IS475/675: While you are waiting for class to start...

- **Login to SQL Server Management Studio**
- **Execute the file called "Lab4CreateEmp4Dept4.sql"**
- **Execute the file called "SQLLab5.sql".**
- The files are located on the k: drive in the Cob\IS475\LabFiles folder.
- **Open SQL Lab Exercises 4 and 5 from WebCampus.**

Agenda for today (03/31/2025)

- Answer questions.
- Discuss group functions.
- Present SQL Join statements.

What is the goal of a SQL query?

- To produce an **accurate** result table.
- To produce an accurate result table that contains **meaningful information**.
- To produce an accurate result table that contains meaningful information that **will help a person solve** a **problem**.

Review: structure of the SELECT statement

```
SELECT      [all or distinct]
FROM        (table)
WHERE       (condition)
GROUP BY    (grouping fields)
HAVING      (condition)
ORDER BY    (sort fields)
```

Referred to as the "SELECT LIST"

When a SELECT statement is executed, the result is referred to as a "result table". It is a memory-based table.

What is a Group Function?

- A way to summarize data and provide more meaningful and informative output from the database. Sometimes referred to as “summary queries” or “aggregate functions.”
- Group functions differ from single row SELECT statements:
  - A SELECT statement processes every row in the underlying table. The result table (unless a WHERE clause is used) contains one row per row in the underlying table.
  - A group function collects data from multiple rows and produces summarized data in the result table. There should be one row in the result table per group.

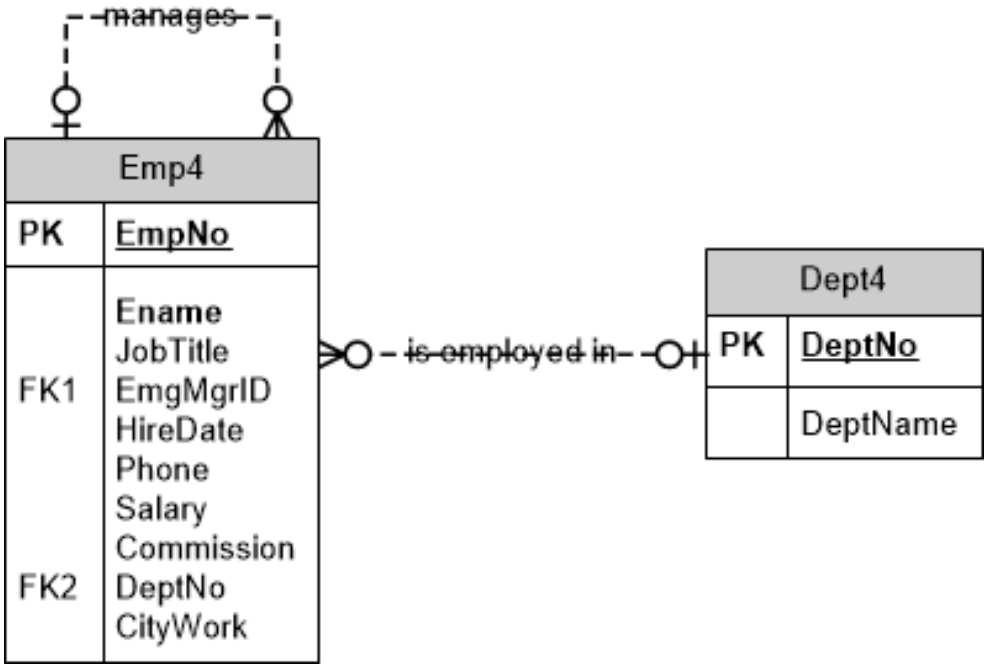
What do group functions produce?

If a group function is run on the whole table, without grouping, it generates a <u>single row</u> result table.

If a group function is run with grouping (the GROUP BY statement) then it generates one row per group in the result table.

Group Function	Description of What is Returned
AVG	Average value of a numeric column; ignores null values
COUNT	Number of rows. When * is used, all rows are returned (including null values and duplicate rows)
MAX	Maximum value of a column; ignores null values
MIN	Minimum value of a column; ignores null values
SUM	Totals the value of a numeric column; ignores null values

Tables in the Examples (SQL Lab #4)



Not maintaining referential integrity

Counting Rows

```
SELECT      COUNT(*)
FROM        Emp4;

SELECT      COUNT(*)
FROM           Emp4
WHERE       jobtitle = 'salesman';

SELECT      COUNT(commission)
FROM           Emp4

SELECT          deptno
FROM            Emp4
ORDER BY deptno;

SELECT          DISTINCT deptno
FROM Emp4;

SELECT      COUNT(DISTINCT deptno)
FROM        Emp4;
```

Calculating Averages

```
SELECT      AVG(salary)
FROM        Emp4;
```

Functions are executed starting at the most inner function (the last function) and working from right to left to the most outer function (the first function from left to right is the most outer)

```
SELECT      ROUND(AVG(salary),0)
FROM        Emp4;
```

In this statement, the AVG function is the most inner function and will be executed first. The ROUND function will be executed after the AVG function.

```
SELECT      ROUND(AVG(salary),2)
FROM        Emp4
WHERE       deptno = 20;
```

Use the ROUND function to perform both a mathematical rounding operation and truncate the result to a set number of digits after the decimal point

Finding Minimum and Maximum Values

```
SELECT      MIN(hiredate)
FROM        Emp4;


SELECT      MAX(hiredate)
FROM        Emp4;


SELECT MIN(hiredate)
FROM Emp4;


SELECT MIN(CONVERT(VARCHAR,hiredate,107))
FROM Emp4;
```

Don't convert a date field to a VARCHAR data type prior to doing a MAX or MIN function. The result will be potentially incorrect.

Combining group functions

```
SELECT      COUNT(salary),
            SUM(salary),
            MIN(salary)
FROM        Emp4

SELECT      COUNT(salary),
            SUM(salary),
            MIN(salary)
FROM        Emp4
WHERE       deptno = 10 and salary < 4000;
```

Group function issue…

Combining group functions with single row values - doesn't work!!

```
SELECT      deptno,
            COUNT(salary),
            SUM(salary)
FROM        Emp4
```

Creating summary output by grouping

```
SELECT      deptno,
            SUM(salary)
FROM        Emp4
GROUP BY    deptno;
```

```
SELECT      deptno,
            SUM(salary)
FROM        Emp4
WHERE       salary > 2000
GROUP BY    deptno;
```

Eliminates rows before the grouping occurs.

Cannot use a group function in a WHERE.

“Having” is a condition for groups

```
SELECT      deptno, SUM(salary)
FROM        Emp4
GROUP BY    deptno
HAVING      SUM(salary) > 6000;
```

Eliminates rows in the GROUP, rather than individual rows.

Can use a group function in a HAVING

```
SELECT      deptno, SUM(salary)
FROM        Emp4
GROUP BY    deptno
HAVING      AVG(salary) > 2000;
```

Multi-attribute grouping

```
SELECT     deptno,
           jobtitle,
           SUM(salary), AVG(salary)
FROM       Emp4
GROUP BY   deptno,
           jobtitle;
```

Find the name of the person(s) who make/s the smallest salary

	empno	ename	salary
1	7900	JAMES, KATHERINE	1950.00
2	3310	Nelson, Karen	1950.00

```
SELECT      empno,
            ename,
            MIN(salary)
FROM        emp4
```

These don't work!

```
SELECT      empno,
            ename,
            MIN(salary)
FROM        emp4
GROUP BY    empno, ename
```

Thoughts about fixing the problem…

```
SELECT          empno,
                ename,
                min(salary)
FROM            emp4
WHERE  salary = min(salary)
GROUP BY        empno, ename;

SELECT          empno,
                ename,
                min(salary)
FROM            emp4
GROUP BY        empno, ename
HAVING min(salary) = salary;
```

These don't work, either!

Could use the TOP 1 SELECT...

```
SELECT      TOP 1
            empno,
            ename,
            salary
FROM        emp4
ORDER BY    salary
```

But this works only in the SQL Server environment because TOP 1 is not ANSI-Standard SQL. It also only works if just one person makes the highest salary.

ANSI–Standard SQL
Needs a sub-query to work correctly

```
SELECT      empno,
            ename,
            salary
FROM        emp4
WHERE       salary =
      (SELECT     min(salary)
       FROM       emp4)
```

	empno	ename	salary
1	7900	JAMES, KATHERINE	1950.00
2	3310	Nelson, Karen	1950.00

What is a sub-query?

- A sub-query is a query embedded inside another query.
- The sub-query is executed in the normal operation of the query in which it is embedded.
- The sub-query will return an “answer” result table to the query in which it is embedded.
- A sub-query can be placed in the SELECT list, FROM statement, WHERE clause &/or HAVING clause.

Class summary so far

- The SELECT statement produces a result table. The goal is to produce information from underlying data stored in tables.
- The FROM contains the underlying table(s).
- The SELECT list generates the columns in the result table.
- The WHERE or HAVING filters rows from the underlying table.
- The GROUP BY allows consolidation of rows based on a pre-defined attribute.
- Group functions are especially helpful for consolidation of data to generate information from data.
- SQL Group functions are not conditions. To use a group function in a WHERE condition requires the use of a sub-query.

What is the goal of a SQL query?

- To produce an **accurate** result table.
- To produce an accurate result table that contains **meaningful information**.
- To produce an accurate result table that contains meaningful information that **will help a person solve** a **problem**.
- This usually requires the use of more than one table in a query.

SQL Join

SQL “joins” tables together to create a result table from more than one underlying table.

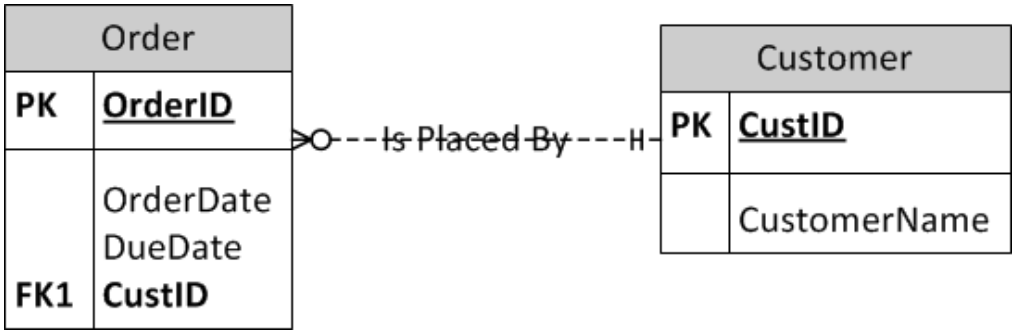
By default, SQL creates a “joined” result table by multiplying the rows of the underlying tables and adding the columns.

We don’t want the default!

SQL Join types

- **Join:** Joins tables based on rows where the value of a primary key is equal to the value of a foreign key.
  - **Inner join**: Returns only those rows with equal data values. Primary key = foreign key.
  - **Outer join**: Returns and same rows as returned with an inner join. Plus adds rows from one of the two tables that weren't returned from the inner join. Primary key = foreign key + any rows from one of the tables that weren't returned previously.

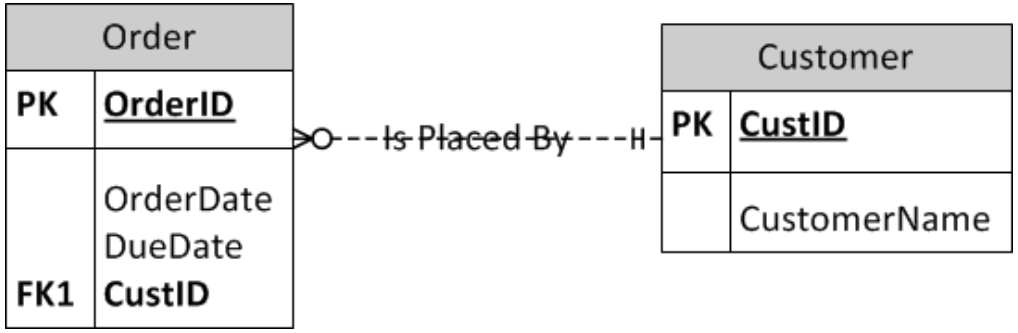
SQL Lab 5 – Task 1 (pg. 2)



What orders are currently in the database and what is the name of the customer who placed each order?

	OrderID	OrderDate	DueDate	CustID	CustomerName
1	100	2025-02-06 00:00:00.000	2025-02-11 00:00:00.000	1234	John Smith
2	200	2025-02-09 00:00:00.000	2025-02-17 00:00:00.000	6773	Bertie Wooster
3	300	2025-02-18 00:00:00.000	2025-03-02 00:00:00.000	1234	John Smith

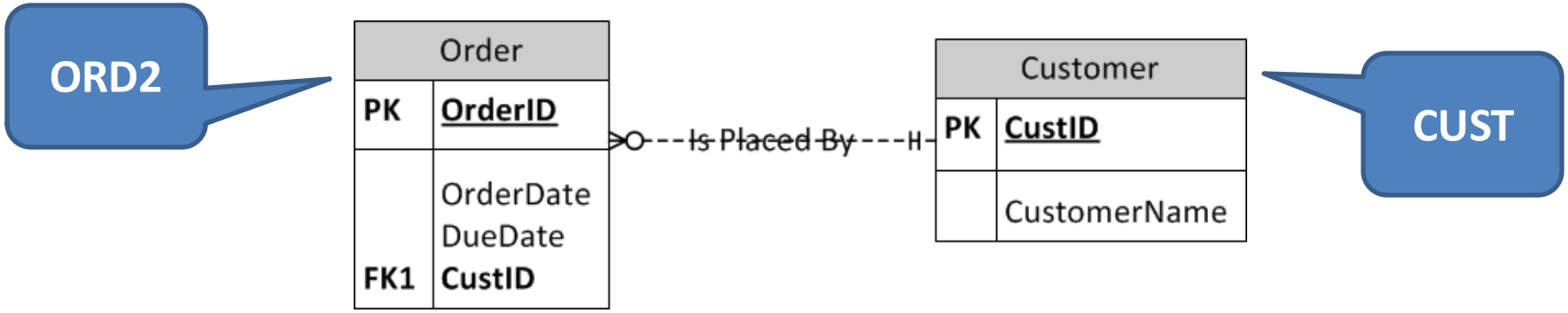
SQL Lab 5 – Task 2 (pg. 7)



What customers are in the database and what orders have been placed by those customers?

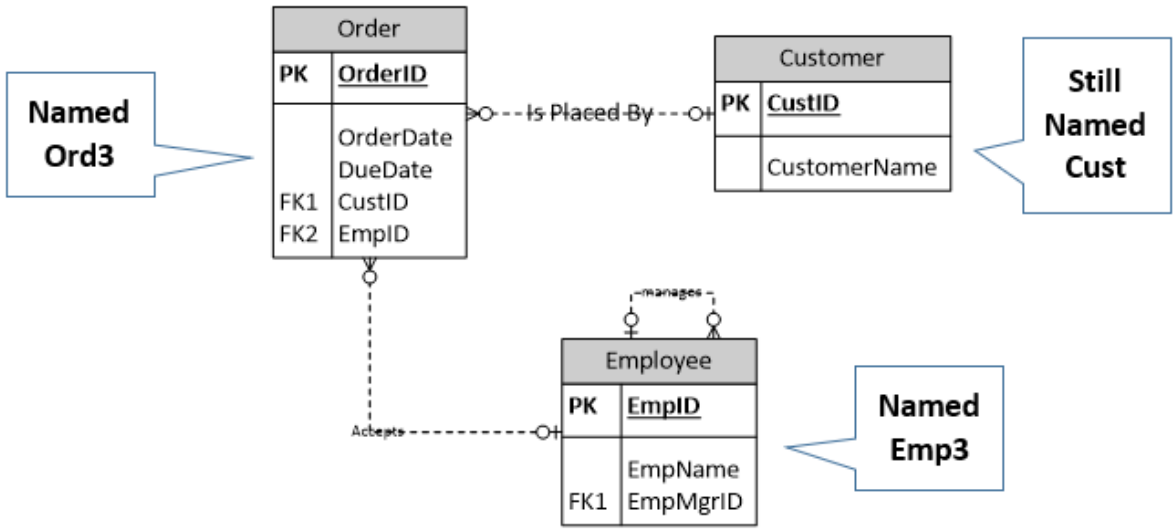
	CustomerName	OrderID	DueDate
1	Bertie Wooster	200	2025-02-17 00:00:00.000
2	Jane Doe	No Order	NULL
3	John Smith	100	2025-02-11 00:00:00.000
4	John Smith	300	2025-03-02 00:00:00.000
5	Martin Cheng	No Order	NULL

SQL Lab 5 – Task 3 (pg. 8)



	OrderID	OrderDate	CustID	DueDate	CustID	CustomerName
1	100	2025-02-06 00:00:00.000	1234	2025-02-11 00:00:00.000	1234	John Smith
2	200	2025-02-09 00:00:00.000	6773	2025-02-17 00:00:00.000	6773	Bertie Wooster
3	300	2025-02-18 00:00:00.000	1234	2025-03-02 00:00:00.000	1234	John Smith
4	400	2025-01-27 00:00:00.000	2555	2025-02-02 00:00:00.000	2555	Jane Doe
5	500	2025-02-12 00:00:00.000	8989	2025-02-22 00:00:00.000	NULL	NULL
6	600	2025-01-28 00:00:00.000	2555	2025-01-31 00:00:00.000	2555	Jane Doe
7	700	2025-02-05 00:00:00.000	2555	2025-02-13 00:00:00.000	2555	Jane Doe

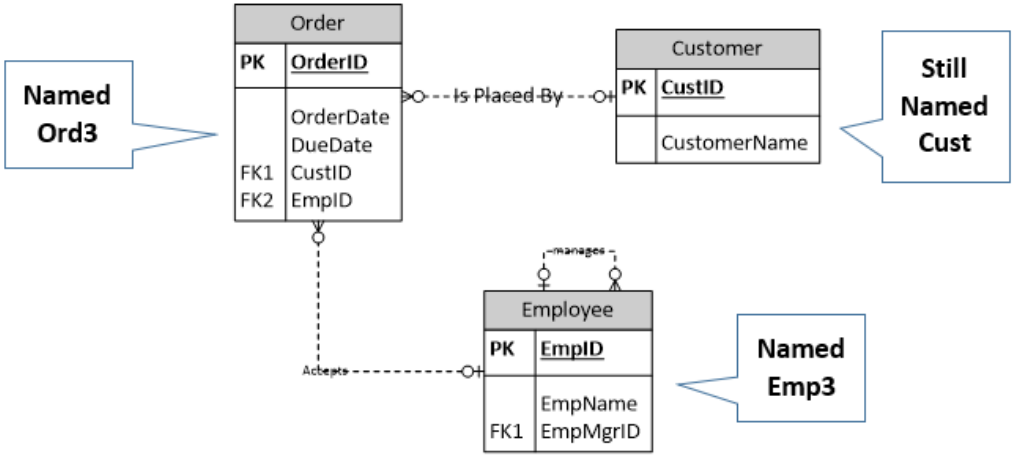
SQL Lab 5 – Task 4 (pg. 11)



What orders are currently in the database and what is the name of the customer who placed each order? What is the name of the employee who accepted the order?

	orderID	OrderDate	DueDate	CustomerName	EmpName
1	100	2025-02-06 00:00:00.000	2025-02-11 00:00:00.000	John Smith	Ling
2	200	2025-02-09 00:00:00.000	2025-02-17 00:00:00.000	Bertie Wooster	Bassett
3	300	2025-02-18 00:00:00.000	2025-03-02 00:00:00.000	John Smith	Bassett
4	400	2025-01-27 00:00:00.000	2025-02-02 00:00:00.000	Jane Doe	Johnson
5	500	2025-02-12 00:00:00.000	2025-02-22 00:00:00.000	NULL	Johnson
6	600	2025-01-28 00:00:00.000	2025-01-31 00:00:00.000	Jane Doe	Johnson
7	700	2025-02-05 00:00:00.000	2025-02-13 00:00:00.000	Jane Doe	Stein

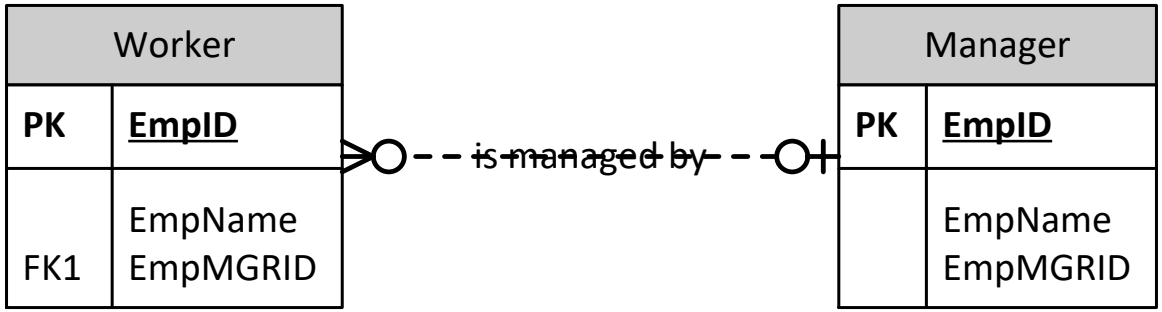
SQL Lab 5 – Task 5 (pg. 14)



What orders are currently in the database and what is the name of the customer who placed each order? What is the name of the employee who accepted the order? What is the name of the manager of the employee who accepted the order?

	orderID	OrderDate	DueDate	CustomerName	EmployeeName	ManagerName
1	100	2025-02-06 00:00:00.000	2025-02-11 00:00:00.000	John Smith	Ling	Torquez
2	200	2025-02-09 00:00:00.000	2025-02-17 00:00:00.000	Bertie Wooster	Bassett	Martinson
3	300	2025-02-18 00:00:00.000	2025-03-02 00:00:00.000	John Smith	Bassett	Martinson
4	400	2025-01-27 00:00:00.000	2025-02-02 00:00:00.000	Jane Doe	Johnson	Torquez
5	500	2025-02-12 00:00:00.000	2025-02-22 00:00:00.000	NULL	Johnson	Torquez
6	600	2025-01-28 00:00:00.000	2025-01-31 00:00:00.000	Jane Doe	Johnson	Torquez
7	700	2025-02-05 00:00:00.000	2025-02-13 00:00:00.000	Jane Doe	Stein	Martinson

SQL Lab 5 – Task 5 (pg. 16)



	WorkerID	WorkerName	ManagerID	ManagerName
1	2	Polanski	1	Martinson
2	3	Torquez	1	Martinson
3	4	Ling	3	Torquez
4	5	Bassett	1	Martinson
5	6	Martinez	1	Martinson
6	7	Johnson	3	Torquez
7	8	Cheng	1	Martinson
8	9	Fukamota	3	Torquez
9	10	Stein	1	Martinson

Outer join

- Returns the same results as an inner join
- Plus returns the rows from a table that were not returned with an inner join
- Left outer join: Returns inner plus the rows in the table declared first in the FROM
- Right outer join: Returns inner plus the rows in the table declared second in the FROM

Summary of joins

- Used to provide more information in the result table.
- The purpose of a join is to combine more than one underlying table into a single result table.
- **Inner join**: Combines the rows from two tables where the foreign key in the child table equals the primary key in the parent table.
- **Outer join**: Combines the rows from two tables where the foreign key in the child table equals the primary key in the parent table. In addition, an outer join will include those rows in one (left, right) or both (full) tab
- **Self join**: Used to join one copy of a table to another copy of the same table. Used when there is a unary relationship in the data model.