Group: Cockatiel

Group Members:

Şimal Yücel 29420
Ulaş Meriç 29244
Arif Kemal Sarı 28999
Tolga Başak 29074
Emre Bülbül 28882

GITHUB LINK:

https://github.com/tolgabasak/cs306/tree/main/Step3Logs

Create View Statements:

```
CREATE VIEW `high_gdp_countries` AS
SELECT `Code`, `Country`, `Year`, `GDP`
FROM `locationdataset`
WHERE `GDP` > (SELECT AVG(`GDP`) FROM `locationdataset` WHERE `GDP` IS NOT NULL);

CREATE VIEW `low_death_rate_countries` AS
SELECT `Code`, `Country`, `Year`, `Deaths  AirP(%)`
FROM `sharedeathsairpollution`
WHERE `Deaths  AirP(%)` < (SELECT AVG(`Deaths  AirP(%)`) FROM `sharedeathsairpollution` WHERE
`Deaths  AirP(%)` IS NOT NULL);

CREATE
   ALGORITHM = UNDEFINED
   DEFINER = `root`@`localhost`
   SQL SECURITY DEFINER
VIEW `emissions_and_ozone` AS
   SELECT
      `l`.`Country` AS `Country`,
      `l`.`Year` AS `Year`,
      `l`.`Code` AS `Code`,
      `c`.`Annual CO2 emissions` AS `Annual CO2 emissions`,
      `f`.`Fossil fuels (TWh)` AS `Fossil fuels (TWh)`,
      `o`.`Ozone concentration-StateofGlobalAir` AS `Ozone concentration-StateofGlobalAir`
   FROM
      (((`locationdataset` `l`
      JOIN `co2emission` `c` ON (((`l`.`Year` = `c`.`Year`)
         AND (`l`.`Code` = `c`.`Code`))))
      JOIN `fossilfuelconsumptionfinal` `f` ON (((`l`.`Year` = `f`.`Year`)
         AND (`l`.`Code` = `f`.`Code`))))
      JOIN `ozonecleaned` `o` ON (((`l`.`Year` = `o`.`Year`)
         AND (`l`.`Code` = `o`.`Code`))));


CREATE
   ALGORITHM = UNDEFINED
   DEFINER = `root`@`localhost`
   SQL SECURITY DEFINER
VIEW `gdp_and_emissions` AS
   SELECT
```

```
        `l`.`Country` AS `Country`,
        `l`.`Year` AS `Year`,
        `l`.`Code` AS `Code`,
        `l`.`GDP` AS `GDP`,
        `c`.`Annual CO2 emissions` AS `Annual CO2 emissions`,
        `f`.`Fossil fuels (TWh)` AS `Fossil fuels (TWh)`
    FROM
        ((`locationdataset` `l`
        JOIN `co2emission` `c` ON (((`l`.`Year` = `c`.`Year`)
            AND (`l`.`Code` = `c`.`Code`))))
        JOIN `fossilfuelconsumptionfinal` `f` ON (((`l`.`Year` = `f`.`Year`)
            AND (`l`.`Code` = `f`.`Code`))));


CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `gdp_and_pollution` AS
    SELECT
        `l`.`Country` AS `Country`,
        `l`.`Year` AS `Year`,
        `l`.`Code` AS `Code`,
        `l`.`GDP` AS `GDP`,
        `o`.`Ozone concentration-StateofGlobalAir` AS `Ozone concentration-StateofGlobalAir`,
        `s`.`Deaths  AirP(%)` AS `Deaths  AirP(%)`
    FROM
        ((`locationdataset` `l`
        JOIN `ozonecleaned` `o` ON (((`l`.`Year` = `o`.`Year`)
            AND (`l`.`Code` = `o`.`Code`))))
        JOIN `sharedeathsairpollution` `s` ON (((`l`.`Year` = `s`.`Year`)
            AND (`l`.`Code` = `s`.`Code`))));

CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `population_and_emissions` AS
    SELECT
        `l`.`Country` AS `Country`,
        `l`.`Year` AS `Year`,
        `l`.`Code` AS `Code`,
        `l`.`Population` AS `Population`,
        `c`.`Annual CO2 emissions` AS `Annual CO2 emissions`,
        `w`.`Wind per capita (kWh - equivalent)` AS `Wind per capita (kWh - equivalent)`
    FROM
        ((`locationdataset` `l`
        JOIN `co2emission` `c` ON (((`l`.`Year` = `c`.`Year`)
            AND (`l`.`Code` = `c`.`Code`))))
        JOIN `windcleaned` `w` ON (((`l`.`Year` = `w`.`Year`)
            AND (`l`.`Code` = `w`.`Code`))));

CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `wind_and_airpollution` AS
    SELECT
        `l`.`Country` AS `Country`,
        `l`.`Year` AS `Year`,
        `l`.`Code` AS `Code`,
        `w`.`Wind per capita (kWh - equivalent)` AS `Wind per capita (kWh - equivalent)`,
        `s`.`Deaths  AirP(%)` AS `Deaths  AirP(%)`
    FROM
        ((`locationdataset` `l`
        JOIN `windcleaned` `w` ON (((`l`.`Year` = `w`.`Year`)
            AND (`l`.`Code` = `w`.`Code`))))
        JOIN `sharedeathsairpollution` `s` ON (((`l`.`Year` = `s`.`Year`)
            AND (`l`.`Code` = `s`.`Code`))));
```

Stored Procedure Statement:

```
CREATE PROCEDURE `GetCovid19DataByCountry` (IN iso_code VARCHAR(50))
BEGIN
  SELECT `date`, `cases`, `deaths`
  FROM `covid19_data`
  WHERE `iso_code` = iso_code
  ORDER BY `date` DESC;
END;
//

DELIMITER ;

CALL `GetCovid19DataByCountry`('US');
CALL `GetCovid19DataByCountry`('GB');
```

Constraint and Trigger Statements:

```
DELIMITER //

CREATE TRIGGER `co2emission_before_insert`
BEFORE INSERT ON `co2emission`
FOR EACH ROW
BEGIN
  IF NEW.`Annual CO2 emissions` < min_value THEN
    SET NEW.`Annual CO2 emissions` = min_value;
  ELSEIF NEW.`Annual CO2 emissions` > max_value THEN
    SET NEW.`Annual CO2 emissions` = max_value;
  END IF;
END;
//

CREATE TRIGGER `co2emission_before_update`
BEFORE UPDATE ON `co2emission`
FOR EACH ROW
BEGIN
  IF NEW.`Annual CO2 emissions` < min_value THEN
    SET NEW.`Annual CO2 emissions` = min_value;
  ELSEIF NEW.`Annual CO2 emissions` > max_value THEN
    SET NEW.`Annual CO2 emissions` = max_value;
  END IF;
END;
//

DELIMITER ;
```

Comparing general constraint and and trigger methods:

General Restrictions:

Pros:

Simplicity: Constraints are more straightforward to comprehend and apply since they clearly specify the guidelines for the data in the table design.
Performance: Since constraints are evaluated during data processing and do not call for additional code execution, they are typically faster than triggers.
Immediate enforcement: When data is manipulated, constraints immediately enforce data integrity, making sure that no erroneous data is ever placed in the table.

Cons:

Limited Flexibility: Constraints may have a limited range of logic that can be used and may not be appropriate for more intricate data validation criteria.
Limited scope: Constraints may not take into account relationships or data in other tables and only apply to the table they are defined on.

Triggers:

Pros:

Flexibility: Since triggers provide more complicated logic and calculations, they provide greater flexibility in terms of the data validation criteria that can be used.
Cross-table operations: Because triggers can access and modify data from linked tables, they can be used to enforce data integrity across many tables.

Cons:

Complexity: Because triggers include procedural code and may be more difficult to trace back during troubleshooting, they can be tougher to comprehend and maintain.
Triggers can affect database performance since they are executed each time an event (such as an INSERT, UPDATE, or DELETE) takes place on the table, potentially slowing down operations.