Use the database company.accdb and finish the following exercises on SQL.

<table>
<thead>
<tr>
<th>Emp</th>
<th>LName</th>
<th>age</th>
<th>salary</th>
<th>Dnum</th>
<th>wYears</th>
</tr>
</thead>
<tbody>
<tr>
<td>E101</td>
<td>Jones</td>
<td>45.00</td>
<td>$56,000.00</td>
<td>D25</td>
<td>12</td>
</tr>
<tr>
<td>E202</td>
<td>Anders</td>
<td>66.00</td>
<td>$46,000.00</td>
<td>D22</td>
<td>25</td>
</tr>
<tr>
<td>E303</td>
<td>Smith</td>
<td>34.00</td>
<td>$25,000.00</td>
<td>D22</td>
<td>9</td>
</tr>
<tr>
<td>E404</td>
<td>Rivera</td>
<td>22.00</td>
<td>$30,000.00</td>
<td>D25</td>
<td>1</td>
</tr>
<tr>
<td>E505</td>
<td>Brown</td>
<td>45.00</td>
<td>$80,000.00</td>
<td>D25</td>
<td>17</td>
</tr>
<tr>
<td>E606</td>
<td>Caldwell</td>
<td>52.00</td>
<td>$70,000.00</td>
<td>D28</td>
<td>20</td>
</tr>
<tr>
<td>E707</td>
<td>Stiles</td>
<td>44.00</td>
<td>$65,000.00</td>
<td>D28</td>
<td>11</td>
</tr>
<tr>
<td>E808</td>
<td>Walker</td>
<td>48.00</td>
<td>$90,000.00</td>
<td>D22</td>
<td>21</td>
</tr>
<tr>
<td>E909</td>
<td>Hartman</td>
<td>20.00</td>
<td>$25,000.00</td>
<td>D28</td>
<td>1</td>
</tr>
<tr>
<td>E222</td>
<td>Carter</td>
<td>29.00</td>
<td>$35,000.00</td>
<td>D25</td>
<td>3</td>
</tr>
</tbody>
</table>

Each select Query involving one table has this basic form:

```
select (list of attributes) 
from (table name) 
where (some condition which can be applied to each row) 
order by (some attribute) ;
```

Note that the “Order by” clause and the “where” condition are optional.

The condition is either a simple condition (value op. value) or a complex condition that is defined in one of the following formats:

```
not (sub condition) 
(sub condition) and (sub condition) 
(sub condition) or (sub condition)
```

The sub-condition can recursively be defined as the condition, falling into the definition of simple or complex condition.

The use of condition (where clause) is to ensure those wanted rows can be selected (by a ‘true’ value) and those unwanted rows be excluded out (by a ‘false’ value). So those selected rows and only selected rows make the condition true. The following is the truth table for the condition evaluation:

<table>
<thead>
<tr>
<th>C1</th>
<th>T</th>
<th>T</th>
<th>F</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>C1 and C2</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>C1 or C2</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>Not C1</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

You may gain some ideas of condition design from the following sample.
Consider the above table and its selected rows with a check value ‘n’

<table>
<thead>
<tr>
<th>Selected ‘n’</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1, 2, 3, 4, 5}</td>
<td>n &lt; 6</td>
</tr>
<tr>
<td>{3, 4, 5}</td>
<td>n &gt; 2 and n &lt; 6</td>
</tr>
<tr>
<td>{3, 4, 5} U {8, 9}</td>
<td>(n &gt; 2 and n &lt; 6) or ( n &gt; 7 and n &lt; 10)</td>
</tr>
<tr>
<td>{3, 4, 5} U {8, 9}</td>
<td>n&gt;2 and n &lt; 10</td>
</tr>
<tr>
<td>{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}</td>
<td>(none)</td>
</tr>
</tbody>
</table>

The following is a select query written in SQL. Different database systems use slightly different versions.

```sql
SELECT [Enum], [Lname], [salary], [age]
FROM emp
WHERE (Emp.age)>50
ORDER BY Lname;
```

When we create a query in Design view, Access automatically creates an SQL version of that query.

- When Access refers to the name of an attribute, it uses the form `tablename.attribute`.
- Brackets are used for attribute names when they consist of more than 1 word.
- The “where” clause has extra sets of parenthesis.

Here is the Sql created by Access

```sql
SELECT Emp.Enum, Emp.LName, Emp.salary, Emp.age
FROM Emp
WHERE ((Emp.age)>50))
ORDER BY Emp.LName;
```
The following SQL query will list all employee numbers, names, ages and monthly salaries, for employees in ‘D22’. Note the need for quotes for any attribute of type text.

```sql
SELECT [Enum], [Lname] AS [Employee Name], [age], [Salary]/12 AS [Monthly Salary]
FROM Emp
WHERE [Dnum]='D22';
```

We can use the “AND” “OR” and “NOT” operators in the where clause as shown below.

- What would be the question that each following query answers?

<table>
<thead>
<tr>
<th>Query</th>
<th>Description</th>
</tr>
</thead>
</table>
| ```sql
SELECT [Enum], [Lname], [age], [Dnum]
FROM Emp
WHERE [age] >19 AND [age]<30;
``` |
| ```sql
SELECT *
FROM Emp
WHERE NOT [Dnum] = 'D21';
``` |
| ```sql
SELECT [Enum], [Lname], [age], [Dnum]
FROM Emp
WHERE [Lname] Like 'A*';
``` |

See how to choose those rows from the table with the Boolean checking in where clause.

- Try to create the following queries by SQL statements (in the SQL view) - Query involving routine operations, such as wild card, compound comparison, sorting, rename of column name, calculated values, parameter, etc.

  1) List the **LName** and **age** of all employees whose name ends in “an”.
  2) List the **Enum** and **LName** of all employees in dept “D22” and making more than 40,000.
  3) List the **Enum, LName, age, and Dnum**, for all employees in Dept “D22” or Dept “D28” sorted by **LName**. Display the field Enum as “Number” and the field LName as “Emp Name”.
  4) List the **Enum, LName, age**, and a calculated column “Years till retirement” (calculated as the number of years to reach age 70), for all employees over 40.
  5) List the **LName, Enum, and Salary** of all employees who are over the age entered by the user.

**SQL statements involving statistics and aggregation.**

Consider using the following SQL statement to access the information in a table.

```sql
select (field grouped by, and a list of functions based on attributes)
from (table name)
where (any conditions to be applied to each row before grouping)
group by (field used to group by)
having (any conditions to be applied to each group)
order by (field used to group by)
```
• We need “group by” when we are looking for statistical values of rows in group (refer to phase 3 in lab3).
• “group by” is not necessary when the statistical value comes from all rows in the table as one group.
• Instead of counting every row in each group, we will use “where” to restrict which rows are used before grouping – Filter, refer to phase 2 in lab 3.
• When we are only interested in groups that meet certain conditions, “having” will be used – refer to phase 4 in lab 3. This incurs a major difference between SQL programming and ACCESS implementation!

> Check the different use of where and having.

a) SELECT max(B) as [Max B] FROM T;

b) SELECT max(B) as [Max B] FROM T GROUP BY C;

c) SELECT max(B) as [Max B] FROM T GROUP BY C HAVING max(B) > 50;

d) SELECT max(B) as [Max B] FROM T GROUP BY C HAVING C like “PA” or C like ”DE”;

? why C and max can be used in having clause?


c) SELECT max(B) as [Max B]
FROM T
WHERE B < 50
GROUP BY C;

why CANNOT use max(b) in where clause?

✓ Indicate the result of the query

a) SELECT C, max(B) AS [Max B], count(Id) AS [Count]
FROM T
GROUP BY C;

b) SELECT C, max(B) AS [Max B], count(Id) AS [Count]
FROM T
WHERE A>5
GROUP BY C;

c) SELECT C, max(B) AS [Max B], count(Id) AS [Count]
FROM T
WHERE A>5
GROUP BY C
HAVING count(Id)>2;

✓ Write Sql statements for each of the following and experience the above evolution
with aggregations.

d) For each D, show D, the min A and Max B, and a count of rows.

e) For each D, show D, min A Max B and a count of these rows that have a B value above 40.

f) For each D that has 2 or more rows where B is above 40, show D, min A, Max B and a count of these rows

✓ Homework - What would be the question that each following SQL query answers?

```sql
<table>
<thead>
<tr>
<th>SQL Statement</th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT [Dnum], Avg(salary) AS [Avg salary], Count(Enum) AS [Employee Count] FROM Emp GROUP BY Dnum ORDER BY Dnum;</td>
<td>What is the average salary and employee count for each department, ordered by department number?</td>
</tr>
<tr>
<td>SELECT [Dnum], Avg(salary) AS [Avg salary], Count(Enum) AS [Employee Count] FROM Emp where [age] &gt; 40 GROUP BY Dnum ORDER BY Dnum;</td>
<td>What is the average salary and employee count for departments with employees older than 40, ordered by department number?</td>
</tr>
<tr>
<td>SELECT [Dnum], Avg(salary) AS [Average salary], Count(Enum) AS [Employee Count] FROM Emp where [age] &gt;40 GROUP BY Dnum Having count(enum) &gt;2 ORDER BY Dnum;</td>
<td>What is the average salary and employee count for departments with at least 2 employees older than 40, ordered by department number?</td>
</tr>
</tbody>
</table>
```
Try to create the following queries by SQL statements (in the SQL view)

6) For each Dnum list the number of employees and their average of salary.
7) For each Dnum list the number of employees above 30 and their average salaries
8) For each Dnum list the number of employees and average salaries, but only include Depts with more than 3 employees.
9) For each Dnum with 2 or more than 2 employees younger than 50 but working at least 10 years, list the Dnum along with the count of these employees and their average age.

### Queries involving multiple tables

<table>
<thead>
<tr>
<th>Emp</th>
<th>Dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enum</td>
<td>LName</td>
</tr>
<tr>
<td>E101</td>
<td>Jones</td>
</tr>
<tr>
<td>E202</td>
<td>Anders</td>
</tr>
<tr>
<td>E303</td>
<td>Smith</td>
</tr>
<tr>
<td>E404</td>
<td>Rivera</td>
</tr>
<tr>
<td>E505</td>
<td>Brown</td>
</tr>
<tr>
<td>E606</td>
<td>Caldwell</td>
</tr>
<tr>
<td>E707</td>
<td>Stiles</td>
</tr>
<tr>
<td>E808</td>
<td>Walker</td>
</tr>
<tr>
<td>E909</td>
<td>Hartman</td>
</tr>
<tr>
<td>E222</td>
<td>Carter</td>
</tr>
</tbody>
</table>

Suppose we need to get information from both tables. Notice there is a chance to create a lot of redundant information. We need to imagine/create one table from the data in both tables, by matching rows with the same value in the common column. Note that each employee is lined up with info for the department that they work in by matching values in the Dnum column.

<p>| Select Emp, Dept where Emp.Dnum = Dept.Dnum |</p>
<table>
<thead>
<tr>
<th>Enum</th>
<th>LName</th>
<th>age</th>
<th>salary</th>
<th>Dnum</th>
<th>Dname</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>E202</td>
<td>Anders</td>
<td>66.00</td>
<td>$46,000.00</td>
<td>D22</td>
<td>Web</td>
<td>WC</td>
</tr>
<tr>
<td>E303</td>
<td>Smith</td>
<td>34.00</td>
<td>$25,000.00</td>
<td>D22</td>
<td>Web</td>
<td>WC</td>
</tr>
<tr>
<td>E808</td>
<td>Walker</td>
<td>48.00</td>
<td>$90,000.00</td>
<td>D22</td>
<td>Web</td>
<td>WC</td>
</tr>
<tr>
<td>E101</td>
<td>Jones</td>
<td>45.00</td>
<td>$56,000.00</td>
<td>D25</td>
<td>Databases</td>
<td>NY</td>
</tr>
<tr>
<td>E404</td>
<td>Rivera</td>
<td>22.00</td>
<td>$30,000.00</td>
<td>D25</td>
<td>Databases</td>
<td>NY</td>
</tr>
<tr>
<td>E505</td>
<td>Brown</td>
<td>45.00</td>
<td>$80,000.00</td>
<td>D25</td>
<td>Databases</td>
<td>NY</td>
</tr>
<tr>
<td>E222</td>
<td>Carter</td>
<td>29.00</td>
<td>$35,000.00</td>
<td>D25</td>
<td>Databases</td>
<td>NY</td>
</tr>
<tr>
<td>E606</td>
<td>Caldwell</td>
<td>52.00</td>
<td>$70,000.00</td>
<td>D28</td>
<td>Software</td>
<td>LA</td>
</tr>
<tr>
<td>E707</td>
<td>Stiles</td>
<td>44.00</td>
<td>$65,000.00</td>
<td>D28</td>
<td>Software</td>
<td>LA</td>
</tr>
<tr>
<td>E909</td>
<td>Hartman</td>
<td>20.00</td>
<td>$25,000.00</td>
<td>D28</td>
<td>Software</td>
<td>LA</td>
</tr>
</tbody>
</table>
Once we conceive the data from both tables as one table, then we just choose the columns we need and the condition to apply to each row. This is the same as for single tables.

Consider the following query:
For each employee over 30, list number, name, age, dname and city.
This involves getting data from two different tables. To accomplish this we need to:
- create one table from the two tables involved
- choose the columns we want (from both tables)
- apply any conditions to be applied to each row of the joined table

In SQL we DO NOT need to actually type the word “Join”, we can do this using a where clause.

The form used is as follows

```
Select (list of attributes)
From   (list of tables)
where    table1.commonCol=table2.commonCol (for each table that is related)
         AND ( list of conditions to be applied to the final joint table)
```

Here is an example: For each employee over 30, list number, name, age, dept name and city

```
SELECT [enum],[Lname],[age],[Dname],[city]
FROM Dept, Emp
where    Dept.[Dnum]=Emp.[dnum]
    and age>30;
```

What is the meaning of this query?

```
SELECT [Dnum],[city],[enum],[Lname],[salary]
FROM Dept, Emp
where    Dept.[Dnum]=Emp.[dnum]
    and Dept.[Dname]='Sales';
```

Create the following queries by SQL. (Click the datasheet view to check the expected result.)

10) For D22 list the Dname, city, and each employee (enum, age, salary) in that department working more than 10 years.
11) For any department located in ‘NY’, list the count of the employees who work in such a department.

Summary

```
Select   (list of attributes)
From     (table name)
where    (some condition which can be applied to each row)
group by (group division)
having   (any condition to be applied to select groups)
Order by (some attribute ) [asc/desc] ;
```

1. Select display
2. Find the places that contain such information (from tables)
3. Check attribute name unique
a) add table name (e.g., [emp].xxx) if necessary
b) add table connector in **where** expression when multiple tables are used

4. Ensure the **group** division when set/group value is needed (but not for the entire database, see why?) by key attributes

5. Before the group division, make sure a filter is set within **where** expression: and, or, not; in conjunction with multiple table joint

6. Group selection with “having” by the group value such as avg, min, or max (**having**)

7. Order