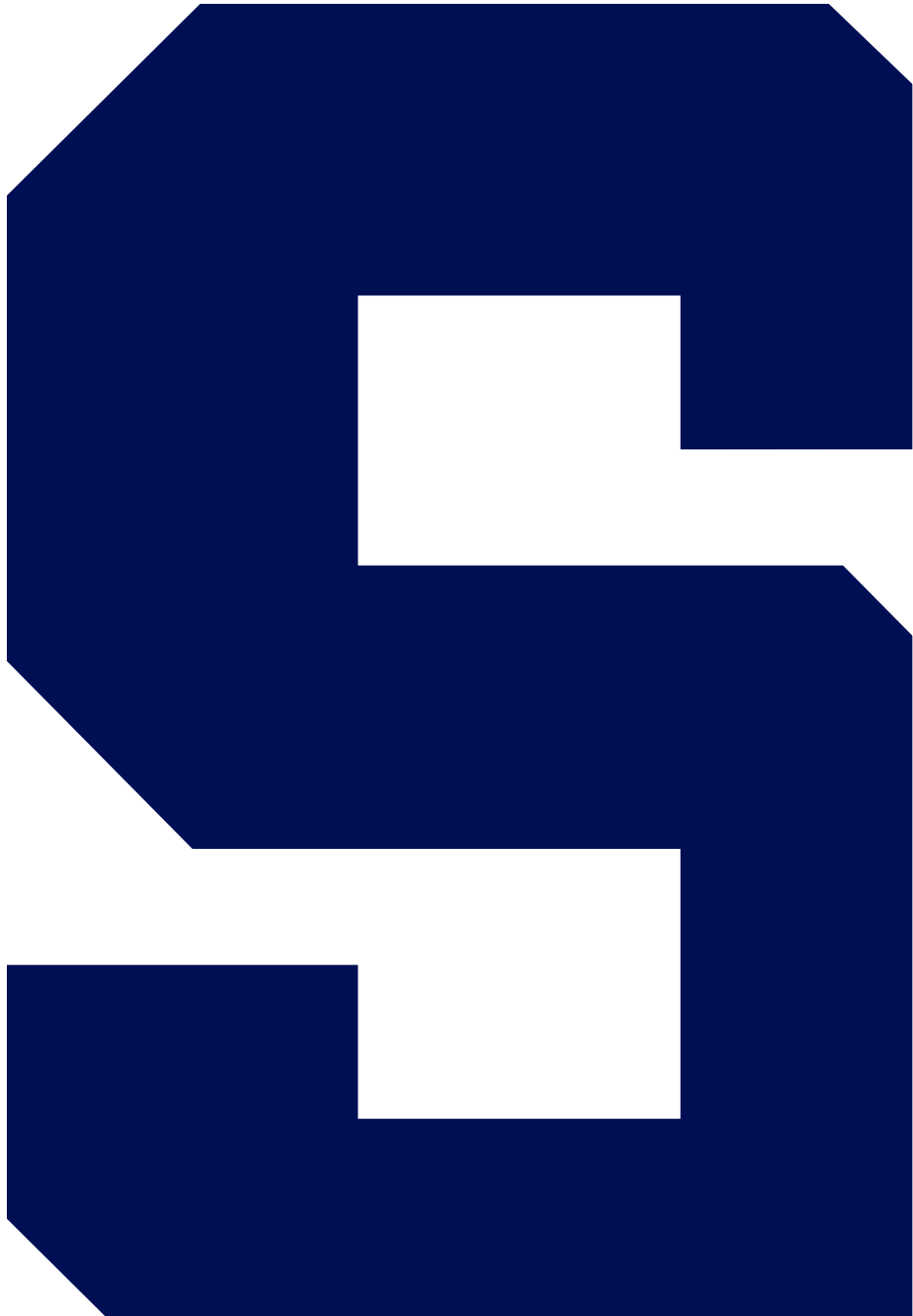


SQL SELECT

Part II

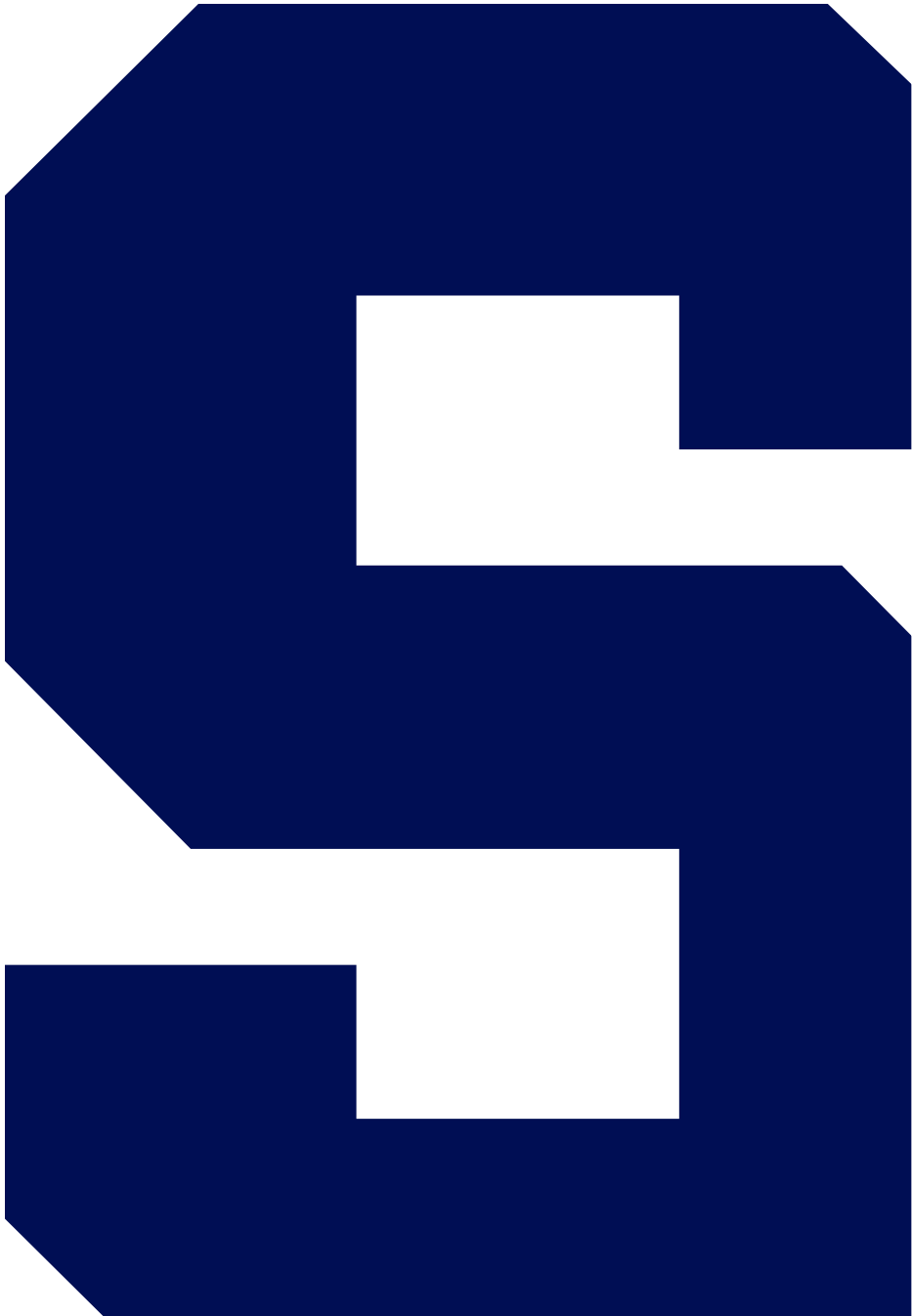
Agenda

- Map vs. reduce operations
- Aggregations: GROUP BY, HAVING, aggregate functions
- Sub-selects and WITH clause
- Window functions: aggregate, value, and ranking
- Views

SQL SELECT: Part II

The End

Syracuse University
School of Information Studies



Maps and Reduces

Maps vs. Reduces

Mapping function

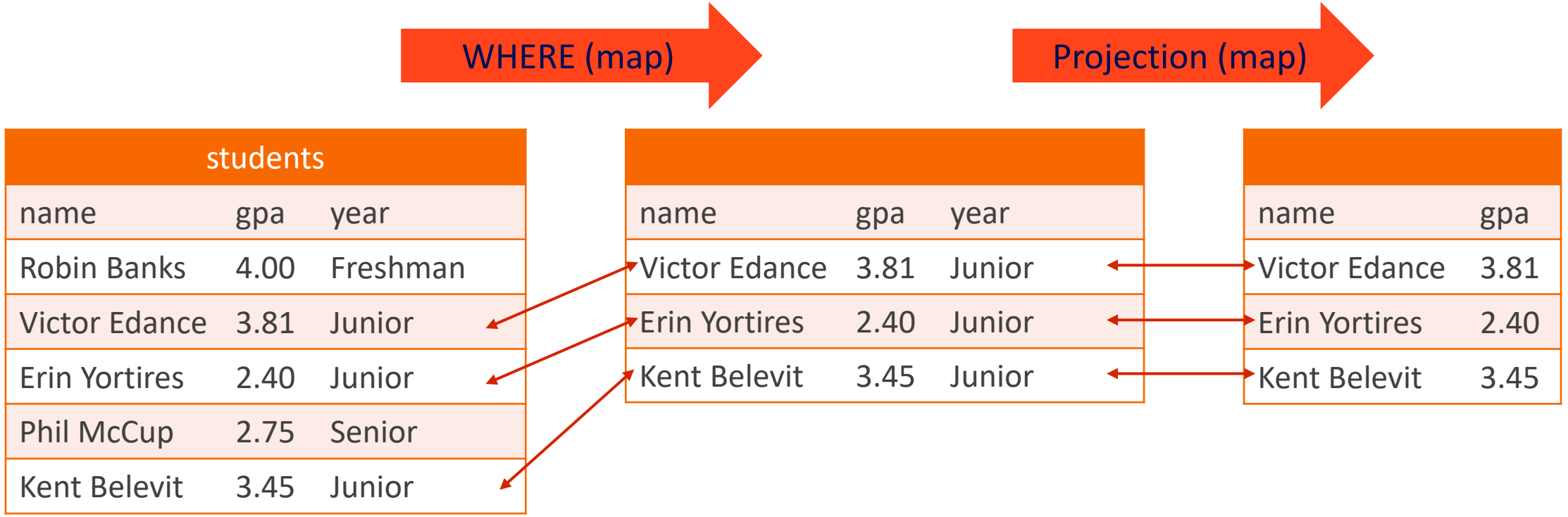
- A mapping function applies to a single row in the query producing a single output.

Reducing function

- A reducing function applies to a set of rows in the query producing a single output.

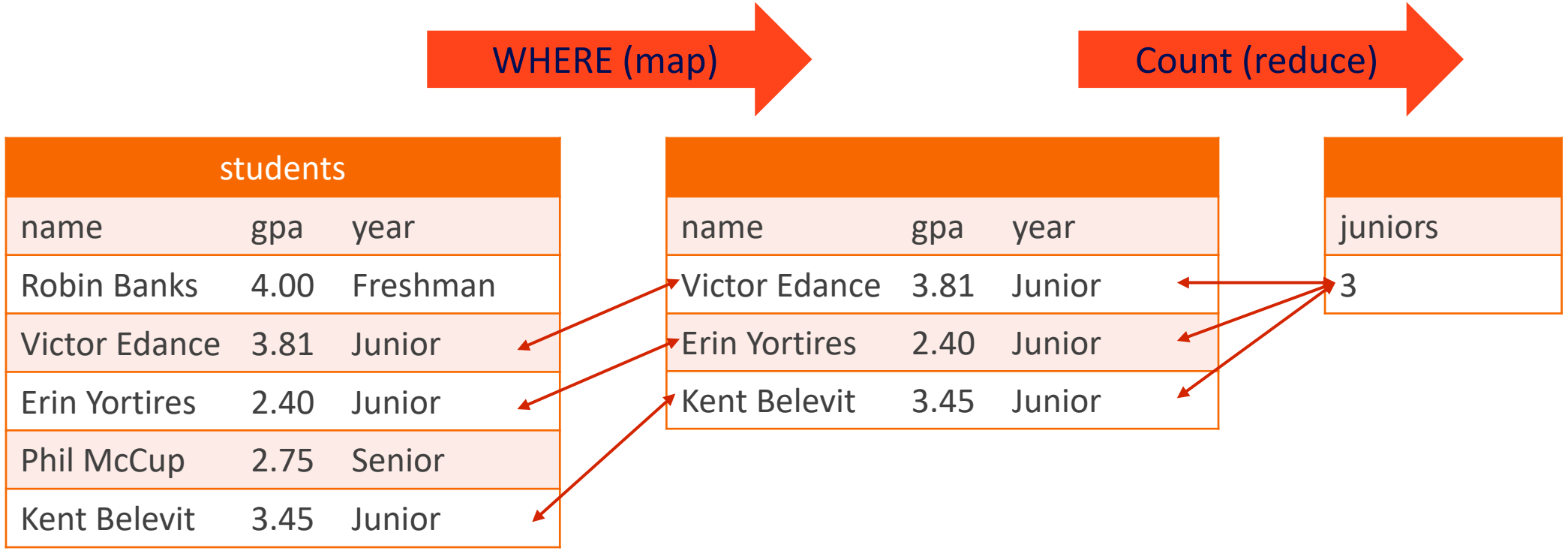
Map Example: SELECT Processing

```
SELECT name, gpa FROM students WHERE year = 'Junior'
```



Reduce Example: SELECT Processing

```
SELECT count(*) as juniors FROM students WHERE year = 'Junior'
```

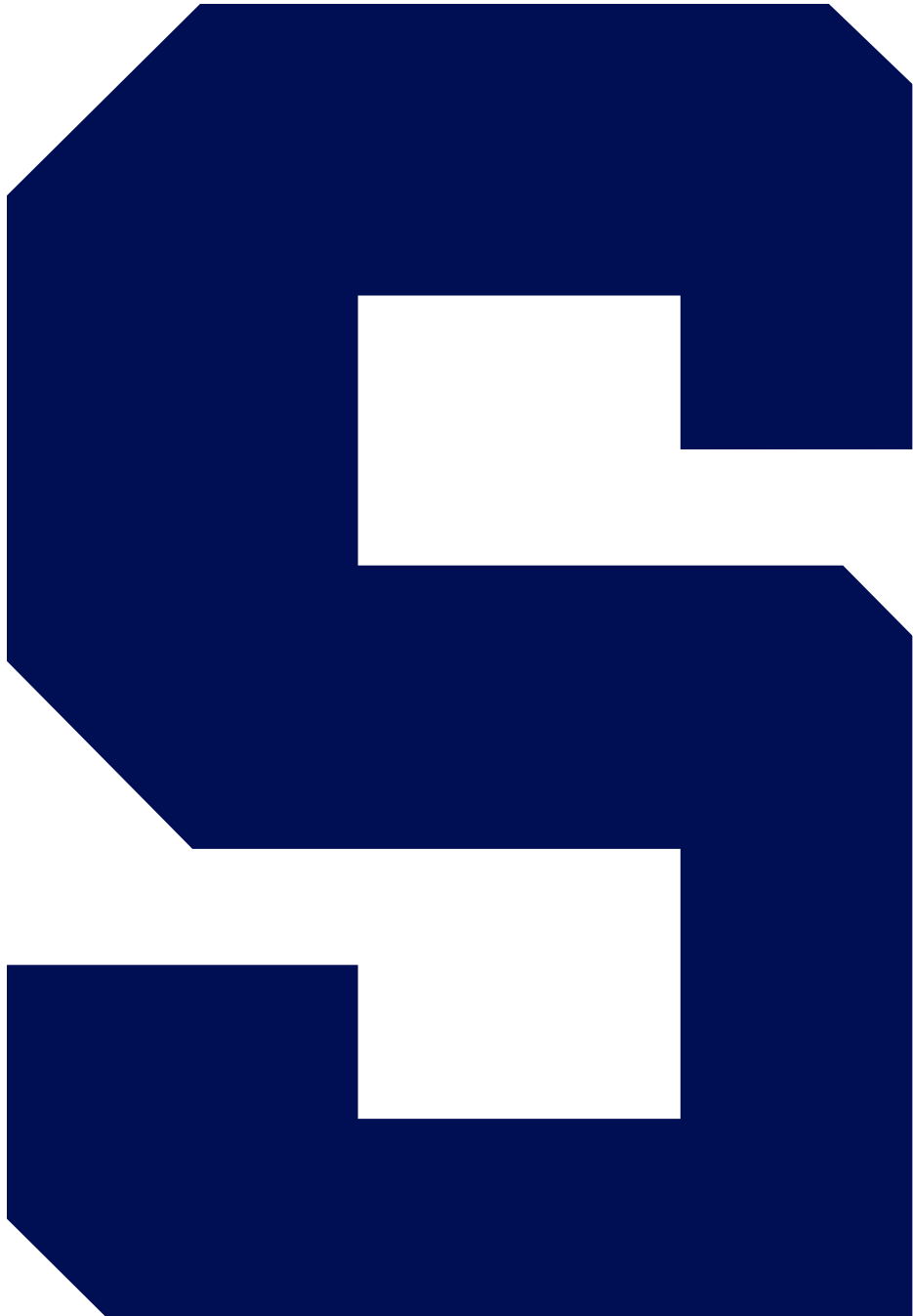


Maps and Reduces

The End

Syracuse University
School of Information Studies

SELECT Revisited



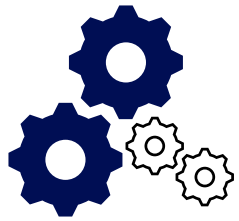
Recall: SQL SELECT

```
SELECT TOP n DISTINCT
    columns
    FROM table
    WHERE condition
    ORDER BY columns
```

SQL SELECT V2.0

```
SELECT TOP n DISTINCT
    columns
    FROM table
    WHERE condition
    GROUP BY columns
    HAVING condition
    ORDER BY columns
```

SELECT Statement Processing V2.0



How we write it

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

How it is processed

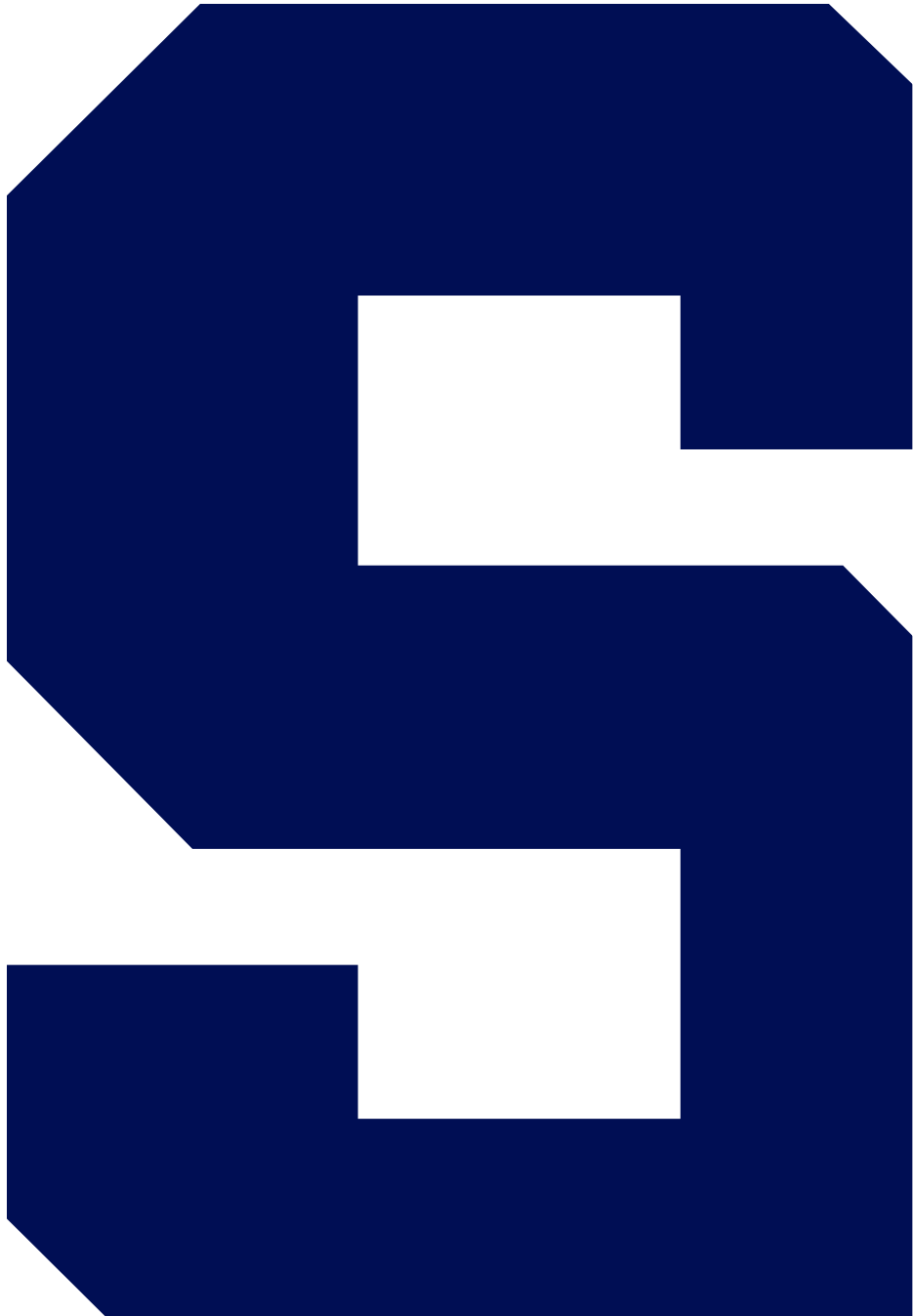
1. FROM
2. WHERE (selection)
3. GROUP BY
4. HAVING
5. (Projection)
6. ORDER BY ← materialized
7. TOP/DISTINCT

Syracuse University
School of Information Studies

SELECT Revisited

The End

Syracuse University
School of Information Studies



SQL Aggregates

SQL Aggregates

- Aggregates allow us to perform a reduce operation over our data.
- The output no longer corresponds to an actual row in the table.
- The query output is now a summary of original rows.

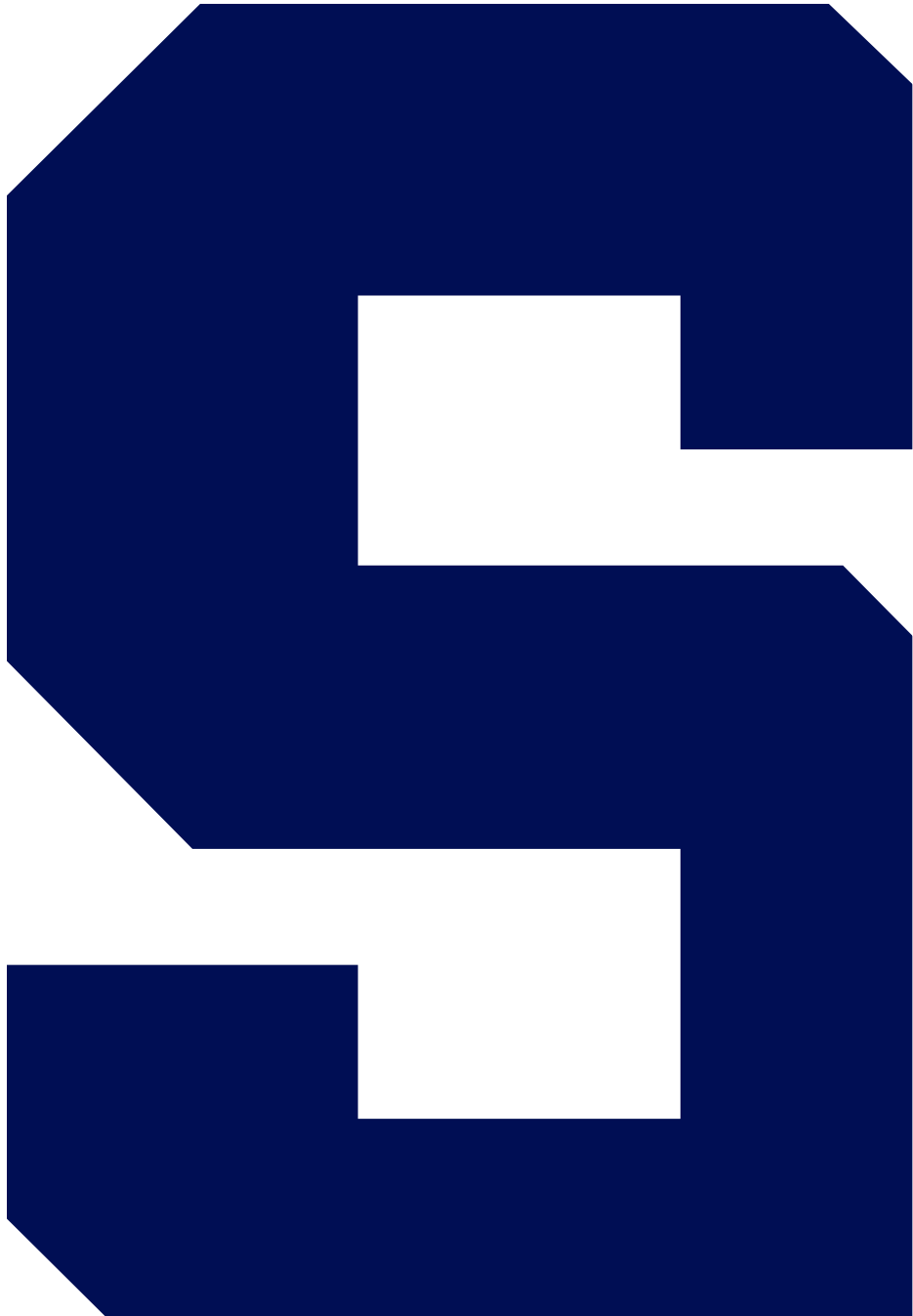
SQL Aggregate Functions

Function	Purpose
`count(*)`	Count rows
`count(column)`	Count non-null values in column
`count(distinct column)`	Count the unique non-null values in column
`min(column)`	Return the smallest value in column
`max(column)`	Return the highest value in column
`avg(column)`	Calculate the average values of column; numeric data types only
`sum(column)`	Calculate the total values of column; numeric data types only

SQL Aggregates

The End

Syracuse University
School of Information Studies



Demo

SQL Aggregates

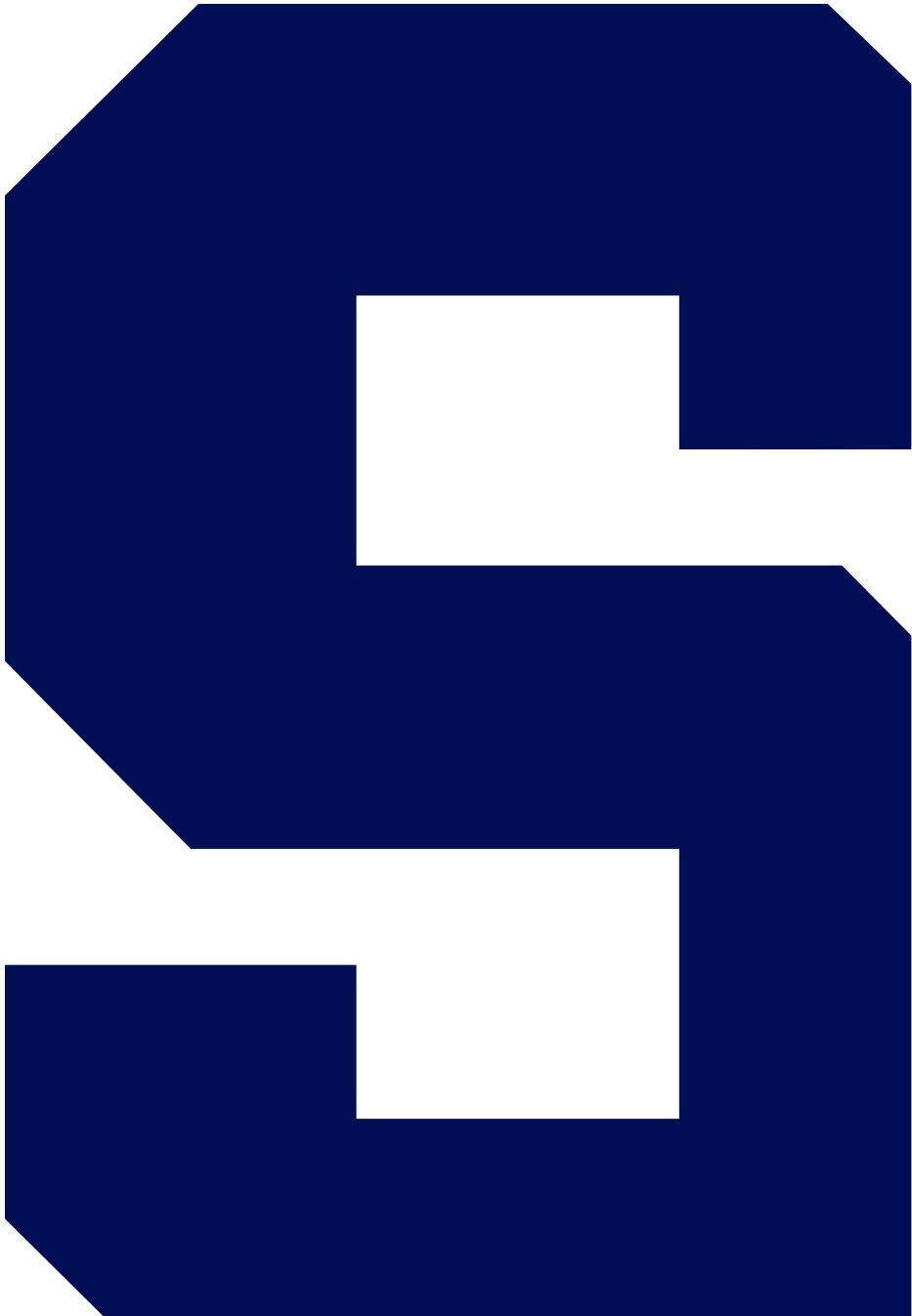
Demo: SQL Aggregate Functions

- We will use the payroll database
- Count employees
- Count departments, count distinct
- Minimum/maximum on different types: What happens?

Demo: SQL Aggregates

The End

Syracuse University
School of Information Studies



GROUP BY and HAVING

GROUP BY

Columns in the projection must aggregate functions or appear in the GROUP BY.

```
1   select employee_department,
2       count(*) as row_count
3   from employees
4   group by employee_department
```

Results Messages

	employee_department	row_count
1	Clothing	5
2	Customer Service	15
3	Electronics	11
4	Hardware	10
5	Housewares	6
6	Sporting Goods	10
7	Toys	10

A Bad GROUP BY!

```
1    select employee_department, employee_id,
2        count(*) as row_count
3    from employees
4    group by employee_department
```

Messages

```
8:47:17 AM    Started executing query at Line 1
              Msg 8120, Level 16, State 1, Line 1
              Column 'employees.employee_id' is invalid
               GROUP BY clause.
```

HAVING

```
1  select employee_department,
2      count(*) as row_count
3  from employees
4  where employee_jobtitle = 'Sales Associate'
5  group by employee_department
6
```

Results | Messages

	employee_department	row_count
1	Clothing	4
2	Customer Service	11
3	Electronics	10
4	Hardware	9
5	Housewares	5
6	Sporting Goods	9
7	Toys	9

```
1  select employee_department,
2      count(*) as row_count
3  from employees
4  where employee_jobtitle = 'Sales Associate'
5  group by employee_department
6  having count(*) >=10
```

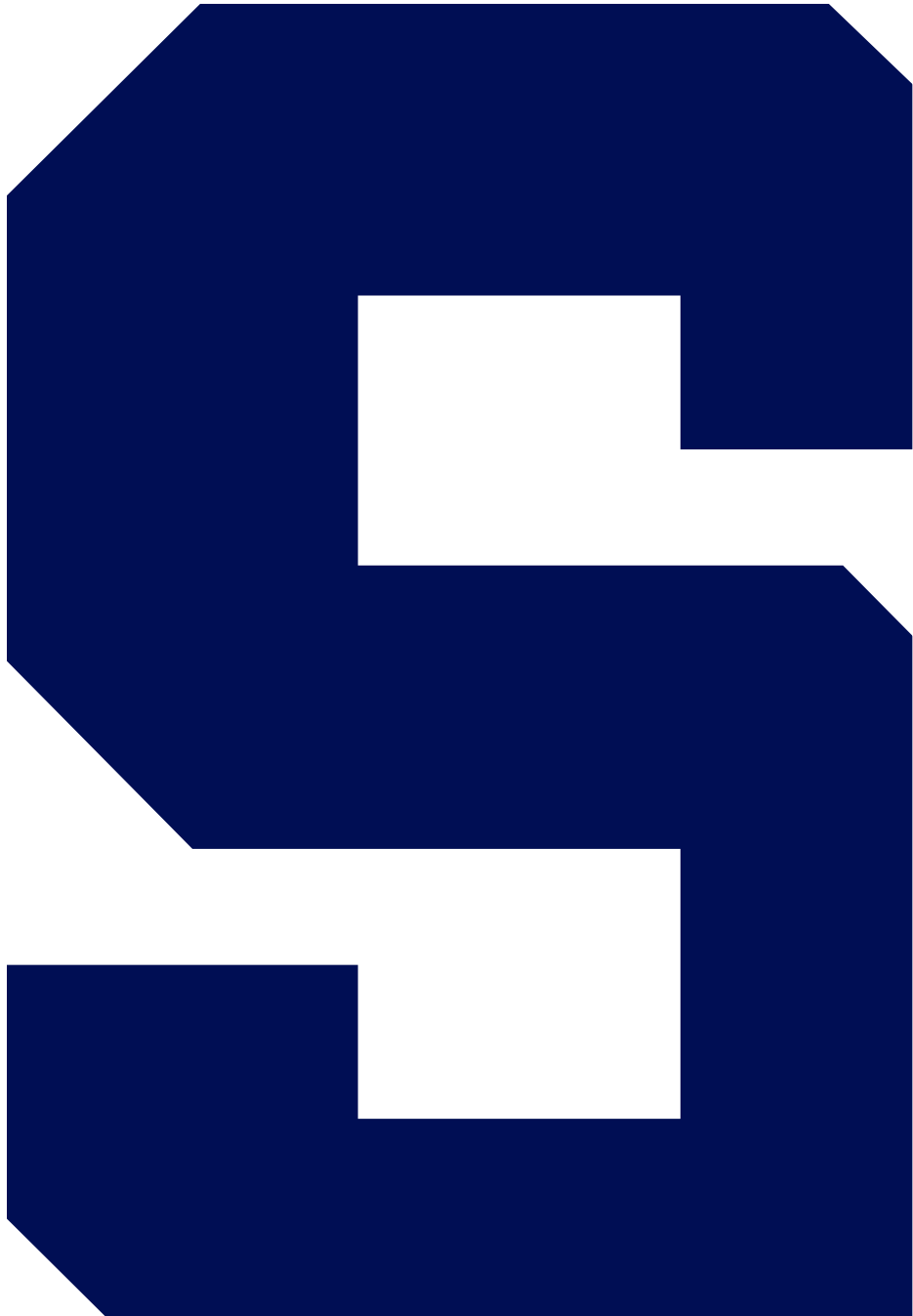
Results | Messages

	employee_department	row_count
1	Customer Service	11
2	Electronics	10

The HAVING clause filters row output post-aggregation.

GROUP BY and HAVING

The End



Demo

GROUP BY and HAVING

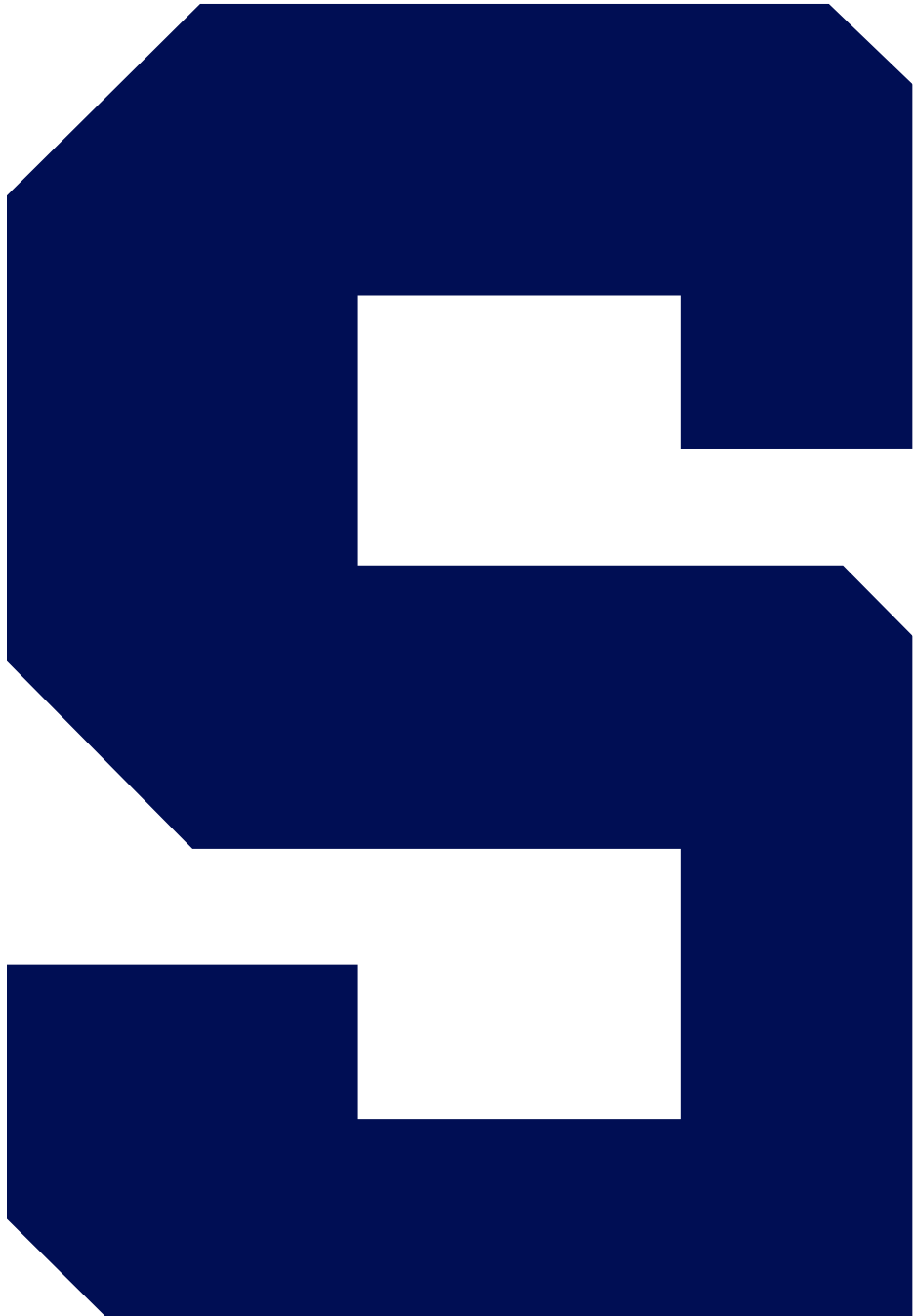
Demo: GROUP BY and HAVING

- We will use the Azure Data Studio application
- We will use the payroll database
- Analyze paychecks in Q1 of 2020 (January to March)
- Total payroll by hourly vs. salary; hours and total pay
- Payroll by department total hours and total pay
- Small departments only

Demo: GROUP BY and HAVING

The End

Syracuse University
School of Information Studies



Common Table Expressions

Sub-Selects

- You may nest SELECT statements in any portion of an existing SELECT
- A SELECT in the WHERE clause to locate a value IN a set
- A SELECT in the FROM clause to create a temporary table
- A SELECT in the PROJECTION to derive a column

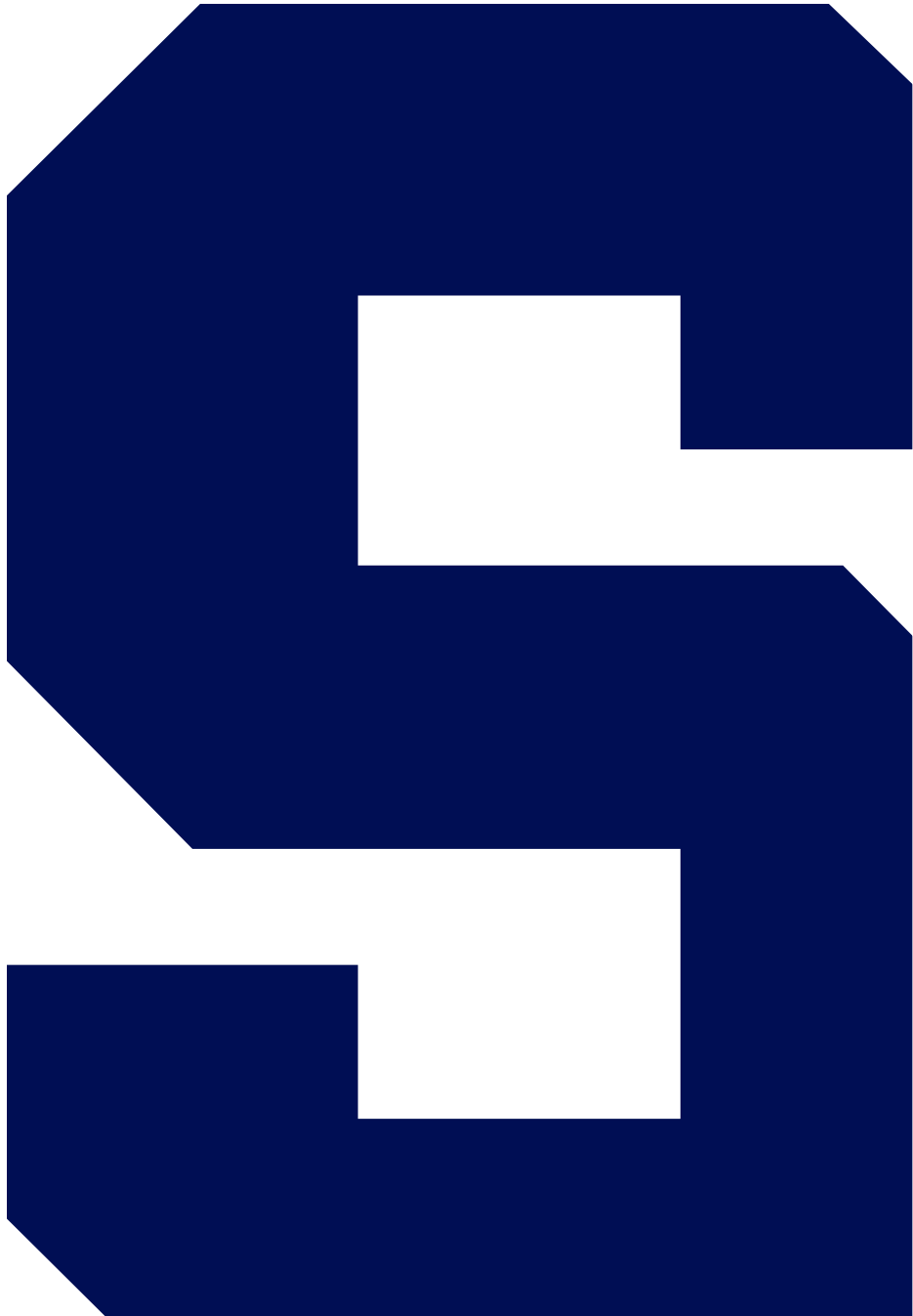
Common Table Expressions

- Common table expressions simplify complex queries using sub-selects.
- The WITH clause allows you to name a query.

```
WITH named_query (columns) AS (
     SELECT-statement
),
another-named_query (columns) AS (
     SELECT-statement
)
SELECT-statement-using-named-queries
```

Common Table Expressions

The End



Demo

Common Table Expressions

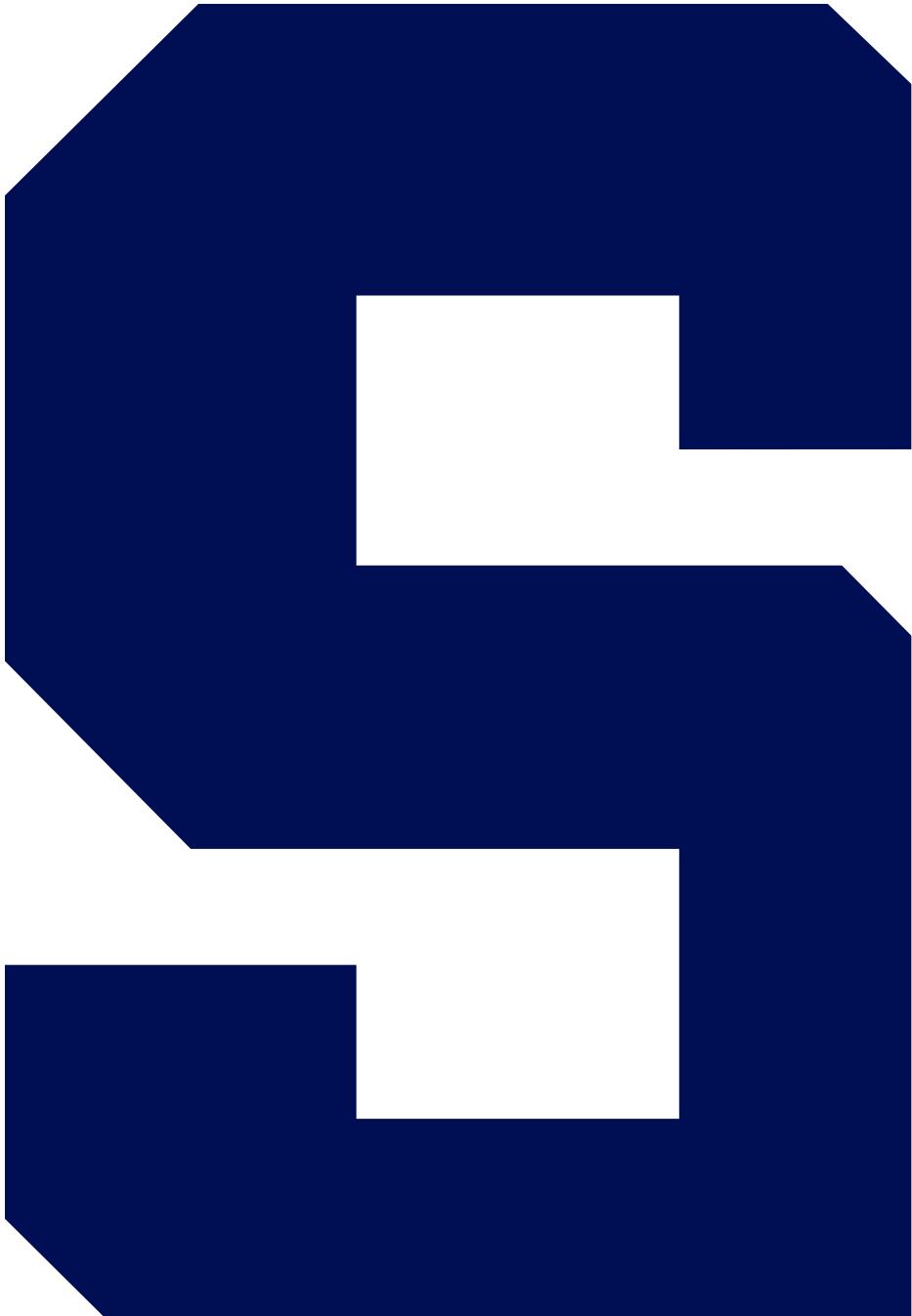
Demo: Common Table Expressions

- We will use the payroll database
- Departments with more employees than the average
- Simplify with clause, CTE
- WITH multiple selects: students and employees

Demo: Common Table Expressions

The End

Window Functions



SQL Window Functions

- A function that takes a set of rows as input and produces a single output for every row: The set of input rows is called the window.
- The set of rows as input are defined by partitions over the data in the FROM clause.
- If the order of the input matters to the window function, the partition can be sorted with ORDER BY.
- The window function acts like a reducer/aggregate function but operates as a map!

Map

```
select name, gpa, year
from students
where year = 'Junior'
```

name	gpa	year
Victor Edance	3.81	Junior
Erin Yortires	2.40	Junior
Kent Belevit	3.45	Junior

students

name	gpa	year
Robin Banks	4.00	Freshman
Victor Edance	3.81	Junior
Erin Yortires	2.40	Junior
Bette Alott	2.00	Freshman
Kent Belevit	3.45	Junior

Reduce / Agg.

```
select
  avg(gpa) as avg_gpa, year
from students
group by year
```

avg_gpa	year
3.00	Freshman
3.22	Junior

Window

```
select rank() over (partition by year
  order by gpa) as rank_by_year, gpa, year
from students
```

rank_by_year	gpa	year
1	4.00	Freshman
2	2.00	Freshman
1	3.81	Junior
2	3.45	Junior
3	2.40	Junior

SQL Window Functions (cont.)

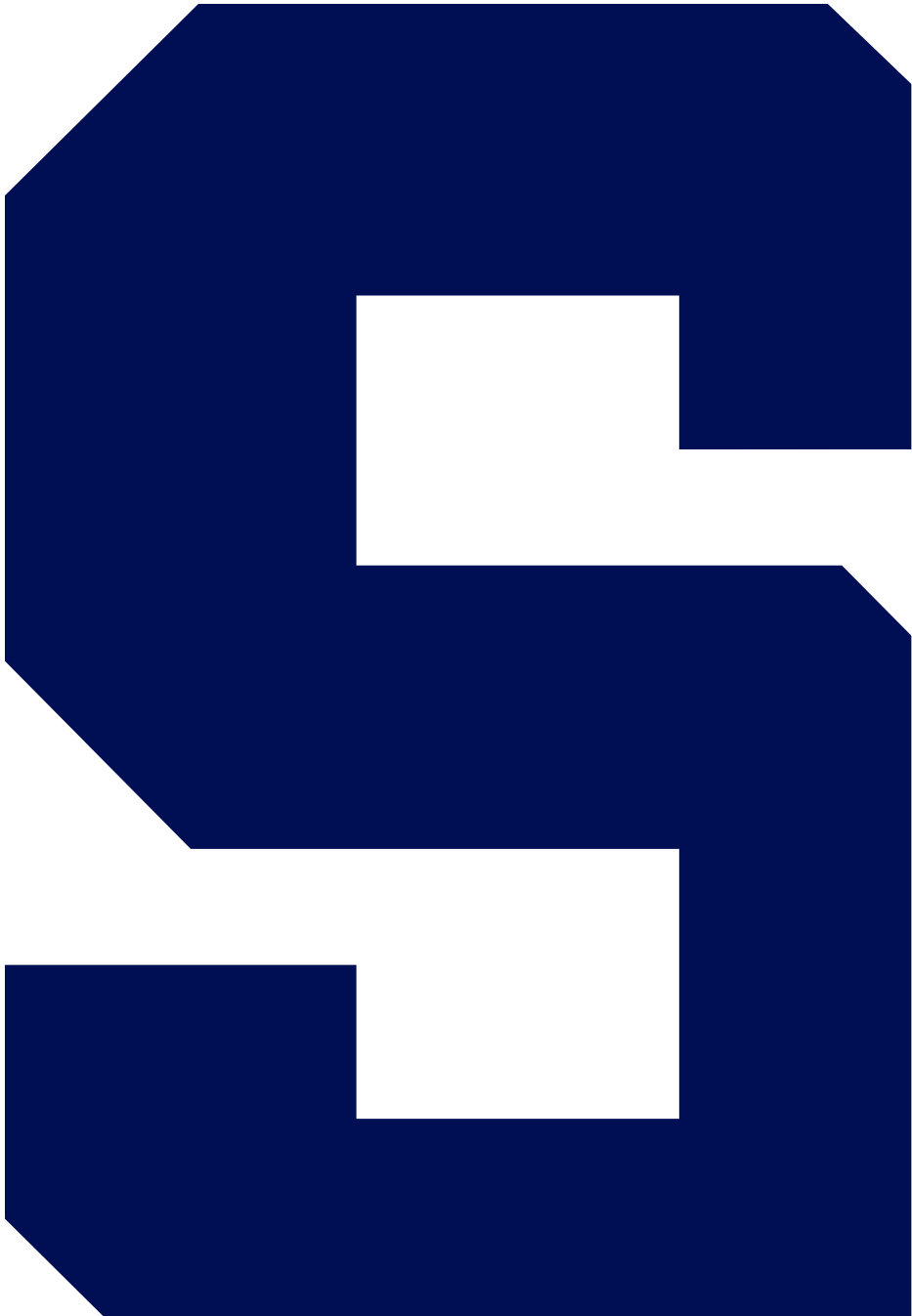
```
window_function()
     OVER (
          [ PARTITION BY columns ]
          [ORDER BY columns]
     )
```

- Three types of window functions
  1. Aggregate functions return a summary value for the included set of rows.
  2. Value functions return a specific value from the included set of rows.
  3. Ranking functions return an ordinal position for each value in the set of rows.

Window Functions

The End

Syracuse University
School of Information Studies



Aggregate Window Functions

Aggregate Window Functions

These functions return a summary value over the partition. The same value is returned for each row within the partition.

Function	Purpose
`count(*)`	Count rows over the partition
`count(`*`column`*`)`	Count non-null values in column over the partition
`min(`*`column`*`)`	Return the smallest value in column over the partition
`max(`*`column`*`)`	Return the highest value in column over the partition
`avg(`*`column`*`)`	Calculate the average values of column over the partition; numeric data types only
`sum(`*`column`*`)`	Calculate the total values of column over the partition; numeric data types only

Aggregate Window Functions

The End

Syracuse University
School of Information Studies

Demo

Aggregate Window Functions

Demo: Aggregate Window Functions

- We will use the payroll database
- Analysis of sales associate pay rates
- Include average by department
- Include overall average
- Calculate differences—it gets ugly!
- WITH clause to the rescue!

Demo: Aggregate Window Functions

The End

Value Window Functions

Value Window Functions

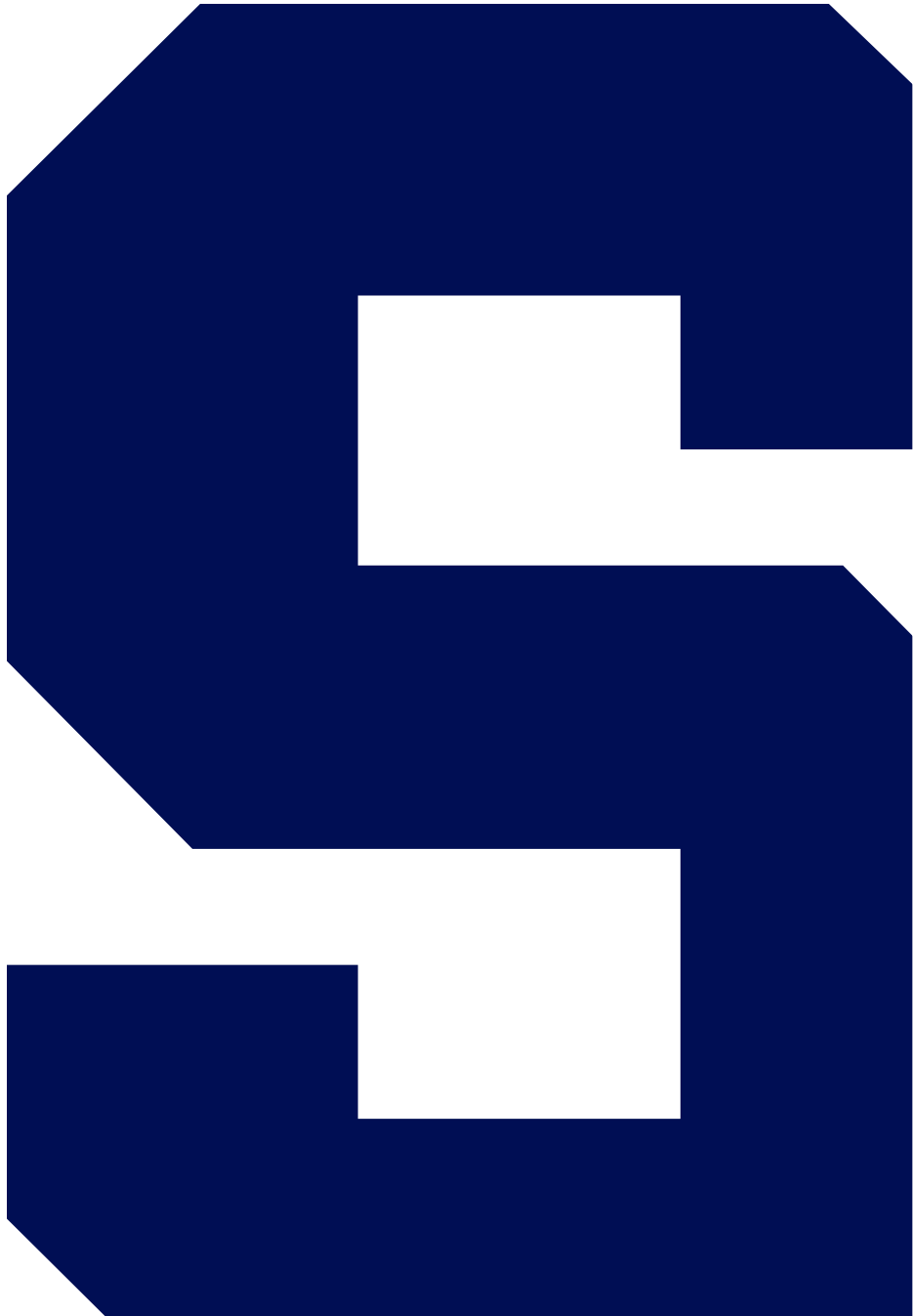
These functions retrieve a specific value from the set of values in the partition. Ordering within the partition is required, ORDER BY.

Function	Purpose
`lag(`*`column [, offset`*`])`	Returns the value from the previous row, within the partition, relative to the current row; offset can be provided to return the third or fourth previous row, for example
`lead(`*`column [, offset`*`])`	Returns the value from the next row, within the partition relative to the current row; offset can be provided to return the third or fourth next row, for example
`first_value(`*`column`*`)`	Returns the first value within the partition
`last_value(`*`column`*`)`	Returns the last value within the partition

Value Window Functions

The End

Syracuse University
School of Information Studies



Demo

Value Window Functions

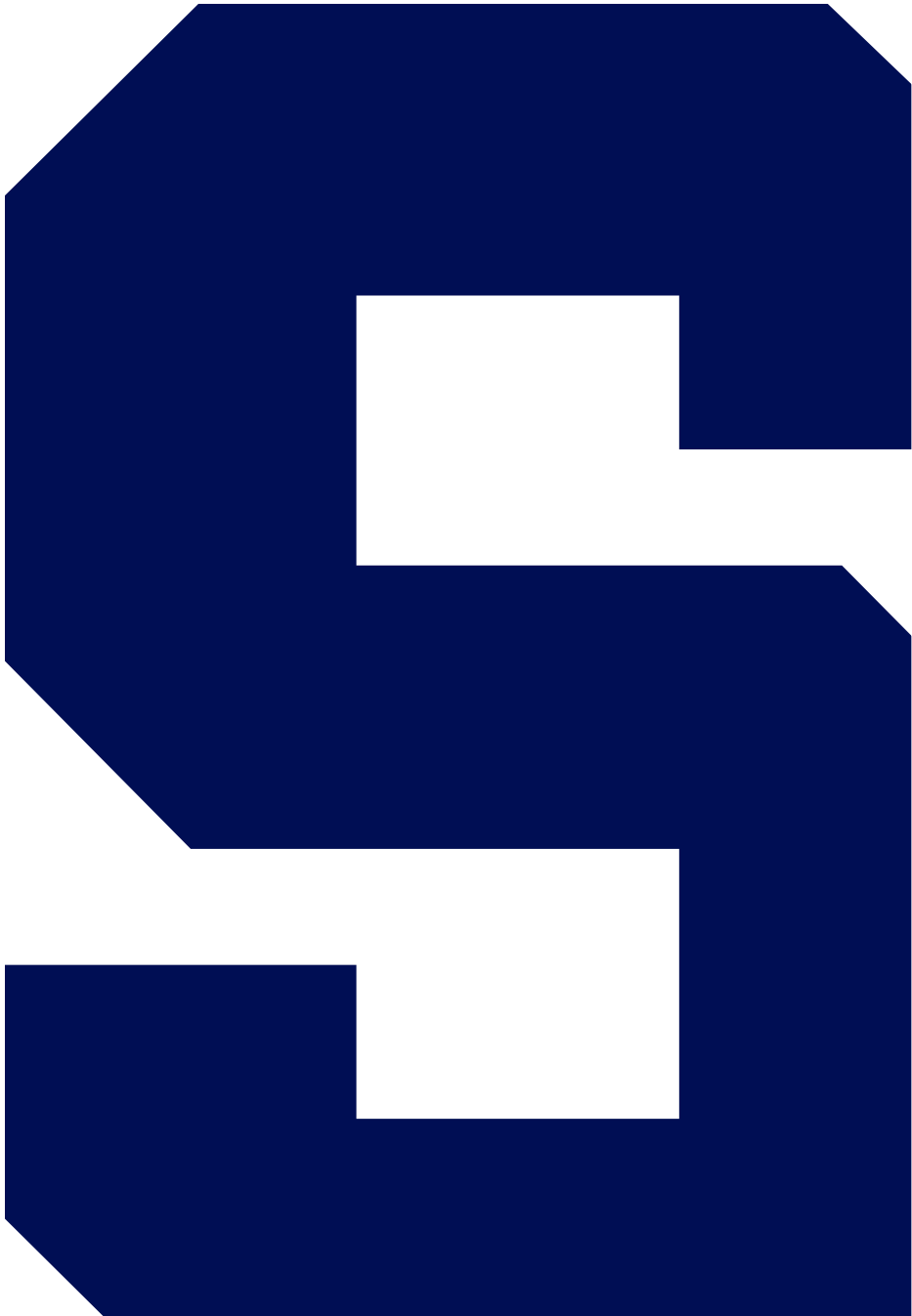
Demo: Value Window Functions

- We will use the payroll database.
- Look at employee paychecks from 2020 Q1.
- Show first paycheck and current paycheck.
- Add last paycheck.
- Add paycheck from two pay periods prior.

Demo: Value Window Functions

The End

Ranking Window Functions



Ranking Window Functions

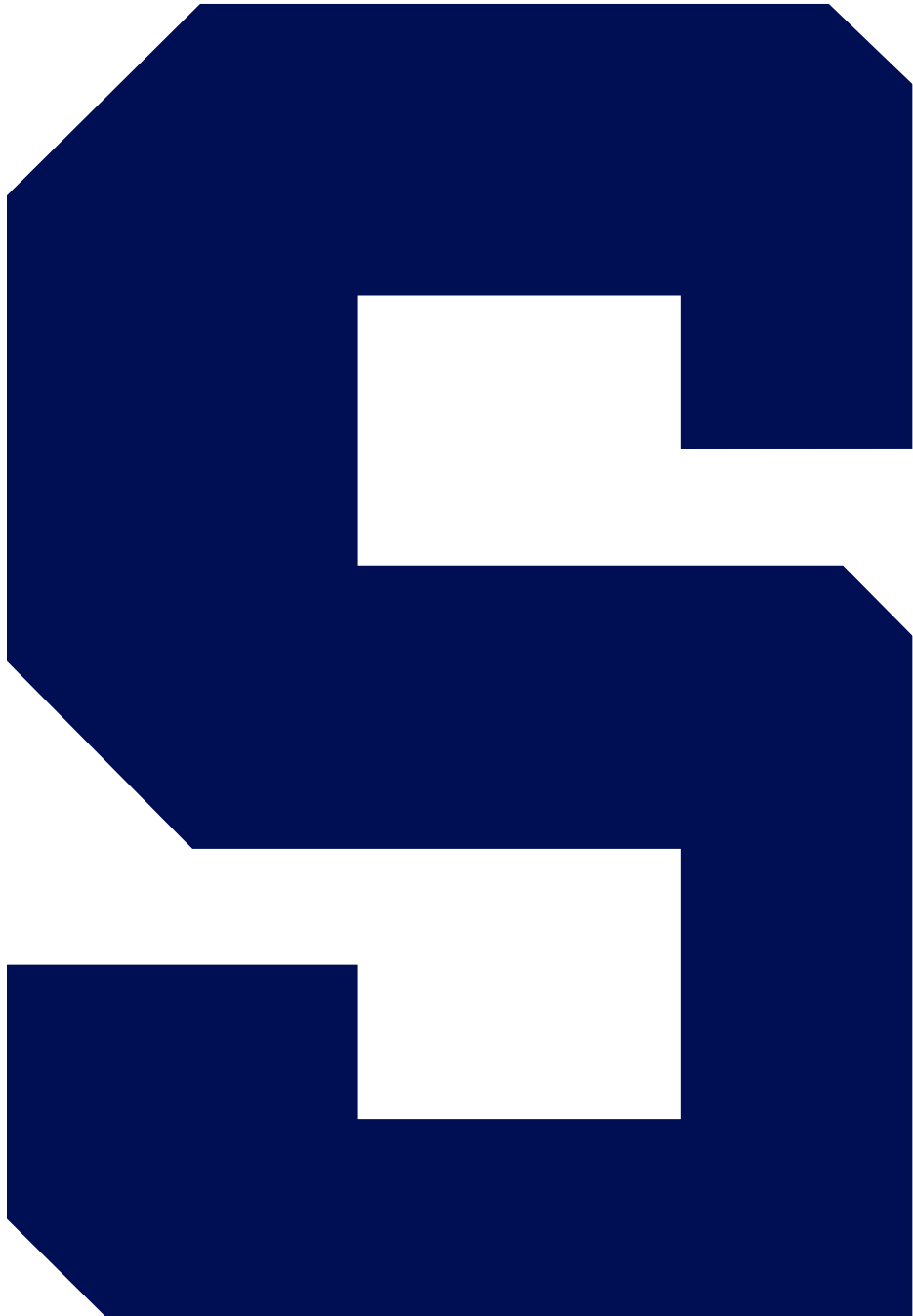
These functions retrieve an ordinal position from the set of values in the partition. In all cases, ORDER BY is required over the partition.

Function	Purpose
`row_number()`	Determines the ordinal number of the row over the ordered partition of values; each row will have a unique value in the partition
`rank()`	Returns the rank of a value within the ordered partition of values; similar to row_number() except that, if one or more values have the same rank, the value is shared; for example, if the first two values are the same they have rank 1, the next rank is 3
`dense_rank()`	Same as rank(), but there are no gaps in the numbers; for example, if the first two values have rank 1, the next rank is 2
`ntile(`*`number`*`)`	Divides the total number of rows in the ordered partition into the number of equal partitions, assigning each a value from 1 to number
`percent_rank()`	Calculates the percent rank over the ordered partition using the following formula: (rank -1) / (total rows in partition -1); for example the third row in a partition of five rows would be 0.5
`cume_dist()`	Calculates the cumulative distribution over the ordered partition using the formula: (current row) / (total rows in partition); for example the third row in a partition of five rows would be 0.6

Ranking Window Functions

The End

Syracuse University
School of Information Studies



Demo

Ranking Window Functions

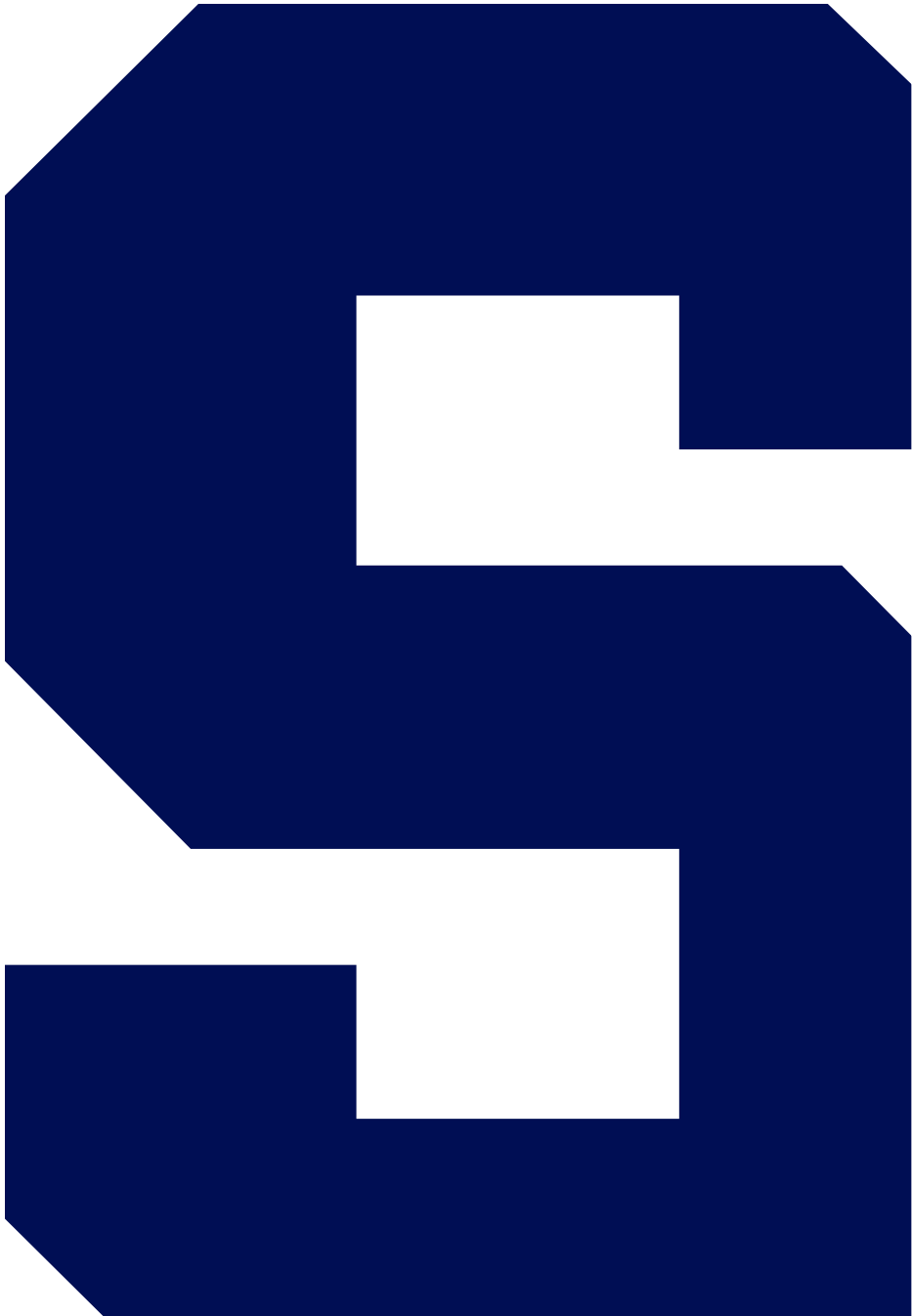
Demo: Ranking Window Functions

- We will use the Azure Data Studio application.
- We will use the payroll database.
- Let’s try out the different ranking functions to understand them.

Demo: Ranking Window Functions

The End

Syracuse University
School of Information Studies



SQL Views

SQL Views

- A View is a virtual table based on an SQL SELECT statement
- The SQL SELECT statement is saved to the database under a name
- This makes a view an object/metadata
- It is managed with the DDL language
- CREATE/ALTER/DROP
- Dropping a view deletes the SQL SELECT statement but not the underlying data

Views: Data Definition Language

```
create view view_name as
select ...
alter view view_name as
select ...
drop view view_name
```

To see the views in the database including their definitions, you can use INFORMATION_SCHEMA:

```
select * from INFORMATION_SCHEMA.VIEWS
```

Using Views

- To query a view, treat it like a table.

```
SELECT * FROM view_name
```

- DELETE, UPDATE, and INSERT can be used on views that do not violate data-integrity constraints.
- Views using JOIN, GROUP BY, or WITH clauses are considered SELECT-only.

SQL Views

The End

Syracuse University
School of Information Studies

Demo

Views

Demo: Views

- We will use the Azure Data Studio application.
- We will use the payroll database.
- Let’s create a payroll view.
- Execute the view as is if it were a table.
- Find the view in the database with INFORMATION_SCHEMA.VIEWS.
- Drop the view, gracefully.

Syracuse University
School of Information Studies

Demo: Views

The End

Syracuse University
School of Information Studies

Summary

Summary

- Aggregate functions and GROUP BY allow us to summarize row output.
- The WITH statement can simplify complex queries that involve multiple SELECT statements.
- Window functions allow us to apply a function to a set of data, yielding a single result per row, rather than a summarization.
- SQL Views can be used to save a query by name into the database. It is a DDL command.

Syracuse University
School of Information Studies

Summary

The End