CS306 Group Project – Step 3

Group Number 16

Berke Ceylan	27895
İsmail Çakmak	29496
Nisa Erdal	28943
Salih Kaya	27890
Yunus Topçu	28880

Can be accessed at https://github.com/Salihmaya/CS306_Group16

Tobacco Consumption: Causes and Effects

Introduction

In Step 3 of the project, further relationships between the datasets and meaningful ways of storing these relationships were pursued. As illustrated in detail below, datasets were utilized for creating views to highlight significant data points that differed from the rest of the entries. These views were then combined using JOIN and EXCEPT statements to test where two views could jointly give insight. Views were further utilized to write IN / EXISTS for comparison between two views and Aggregate operators, namely (SUM, AVG, COUNT, MIN, MAX) were practiced, exploring the datasets even further.

In order to make the current tables more practical and increase future usability, logical Constraints on table inserts and appropriate Triggers for insertion / updates were added. Stored Procedures were also created to provide practical access to the tables that would output related data entries for a given country iso_code.

Views

Datasets were compared to their respective averages and views were created for:

Death Rates that were **lower than the average**:

7 lines (6 sloc) | 247 Bytes

```sql
CREATE VIEW low_death_rate_smoking AS
SELECT c.country_name, d.year
FROM death_rate_smoking d
JOIN countries c ON d.iso_code = c.iso_code
WHERE d.death_rate < (SELECT AVG(death_rate) FROM death_rate_smoking);

SELECT * FROM low_death_rate_smoking;
```

And Lung Cancer Deaths, Public Health Expenditures, Smoking Quit Helps and Advertisement Bans that were **greater than the average**:

9 lines (7 sloc) | 344 Bytes

```sql
CREATE VIEW high_lung_cancer_deaths AS
SELECT c.country_name, l.year
FROM lung_cancer_deaths l
JOIN countries c ON l.iso_code = c.iso_code
WHERE l.male_death_rate > (SELECT AVG(male_death_rate) FROM lung_cancer_deaths)
AND l.female_death_rate > (SELECT AVG(female_death_rate) FROM lung_cancer_deaths);

SELECT * FROM high_lung_cancer_deaths;

```

7 lines (6 sloc) | 307 Bytes

```sql
CREATE VIEW high_expenditure_years AS
SELECT c.country_name, p.year, p.expenditure_pc_gdp
FROM public_health_expenditure p
JOIN countries c ON p.iso_code = c.iso_code
WHERE p.expenditure_pc_gdp > (SELECT AVG(expenditure_pc_gdp) FROM public_health_expenditure);

SELECT * FROM high_expenditure_years;
```

10 lines (7 sloc) | 327 Bytes

```sql

DROP VIEW high_advertisementban;

CREATE VIEW high_advertisementban AS
SELECT c.country_name, adv.ban_indicator, adv.year
FROM cigarette_advertisements adv
JOIN countries c ON adv.iso_code = c.iso_code
WHERE adv.ban_indicator > (SELECT AVG(ban_indicator) FROM cigarette_advertisements) ;

SELECT * FROM high_advertisementban;
```

11 lines (6 sloc) | 244 Bytes

```sql
CREATE VIEW High_Smoke_Quit_Help AS

SELECT x.name, y.year

FROM smoking_quit_help y

JOIN countries x ON y.iso_code =x.iso_code

Where y.help_indicator > (SELECT AVG(help_indicator) FROM smoking_quit_help);

Select *from High_Smoke_Quit_Help;
```

Joins and Set Operations

To further explore these datasets, these views were joined where it was logically applicable to do so, creating groups of two:

EXCEPT and OUTER JOIN were used interchangeably in order to illustrate that they can be used to give the same results.

- High Lung Cancer Deaths vs High Advertisement Bans

11 lines (10 sloc) | 302 Bytes

```sql
SELECT country_name, year
FROM high_advertisementban
EXCEPT
SELECT country_name, year
FROM high_lung_cancer_deaths;

SELECT a.country_name, a.year
FROM high_advertisementban a
LEFT OUTER JOIN high_lung_cancer_deaths l
ON l.country_name = a.country_name AND l.year = a.year
WHERE l.country_name IS NULL
```

- Low Death Rate vs High Lung Cancer Deaths

25 lines (21 sloc) | 800 Bytes

```sql
CREATE TABLE low_eath_rate_vs_high_lung_cancer_leftjoin (
   country_name VARCHAR(50),
   year INT
);

INSERT INTO low_eath_rate_vs_high_lung_cancer_leftjoin (country_name, year)
SELECT high_lung_cancer_deaths.country_name, high_lung_cancer_deaths.year
FROM high_lung_cancer_deaths
LEFT JOIN low_death_rate_smoking
ON high_lung_cancer_deaths.country_name = low_death_rate_smoking.country_name
AND high_lung_cancer_deaths.year = low_death_rate_smoking.year
WHERE low_death_rate_smoking.country_name IS NULL;

CREATE TABLE low_eath_rate_vs_high_lung_cancer_except (
   country_name VARCHAR(50),
   year INT
);
INSERT INTO low_eath_rate_vs_high_lung_cancer_except (country_name, year)
SELECT country_name, year
FROM high_lung_cancer_deaths
EXCEPT
SELECT country_name, year
FROM low_death_rate_smoking


```

- High Health Expenditure vs High Lung Cancer Deaths

12 lines (11 sloc) | 457 Bytes | Raw | Blame

```sql
SELECT high_expenditure_years.country_name, high_expenditure_years.year
FROM high_expenditure_years
LEFT JOIN high_lung_cancer_deaths
ON high_expenditure_years.country_name = high_lung_cancer_deaths.country_name
AND high_expenditure_years.year = high_lung_cancer_deaths.year
WHERE high_lung_cancer_deaths.country_name IS NULL;

SELECT country_name, year
FROM high_expenditure_years
EXCEPT
SELECT country_name, year
FROM high_lung_cancer_deaths;
```

- High Quit Help vs Low Death Rate

12 lines (11 sloc) | 496 Bytes | Raw | Blame

```sql
CREATE TABLE High_Quit_Help_and_Low_Death_Rate_outerjoin(
   country_name VARCHAR(50),
   year INT
);

INSERT INTO High_Quit_Help_and_Low_Death_Rate_outerjoin(country_name, year)
SELECT high_smoke_quit_help.country_name, low_death_rate_smoking.year
FROM high_smoke_quit_help
LEFT OUTER JOIN low_death_rate_smoking
ON high_smoke_quit_help.country_name=low_death_rate_smoking.country_name
AND high_smoke_quit_help.year=low_death_rate_smoking.year
Where low_death_rate_smoking.country_name IS NULL;
```

- High Advertisement Ban vs Low Death Rate

6 lines (5 sloc) | 203 Bytes | Raw | Blame

```sql

CREATE VIEW high_bans_and_low_smoking_deaths AS
SELECT a.country_name, a.year
FROM high_advertisementban a
LEFT OUTER JOIN low_death_rate_smoking l
ON l.country_name = a.country_name AND l.year = a.year
```

IN and EXISTS Statements

For the previously created views, statements that would result in the same outputs were created to showcase and practice that IN and EXISTS statements can be used interchangeably in this context.

Following are the practices on various views:

Smoking Death Rate View

21 lines (17 sloc) | 328 Bytes

Raw Blame

```sql
SELECT *
FROM death_rate_smoking
WHERE iso_code IN (
  SELECT iso_code
  FROM death_rate_smoking
  WHERE year = 1990
  AND death_rate > 20
);

SELECT *
FROM death_rate_smoking
WHERE EXISTS (
  SELECT iso_code
  FROM death_rate_smoking
  WHERE year = 1990
  AND death_rate > 20
  AND iso_code = death_rate_smoking.iso_code
);



```

High Lung Cancer Deaths and High Expenditure Years View

15 lines (13 sloc) | 329 Bytes

Raw Blame

```sql
SELECT *
FROM high_lung_cancer_deaths
WHERE (country_name, year) IN
  (SELECT country_name, year
   FROM high_expenditure_years);


SELECT *
FROM high_lung_cancer_deaths l
WHERE EXISTS (
  SELECT country_name, year
  FROM high_expenditure_years e
  WHERE l.country_name = e.country_name
    AND l.year = e.year
);
```

High Lung Cancer Deaths and High Advertisement Bans View

15 lines (13 sloc) | 315 Bytes

Raw Blame

```sql
SELECT *
FROM high_lung_cancer_deaths
WHERE (country_name, year) IN
  (SELECT country_name, year
   FROM high_advertisementban);

SELECT *
FROM high_lung_cancer_deaths l
WHERE EXISTS (
  SELECT country_name, year
  FROM high_advertisementban a
  WHERE l.country_name = a.country_name
    AND l.year = a.year

);
```

High Advertisement Bans and Low Smoking Death Rate View

14 lines (13 sloc) | 328 Bytes

Raw Blame

```sql
SELECT  *
FROM high_advertisementban
WHERE ban_indicator = 5
AND country_name IN (SELECT distinct country_name
                                        FROM low_death_rate_smoking);

SELECT *
FROM high_advertisementban hb
WHERE EXISTS (
    SELECT *
    FROM low_death_rate_smoking ls
    WHERE hb.country_name = ls.country_name
    AND hb.ban_indicator = 5
);
```

High Smoking Quit Help and Low Smoking Death Rate View

16 lines (14 sloc) | 364 Bytes

Raw Blame

```sql
-- Select statements were utilized with IN and EXISTS OPERATORS

Select *
FROM high_smoke_quit_help
Where (country_name, year) IN
(SELECT country_name,year
FROM low_death_rate_smoking);

SELECT *
FROM high_smoke_quit_help q
WHERE EXISTS (
  SELECT country_name, year
  FROM low_death_rate_smoking d
  WHERE q.country_name = d.country_name
  AND q.year = d.year
);
```

It has been concluded that IN and EXISTS statements can indeed be used interchangeably and the outputs for the SELECT statements are the same for each view.

Aggregate Operators

In order to practice with the aggregate operators which were chosen as:

- SUM
- AVG
- COUNT
- MIN
- MAX

Previously created views were explored to practice with these statements and each were used at least once in the following code examples, whereas in some cases a combination of them were used.

COUNT Operator on Smoking Death Rate and Lung Cancer Deaths

13 lines (5 sloc) | 404 Bytes

```sql
-- finds the lung cancer death rates and death rate smoking of the countries results to only include groups with a only death_rates highger than 40.

SELECT COUNT(DISTINCT d.iso_code, d.year) AS common_columns
FROM death_rate_smoking AS d
JOIN lung_cancer_deaths AS l ON d.iso_code = l.iso_code AND d.year = l.year
WHERE d.death_rate > 40 AND (l.male_death_rate > 40 OR l.female_death_rate > 40);
```

MIN Operator on Lung Cancer Deaths and Advertisement Bans

14 lines (13 sloc) | 476 Bytes

```sql
-- finds the lung cancer death rates of the countries with minimum advertisement bans in a specific year

SELECT lc.iso_code, lc.year, lc.male_death_rate, lc.female_death_rate
FROM lung_cancer_deaths lc
INNER JOIN (
  SELECT ca.iso_code, ca.year
  FROM cigarette_advertisements ca
  GROUP BY ca.iso_code, ca.year
  HAVING MIN(ca.ban_indicator) = (
    SELECT MIN(ban_indicator)
    FROM cigarette_advertisements
  )
) AS ca
ON lc.iso_code = ca.iso_code AND lc.year = ca.year;
```

MAX Operator on Advertisement Bans and Smoking Death Rate

15 lines (8 sloc) | 369 Bytes

```sql


-- create table of years and the max death rate of the countries that bans ads most strictly

SELECT ads.year, MAX(smk.death_rate)
FROM cigarette_advertisements ads INNER JOIN death_rate_smoking smk
ON ads.iso_code = smk.iso_code
WHERE ads.ban_indicator = (      SELECT MAX(ban_indicator)
                                 FROM cigarette_advertisements)
GROUP BY ads.year
HAVING ads.year > 2000
```

AVG Operator on Smoking Quit Help and <u>Low Smoking Death Rate</u>

7 lines (7 sloc) | 434 Bytes

```sql
-- AVG Statement was used to select the help indicator values bigger than the average
SELECT q.iso_code, q.year, AVG(q.help_indicator) AS avg_help_indicator
FROM smoking_quit_help q
LEFT JOIN low_death_rate_smoking d ON q.iso_code = (SELECT iso_code FROM countries WHERE country_name = d.country_name) AND q.year = d.year
GROUP BY q.iso_code, q.year
HAVING AVG(q.help_indicator) > (Select AVG(help_indicator)
from smoking_quit_help);
```

<u>SUM and AVG Operators</u> on Public Health Expenditure

6 lines (6 sloc) | 364 Bytes

```sql
SELECT he.year, SUM(he.expenditure_pc_gdp) AS total_expenditure_pc_gdp, AVG(he.expenditure_pc_gdp) AS avg_expenditure_pc_gdp
FROM public_health_expenditure he
INNER JOIN high_expenditure_years hey ON
        he.year = hey.year AND he.iso_code = (SELECT iso_code FROM countries WHERE country_name = hey.country_name)
GROUP BY he.year
HAVING COUNT(he.iso_code) > 1;
```

Hence all operators were used on both individual views and their combinations in this step.

Constraints and Triggers

To ensure that any future alterations & insertions to the tables would comply with the current format of the data entries:

- Constraints regarding the respective attributes were added
- Triggers were added for the cases:
    - Before Insertion to the Table
    - Before Update to the Table

Following are the constraints and triggers created for each table:

Smoking Death Rate

42 lines (32 sloc) | 1.44 KB   Raw Blame

```sql
-- Step 1: Determine the minimum and maximum values of the death_rate column
SELECT MIN(death_rate) AS min_rate, MAX(death_rate) AS max_rate
FROM death_rate_smoking;

-- Step 2: Add a BEFORE INSERT trigger to ensure the death_rate value is within the allowed range
DELIMITER $$
CREATE TRIGGER death_rate_trigger
BEFORE INSERT ON death_rate_smoking
FOR EACH ROW
BEGIN
    IF NEW.death_rate NOT BETWEEN (SELECT MIN(death_rate) FROM death_rate_smoking) AND (SELECT MAX(death_rate) FROM death_rate_smoking) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'The death rate value must be within the allowed range.';
    END IF;
END $$
DELIMITER ;

INSERT INTO death_rate_smoking (iso_code, year, death_rate)
VALUES ('USA', 2022, 1000);

-- Step 3: Add a BEFORE UPDATE trigger to fix the death_rate value if it's outside the allowed range
DELIMITER $$
CREATE TRIGGER death_rate_smoking_before_update
BEFORE UPDATE ON death_rate_smoking
FOR EACH ROW
BEGIN
    IF NEW.death_rate < (SELECT MIN(death_rate) FROM death_rate_smoking) THEN
        SET NEW.death_rate = (SELECT MIN(death_rate) FROM death_rate_smoking);
    ELSEIF NEW.death_rate > (SELECT MAX(death_rate) FROM death_rate_smoking) THEN
        SET NEW.death_rate = (SELECT MAX(death_rate) FROM death_rate_smoking);
    END IF;
END $$
DELIMITER ;

INSERT INTO death_rate_smoking (iso_code, year, death_rate)
VALUES ('USA', 2022, 1000);

UPDATE death_rate_smoking SET death_rate = 30.0 WHERE iso_code = 'USA';
```

Lung Cancer Deaths

71 lines (62 sloc) | 1.6 KB

Raw Blame

```sql
-- the constraints for male_death_rate and female_death_rate

ALTER TABLE lung_cancer_deaths
ADD CONSTRAINT male_death_rate_check CHECK (male_death_rate >= 0 AND male_death_rate <= 100);

ALTER TABLE lung_cancer_deaths
ADD CONSTRAINT female_death_rate_check CHECK (female_death_rate >= 0 AND female_death_rate <= 100);
-- insertion check

INSERT INTO lung_cancer_deaths(iso_code, year, male_death_rate, female_death_rate)
VALUES ('XXX', 2023, -5, 140);

-- the before insertion trigger

DELIMITER //
CREATE TRIGGER before_insert_male
BEFORE INSERT ON lung_cancer_deaths
FOR EACH ROW
BEGIN
  IF NEW.male_death_rate < 0 THEN
    SET NEW.male_death_rate = 0;
  ELSEIF NEW.male_death_rate > 100 THEN
    SET NEW.male_death_rate = 100;
  END IF;
END;
//
DELIMITER ;

DELIMITER //
CREATE TRIGGER before_insert_female
BEFORE INSERT ON lung_cancer_deaths
FOR EACH ROW
BEGIN
  IF NEW.female_death_rate < 0 THEN
    SET NEW.female_death_rate = 0;
  ELSEIF NEW.female_death_rate > 100 THEN
    SET NEW.female_death_rate = 100;
  END IF;
END;
//
DELIMITER ;

-- the before update trigger

DELIMITER //
CREATE TRIGGER before_update_male
BEFORE UPDATE ON lung_cancer_deaths
FOR EACH ROW
BEGIN
  IF NEW.male_death_rate < 0 THEN
    SET NEW.male_death_rate = 0;
  ELSEIF NEW.male_death_rate > 100 THEN
    SET NEW.male_death_rate = 100;
  END IF;
END;
//
DELIMITER ;

DELIMITER //
CREATE TRIGGER before_update_female
BEFORE UPDATE ON lung_cancer_deaths
FOR EACH ROW
BEGIN
  IF NEW.female_death_rate < 0 THEN
    SET NEW.female_death_rate = 0;
  ELSEIF NEW.female_death_rate > 100 THEN
    SET NEW.female_death_rate = 100;
  END IF;
END;
//
DELIMITER ;
```

Smoking Quit Help

44 lines (36 sloc) | 962 Bytes

Raw Blame

```sql
-- CHECK MIN AND MAX VALUES ON DATASET
SELECT MIN(help_indicator) AS min_indicator, Max(help_indicator) AS max_indicator
FROM smoking_quit_help;

-- CONSTRATINT ADDITION
ALTER TABLE smoking_quit_help
ADD CONSTRAINT help_indicator CHECK (help_indicator BETWEEN 1 AND 5);

-- TRIGGER CREATION WITHIN THE REQUIRED RANGE OF VALUES
DELIMITER //
CREATE TRIGGER trigger_before
Before insert on smoking_quit_help
for each row
Begin

        IF new.help_indicator < 1 THEN
                SET new.help_indicator=1;
        ELSEIF new.help_indicator >5 THEN
                Set new.help_indicator =5;
        end if;
END;
//
DELIMITER ;

DELIMITER //
CREATE TRIGGER trigger_after
Before update on smoking_quit_help
for each row
Begin

        IF new.help_indicator < 1 THEN
                SET new.help_indicator=1;
        ELSEIF new.help_indicator >5 THEN
                Set new.help_indicator =5;
        end if;
END;
//
DELIMITER ;


-- AFTER THE FOLLOWING INSERTION THE MAX VALUE WAS STILL 5
INSERT INTO smoking_quit_help(iso_code)
VALUES ('ABC', 2023, 120);

```

Public Health Expenditure

43 lines (34 sloc) | 1.02 KB   Raw Blame

```sql
-- Add the constraint

ALTER TABLE public_health_expenditure
ADD CONSTRAINT chk_expenditure_pc_gdp CHECK (expenditure_pc_gdp >= 0 AND expenditure_pc_gdp <= 100);

-- Just a check to see whether it actually works or not

INSERT INTO public_health_expenditure(iso_code, year, expenditure_pc_gdp)
VALUES ('WWW', 2023, 120);

-- Add the before insertion trigger

DELIMITER //
CREATE TRIGGER trg_before_insert_expenditure
BEFORE INSERT ON public_health_expenditure
FOR EACH ROW
BEGIN
  IF NEW.expenditure_pc_gdp < 0 THEN
    SET NEW.expenditure_pc_gdp = 0;
  ELSEIF NEW.expenditure_pc_gdp > 100 THEN
    SET NEW.expenditure_pc_gdp = 100;
  END IF;
END;
//
DELIMITER ;

-- Add the before update trigger

DELIMITER //
CREATE TRIGGER trg_before_update_expenditure
BEFORE UPDATE ON public_health_expenditure
FOR EACH ROW
BEGIN
  IF NEW.expenditure_pc_gdp < 0 THEN
    SET NEW.expenditure_pc_gdp = 0;
  ELSEIF NEW.expenditure_pc_gdp > 100 THEN
    SET NEW.expenditure_pc_gdp = 100;
  END IF;
END;
//
DELIMITER ;

```

Advertisement Bans

52 lines (27 sloc) | 922 Bytes

```sql

-- contraint for ban_indicator which needs to be btw 1-5

alter table cigarette_advertisements
add constraint ban_indicator_check CHECK (ban_indicator > 0 and ban_indicator <= 5);



-- insertion check

insert into cigarette_advertisements(iso_code, year, ban_indicator)
values("TUR", 1990, 6);





-- trigger before insertion

DELIMITER $$
CREATE TRIGGER before_insertion_trigger BEFORE INSERT ON cigarette_advertisements
FOR EACH ROW
BEGIN
        IF new.ban_indicator < 1 THEN SET new.ban_indicator = 1;
    ELSEIF new.ban_indicator > 5 THEN SET new.ban_indicator = 5;
    END IF;
END$$

DELIMITER ;



-- trigger before update

DELIMITER $$
CREATE TRIGGER before_update_trigger BEFORE UPDATE ON cigarette_advertisements
FOR EACH ROW
BEGIN
        IF new.ban_indicator < 1 THEN SET new.ban_indicator = 1;
    ELSEIF new.ban_indicator > 5 THEN SET new.ban_indicator = 5;
    END IF;
END$$

DELIMITER ;




show triggers;



```

Following the creation of these constraints and triggers, correct & incorrect insertions / updates were performed on the tables as shown in the log files.

Comparison of General Constraints and Triggers

Creating general constraints and creating triggers are two different methods of enforcing data integrity in a database. Both methods have their pros and cons, and understanding these differences can help determine which approach is best suited for a particular situation.

Pros and Cons of General Constraints:

Pros:

- Enforces data integrity rules directly within the table schema, making them easy to understand and maintain.
- Simpler to implement than triggers, as they are declarative and do not require procedural logic.
- Improved performance compared to triggers, as constraints are evaluated by the database engine at the time of data manipulation.

Cons:

- Limited in complexity and flexibility, as they can only enforce simple validation rules and cannot perform complex logic.
- Cannot reference other tables or columns, limiting their ability to enforce referential integrity.
- Unable to perform custom error handling, resulting in generic error messages.

Pros and Cons of Triggers:

Pros:

- Highly flexible and powerful, as they can contain complex procedural logic and reference other tables or columns.
- Can perform custom error handling, allowing for more informative error messages.
- Can be used to enforce referential integrity and maintain relationships between tables.

Cons:

- More difficult to understand and maintain than general constraints, as they require procedural logic and may be stored separately from the table schema.
- Can negatively impact performance, as triggers are executed by the database engine for each affected row during data manipulation.
- Triggers may introduce unintended side effects, as they can be triggered by cascading actions from other tables.

In conclusion, general constraints are well-suited for enforcing simple data validation rules and offer better performance, while triggers provide more flexibility and power for complex data integrity scenarios. When choosing between the two, it is essential to weigh the pros and cons and consider the specific requirements of the data integrity rules being enforced.

Stored Procedures

To further improve the future functionality of the database(s), stored procedures were created for each table so that for a given iso_code input, related information can be obtained from each and every table without the need to reproduce the statements repetitively.

For practice, one of the stored procedures was created to take an integer input, denoting the "ban indicators" for the Advertisement Ban dataset.

Following are the stored procedure implementations for the tables:

Smoking Death Rate

20 lines (13 sloc) | 476 Bytes

```
DELIMITER $$
CREATE PROCEDURE get_death_rate_smoking(IN p_iso_code VARCHAR(3))
BEGIN
    IF p_iso_code = 'USA' THEN
        SELECT * FROM death_rate_smoking WHERE iso_code = p_iso_code AND year > 2010;
    ELSEIF p_iso_code = 'GBR' THEN
        SELECT * FROM death_rate_smoking WHERE iso_code = p_iso_code AND year < 2010;
    ELSE
        SELECT * FROM death_rate_smoking WHERE iso_code = p_iso_code;
    END IF;
END $$
DELIMITER ;

CALL get_death_rate_smoking('GBR');
```

Public Health Expenditure

23 lines (20 sloc) | 596 Bytes

Raw Blame

```sql
DELIMITER //
CREATE PROCEDURE GetAverageExpenditureByCountry(IN iso_code_param VARCHAR(5))
BEGIN
    SELECT
        c.iso_code,
        c.country_name,
        AVG(p.expenditure_pc_gdp) AS avg_expenditure_pc_gdp
    FROM
        countries c
    JOIN
        public_health_expenditure p ON c.iso_code = p.iso_code
    WHERE
        c.iso_code = iso_code_param
    GROUP BY
        c.iso_code, c.country_name;
END //
DELIMITER ;


-- Trying out the two values as specified in the task

CALL GetAverageExpenditureByCountry('KOR');
CALL GetAverageExpenditureByCountry('JPN');
```

Lung Cancer Deaths

20 lines (19 sloc) | 520 Bytes

Raw Blame

```sql
DELIMITER //
CREATE PROCEDURE GetAverageDeathsByCountry(IN iso_code_param VARCHAR(5))
BEGIN
    SELECT
        c.iso_code,
        AVG(d.male_death_rate) AS avg_male_death_rate,
        AVG(d.female_death_rate) AS avg_female_death_rate
    FROM
        countries c
        JOIN
        lung_cancer_deaths d ON c.iso_code = d.iso_code
    WHERE
        c.iso_code = iso_code_param
    GROUP BY
        c.iso_code, c.country_name;
END //
DELIMITER ;

CALL GetAverageDeathsByCountry('TUR');
CALL GetAverageDeathsByCountry('ITA');
```

Smoking Quit Help

```
DELIMITER //
CREATE PROCEDURE Avg_Help_Indicator(IN param_iso_code VARCHAR (5))
BEGIN
        SELECT
    h.iso_code,
    AVG(q.help_indicator) AS AverageHelpIndicator
        FROM
                countries h
        Join
        smoking_quit_help q ON q.iso_code=h.iso_code
        WHERE
                h.iso_code=param_iso_code
        Group BY
                h.iso_code, h.country_name;
END //
DELIMITER ;

Call Avg_Help_Indicator('TUR')
```

Advertisement Ban

```



DELIMITER $$
CREATE PROCEDURE find_countries(IN ban_indicator INT)
BEGIN
        SELECT *
    FROM cigarette_advertisements c
    WHERE c.ban_indicator = ban_indicator;
END$$

DELIMITER ;


CALL find_countries(5);
CALL find_countries(1);
```

Conclusion

Throughout this project step, our team has successfully explored the relationships between various datasets related to tobacco consumption and its effects, delving deep into the intricacies of the data to uncover meaningful patterns, trends, and correlations. We have demonstrated our SQL skills by creating views, using joins and set operators, writing nested SELECT statements with IN and EXISTS operators, and applying aggregate operators to analyze the data. Furthermore, we have also effectively utilized constraints, triggers, and stored procedures to ensure the integrity, practicality, and maintainability of our database schema.

Our approach to the simple data analysis performed in this step has enabled us to extract valuable insights and expand our understanding of the complex dynamics surrounding tobacco consumption and its public health implications.