# FILTRANDO GRUPOS

# LA CLAUSULA HAVING

- Así como la cláusula "where" permite seleccionar (o rechazar) registros individuales; la cláusula "having" permite seleccionar (o rechazar) un grupo de registros.

```
SELECT title, AVG(salary)
FROM employees
GROUP BY empid
HAVING AVG(salary) > 25000
```

# DISTINCT

- La palabra clave DISTINCT elimina las filas duplicadas de los resultados de una instrucción SELECT. Si no se especifica DISTINCT, se devuelven todas las filas, incluidas las duplicadas.

```
SELECT  DISTINCT productid
FROM Sales.Orderdetails
```

# CONSULTAS CON MULTIPLES TABLAS

Bienvenido al mundo real!!!

# Introducción

- Hasta el momento hemos trabajado con una sola tabla, pero generalmente, se trabaja con más de una.

- Si distribuimos la información en varias tablas evitamos la redundancia de datos y ocupamos menos espacio físico en el disco. **Un join** es una operación que relaciona dos o más tablas para obtener un resultado que incluya datos (campos y registros) de ambas; las tablas participantes se combinan según los campos comunes a ambas tablas.

# JOINS

- Un join es una operación que relaciona dos o más tablas para obtener un resultado que incluya datos (campos y registros) de ambas; las tablas participantes se combinan según los campos comunes a ambas tablas.

- Hay tres tipos de combinaciones:
  - combinaciones internas (inner join o join),
  - combinaciones externas y
  - combinaciones cruzadas.

- También es posible emplear varias combinaciones en una consulta "select", incluso puede combinarse una tabla consigo misma (self join).

# Inner Join

- Se emplea para obtener información de dos tablas y combinar dicha información en una salida siempre y cuando se cumpla la condición especificada en la clausula ON.

```
SELECT [campos] -- campos a listar
FROM tabla1
INNER JOIN  tabla2 -- tabla con la que se combina
ON  tabla1.campox = tabla2.campox – condición
```

# Ejemplos

```
SELECT
        Sales.Orders.orderid,
        HR.Employees.empid,
        HR.Employees.lastname,
        HR.Employees.firstname
FROM HR.Employees
                INNER JOIN
         Sales.Orders
        ON HR.Employees.empid = Sales.Orders.empid
ORDER BY Sales.Orders.orderid
```

# Usando “Alias”

```
SELECT
        ord.orderid,
        emp.empid,
        emp.lastname,
        emp.firstname
FROM HR.Employees AS emp
        INNER JOIN
Sales.Orders AS ord
        ON emp.empid = ord.empid
ORDER BY ord.orderid
```

# LEFT JOIN

- Se emplea una combinación externa izquierda para mostrar todos los registros de la tabla de la izquierda. Si no encuentra coincidencia con la tabla de la derecha, el registro muestra los campos de la segunda tabla seteados a "null“

- **SELECT** c.customerid, o.orderid
- **FROM** Customers **AS** c
- **LEFT JOIN**
- Orders **AS** o
- on c.customerid = o.customerid
- **WHERE** o.orderid is null

# RIGHT JOIN

- Vimos que una combinación externa izquierda (left join) encuentra registros de la tabla izquierda que se correspondan con los registros de la tabla derecha y muestra los campos correspondientes a la tabla de la derecha seteados a "null".
- Una combinación externa derecha ("right outer join" o "right join") opera del mismo modo sólo que la tabla derecha es la que localiza los registros en la tabla izquierda.

```
SELECT
    c.customerid,
    c.Companyname,
    o.orderid,
    o.orderdate
FROM orders as o right outer join customers as c
ON c.customerid = o.customerid
WHERE OrderID is null
```

# Cross Join

- Las combinaciones cruzadas (cross join) muestran todas las combinaciones de todos los registros de las tablas combinadas. Para este tipo de join no se incluye una condición de enlace. Se genera el producto cartesiano en el que el número de filas del resultado es igual al número de registros de la primera tabla multiplicado por el número de registros de la segunda tabla, es decir, si hay 5 registros en una tabla y 6 en la otra, retorna 30 filas.

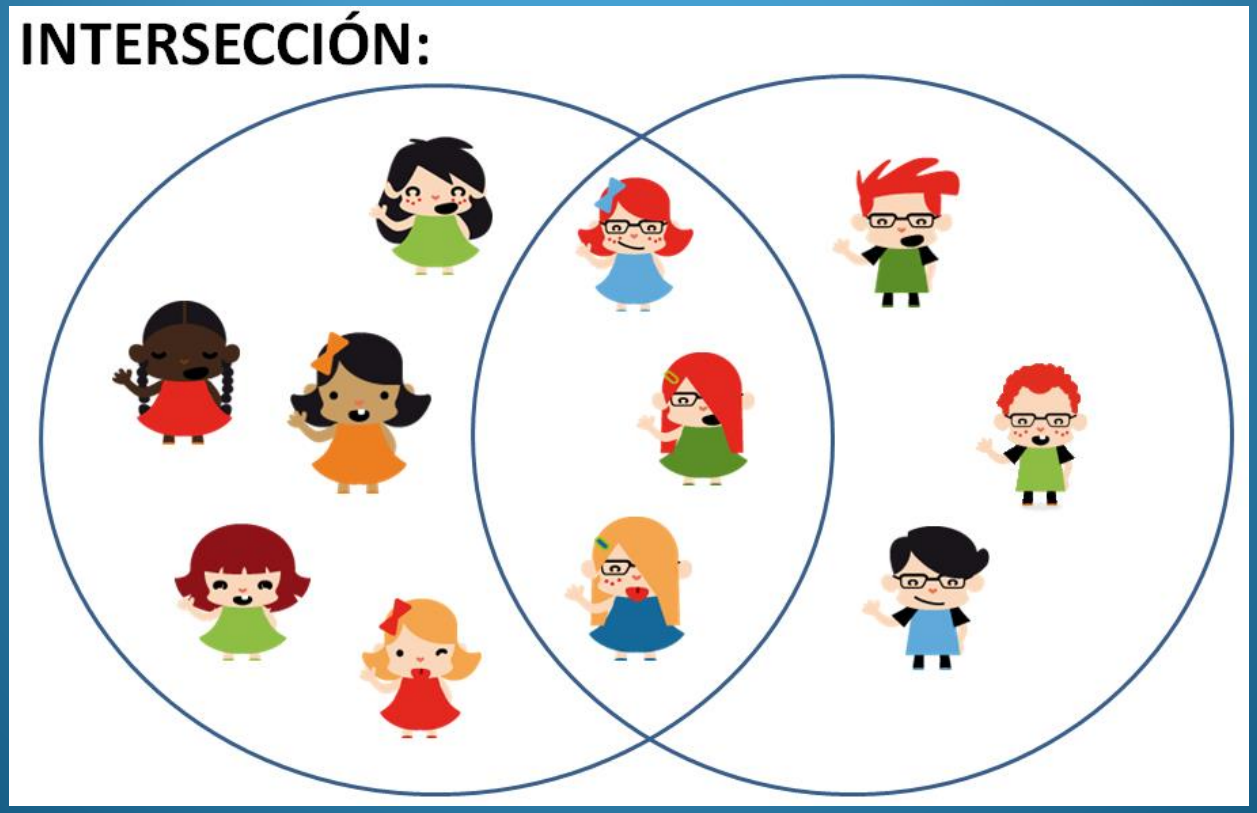
```
SELECT
        e.employeeid,
        o.ordered
FROM employees e CROSS JOIN orders o
```

# SELF JOIN

- En una autocombinación se combina una tabla con una copia de si misma. Para ello debemos utilizar 2 alias para la tabla. Para evitar que aparezcan filas duplicadas, debemos emplear un "where".

```
SELECT
        e.employeeID,
        e.FirstName + Space(1) + e.LastName as Empleado,
        e.title,
        su.employeeID SupervisorID,
        su.FirstName + Space(1) + su.LastName as Supervisor,
        su.title
FROM Employees e
                INNER JOIN
         Employees su
                ON e.ReportsTo= su.employeeid
```

# OPERADORES DE CONJUNTO



# Introducción

- Los operadores de conjuntos, permiten combinar resultados de varias consultas en un unico resultado

- UNION
- INTERCEPT
- EXCEPT

# UNION

- Combina los resultados de dos o más consultas en un solo conjunto de resultados que incluye todas las filas que pertenecen a las consultas de la unión. La operación UNION es distinta de la utilización de combinaciones de columnas de dos tablas.

```
SELECT campo1, campo2 FROM tablax
UNION
SELECT campo1, campo2 FROM tablax
```

# EXCEPT

- EXCEPT devuelve filas distintas de la consulta de entrada izquierda que no son de salida en la consulta de entrada derecha.

```
SELECT campo1, campo2 FROM tablax
EXCEPT
SELECT campo1, campo2 FROM tablax
```

# INTERSECT

- INTERSECT devuelve los valores distintos que devuelven tanto la consulta de la parte izquierda como la consulta de la parte derecha del operador INTERSECT. Los conjuntos de resultados que se comparan con EXCEPT o INTERSECT deben tener la misma estructura. Deben tener el mismo número de columnas y las columnas del conjunto de resultados deben tener tipos de datos compatibles.

```
SELECT campo1, campo2 FROM tablax
INTERSECT
SELECT campo1, campo2 FROM tablax
```