COMP Lab #2 Complex Queries

- Use the Pine Valley database. *This is the database that you installed in Lab 1.*
  For most of the beginning queries, you don't need the "Big" version of Pine Valley.
  For the "more complicated" queries, use the "Big" version of the Pine Valley Database.
  It has more tables, some structural changes, and a lot more data.
  For this lab, you will need the DDL and the first of the two data files (load1)
  For Homework 4, you will also need load2.

Cross Product (no join condition)

```
SELECT *
FROM customer_t, order_t;

SELECT *
FROM customer_t
JOIN order_t;
```

With a join condition: NOTE that these are INNER JOINS (deprecated in version 5 as a clause)

- duplicate column for join field (Equijoin)
- only those records with matching join field are included

```
SELECT Customer_t.CustomerID, Order_T.CustomerID, CustomerName,OrderID
FROM Customer_T,Order_t
WHERE Customer_T.CustomerID  = Order_T.CustomerID
ORDER BY OrderID;

alternatively:  here we explicitly select both CustomerID fields
SELECT Customer_t.CustomerID, Order_T.CustomerID, CustomerName,OrderID
FROM Customer_T JOIN Order_t
USING (CustomerID)
ORDER BY OrderID;
```

Natural Join much more common

```
SELECT *
FROM Customer_T NATURAL JOIN Order_t;
```

Outer Join (LEFT [OUTER] JOIN)

```
SELECT Customer_t.CustomerID,CustomerName,OrderID
FROM Customer_T LEFT OUTER JOIN Order_t
Using (CustomerID);
```

Outer Join (RIGHT [OUTER] JOIN) the FK constraint makes this not so outer

```
SELECT Customer_t.CustomerID,CustomerName,OrderID
FROM Customer_T RIGHT OUTER JOIN Order_t
Using (CustomerID);
```

Example joining 4 tables:

Assemble all information necessary to create an invoice for order number 1006.

Self Join (often for a unary relationship)

What are the employee ID and name of each employee and the name of his or her supervisor (label the supervisor's name Manager)?

FROM Employee_t E, Employee_t M

Please note that this query is run twice: once on Pine Valley, and once on "Big" Pine Valley.

Subquery (sometimes called a "nested" subquery)

What are the name and address of the customer who placed order number 1008?
Try this with a JOIN on Customer and Order, which could get slow and $$.)

(Try the same query, constructed as a subquery.)

What are the names of customers who have placed orders?

Which customers have not placed any orders for computer desks?

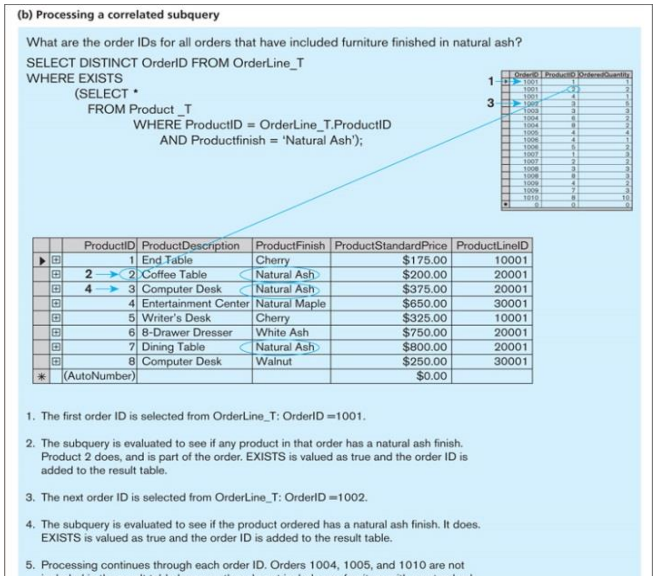
EXISTS: returns true or false

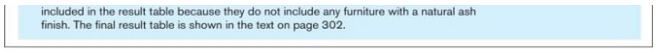
Correlated subquery: use the result of the outer query to compute the inner query.

What are the OrderIDs for all orders that have included
furniture finished in Natural Ash?

(We'll work through this one together!)

```
SELECT DISTINCT OrderID FROM OrderLine_t
WHERE EXISTS
        (SELECT *
        FROM Product_T
        WHERE ProductID=OrderLine_t.ProductID
        AND ProductFinish='Natural Ash');
```




included in the result table because they do not include any furniture with a natural ash finish. The final result table is shown in the text on page 302.

The subquery is executed for each order line in the outer query. The subquery checks for each order line to see if the finish for the product on that order line is natural ash. If this is true (EXISTS), the outer querydisplays the order ID for that order. The outer query checks this one row at a tme for every row in the set of referenced rows (the OrderLine_T table). There have been seven such orders, as the results shows.

List the details about the product with the highest standard price.

```
SELECT ProductDescription, ProductFinish, ProductStandardPrice
FROM Product_t PA
WHERE PA.ProductStandardPrice > ALL
        (SELECT ProductStandardPrice FROM Product_t PB
        WHERE PB.ProductID !=PA.ProductID);
```

Derived Tables

Show the product description, product standard price, and overall average standard price for all products that have a standard price that is higher than the average standard price.

NOTE: We actually demonstrated this example in Lab 1.

```
SELECT ProductDescription, ProductStandardPrice,AvgPrice
FROM
   (SELECT AVG(ProductStandardPrice)AS AvgPrice FROM Product_t) AS ProdAvg, Product_T
WHERE ProductStandardPrice > AvgPrice;
```

UNION (just an example)

The following query determines the customer(s) who has in a given line item purchased the largest quantity of any Pine Valley product and the customer(s) who has in a given line item purchased the smallest quantity and reutrns the result in one table.

```
SELECT C1.CustomerID, CustomerName,OrderedQuantity,'Largest Quantity' AS Quantity
FROM Customer_T C1, Order_T O1, OrderLine_T Q1
        WHERE C1.CustomerID =O1.CustomerID
        AND O1.OrderID = Q1.OrderID
        AND OrderedQuantity =
            (SELECT MAX(OrderedQuantity)
             FROM OrderLine_T)
UNION
SELECT C1.CustomerID, CustomerName,OrderedQuantity,'SMallest Quantity' AS Quantity
FROM Customer_T C1, Order_T O1, OrderLine_T Q1
        WHERE C1.CustomerID =O1.CustomerID
        AND O1.OrderID = Q1.OrderID
        AND OrderedQuantity =
            (SELECT MIN(OrderedQuantity)
```

```
                FROM OrderLine_T)
    ORDER BY 3;
```

There is also INTERSECT and MINUS (Or DIFFERENCE OR EXCEPT)

Conditional Expressions

"What products are included in ProductLine 1?"

```
SELECT CASE
        WHEN ProductLineID = 1 THEN ProductDescription
        ELSE '####'
END AS ProductDescription
FROM Product_t;
```

More complicated SQL Queries

- For each salesperson, list his or her biggest-selling product.
  Hint: You may find it helpful to first define a view, then query against that view.

--> (use the BIG version of PVFC for this query)

```
CREATE VIEW Tsales AS
        SELECT SalespersonName, ProductDescription,
                SUM(OrderedQuantity) AS TotOrders
        FROM Salesperson_T, OrderLine_T ,Product_T, Order_T
        WHERE Salesperson_T.SalespersonID = Order_T.SalespersonID
        AND Order_T.OrderID = Orderline_T.OrderID
        AND orderLine_T.ProductID = Product_t.ProductID
        GROUP BY SalespersonName,ProductDescription;
```

Next, we write a correlated subquery using the view:

```
SELECT SalespersonName,ProductDescription
FROM Tsales AS A
        WHERE Totorders = (SELECT MAX(Totorders) FROM Tsales B
                                WHERE B.SalespersonName = A.SalespersonName);
```

- Write an SQL query to list all salespersons who work in the territory where the most end tables have been sold.

  - First, you have to create a table to store the query results.
    Create table TopTerritory;
    Then, INSERT INTO TOPTERRITORY (SELECT…..)
    Or, do it in one statement: CREATE TABLE TOPTERRITORY SELECT…

    ```
    SELECT Territory_t.TerritoryID,
            SUM(OrderedQuantity) AS TotSales
    FROM Territory_t JOIN (Product_T JOIN
            (((Customer_T JOIN DoesBusinessIn_T ON
            Customer_T.CustomerID = DoesBusinessIn_t.CustomerID)
            JOIN Order_t ON Customer_t.CustomerID =
            Order_t.CustomerID) Join OrderLine_t ON
            Order_T.OrderID=OrderLine_t.OrderID) ON
            Product_t.ProductID = OrderLine_t.PRoductID) ON
            Territory_t.TerritoryID = DoesBusinessIn_t.TerritoryID
    WHERE ((ProductDescription)='End Table')
    GROUP BY Territory_t.TerritoryID
    ORDER BY TotSales DESC
    LIMIT 1;
    ```

  - OR, you can use this:
    SELECT Territory_t.TerritoryID,
    SUM(OrderedQuantity) AS TotSales
    FROM Territory_t, Customer_t, DoesBusinessIn_t, OrderLine_t, Order_t, Product_t
    WHERE Customer_T.CustomerID = DoesBusinessIn_t.CustomerID
    AND Order_T.OrderID=OrderLine_t.OrderID
    AND Customer_t.CustomerID = Order_t.CustomerID
    AND Product_t.ProductID = OrderLine_t.ProductID
    AND Territory_t.TerritoryID = DoesBusinessIn_t.TerritoryID
    AND ProductDescription = 'End Table'
    GROUP BY Territory_t.TerritoryID
    ORDER BY TotSales DESC
    LIMIT 1;

  - *THEN, pose a query against the new table (or you could have made it a view, whatever)*

    ```
    SELECT SalesPerson_T.SalesPersonID,SalesPersonName
    FROM Territory_T JOIN SalesPerson_T ON
            Territory_T.TerritoryID = SalesPerson_T.TerritoryID
    WHERE SalesPerson_T.TerritoryID IN
    ```

(SELECT TerritoryID FROM TopTerritory);

[Your Company Name]

[Your Company Slogan]

[Address]
[Town, County Postal Code]
Phone [01234 567890] Fax [01234 567890]

INVOICE

INVOICE No [100]
DATE: 9 October, 2011

Billing Address:
[Name]
[Company]
[Address]
[Town, County Postal Code]
[Phone]

Delivery Address:
[Name]
[Company]
[Address]
[Town, County Postal Code]
[Phone]

Comments or special instructions:

SALESPERSON	P.O. NUMBER	SENT DATE	SENT VIA	F.O.B. POINT	TERMS
					Due on receipt

QUANTITY	DESCRIPTION	UNIT PRICE	AMOUNT
		SUBTOTAL	
		SALES TAX	
		P&P	
		TOTAL DUE	

Make all cheques payable to **[Your Company Name]**
If you have any questions concerning this invoice, contact [Name, Phone Number, E-mail]

THANK YOU FOR YOUR BUSINESS!