# CIS560

Single-Table Queries - Part 2

## Topics

- HAVING
- ORDER BY
- DISTINCT
- TOP
- OFFSET…FETCH…
- Logical Processing Order

## HAVING Element

- Provides a post-grouping filter
- Like WHERE, accepts any boolean expression
- Aggregated computations can be used in the filter

# ORDER BY Element

- Provides ability to sort the rows of the result set
  - Useful for presentation, such as in a report or ad-hoc query
  - Useful for some data processing or loading algorithms
- Ascending and descending sort orders are supported
  - Optional ASC or DESC keywords can follow each expression sorted
  - ASC is the default behavior

# SELECT Statement Processing Order

- Major elements of SELECT

- ANSI Processing Order (Logical)

```
5   SELECT ...
1   FROM ...
2   WHERE ...
3   GROUP BY ...
4   HAVING ...
6   ORDER BY ...
```

## SELECT DISTINCT

- Guarantees uniqueness in result
- All columns of the result are evaluated to remove duplicates
- Like with aggregates, ALL is the default if DISTINCT not specified
- The result is a true set with unique tuples

## TOP Filter

- Filters rows based on ordering
- Accepts a numeric expression

```
TOP (expression) [PERCENT] [ WITH TIES ]
```

  - PERCENT: The expression defines the TOP N% of rows to return.
  - WITH TIES: Allows additional rows with same value as the last row.
- TOP is non-standard

## OFFSET-FETCH Filter

- Like TOP, filters based on ordering
- Unlike TOP:
  - It is standard SQL
  - Supports an offset
- Syntax

```
OFFSET <int. expr> { ROW | ROWS }
[FETCH {FIRST | NEXT} <int. expr> {ROW | ROWS} ONLY ]
```

- Gives options for readability
  - 1 ROW vs. 2 ROWS
  - FETCH FIRST 100 vs. FETCH NEXT 100

## Review

- ANSI Processing Order (Logical)

```
                    6           8
5   SELECT [DISTINCT | TOP]...
1   FROM ...
2   WHERE ...
3   GROUP BY ...
4   HAVING ...
7   ORDER BY ...
    OFFSET … FETCH …
```

- OFFSET-FETCH is part of the ORDER BY clause

## Syntax

```
SELECT [ ALL | DISTINCT ] [TOP ( expression ) [PERCENT] [ WITH TIES ] ]
  < select_list >
[ FROM { <table_source> } [ ,...n ] ]
[ WHERE <search_condition> ]
[ GROUP BY { column_expression } [ ,...n ] ]
[ HAVING < search_condition > ]
[
  ORDER BY { order_by_expression [ ASC | DESC ] } [ ,...n ]
  [
    OFFSET { offset_count_expr } { ROW | ROWS }
    [ FETCH { FIRST | NEXT } { fetch_count_expr } { ROW | ROWS } ONLY ]
  ]
]
```

## Examples

```
SELECT DISTINCT YEAR(O.OrderDate) AS OrderYear,
   O.CustomerID
FROM Sales.Orders O
ORDER BY OrderYear ASC;
```

```
SELECT TOP(2)
   YEAR(O.OrderDate) AS OrderYear,
   COUNT(*) AS OrderCount,
   MIN(O.OrderDate) AS FirstOrderDate,
   MAX(O.OrderDate) AS LastOrderDate
FROM Sales.Orders O
GROUP BY YEAR(O.OrderDate)
ORDER BY OrderCount DESC;
```

```
SELECT O.OrderID, O.OrderDate, O.CustomerID
FROM Sales.Orders O
ORDER BY O.OrderID ASC
OFFSET 1000 ROWS FETCH NEXT 1000 ROWS ONLY;
```

## Questions?