# Causes of Death STEP 3

Group AKAA

Alaa Almouradi  00030123

Filzah Azeem 00030138

Fatma Khalil 00029999

M Safwan Yasin 00030037

Saleh Alshurafa 00030002

CS306 Spring 22-23

GitHub Repository

## Preface:

Given that not all parts of this step needed to be done by each group member, the workload might not seem equally divided. But we have all worked together even if not on our respectively named files.

## Food Management (Alaa Almouradi):

Everything is in the folder named Alaa Almouradi STEP3 (Food)

- 1a:

  1a.sql:

```
CREATE VIEW meatmanagementpercountry AS
    SELECT
        *
    FROM
        foodmanagementpercountry F
    WHERE
        F.typeOfFood LIKE 'Meat';


CREATE VIEW vegetablesmanagementpercountry AS
    SELECT
        *
    FROM
        foodmanagementpercountry F
    WHERE
        F.typeOfFood LIKE 'Vegetables';
```

```
SELECT
    M.year,
    M.countryCode,
    M.production_1000_tonnes AS Meat_production_1000_tonnes,
    M.domestic_supply_quantity_1000_tonnes AS
Meat_domestic_supply_quantity_1000_tonnes,
    V.production_1000_tonnes AS Veg_production_1000_tonnes,
    V.domestic_supply_quantity_1000_tonnes AS
Veg_domestic_supply_quantity_1000_tonnes
FROM
    cs306project.meatmanagementpercountry M
        INNER JOIN
    cs306project.vegetablesmanagementpercountry V ON M.countryCode =
V.countryCode
        AND M.year = V.year;
```

I created a view that shows the management of meat only and another view that shows management of vegetables only, as those two categories are important.

- 1b:

1b.sql:

```
# years with the countries that were above average production of meat in them
(SELECT
```

```
    M.year, M.countryCode
FROM
    cs306project.meatmanagementpercountry M
        JOIN
    (SELECT
        AVG(production_1000_tonnes) AS Avgprod, year
    FROM
        cs306project.meatmanagementpercountry M
    GROUP BY year) b ON M.year = b.year
WHERE
    M.production_1000_tonnes > b.Avgprod)

EXCEPT

# years with the countries that were above average production of vegetables in them
(SELECT
    V.year, V.countryCode
FROM
    cs306project.vegetablesmanagementpercountry V
        JOIN
    (SELECT
        AVG(production_1000_tonnes) AS Avgprod, year
    FROM
```

```
        cs306project.vegetablesmanagementpercountry V
    GROUP BY year) b ON V.year = b.year
WHERE
    V.production_1000_tonnes > b.Avgprod);



SELECT A.year, A.countryCode
FROM
(SELECT
    M.year, M.countryCode
FROM
    cs306project.meatmanagementpercountry M
        JOIN
    (SELECT
        AVG(production_1000_tonnes) AS Avgprod, year
    FROM
        cs306project.meatmanagementpercountry M
    GROUP BY year) b ON M.year = b.year
WHERE
    M.production_1000_tonnes > b.Avgprod) A
LEFT JOIN
(SELECT
    V.year, V.countryCode
```

```
FROM
    cs306project.vegetablesmanagementpercountry V
        JOIN
    (SELECT
        AVG(production_1000_tonnes) AS Avgprod, year
    FROM
        cs306project.vegetablesmanagementpercountry V
    GROUP BY year) b ON V.year = b.year
WHERE
    V.production_1000_tonnes > b.Avgprod) B

ON A.year = B.year AND A.countryCode = B.countryCode
WHERE B.countryCode IS NULL;
```

I made two sets that represent the countries in each year that produced above average in each of meat and vegetables categories and used except and then did the same functionality with right join (proof of this in the LOGS.log file). The result of either operation shows the countries in each year that are above average only in meat production.

- 1c:

  1c.sql:

```
SELECT
    M.year, M.countryCode
FROM
```

```
    meatmanagementpercountry M
WHERE
    M.year = 2000
        AND M.production_1000_tonnes IN (SELECT
            M2.production_1000_tonnes
        FROM
            meatmanagementpercountry M2
        WHERE
            M2.year = 2000
                AND M.production_1000_tonnes > (SELECT
                    AVG(M2.production_1000_tonnes)
                FROM
                    meatmanagementpercountry M2
                WHERE
                    M2.year = 2000));

SELECT
    M.year, M.countryCode
FROM
    meatmanagementpercountry M
WHERE
    EXISTS( SELECT
            M2.production_1000_tonnes
```

```
    FROM
        meatmanagementpercountry M2
    WHERE
        M2.year = 2000 AND M.year = 2000
            AND M.production_1000_tonnes = M2.production_1000_tonnes
            AND M.production_1000_tonnes > (SELECT
                AVG(M3.production_1000_tonnes)
            FROM
                meatmanagementpercountry M3
            WHERE
                M3.year = 2000));
```

I used IN and EXIST to get the countries that are above average in the year 2000 in meat production. (proof of this in the LOGS.log file)

- 1d: explanation after all sqls

  1d part1.sql:

```
SELECT
    M.year,
    M.countryCode,
    M.production_1000_tonnes AS Meat_production_1000_tonnes,
    M.domestic_supply_quantity_1000_tonnes AS
Meat_domestic_supply_quantity_1000_tonnes,
    V.production_1000_tonnes AS Veg_production_1000_tonnes,
```

```
    V.domestic_supply_quantity_1000_tonnes AS
Veg_domestic_supply_quantity_1000_tonnes
FROM
    cs306project.meatmanagementpercountry M
        INNER JOIN
    cs306project.vegetablesmanagementpercountry V ON M.countryCode =
V.countryCode
        AND M.year = V.year
WHERE
    M.production_1000_tonnes IN (SELECT
            MAX(M2.production_1000_tonnes)
        FROM
            cs306project.meatmanagementpercountry M2
                INNER JOIN
            cs306project.vegetablesmanagementpercountry V2 ON M2.countryCode =
V2.countryCode
                AND M2.year = V2.year
        GROUP BY M2.year
        HAVING (SUM(M2.production_1000_tonnes) -
SUM(M2.domestic_supply_quantity_1000_tonnes)) >
(SUM(V2.production_1000_tonnes) -
SUM(V2.domestic_supply_quantity_1000_tonnes)))
```

1d part2.sql:

```
SELECT
    M.year,
    M.countryCode,
    M.production_1000_tonnes AS Meat_production_1000_tonnes,
    M.domestic_supply_quantity_1000_tonnes AS
Meat_domestic_supply_quantity_1000_tonnes,
    V.production_1000_tonnes AS Veg_production_1000_tonnes,
    V.domestic_supply_quantity_1000_tonnes AS
Veg_domestic_supply_quantity_1000_tonnes
FROM
    cs306project.meatmanagementpercountry M
        INNER JOIN
    cs306project.vegetablesmanagementpercountry V ON M.countryCode =
V.countryCode
        AND M.year = V.year
WHERE
    V.production_1000_tonnes IN (SELECT
            MAX(V2.production_1000_tonnes)
        FROM
            cs306project.meatmanagementpercountry M2
                INNER JOIN
            cs306project.vegetablesmanagementpercountry V2 ON M2.countryCode =
V2.countryCode
```

```
            AND M2.year = V2.year
        GROUP BY M2.year
        HAVING (SUM(M2.production_1000_tonnes) -
SUM(M2.domestic_supply_quantity_1000_tonnes)) <
(SUM(V2.production_1000_tonnes) -
SUM(V2.domestic_supply_quantity_1000_tonnes)))
```

1d part3.sql:

```
SELECT
    M.year,
    M.countryCode,
    M.production_1000_tonnes AS Meat_production_1000_tonnes,
    M.domestic_supply_quantity_1000_tonnes AS
Meat_domestic_supply_quantity_1000_tonnes,
    V.production_1000_tonnes AS Veg_production_1000_tonnes,
    V.domestic_supply_quantity_1000_tonnes AS
Veg_domestic_supply_quantity_1000_tonnes
FROM
    cs306project.meatmanagementpercountry M
        INNER JOIN
    cs306project.vegetablesmanagementpercountry V ON M.countryCode =
V.countryCode
        AND M.year = V.year
WHERE
```

```
    M.production_1000_tonnes IN (SELECT
        MAX(M2.production_1000_tonnes)
      FROM
        cs306project.meatmanagementpercountry M2
          INNER JOIN
        cs306project.vegetablesmanagementpercountry V2 ON M2.countryCode =
V2.countryCode
          AND M2.year = V2.year
      WHERE
        M2.countryCode NOT LIKE 'CHN'
      GROUP BY M2.year
      HAVING (SUM(M2.production_1000_tonnes) -
SUM(M2.domestic_supply_quantity_1000_tonnes)) >
(SUM(V2.production_1000_tonnes) -
SUM(V2.domestic_supply_quantity_1000_tonnes)))
```

1d part4.sql:

```
SELECT
    M.year,
    M.countryCode,
    M.production_1000_tonnes AS Meat_production_1000_tonnes,
    M.domestic_supply_quantity_1000_tonnes AS
Meat_domestic_supply_quantity_1000_tonnes,
    V.production_1000_tonnes AS Veg_production_1000_tonnes,
```

```
    V.domestic_supply_quantity_1000_tonnes AS
Veg_domestic_supply_quantity_1000_tonnes
FROM
    cs306project.meatmanagementpercountry M
        INNER JOIN
    cs306project.vegetablesmanagementpercountry V ON M.countryCode =
V.countryCode
        AND M.year = V.year
WHERE
    V.production_1000_tonnes IN (SELECT
            MAX(V2.production_1000_tonnes)
        FROM
            cs306project.meatmanagementpercountry M2
                INNER JOIN
            cs306project.vegetablesmanagementpercountry V2 ON M2.countryCode =
V2.countryCode
                AND M2.year = V2.year
        WHERE
            M2.countryCode NOT LIKE 'CHN'
        GROUP BY M2.year
        HAVING (SUM(M2.production_1000_tonnes) -
SUM(M2.domestic_supply_quantity_1000_tonnes)) <
```

```
(SUM(V2.production_1000_tonnes) -
SUM(V2.domestic_supply_quantity_1000_tonnes)))
```

1d part5.sql:

```
SELECT
    M.year,
    M.countryCode,
    M.production_1000_tonnes AS Meat_production_1000_tonnes,
    M.domestic_supply_quantity_1000_tonnes AS
Meat_domestic_supply_quantity_1000_tonnes,
    V.production_1000_tonnes AS Veg_production_1000_tonnes,
    V.domestic_supply_quantity_1000_tonnes AS
Veg_domestic_supply_quantity_1000_tonnes
FROM
    cs306project.meatmanagementpercountry M
        INNER JOIN
    cs306project.vegetablesmanagementpercountry V ON M.countryCode =
V.countryCode
        AND M.year = V.year
WHERE
    V.production_1000_tonnes IN (SELECT
            MAX(V2.production_1000_tonnes)
        FROM
            cs306project.meatmanagementpercountry M2
```

```
            INNER JOIN
        cs306project.vegetablesmanagementpercountry V2 ON M2.countryCode =
V2.countryCode
            AND M2.year = V2.year
    WHERE
        M2.countryCode NOT LIKE 'CHN'
    GROUP BY M2.year)
```

In part 1 I get max producers of meat per year in years where net meat production was more than net vegetable production

Part 2 max producers of vegetables per year in years where net vegetables production was more than net meat production

From those two parts China proves dominating in both categories all years except in meat production year 1990 which was the USA. So I made the same but excluding China in part 3 and 4 and it turns out the USA dominates all years in meat production. And there exists no year where net vegetable production was more than net meat production without china so part 4 returns 0 rows so I made part 5 which just gets the max producers of vegetables per year excluding without that net condition and turns out India dominates all years.

**M Safwan Yasin MYSQL Statements STEP3.sql**

a. **Views**

The following view that would calculate the average of DALY from 1990 to 2015 and sort them in descending order of country. This showed the countries where the DALY is high thus highlighting the countries where people on average lose years of good health due to disease

```
CREATE VIEW avgDalyPerCountrySorted AS
SELECT c.countryName, AVG(DALY) AS avgDaly
FROM mentalIllnessPerCountry m, countries c
WHERE c.countryCode = m.countryCode
GROUP BY m.countryCode
ORDER BY avgDaly DESC;
```

**Aggregate operators and joins**

This select statement finds the year with the minimum value of prevalenceAlcoholUseDisorder_Percent for each countryCode from the table called substanceAbusePerCountry and then finds the numberOfDeaths with cause = Alcohol Use Disorders in the year with the minimum value of prevalenceAlcoholUseDisorder_Percent.

```
SELECT
  sac.countryCode,
  sac.Year AS YearWithMinPrevalence,
  sac.prevalenceAlcoholUseDisorder_Percent AS MinPrevalence,
  d.numberOfDeaths AS NumberOfDeathsWithMinPrevalence
FROM
  substanceAbusePerCountry sac
```

```
JOIN
  Deaths d ON sac.countryCode = d.countryCode AND sac.Year = d.Year
WHERE
  sac.prevalenceAlcoholUseDisorder_Percent = (
    SELECT
      MIN(prevalenceAlcoholUseDisorder_Percent)
    FROM
      substanceAbusePerCountry
    WHERE
      countryCode = sac.countryCode
  )
  AND d.cause = 'Alcohol Use Disorders'
GROUP BY
  sac.countryCode,
  sac.Year,
  sac.prevalenceAlcoholUseDisorder_Percent
HAVING sac.prevalenceAlcoholUseDisorder_Percent <= 1;
```

## Death:

**Fatma Khalil MYSQL Statements STEP3 (Death).sql**

**a-Views:**

1) CREATE VIEW DeathByAlcoholDisorders AS

```
SELECT
   countryCode, cause,
   SUM(numberOfDeaths) AS totalDeaths
FROM
   deaths
WHERE
   cause = 'Alcohol Use Disorders'
GROUP BY countryCode;
```

2) CREATE VIEW DeathByDrugs AS

```
SELECT
   countryCode, cause, SUM(numberOfDeaths) AS totalDeaths
FROM
   deaths
WHERE
   cause = 'Drug Use Disorders'
GROUP BY countryCode;
```

I created two views to see how much death is caused by alcohol and drug misuse. These views were chosen in order to see how many deaths are caused by substance misuse, since from all the causes in our datasets, these two causes were the only ones that were self-inflicted.

**b- Join and Set Operators:**

SELECT D.countryCode, D.Year, D.cause, D.numberOfDeaths FROM deaths D, deathbyalcoholdisorders A where D.cause != A.cause

INTERSECT

```
SELECT D.countryCode, D.Year, D.cause, D.numberOfDeaths FROM deaths D, deathbydrugs S where
D.cause = S.cause;
SELECT DISTINCT
    D.countryCode, D.Year, D.cause, D.numberOfDeaths
FROM
    deaths D
        INNER JOIN
    deathbydrugs S ON D.cause = S.cause;
```

I used the set operator INTERSECT and the join operator INNER JOIN and made the select statement choose all of the information about the cause of death by drug misuse. The view created earlier only shows the total deaths from drug misuse of a country but without stating how much deaths happened in a year, so this select statement shows the amount of deaths caused by drugs each year from each country from the year 1990 to 2019. Then I modified the select statement with INTERSECT in order to support the INNER JOIN operator. The results of both of these statements return the same information with the same amount of rows, 6120 rows (found in the **Fatma Khalil STEP3 Log File (Death).log file**).

**c- In and Exists Operators:**

```
SELECT
    *
FROM
    deaths D
WHERE
    cause IN (SELECT
            cause
        FROM
            deathbyalcoholdisorders A
```

```
        WHERE
            D.cause = A.cause);


SELECT
    *
FROM
    deaths D
WHERE
    EXISTS( SELECT
            *
        FROM
            deathbyalcoholdisorders A
        WHERE
            D.cause = A.cause);
```

The IN and EXISTS Operator select statements do the exact job of the INTERSECT and INNER JOIN operator select statement above, however choosing the alcohol misuse deaths instead of drug misuse deaths. The IN and EXISTS Operator statements select the cause of death by alcohol misuse and returns the deaths caused by alcohol misuse for each country each year from 1990 to 2019. Again, both of these select statements returned the same information and the same amount of rows (6120 rows, (found in the **Fatma Khalil STEP3 Log File (Death).log file**)).

**d-Aggregate Operators:**

```
CREATE VIEW DeathBySubstances AS
    SELECT
        D.cause,
        SUM(S.totalDeaths + A.totalDeaths) AS DeathBySubstances
```

```
FROM
    deaths D,
    DeathByAlcoholDisorders A,
    DeathByDrugs S
        INNER JOIN
    DeathByAlcoholDisorders Alcohol ON Alcohol.countryCode = S.countryCode
GROUP BY D.cause
HAVING deathbysubstances > 0;
```

I have made the select statement into a view since running the select statement takes over 1 hour to process on my computing device. In order to see the results of the select statement, run the SELECT * FROM deathbysubstances statement. I have used the SUM aggregate function in order to see how many deaths are caused by substance misuse (so alcohol misuse and drug misuse together instead of separately). This would help us identify the percentage of deaths caused by substance misuse out of all deaths in a country.

**2-Constraints and Triggers:**

```
CREATE TABLE TotalCountryDeaths AS
    SELECT
        countryCode, SUM(numberOfDeaths) AS totalDeaths
    FROM
        deaths
    GROUP BY countryCode;

select min(totalDeaths) from totalcountrydeaths;
select max(totalDeaths) from totalcountrydeaths;
```

```
ALter table totalcountrydeaths ADD constraint totalDeaths check( totalDeaths >= 299 AND totalDeaths
<= 252783134);
insert into totalcountrydeaths(countryCode, totalDeaths) values('CCK', 252783135);

delimiter \
create trigger beforeUpdate before update on totalcountrydeaths
for each row
begin
 if NEW.totalDeaths < 299 then set New.totalDeaths = 299;
 elseif NEW.totalDeaths > 252783134 then set New.totalDeaths = 252783134; end if; end \
 delimiter ;

 delimiter \
create trigger beforeInsert before insert on totalcountrydeaths
for each row
begin
 if NEW.totalDeaths < 299 then set New.totalDeaths = 299;
 elseif NEW.totalDeaths > 252783134 then set New.totalDeaths = 252783134; end if; end \
 delimiter ;
insert into totalcountrydeaths(countryCode, totalDeaths) values('CCK', 252783135);
```

First, I created a table to see the total numbers of deaths in a country from the year 1990 to 2019, so instead of showing the amount of deaths per year it shows all the deaths from 1990 to 2019. Then to identify the MIN and MAX values in the table, I used the MIN and MAX operators in select statements then used the values returned in order to properly alter the table and add the constraint preventing an insertion smaller than the MIN value and bigger than the MAX value. After trying to insert a value that

violates these constraints, I received an error message and the insertion failed. Then I created the required triggers (before update and before insertion), after that I also tried to insert a tuple that had a value greater than the value found in the select statement. Because of the trigger, the insertion was successful but the value of the tuple was changed into the MAX value found by the select statement instead of the value that was in the tuple originally. I found that in order to properly protect data and to insure its integrity, it was better to make a constraint that rejects the insertion than making a trigger that modifies the value of the tuple to be inserted. The error and successful insertion  messages are found in the **Fatma Khalil STEP3 Log File (Death).log file**.

**3-Stored Procedure:**

```
delimiter \
CREATE PROCEDURE showCountryDeaths(in ISOCode char(5), Out countryDeaths int)
BEGIN
   SELECT totalDeaths INTO countryDeaths FROM totalcountrydeaths WHERE countryCode =
ISOCode;
END \
delimiter ;

call showCountryDeaths('AFG', @countryDeaths);
select @countryDeaths;

call showCountryDeaths('USA', @countryDeaths);
select @countryDeaths;
```

The purpose of this stored procedure is to see the total number of deaths in a country (all deaths from 1990 to 2019 regardless of cause). Then to test the stored procedure I entered the countryCode ‘AFG’ to see all the total number of deaths in Afghanistan (it is 5270825, also found in the **Fatma Khalil STEP3**

**Log File (Death).log file**). I also used the same procedure but entered the countryCode 'USA' in order to see the total numbers of deaths in the United States of America (it is 68979351, also found in the **Fatma Khalil STEP3 Log File (Death).log file**).

## Substance Abuse:

**Filzah Azeem SQL STEP3(substance).sql**

**a- Views:**

```
CREATE VIEW countries_with_alcohol_higher_than_2 AS
SELECT countryCode, Year, prevalenceAlcoholUseDisorder_Percent
FROM substanceabusepercountry
WHERE prevalenceAlcoholUseDisorder_Percent >= 2;
CREATE VIEW countries_with_drug_higher_than_alcohol AS
SELECT countryCode, Year, prevalenceAlcoholUseDisorder_Percent,
prevalenceDrugUseDisorder_Percent
FROM substanceabusepercountry
WHERE prevalenceDrugUseDisorder_Percent >= prevalenceAlcoholUseDisorder_Percent;
```

In the first view I created, it returns all the fields that have a value of greater than or equal to 2 in the prevalenceAlcoholUseDisorder-Percent column, and displays the results with countryCode, Year, and prevalenceAlcoholUseDisorder_Percent.

In the second view, It returns the countryCodes, Year, prevalenceAlcoholUseDisorder_Percent, and, prevalenceDrugUseDisorder_Percent, but only the rows where the drug use percentage is higher than the alcohol use percentage.

For each of these views I was interested in finding out which countries had a higher percentage of alcohol disorders and for the 2nd view, as most countries have a higher alcohol Use, I wanted to find out which countries had more drug use orders than alcohol use disorders.

**d- Aggregate Operator:**

```
SELECT DISTINCT D.countryCode, COUNT(D.Year)
FROM countries_with_alcohol_higher_than_2 A, countries_with_drug_higher_than_alcohol D
WHERE D.countryCode = A.countryCode
GROUP BY D. countryCode, D.Year
HAVING ( SELECT COUNT(D.Year)
        FROM countries_with_alcohol_higher_than_2 A,
countries_with_drug_higher_than_alcohol D
        WHERE A.prevalenceAlcoholUseDisorder_Percent <=
D.prevalenceAlcoholUseDisorder_Percent)
```

I used the aggregate COUNT() to find the number of Years for each countryCode where the percentage of alcohol use disorder was greater than or equal to 2, for countries where drug use disorder percentage was greater than the alcohol use percentage. Over here I wanted to know which countries had a high alcohol use along with greater drug use disorder.

**Filzah Azeem STEP3(substance) LogFile.log**

In this file you can see the successful creation of the views as well as the number of rows returned by each view.

# Air Pollution:

### Saleh Alshurafa SQL STEP3 (Air Pollution).sql

**a- Views:**

```
#View (1)
CREATE VIEW population_exposed AS
SELECT countryCode, Year, `PM2.5, pop. exposed to levels > WHO guideline value (% TOT)`
```

```
FROM `cs306-project`.airpollution
WHERE `PM2.5, pop. exposed to levels > WHO guideline value (% TOT)` >= 50;
#View (2)
CREATE VIEW pollution_production AS
SELECT countryCode, Year, `Total including LUCF in tonnes`
FROM `cs306-project`.airpollution
WHERE `Total including LUCF in tonnes` >= 50000000;
```

I created 2 views from the Air Pollution table made in STEP 1. One view “population_exposed” displays the codes for countries whose percentage of population exposed to air quality that is lower than WHO standards is 50 or more along with the corresponding year for the data. The other view “pollution_produced” displays the codes for countries whose pollution production exceeds 50000000 tonnes along with the corresponding year for the data. I chose to create these 2 views so that we can see how much the total pollution produced by a country affects the exposure of the population to air pollution. It may seem trivial that the more pollution produced the more the people exposed to pollution, but I wanted to see to what extent it affects the exposure of the population.

**b- Joins and Set Operators**

```
#Using Set Operator NOT IN
SELECT E.countryCode, E.`PM2.5, pop. exposed to levels > WHO guideline value (% TOT)`, E.Year
FROM population_exposed E
WHERE E.countryCode NOT IN (
        SELECT   E.countryCode
        FROM population_exposed E, pollution_production P
        WHERE E.countryCode = P.countryCode);
#Using LEFT OUTER JOIN
SELECT E.countryCode, E.`PM2.5, pop. exposed to levels > WHO guideline value (% TOT)`, E.Year
FROM population_exposed E LEFT OUTER JOIN pollution_production P
```

```
ON E.countryCode = P.countryCode
WHERE P.countryCode IS NULL;
```

The first code uses NOT IN to create the set difference effect that EXCEPT does. EXCEPT is not supported by MySQL hence why I used NOT IN instead. The purpose of this code is to check which countries out of those who have at least 50% of their population exposed to pollution at a level that exceeds WHO air quality standards does not produce more than 50000000 tonnes of air pollution. Since some countries might have exceeded 50000000 tonnes of air pollution in certain years, I also show the year data so that we can know in which years they exceed 50% of population exposed while producing under 50000000 tonnes of air production. This shows that despite some countries producing less pollution than others, there has been at least a year where they still had at least half of their population living under terrible air quality. The second code does the same exact thing as the first one, but it replaces the NOT IN operator with LEFT OUTER JOIN. Both of them produce the same results with 1013 rows each as indicated by **Saleh Alshurafa Logs STEP3 (Air Pollution).log** file.

**C- In and Exists**

```
# Using IN
SELECT E.countryCode, E.`PM2.5, pop. exposed to levels > WHO guideline value (% TOT)`, E.Year
FROM population_exposed E
WHERE E.countryCode IN (
                        SELECT P.countryCode
                    FROM pollution_production P
                    WHERE E.countryCode = P.countryCode);

# Using EXISTS
SELECT E.countryCode,  E.`PM2.5, pop. exposed to levels > WHO guideline value (% TOT)`, E.Year
```

```
FROM population_exposed E
WHERE EXISTS (
                    SELECT P.countryCode
                    FROM pollution_production P
                    WHERE E.countryCode = P.countryCode);
```

Both codes do the opposite task of the codes of the NOT IN and LEFT OUTER JOIN discussed previously. They display the countries that fit both having at least 50% of their population exposed to air with quality less than WHO standards and producing at least 50000000 tonnes of air pollution for the corresponding years. The purpose of this is to check which countries appear as a result of this query that do not appear in the previous where I used NOT IN and LEFT OUTER JOIN operators. This also shows that while there are countries with less air pollution having at least 50% of their population affected, there are also other countries who produce more and also have at least 50% of the population exposed. The results raise questions about the true effect of air pollution production on the amount of population affected since in both cases you can find multiple countries where the population is affected. However, it could also mean that we will need to look at other factors to understand how air pollution production affects the population. Both codes produce the same results displaying 1082 rows as indicated by **Saleh Alshurafa Logs STEP3 (Air Pollution).log** file. The difference between the codes is that one uses IN while the other uses EXISTS.

**d- Aggregate Operators**

```
SELECT P.countryCode,
        AVG(P.`Total including LUCF in tonnes`) AS `Average Pollution Production Over 1990-2017`,
        AVG(E.`PM2.5, pop. exposed to levels > WHO guideline value (% TOT)`) AS `Average
Percentage of Population In Danger over 1990-2017`
FROM pollution_production P, population_exposed E
```

```
WHERE P.countryCode = E.countryCode
GROUP BY P.countryCode
HAVING 90 < (SELECT AVG(E.`PM2.5, pop. exposed to levels > WHO guideline value (% TOT)`)
                FROM population_exposed E
                WHERE E.countryCode = P.countryCode);
```

Finally, I used the AVG aggregate operator to check the average total pollution produced and average percentage of population living in air quality that is below WHO standards for each country from 1990 till 2017. However, I wanted to only see which countries averaged over 90% of their population living in such conditions. I selected 90 because it shows that across the given time period these countries have not improved a lot or even at all when it comes to the fight against air pollution. I wanted to see how much air pollution they have produced on average over these years so that I can try to approximately identify a specific number that countries should try to go under in order to decrease the amount of the population living under terrible air quality. We got 85 rows from this query as indicated by **Saleh Alshurafa Logs STEP3 (Air Pollution).log** file which indicates that 85 countries average over 90% of their population living in terrible air conditions along with the average air pollution production for these countries.