE_NAME_B].[COLUMN_2]
```

RIGHT JOIN returns all of the rows from Table B and only the matching rows from Table A

```
SELECT *
FROM [TABLE_NAME_A]
LEFT JOIN [TABLE_NAME_B] ON
[TABLE_NAME_A].[COLUMN_1] = [TABLE_NAME_B].[COLUMN_2]
```

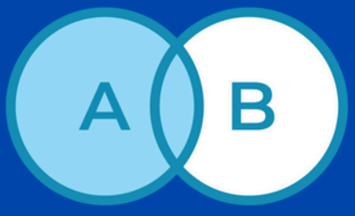
INNER JOIN returns online the rows that match between both Table A and Table B

```
SELECT *
FROM [TABLE_NAME_A]
INNER JOIN [TABLE_NAME_B] ON
[TABLE_NAME_A].[COLUMN_1] = [TABLE_NAME_B].[COLUMN_2]
```

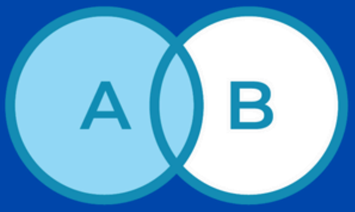
FULL OUTER JOIN returns all of the rows from both Table A and Table B regardless even if a match is not found

```
SELECT *
FROM [TABLE_NAME_A]
FULL OUTER JOIN [TABLE_NAME_B] ON
[TABLE_NAME_A].[COLUMN_1] = [TABLE_NAME_B].[COLUMN_2]
```

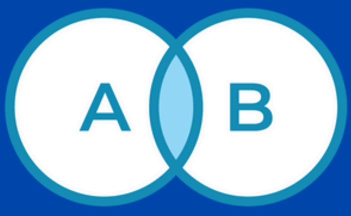
SQL JOINS



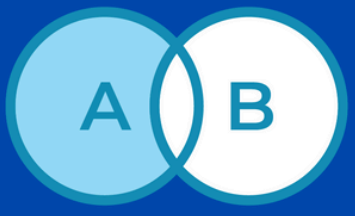
```
SELECT * FROM A
LEFT JOIN B ON
A.KEY = B.KEY
```



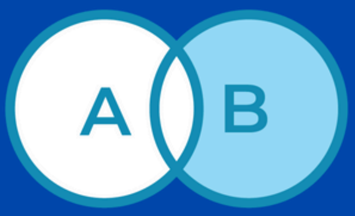
```
SELECT * FROM A
RIGHT JOIN B ON
A.KEY = B.KEY
```



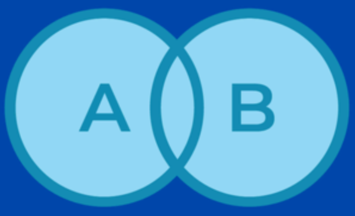
```
SELECT * FROM A
LEFT JOIN B ON
A.KEY = B.KEY
```



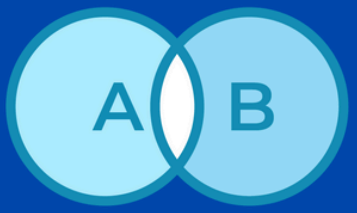
```
SELECT * FROM A
LEFT JOIN B ON
A.KEY = B.KEY IS
NULL
```



```
SELECT * FROM A
RIGHT JOIN B ON
A.KEY = B.KEY
WHERE B.KEY IS NULL
```

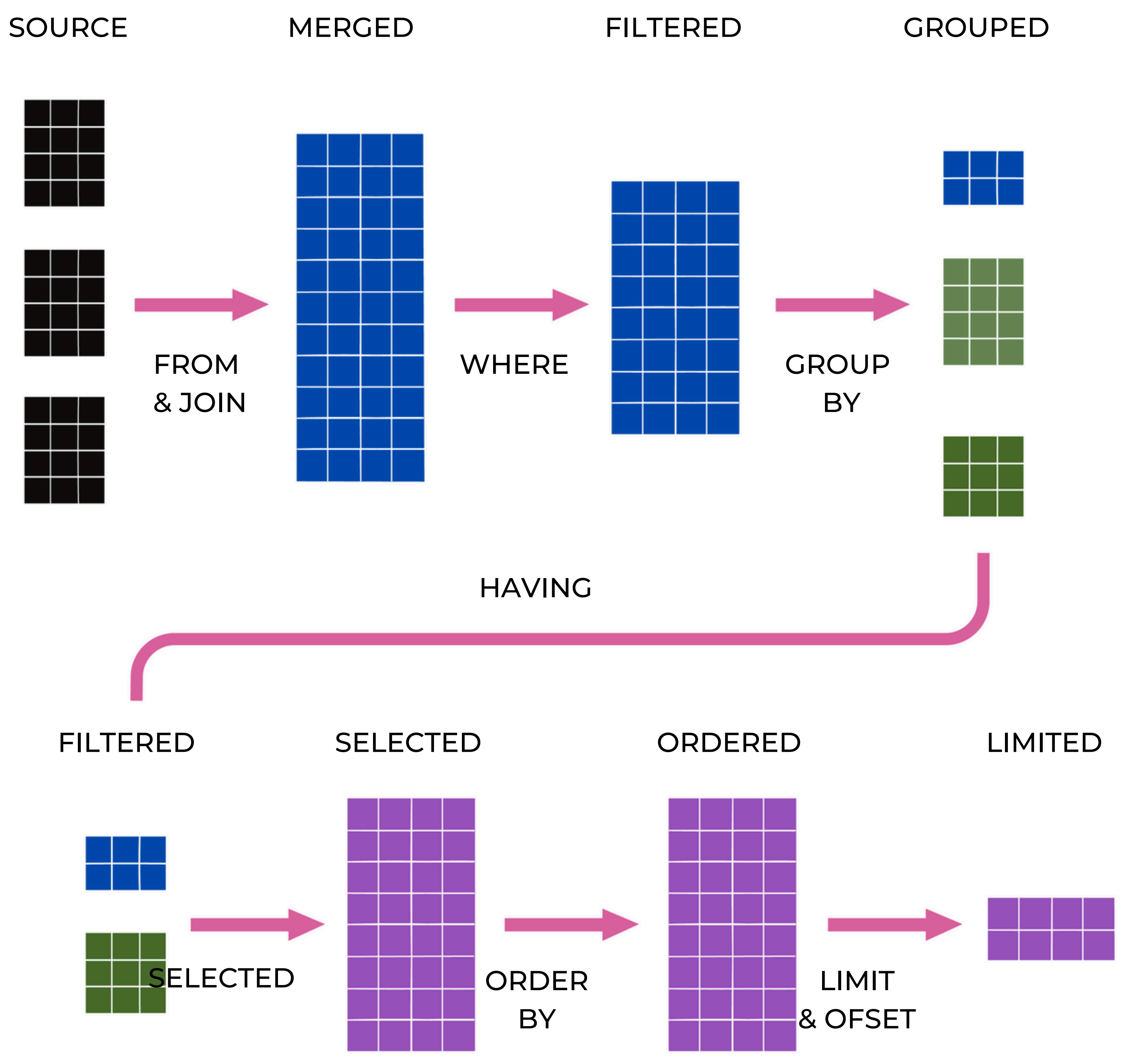


```
SELECT * FROM A
FULL OUTER JOIN B
ON A.KEY = B.KEY
```



```
SELECT * FROM A
FULL OUTER JOIN B
ON A.KEY = B.KEY
WHERE A.KEY IS NULL
```

ORDER THAT A SQL QUERY IS EXECUTED


SOURCE
MERGED
FILTERED
GROUPED
FROM & JOIN
WHERE
GROUP BY
HAVING
FILTERED
SELECTED
ORDERED
LIMITED
SELECTED
ORDER BY
LIMIT & OFSET