Mervan Kaya, 000605-6352, mckaya@kth.se
Samhar Alzghaier, xxxxxx-xxxx, samhara@kth.se

Lab 1

1. and 2.

```
/* ALTER TABLE Country
RENAME COLUMN Name to CountryName;
*/

SELECT CountryName, Area, Population
FROM Country
WHERE Population < 1000;
```

The first two lines of the query specifies changing the table. We query that we want to change the table, and then specify what is going to be changed to what.

After that, we query the projection of the attributes we want, choose from which relation, and then a conditional select where we only choose tuples whose population is less than 1000. This can be written in relational algebra as:

$$\pi_{CountryName,Area,Population}(\sigma_{Population<1000}(Country))$$

The output is thus the following:

Note the changed attribute of name.

	CountryName	Area	Population
1	Holy See	0.44	842
2	Cocos Islands	14	596
3	Pitcairn	47	56

3.

```
SELECT CountryName, Population, Area
FROM Country
WHERE Population > 1000 AND Population < 2000 AND Area >= 1.0;
```

This query is very much alike the previous query, only that we have three conditions that have to be *True* for the tuple to be chosen. This cam be written as:

$$\pi_{CountryName,Area,Population}(\sigma_{Population>1000\ AND\ Population<2000\ AND\ Area\geq 1.0}(Country))$$

The output is:

	CountryName	Population	Area
1	Svalbard	1872	62049
2	Niue	1611	260
3	Tokelau	1383	12

4.

```
SELECT Code
FROM Country
WHERE CountryName = "Norway";
```

Mervan Kaya, 000605-6352, mckaya@kth.se
Samhar Alzghaier, xxxxxx-xxxx, samhara@kth.se

This query is again very similar to the ones above, only slight changes have been made to the conditions of the operators. This can be written as:

$$\pi_{Code}(\sigma_{CountryName=Norway}(Country))$$

The output is as following:

	Code
1	N

5.

```
SELECT Country, Name, Population
FROM City
WHERE Country = "S" and Population > 500000;
```

Again, a very similar query. We project Country, Name, Population of City, and then using a conditional select we choose which tuples we want. We analyzed the elements of the attribute Country in order to find how Sweden is represented. This can be written as:

$$\pi_{Country,Name,Population}(\sigma_{Country=S\ AND\ Population>500000}(City))$$

Output is following:

	Country	Name	Population
1	S	Göteborg	526089
2	S	Stockholm	881235

6.

```
SELECT Name, Population, Elevation
FROM City
WHERE Elevation < 0;
```

Again, a very similar query as the ones previously. Can be written as:

$$\pi_{Name,Population,Elevation}(\sigma_{Elevation<0}(City))$$

The output is:

#	Name	Population	Elevation
1	Astrakhan	527345	-28
2	Lelystad	76252	-3
3	Almere	196244	-3
4	Babol	250217	-2
5	Baku	2150800	-28
6	Atyrau	196494	-20
7	Aktau	181526	-8
8	New Orleans	343829	-2
9	David	144858	-6
10	Georgetown	118363	-2

Mervan Kaya, 000605-6352, mckaya@kth.se
Samhar Alzghaier, xxxxxx-xxxx, samhara@kth.se

7.

```
SELECT SUM(Population) as total, AVG(Population) as average,
MIN(Population) as minimum, MAX(Population) as maximum
FROM City
WHERE Elevation < 0
```

Here, aggregation is used to find the sum, avg, min and max of the attribute Population. This is very is easy to do as you can simply write SUM(attributeName) and so on.

The output is:

	total	average	minimum	maximum
1	4185928	418592.8	76252	2150800

8. Without EXCEPT

```
SELECT Name
FROM City
WHERE Name LIKE 'los%' AND Name NOT LIKE '%is' or Name LIKE
'%holm'
```

We first project the name and choose the relation. Next, we have to define the conditional selection of the tuples we want. We can use LIKE is combination with ‘%someString’ or ‘someString%’ to choose Names that end with a certain string or begins with a certain string. Here, we specify strings that begin with “los” but do not end ‘is’ or strings that end with ‘holm’. The order is of the conditions is important as you can get different answers otherwise.

The output is:

	Name
1	Los Angeles
2	Los Teques
3	Stockholm

8. With EXCEPT

```
SELECT Name
FROM City
WHERE Name LIKE 'los%' or Name LIKE '%holm'
EXCEPT
Select Name
From City
WHERE Name LIKE '%is'
```

We can use except to merge two queries and to create and exception of the output of the first query. We proceed with the first query as normal and create output tuples, however we add exception to the output tuples that match the tuples generated by the exception query. Thus, we get the same output as the other iteration of problem 8.

Mervan Kaya, 000605-6352, mckaya@kth.se
Samhar Alzghaier, xxxxxx-xxxx, samhara@kth.se

9.

```
SELECT CountryName, Population
FROM Country
ORDER BY Population DESC
LIMIT 5;
```

By using order, we can order the output table by as we want it. Here, we ordered it according to the Population attribute and in a descending fashion. We also limited the output table to only 5 tuples by using LIMIT so we only see the 5 biggest populations of the world.

The output is following:

	CountryName	Population
1	China	1360720000
2	India	1210854977
3	United States	318857056
4	Indonesia	252124458
5	Pakistan	207776954

10.

```
SELECT Country.CountryName, City.Name, City.Elevation
FROM City
INNER JOIN Country
ON City.Country = Country.Code
WHERE Elevation IS NOT NULL
ORDER BY Elevation ASC
LIMIT 5;
```

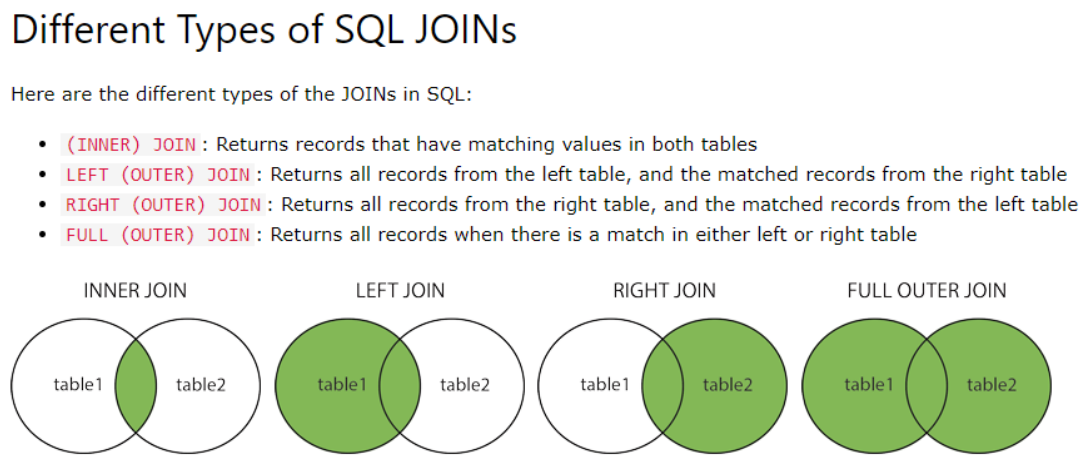

Figure 1 *Joins, W3Schools, https://www.w3schools.com/sql/sql_join.asp*

Here we want to combine information from different tables. We want to join tuples which share the same country code as each other. To do this, we have to use INNER JOIN as it only includes tuples which share the same country code in both City and Country. We also, specify

that we do not want to include any Null values for elevation and that we want to order elevation in an ascending fashion. We also limit the result to only the 5 lowest elevations.

The output is following:

	CountryName	Name	Elevation
1	Russia	Astrakhan	-28
2	Azerbaijan	Baku	-28
3	Kazakhstan	Atyrau	-20
4	Kazakhstan	Aktau	-8
5	Panama	David	-6

11.

```
SELECT Name
FROM City
WHERE Name LIKE '%x'
UNION ALL
SELECT CountryName as Name
FROM Country
WHERE Name LIKE 'Y%'
```

Here we wanted to combine the entire tables of Name and Country. This is best done by using UNION as it combines the tables (It takes the union of the sets). We can thus utilize it to find cities ending with ‘x’ or countries starting with ‘Y’.

The output is the following:

	Name
1	Bordeaux
2	Halifax
3	Jizzax
4	Phoenix
5	Sfax
6	Yemen