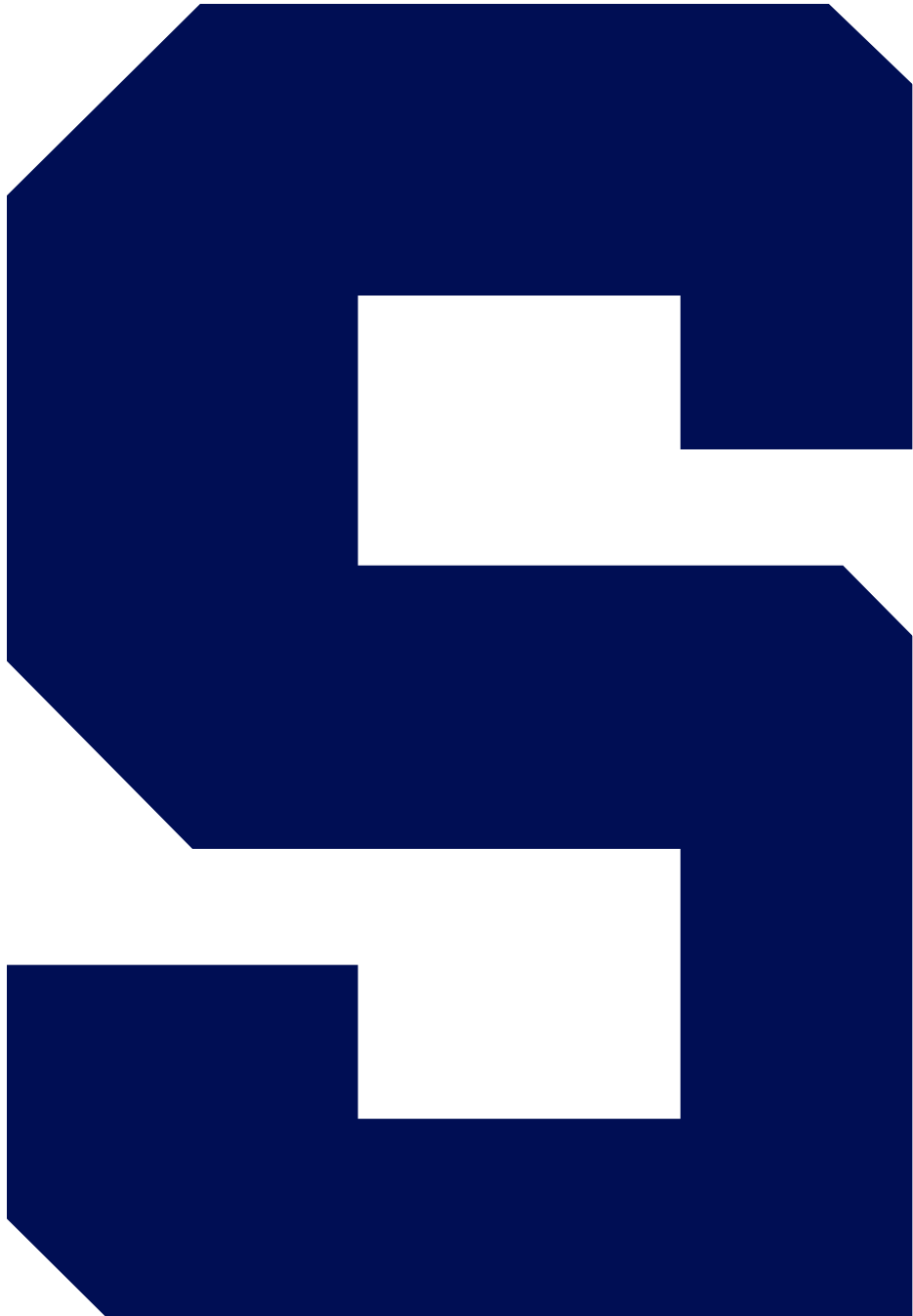
Syracuse University
School of Information Studies



SQL SELECT

Part I

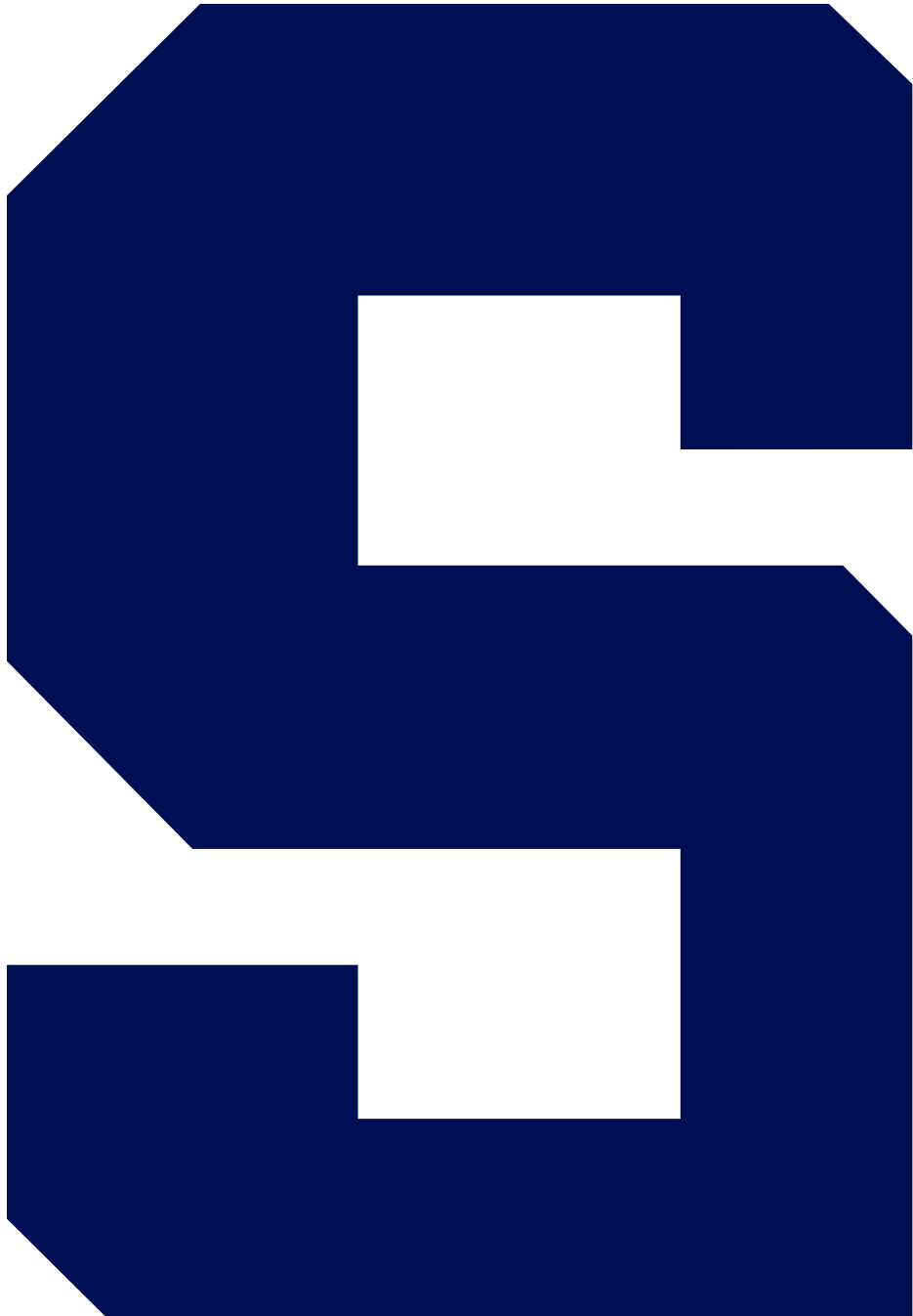
Agenda

- Column projections and aliasing
- The WHERE clause
- The ORDER BY clause
- Table inner joins
- Table outer joins
- Multi-table and self-referencing joins

SQL SELECT: Part I

The End

Syracuse University
School of Information Studies



SQL SELECT

SQL SELECT Statement

- The most popular and complex statement in the SQL language
- Used to query data; queries can range from the simple to the very complex
- Many different components to the statement, which are called clauses
- These clauses affect the operation of the command itself
- Most queries are intuitive

Recall: SQL SELECT

```
SELECT
    col1, col2, ...
    FROM table
    WHERE condition
    ORDER BY columns
```

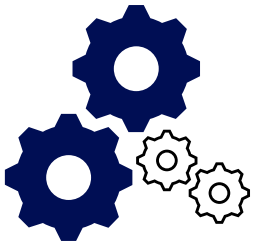
Columns to display (projection)

Table to use

Only return rows matching this condition

Sort row output by data in these columns

SELECT Statement Processing v1.0



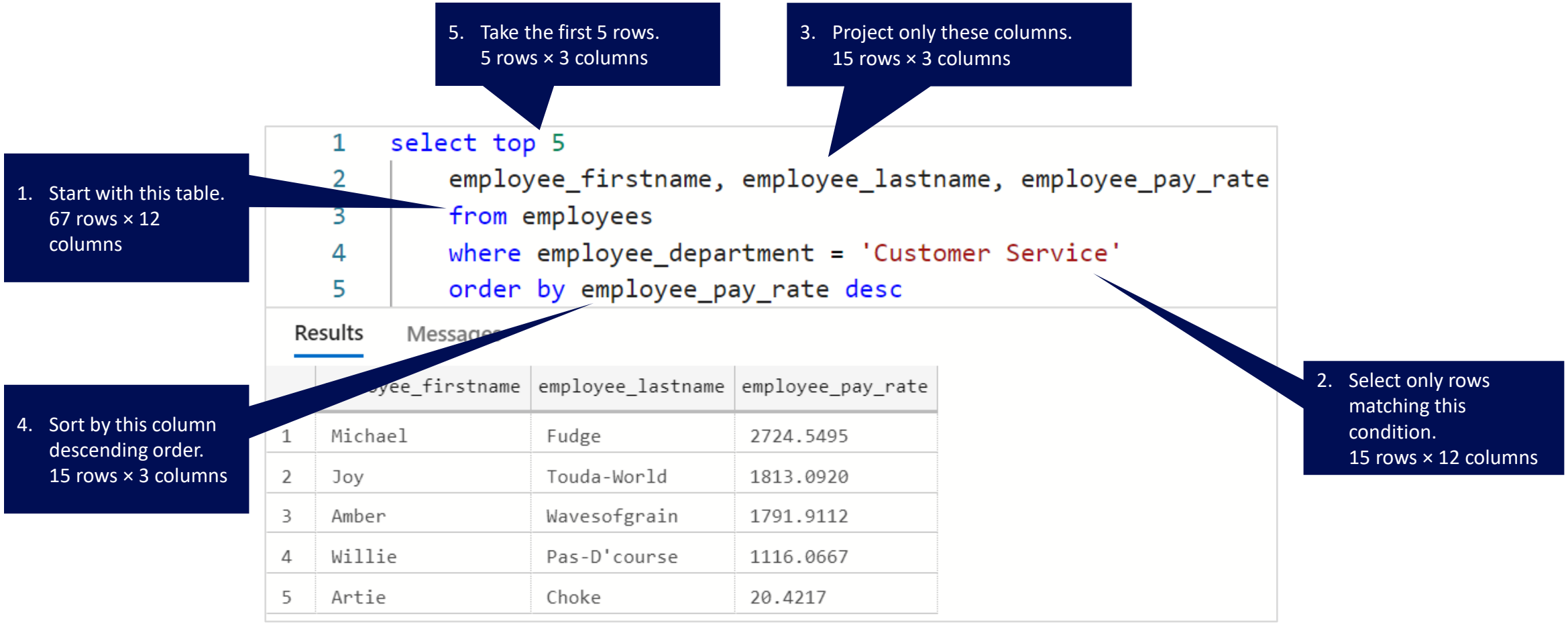
How we write it

1. TOP/DISTINCT
2. (Projection)
3. FROM
4. WHERE (selection)
5. ORDER BY

How it is processed

1. FROM
2. WHERE (selection)
3. (Projection)
4. ORDER BY
5. TOP/DISTINCT

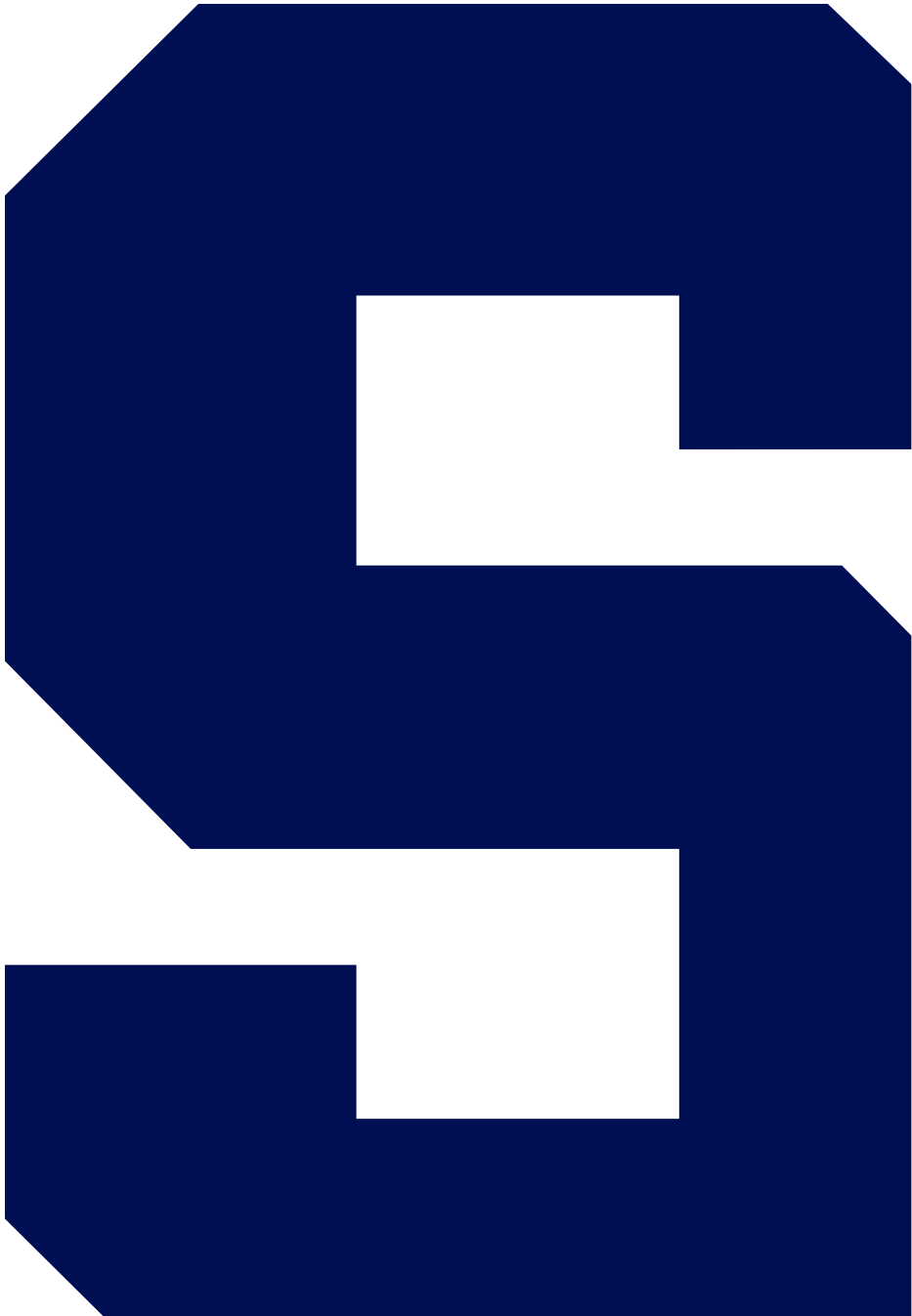
Query Processing



SQL SELECT

The End

Syracuse University
School of Information Studies



SELECT Projections

Projections

- Projections determine the columns in the query output.

```
SELECT student_name, student_gpa FROM students
```

students			
student_id	student_name	student_gpa	student_year
1	Robin Banks	4.00	Freshman
2	Victor Edance	3.81	Junior
3	Erin Yortires	2.40	Junior
4	Phil McCup	2.75	Senior

student_name	student_gpa
Robin Banks	4.00
Victor Edance	3.81
Erin Yortires	2.40
Phil McCup	2.75

Projections (cont.)

- If we do not know the columns, we can use the asterisk (*) to project all columns in the FROM.
- Since the output is not a table, we can project the same column multiple times.
- We can project expressions, which are applied to every row. These are called derived columns.
- We can rename any column using the AS keyword, creating a column alias.

Arithmetic Operators

Operator	Purpose	For types	Example
+	Addition	Exact and approximate numeric types	`7 + 5.6 = 12.6`
–	Subtraction	Exact and approximate numeric types	`14 – 3 = 11`
*	Multiplication	Exact and approximate numeric types	`0.1 * 3 = 0.3`
/	Integer division	Integers	`14 / 4 = 3`
/	Division	Exact and approximate numeric types	`14 / 4.0 = 3.5`
%	Remainder	Integers	`14 % 4 = 2`
+	Concatenation	String types	`‘Mi’ + ‘ke’ = ‘Mike’`

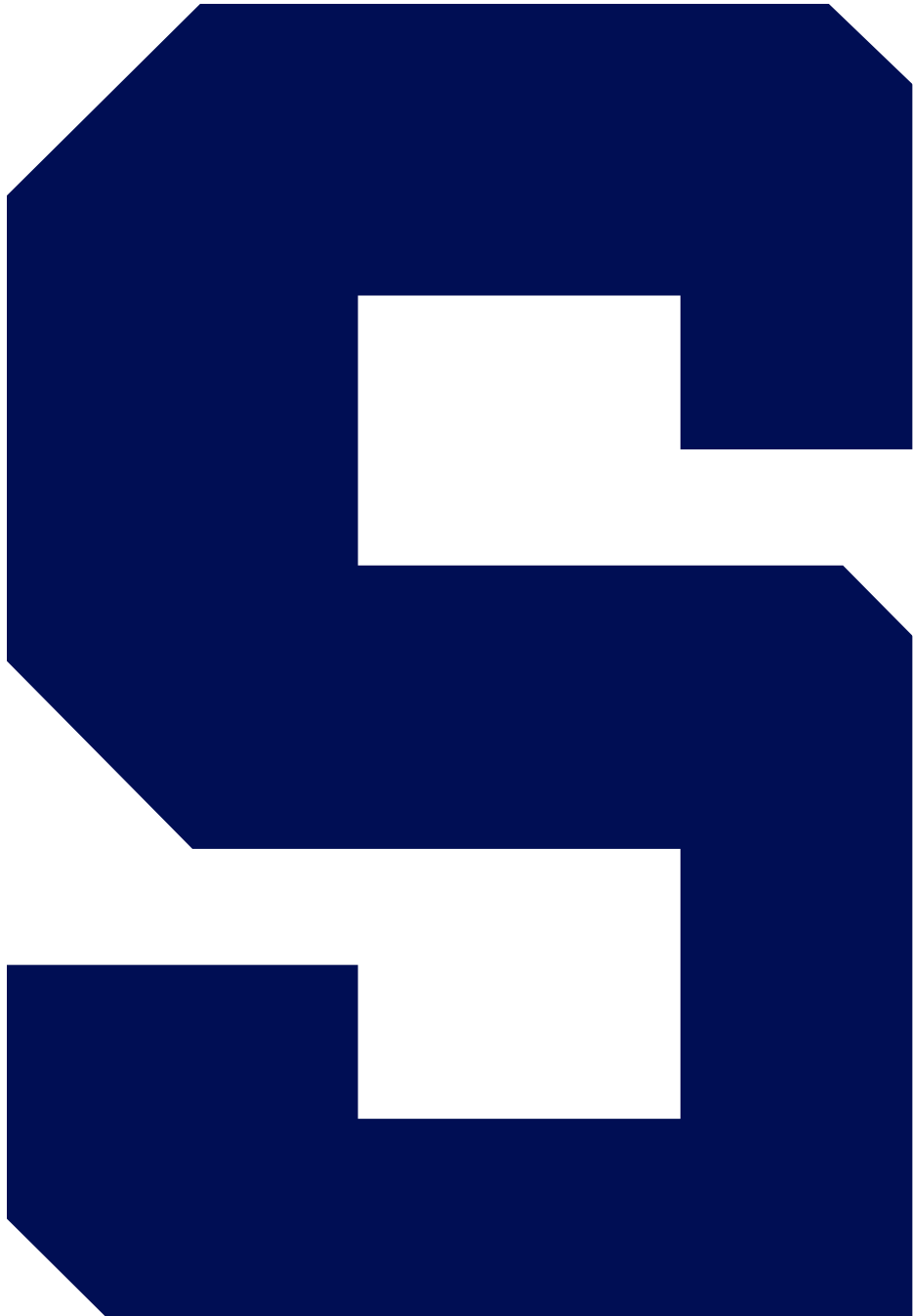
Data Type Casting

- Change the data type of an expression
- Useful for type conversion, truncation, parsing
- `CAST(expression AS new_type)`

SELECT Projections

The End

Syracuse University
School of Information Studies



Demo

SELECT Projections

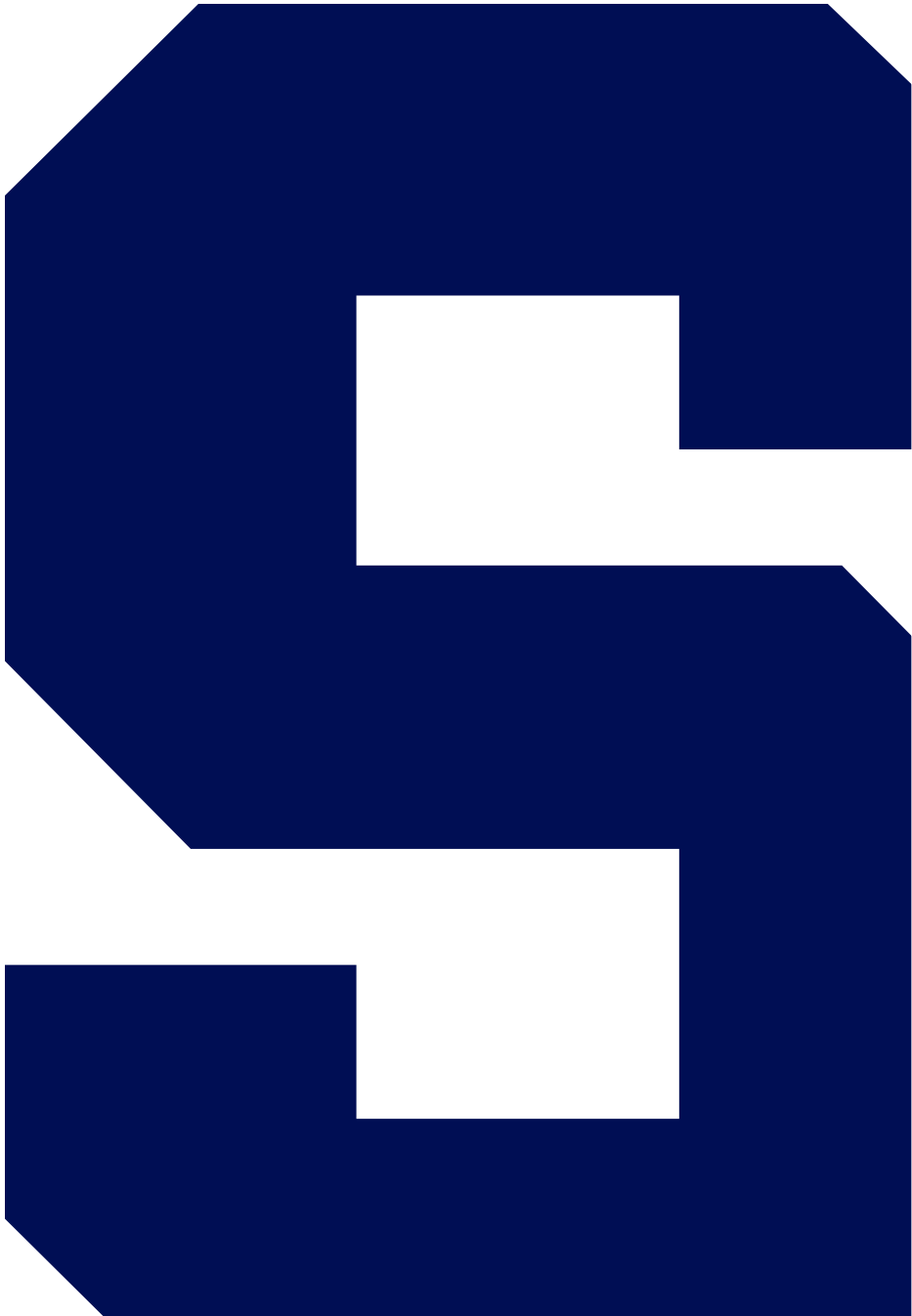
Demo: SELECT Projections

- We will use the payroll database.
- Selecting specific columns
- Column concatenation
- Column aliasing with AS
- CAST() data to different types
- The need for CAST(): conversion errors

Demo: SELECT Projections

The End

Syracuse University
School of Information Studies



SELECT WHERE

Selections (WHERE Clause)

- Selections determine the rows in the query output.

```
SELECT * FROM students WHERE student_year = 'Junior'
```

students			
student_id	student_name	student_gpa	student_year
1	Robin Banks	4.00	Freshman
2	Victor Edance	3.81	Junior
3	Erin Yortires	2.40	Junior
4	Phil McCup	2.75	Senior

student_id	student_name	student_gpa	student_year
2	Victor Edance	3.81	Junior
3	Erin Yortires	2.40	Junior

Selections (WHERE Clause) (cont.)

- If the WHERE clause is omitted, all rows are returned.
- The condition included in the WHERE clause is a Boolean expression.
- If the Boolean expression is true, the row is included in the output.
- Relational operators are Boolean expressions that compare values.
- Logical operators are Boolean expressions to connect relational expressions.

Relational Operators

Operator	Definition	Example of use
=	Equal to	`1 = 1`
!=	Not equal to	`'Mike' != 'Fudge'`
>	Greater than	`3 > 1`
<	Less than	`1 < 3`
>=	Greater than or equal to	`4 >= 4`
<=	Less than or equal to	`6 <= 10`
In	Set inclusion	`4 in (2,4,6,8)`
Not in	Set exclusion	`‘eggs’ not in (‘bacon’, ‘cheese’, ‘toast’)`
Is null	Column null check	`Column_name is null`
Is not null	Column not null check	`Column_name is not null`
Between	Inclusive numeric range	`Column_name between 0 and 4`
Like	Text pattern matching	`'Michael' like 'Mi%'`
Not Like	Text pattern not matching	`'Michael' not like 'Mike%'`

Logical Operators

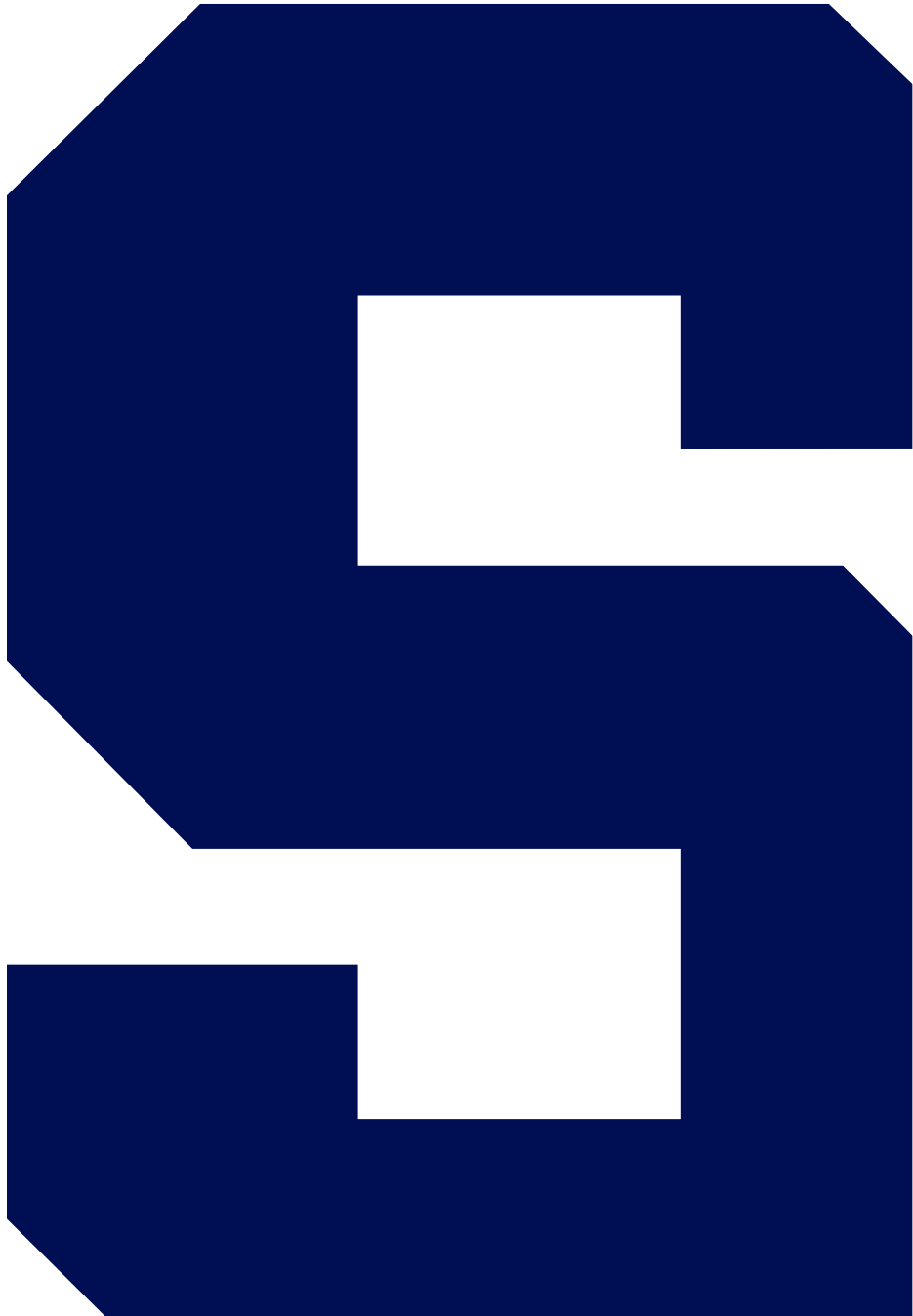
Operator	Definition	Example of use
AND	Evaluates to true only when both expressions are true, otherwise false	Customers with a gmail.com email address in the 315 area code `customer_phone like '315%' AND customer email like '%@gmail.com'`
OR	Evaluates to false only when both expressions are false, otherwise true	Students on academic warning below 2.0 GPA or on the Dean's list above 3.4 GPA `student_GPA >= 3.4 OR student_gpa < 2.0`

NOT	Negation; false becomes true, true becomes false	Employees not in the 'Toys' department
		`NOT Employee_department = 'Toys'`
		Note—this is the equivalent to:
		`Employee_department != 'Toys'`

SELECT WHERE

The End

Syracuse University
School of Information Studies



Demo

SELECT With WHERE

Demo: SELECT With WHERE

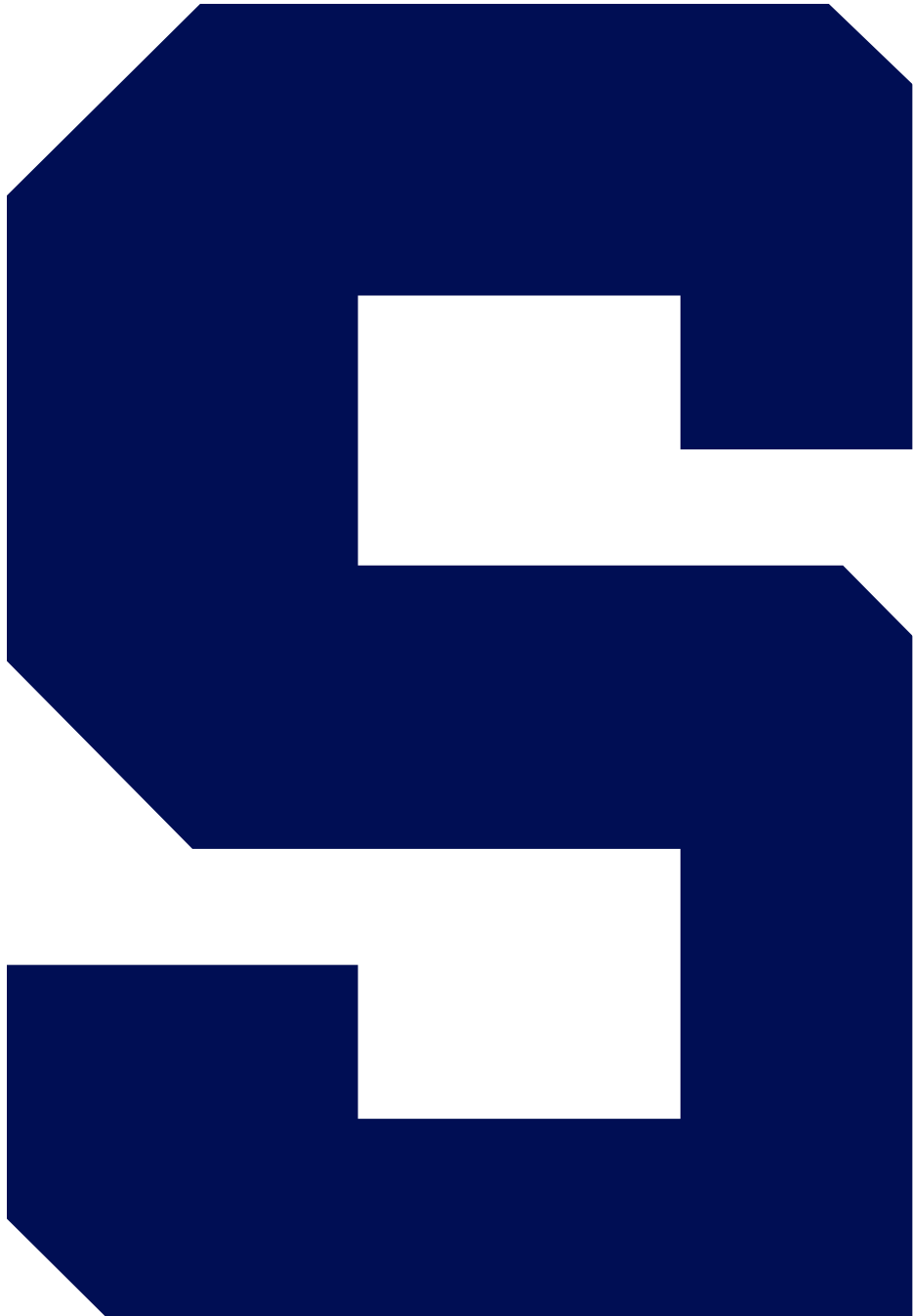
- We will use the payroll database
- WHERE clauses
- Relational operators >=, <
- Pattern matching with Like % and Like?
- Logical operators, AND/OR

Syracuse University
School of Information Studies

Demo: SELECT With WHERE

The End

SELECT CASE



SELECT

CASE

CASE Statement

- Tests a sequence of Boolean expressions
- The first one true returns the value
- Can be added anywhere a value is yielded—for example, projection, selection

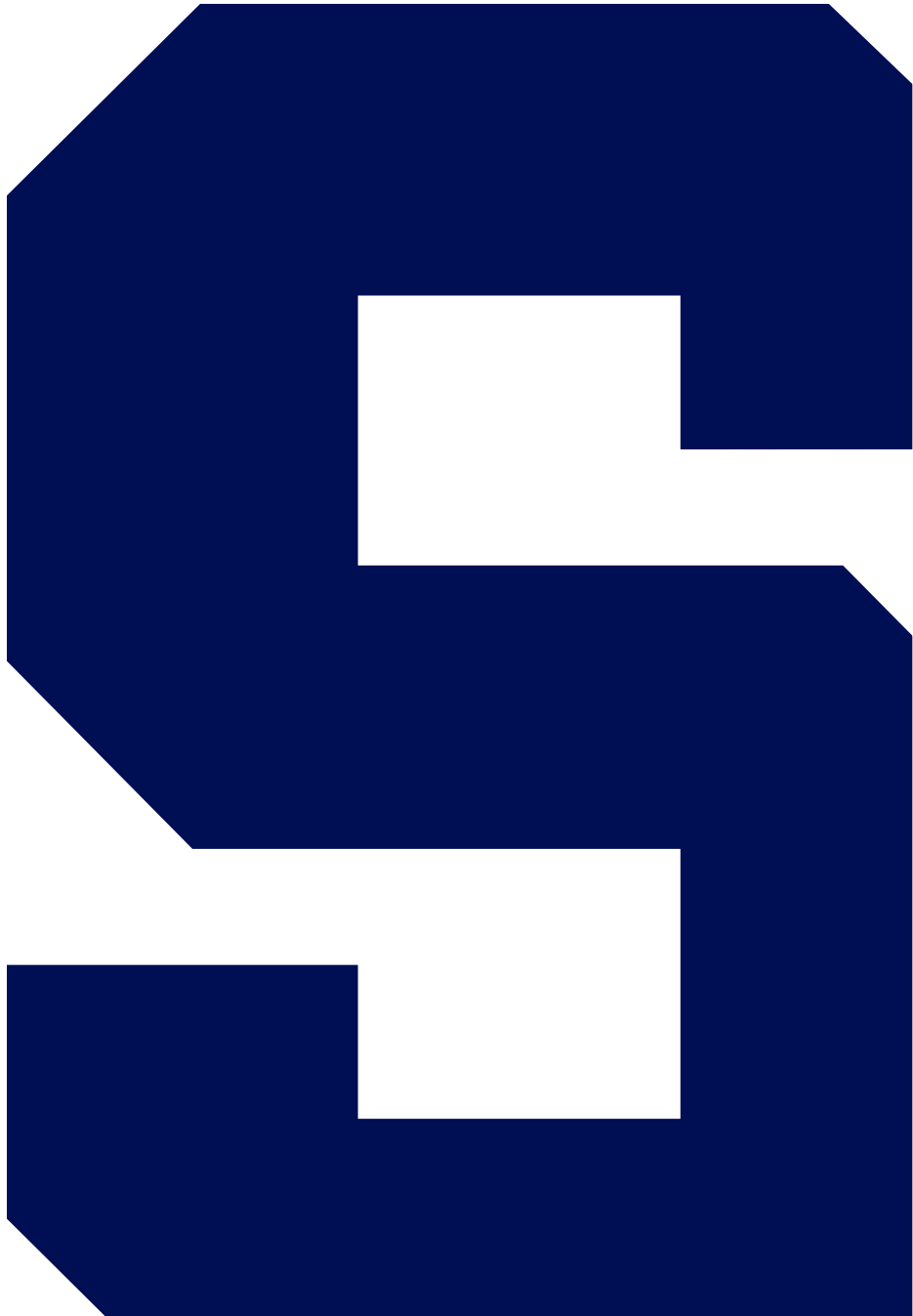
```
CASE
   WHEN condition1 THEN value1
   WHEN condition2 THEN value2
   ...
   [ ELSE catch_all_value ]
END
```

Syracuse University
School of Information Studies

SELECT CASE

The End

Syracuse University
School of Information Studies



Demo

CASE

Demo: SELECT With CASE

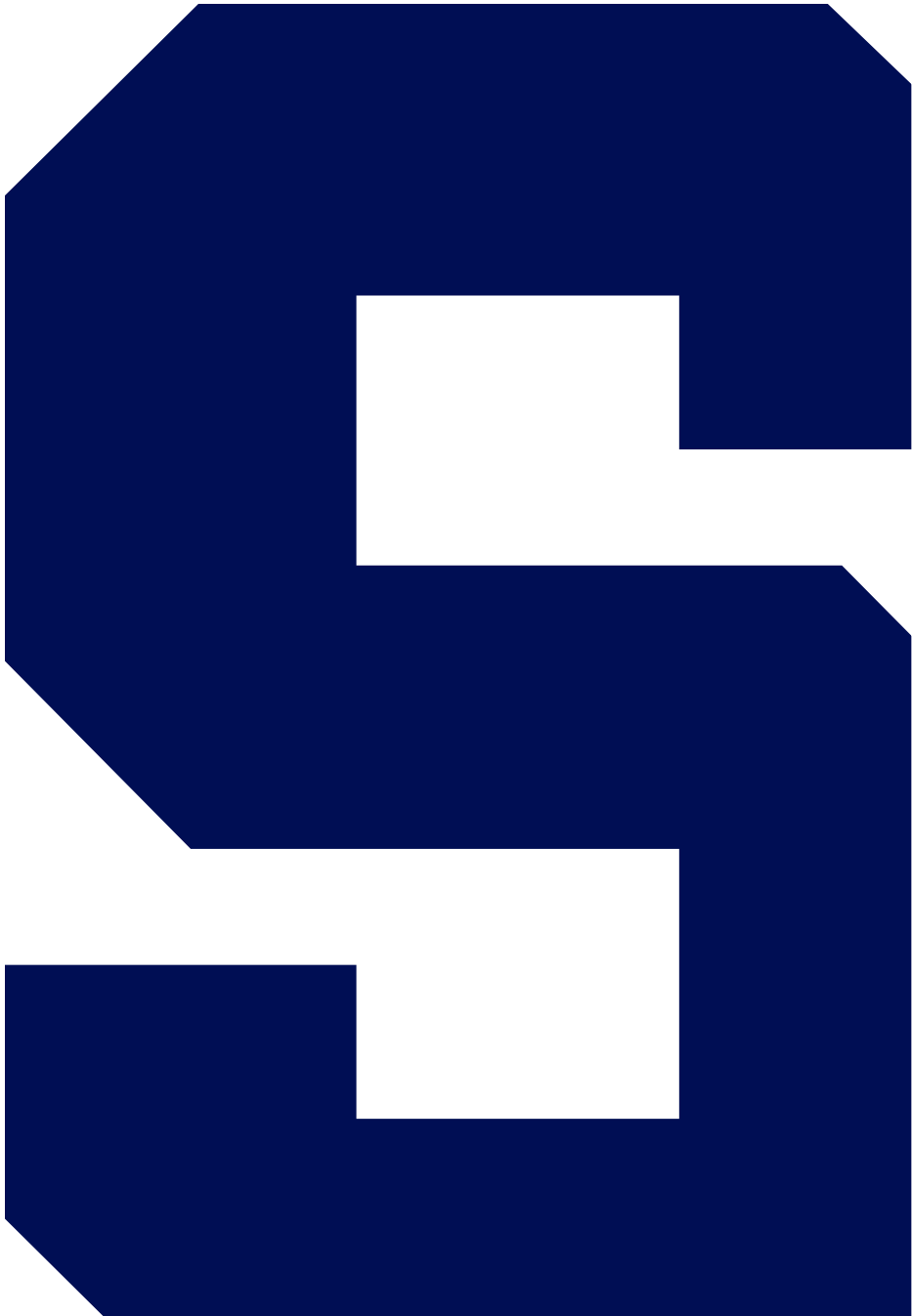
- We will use the payroll database
- CASE to create pay bands for sales associates
- Why doesn’t this work in the WHERE clause?
- Workarounds for this?

Syracuse University
School of Information Studies

Demo: CASE

The End

SELECT ORDER BY



Sorting Output (ORDER BY Clause)

- The ORDER BY clause is used to sort the query output.
- You must specify the columns by which you will sort.
- The ASC keyword sorts in ascending order and is the default.
- The DESC keyword sorts in descending order.
- If the ORDER BY clause is omitted, rows are returned in the default table order based on the primary key.

TOP and DISTINCT Keywords

TOP n

- Limits output to the first n rows
- This is based on the sorted output of the query

DISTINCT

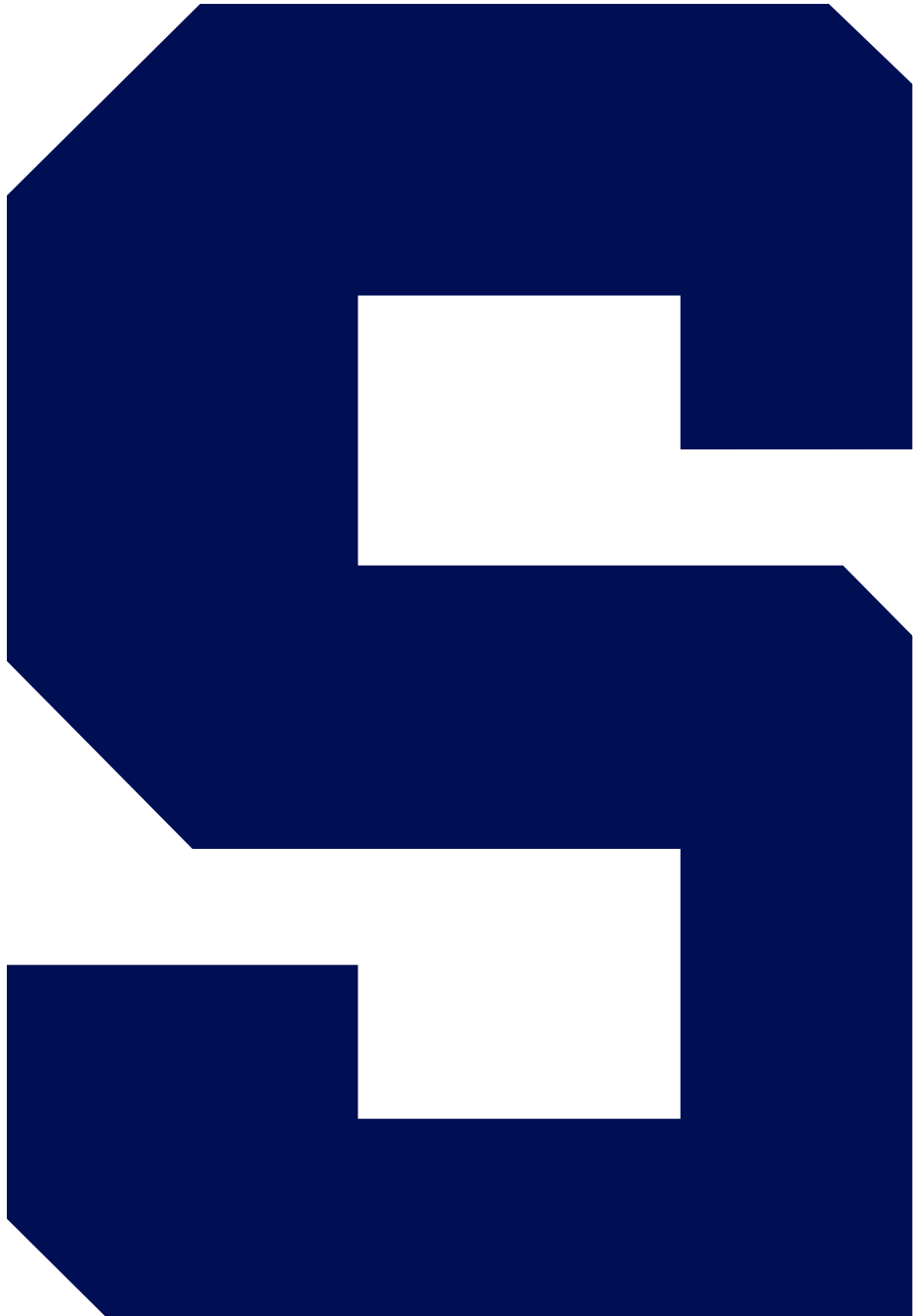
- Removes duplicate rows from the query output
- Guarantees the output has entity integrity

Syracuse University
School of Information Studies

SELECT ORDER BY

The End

Syracuse University
School of Information Studies



Demo

TOP, DISTINCT, ORDER BY

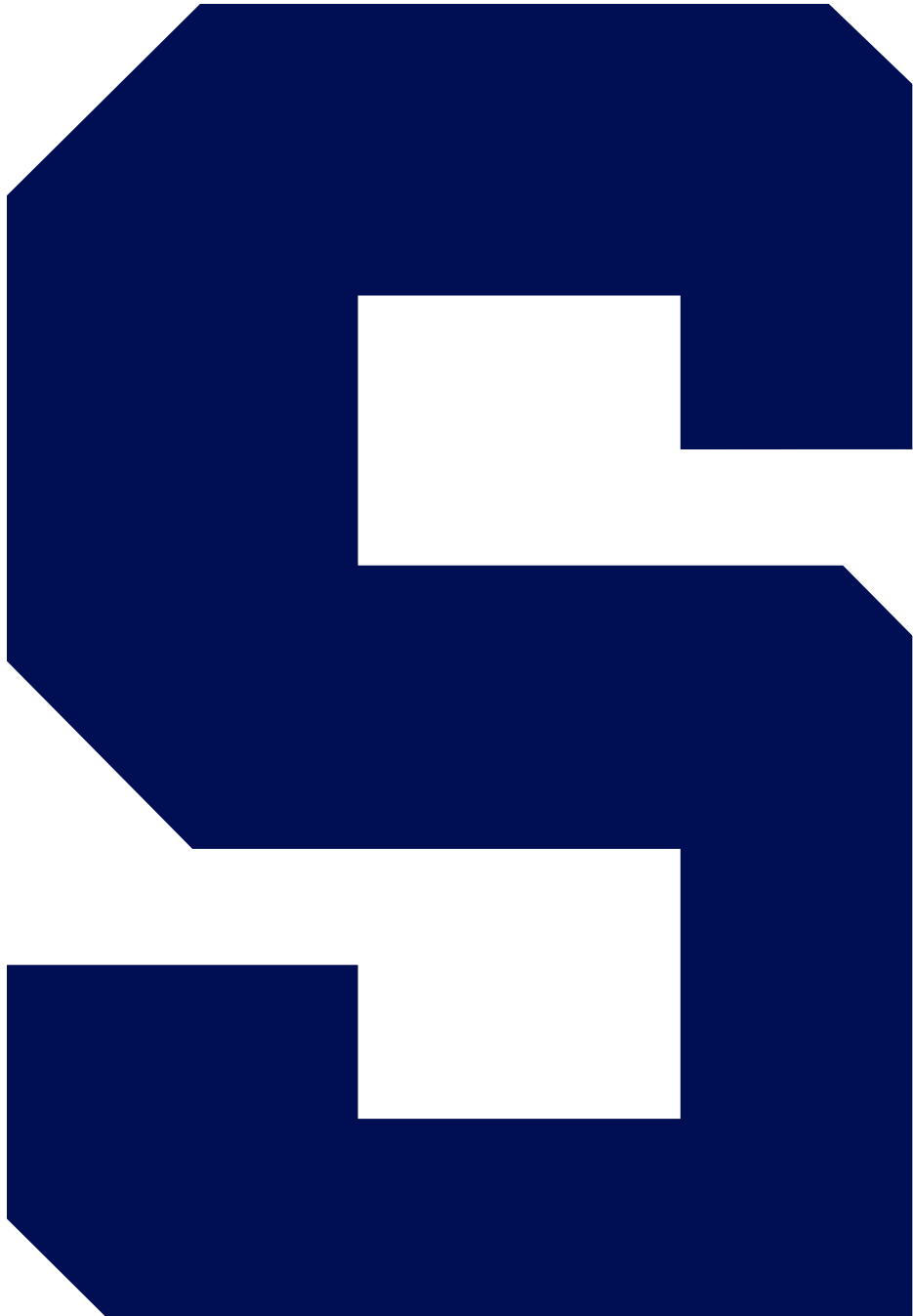
Demo: TOP, DISTINCT, ORDER BY

- We will use the payroll database
- DISTINCT Example
- ORDER BY to arrange output
- Combined with TOP to produce a specific number of rows, such as the highest or lowest

Demo: TOP, DISTINCT, ORDER BY

The End

Syracuse University
School of Information Studies

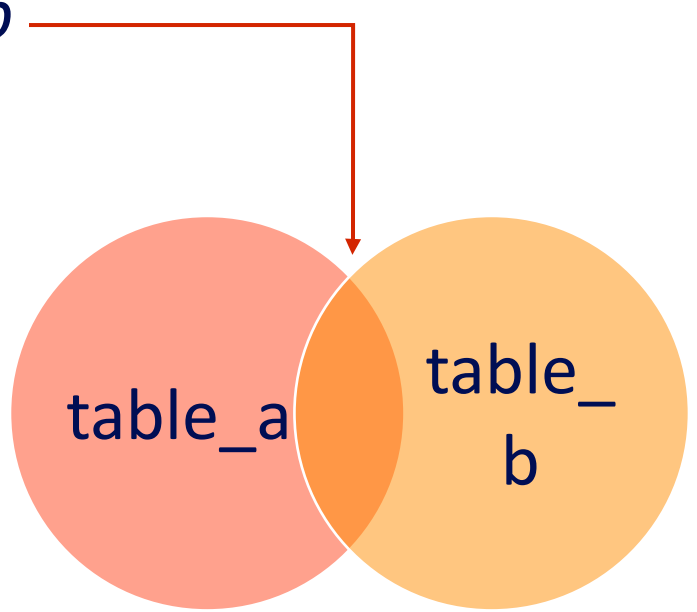


SELECT Joins

Joins: Inner Join

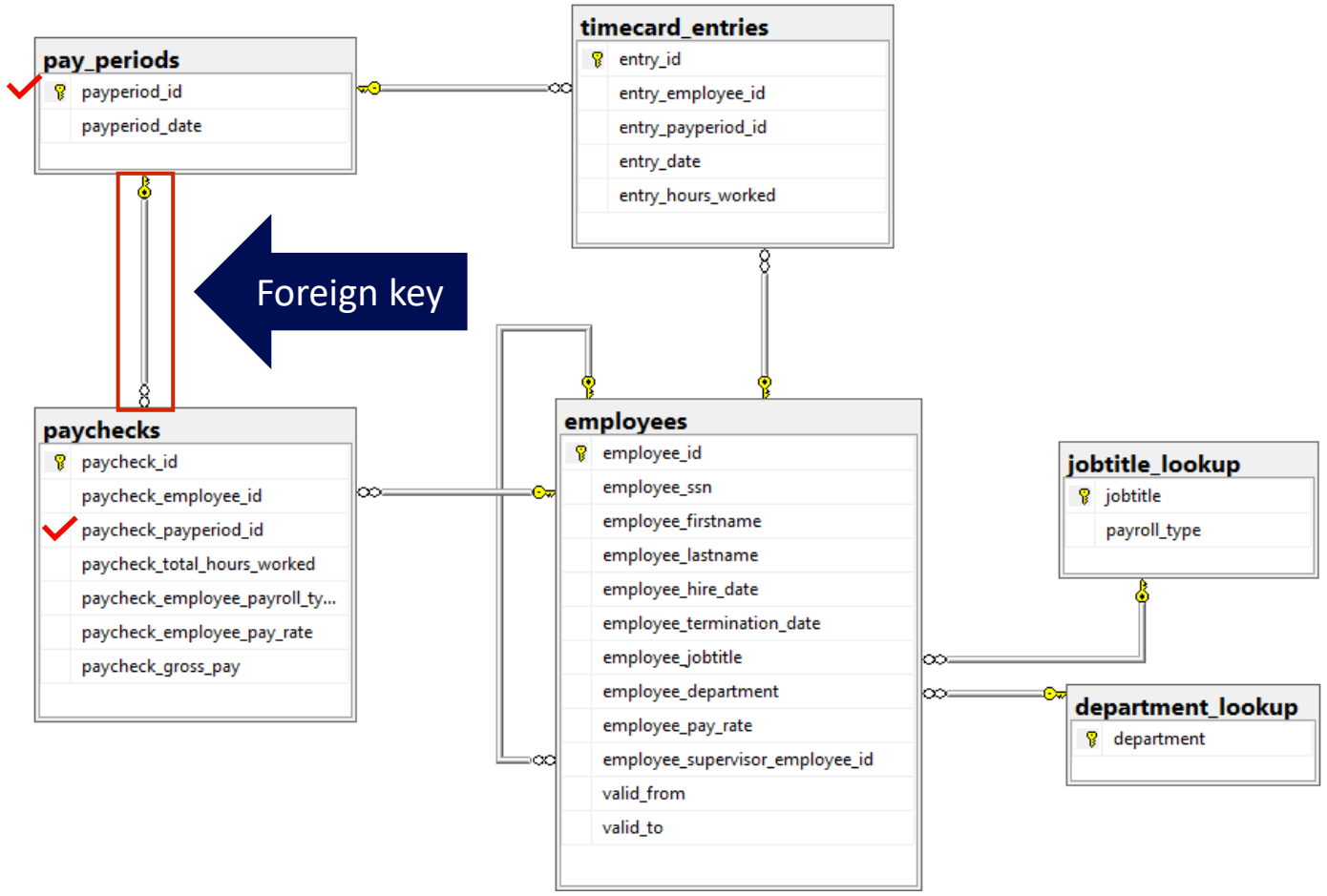
```
FROM table_a
   JOIN table_b ON column_a = column_b
```

- Processed first in the SELECT statement
- Columns must be the same data type
- Typically PK–FK pairs
- Output is a temporary table in the query pipeline consisting of matching rows



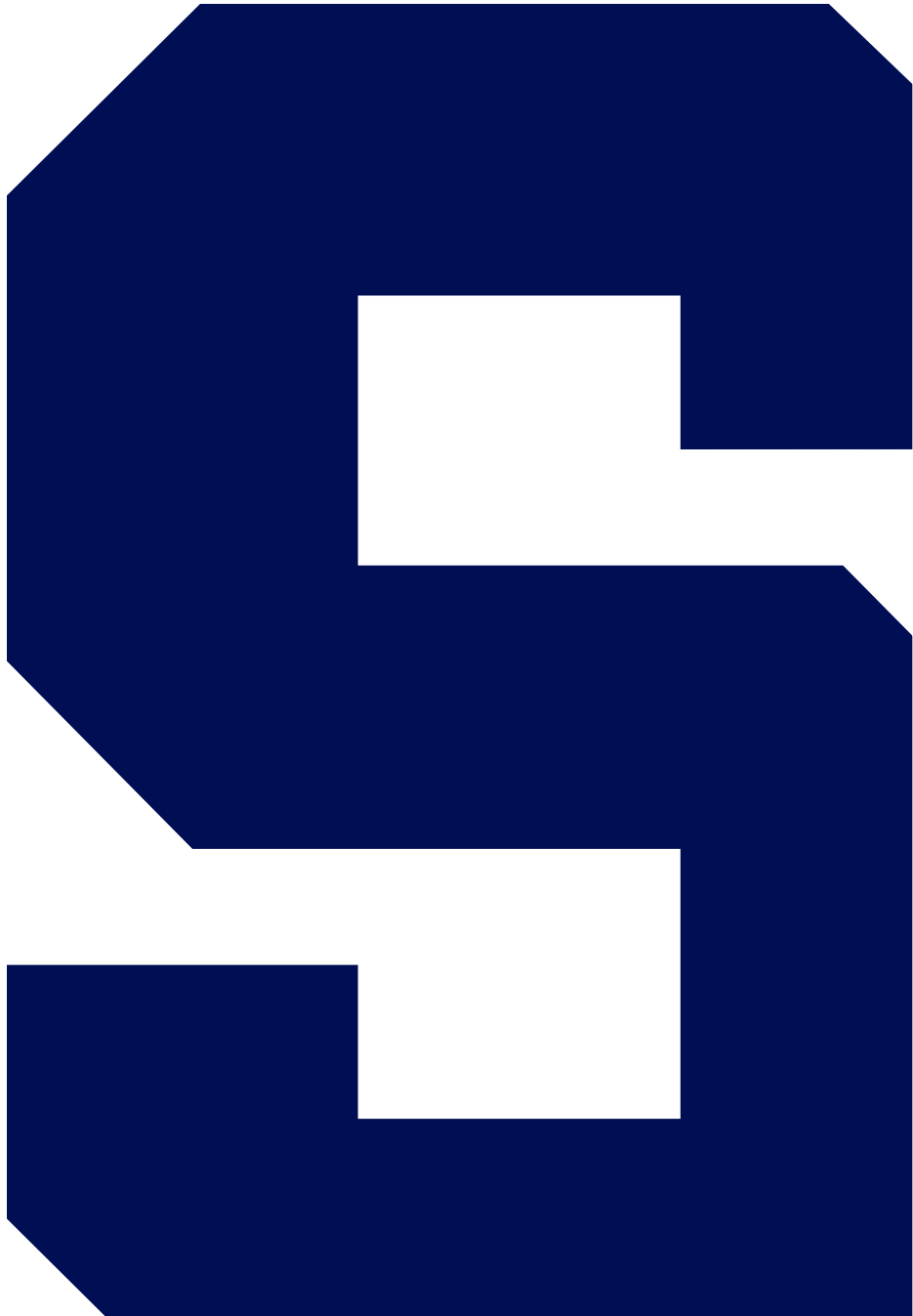
Payroll Database Internal Model

```
FROM paychecks
   JOIN pay_periods
     ON payperiod_id =
paycheck_payperiod_id
```



SELECT Joins

The End



Demo

Inner Join

Demo: Inner Join

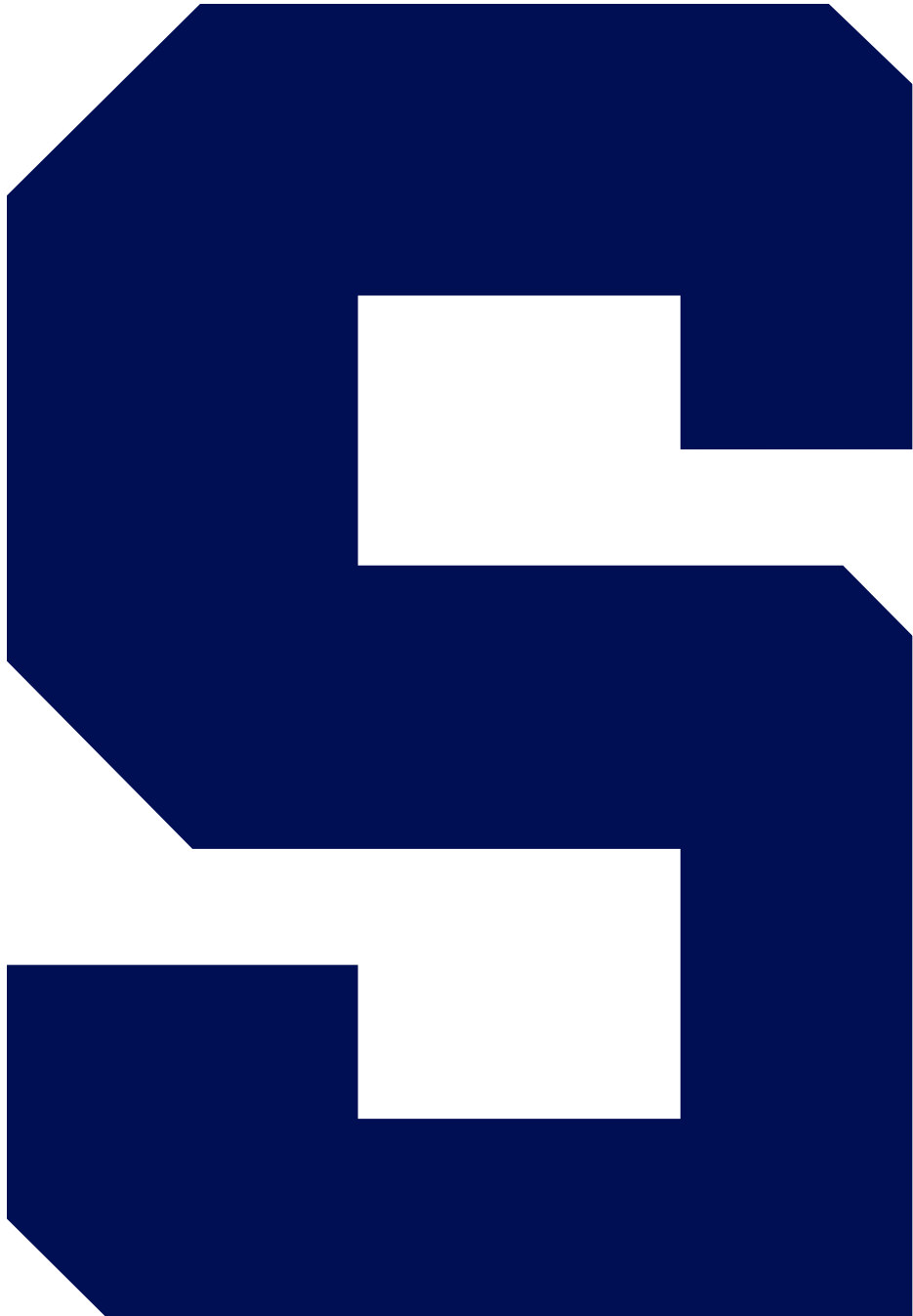
- We will use the demo and payroll databases.
- Inspect bbplayers and bbteams tables.
- Implement inner join.
- Join output is a subset of both tables.
- Note that you can join any column of the same data type, but that does not mean it is sensible to do so.

Syracuse University
School of Information Studies

Demo: Inner Join

The End

Syracuse University
School of Information Studies

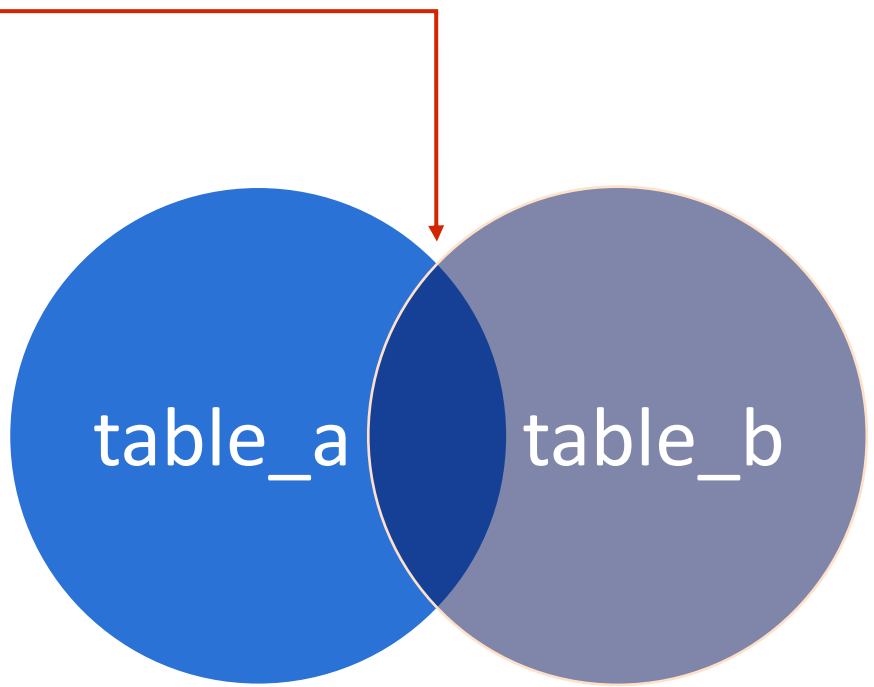


Outer Joins

Joins: Left Outer Join

```
FROM table_a LEFT JOIN table_b
   ON column_a = column_b
```

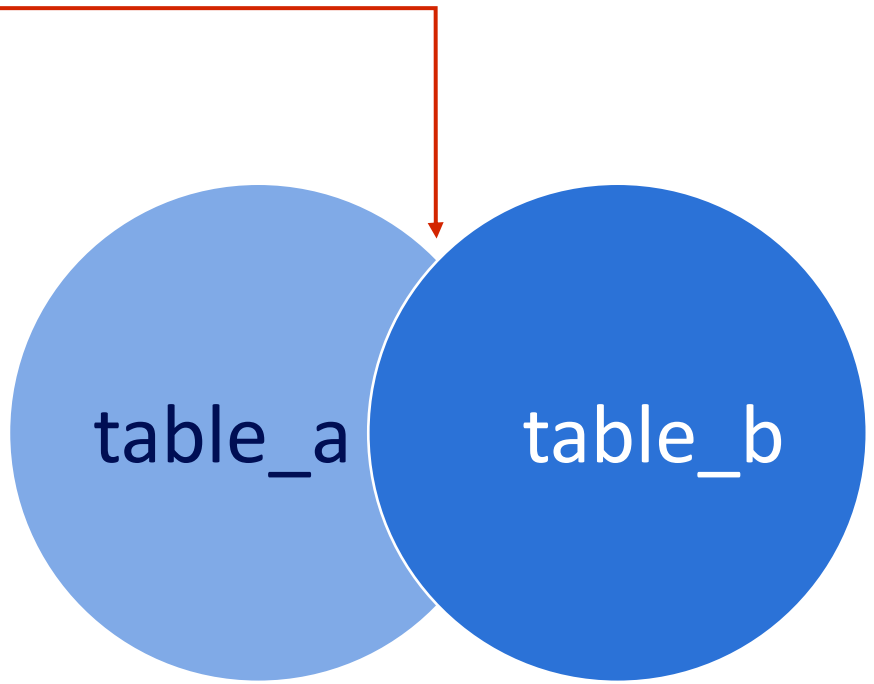
- All rows in table_a are included, including those matching the ON condition.



Joins: Right Outer Join

```
FROM table_a RIGHT JOIN table_b
   ON column_a = column_b
```

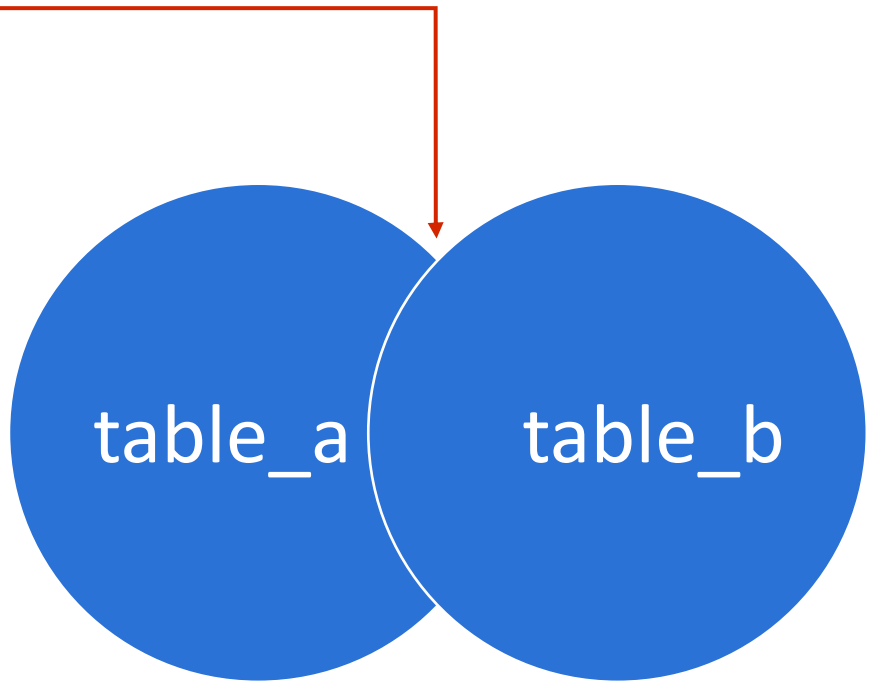
- All rows in table_b are included, including
  those matching the ON condition.
- LEFT and RIGHT determine which table participates in the outer join.



Joins: Full Outer Join

```
FROM table_a FULL JOIN table_b
   ON column_a = column_b
```

- All rows in table_b are included, including
  those matching the ON condition
- Like performing a LEFT and RIGHT join simultaneously



Outer Joins

The End

Demo

Outer Joins

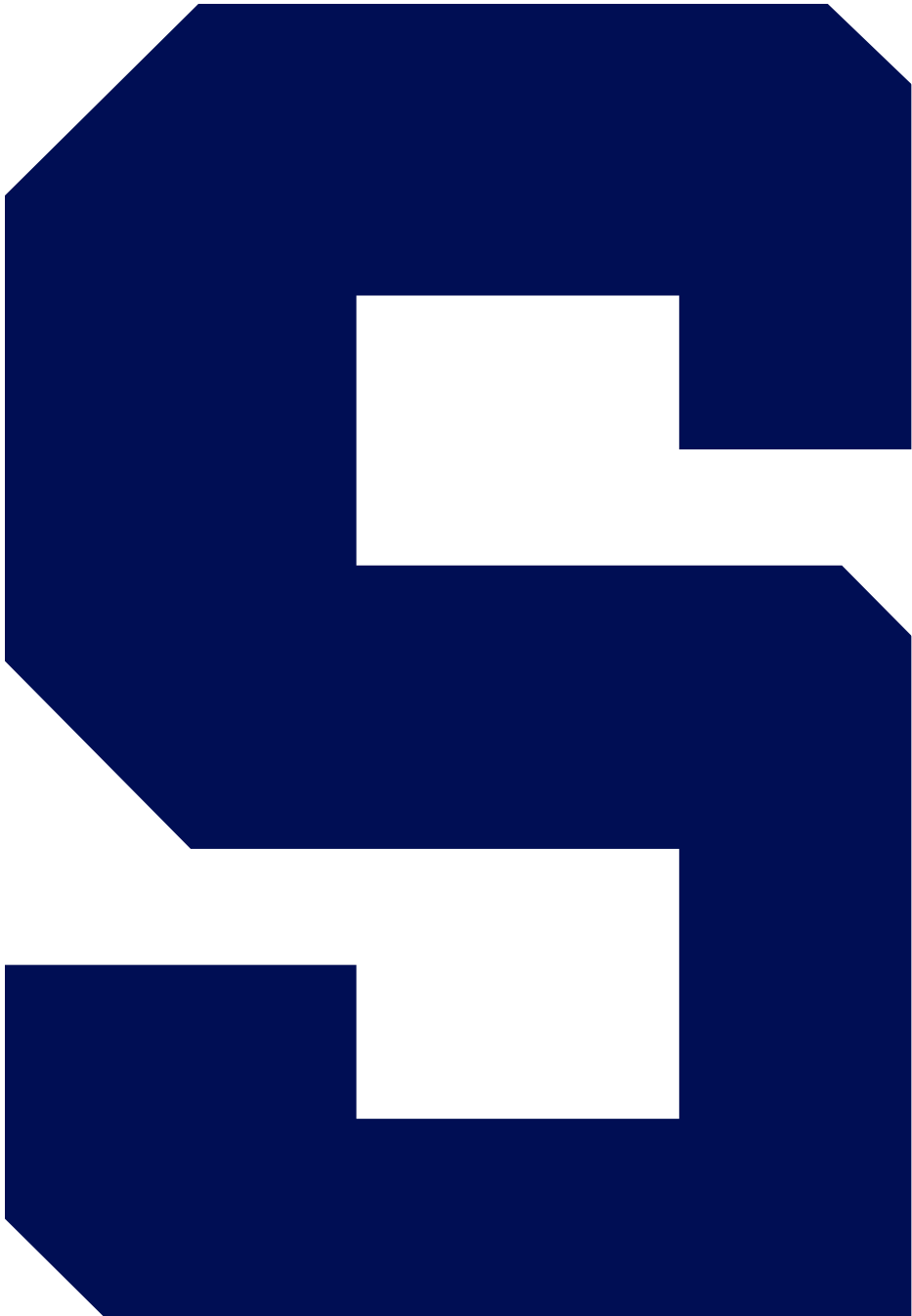
Demo: Outer Joins

- We will use the demo and payroll databases
- Players not on a team
- Teams without players
- Both!
- Null checks in the query to get exceptions, such as only players without teams
- Which employees have not filled out a timecard?

Demo: Outer Joins

The End

Syracuse University
School of Information Studies



Advanced Joins

More Joins

- It is not limited to two table joins. We can join multiple tables.

```
SELECT *
FROM table_a
   JOIN table_b ON table_b_col = table_a_col
   JOIN table_c ON table_c_col = table_a_or_b_col
```

- We can alias tables to simplify complex joins.

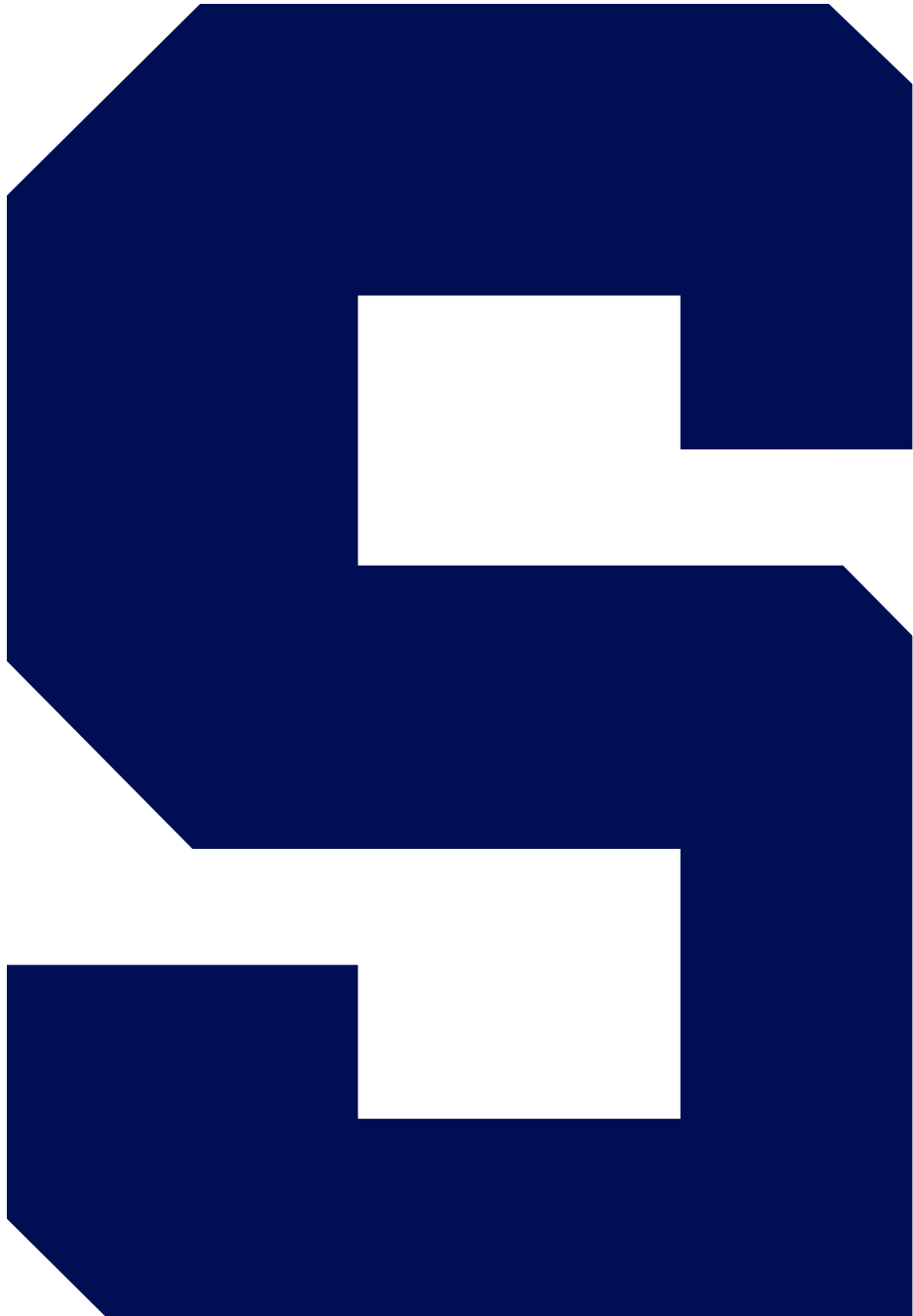
```
SELECT a.*
FROM table_a AS a
   JOIN table_b ON table_b_col = a.table_a_col
```

- We can use table aliasing to join a table to itself!

Advanced Joins

The End

Syracuse University
School of Information Studies



Demo

Multi-Table Joins and Self-Joins

Demo: Multi-Table Joins and Self-Joins

- We will use the payroll database
- Joining three tables in a complex join
- Table aliasing in a complex join
- Joining a table to itself; employees and supervisors

Demo: Multi-Table Joins and Self-Joins

The End

Syracuse University
School of Information Studies

Summary

Summary

- The processing order of the SELECT statement differs from how it is written.
- Projections filter columns.
- Selections filter rows with the WHERE clause.
- The CASE statement is used for conditional evaluation.
- Joins combine tables based on matching columns.
- Outer joins allow for including rows in a table that do not match the inner join output.

Syracuse University
School of Information Studies

Summary

The End