GROUP 1 CS306 STEP 3

This is a submission for Group 1 members Göktuğ Aygün, Kerim Demir, Berk Bilgiç, Steven El Khaldi (30048), Kemal Yılmaz (31097).

This part of the submission belongs to Göktuğ Aygün 30608.

In this step, I used MySQL workbench utilities to discover insights. In order to do this, I used views, various operators and other functionalities like constraints, triggers or procedures. To be more precise, in file “Emission Above Average.sql”:

View “Average_Emission”

Starting from line 4, I created a view to have the average amount of emission of 4 different gases by all countries. This will be utilized in “Air Pollution Deaths Above Average.sql”. I renamed and updated my select statements.

View “Worldwide_Emissions”

Starting from line 38, I created another view this time to have the average emission of these gases not specific to a country but worldwide.

View “countries_with_high_emission”

Starting from line 52, I used sub-queries and the union operator to have the list of countries that have higher emissions than the average to create the view “”. Sub-queries have 21, 24, 17 and 17 countries respectively which sum up to 89. However, when we check the total number of countries in the outer query, we see that the number is 32. This means that there are many countries that overlap and exist in many tables.

View “total_nature_casualties”

The last view of this file called starts in line 74. This view is used to check the total number of deaths due to nature related reasons all around the globe.

In the file “Air Pollution Deaths Above Average.sql”, I used similar techniques and made some changes to achieve different results. To start with:

View “air_pollution_deaths_of_countries”

I created this view starting in line 6 to have the number of deaths caused only by air pollution for each country. This view will be used for the creation of the next one.

View “air_pollution_deaths_worldwide”

This view is created starting in line 18 to generalize the results achieved from the previous view.

View “countries_with_high_deaths”

By creating sub-queries and using the union operator as done before, I created this view starting in line 29. The sub-queries have 23, 21, 25 countries but the resulting main query has 29 countries which again proves the existence of overlapping sets.

In the file “Insights.sql”, the main focus is on the use of operators such as except or Left ‘Outer’ Join. In line 4, we can see that the intersection of countries with higher than average emissions and higher than average deaths related to air pollution have 10 members in common. We can use the view created between the lines 8-14 to see the number of countries that are in “countries_with_high_deaths” but not in “countries_with_high_emission”. The number of such countries is 19 which sums up to 29 with the 10 countries from the intersection part which validates are calculations. Between the lines 19-22, I achieved the same result by using Left Join instead of Except operator by utilizing the where statement as well.

Views “high_air_pollution_deaths” and “high_household_air_pollution_deaths”

Between lines 26-34, I again created sub-queries to be used to check the intersection and in operators. In line 52,we can see that the intersection for these views has 15 countries in total. This is the same number we get from the next query which utilizes the use of an inner and an outer query. Finally, the same result is achieved by using the ‘Exists’ operator as well.

In the file “Q2.sql”, I start with a query to check the number of countries from each continent that have countries with high emission rates -by high, I mean above the average throughout this documentation-. We can observe that the continent with the highest number of such countries is Asia with 15. Then, the use of variables and ‘Alter Table’ commands can be seen. These are to insert new constraints or drop existing constraints. In line 20, a constraint is

created using the hard coded numbers because aggregate operators cannot be used in constraints -I also asked for the opinion of our TA Hasan Ertuğrul at this step-.

Then a procedure called “get_death_numbers” is created to learn about the death numbers of each country. This procedure focuses on nature related deaths and brings results about Deaths by Unsafe Water Source or Unsafe Sanitation with the given country code, between the years 1990-2019. These fields are the remaining ones that have not been investigated before.

At the end, we can see two triggers being created. One to used before insertion and the other for before updating. These triggers don’t let the user or admin to insert or change the values to a number outside of the interval “0-2450944” in ‘Deaths by Unsafe Water Source’ column. These are not randomly selected numbers but actually are the minimum and maximum values. Note that type casting is applied to solve the problem of treating numbers as signed/unsigned numbers which causes unexpected behavior.

The functionality of these triggers are tested between the lines 83-121 and the results can be verified by using the procedure “call_death_numbers” with “TUR” iso_code as a parameter.

Thank you for your time

This part of the submission belongs to **Kerim Demir 28853**.

→ UPDATE substance_deaths SET total_deaths = (deaths_by_tobacco + deaths_by_drug_use + deaths_by_alcohol_use + deaths_by_smoking + deaths_by_secondhand_smoke);

- Before starting to write my commands, I added a new column to the substance_deaths table to see the total death count each year from substance usage.

→ Create view smoking_deaths_gt_avg as SELECT C.name, S.year, S.deaths_by_smoking, AVG(S2.deaths_by_smoking) AS avg_smoking_deaths_per_year, S.total_deaths

FROM substance_deaths S

```
JOIN countries C ON S.iso_code = C.iso_code

JOIN substance_deaths S2 ON S.iso_code = S2.iso_code

GROUP BY S.iso_code, S.year

HAVING S.deaths_by_smoking > AVG(S2.deaths_by_smoking);
```

- Created a view named smoking_deaths_gt_avg. In this view, the countries that have a higher death rate from smoking than the average are listed (year by year).

```
Create view cardiovascular_diseases_gt_avg as SELECT C.name, H.year,
H.deaths_by_cardiovascular_diseases, AVG(H2.deaths_by_cardiovascular_diseases) AS
avg_cardiovascular_deaths_per_year

FROM health_deaths H

JOIN countries C ON H.iso_code = C.iso_code

JOIN health_deaths H2 ON H.iso_code = H2.iso_code

GROUP BY H.iso_code, H.year

HAVING H.deaths_by_cardiovascular_diseases > AVG(H2.deaths_by_cardiovascular_diseases);
```

- Created a view named cardiovascular_diseases_gt_avg. In this view, the countries that have a higher disease rate from cardiovascular diseases than the average are listed (year by year).

```
→ SELECT name, year

FROM smoking_deaths_gt_avg

WHERE year and name IN (

  SELECT year and name

  FROM cardiovascular_diseases_gt_avg);
```

- My **IN** statement: Getting the name and year data from the database where name and year are in both tables which have a higher rate than the average.

→ SELECT name, year

FROM smoking_deaths_gt_avg S

WHERE EXISTS (

SELECT *

FROM cardiovascular_diseases_gt_avg C

WHERE C.year = S.year and C.name = S.name);

- Getting the same data with using exists instead of in.

→ select S.name, S.year, S.deaths_by_smoking, C.deaths_by_cardiovascular_diseases, S.total_deaths as substance_total_deaths from smoking_deaths_gt_avg S INNER JOIN cardiovascular_diseases_gt_avg C on C.year = S.year and C.name = S.name;

- gives the list of countries with their data listed in the query where the deaths from smoking and number of cardiovascular diseases are above their average.

→ select S.name, S.year from smoking_deaths_gt_avg S intersect select C.name, C.year from cardiovascular_diseases_gt_avg C;

- Getting the same data with the set operator intersect.

→ select C.name, S.* from substance_deaths S, countries C where C.iso_code = S.iso_code and deaths_by_smoking = (select max(deaths_by_smoking) from substance_deaths);

- Getting the row with the max smoking deaths using MAX().

→ select C.name, S.* from substance_deaths S, countries C where C.iso_code = S.iso_code and deaths_by_smoking = (select min(deaths_by_smoking) from substance_deaths);

- Getting the row with the min smoking deaths using MIN().

→ set @max_smoking_deaths = (select max(deaths_by_smoking) from substance_deaths);

→ set @min_smoking_deaths = (select min(deaths_by_smoking) from substance_deaths);

- Stored min and max values in variables to use later.

→ ALTER TABLE substance_deaths ADD CONSTRAINT in_range_smoking_deaths

check (deaths_by_smoking >= 1 and deaths_by_smoking <= 7693368);

- Adding a general constraint for smoking to make sure that new data with invalid inputs cannot be entered to the table.

→ INSERT INTO substance_deaths (iso_code, year, deaths_by_tobacco, deaths_by_drug_use, deaths_by_alcohol_use, deaths_by_smoking, deaths_by_secondhand_smoke, total_deaths) VALUES ("AFG", 2020, 16000, 750, 5, 0, 6100, 33.849);

- Trying to insert a row that doesn't meet the constraint format.

→

```
delimiter //
 CREATE TRIGGER ins_check
BEFORE INSERT ON substance_deaths
for each row
Begin
        IF NEW.deaths_by_smoking < @min_smoking_deaths then
                set NEW.deaths_by_smoking = @min_smoking_deaths;
        ELSEIF NEW.deaths_by_smoking  > @max_smoking_deaths then
                set NEW.deaths_by_smoking  = @max_smoking_deaths;
        END IF;
END; //
delimiter ;
```

- Created an insert trigger so that no invalid inputs will be entered to the table.

→

→

```
delimiter //
CREATE TRIGGER upd_check BEFORE UPDATE ON substance_deaths
FOR EACH ROW
BEGIN
    IF NEW.deaths_by_smoking < @min_smoking_deaths then
        set NEW.deaths_by_smoking = @min_smoking_deaths;
    ELSEIF NEW.deaths_by_smoking > @max_smoking_deaths then
        set NEW.deaths_by_smoking = @max_smoking_deaths;
    END IF;
END;//
delimiter ;
```

- Created an update trigger so that new values can't be out of range.

→

```
delimiter //
```

→ CREATE PROCEDURE substance_total_death(code VARCHAR(5))

```
BEGIN
   (select iso_code, sum(total_deaths) as total_deaths_by_substance_use from substance_deaths
where iso_code = code group by iso_code);
END //
delimiter ;
```

- Procedure that returns the total death count from substance usage of the given country.

Samples:

→ call substance_total_death('AFG');

→ call substance_total_death('ITA');

→ call substance_total_death('USA');

Creating a trigger VS Creating a general constraint:

→ In the insert trigger that we created we are able to insert into the table if it's in the given range, however if the input value is outside of the given range our trigger is activated and sets the value to max if it's bigger than max and sets it to min if its smaller than min. Also we can make our trigger more complex than our constraint because in constraint we are only putting a restriction to our input values. On the other hand, we can write different queries for different cases in our triggers.

This part of the submission belongs to **Steven El Khaldi 30048**.

Files referenced can be found under my folder in the step 3 branch and folder in our github repo.

1. **Discover Insights:**
    a. **Create Views: (*view.sql* + *view_time_trend.sql)***

```
# create view to show countries with above average deaths due to nutritional deficiencies
CREATE VIEW above_average_diet_deaths_by_nutritional_deficiencies AS
SELECT count(*) as num_years, iso_code from diet_deaths
WHERE deaths_by_nutritional_deficiencies > (select avg(deaths_by_nutritional_deficiencies) from diet_deaths)
GROUP BY iso_code;
```

This CREATE VIEW statement was made in order to create a view to **show all countries (using their iso codes) that recorded a number of deaths by nutritional deficiencies in at least one year that was greater than the overall average of deaths by nutritional deficiencies over all countries in all years, and in how many years it recorded such an above average number of deaths.** Its purpose is to be used later during data analysis to find intersecting countries that record such numbers and record numbers in other categories (using other views), to find the set difference between these above average countries in this category and countries that record a specific criterion in other categories, and so on.

```
# create view to show countries with above average deaths due to diet low in fruits
CREATE VIEW above_average_diet_deaths_by_diet_low_in_fruits AS
SELECT count(*) AS num_years, iso_code from diet_deaths
WHERE deaths_by_diet_low_in_fruits > (select avg(deaths_by_diet_low_in_fruits)
from diet_deaths)
GROUP BY iso_code;
```

This CREATE VIEW statement was made in order to create a view to **show all countries (using their iso codes) that recorded a number of deaths by diet low in fruits in at least one year that was greater than the overall average of deaths by diet low in fruits over all countries in all years, and in how many years it recorded such an above average number of deaths**. Its purpose is to be used later during data analysis to find intersecting countries that record such numbers and record numbers in other categories (using other views), to find the set difference between these above average countries in this category and countries that record a specific criterion in other categories, and so on.

```
# create view to show countries with above average deaths due to diet low in whole
grains
CREATE VIEW above_average_diet_deaths_by_diet_low_in_whole_grains AS
SELECT count(*) AS num_years, iso_code from diet_deaths
WHERE deaths_by_diet_low_in_whole_grains > (select
avg(deaths_by_diet_low_in_whole_grains) from diet_deaths)
GROUP BY iso_code;
```

This CREATE VIEW statement was made in order to create a view to **show all countries (using their iso codes) that recorded a number of deaths by diet low in whole grains in at least one year that was greater than the overall average of deaths by diet low in**

whole grains over all countries in all years, and in how many years it recorded such an above average number of deaths. Its purpose is to be used later during data analysis to find intersecting countries that record such numbers and record numbers in other categories (using other views), to find the set difference between these above average countries in this category and countries that record a specific criterion in other categories, and so on.

```
# create view to show development of number of deaths by nutritional deficiencies in
the world over time
CREATE VIEW nutritional_deficiencies_diet_deaths_period AS
SELECT SUM(deaths_by_nutritional_deficiencies) AS
total_deaths_by_nutritional_deficiencies,
AVG(deaths_by_nutritional_deficiencies) AS
avg_deaths_by_nutritional_deficiencies_per_country,
COUNT(*) AS num_countries,
year
FROM diet_deaths
GROUP BY year;
```

This CREATE VIEW statement was made in order to create a view to **show the development and trend in the total and average number of deaths by nutritional deficiencies over time year by year from 1990 to 2019** (as well as the number of recorded countries/regions in each year). As can be seen from the view, the total and average number of deaths by nutritional deficiencies around the world recorded a general decrease over the years. This will also later on be used for data visualization.

b. Joins and Set Operators: (*except_outerjoin.sql)*

```
# except (returns 13 rows)
SELECT A.iso_code
FROM above_average_diet_deaths_by_nutritional_deficiencies A
EXCEPT
SELECT B.iso_code
FROM above_average_diet_deaths_by_diet_low_in_whole_grains B;

# outer join (also returns 13 rows)
SELECT A.iso_code
```

```
FROM above_average_diet_deaths_by_nutritional_deficiencies A
LEFT OUTER JOIN above_average_diet_deaths_by_diet_low_in_whole_grains B
ON A.iso_code = B.iso_code
WHERE B.iso_code IS NULL;
```

Both SELECT statements were made to find the iso codes of those countries that ever recorded an above average number of deaths by nutritional deficiencies but never recorded an above average number of deaths by diet low in whole grains, and both returned the same 13 iso codes. This can later be used for data analysis and conclusions about these countries.

c. "In" and "Exists": (*in_exists.sql)*

```
#IN (returns 21 rows)
SELECT A.iso_code
FROM above_average_diet_deaths_by_nutritional_deficiencies A
WHERE A.iso_code in (
SELECT B.iso_code
FROM above_average_diet_deaths_by_diet_low_in_fruits B
);

# EXISTS (also returns 21 rows)
SELECT A.iso_code
FROM above_average_diet_deaths_by_nutritional_deficiencies A
WHERE EXISTS (
SELECT B.iso_code
FROM above_average_diet_deaths_by_diet_low_in_fruits B
WHERE A.iso_code = B.iso_code
);
```

Both SELECT statements were made to find the iso codes of those countries that ever recorded an above average number of deaths by nutritional deficiencies and also ever recorded an above average number of deaths by diet low in fruits, and both returned the same 21 iso codes. This can later be used for data analysis and conclusions about these countries/regions.

d. Aggregate Operators: (*aggregate_operators.sql)*

```
# avg (returns 25 rows)
SELECT C.name as country, D.iso_code as iso_code,
AVG(D.deaths_by_diet_low_in_fruits) as
yearly_avg_deaths_by_diet_low_in_fruits
FROM diet_deaths D
JOIN countries C ON D.iso_code = C.iso_code
GROUP BY C.name, D.iso_code
HAVING AVG(D.deaths_by_diet_low_in_fruits) > (SELECT
AVG(deaths_by_diet_low_in_fruits) FROM diet_deaths);
```

Returns the countries (their names and iso codes) whose yearly deaths by diet low in fruits average is greater than the overall average over all countries and years, and the corresponding average. Can be used for data analysis in later steps.

```
# min (returns 283 rows)
SELECT C.name as country, D.iso_code as iso_code,
MIN(D.deaths_by_nutritional_deficiencies) as
min_deaths_by_nutritional_deficiencies, D.year as year
FROM diet_deaths D
JOIN countries C ON D.iso_code = C.iso_code
GROUP BY C.name, D.iso_code, D.year
HAVING MIN(D.deaths_by_nutritional_deficiencies) = (SELECT
MIN(deaths_by_nutritional_deficiencies) FROM diet_deaths);
```

Returns every instance a country or region recorded the overall minimum deaths by nutritional deficiencies over all years and countries (283 instances).

```
# max (returns 1 row)
SELECT C.name as country, D.iso_code as iso_code,
MAX(D.deaths_by_nutritional_deficiencies) as
max_deaths_by_nutritional_deficiencies, D.year as year
FROM diet_deaths D
JOIN countries C ON D.iso_code = C.iso_code
GROUP BY C.name, D.iso_code, D.year
HAVING MAX(D.deaths_by_nutritional_deficiencies) = (SELECT
MAX(deaths_by_nutritional_deficiencies) FROM diet_deaths);
```

Returns every instance a country or region recorded the overall maximum deaths by nutritional deficiencies over all years and countries (1 instance).

```
# count (returns 211 rows)
SELECT count(*) AS times_below_avg_deaths_by_iron_deficiency, C.name as
```

```
country, D.iso_code as iso_code
FROM diet_deaths D
JOIN countries C on D.iso_code = C.iso_code
WHERE D.deaths_by_iron_deficiency < (SELECT
AVG(deaths_by_iron_deficiency) from diet_deaths)
GROUP BY C.name, D.iso_code;
```

Returns how many times each country recorded a number of deaths by iron deficiency less than the overall average of deaths by iron deficiency over all countries and years.

```
# sum (returns 30 rows)
SELECT D.year as year, SUM(D.deaths_by_nutritional_deficiencies) AS
total_deaths_by_nutritional_deficiencies,
      (SELECT C.name FROM countries C
       JOIN diet_deaths D1 ON C.iso_code = D1.iso_code
       WHERE D1.year = D.year AND D1.iso_code != "WWW"
       ORDER BY D1.deaths_by_nutritional_deficiencies DESC
       LIMIT 1) AS max_deaths_country
FROM diet_deaths D
JOIN countries C ON D.iso_code = C.iso_code
GROUP BY D.year;
```

Returns the total number of deaths over all countries each year by nutritional deficiencies, and the name of the country or region that recorded the maximum number of deaths by nutritional deficiencies that year (not taking into account the world with iso code WWW).

2. Constraints and Triggers: (*constraint.sql)*

```
# min value of deaths by nutritional deficiencies
SELECT min(deaths_by_nutritional_deficiencies)
FROM diet_deaths;
# returns 0

# max value of deaths by nutritional deficiencies
SELECT max(deaths_by_nutritional_deficiencies)
FROM diet_deaths;
# returns 757152

# add general constraint to diet_deaths
ALTER TABLE diet_deaths ADD CONSTRAINT
```

```
deaths_by_nutritional_deficiencies_range CHECK (deaths_by_nutritional_deficiencies
>= 0 AND deaths_by_nutritional_deficiencies <= 757152);

# test constraint
INSERT INTO diet_deaths (iso_code, deaths_by_nutritional_deficiencies, year) VALUES
("LEB", 757153, 1990);
# violates the range constraint on deaths_by_nutritional_deficiencies

DELIMITER $$
# before insert trigger
CREATE TRIGGER fix_deaths_by_nutritional_deficiencies_before_insert
BEFORE INSERT
ON diet_deaths FOR EACH ROW
BEGIN
  IF NEW.deaths_by_nutritional_deficiencies < 0 THEN
    SET NEW.deaths_by_nutritional_deficiencies = 0;
  ELSEIF NEW.deaths_by_nutritional_deficiencies > 757152 THEN
    SET NEW.deaths_by_nutritional_deficiencies = 757152;
  END IF;
END $$

# before update trigger
CREATE TRIGGER fix_deaths_by_nutritional_deficiencies_before_update
BEFORE UPDATE
ON diet_deaths FOR EACH ROW
BEGIN
  IF NEW.deaths_by_nutritional_deficiencies < 0 THEN
    SET NEW.deaths_by_nutritional_deficiencies = 0;
  ELSEIF NEW.deaths_by_nutritional_deficiencies > 757152 THEN
    SET NEW.deaths_by_nutritional_deficiencies = 757152;
  END IF;
END $$
DELIMITER ;
```

The general constraint added (**deaths_by_nutritional_deficiencies_range**) to the diet_deaths table makes sure any new row added has a **deaths_by_nutritional_deficiencies** value between 0, the min value, and 757152, the max value.

The two triggers created on the diet_deaths table are called either before an item is inserted or updated in the table to make sure the deaths_by_nutritional_deficiencies value is within the [min, max] range of [0, 757152]. If the value is less than 0, it is set to 0, and if it is more than 757152, it is set to 757152.

3. Stored Procedure:

```
# procedure
DELIMITER $$
CREATE PROCEDURE get_deaths_by_nutritional_deficiencies (IN iso_code
VARCHAR(5))
BEGIN
  DECLARE count_iso_code INT;
  SELECT COUNT(*) INTO count_iso_code FROM diet_deaths A WHERE A.iso_code
= iso_code;
  IF count_iso_code > 0 THEN
    SELECT AVG(deaths_by_nutritional_deficiencies) AS avg_deaths,
    MAX(deaths_by_nutritional_deficiencies) AS max_deaths
    FROM diet_deaths A WHERE A.iso_code = iso_code;
  ELSE
    SELECT 'Such an ISO code does not exist in the table' AS error_message;
  END IF;
END$$
DELIMITER ;

# two different calls with two different inputs
CALL get_deaths_by_nutritional_deficiencies('AGO'); # avg_deaths: 7839.3667,
max_deaths: 12232
CALL get_deaths_by_nutritional_deficiencies('AWH'); # avg_deaths: 164695.3333,
max_deaths: 217480
```

The procedure created above returns the average number of deaths by nutritional deficiencies and the maximum number of deaths by nutritional deficiencies recorded by a country given its iso code. If the iso code isn't valid, an error message is returned instead. Two example calls to this procedure with the corresponding different outputs are mentioned above in the code with comments.

This part of submission belongs to **Berk Bilgiç 29157**

→ ALTER TABLE nature_deaths ADD COLUMN total_nature_deaths INT;

SET SQL_SAFE_UPDATES = 0;

UPDATE nature_deaths SET total_nature_deaths = (deaths_by_outdoor_air_pollution+

deaths_by_unsafe_water_source+deaths_by_household_air_pollution_from_solid_fuels+

deaths_by_air_pollution+deaths_by_unsafe_sanitation);

**Before writing our commands we update our table with the total death caused by all natural causes to have a more comprehensive understanding of the table. SET SQL_SAFE_UPDATES= 0 is required for us to get the access to use UPDATE.

→ CREATE VIEW average_deaths_by_year AS

SELECT year,

AVG(deaths_by_outdoor_air_pollution) AS avg_outdoor_pollution,

AVG(deaths_by_unsafe_water_source) AS avg_unsafe_water,

AVG(deaths_by_household_air_pollution_from_solid_fuels) AS avg_household_air_pollution,

AVG(deaths_by_air_pollution) AS avg_air_pollution,

AVG(deaths_by_unsafe_sanitation) AS avg_unsafe_sanitation

FROM nature_deaths

GROUP BY year;

SELECT * FROM average_deaths_by_year;

** In this query we basically get the average deaths by specific natural causes in years between 1990 and 2019 and then we view it with the “SELECT * FROM average_deaths_by_year” statement.

→ CREATE VIEW average_comparision AS

SELECT nature_deaths.iso_code, nature_deaths.year

FROM nature_deaths

INNER JOIN average_deaths_by_year ON nature_deaths.year = average_deaths_by_year.year

WHERE nature_deaths.deaths_by_outdoor_air_pollution >
average_deaths_by_year.avg_outdoor_pollution

AND nature_deaths.deaths_by_unsafe_water_source >
average_deaths_by_year.avg_unsafe_water;

SELECT * FROM average_comparision;

**In this query we use INNER JOIN and combine our average_deaths_by_year view with our original nature_deaths table in rows where nature_deaths years are equal to average_deaths_by_year views years and after that we choose the countries and the years when their deaths related to unsafe water source and outdoor air pollution is above the average. We observe that some countries were always above the average in the years 1990-2019.

→ SELECT countries.name

FROM countries

WHERE countries.iso_code IN (SELECT iso_code FROM average_comparision);

**With the help of the countries table and query we get the names of the countries that are above average in the previous average_comparision view.

→ SELECT countries.name

FROM countries

WHERE EXISTS (SELECT iso_code FROM average_comparision WHERE average_comparision.iso_code=countries.iso_code);

**This is the same query with the use of EXISTS instead of IN

→SELECT * FROM nature_deaths N WHERE N.deaths_by_unsafe_water_source=(SELECT MAX(deaths_by_unsafe_water_source) FROM nature_deaths);

**In this statement we get the row that holds the information when most death occured due to unsafe_water_source from nature_deaths table

→SET @max_deaths_by_unsafe_water_source= (SELECT MAX(deaths_by_unsafe_water_source) FROM nature_deaths);

SET @min_deaths_by_unsafe_water_source= (SELECT MIN(deaths_by_unsafe_water_source) FROM nature_deaths);

**With these statements we set max and min variables related to unsafe water source deaths.

→ALTER TABLE nature_deaths ADD CONSTRAINT range_death

CHECK (0 <= deaths_by_unsafe_water_source AND 2450944 >= deaths_by_unsafe_water_source);

INSERT INTO nature_deaths

VALUES('AFG', 2020 , 12000, 2450945, 34000, 37000, 3000, 80000);

DELETE FROM nature_deaths WHERE iso_code='AFG' AND year=2020;

**We add a constraint according to min and max values in unsafe water source deaths and in this example after we initiate our constraint we try to add values which consists of value bigger than the max and as a result we get an error. You can try this multiple times by deleting the values inserted and insert it again.

→DELIMITER //

CREATE TRIGGER range_checker

BEFORE INSERT ON nature_deaths

FOR EACH ROW

BEGIN

IF NEW.deaths_by_unsafe_water_source < @min_deaths_by_unsafe_water_source THEN

SET NEW.deaths_by_unsafe_water_source = @min_deaths_by_unsafe_water_source;

ELSEIF NEW.deaths_by_unsafe_water_source > @max_deaths_by_unsafe_water_source THEN

SET NEW.deaths_by_unsafe_water_source = @max_deaths_by_unsafe_water_source;

END IF;

END//

DELIMITER ;

**With the help of this insert trigger we can’t get values out of range into our table.

→DELIMITER //

```
CREATE TRIGGER range_checker_update

BEFORE UPDATE ON nature_deaths

FOR EACH ROW

BEGIN

        IF NEW.deaths_by_unsafe_water_source < @min_deaths_by_unsafe_water_source
THEN

                SET NEW.deaths_by_unsafe_water_source =
@min_deaths_by_unsafe_water_source;

        ELSEIF NEW.deaths_by_unsafe_water_source >
@max_deaths_by_unsafe_water_source THEN

                SET NEW.deaths_by_unsafe_water_source =
@max_deaths_by_unsafe_water_source;

   END IF;

END//

DELIMITER ;
```

**With the help of this update trigger we provide that new values aren’t out of range.

→DELIMITER //

```
CREATE PROCEDURE iso_unsafe_sanitation_and_air_pollution(IN param VARCHAR(35))

BEGIN

        SELECT year, deaths_by_air_pollution + deaths_by_unsafe_sanitation AS
air_and_sanitation

   FROM nature_deaths
```

```
    WHERE iso_code=param;

END//

DELIMITER ;


CALL iso_unsafe_sanitation_and_air_pollution ('AFG')
```

**In this procedure we basically get the year and the sum of deaths caused by air pollution and unsafe sanitation in years between 1990-2019 for a specific country which is given as an iso_code parameter in the procedure. “CALL iso_unsafe_sanitation_and_air_pollution ('AFG')” is a sample.

Kemal Yılmaz 31097

In this analysis I checked the correlation between child deaths due to inability to grow and deaths due to insufficient breastfeeding.

```
ALTER TABLE health_deaths ADD COLUMN stunting_wasting_total INT;

SET SQL_SAFE_UPDATES = 0;

UPDATE health_deaths SET stunting_wasting_total = (deaths_by_child_stunting +
deaths_by_child_wasting);
```

First I summed up the child wasting and stunting deaths.

```
CREATE VIEW high_death_countries AS

SELECT h.iso_code, h.stunting_wasting_total, cn.continent_code

FROM health_deaths h
```

```
JOIN locatedin cn ON h.iso_code = cn.iso_code
WHERE h.stunting_wasting_total > (
        SELECT AVG(c2.stunting_wasting_total)
   FROM health_deaths c2
   JOIN locatedin cn2 ON c2.iso_code = cn2.iso_code
   WHERE cn2.continent_code != 'REG'
 )
AND cn.continent_code != 'REG';
```

Then I found the countries that are having above average death rates for each year.

```
ALTER VIEW high_death_countries AS
SELECT h.iso_code, h.stunting_wasting_total, cn.continent_code, h.year
FROM health_deaths h
JOIN locatedin cn ON h.iso_code = cn.iso_code
WHERE h.stunting_wasting_total > (
        SELECT AVG(c2.stunting_wasting_total)
   FROM health_deaths c2
   JOIN locatedin cn2 ON c2.iso_code = cn2.iso_code
   WHERE cn2.continent_code != 'REG'
 )
AND cn.continent_code != 'REG';
```

I added the year information

```
CREATE VIEW high_death_countries_from_breastfeeding AS
SELECT h.iso_code, h.deaths_by_discontinued_breastfeeding, cn.continent_code, h.year
FROM diet_deaths h
JOIN locatedin cn ON h.iso_code = cn.iso_code
WHERE h.deaths_by_discontinued_breastfeeding > (
        SELECT AVG(c2.deaths_by_discontinued_breastfeeding)
   FROM diet_deaths c2
   JOIN locatedin cn2 ON c2.iso_code = cn2.iso_code
   WHERE cn2.continent_code != 'REG'
 )
AND cn.continent_code != 'REG';
```

Then I found the countries that are having above average death rates for each year.

```
SELECT iso_code, year
FROM high_death_countries
WHERE (iso_code,year) IN (
   SELECT iso_code,year
   FROM high_death_countries_from_breastfeeding
);
```

```
SELECT iso_code, year
FROM high_death_countries hdc
WHERE EXISTS (
    SELECT *
    FROM high_death_countries_from_breastfeeding hdb
    WHERE hdb.iso_code = hdc.iso_code AND hdb.year = hdc.year
);
```

I find the intersection of 2 views and IN and EXISTS gives the same result of 905 lines.

```
SELECT
cd.iso_code,cd.year,cd.stunting_wasting_total,bf.iso_code,bf.year,bf.deaths_by_discontinued_br
eastfeeding
FROM high_death_countries cd
JOIN high_death_countries_from_breastfeeding bf
ON cd.iso_code = bf.iso_code and cd.year = bf.year;
```

I find the intersection of 2 views without in and exists

```
SELECT MAX(deaths_by_discontinued_breastfeeding) FROM diet_deaths;
SELECT MIN(deaths_by_discontinued_breastfeeding) FROM diet_deaths;
```

Finding max and min of the discontinued breastfeeding death rates.

```
SET SQL_SAFE_UPDATES = 0;

ALTER TABLE diet_deaths
ADD CONSTRAINT value_constraint
```

CHECK (33106 >= deaths_by_discontinued_breastfeeding and 0 <= deaths_by_discontinued_breastfeeding);

I put the constraint

INSERT INTO diet_deaths

VALUES ('USA',2023,-1,-1,-1,-1,-1,-1,-1,-1,-1);

DROP TRIGGER IF EXISTS my_insert_trigger;

DELIMITER //

CREATE TRIGGER my_insert_trigger

BEFORE INSERT ON diet_deaths

FOR EACH ROW

BEGIN

  IF NEW.deaths_by_discontinued_breastfeeding < (SELECT MIN(deaths_by_discontinued_breastfeeding) FROM diet_deaths) THEN

    SET NEW.deaths_by_discontinued_breastfeeding = (SELECT MIN(deaths_by_discontinued_breastfeeding) FROM diet_deaths);

  ELSEIF NEW.deaths_by_discontinued_breastfeeding > (SELECT MAX(deaths_by_discontinued_breastfeeding) FROM diet_deaths) THEN

    SET NEW.deaths_by_discontinued_breastfeeding = (SELECT MAX(deaths_by_discontinued_breastfeeding) FROM diet_deaths);

  END IF;

END;

DELIMITER ;

DROP TRIGGER IF EXISTS my_update_trigger;

DELIMITER //

```
CREATE TRIGGER my_update_trigger

BEFORE UPDATE ON diet_deaths

FOR EACH ROW

BEGIN

   IF NEW.deaths_by_discontinued_breastfeeding < (SELECT
MIN(deaths_by_discontinued_breastfeeding) FROM diet_deaths) THEN

      SET NEW.deaths_by_discontinued_breastfeeding = (SELECT
MIN(deaths_by_discontinued_breastfeeding) FROM diet_deaths);

   ELSEIF NEW.deaths_by_discontinued_breastfeeding > (SELECT
MAX(deaths_by_discontinued_breastfeeding) FROM diet_deaths) THEN

      SET NEW.deaths_by_discontinued_breastfeeding = (SELECT
MAX(deaths_by_discontinued_breastfeeding) FROM diet_deaths);

   END IF;

END;


DELIMITER ;


DROP Procedure IF EXISTS breast_feeding_deaths_every_year;

DELIMITER //


CREATE PROCEDURE breast_feeding_deaths_every_year(IN param VARCHAR(3))

BEGIN

        SELECT year,iso_code,deaths_by_discontinued_breastfeeding

   FROM diet_deaths
```

```
    WHERE iso_code = param;


END//

DELIMITER ;


CALL breast_feeding_deaths_every_year("AFG");

CALL breast_feeding_deaths_every_year("USA");

CALL breast_feeding_deaths_every_year("TUR");
```

The link for our Github repository is https://github.com/kerimdemir9/Cs306-Project