**OTHER JOIN TYPES – PRACTICE QUESTIONS**

To recap, an inner join will only show rows where there are matching values in both tables e.g.

SELECT i.ordid, i.itemid, i.qty, di.qty

FROM ord o

INNER JOIN item i ON o.ordid = i.ordid

INNER JOIN delivered\_item di ON i.ordid = di.ordid

AND i.itemid = di.itemid

WHERE o.custid = 104

ORDER BY i.ordid, i.itemid;

|  |  |  |  |
| --- | --- | --- | --- |
| **ORDID** | **ITEMID** | **QTY** | **QTY** |
| 612 | 1 | 100 | 75 |
| 612 | 1 | 100 | 25 |
| 612 | 2 | 20 | 20 |
| 612 | 3 | 150 | 100 |
| 612 | 3 | 150 | 50 |
| 612 | 4 | 100 | 100 |
| 619 | 1 | 100 | 100 |
| 619 | 2 | 100 | 100 |
| 619 | 3 | 100 | 100 |
| 619 | 4 | 50 | 50 |

**OUTER JOINS**

Sometimes you want to identify records where there are no matching values in two tables. For example, which items have and have not been delivered for a particular customer. If an item has not been delivered there will be no delivered\_item record and an inner join will not show that item so we need to use an OUTER JOIN to display those rows.

1. Display all items ordered and whether they have been delivered or not for customer id 104.

SELECT i.ordid, i.itemid, i.qty AS 'Item Qty', di.qty AS 'Delivered Qty'

FROM ord o

INNER JOIN item i ON o.ordid = i.ordid

LEFT OUTER JOIN delivered\_item di ON i.ordid = di.ordid

AND i.itemid = di.itemid

WHERE o.custid = 104

ORDER BY i.ordid, i.itemid;

|  |  |  |  |
| --- | --- | --- | --- |
| **ORDID** | **ITEMID** | **ITEM QTY** | **DELIVERED QTY** |
| 607 | 1 | 1 | NULL |
| 608 | 1 | 1 | NULL |
| 608 | 2 | 2 | NULL |
| 612 | 1 | 100 | 75 |
| 612 | 1 | 100 | 25 |
| 612 | 2 | 20 | 20 |
| 612 | 3 | 150 | 100 |
| 612 | 3 | 150 | 50 |
| 612 | 4 | 100 | 100 |
| 619 | 1 | 100 | 100 |
| 619 | 2 | 100 | 100 |
| 619 | 3 | 100 | 100 |
| 619 | 4 | 50 | 50 |

Using a LEFT OUTER JOIN will cause any records on the left hand-side of the join (item in this case) that have no matching records on the right hand side of the join (delivered\_item). Orders 607 and 608 have not been delivered and now they are being displayed.

2.  If you wanted to find out all the order items for customer 104 that were delivered in more than one consignment, you might try using **di.qty < i.qty** as follows:

SELECT i.ordid, i.itemid, i.qty, di.qty

FROM ord o

INNER JOIN item i ON o.ordid = i.ordid

LEFT OUTER JOIN delivered\_item di ON i.ordid = di.ordid

AND i.itemid = di.itemid

WHERE o.custid = 104

AND di.qty < i.qty

ORDER BY i.ordid, i.itemid;

|  |  |  |  |
| --- | --- | --- | --- |
| **ORDID** | **ITEMID** | **QTY** | **QTY** |
| 612 | 1 | 100 | 75 |
| 612 | 1 | 100 | 25 |
| 612 | 3 | 150 | 100 |
| 612 | 3 | 150 | 50 |

3. If you want to find out all the order items that had not been delivered at all, the delivery quantity will be NULL so you have to adjust the query as follows:

SELECT i.ordid, i.itemid, i.qty, di.qty

FROM ord o

INNER JOIN item i ON o.ordid = i.ordid

LEFT OUTER JOIN delivered\_item di ON i.ordid = di.ordid

AND i.itemid = di.itemid

WHERE o.custid = 104

AND di.qty IS NULL

ORDER BY i.ordid, i.itemid;

|  |  |  |  |
| --- | --- | --- | --- |
| **ORDID** | **ITEMID** | **QTY** | **QTY** |
| 607 | 1 | 1 | NULL |
| 608 | 1 | 1 | NULL |
| 608 | 2 | 2 | NULL |

4. You want to display all rackets and the orders that have been placed for them irrespective of whether a product has been ordered or not.

SELECT prodid, descrip, o.ordid, i.itemid, i.qty

FROM product p

LEFT OUTER JOIN item i ON p.prodid = i.prodid

LEFT OUTER JOIN ord o ON o.ordid = i.ordid

WHERE descrip LIKE '%RACKET%'

ORDER BY p.prodid;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PRODID** | **DESCRIP** | **ORDID** | **ITEMID** | **QTY** |
| 100860 | ACE TENNIS RACKET I | 612 | 1 | 100 |
| 100860 | ACE TENNIS RACKET I | 604 | 3 | 10 |
| 100860 | ACE TENNIS RACKET I | 603 | 1 | 4 |

….

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 101863 | SP JUNIOR RACKET | 605 | 5 | 100 |
| 100860 | SQUASH RACKET I | NULL | NULL | NULL |
| 100860 | SQUASH RACKET II | NULL | NULL | NULL |

27 rows selected

Note: You have to use LEFT OUTER JOIN for both joins (including between item and ord) otherwise the products that haven’t been ordered will not be displayed.

5. Display all customers irrespective of whether they have a sales rep associated with them or not and all employees irrespective of whether they are a sales rep for a customer or not. This is a FULL OUTER JOIN so in SQLite, we need to use two separate LEFT OUTER JOINS and merge them together using a UNION clause.

SELECT ename AS REP\_NAME, name AS CUST\_NAME

FROM emp e

LEFT OUTER JOIN customer c ON c.repid = e.empno

UNION

SELECT ename AS REP\_NAME, name AS CUST\_NAME

FROM customer c

LEFT OUTER JOIN emp e ON e.empno = c.repid;

|  |  |
| --- | --- |
| REP\_NAME | CUST\_NAME |
| NULL | KW SPORTS |
| ADAMS | NULL |
| ALLEN | EVERY MOUNTAIN |
| ALLEN | WOMENS SPORTS |
| BLAKE | NULL |
| CLARK | NULL |
| FORD | NULL |
| JAMES | NULL |
| JONES | NULL |
| KING | NULL |
| MARTIN | TKB SPORT SHOP |
| MARTIN | VOLLEYRITE |
| MILLER | NULL |
| SCOTT | NULL |
| SMITH | NULL |
| TURNER | JOCKSPORTS |
| TURNER | K + T SPORTS |
| TURNER | NORTH WOODS HEALTH AND FITNESS SUPPLY CENTRE |
| WARD | JUST TENNIS |
| WARD | SHAPE UP |

**SELF-JOINS**

The mgr column in the emp table contains the employee number (empno) of the employee's manager.   Emp is therefore a self-referencing table.  To extract details of both an employee name and his/her manager’s name, the table needs to be joined to itself.  This is done using two different table aliases for the same table.

Here the instance of emp called e represents the employee, whilst the instance of emp called m represents his/her manager.

6. SELECT e.empno AS 'Emp No', e.ename 'Emp Name', m.empno AS 'Mgr No', m.ename AS 'Mgr Name'

FROM emp e

INNER JOIN emp m ON e.mgr = m.empno

ORDER BY m.ename;

|  |  |  |  |
| --- | --- | --- | --- |
| **Emp No** | **Emp Name** | **Mgr No** | **Mgr Name** |
| 7654 | MARTIN | 7698 | BLAKE |
| 7499 | ALLEN | 7698 | BLAKE |
| 7900 | JAMES | 7698 | BLAKE |
| 7844 | TURNER | 7698 | BLAKE |
| 7521 | WARD | 7698 | BLAKE |
| 7934 | MILLER | 7782 | CLARK |
| 7369 | SMITH | 7902 | FORD |
| 7902 | FORD | 7566 | JONES |
| 7788 | SCOTT | 7566 | JONES |
| 7698 | BLAKE | 7839 | KING |
| 7782 | CLARK | 7839 | KING |
| 7566 | JONES | 7839 | KING |
| 7876 | ADAMS | 7788 | SCOTT |

We need to change the column headings to make the output understandable.

**Note that this query only retrieves 13 rows because one employee (KING) does not have a manager.  Using a LEFT OUTER JOIN instead of an inner join will cause all 14 employees to be retrieved as follows:**

|  |  |  |  |
| --- | --- | --- | --- |
| **EMPNO** | **ENAME** | **Mgr No** | **Mgr Name** |
| 7839 | KING | NULL | NULL |
| 7654 | MARTIN | 7698 | BLAKE |
| 7499 | ALLEN | 7698 | BLAKE |
| 7900 | JAMES | 7698 | BLAKE |
| 7844 | TURNER | 7698 | BLAKE |
| 7521 | WARD | 7698 | BLAKE |
| 7934 | MILLER | 7782 | CLARK |
| 7369 | SMITH | 7902 | FORD |
| 7902 | FORD | 7566 | JONES |
| 7788 | SCOTT | 7566 | JONES |
| 7698 | BLAKE | 7839 | KING |
| 7782 | CLARK | 7839 | KING |
| 7566 | JONES | 7839 | KING |
| 7876 | ADAMS | 7788 | SCOTT |

**NON-EQUI JOINS**

Occasionally, where there is no defined foreign key relationship, tables are joined on the basis of a value in one belonging to a range in another **(i.e. joins based on something other than equal values)**

7.       We want to display the employee name, monthly salary and their grade for all employees beginning with a J or an S. Sort the results by employee name.

SELECT ename, monthly\_sal, grade

FROM emp e

INNER JOIN salgrade s ON e.monthly\_sal BETWEEN s.losal AND s.hisal

AND (ename LIKE 'S%'

OR ename LIKE 'J%')

ORDER BY ename;

|  |  |  |
| --- | --- | --- |
| **ENAME** | **MONTHLY\_SAL** | **GRADE** |
| JAMES | 950 | 1 |
| JONES | 2975 | 4 |
| SCOTT | 3000 | 4 |
| SMITH | 800 | 1 |