CS 306 Project Step 3

Group Members:
-Eyşan Mutlu 29470
-İrem Yeşilbaş 28327
-Merve Bilgi 29117
-Emir Çolakbüyük 29477

GitHub repositories: https://github.com/xmrvx/CS306-project-group7

SQL file names for step3:

STEP3_1-a-CreateViews.sql
STEP3_1-b-JoinsandSetOperators.sql
STEP3_1-c-InAndExists.sql
STEP3_1-d-AggregateOperators.sql

STEP3_2-ConstraintsAnd Triggers_CheckTrigger.sql
STEP3_2-ConstraintsandTriggers_insert_examples.sql

STEP3_3-3Stored Procedure_procedure.sql
STEP3_3-Stored Procedure_call_examples.sql

SQL action log file name for step3: sql_actions_Local_instance_3306.log

STEP3_1-a-CreateViews.sql

In this step, we created a view of countries with high death number by grouping the countries that have a death number of 12000 or more over their population. We calculated this number based on the average of death_num column in HIV_death table. We round the average number as 12000. We repeated this step for countries with low death number. Second view that we created was based on countries with high dead woman ratio. We compared based on a number close to average of woman_ratio column in HIV_cases table. We also created a view of countries with low death woman ratio. Our third view shows countries with low education and high education. We took the average of total_knowledge_rate column in HIV_education table and used that number to write the statement. Finally, we created view for countries with low funding and high funding. Again repeating the same logic, we used average of total_funding column in HIV_funding.

```
SELECT C.ISOYEAR, D.death_num

FROM countries C, HIV_death D
```

```
WHERE D.death_num > 12000;

CREATE VIEW highDeath (ISOYEAR, death_num)
AS SELECT D.ISOYEAR, D.death_num
FROM  HIV_death D
WHERE D.death_num > 12000;

SELECT C.ISOYEAR, D.death_num
FROM countries C, HIV_death D
WHERE D.death_num < 12000;

CREATE VIEW lowDeath (ISOYEAR, death_num)
AS SELECT D.ISOYEAR, D.death_num
FROM  HIV_death D
WHERE D.death_num < 12000;

SELECT C.ISOYEAR, C.woman_ratio
FROM  HIV_cases C
WHERE C.woman_ratio > 55;

CREATE VIEW highWomanDeath(ISOYEAR, woman_ratio)
AS SELECT D.ISOYEAR, D.woman_ratio
FROM  HIV_cases D
WHERE D.woman_ratio > 55;
```

```
SELECT C.ISOYEAR, C.woman_ratio

FROM  HIV_cases C

WHERE C.woman_ratio < 55;


CREATE VIEW lowWomanDeath(ISOYEAR, woman_ratio)

AS SELECT D.ISOYEAR, D.woman_ratio

FROM  HIV_cases D

WHERE D.woman_ratio < 55;


SELECT E.ISOYEAR, E.total_knowledge_rate

FROM  HIV_education E

WHERE E.total_knowledge_rate < 33;


CREATE VIEW lowEducation (ISOYEAR, total_knowledge_rate)

AS SELECT E.ISOYEAR, E.total_knowledge_rate

FROM  HIV_education E

WHERE E.total_knowledge_rate < 33;


SELECT E.ISOYEAR, E.total_knowledge_rate

FROM  HIV_education E

WHERE E.total_knowledge_rate > 33;


CREATE VIEW highEducation (ISOYEAR, total_knowledge_rate)
```

```
AS SELECT E.ISOYEAR, E.total_knowledge_rate
FROM  HIV_education E
WHERE E.total_knowledge_rate > 33;

SELECT F.ISOYEAR, F.total_funding
FROM HIV_funding F
WHERE F.total_funding > 77185666;

CREATE VIEW highFunding (ISOYEAR, total_funding)
AS SELECT F.ISOYEAR, F.total_funding
FROM  HIV_funding F
WHERE F.total_funding > 77185666;

SELECT F.ISOYEAR, F.total_funding
FROM HIV_funding F
WHERE F.total_funding < 77185666;

CREATE VIEW lowFunding (ISOYEAR, total_funding)
AS SELECT F.ISOYEAR, F.total_funding
FROM  HIV_funding F
WHERE F.total_funding < 77185666;
```

STEP3_1-b-JoinsandSetOperators.sql

In this step, we used set operators in order to show relations between our views. For instance, by filtering the countries that had an above average funding in HIV and countries that had less than average in death caused by HIV, we were able to relate that with more funding it is possible to have a decrease in death numbers compared to countries with less fundings in HIV. In order to obtain our sets, we used the NOT IN operator to get rid of the undesired parts of the sets and compare it with our other filtered sets in a WHERE condition resulting in the desired data set. For the next step, we proved that by using LEFT OUTER JOIN, we were able to obtain the same data set we obtained in the previous step.

```
SELECT F.ISOYEAR, L.ISOYEAR, L.death_num, F.total_funding
FROM HIV_death L, HIV_funding F
WHERE F.total_funding > 1844490 AND F.ISOYEAR NOT IN( L.death_num >12000 ) AND
(F.ISOYEAR=L.ISOYEAR);

SELECT F.ISOYEAR, L.ISOYEAR, L.death_num, F.total_funding
FROM HIV_death L, HIV_funding F
WHERE F.total_funding > 1844490 AND F.ISOYEAR NOT IN( L.death_num <12000 ) AND
(F.ISOYEAR=L.ISOYEAR);


/* OUTER JOIN VERSION
*/
SELECT D.death_num, F.total_funding
FROM HIV_death D LEFT OUTER JOIN HIV_funding F
ON F.ISOYEAR = D.ISOYEAR WHERE F.total_funding > 1844490 AND D.death_num >12000;

SELECT D.death_num, F.total_funding
FROM HIV_death D LEFT OUTER JOIN HIV_funding F
ON F.ISOYEAR = D.ISOYEAR WHERE F.total_funding > 1844490 AND D.death_num <12000;
```

STEP3_1-c-InAndExists.sql

In this step, we proved that by using IN and EXISTS for our nested statements we receive the same results and the same count of tuples in our log files.

```
SELECT F.ISOYEAR, F.total_funding, D.death_num
FROM highFunding F, lowDeath D
WHERE F.ISOYEAR = D.ISOYEAR
  AND F.ISOYEAR IN (
    SELECT ISOYEAR
    FROM lowDeath
  );


SELECT F.ISOYEAR, F.total_funding, D.death_num
FROM highFunding F, lowDeath D
WHERE F.ISOYEAR = D.ISOYEAR
  AND EXISTS (
    SELECT S.ISOYEAR
    FROM lowDeath S
    WHERE ISOYEAR = F.ISOYEAR
  );
```

STEP3_1-d-AggregateOperators.sql

In this step, we wrote 4 sql statements that join 2 different tables. In first statement, we joined HIV_cases and HIV_education tables. There are death rates of women in HIV_cases table and there are education rates of countries in HIV_education table. This statement shows ISOYEAR code of countries which have higher values than average in terms of education and woman death. We joined these tables using matching ISOYEAR values. We used average function in this statement to find average value for education and woman death. In the second sql statement, we joined HIV_funding table with countries table. HIV_funding table has funding amounts of countries over several years. Countries table has countries with their specific ISOYEAR codes. We joined these 2 tables based on their matching ISOYEAR codes. Yearcount column gives the total number of years for each country and totalsum column gives total amount of funding for each country. We used sum function to calculate total funding and count as a counter of the rows. In the third sql statement, we joined HIV_death and countries tables. We used max function to calculate the maximum value of death number in HIV_death. Also we used min function to calculate the minimum value of death number in HIV_death. These minimum and maximum values are used to calculate estimated standard deviation. In this statement we can also see year count due to count function and we can see number of total death due to sum function. Final sql statement shows estimated standard deviation of each country and their total_knowledge_rate. We used minimum and maximum functions to calculate the estimated standard deviation. HIV_death and knowledge tables were joined by using ISOYEAR code.

```
SELECT C.woman_ratio, C.ISOYEAR, AVG(E.total_knowledge_rate) AS higherthanavr
FROM HIV_cases C
INNER JOIN HIV_education E ON C.ISOYEAR = E.ISOYEAR
WHERE C.woman_ratio > (SELECT AVG(woman_ratio) FROM HIV_cases)
GROUP BY C.woman_ratio, C.ISOYEAR
HAVING AVG(E.total_knowledge_rate) <(SELECT AVG(total_knowledge_rate) FROM
HIV_education);

SELECT c.country_name, COUNT(*) AS yearcount, SUM(f.total_funding) AS totalsum
FROM HIV_funding f
JOIN countries c ON f.ISOYEAR = c.ISOYEAR
GROUP BY c.country_name
HAVING totalsum > AVG(f.total_funding);

SELECT c.country_name, COUNT(*) AS yearcount, SUM(D.death_num) AS totaldeath
,MIN(D.death_num) AS mindeath
,MAX(D.death_num) AS maxdeath
,(MAX(D.death_num) - MIN(D.death_num))/4 AS estimated_std
FROM HIV_death D
JOIN countries c ON D.ISOYEAR = c.ISOYEAR
GROUP BY c.country_name
HAVING totaldeath > AVG(D.death_num);

SELECT  c.country_name,
      ((MAX(D.death_num) - MIN(D.death_num))/4)AS estimated_std,
      e.total_knowledge_rate
  FROM HIV_death D
  INNER JOIN countries c
  ON D.ISOYEAR = c.ISOYEAR
  INNER JOIN knowledge e
  ON e.country_name = c.country_name
  GROUP BY c.country_name, e.total_knowledge_rate
  HAVING e.total_knowledge_rate>33;
```

STEP3_2-ConstraintsAndTriggers_CheckTrigger.sql

First of all we determined minimum and maximum values in HIV_death table. Minimum number was 0 and maximum value was 1844480. Our purpose in this step was to add a general constraint to alter the table which prevents entering values lower than minimum and higher than maximum. We used ALTER TABLE command to change the chosen table in order to add a constraint. Then we used CHECK command to check if the input is between the range of minimum and maximum. Then we wrote before insert and before update triggers. Purpose was to fix given values if they are out of the range. These triggers prevents program to crash or fail. Since triggers handle exceptions, program continues computing even if the given input is out of range.

```
SELECT MIN(death_num), MAX(death_num) FROM hiv_death;

ALTER TABLE HIV_death
ADD CONSTRAINT dtc
CHECK(death_num >= 0 AND
      death_num <= 1844490);

-- Create a "before insert" trigger
DELIMITER $$
CREATE TRIGGER trg_HIV_death_before_insert
BEFORE INSERT ON HIV_death
FOR EACH ROW
BEGIN
   IF NEW.death_num < 0 THEN
      SET NEW.death_num = 0;
   ELSEIF NEW.death_num > 1844490 THEN
      SET NEW.death_num = 1844490;
   END IF;
END$$
DELIMITER ;

-- Create a "before update" trigger
DELIMITER $$
CREATE TRIGGER trg_HIV_death_before_update
BEFORE UPDATE ON HIV_death
FOR EACH ROW
BEGIN
   IF NEW.death_num < 0 THEN
      SET NEW.death_num = 0;
   ELSEIF NEW.death_num > 1844490 THEN
      SET NEW.death_num = 1844490;
   END IF;
```

```
END$$
DELIMITER ;
```

STEP3_2-ConstraintsandTriggers_insert_examples.sql

In this step, we tested our constraints and triggers which were previously written. We tested with 3 insert statements. First statement successfully inserted 2022-Turkey value in a new row. Because the number 15000 is between minimum and maximum. However second statement couldn't inserted into the table since -5 is lower than the minimum value. This problem was cathed with the help of written constraints and triggers. Also third statement couldn't inserted into the table because 999999999999999 is higher than maximum value.

```
INSERT INTO HIV_death (dyear, death_num, ISOYEAR) VALUES (2022, 15000,
'2022-Turkey');
INSERT INTO HIV_death (dyear, death_num, ISOYEAR) VALUES (2022, -5, '2022-Turkey');
INSERT INTO HIV_death (dyear, death_num, ISOYEAR) VALUES (2023, 999999999999999,
'2023-France');
```

STEP3_3-3Stored Procedure_procedure.sql

In this step, we combined all the views in a single view in order to avoid any missing values that would have caused problems when searching for the requested ISOYEAR. For instance, some views did not include ISOYEAR's that existed on different views causing the call step to give errors while searching for the requested values. To avoid such instances, we created an empty view and filled it with views that we have already created. For values that did not exist in our views, we inserted Null values according to their ISOYEAR's. By doing so, we were able to keep the main tables unchanged and avoided calls to return an error.

```
CREATE OR REPLACE VIEW hiv_summary AS
SELECT c.ISOYEAR,
    COALESCE(hed.total_knowledge_rate, null) AS total_knowledge_rate,
    COALESCE(hd.death_num, null) AS death_num,
    COALESCE(hf.total_funding, null) AS total_funding,
    COALESCE(hc.woman_ratio, null) AS woman_ratio
FROM countries c
LEFT JOIN HIV_education hed ON c.ISOYEAR = hed.ISOYEAR
LEFT JOIN HIV_death hd ON c.ISOYEAR = hd.ISOYEAR
LEFT JOIN HIV_funding hf ON c.ISOYEAR = hf.ISOYEAR
LEFT JOIN HIV_cases hc ON c.ISOYEAR = hc.ISOYEAR
ORDER BY c.ISOYEAR;
```

```
DROP PROCEDURE IF EXISTS 306_step2.searchedISOYEAR;

DELIMITER //

CREATE PROCEDURE searchedISOYEAR(IN isocheck VARCHAR(50))
BEGIN
   SELECT S.ISOYEAR, S.death_num, S.total_knowledge_rate,S.total_funding,S.woman_ratio
   FROM hiv_summary S
   WHERE S.ISOYEAR = isocheck;

END //

DELIMITER ;
```

STEP3_3-Stored Procedure_call_examples.sql

In this step, we made CALL's to show that for any ISOYEAR the latest view to returns the desired values, total_knowledge_rate/death_num/total_funding/woman_ratio.

```
CALL searchedISOYEAR('AFG2016');
CALL searchedISOYEAR('CPV2007');
CALL searchedISOYEAR('GTM2002');
```