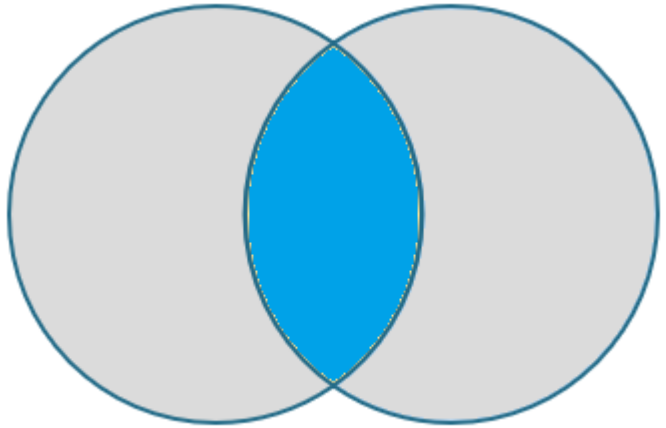
Joining tables: INNER JOIN

- The columns from two tables where there is a match on a key
- Syntax



SELECT
<table1>.<col1>,<table2>.<col2>
FROM <table1>

[INNER] JOIN <table2>

ON <table1>.<col1> = <table2>.<col1>

Old join syntax Comma Join (DON’T USE)

```
SELECT
Col1, Col1
FROM table1, table2
Where table1.col1 = table2.col1
```

INNER JOIN

Customer	
CustomerID (Primary Key)	**Name**
1	John
2	Sharon
3	Dana
4	Fox

Sale		
SaleID	**CustomerID (Foreign Key)**	**Amt**
1	3	100
2	1	200
3	3	75
4	3	90
5	1	100

Query results			
SaleID	**CustomerID**	**Name**	**Amt**
1	3	Dana	100
2	1	John	200
3	3	Dana	75
4	3	Dana	90
5	1	John	100

Demo: INNER JOIN

DEMO NOTES

- 5-1 to 5-5

Lab: Joining Tables with INNER JOIN

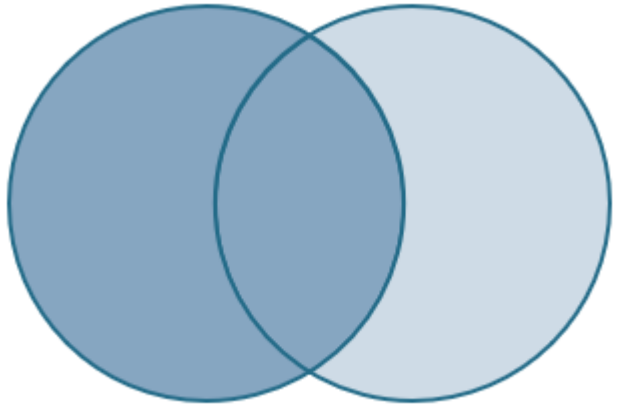
- Exercise 5-1, questions 1 - 3 on page 1

```
SELECT a.col1, a.col2, b.col3
FROM tableA AS A
INNER JOIN tableB AS B
ON a.col1 = b.Col1
```

Joining tables: OUTER JOIN

- All the rows from one table even if they don’t match
- LEFT, RIGHT, FULL
- Syntax

```
SELECT <table1>.<col1>, <table2><col2>
FROM <table1>
LEFT [OUTER] JOIN <table2>
ON <table1>.<col1> = <table2>.<col2>
```



OUTER JOIN

Customer	
CustomerID	**Name**
1	John
2	Sharon
3	Dana
4	Fox

Sale		
SaleID	**CustomerID**	**Amt**
1	3	100
2	1	200
3	3	75
4	3	90
5	1	100

Query results			
SaleID	**CustomerID**	**Name**	**Amt**
1	3	Dana	100
2	1	John	200
3	3	Dana	75
4	3	Dana	90
5	1	John	100
NULL	2	Sharon	NULL
NULL	4	Fox	NULL

LEFT OUTER JOIN with RIGHT Filter

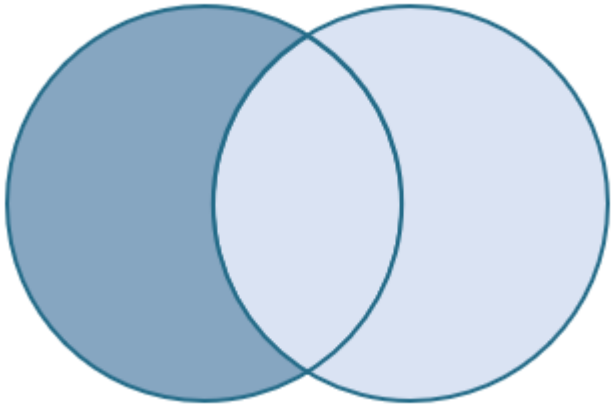
- Use to find rows that don’t match
- Filter on a key from the table on the right
- Syntax

SELECT <table1>.<col1>,<table2>.<col2>

FROM <table1>

LEFT [OUTER] JOIN <table2>

ON <table1>.<col1 > = <table2>.<col1>
WHERE <table2>.<col1> IS NULL



OUTER JOIN with Right table filter

Customer	
CustomerID	**Name**
1	John
2	Sharon
3	Dana
4	Fox

Sale		
SaleID	**CustomerID**	**Amt**
1	3	100
2	1	200
3	3	75
4	3	90
5	1	100

Query results			
SaleID	**CustomerID**	**Name**	**Amt**
NULL	2	Sharon	NULL
NULL	4	Fox	NULL

OUTER JOIN with Right table filter Amt > 100

Customer	
CustomerID	**Name**
1	John
2	Sharon
3	Dana
4	Fox

Sale		
SaleID	**CustomerID**	**Amt**
1	3	100
2	1	200
3	3	75
4	3	90
5	1	100

Query results			
SaleID	**CustomerID**	**Name**	**Amt**
NULL	2	Sharon	NULL
NULL	4	Fox	NULL
2	1	John	200
NULL	3	Dana	NULL

```
--Lose the customers with no sales
SELECT s.SaleID, C.CustomerID, C.Name, s.Amt
FROM Customer AS C
LEFT OUTER JOIN Sales AS S
ON C.CustomerID = S.CustomerID
WHERE S.Amt > 100
--Returns customers with no sales or Amt > 100
SELECT s.SaleID, C.CustomerID, C.Name, s.Amt
FROM Customer AS C
LEFT OUTER JOIN Sales AS S
ON C.CustomerID = S.CustomerID
   AND S.Amt > 100
```

Demo: OUTER JOIN

DEMO notes

- 5-7 to 5-11

Lab: OUTER JOIN

- Exercise 5-2, questions 1, 2, 3 on page 118
- Note that three tables are involved in the query

```
SELECT A.Col1, B.Col2, C.Col3
FROM tableA AS A
LEFT OUTER JOIN tableB as B
        On A.Col1 = B.Col1
LEFT OUTER JOIN TableC as C
        ON C.Col2 = B.Col2
```

IN Subquery

- Use a query to generate a list for the WHERE clause

SELECT <column list>
FROM <schema>.<table1>
WHERE <col> IN (SELECT <col> FROM <schema>.<table2>)

SELECT <column list>
FROM <schema>.<table1>
WHERE <col> NOT IN (SELECT <col> FROM <schema>.<table2>)

DEMO: IN Subquery

LAB: IN Subquery

- Exercise 6-1, questions 1 & 2 on Page 135

```
SELECT Col1, Col2
FROM TableA
WHERE Col1 IN (SELECT Col1 FROM TableB);
```

UNION

- Combine the results of two queries
- Rules
  - Same number of columns
  - Compatible data types
  - Names from first query
  - ORDER BY at end
- Also UNION ALL, EXCEPT, and INTERCEPT

UNION – Eliminates Duplicates

Names

ID	Name
1	Kevin
2	Sam
3	Jane
4	Kathi

Customers

CustID	FirstName
100	Bill
120	Denise
130	Anna
4	Kathi

ID	Name
1	Kevin
2	Sam
3	Jane
4	Kathi
100	Bill
120	Denise
130	Anna

```
SELECT ID, NAME
FROM Names
UNION
SELECT CustID, FirstName
FROM Customers;
```

UNION ALL – Retains duplicates

Names

ID	Name
1	Kevin
2	Sam
3	Jane
4	Kathi

Customers

CustID	FName
100	Bill
120	Denise
130	Anna
4	Kathi

ID	Name
1	Kevin
2	Sam
3	Jane
4	Kathi
100	Bill
120	Denise
130	Anna
4	Kathi

```
SELECT ID, NAME
FROM Names
UNION  ALL
SELECT CustID, FirstName
FROM Customers;
```

Except – Find items that don't match

Names

ID	Name
1	Kevin
2	Sam
3	Jane
4	Kathi

Customers

CustID	FName
100	Bill
120	Denise
130	Anna
4	Kathi

ID	Name
1	Kevin
2	Sam
3	Jane

```
SELECT ID, NAME
FROM Names
EXCEPT
SELECT CustID, FirstName
FROM Customers;
```

INTERSECT – Find rows that match

Names

ID	Name
1	Kevin
2	Sam
3	Jane
4	Kathi

Customers

CustID	FName
100	Bill
120	Denise
130	Anna
4	Kathi

ID	Name
4	Kathi

```
SELECT ID, NAME
FROM Names
INTERSECT
SELECT CustID, FirstName
FROM Customers;
```

Demo: UNION

DEMO: UNION

- 6-6 to 6-7

Lab

- Exercise 6-1, question 5 on page 135

Grouping

- Summarize the data
- Special functions called aggregate
  - COUNT
  - SUM
  - AVG
  - MIN
  - MAX
- Return one row per group

GROUP BY clause

- Columns in the SELECT list or ORDER BY must be in an aggregate function or in the GROUP BY
- Syntax

```
SELECT <col1>, <ag function>(<col2>)
FROM <table1>
GROUP BY <col1>
```

SELECT

SELECT

FROM

ON

WHERE

GROUP BY

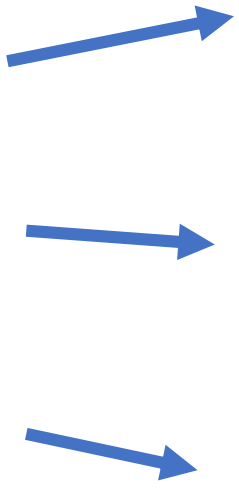
ORDER BY

Aggregate Queries

CustID	OrderID	TotalDue
1	1	50
2	2	76
1	3	150
1	4	25
3	5	100
3	6	30
2	7	60
1	8	90

Group by CustID

CustID	OrderID	TotalDue
1	1	50
2	2	76
1	3	150
1	4	25
3	5	100
3	6	30
2	7	60
1	8	90



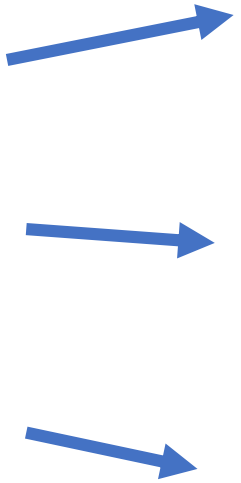
CustID	OrderID	TotalDue
1	1	50
1	3	150
1	4	25
1	8	90

CustID	OrderID	TotalDue
2	2	76
2	7	60

CustID	OrderID	TotalDue
3	5	100
3	6	30

Apply any aggregate functions: SUM, AVG, COUNT…

CustID	OrderID	TotalDue
1	1	50
2	2	76
1	3	150
1	4	25
3	5	100
3	6	30
2	7	60
1	8	90



CustID	OrderID	TotalDue
1	1	50
1	3	150
1	4	25
1	8	90

COUNT(OrderID) = 4
SUM(TotalDue) =315
MAX(OrderID) = 8

CustID	OrderID	TotalDue
2	2	76
2	7	60

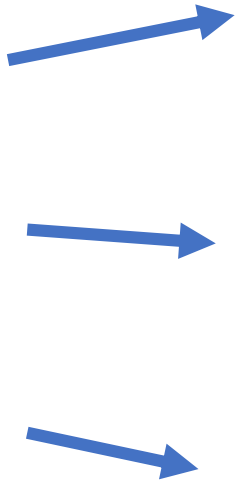
COUNT(OrderID) = 2
SUM(TotalDue) =136
MAX(OrderID) = 7

CustID	OrderID	TotalDue
3	5	100
3	6	30

COUNT(OrderID) = 2
SUM(TotalDue) =130
MAX(OrderID) = 6

Return 1 row per group

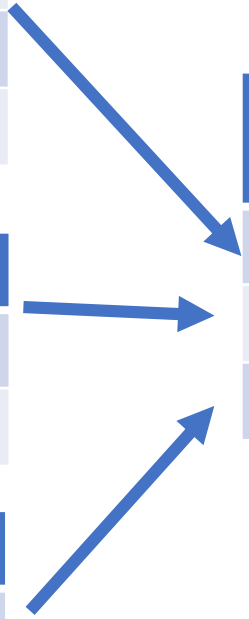
CustID	OrderID	TotalDue
1	1	50
2	2	76
1	3	150
1	4	25
3	5	100
3	6	30
2	7	60
1	8	90



CustID	OrderID	TotalDue
1	1	50
1	3	150
1	4	25
1	8	90

CustID	OrderID	TotalDue
2	2	76
2	7	60

CustID	OrderID	TotalDue
3	5	100
3	6	30



CustID	Order Count	Last Order	Total
1	4	8	315
2	2	7	136
3	2	6	130

DEMO: GROUP BY

Demo note

- 7-1 to 7-3

Lab

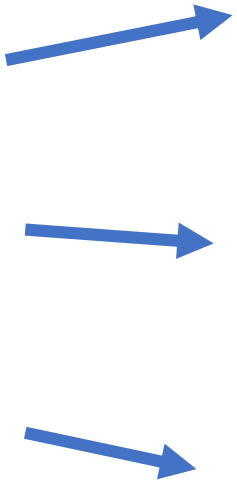
- Exercise 7-1, questions 1 – 5 on page 149
- Exercise 7-2, question 1 - 2 on page 152

HAVING

- Use to filter the groups using an aggregate function
- Use WHERE to filter rows
- Use HAVING to filter groups

Return 1 row per group

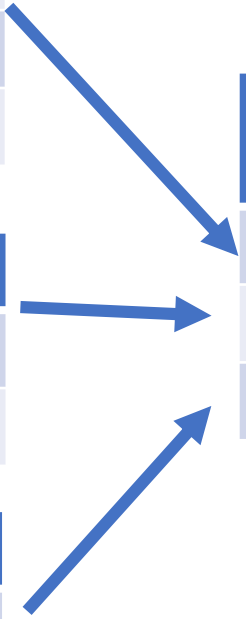
CustID	OrderID	TotalDue
1	1	50
2	2	76
1	3	150
1	4	25
3	5	100
3	6	30
2	7	60
1	8	90



CustID	OrderID	TotalDue
1	1	50
1	3	150
1	4	25
1	8	90

CustID	OrderID	TotalDue
2	2	76
2	7	60

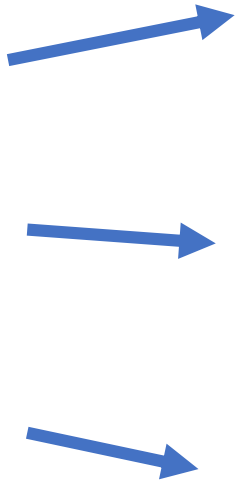
CustID	OrderID	TotalDue
3	5	100
3	6	30



CustID	Order Count	Last Order	Total
1	4	8	315
2	2	7	136
3	2	6	130

HAVING COUNT(OrderID) > 2

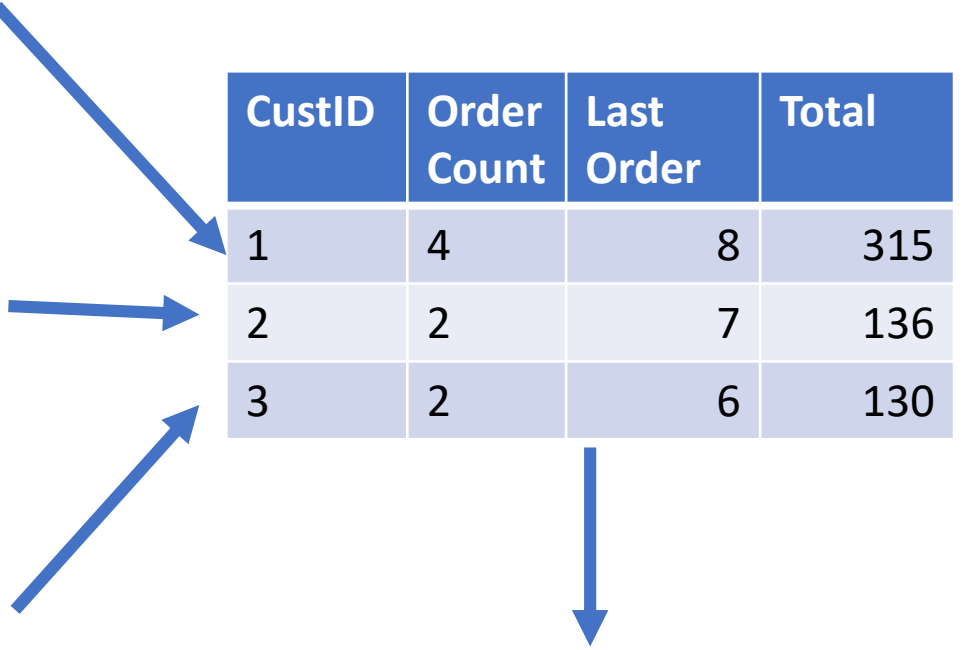
CustID	OrderID	TotalDue
1	1	50
2	2	76
1	3	150
1	4	25
3	5	100
3	6	30
2	7	60
1	8	90



CustID	OrderID	TotalDue
1	1	50
1	3	150
1	4	25
1	8	90

CustID	OrderID	TotalDue
2	2	76
2	7	60

CustID	OrderID	TotalDue
3	5	100
3	6	30



CustID	Order Count	Last Order	Total
1	4	8	315
2	2	7	136
3	2	6	130

CustID	Order Count	Last Order	Total
1	4	8	315

DEMO: HAVING

LAB: HAVING

- Exercise 7-3, 1, 2, 3, Page 157

Resources

- CoderGirl Videos (See included text files)
- Venn diagrams
- Article about group by
- Stairway series
- https://www.red-gate.com/simple-talk/sql/t-sql-programming/introduction-to-t-sql-window-functions/
- https://docs.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver15