Assignment 3 - Penna

CS336

Part 1 - 30%

Powering simple interface to Penna table

API1(candidate, timestamp, precinct)

Given a candidate C, timestamp T and precinct P, return how many votes did the candidate C have at T or largest timestamp T' smaller than T, in case T does not appear in Penna.

```
DROP PROCEDURE IF EXISTS API1;
DELIMITER $$
CREATE PROCEDURE API1(
        IN c VARCHAR(6),
    IN ts Timestamp,
    IN p VARCHAR(50)
)
BEGIN
        DECLARE votesForCandidate INT DEFAULT 0;

        IF (c NOT IN ('Biden','Trump')) THEN
                SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = 'Invalid candidate';
        END IF;

    IF (p NOT IN (SELECT DISTINCT precinct FROM Penna)) THEN
                SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = 'Invalid precinct';
        END IF;

    if (ts >= (SELECT MIN(Timestamp) FROM Penna)) THEN
                SELECT
                        CASE
                                WHEN c='Biden' THEN Biden
                                WHEN c='Trump' THEN Trump
                                ELSE -1
                        END
                INTO votesForCandidate
                FROM Penna
                WHERE precinct=p AND Timestamp <= ts
```

```
                ORDER BY Timestamp DESC
                LIMIT 1;
        END IF;

    SELECT votesForCandidate;
END$$
DELIMITER ;
```

API2(date)

Given a date, return the candidate who had the most votes at the last timestamp for this date as well as how many votes he got. For example the last timestamp for 2020-11-06 will be 2020-11-06 23:51:43.

```
DROP PROCEDURE IF EXISTS API2;
DELIMITER $$
CREATE PROCEDURE API2(
        IN d DATE
)
BEGIN
        DECLARE maxTimestamp Timestamp;
    DECLARE minTimestamp Timestamp;
        DECLARE lastTimestamp Timestamp;

    SELECT MAX(Timestamp) INTO maxTimestamp FROM Penna;
    SELECT MIN(Timestamp) INTO minTimestamp FROM Penna;
    SELECT MAX(Timestamp) INTO lastTimestamp FROM Penna WHERE Timestamp <
DATE_ADD(maxTimestamp, INTERVAL 1 DAY);

    IF (d > maxTimestamp) THEN
                SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = 'Date is after election ended.';
    ELSEIF (d < minTimestamp) THEN
                SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = 'Date is before election began.';
    END IF;

        SELECT
                IF (SUM(Trump) > SUM(Biden), "Trump", "Biden") as winner,
                IF (SUM(Trump) > SUM(Biden), SUM(Trump), SUM(Biden)) as votes
        FROM Penna
        WHERE
                Timestamp = lastTimestamp
        ORDER BY Timestamp DESC
        LIMIT 1;
```

```
END$$
DELIMITER ;
```

API3(candidate)

Given a candidate return top 10 precincts that this candidate won. Order precincts by total votes and list TOP 10 in descending order of totalvotes.

```
DROP PROCEDURE IF EXISTS API3;
DELIMITER $$
CREATE PROCEDURE API3(
        IN candidate VARCHAR(6)
)
BEGIN
        IF (candidate NOT IN ('Biden','Trump')) THEN
                SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = 'Invalid candidate';
        END IF;

        SELECT p.precinct
        FROM (
                SELECT DISTINCT precinct
                FROM Penna
                WHERE 1 = CASE
                                WHEN candidate='Biden' THEN Biden>Trump
                                WHEN candidate='Trump' THEN Trump>Biden
                                ELSE -1
                        END
        ) p
        ORDER BY (
                SELECT MAX(totalvotes) FROM Penna WHERE precinct=p.precinct
        ) DESC
        LIMIT 10;
END$$
DELIMITER ;
```

API4(precinct)

Given a precinct, show who won this precinct (Trump or Biden) as well as what percentage of total votes went to the winner.

```
DROP PROCEDURE IF EXISTS API4;
DELIMITER $$
CREATE PROCEDURE API4(
        IN p VARCHAR(64)
```

```
)
BEGIN
        IF (p NOT IN (SELECT DISTINCT precinct FROM Penna)) THEN
                SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = 'Invalid precinct';
        END IF;

        SELECT
                IF(Biden>Trump,'Biden','Trump') as winner,
        CONCAT(FORMAT(IF(
                Biden>Trump,
        Biden/totalVotes,
        Trump/totalVotes
        )*100, 2), "%") as percentage
        FROM (SELECT * FROM Penna WHERE precinct=p) penna
        WHERE penna.Timestamp = (SELECT MAX(Timestamp) FROM penna);
END$$
DELIMITER ;
```

API5(string)

Given a string s of characters, create a stored procedure which determines who won more votes in all precincts whose names contain this string s and how many votes did they get in total. For example, for s='Township', the procedure will return the name (Trump or Biden) who won more votes in union of precints which have "Township" in their name as well as the sum of votes for the winner.

```
DROP PROCEDURE IF EXISTS API5;
DELIMITER $$
CREATE PROCEDURE API5(
        IN s VARCHAR(64)
)
BEGIN
        DECLARE sumBiden INT;
    DECLARE sumTrump INT;
        SELECT SUM(Biden), SUM(Trump)
        INTO sumBiden, sumTrump
        FROM (
                SELECT Timestamp, Trump, Biden
                FROM Penna
                WHERE LOCATE(s, precinct)
        ) penna
        WHERE penna.Timestamp = (SELECT MAX(Timestamp) FROM penna);

    IF sumBiden > sumTrump THEN
```

```
                SELECT "Biden" as winner, sumBiden as votes;
    ELSEIF sumTrump > sumBiden THEN
                SELECT "Trump" as winner, sumTrump as votes;
    ELSEIF sumTrump IS NULL OR sumBiden IS NULL THEN
                SIGNAL SQLSTATE '45000'
                        SET MESSAGE_TEXT = "Did not find any precincts containing
this substring.";
    END IF;
END$$
DELIMITER ;
```

Part 2 (30%)

newPenna(precinct, Timestamp, totalVotes, Trump, Biden)

This stored procedure will create a table newPenna, showing for each precinct how many votes were added to totalvotes, Trump, Biden between timestamp T and the las timestamp directly preceding T. In other words, create a table like Penna but replace totalvotes with newvotes, Trump with new_Trump and Biden with new_Biden. Stored procedure with cursor is recommended

```
DROP PROCEDURE IF EXISTS newPenna;
DELIMITER $$
CREATE PROCEDURE newPenna()
BEGIN
    DECLARE iter INT DEFAULT 0;
    DECLARE iterEnd INT;

        DECLARE iTimestamp TIMESTAMP;
    DECLARE iPrecinct VARCHAR(64);
    DECLARE iTotalVotes INT DEFAULT 0;
    DECLARE iBiden INT;
    DECLARE iTrump INT;

    DECLARE jTimestamp TIMESTAMP;
    DECLARE jPrecinct VARCHAR(64);
    DECLARE jTotalVotes INT DEFAULT 0;
    DECLARE jBiden INT;
    DECLARE jTrump INT;

    DECLARE cur CURSOR FOR (
                SELECT Timestamp, precinct, totalVotes, Biden, Trump
        FROM Penna
        ORDER BY precinct, Timestamp ASC
    );
```

```
   DROP TABLE IF EXISTS newPenna;
        CREATE TABLE newPenna (
                precinct VARCHAR(64),
        Timestamp Timestamp,
        newVotes INT,
        newTrump INT,
        newBiden INT
    );

    SELECT COUNT(*) INTO iterEnd FROM Penna;
        OPEN cur;
    FETCH cur INTO iTimestamp, iPrecinct, iTotalVotes, iBiden, iTrump;
        SET iter = iter + 1;
    WHILE iter < iterEnd DO
                FETCH cur INTO jTimestamp, jPrecinct, jTotalVotes, jBiden, jTrump;
                SET iter = iter + 1;

                IF (jPrecinct = iPrecinct AND (jTotalVotes != iTotalVotes OR jTrump
!= iTrump or jBiden != iBiden)) THEN
                        INSERT INTO newPenna
                        VALUES (iPrecinct, jTimestamp, jTotalVotes-iTotalVotes,
jTrump-iTrump, jBiden-iBiden);
                END IF;

                SELECT jTimestamp, jPrecinct, jTotalVotes, jBiden, jTrump
                INTO iTimestamp, iPrecinct, iTotalVotes, iBiden, iTrump;
    END WHILE;

    CLOSE cur;
END$$
```

Switch(precinct, timestamp, fromCandidate, toCandidate)

This stored procedure will return list of precincts, which have switched their winner from one candidate in last 24 hours of vote collection (i.e 24 hours before the last Timestamp data was collected) and that candidate was the ultimate winner of this precinct.

```
DROP PROCEDURE IF EXISTS Switch;
DELIMITER $$
CREATE PROCEDURE Switch()
BEGIN
        DECLARE maxTimestamp TIMESTAMP DEFAULT (SELECT MAX(Timestamp) FROM Penna);
    DECLARE minTimestamp TIMESTAMP DEFAULT DATE_ADD(maxTimestamp, INTERVAL -1 DAY);
```

```
DECLARE iter INT DEFAULT 0;
DECLARE iterEnd INT;

DECLARE precinctFlag VARCHAR(64) DEFAULT 0;

    DECLARE iTimestamp TIMESTAMP;
DECLARE iPrecinct VARCHAR(64);
DECLARE iBiden INT;
DECLARE iTrump INT;

DECLARE jTimestamp TIMESTAMP;
DECLARE jPrecinct VARCHAR(64);
DECLARE jBiden INT;
DECLARE jTrump INT;

DECLARE cur CURSOR FOR (
            SELECT Timestamp, precinct, Biden, Trump
    FROM Penna
    -- WHERE Timestamp >= minTimestamp
    ORDER BY precinct, Timestamp DESC
);

DROP TEMPORARY TABLE IF EXISTS switchResults;
CREATE TEMPORARY TABLE switchResults(
            precinct VARCHAR(64),
    Timestamp TIMESTAMP,
    fromCandidate VARCHAR(6),
    toCandidate VARCHAR(6)
);

SELECT COUNT(*) INTO iterEnd FROM Penna;
    OPEN cur;
outerWhile: WHILE iter < iterEnd-1 DO
            FETCH cur INTO iTimestamp, iPrecinct, iBiden, iTrump;
    SET iter = iter + 1;
    innerWhile: WHILE 1 DO
                    FETCH cur INTO jTimestamp, jPrecinct, jBiden, jTrump;
        SET iter = iter + 1;

        IF (jPrecinct != iPrecinct) THEN
                            LEAVE innerWhile;
                    END IF;

        IF precinctFlag = iPrecinct THEN
                            ITERATE innerWhile;
                    END IF;
```

```
            IF ( iBiden > iTrump AND jBiden < jTrump) THEN
                INSERT INTO switchResults
                VALUES (iPrecinct, iTimestamp, "Trump", "Biden");
                SET precinctFlag = iPrecinct;
            ELSEIF (iBiden < iTrump AND jBiden > jTrump) THEN
                                INSERT INTO switchResults
                VALUES (iPrecinct, iTimestamp, "Biden", "Trump");
                SET precinctFlag = iPrecinct;
            END IF;

            SELECT jTimestamp, jPrecinct, jBiden, jTrump
            INTO iTimestamp, iPrecinct, iBiden, iTrump;

            IF (iter >= iterEnd) THEN
                                LEAVE outerWhile;
            END IF;
        END WHILE;
    END WHILE;

    CLOSE cur;
    SELECT precinct, Timestamp, fromCandidate, toCandidate FROM switchResults;
    DROP TEMPORARY TABLE switchResults;
END$$
```

Part 3 (10%)

Write SQL queries or stored procedures to check if the following patterns are enforced in the database:

1. The sum of votes for Trump and Biden cannot be larger than totalvotes

```
DROP PROCEDURE IF EXISTS part3_1;
DELIMITER $$
CREATE PROCEDURE part3_1()
BEGIN
        SELECT IF (
                EXISTS ( SELECT Timestamp FROM Penna WHERE totalVotes < Biden +
Trump LIMIT 1),
        FALSE,
        TRUE
        ) as isValid;
END$$
DELIMITER ;
```

2. There cannot be any tuples with timestamps later than Nov 11 and earlier than Nov3

```
DROP PROCEDURE IF EXISTS part3_2;
DELIMITER $$
CREATE PROCEDURE part3_2()
BEGIN
        SELECT IF (
                EXISTS ( SELECT Timestamp FROM Penna WHERE Timestamp < '2020-11-03'
OR Timestamp >= '2020-11-12'),
        FALSE,
        TRUE
        ) as isValid;
END$$
DELIMITER ;
```

3. Totalvotes for any precinct and at any timestamp T > 2020-11-05 00:00:00, will be larger or equal to totalvotes at T'<T where T'>2020-11-05 00:00:00 for that precinct.

```
DROP PROCEDURE IF EXISTS part3_3;
DELIMITER $$
CREATE PROCEDURE part3_3()
BEGIN
        DECLARE iter INT DEFAULT 0;
    DECLARE iterEnd INT;

    DECLARE iTimestamp TIMESTAMP;
    DECLARE iPrecinct VARCHAR(64);
    DECLARE iTotalVotes INT;

    DECLARE jTimestamp TIMESTAMP;
    DECLARE jPrecinct VARCHAR(64);
    DECLARE jTotalVotes INT;

    DECLARE isValid BOOLEAN DEFAULT TRUE;

    DECLARE cur CURSOR FOR (
                SELECT Timestamp, precinct, totalVotes
        FROM Penna
        WHERE Timestamp LIKE '2020-11-05 %'
        ORDER BY precinct, Timestamp DESC
    );

    SELECT COUNT(*) INTO iterEnd FROM Penna;
    OPEN cur;
    FETCH cur INTO iTimestamp, iPrecinct, iTotalVotes;
```

```
    whileLoop: WHILE iter < iterEnd DO
                SET iter = iter + 1;
        FETCH cur INTO jTimestamp, jPrecinct, jTotalVotes;
        IF (jPrecinct = iPrecinct) THEN
                        IF (iTotalVotes < jTotalVotes) THEN
                                SET isValid = FALSE;
                                LEAVE whileLoop;
                        END IF;
        END IF;
        SELECT jTimestamp, jPrecinct, jTotalVotes INTO iTimestamp, iPrecinct,
iTotalVotes;
    END WHILE;
    SELECT isValid;
END$$
DELIMITER ;
```

You should write SQL queries to verify the constraints and return TRUE or FALSE (in case constraint is not satisfied).Queries that don't return a boolean value won't be accepted.

Part 4 (30%)

Triggers and UPDATE driven Stored Procedures

Create three tables Updated Tuples, Inserted Tuples and Deleted Tuples. All three tables should have the same schema as Penna and should store any tuples which were updated (store them as they were before the update), any tuples which were inserted, and any tuples which were deleted in their corresponding tables. The triggers should populate these tables upon each update/insertion/deletion. There will be one trigger for the update operation, one trigger for the insert operation and one trigger for the delete operation.

Triggers

```
DROP TRIGGER IF EXISTS onUpdate;
DELIMITER $$
CREATE TRIGGER onUpdate AFTER UPDATE
ON Penna FOR EACH ROW
BEGIN
        INSERT INTO updatedPenna
    VALUES (OLD.ID, OLD.Timestamp, OLD.state, OLD.locality, OLD.precinct, OLD.geo,
OLD.totalvotes, OLD.Biden, OLD.Trump, OLD.filestamp);
END$$
DELIMITER ;
```

```
DROP TRIGGER IF EXISTS onInsert;
DELIMITER $$
CREATE TRIGGER onInsert AFTER INSERT
ON Penna FOR EACH ROW
BEGIN
        INSERT INTO insertedPenna
        VALUES (NEW.ID, NEW.Timestamp, NEW.state, NEW.locality, NEW.precinct,
NEW.geo, NEW.totalvotes, NEW.Biden, NEW.Trump, NEW.filestamp);
END$$
DELIMITER ;
```

```
DROP TRIGGER IF EXISTS onDelete;
DELIMITER $$
CREATE TRIGGER onDelete BEFORE DELETE
ON Penna FOR EACH ROW
BEGIN
        INSERT INTO deletedPenna
        VALUES (OLD.ID, OLD.Timestamp, OLD.state, OLD.locality, OLD.precinct,
OLD.geo, OLD.totalvotes, OLD.Biden, OLD.Trump, OLD.filestamp);
END$$
DELIMITER ;
```

Tester

```
DROP PROCEDURE IF EXISTS part4reset;
DELIMITER $$
CREATE PROCEDURE part4reset()
BEGIN
        CREATE TABLE IF NOT EXISTS pennaTriggers LIKE Penna;
    CREATE TABLE IF NOT EXISTS insertedPenna LIKE Penna;
    CREATE TABLE IF NOT EXISTS deletedPenna LIKE Penna;
    CREATE TABLE IF NOT EXISTS updatedPenna LIKE Penna;

        SET SQL_SAFE_UPDATES = 0;
    DELETE FROM insertedPenna;
    DELETE FROM deletedPenna;
    DELETE FROM updatedPenna;
    DELETE FROM pennaTriggers;

    INSERT INTO pennaTriggers SELECT * FROM Penna ORDER BY ID ASC LIMIT 10;
    UPDATE pennaTriggers SET ID = ID + 1 WHERE ID < 5;
    DELETE FROM pennaTriggers WHERE ID > 5;
END$$
```

Stored Procedure Simulating Trigger

MoveVotes(Precinct, Timestamp, Candidate, Number_of_Moved_Votes)

1. Precinct – one of the existing precinct
2. Timestamp must be existing timestamp. If Timestamp does not appear in Penna than MoveVotes should display a message “Unknown Timestamp”.
3. The Number_of_Moved_Votes parameter (always positive integer) shows the number of votes to be moved from the Candidate to another candidate and it cannot be larger than number of votes that the Candidate has at the Timestamp. If this is the case MoveVotes () should display a message “Not enough votes”.
4. Of course if CoreCandidate is neither Trump nor Biden, MoveVotes() should say “Wrong Candidate”

```
DROP PROCEDURE IF EXISTS MoveVotes;
DELIMITER $$
CREATE PROCEDURE MoveVotes(
        IN p VARCHAR(64),
    IN ts TIMESTAMP,
    IN numVotes INT,
    IN c VARCHAR(6)
)
BEGIN
        DECLARE numBiden INT;
    DECLARE numTrump INT;

        IF (p NOT IN (SELECT DISTINCT precinct FROM Penna)) THEN
                SIGNAL SQLSTATE '45000'
                        SET message_text = "Invalid precinct name.";
    END IF;

    IF (ts NOT IN (SELECT DISTINCT Timestamp FROM Penna WHERE precinct=p)) THEN
                SIGNAL SQLSTATE '45000'
                        SET message_text = "Invalid timestamp for given precinct.";
    END IF;

    IF (c NOT IN ("Trump", "Biden")) THEN
                SIGNAL SQLSTATE '45000'
                        SET message_text = "Invalid candidate name.";
    END IF;

    SELECT Biden, Trump INTO numBiden, numTrump FROM Penna WHERE Timestamp = ts AND
precinct = p;
```

```
    IF ( (c = "Trump" and numVotes > numTrump) OR (c = "Biden" and numVotes >
numBiden) ) THEN
                SIGNAL SQLSTATE '45000'
                        SET message_text = "Number of votes to be moved exceeds
number of votes for candidate.";
    END IF;



    IF (c = "Trump") THEN
                UPDATE Penna
        SET Trump = Trump - numVotes, Biden = Biden + numVotes
        WHERE precinct = p AND Timestamp >= ts;
    ELSEIF (c = "Biden") THEN
                UPDATE Penna
        SET Biden = Biden - numVotes, Trump = Trump + numVotes
        WHERE precinct = p AND Timestamp >= ts;
    END IF;
END$$
DELIMITER ;
```

After you are done with exceptions, you should move the Number_of_Moved_Votes from CoreCandidate to another candidate (there are only two) and do it not just for this Timestamp (the first parameter) but also for all T>Timestamp, that is all future timestamps in the given precinct.

For example MoveVotes(Red Hill, 2020-11-06 15:38:36,'Trump',100) will remove 100 votes from Trump and move it to Biden at 2020-11-06 15:38:36 and all future timestamps after that in the Red Hill precinct.