SQL Practical Exercise

Alex Sikorski

1.1)

```
SELECT CustomerID, CompanyName,City
FROM Customers
WHERE City = 'Paris' OR City = 'London'
```

1.2)

```
SELECT ProductName
FROM Products
WHERE QuantityPerUnit LIKE'%bottle%' -- Not doing %bottles because there is an en-
try with just 'bottle'
```

1.3)

```
SELECT p.ProductName, s.CompanyName, s.Country
FROM Products p, Suppliers s
WHERE s.SupplierID = p.SupplierID
AND p.QuantityPerUnit LIKE'%bottle%' -- Supplier as s, used SupplierID key to match tables
```

1.4)

```
SELECT c.CategoryName, COUNT(*) AS "Number of Products in Category" --  COUNT(*) aggre-
gate function, counts all entries in category
FROM Products p
INNER JOIN Categories c ON c.CategoryID = p.CategoryID -- JOINs the two tables to-
gether, matched on CategoryID
GROUP BY c.CategoryName -- GROUPS BY the CategoryName from Categories
ORDER BY "Number of Products in Category" DESC -- Can also use ORDER BY 2 DESC, or-
der by 2nd column from the left
```

1.5)

```
SELECT TitleOfCourtesy + FirstName + ' ' + LastName AS "Names of Employees", City -- sin-
gle quotes and using string concatenation with +
FROM Employees
```

1.6)

```
SELECT t.RegionID, FORMAT(ROUND(SUM((1 - od.Discount)*od.Quantity*od.Unit-
Price), 2), '##.##') AS "Total Sales" -- desired return values, RegionID and aggre-
gated sum of prices in desired format
FROM [Order Details] od  -- start to 'traverse' through tables
INNER JOIN Orders o ON o.OrderID = od.OrderID -- OrderDetails -> Orders with key OrderID
INNER JOIN Employees e ON e.EmployeeID = o.EmployeeID -- Orders -> Employees with key Em-
ployeeID
INNER JOIN EmployeeTerritories et ON et.EmployeeID = e.EmployeeID -- Employes --> Employ-
eeTerritories with key EmployeeID
INNER JOIN Territories t ON t.TerritoryID = et.TerritoryID -- EmployeeTerritories --> Ter-
ritories with key TerritoryID
GROUP BY t.RegionID -- Now connected all tables and can group by RegionID from Territories
HAVING SUM((1 - od.Discount)*od.Quantity*od.UnitPrice) >= 1000000 -- essen-
tially WHERE but aggregate, GROUPED entries are added up and com-
pared to greater or equal to 1,000,000
ORDER BY t.RegionID -- ORDER BY ASCending default
```

1.7)

```
SELECT COUNT(Freight) AS "Orders" FROM Orders -- Aggregate used COUNT to count entries
WHERE Freight > 100 AND ShipCountry IN('USA', 'UK') -- Condi-
tion if freight greater than 100 and country is USA or UK
```

1.8)

```
SELECT TOP 1 OrderID, SUM(UnitPrice * Quantity * Discount) as "Value of Discount" -- re-
turn top 1 result, OrderID and aggregate SUM of prices
FROM [Order Details] -- [] Used because space in table name
GROUP BY OrderID -- Using OrderID to group
ORDER BY "Value of Discount" DESC -- Descending ordered, highest value appears at the top
```

2.1)

```
CREATE DATABASE alex_db -- Creates a database
USE alex_db; -- Use a database
DROP TABLE IF EXISTS spartans_table; -- Drop if exists

CREATE TABLE spartans_table(
    spartan_id INT IDENTITY(1,1) PRIMARY KEY, -- SpartanID primary key, set as INT IDEN-
TITY where starts at 1 and increments by 1.
    title VARCHAR(10), -- Variable character set to 10
    first_name VARCHAR(30),
    last_name VARCHAR(30),
    university VARCHAR(50),
    course_taken VARCHAR(50),
    marks_achieved CHAR(3) -- CHAR(3), doesn't need to be variable charac-
ter as marks can only be 2:3, 2:2 2:1 or 1st which are all 3 characters
)
```

2.2)

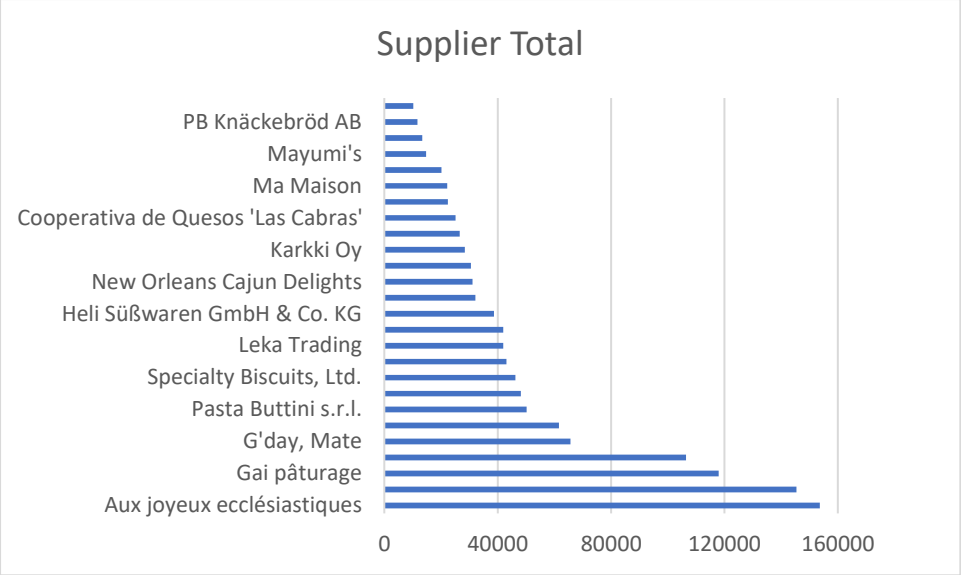
```
INSERT INTO spartans_table( -- Inserts into created table
    title, first_name, last_name, university, course_taken, marks_achieved  -- Column names
)
VALUES
(
    'Mr', 'Jakub', 'Matyjewicz', 'Poznan University of Technology', 'Technical Phys-
ics', NULL -- NULL entered because Jakub has not graduated but dropped out
),
(
    'Mr', 'Alex', 'Sikorski', 'University of Essex', 'Computer Science', '1st'
),
(
    'Mr', 'Matthew', 'Holmes', 'University of Bath', 'Computer Science and Mathemat-
ics', '2:2'
)

SELECT * FROM spartans_table
```

3.1)

```
SELECT * FROM Employees
SELECT CONCAT(e.FirstName,' ',e.LastName) AS "Employee Name", CONCAT(emp.First-
Name,' ',emp.LastName) AS "Reports To" -- Concatenates First and Last names as Em-
ployee Name and Reports to
FROM Employees e -- From Employees table
LEFT JOIN Employees emp ON e.ReportsTo = emp.EmployeeID -- Joining Employees to Employ-
ees as desired information is stored there
```

3.2)

```
SELECT s.CompanyName, SUM((1-od.Discount)*od.UnitPrice*od.Quantity) AS "Sales" -- Select-
ing Company Name and aggregate SUM of (decrease % * unit price * quantity)
FROM [Order Details] od
INNER JOIN Products p ON od.ProductID = p.ProductID -- Order Details --> Prod-
ucts with ProductID
INNER JOIN Suppliers s ON p.SupplierID = s.SupplierID -- Products --> Suppliers with Sup-
plierID
GROUP BY s.CompanyName -- GROUP BY the Company Name
HAVING SUM((1 - od.Discount)*od.Quantity*od.UnitPrice) > 10000 -
- Where the sales sum (does the calculation again) is greater than 10,000
ORDER BY "Sales" DESC -- Not necesarry but returns sales in descending order
```

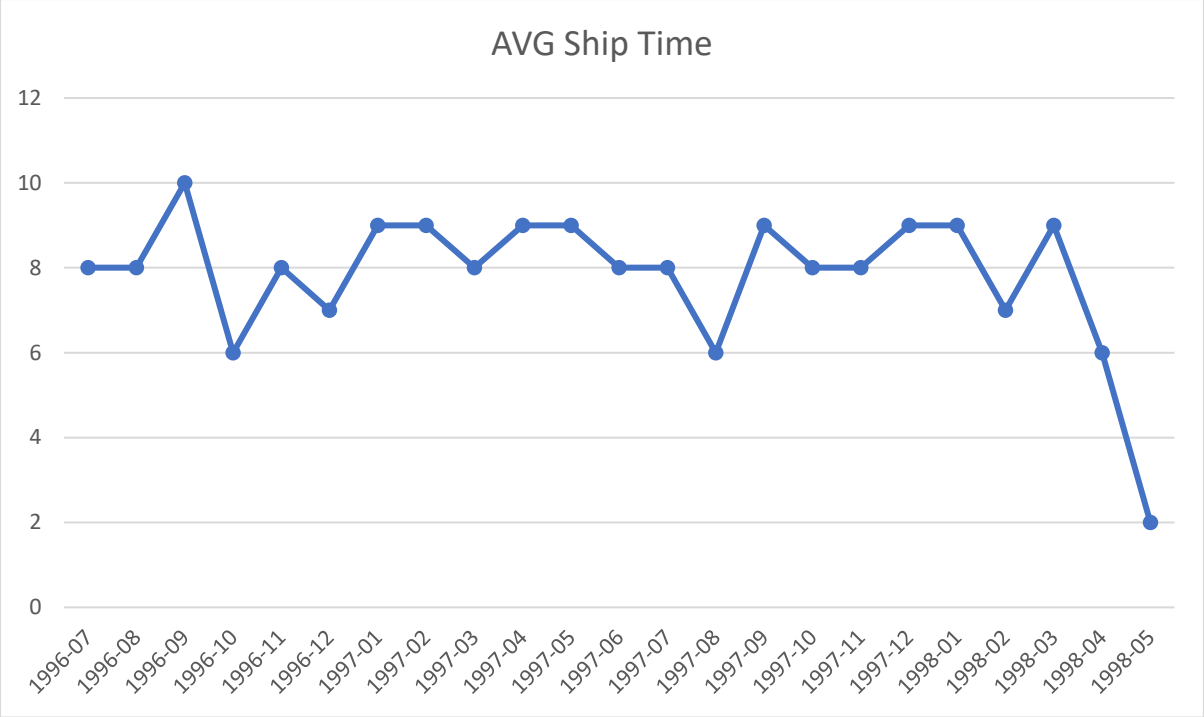


3.3)

```
SELECT TOP 10 c.CompanyName, SUM(od.Quantity*od.UnitPrice) AS "Value of Or-
ders Shipped" FROM Customers c -- Selecting top 10, SUM of value (no discount as VALUE)
INNER JOIN Orders o ON c.CustomerID = o.CustomerID -- Customers --> Orders with CustomerID
INNER JOIN [Order Details] od  ON o.OrderID = od.OrderID -- Orders --> Orders with Custom-
erID
WHERE YEAR(o.OrderDate) = YEAR((SELECT TOP 1 OrderDate FROM Orders ORDER BY Order-
Date DESC)) -- Where the year of entry ORDER date = 1998 (found by top se-
lect sub query of dates, ordered in desc)
AND o.ShippedDate IS NOT NULL -- AND when item has been shipped
GROUP BY c.CompanyName -- Grouped by the Company Name
ORDER BY 2 DESC -- Order second column in desc to show greatest value first
```

3.4)

```
SELECT FORMAT(o.OrderDate, 'yyyy-MM') AS "Year/Month", -- Selecting OrderDate in year-
month format
AVG(DATEDIFF(d,o.OrderDate, o.ShippedDate)) AS "AVG Ship Time" FROM Orders o -- Aggre-
gate AVG function with DATEDIFF, so calculating datediff and average based off year-
month groups
GROUP BY FORMAT(o.OrderDate, 'yyyy-MM') -- Grouping by yyyy-MM
```


AVG Ship Time
12
10
8
6
4
2
0
1996-07
1996-08
1996-09
1996-10
1996-11
1996-12
1997-01
1997-02
1997-03
1997-04
1997-05
1997-06
1997-07
1997-08
1997-09
1997-10
1997-11
1997-12
1998-01
1998-02
1998-03
1998-04
1998-05